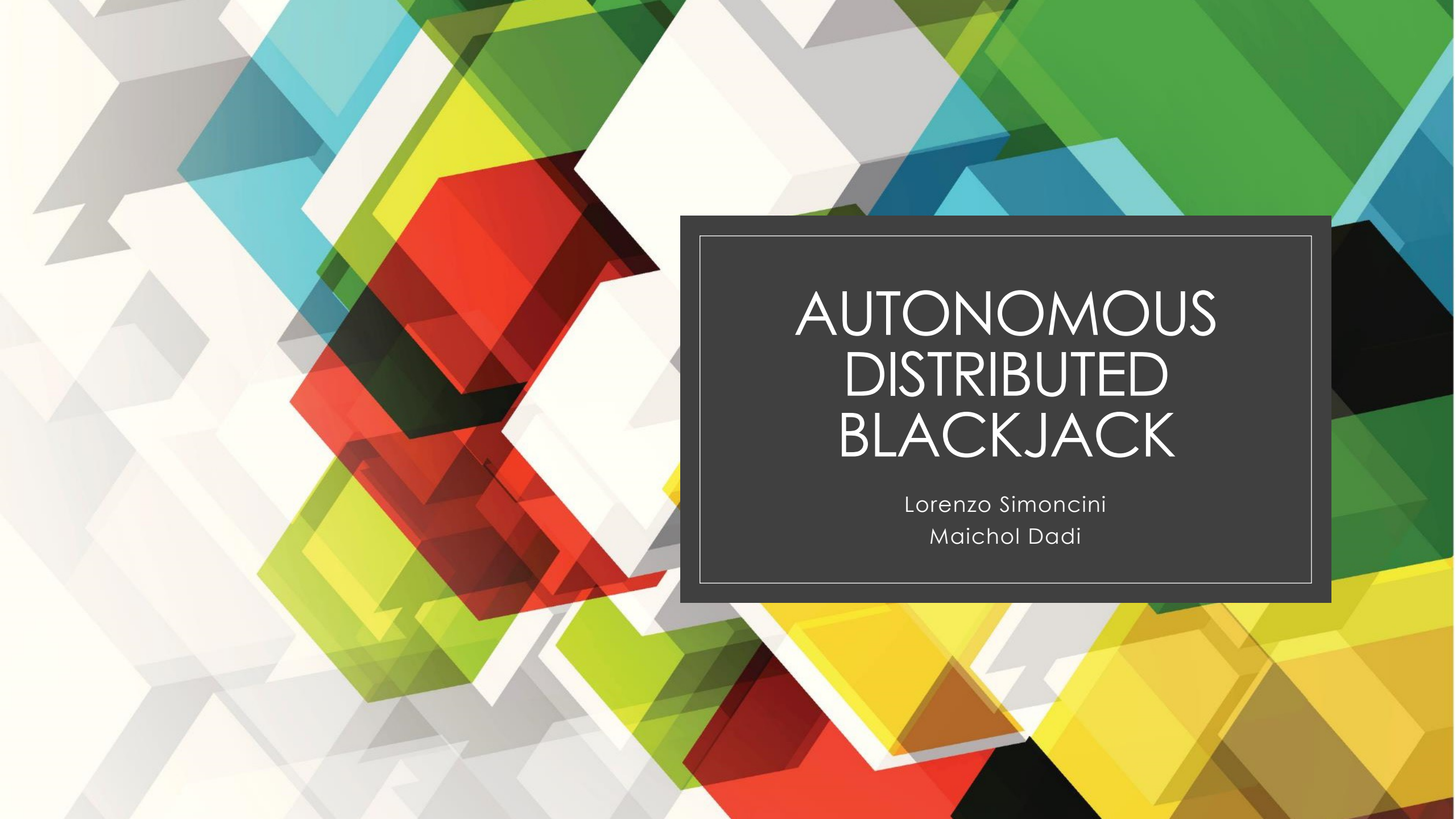


AUTONOMOUS DISTRIBUTED BLACKJACK

Lorenzo Simoncini
Maichol Dadi

Introduzione

- Lo scopo del progetto è la realizzazione di una versione autonoma e distribuita del gioco di carte blackjack. Ogni giocatore partecipa alla partita in maniera autonoma, scegliendo quali mosse effettuare nel proprio turno per vincere la partita. Il sistema è realizzato con un'architettura client-server mediante l'utilizzo di WebSocket per la gestione della comunicazione tra le parti, con l'implementazione di un timeout che vada ad escludere eventuali giocatori che per qualsiasi motivo dovessero interrompere la comunicazione con il server.

Regole del gioco

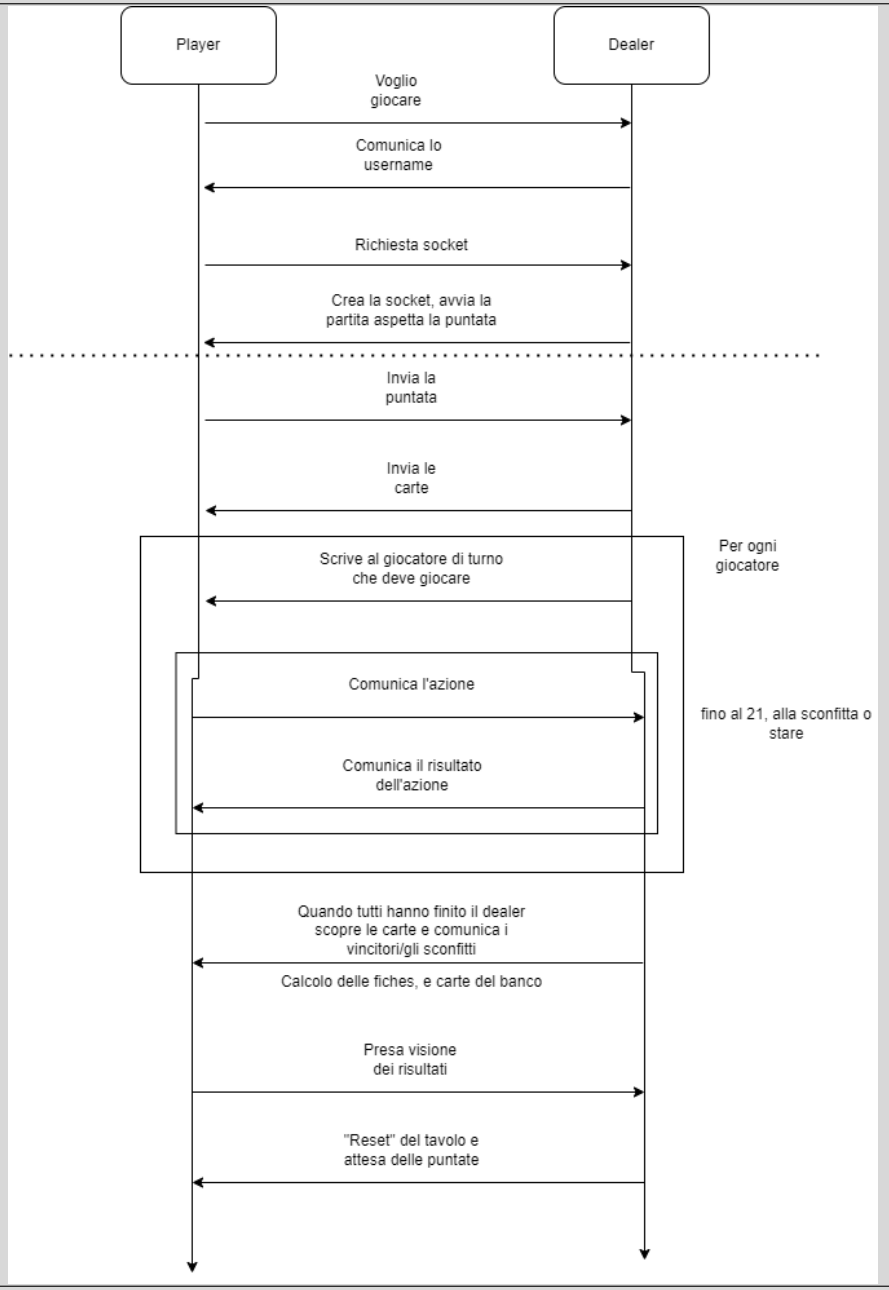
- Il Blackjack si svolge tra il banco (o dealer) ed un numero variabile di giocatori; essi ricevono delle carte dal banco e ne possono richiedere altre nel corso della partita. Lo scopo di ogni giocatore è quello di avere delle carte il cui valore superi quello del manco ma che rimanga inferiore a 22. Per farlo, un giocatore può effettuare tre azioni diverse ad ogni turno:
- Hit: richiede una carta al banco;
- Double: raddoppia la puntata corrente, riceve una carta dal banco e passa il turno;
- Stand: mantiene le carte ricevute inizialmente e passa il turno.
- Al termine del turno di ogni giocatore il dealer esegue la propria mossa ed in seguito scopre le sue carte e determina i vincitori della partita.

Architettura generale

- Per la comunicazione tra le parti che comprendono il gioco (giocatori e dealer) è stata realizzata un'architettura client-server tramite Vert.x; i client dovranno connettersi al server per entrare in partita e ricevere da esso tutte le informazioni necessarie allo svolgimento del gioco e alla corretta comunicazione e sincronizzazione tra le parti. La comunicazione fa uso di WebSocket e di alcune API per inviare e ricevere messaggi ai giocatori connessi.

Flusso di gioco

Nel diagramma presentato a lato è possibile vedere le azioni di base che compongono il flusso del gioco relativamente ad uno specifico giocatore.

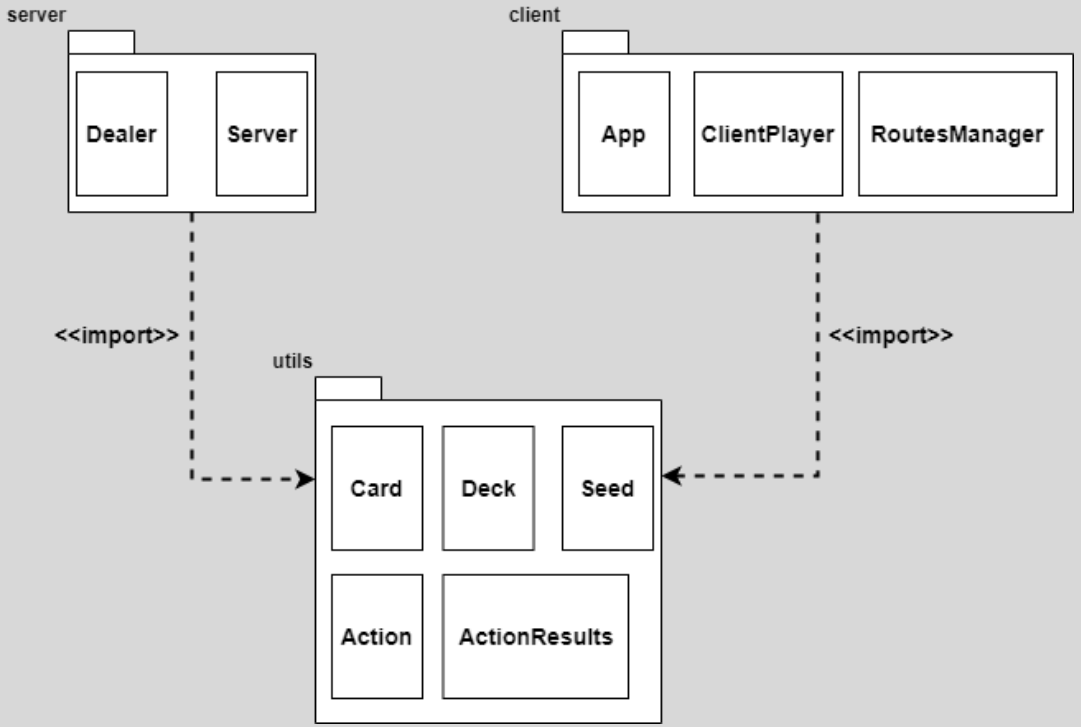


Entrata e uscita dei giocatori

- Nuovi giocatori possono unirsi ad una sessione di gioco in qualsiasi momento inviando una richiesta di partecipazione al server, per poi essere inseriti nella partita successiva a quella in corso. Per fare ciò, al momento della richiesta, se è già in corso una partita, il giocatore verrà inserito in una lista di attesa per poi essere aggiunto alla lista effettiva di giocatori al termine della partita corrente.
- Un giocatore può uscire da una sessione di gioco in diversi modi, che verranno poi gestiti di conseguenza:
- Terminazione delle fiches: la modalità di uscita attesa per la maggior parte dei giocatori; una volta terminate le fiches a disposizione, il giocatore verrà escluso dalla sessione;
- Crash del client: all'avvenire di questo evento il server dovrà escludere il giocatore dalla sessione, senza che ci siano ripercussioni sul corretto svolgimento della partita;
- Mancata ricezione dei messaggi: allo scattare di un timeout il messaggio relativo verrà considerato perso e il giocatore che lo ha inviato/che lo ha ricevuto verrà escluso dalla sessione.

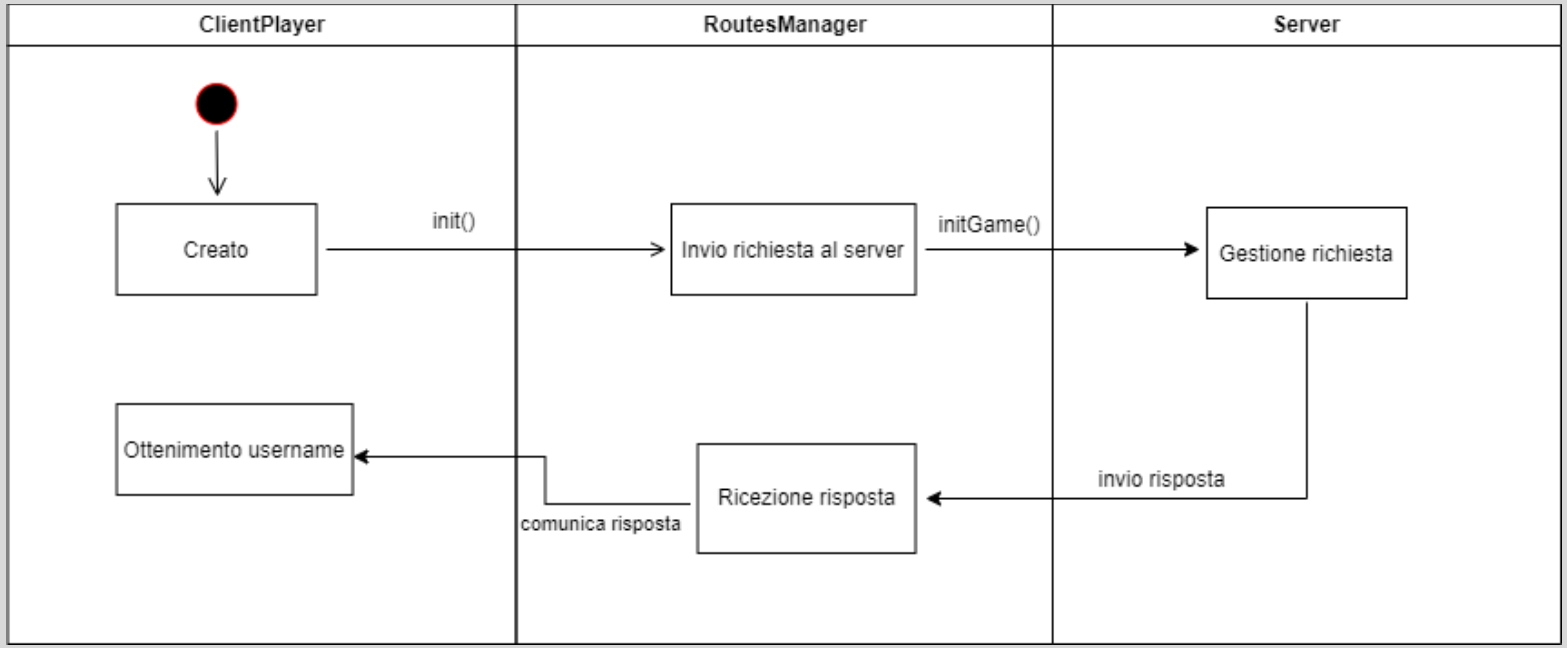
Struttura del progetto

- Di seguito vengono presentati i package in cui è diviso il progetto e le classi al loro interno.
- Sono presenti tre package principali, server, client e utils; i primi due contengono classi relativamente server-side e client-side, mentre il terzo contiene delle classi di utility che vengono importate in caso di necessità dalle classi degli altri due package.



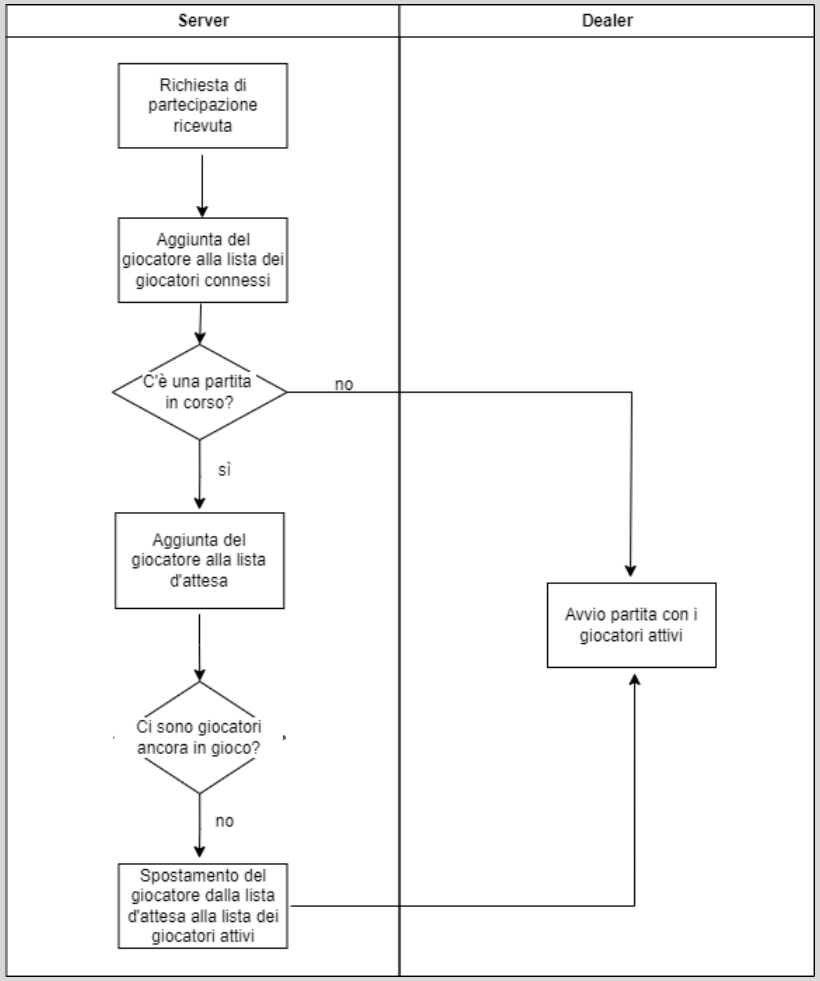
Attesa di nuovi giocatori

- Nel diagramma delle attività qui presentato è possibile vedere nel dettaglio lo scambio di informazioni che avviene tra le varie entità per gestire l'entrata in sessione di un nuovo giocatore e l'ottenimento di uno username univoco.



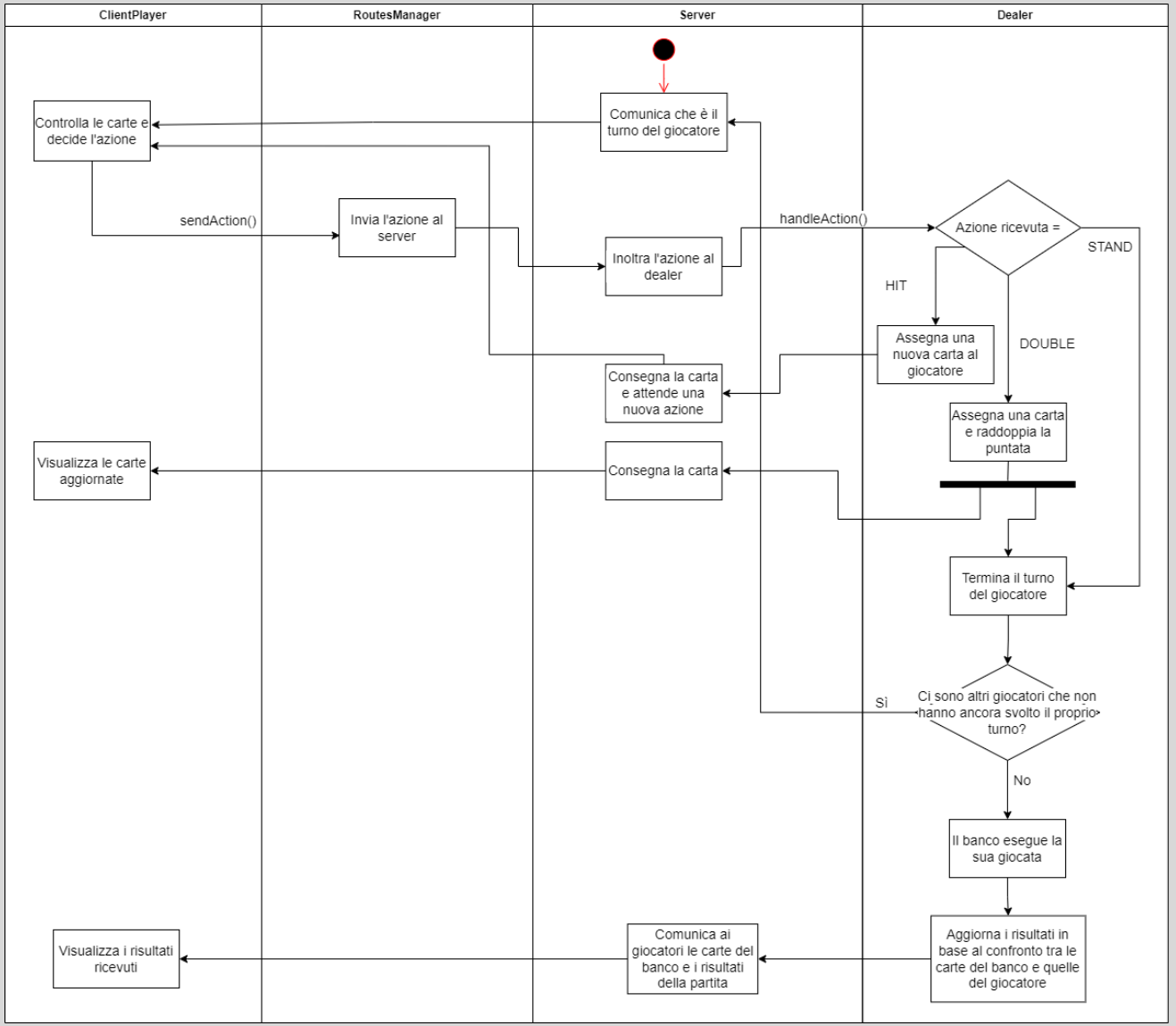
Inizio partita e gestione lista d'attesa

- Quando il server riceve una richiesta di inizializzazione da un nuovo giocatore lo aggiunge alla propria lista di partecipanti connessi e, se non c'è una partita già in corso comunica al dealer di avviarne una con i giocatori correnti; viceversa, aggiunge momentaneamente il giocatore ad un una lista d'attesa e controlla l'eventuale presenza di giocatori attivi (poiché è possibile ad esempio che ci sia un partita in corso ma che tutti i giocatori che vi partecipavano abbiano perso, venendo esclusi dalla lista di giocatori attivi), e se non ne risultano sposta il giocatore appena messo in lista d'attesa nella lista dei giocatori attivi, avviando una nuova partita.



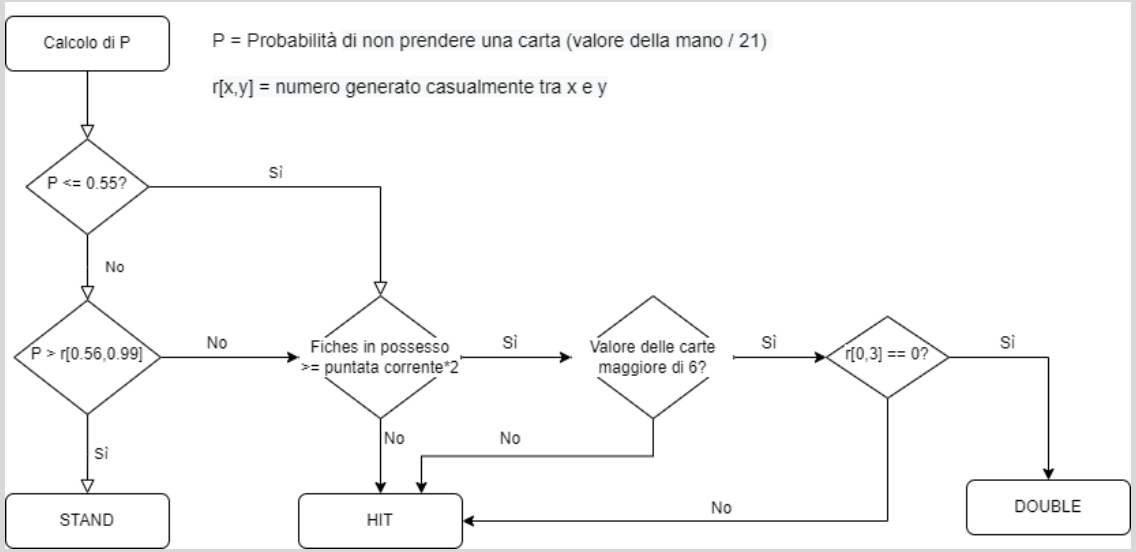
Svolgimento del turno

- Nel seguente diagramma vengono mostrate le azioni che vanno a comporre lo svolgimento del turno per ogni giocatore, in particolare;
- Invio dell'azione (Hit, Double, o Stand) al banco
- Ricezione dell'azione e gestione di essa da parte del banco
- Eventuale terminazione della partita e visualizzazione dei risultati



Logica del giocatore

- Il giocatore decide quale mossa eseguire basandosi sul valore delle carte in suo possesso; il processo decisionale è riassunto nello schema seguente e nella tabella è possibile vedere le probabilità di effettuare ciascuna mossa in base alle carte in mano. Il rapporto tra le probabilità di «Hit» e «Double» viene mantenuto 3:1, mentre la probabilità di «Stand» è più alta tanto più alto è il valore delle carte possedute.



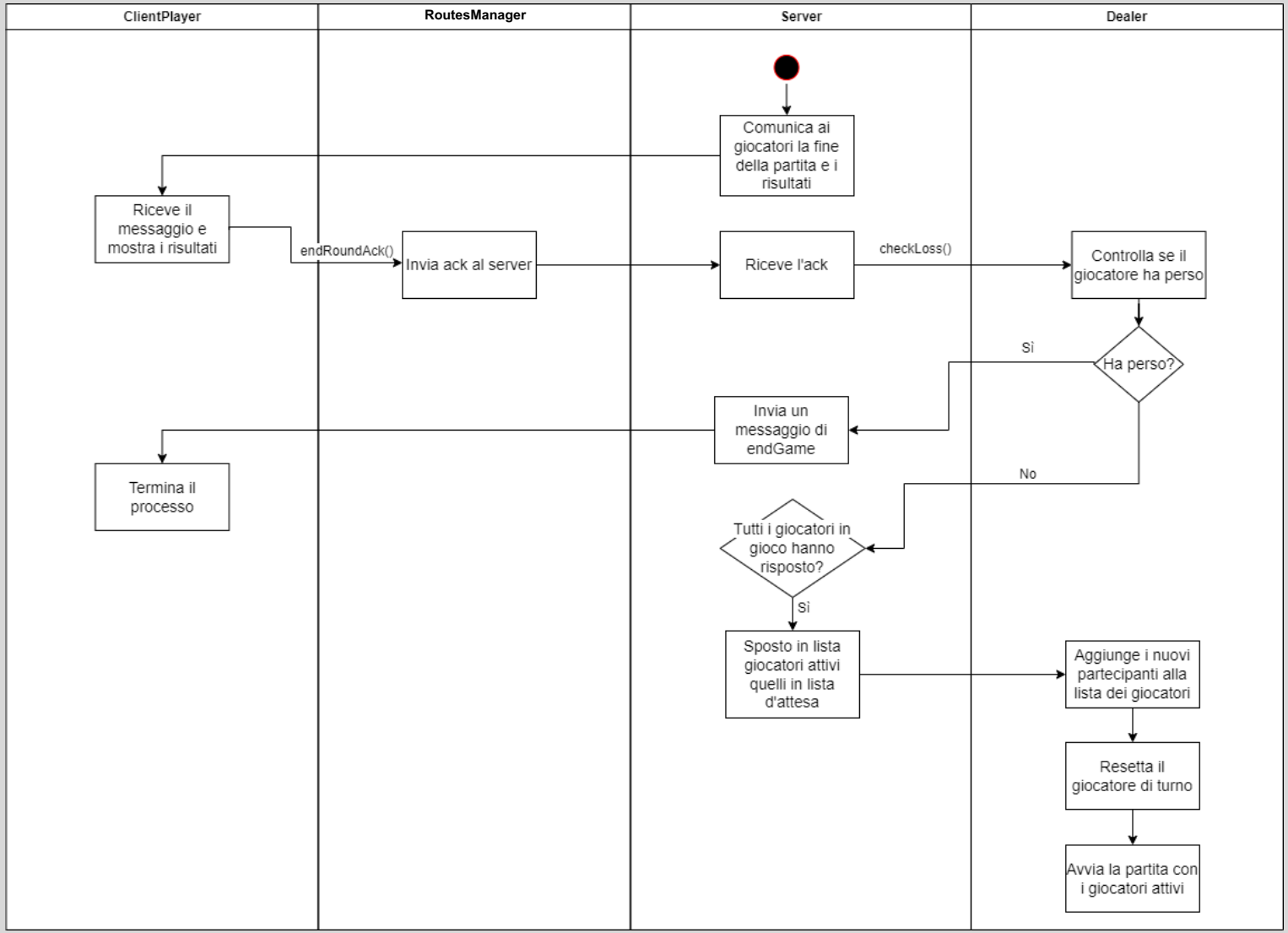
Decisione della mossa			
Valore	Stand %	Hit %	Double %
4	0%	100%	0%
5	0%	100%	0%
6	0%	100%	0%
7	0%	75%	25%
8	0%	75%	25%
9	0%	75%	25%
10	0%	75%	25%
11	0%	75%	25%
12	57%	32%	11%
13	62%	29%	10%
14	67%	25%	8%
15	71%	21%	7%
16	76%	18%	6%
17	81%	14%	5%
18	86%	11%	4%
19	90%	7%	2%
20	95%	4%	1%
21	100%	0%	0%

Logica del banco

- Nel momento in cui deve eseguire la sua giocata, il banco procede seguendo le regole del Blackjack, ovvero pescando una carta fino a che il valore della sua mano non sia pari o superiore a 17.

Fine della partita

- Di seguito vengono presentate le azioni da eseguire al termine di ogni partita:
- Invio dei risultati ai giocatori
- Eventuale notifica della sconfitta
- Spostamento dei giocatori in attesa nella lista dei giocatori attivi
- Avvio di una nuova partita



Interazioni client-server: API

- La comunicazione tramite client e server avviene sia tramite le API progettate che tramite scambio di messaggi sulle websocket.
- Esistono quattro API utilizzate dai client in diversi momenti del gioco.
- init (GET): viene utilizzata per comunicare la propria presenza e per unirsi alla sessione di gioco; non ha parametri.
- bet (PUT): viene utilizzata per inviare al server la puntata relativa alla partita in corso; ha come parametri username (l'identificativo del giocatore) e bet (il valore della puntata).
- makePlay (PUT): viene utilizzata per comunicare la propria mossa all'interno del turno; ha come parametri username (l'identificativo del giocatore) e action (la mossa da eseguire).
- endRound (PUT): viene utilizzata per comunicare al server la ricezione dei risultati a fine partita; ha come parametro username (l'identificativo del giocatore)

Interazioni client-server: WebSocket

- I messaggi inviati sulle websocket (dal server al client) possono essere di diverso tipo, e in base ad esso contenere informazioni differenti:
- startGame: viene inviato all'inizio di ogni partita, e contiene le fiches attuali del giocatore.
- cardDistribution: viene inviato quando il server distribuisce una o più carte al giocatore, e contiene le carte possedute da ogni giocatore e dal banco.
- makeMove: viene inviato per comunicare al giocatore che è il suo turno e che può quindi effettuare la propria mossa; non contiene informazioni aggiuntive.
- endOfRound: viene inviato per comunicare la fine della partita; contiene le carte del dealer (ora scoperte) e il moltiplicatore delle fiches del giocatore (il cui valore è stato calcolato in base all'esito della partita).
- endGame: viene inviato al giocatore in seguito alla sua sconfitta; non contiene ulteriori informazioni.

Interfaccia grafica

- Per mostrare a schermo le informazioni importanti viene effettuata una semplice stampa su console che mostri le carte possedute dai giocatori e le azioni che compiono (lato client) e le richieste ricevute e le carte del dealer (lato server).

```
-----Action: cardDistribution------

Player: 1
|8 Spade |
|6 Hearts |
|2 Hearts |
|9 Hearts |
---------

Player: 2
|5 Hearts |
|11 Clubs |
|2 Spade |
|2 Diamonds |
|4 Spade |
---------

Dealer
|8 Clubs |
|Covered Card|
---------

-----Action: makeMove------

------Giocatore 2------
| Cards value : 23 |
-----------------------
| Action:  STAND  |
-----------------------


-----Action: endOfRound------

Dealer cards
|8 Clubs |
|10 Clubs |

------Giocatore 2------
| Risultato round : perso|
-----------------------
```

```
[SERVICE] PUT - Richiesta da :http://localhost:12000/api/makePlay/0/HIT
        Gestione Hit
[SERVICE] Distribuzione carte
[SERVICE] PUT - Richiesta da :http://localhost:12000/api/makePlay/0/STAND
        Gestione Stand
        Turno del banco
        Valore carte: 9
        Pesco una carta perché il mio valore è minore di 17
        Pesco una carta perché il mio valore è minore di 17
        Gestione fine partita
        Valore Carte: 18
        Fine della partita gestita
```

Test

- Sono stati realizzati dei semplici test JUnit per verificare il corretto funzionamento di alcune logiche di gioco:
- Verifica della corretta inizializzazione di un giocatore
- Verifica della ricezione delle carte inviate dal server
- Verifica dell'assegnazione del moltiplicatore di punti corretto in seguito ad un BlackJack
- Verifica del corretto calcolo del valore degli assi (che può essere sia 11 che 1 a seconda del valore della mano; 11 se il valore è inferiore o uguale a 21, altrimenti 1