

# Blackjack

Alberto Di Girolamo

`alberto.digirolamo2@studio.unibo.it`

February 2023

Il progetto consiste nello sviluppo di una versione semplificata del gioco d'azzardo Blackjack in versione online.

Ogni tavolo di gioco è composto da almeno un giocatore fino ad un massimo di tre.

Ogni partita inizia con la scelta della scommessa da puntare da parte degli utenti. Successivamente vengono assegnate due carte ad ogni giocatore e viene mostrata la prima carta del croupier.

Ogni giocatore effettua il suo gioco, ha la possibilità di chiedere altre carte oppure di stare.

Quando tutti i giocatori hanno completato il loro gioco, vengono estratte e mostrate agli utenti le carte del banco.

Quando il valore delle carte del banco è di almeno 17 punti, il banco termina l'estrazione delle carte e stabilisce i risultati della partita.

Se il valore delle carte di un utente è superiore a 21, oppure se il valore delle carte del **Croupier** è superiore a quella dell'utente, viene sottratta la scommessa.

Se il valore delle carte del banco è superiore a 21 oppure è minore al valore delle carte dell'utente viene vinta la scommessa.

Se l'utente riceve il Blackjack vince la scommessa e ha diritto al pagamento di 3 a 2 della scommessa.

Se invece è il **Croupier** a effettuare il Blackjack tutti gli utenti in gioco perdono la scommessa.

## 1 Goal/Requirements

Di seguito vengono riportate delle Q/A per esplicitare più dettagliatamente gli aspetti del progetto con le relative implementazioni.

### 1.1 Questions/Answers

- *Che caratteristiche deve avere il nickname scelto dall'utente in fase di registrazione?*

L'unica condizione che deve rispettare l'utente è quella di scegliere un nickname univoco, non può utilizzare quindi un nome che attualmente è utilizzato da un altro utente. Solo quando quest'ultimo lascerà il gioco, il nickname sarà riutilizzabile nuovamente.

- *Come vengono gestite le Lobby di gioco?*

Una Lobby corrisponde ad un tavolo di gioco ed è composta da un Croupier e un numero massimo di 3 giocatori.

Quando un nuovo utente si registra, gli viene assegnata automaticamente una lobby.

Se nessuna Lobby è già presente, ne viene creata una e l'utente viene aggiunto.

Se invece sono presenti già alcune lobby, viene fatto un controllo del numero di giocatori presenti. Nel caso in cui si identifica una Lobby con un numero di giocatori minore di 3, l'utente viene aggiunto. Se invece, tutti i tavoli siano pieni, allora verrà aggiunta una nuova lobby.

- *Come viene determinato il saldo iniziale?*

Il saldo iniziale viene assegnato dal sistema ed ammonta a \$30.

- *Quali sono le possibili mosse che un utente può fare durante una partita?*

Dopo aver ottenuto le prime due carte iniziali, l'utente può decidere se chiedere altre carte, sperando di non superare 21 come somma del valore della propria mano, oppure di restare e aspettare che anche gli altri utenti facciano il loro gioco.

- *Come funziona l'aggiornamento del proprio saldo alla fine di una mano?*

Quando una mano di gioco termina, il sistema aggiorna automaticamente il saldo del giocatore in base al risultato che ha ottenuto.

Nel caso in cui il giocatore decida di effettuare una nuova partita, avrà a disposizione il saldo aggiornato.

## 1.2 Scenarios

All'avvio dell'applicazione l'utente deve inserire un nickname che verrà utilizzato dal sistema per identificare il giocatore.

La prossima scelta che l'utente deve effettuare è quella di stabilire la scommessa da puntare. Il valore da inserire deve essere minore o uguale al proprio saldo.

Quando un utente riceve le carte iniziali può chiedere al sistema di estrarre ulteriori nuove carte. Quando la somma dei punti delle carte ricevute soddisfano l'utente, si comunica al sistema la volontà di restare ed attendendo così gli eventuali altri giocatori.

Al termine di una partita verrà chiesto all'utente se vuole continuare a giocare oppure preferisce terminare la partita.

Nel caso in cui decidesse di continuare si avvierebbe una nuova partita, altrimenti verrebbe chiusa l'applicazione.

### 1.3 Self-assessment policy

Per il controllo della qualità e per stabilire l'efficacia del progetto, sono stati prodotti dei test automatici JUnit.

I test sono stati realizzati per controllare più aspetti del sistema, partendo da test che hanno lo scopo di controllare che il sistema rimanga consistente (evitando ad esempio l'inserimento di più utenti con lo stesso nickname), fino alla realizzazione di test che simulano una partita intera, controllando ad esempio la logica che stabilisce il risultato di ogni partita.

## 2 Requirements Analysis

Di seguito vengono riportati dei diagrammi di caso d'uso per individuare chi utilizza il sistema (attore/utente) e cosa viene fatto, mostrando il comportamento del sistema su risultati osservabili.

### 2.1 Initial logical

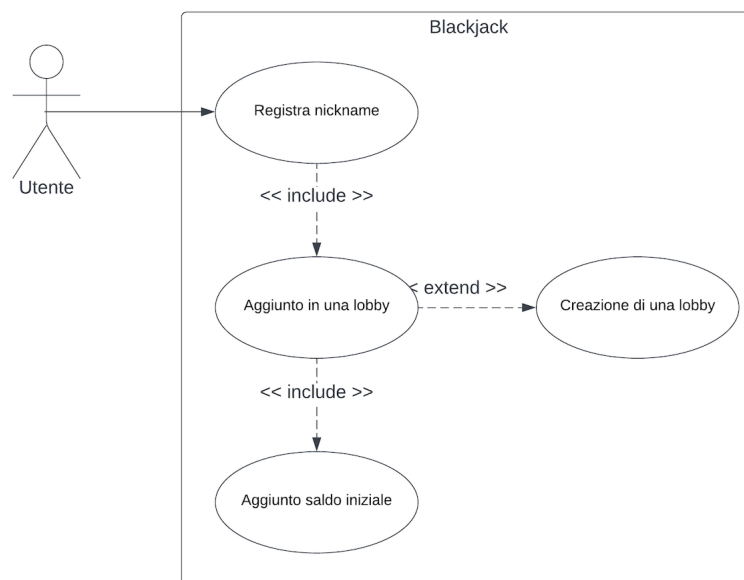


Figure 1: Initial logical Use Case

All'avvio dell'applicazione l'utente deve solamente inserire il proprio nickname, successivamente sarà gestita interamente dal sistema la scelta di una Lobby esistente nella quale inserire l'utente o eventualmente la creazione di una nuova.

Successivamente il sistema assegna un saldo iniziale, il quale può aumentare o diminuire solo tramite le scommesse effettuate durante il gioco.

Terminato il saldo non può essere rigenerato. L'unico modo per continuare a giocare è ricominciare una nuova partita.

## 2.2 Game's initial logical

- Seleziona scommessa
  - Assegnate prime due carte
- Richiede nuova carta (hit)
- Confermare le proprie carte
- Effettuare una nuova partita
- Terminare gioco

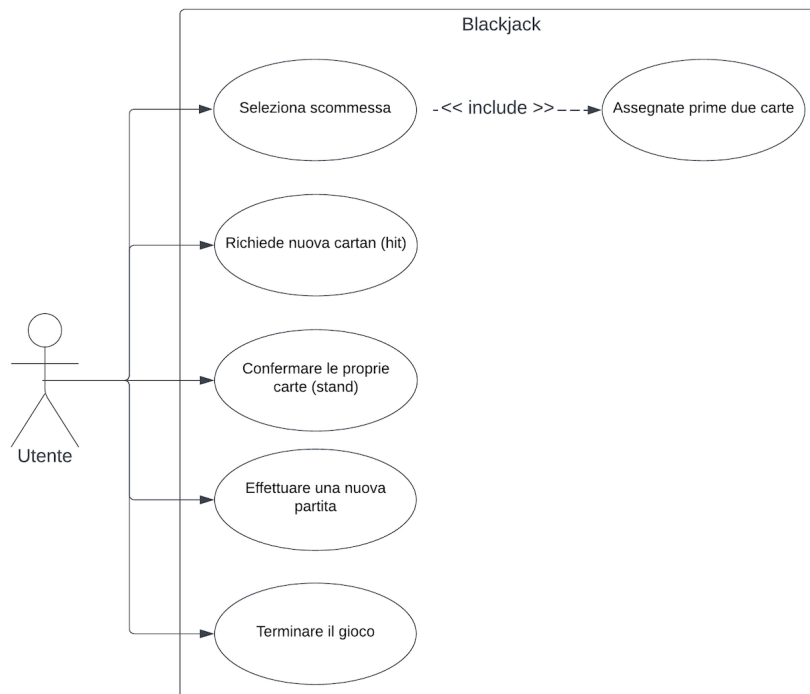


Figure 2: Game logical Use Case

L'utente all'avvio di ogni mano deve decidere una scommessa e successivamente il sistema gli assegna le prime due carte.

L'attore interagendo con il sistema sceglie se richiedere nuove carte oppure di stare confermandole. Al termine può decidere se effettuare una nuova partita o di terminare il gioco.

Sulla base della struttura del sistema è stato deciso di implementarlo tramite architettura **client-server**.

Il **Server** sarà implementato come un Web Service che utilizza come protocollo di comunicazione **HTTP (POST, GET, PUT, DELETE)** per comunicare.

### 3 Design

Prendendo in considerazione l'analisi dei requisiti e i casi d'uso possibili, viene successivamente riportato il design scelto per la realizzazione del sistema che implementa il gioco Blackjack.

La struttura del design scelto vuole mantenere un elevato livello di astrazione del sistema, cercando di realizzare un'architettura che non abbia forti dipendenze con l'implementazione realizzata, ma bensì di renderlo il più possibile indipendente.

#### 3.1 Structure

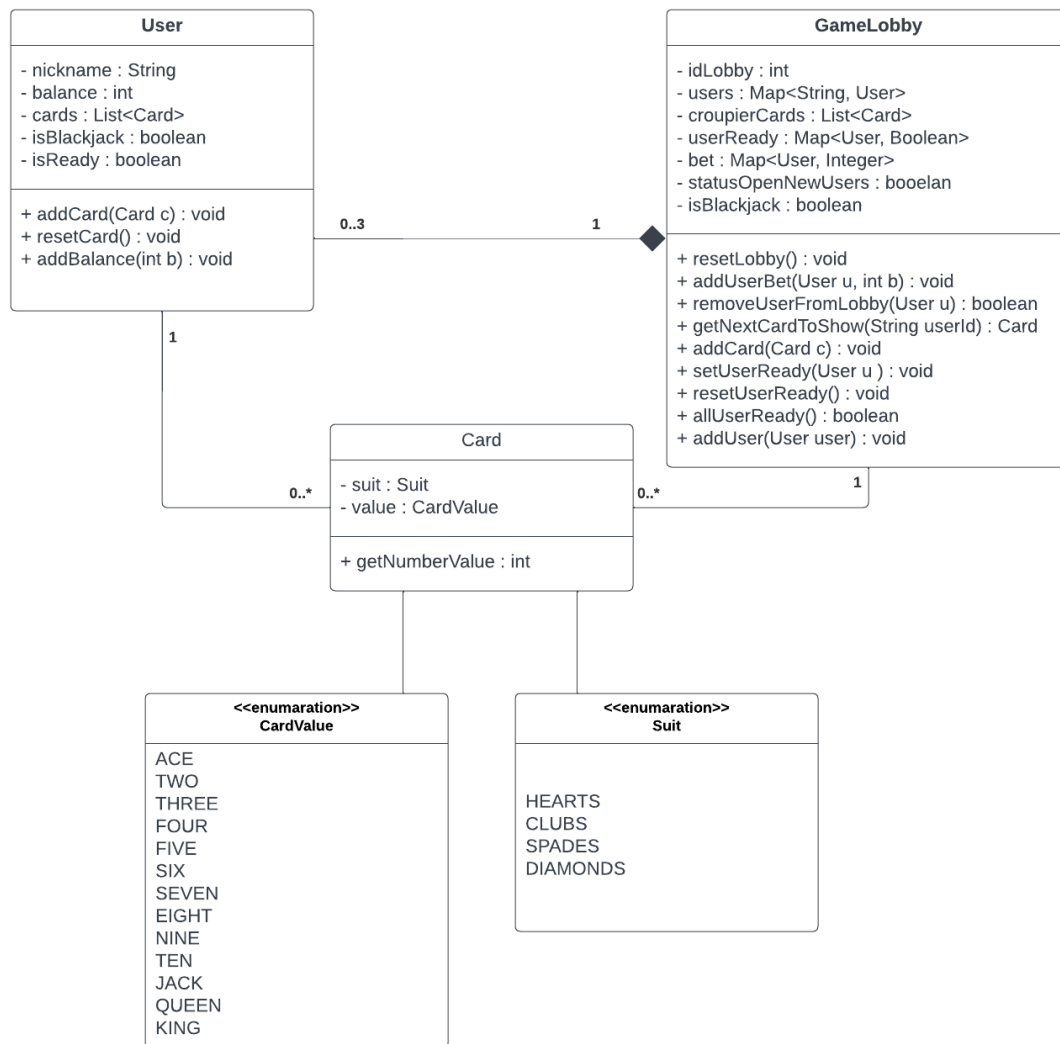


Figure 3: Class Diagram

La struttura principale dei componenti del sistema è stata rappresentata tramite diagramma delle classi UML.

L'entità **GameLobby** rappresenta un tavolo di gioco formato da un **Croupier** (gestito interamente dal sistema) e da un massimo di tre **User** (giocatori). La **Lobby** contiene un id come identificatore e memorizza tutte le informazioni importanti per la partita, come le carte assegnate al banco e le scommesse effettuate da ogni utente all'inizio di ogni partita. Possiede tutti i metodi necessari per gestire la sincronizzazione degli utenti, memorizzando lo stato di ognuno di questi per determinare se hanno effettuato il loro gioco e sono pronti a procedere. Possiede inoltre delle informazioni utili alla gestione del sistema, come per esempio un campo che determina se la partita è cominciata negando quindi la possibilità ad altri giocatori di unirsi alla lobby. Se invece il gioco è terminato dà l'opportunità ad altri giocatori di entrare finché non viene avviata la prossima partita. Altri metodi posseduti dall'entità **GameLobby** permettono di memorizzare l'eventuale Blackjack da parte del croupier, la rimozione di un utente dalla **Lobby** e la reinizializzazione della partita una volta che questa viene terminata.

L'entità **User** rappresenta un singolo utente che viene identificato da un nickname. Durante la fase di registrazione iniziale viene assegnato un saldo che sarà aggiornato ad ogni fine partita in base all'esito della propria mano di gioco. Se l'utente riceve il Blackjack l'informazione viene memorizzata, verrà poi gestita successivamente dal sistema una volta che viene stabilito il risultato della mano al termine della partita. Ogni **User** mantiene al suo interno un campo che stabilisce il suo stato di attività: quando il giocatore ha finito di effettuare le proprie scelte di gioco viene settato lo stato di Ready, sarà poi il sistema a gestire e modificare questa informazione. L'utente mantiene memorizzate le proprie carte possedute durante una singola partita. Le carte sono entità formate da due campi: **SUIT** e **CARDVALUE**. Entrambi sono realizzati tramite enumerazione e l'unione di queste due informazioni, generate randomicamente dal sistema, identificano una precisa carta.

### 3.2 Behaviour

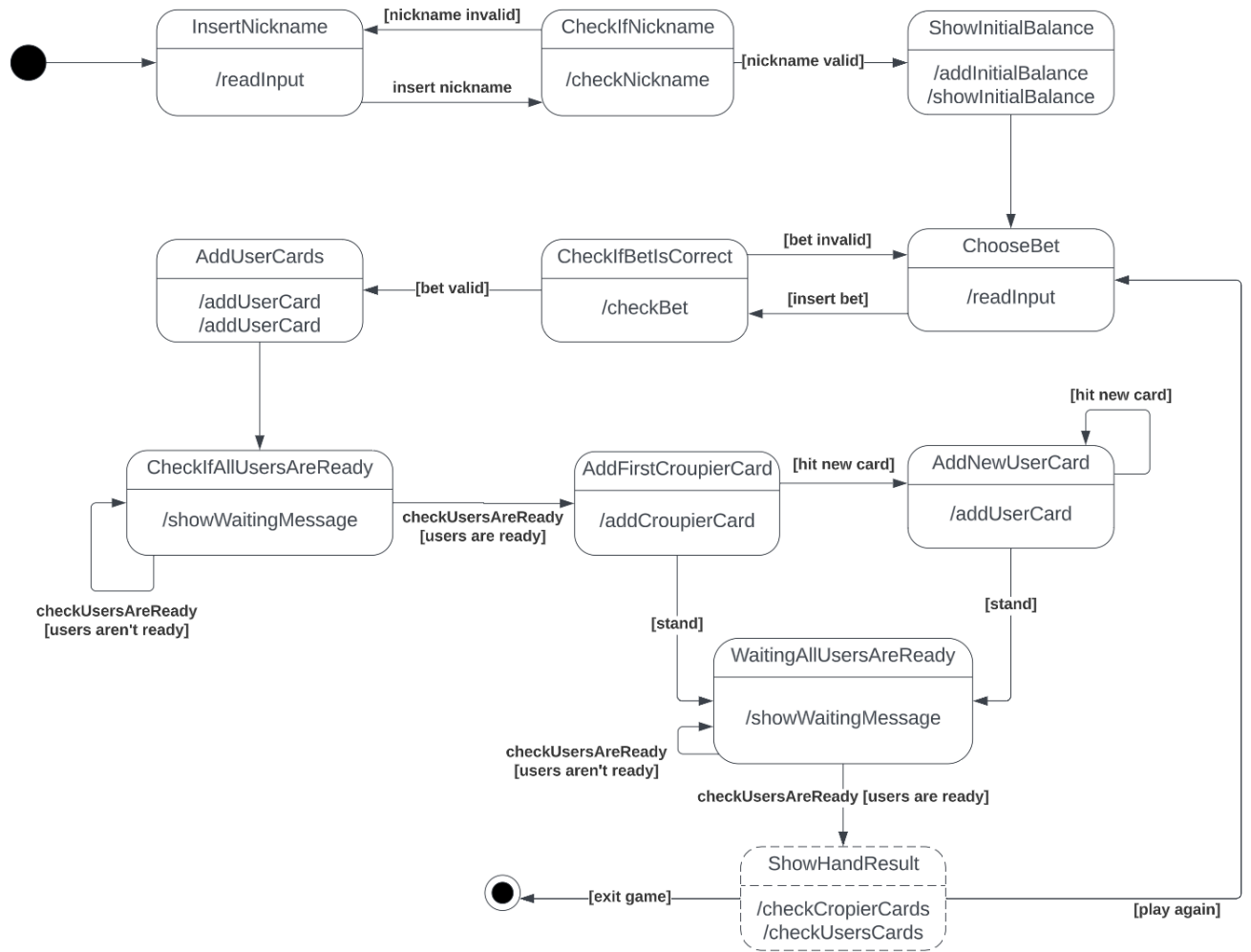


Figure 4: System logic State Diagram

La figura riportata sopra mostra un diagramma degli stati che esplicita il comportamento della logica generale del sistema.

Successivamente alla connessione di un nuovo **user Client** al **Server**, il sistema invita l'utente ad inserire un **nickname**. Se a seguito del controllo del **nickname** inserito (**checkNickname**) si ottiene un riscontro negativo, si ritorna allo stato precedente. Altrimenti il sistema assegna e mostra il saldo iniziale e successivamente rimane in attesa della scelta della scommessa di gioco. Anche in questo secondo blocco di stati è presente un controllo dell'input (**checkBet**) che in base al risultato (**bet invalid** / **bet valid**) determina se proseguire o richiedere l'inserimento di una scommessa. Nel caso in cui la scommessa inserita sia corretta, il sistema assegna automaticamente



due carte (`addUserCard`) e entre in un loop dove ci rimane finchè tutti gli utenti non sono nel medesimo stato (`checkUsersAreReady`). Quando tutti gli `User` presenti nella Lobby sono `Ready` viene estratta la prima carta del `Croupier` (`addCroupierCard`) e ogni `User` ha la possibilità di richiedere altre carte (`hit new card`) oppure di entrare in attesa (`stand`) nello stato `WaitingAllUsersAreReady` mostrando un messaggio di informazione (`showWaitingMessage`). Quando tutti gli utenti sono nel medesimo stato di attesa avviene il controllo del risultato della mano di gioco (riportata successivamente) e viene data la possibilità all'utente di effettuare una nuova partita, riportando così lo stato del sistema all'inserimento di una nuova scommessa, oppure di terminare il gioco ed uscire.

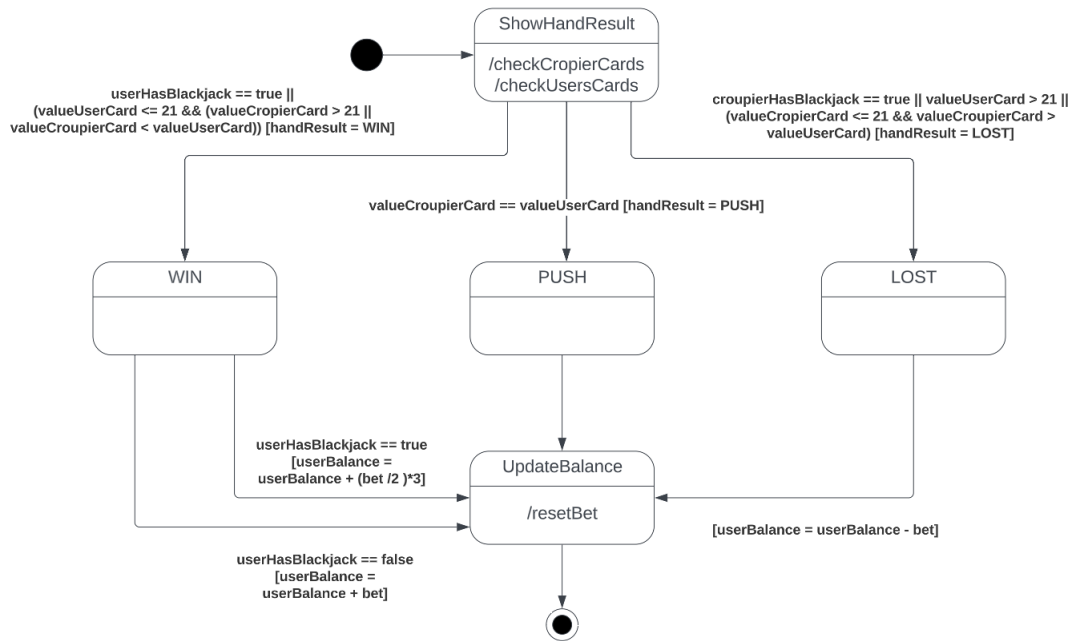


Figure 5: Hand Result State Diagram

In figura 5 viene mostrato il diagramma degli stati che illustra il comportamento del sistema quando avviene il controllo del risultato della mano di gioco per ogni utente. Viene effettuato un controllo delle carte dello `User` e del `Croupier` (`checkUserCards` e `checkCroupierCards`) e in base a delle specifiche condizioni si passa allo stato successivo che determina l'effettivo risultato.

- Se lo `User` ha effettuato il Blackjack oppure se il valore delle sue carte è minore o uguale a 21 e il valore delle carte del `croupier` è maggiore di 21 oppure è minore di quello dello `User` allora si entra nello stato `WIN`.
- Se il `Croupier` ha effettuato il Blackjack oppure se il valore delle carte dello `User` è maggiore di 21 oppure il valore delle carte del `Croupier` è minore o uguale di 21 e maggiore di quelle dell'utente, allora si passa nello stato di `LOST`.

- Se invece il valore delle carte dell'utente è uguale al valore di quelle del **Croupier** si entra nello stato **PUSH**.

A seguire si effettua l'update del valore del **Balance** nello stato **UpdateBalance** in base al risultato ottenuto:

- Se si è entrati in **WIN** e l'utente ha effettuato il Blackjack allora la scommessa viene pagata 3:2.
- Se si è entrati in **WIN** e non si ha effettuato il Blackjack si aggiorna il **Balance** con il valore della scommessa puntata.
- Se si è entrati nello stato di **LOST** si sottrae al balance la scommessa puntata.
- Se invece si è entrati nello stato di **PUSH** si mantiene inalterato il saldo del balance.

A seguire si resettano le puntate (**resetBet**).

### 3.3 Interaction

A seguire vengono riportati dei diagrammi delle sequenze per analizzare la struttura delle interazioni del sistema.

Ogni risorsa presente nel sistema possiede un **Controllore** un insieme di **Api** che si occupano della gestione delle chiamate e delle risposte per i **Client**.

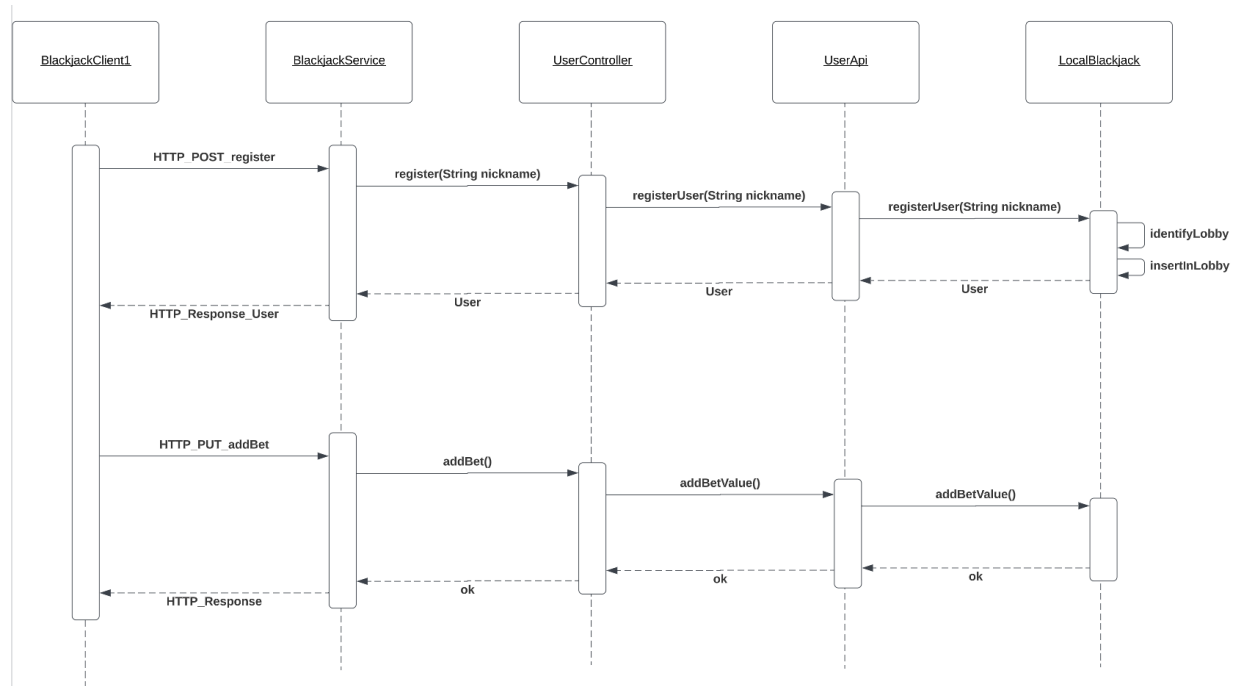


Figure 6: User Sequence Diagram

L'immagine precedentemente riportata mostra la sequenza di chiamate effettuate da un **Client/User** durante la fase di registrazione iniziale. Quando un **Client** effettua una richiesta di registrazione **POST** (`HTTP_POST_register`) al **Server**, essa viene indirizzata allo **UserController**, il quale richiama la corrispondente **API** all'interno di **UserApi** che a sua volta agisce sull'istanza locale del gioco Blackjack presente nel **Server** (**LocalBlackjack**) richiamando dei metodi interni per identificare e inserire l'utente in una lobby.

A seguire, seguendo la stessa catena di interazioni, il **Client** inserisce una scommessa per iniziare la partita.

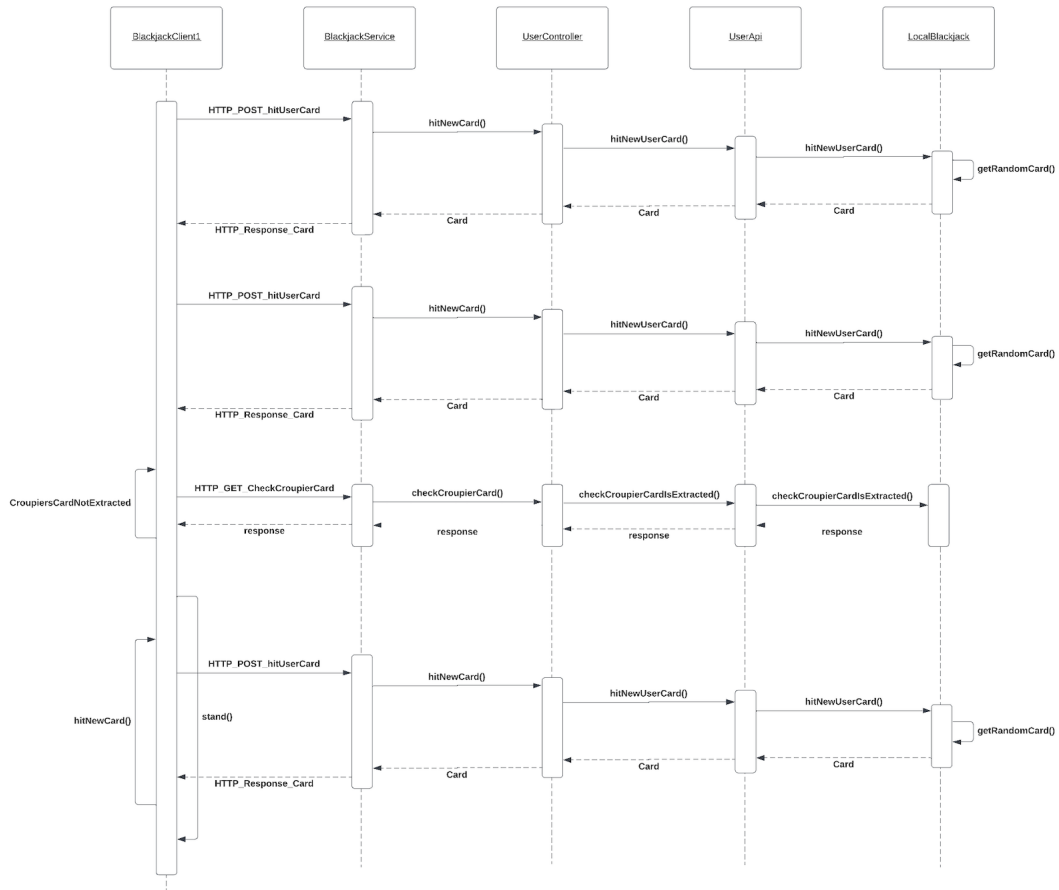


Figure 7: User Game Logic Sequence Diagram

L'immagine precedentemente riportata mostra le chiamate effettuate al **Client** verso il sistema durante l'inizio della partita di gioco.

Lo **User** effettua due chiamate **HTTP POST** per ricevere come risposta le prime due carte e successivamente tramite richieste **GET** in polling controlla che sia stata estratta la prima carta del **Croupier** per proseguire.

Successivamente nel caso in cui lo **User** volesse altre carte verranno effettuate altre chiamate **HTTP POST** finchè il **Client** non deciderà di stare.

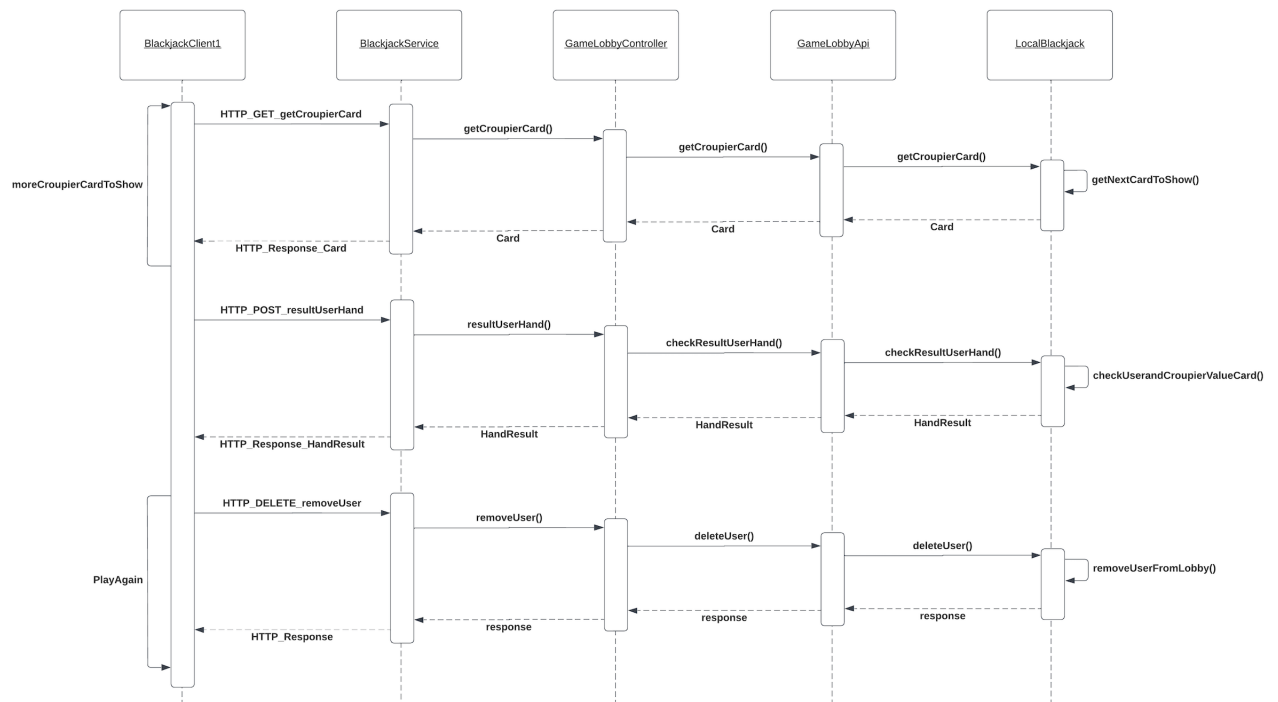


Figure 8: GameLobby Sequence Diagram

La figura 8 mostra le chiamate effettuate dal **Client** verso il **Gamelobby** durante la fase finale di gioco.

Attraverso chiamate **HTTP GET** con tecnica polling si richiedono le carte del Croupier estratte, saranno poi visualizzate a schermo.

Successivamente viene richiesto tramite **HTTP POST** il risultato della propria mano di gioco. L'istanza locale del Blackjack effettua delle chiamate di metodi interni di controllo (**checkUserandCroupierValueCard**) per poi inviare come risposta al **Client** il risultato della partita.

Infine, se l'utente decide di terminare il gioco viene effettuata una chiamata **HTTP DELETE** con lo scopo di far eliminare dal **Server** il **Client**, liberando spazio sulla lobby.

## 4 Implementation Details

In questa sezione verranno mostrate le strategie di implementazione adottate.

Il progetto è stato sviluppato in **Java** ed il sistema è composto principalmente da due parti: **BlackjackClient** e **BlackJackService**.

### 4.1 BlackjackClient

Il **Client** del sistema contiene tutti i metodi e le istanze che sono necessari per effettuare le richieste HTTP al **Server**.

Le chiamate sono state attuate seguendo i seguenti URL:

- */users*: tutti i percorsi che hanno come radice *users* gestiscono le richieste effettuate per il controllo degli utenti e per dirigere il loro comportamento verso il sistema. Alcune delle richieste gestite si occupano dell'aggiunta di un nuovo utente, dell'inserimento delle scommesse e di richiedere le proprie carte ecc.
- */lobby*: è il riferimento di tutte le richieste effettuate per la gestione della **Lobby** nonché il tavolo di gioco. Le richieste fanno riferimento al controllo dei risultati di tutti gli utenti presenti nella **Lobby**, nella gestione per quanto riguarda la reinizializzazione della partita e nella rimozione degli utenti ecc.

### 4.2 BlackjackService

Il **Server** è stato realizzato utilizzando il framework **Javalin** [2]. Strutturato come un **Web Service**, aderisce allo stile architettonico **ReST** fornendo **Api** per permettere di far interagire i **Client** con il **Server**.

**BlackjackService** possiede un **Controller** per ogni risorsa presente nel sistema: **User** e **GameLobby**, hanno lo scopo di indirizzare le richieste dei **Client** in modo tale da soddisfarle. Le **Api** permettono di interagire con uno storage locale del sistema.

users	
POST	/blackjack/v0.1.0/users
PUT	/blackjack/v0.1.0/users/balance
POST	/blackjack/v0.1.0/users/newcard
POST	/blackjack/v0.1.0/users/userready
GET	/blackjack/v0.1.0/users/{userId}
POST	/blackjack/v0.1.0/users/{userId}
GET	/blackjack/v0.1.0/users/getbalance/{userId}
POST	/blackjack/v0.1.0/users/setuserready
GET	/blackjack/v0.1.0/users/userblackjack/{userId}
GET	/blackjack/v0.1.0/lobby/{userId}

Figure 9: Swagger User Doc

lobby	
POST	/blackjack/v0.1.0/lobby/over21
DELETE	/blackjack/v0.1.0/lobby/{userId}
POST	/blackjack/v0.1.0/lobby/resulthand
POST	/blackjack/v0.1.0/lobby/resetplay

Figure 10: Swagger GameLobby Doc

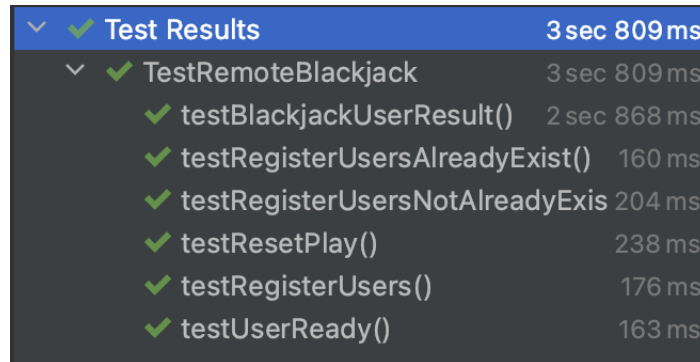
Tramite **Swagger** [4] sono state documentate le rotte registrate nel **Server**. Avviato il **Server**, le rotte sono consultabili seguendo il percorso `/doc/ui`.

Per la serializzazione e la deserializzazione degli oggetti durante la comunicazione è stata utilizzata la libreria **Gson** [1].

## 5 Self-assessment / Validation

Per testare il sistema è stato scelto di implementare tramite il framework **JUnit** [3] dei test automatici oltre al testing manuale fatto durante la realizzazione del sistema.

Tutti i test automatici sono presenti all'interno di un modulo chiamato **bj-test**.



✓	Test Results	3 sec 809 ms
✓	TestRemoteBlackjack	3 sec 809 ms
✓	testBlackjackUserResult()	2 sec 868 ms
✓	testRegisterUsersAlreadyExist()	160 ms
✓	testRegisterUsersNotAlreadyExist()	204 ms
✓	testResetPlay()	238 ms
✓	testRegisterUsers()	176 ms
✓	testUserReady()	163 ms

Figure 11: JUnit test

Tramite l'ausilio di JUnit è stato possibile testare il sistema automaticamente.

La classe **AbstractTestBlackjack** contiene tutti i test realizzati. Nello specifico sono stati implementati all'inizio dei test più selettivi, in modo tale da riuscire a esaminare specifiche funzioni del sistema: come la registrazione di un utente e la sua eliminazione dal **Server**, la reinizializzazione di una nuova partita e il controllo dello stato di **Ready** degli **User**.

Successivamente è stato implementato anche un test che simula una vera partita di gioco che termina con il blackjack da parte dello **User**, controllando il bilancio iniziale e finale del balance e la corretta gestione dei punti delle carte.

## 6 Deployment Instructions

Per il deploy dell'applicazione è necessario utilizzare una macchina che possieda la JVM essendo realizzato in Java.

A seguire vengono riportati i passi necessari per avviare il sistema tramite terminale utilizzando gradle.

1. Inizialmente è necessario avviare il **Server**. Aprire un terminale e eseguire il seguente comando: `./gradlew --console plain bj-server:run`
2. Quando il **Server** è avviato è possibile eseguire il comando `./gradlew --console plain bj-client:run` per avviare un **Client**. Nel caso in cui si volessero aggiungere più **Client** basterebbe eseguirlo nuovamente su un nuovo terminale.

È stato deciso di utilizzare l'opzione `--console plain` per non visualizzare i messaggi di gradle.



## 7 Usage Examples

Una volta avviato il **Server** è possibile avviare un **Client** per ogni giocatore presente.

Un messaggio di benvenuto viene mostrato in output al **Client** invitandolo ad inserire un nickname.

Nel caso in cui venisse scritto un nickname già presente nel sistema, verrebbe visualizzato un messaggio di errore che inviterebbe l'utente ad inserirne un'altro.



Figure 12: Game start

Avvenuta la registrazione, viene visualizzato il messaggio relativo al valore iniziale del saldo, seguito da un messaggio che invita l'utente a selezionare una scommessa per cominciare la partita.

Il valore da inserire deve essere un numero minore o uguale al valore del saldo, altrimenti verrebbe visualizzato un messaggio di errore.

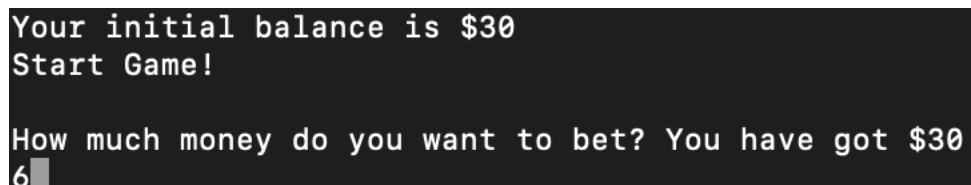
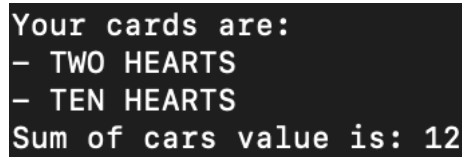


Figure 13: Choose bet

Selezionata la scommessa, vengono estratte le prime due carte del **Client** e visualizzata la somma dei punti per facilitare la comprensione della partita.



```
Your cards are:  
- TWO HEARTS  
- TEN HEARTS  
Sum of cars value is: 12
```

Figure 14: User's card

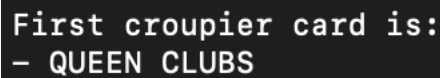
Il sistema avvisa l'utente che prima di procedere con il gioco deve attendere che tutti gli altri **Users** presenti nella **Lobby** siano anche loro pronti a procedere. Il sistema aggiorna automaticamente l'output dei **Client**, non serve alcun input da parte dell'utente.



```
Wait for the other players to be ready...
```

Figure 15: Waiting message

Una volta che a tutti gli **User** sono state assegnate le prime due carte, viene estratta la carta del **Croupier** e visualizzata a schermo.



```
First croupier card is:  
- QUEEN CLUBS
```

Figure 16: First Croupier's card

A questo punto ogni utente deve effettuare il proprio gioco e decidere se richiedere altre carte premendo h (hit) oppure di premere s (stand) per confermare le proprie carte

```
Do you want another card? (Press h for hit a new card or s for stand)

The dealer's cards:
    QUEEN CLUBS
Total cards value : 10

Your cards:
    TWO HEARTS - TEN HEARTS
Total cards value : 12

h
```

Figure 17: Hit a new card or stand

Premendo h viene estratta casualmente una nuova carta e assegnata all'utente.

```
.
.
.
Your new card is:
- ACE CLUBS
```

Figure 18: New User's card

Se con la carta estratta la somma dei valori delle carte dell'utente è minore o uguale a 21, l'utente può decidere di estrarre un'altra carta. Altrimenti può inserire s per confermare le proprie carte e proseguire con il gioco.

```
Do you want another card? (Press h for hit a new card or s for stand)

The dealer's cards:
    QUEEN CLUBS
Total cards value : 10

Your cards:
    TWO HEARTS - TEN HEARTS - ACE CLUBS
Total cards value : 13

s
```

Figure 19: Stand

Un messaggio mostra che si è in attesa degli altri utenti presenti nella Lobby che completino il loro gioco e premano s.

A black rectangular box with white text that reads "Wait for the other players to be ready..."

Figure 20: Waiting message

A questo punto viene svolto il gioco del croupier estraendo casualmente le sue carte visualizzandole a schermo. Viene mostrato un riassunto della mano di gioco mostrando le carte in gioco dell'utente e del croupier con la somma dei relativi punteggi e successivamente un messaggio che indica il risultato della partita. A questo punto della partita viene aggiornato il saldo disponibile dell'utente.

A black rectangular box with white text showing the hand result. It lists the dealer's cards (Queen Clubs, Four Spades, Queen Spades) with a total value of 24, and the user's cards (Two Hearts, Ten Hearts, Ace Clubs) with a total value of 13. The result is "WIN".

Figure 21: Hand result

Al termine della mano di gioco viene chiesto allo User se vuole effettuare una nuova partita. Nel caso in cui lo User volesse giocare nuovamente (premendo y) verrebbe chiesto il valore della scommessa per la nuova partita. Nel caso in cui invece lo User premesse n, verrebbe eliminato dalla Lobby e verrebbe chiusa l'applicazione.

A black rectangular box with white text that reads "Do you want to play again? (Press y for yes or n for no)" followed by a small grey square.

Figure 22: Final message

## 8 Conclusions

Come progetto è stato deciso di realizzare in una versione semplificata, il gioco Blackjack in versione online.

La progettazione e la realizzazione del sistema si sono basati sull'implementazione di un'architettura **Client-Server** con l'utilizzo di **HTTP** come protocollo di comunicazione e utilizzando le **ReST API** per permettere l'interazione con il servizio web.

Il sistema realizzato è stato sviluppato in modo tale da rispettare la scalabilità riuscendo a gestire un numero di **User** variabile.

### 8.1 Future Works

In una versione futura si potrebbe rendere il sistema persistente aggiungendo un database, permettendo così anche la realizzazione di una classifica e un meccanismo di autenticazione, in modo tale da poter, ad esempio, mantenere memorizzate le informazioni relative al saldo rimanente.

Si potrebbe realizzare una vera interfaccia grafica, permettendo di rendere più realistico il gioco. Inoltre si potrebbe pensare di implementare una chat con la quale comunicare con gli altri giocatori presenti nella lobby, potendo dare e ricevere consigli.

### 8.2 What did we learned

Con la realizzazione del presente progetto sono riuscito a sperimentare ed a mettere in pratica gli argomenti trattati durante il corso. Più in particolare, avendo realizzato il sistema come un Web Service, ho sperimentato e appreso appieno la struttura e il funzionamento di questo pattern per i sistemi distribuiti. Inoltre questo elaborato mi ha permesso di affrontare gli aspetti principali del ciclo di vita di un progetto, affrontando un'analisi iniziale seguita da una progettazione del sistema, passando poi alla fase di sviluppo seguita da una di testing finale.

## 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>.