

# Progetto ingegneria del software

Anno accademico 2022-2023

Lorenzo Magni, matricola 1073257

Marianna Romelli, matricola 1072382

Saif Bouchemal, matricola 1074800

# Lupus In Tabula



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Obiettivo

Realizzazione in Java di un gioco di ruolo multiplayer online basato sul noto gioco da tavolo Lupus in Tabula.

## Lo svolgimento

Il gioco prevede due fasi che si alternano: il giorno e la notte. Durante il giorno i giocatori vivi discutono su chi siano i lupi mannari per linciarli, durante la notte invece il narratore (server) chiama i vari ruoli speciali che hanno la possibilità di compiere azioni. Il gioco termina quando i villici sono riusciti ad uccidere tutti i lupi mannari o quando i lupi mannari sono riusciti a rimanere in numero uguale a quello dei villici.



# Difficoltà incontrate

- Difficoltà nella suddivisione delle mansioni da svolgere
- Difficoltà nella calibrazione delle tempistiche
- Difficoltà di comunicazione con i membri del team
- Difficoltà nella scelta di un interfaccia grafica adeguata alle nostre esigenze



# Paradigma di programmazione/modellazione utilizzato e tools

- Il team ha usato come paradigma di modellazione Java e UML per quanto riguarda la modellazione.
- Il team ha deciso di utilizzare l'IDE IntelliJ Idea per lo sviluppo del software, la suite Office e gli strumenti di GitHub per la scrittura della documentazione e della presentazione e altri strumenti di photoediting per la creazione e modifica delle immagini del gioco e di grafici e diagrammi.
- Per l'organizzazione del lavoro a distanza e dei meeting sono stati anche sfruttati Discord e Monday.



# Software configuration management

- Tutto il lavoro svolto che si tratti di documentazione o di codice è stato salvato in un repository su Github in condivisione con tutti i membri del team.
- Il repository è strutturato nel seguente modo:
- **branches:**
  - *main*: Contiene le versioni stabili del codice e la documentazione
  - *dev*: Contiene le versioni in via di sviluppo del codice
- **cartelle:**
  - *code*: Contiene il codice sorgente del progetto. È diviso in 3 moduli: common, client e server.
  - *Docs*
  - *.github/workflows*

# Software configuration management

- Nei vari meeting settimanali vengono creati i macro-temi che dovranno essere sviluppati nel corso della settimana seguente. Le varie attività sono create come issue. Durante la settimana i membri del team sono liberi di creare nuove issue in base alle necessità. Le issue sono suddivise in 3 categorie principali:
  - *Task*
  - *Bug*
  - *Enhancement*
- A seconda del loro avanzamento, i task possono trovarsi in diversi stati. Per tenere traccia di ciò è stata utilizzata una Board sulla piattaforma Monday.



# Software life cycle

- Per il processo di sviluppo il team ha scelto un approccio di tipo Agile poiché meglio si adatta alla nostra metodologia di lavoro.
- Abbiamo deciso di organizzarci secondo la filosofia della programmazione estrema (l'extreme programming).
- Nel team non c'è una struttura di tipo gerarchico: ci consideriamo tutti allo stesso livello.
- Durante lo sviluppo del software utilizziamo la tecnica del timeboxing
- Il team è propenso al cambiamento. Nel caso in cui il nostro prodotto venga richiesto da possibili investitori saremo lieti di prendere in considerazioni le proposte.
- Come approccio di progettazione software per lo sviluppo del sistema ci siamo attenuti alla model-driven architecture (MDA).

# Requisiti

- I requisiti sono stati scelti insieme a tutto il team in base alle caratteristiche del gioco. Abbiamo concordati insieme i requisiti essenziali, quelli che avremmo voluto implementare e quelli che non abbiamo avuto modo di svolgere.
- **Must Have:**
  - Sviluppo di una versione base del gioco funzionante
  - Inserimento della traduzione italiana e inglese
  - Multigiocatore online
  - Interfaccia grafica semplice e intuitiva
  - Chat testuale
  - Grafica diversificata per le fasi giorno e notte
  - Stabilità e sicurezza del software



# Requisiti

- **Should have:**
  - Traduzione in altre lingue
  - Guida al gioco interna all'applicazione
  - Chat testuale crittografata e moderata
  - Autenticazione degli utenti per fornirgli un nickname fisso e statistiche sui risultati delle partite
  - Consentire la modifica delle impostazioni della partita
  - Timeout delle lobby dopo un periodo di inattività
  - Impostazioni di gioco
  - Effetti sonori
  - Aggiornamento automatico
  - Chat vocale
- **Won't have:**
  - Impostazioni per modificare la grafica
  - Doppia autenticazione tramite codice OTP inviato al numero di telefono
  - Lobby pubbliche



# Architettura

- Il software è stato progettato sfruttando il pattern *MVC* che permette di separare le responsabilità di gestione dei dati, della grafica e della logica del programma. Il pattern è stato applicato sia al client che al server con la seguente divisione dei compiti:
- Il client gestisce la grafica (view) e le comunicazioni con il server e gli input dell'utente.
- Il server gestisce la logica del gioco (controller) e mantiene le informazioni riguardanti le partite in corso (model), nonché eventuali iterazioni con database per la gestione di login.



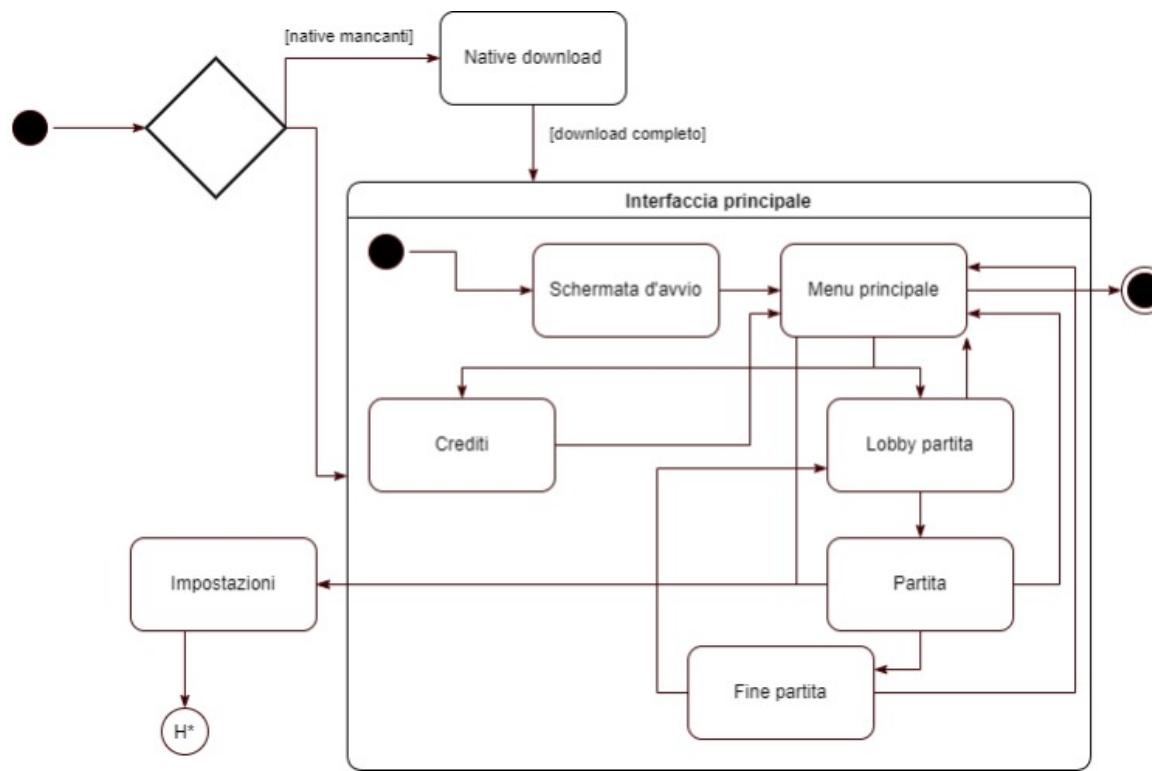
# Design pattern

- Il team ha utilizzato i seguenti design pattern:
  - Factory pattern: nei metodi setter quanto possibile si restituisce l'istanza della classe in modo da facilitare l'uso della programmazione funzionale.
  - Singleton: le classi principali sia del client che del server sono di tipo singleton e hanno un getter statico per poterle ottenere in qualsiasi punto del codice.
  - Observer: all'interno del server quando rimane in attesa di un'azione da parte dell'utente.



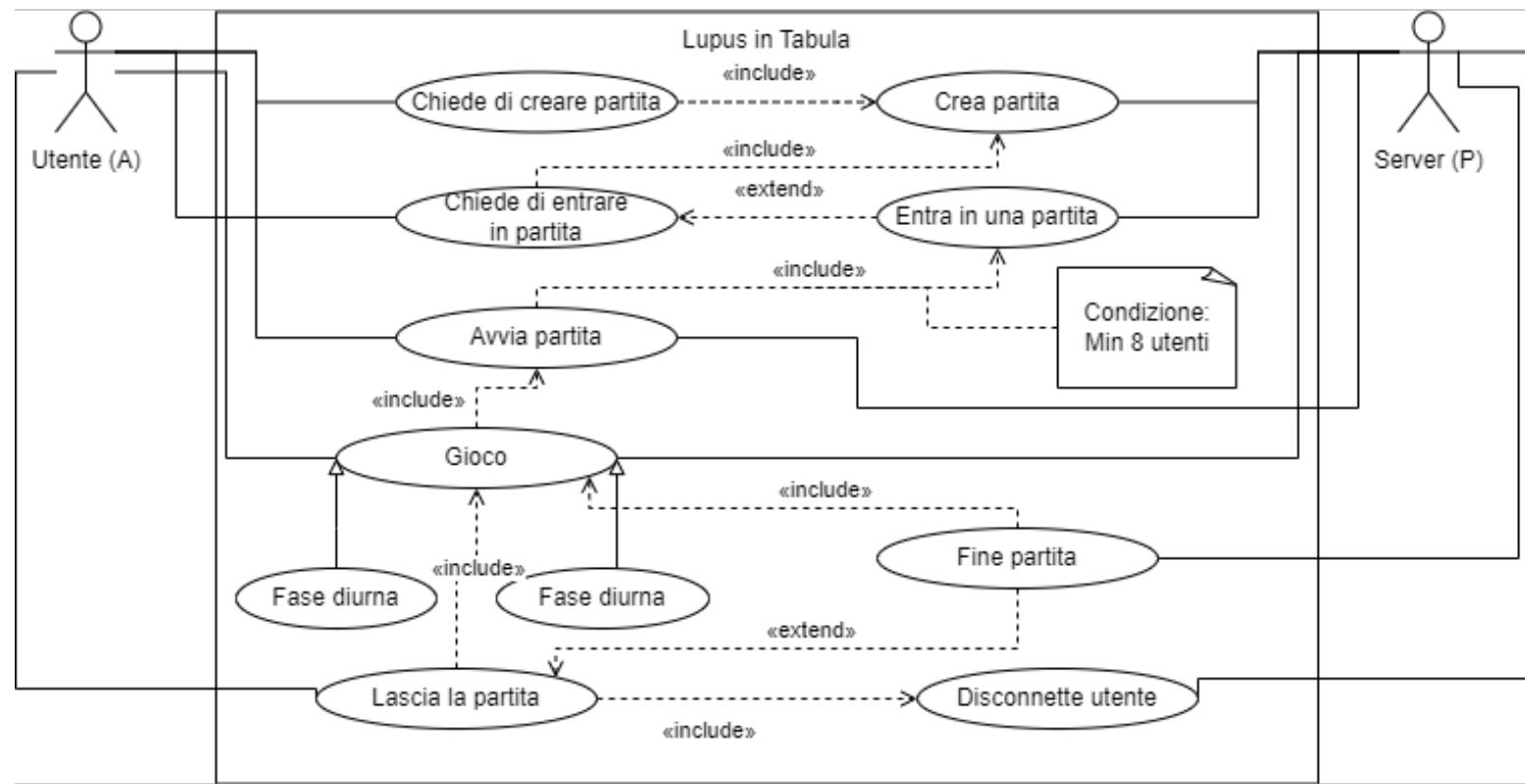
# Modellazione

- State machine diagram



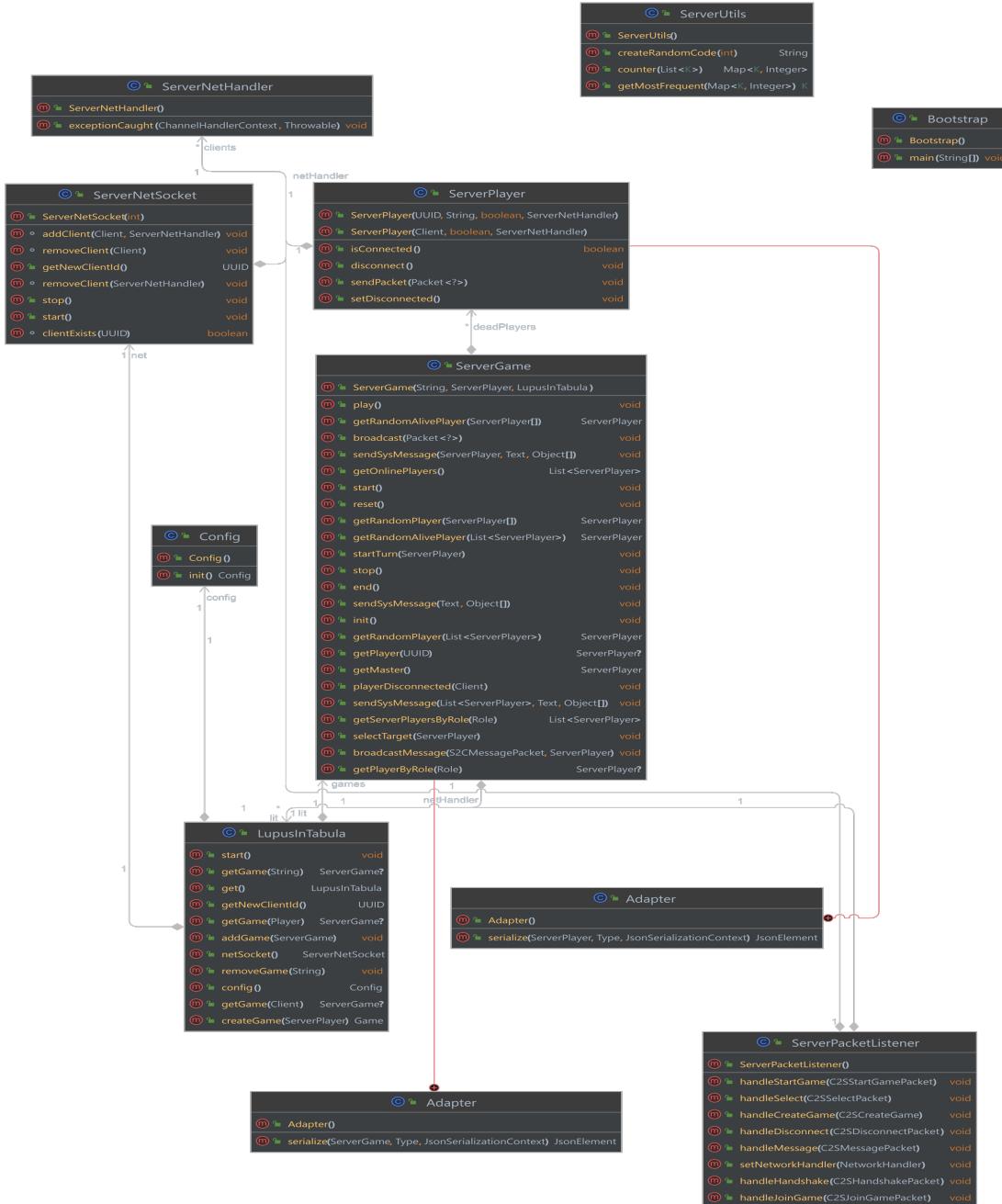
# Modellazione

- Use case diagram



# Modellazione

- Class diagram  
Server



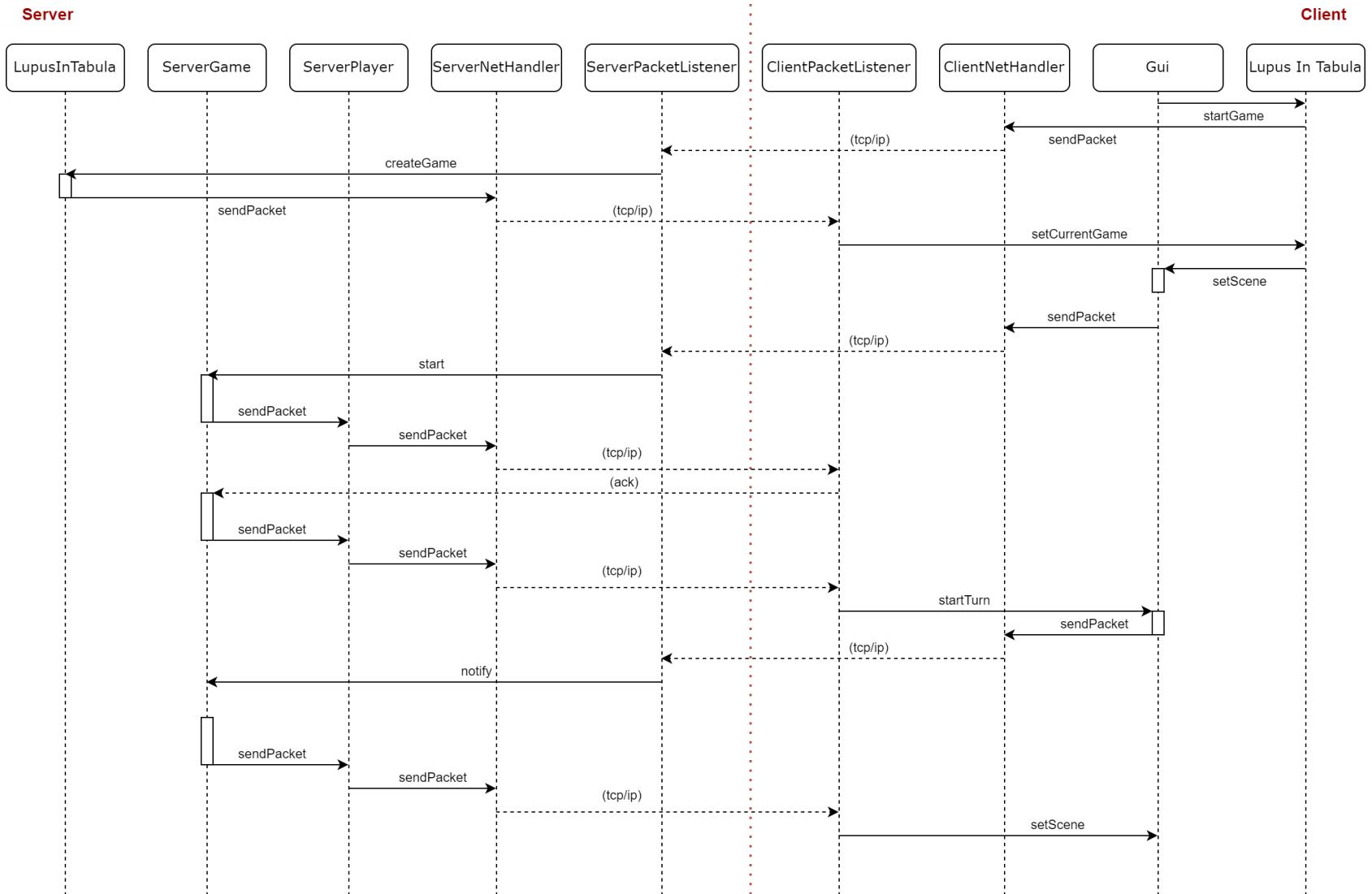
# Modellazione

- Class diagram  
Client



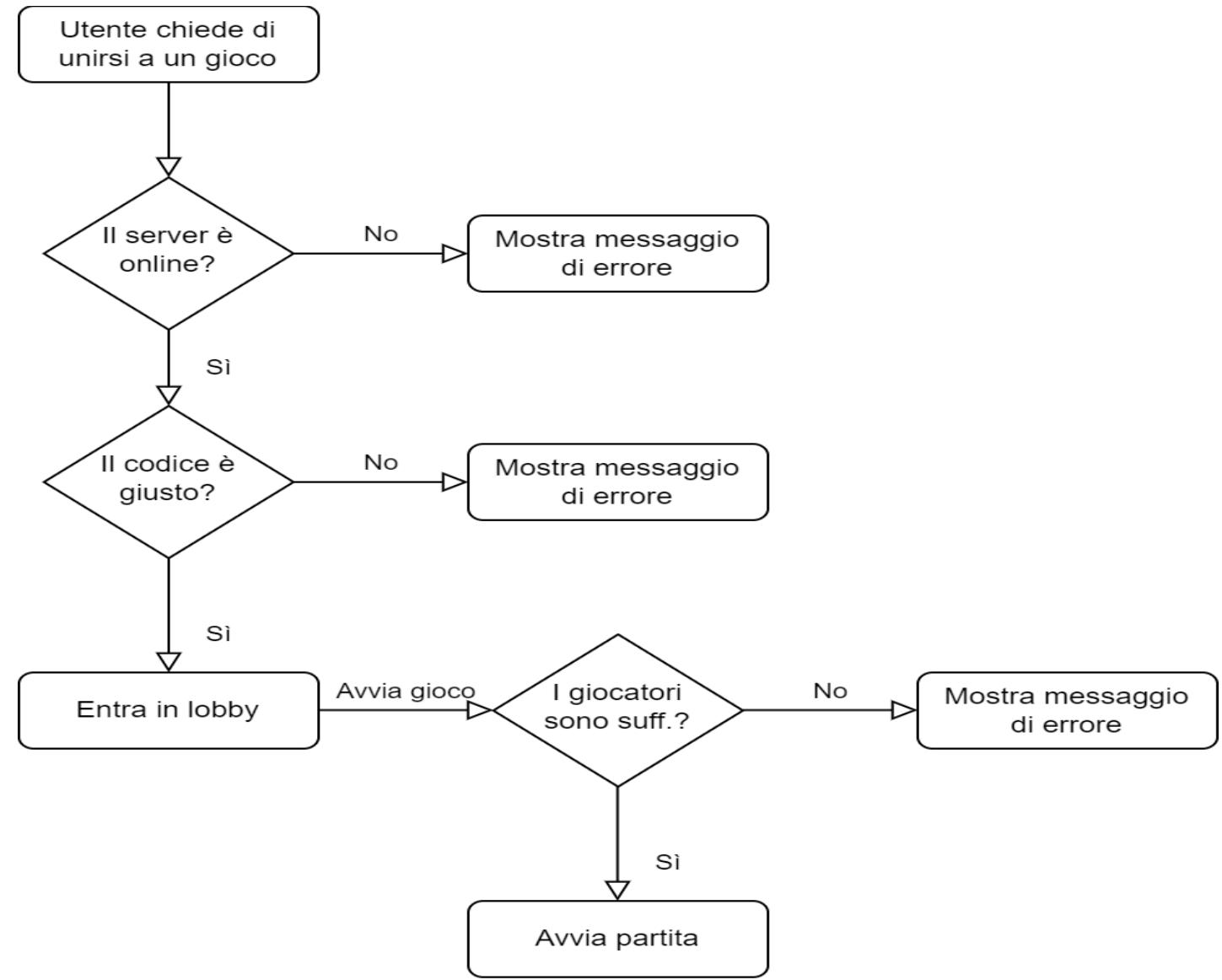
# Modellazione

- Sequence diagram



# Modellazione

- Activity diagram



# Implementazione

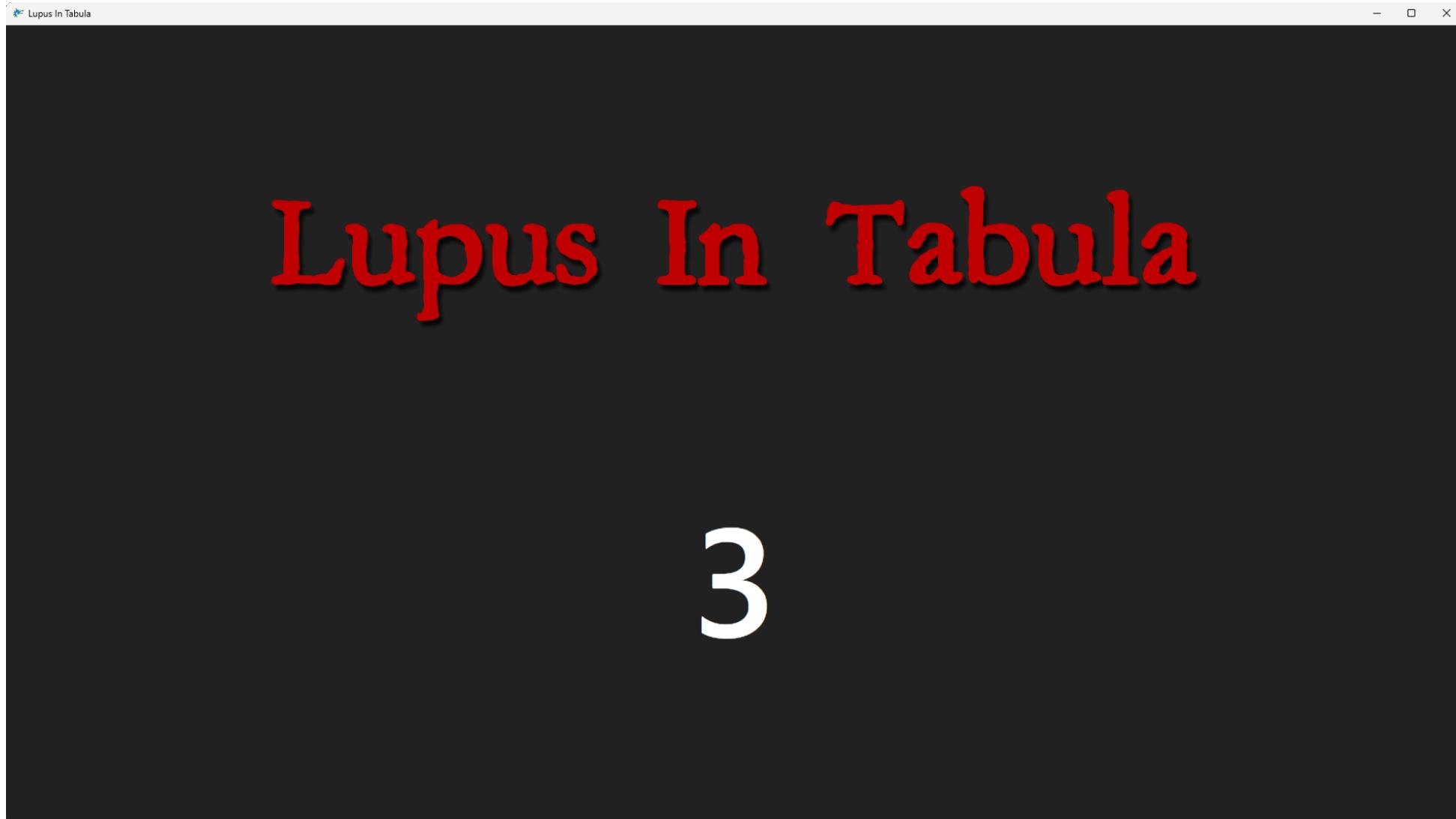
- Per il networking si è scelto di sfruttare la comunicazione tramite pacchetti TCP pertanto si è sviluppata la libreria MCLib-Network basata su Netty.
- Per la gestione dei file di configurazione tramite Json si è usata la libreria Gson.
- Per quanto riguarda la grafica, inizialmente si era pensato di sfruttare OpenGL tuttavia questo avrebbe reso il lavoro molto lungo e complesso. Si è quindi deciso di optare per la libreria JCEF e sfruttare la potenza e versatilità dei linguaggi Web.

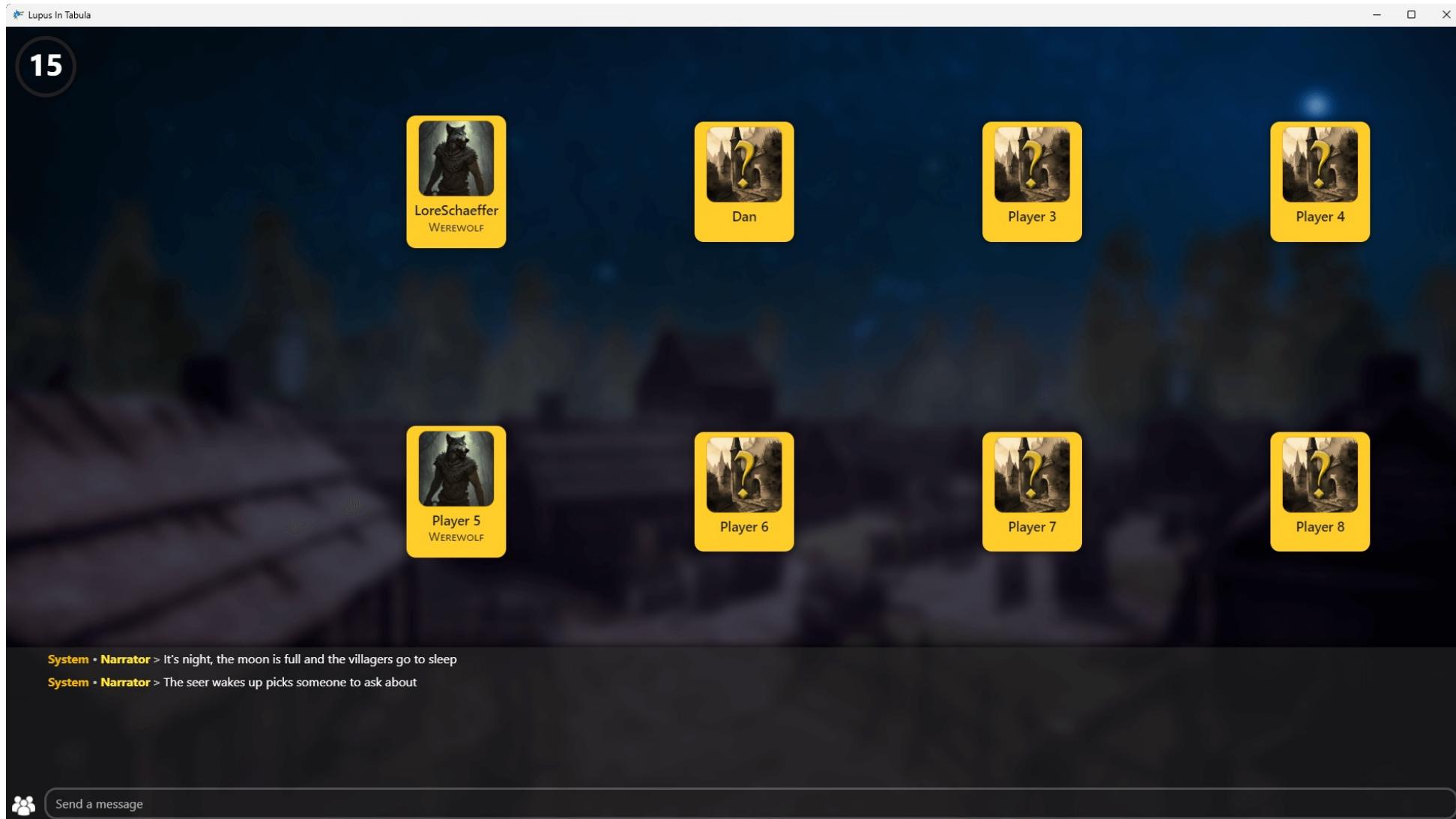


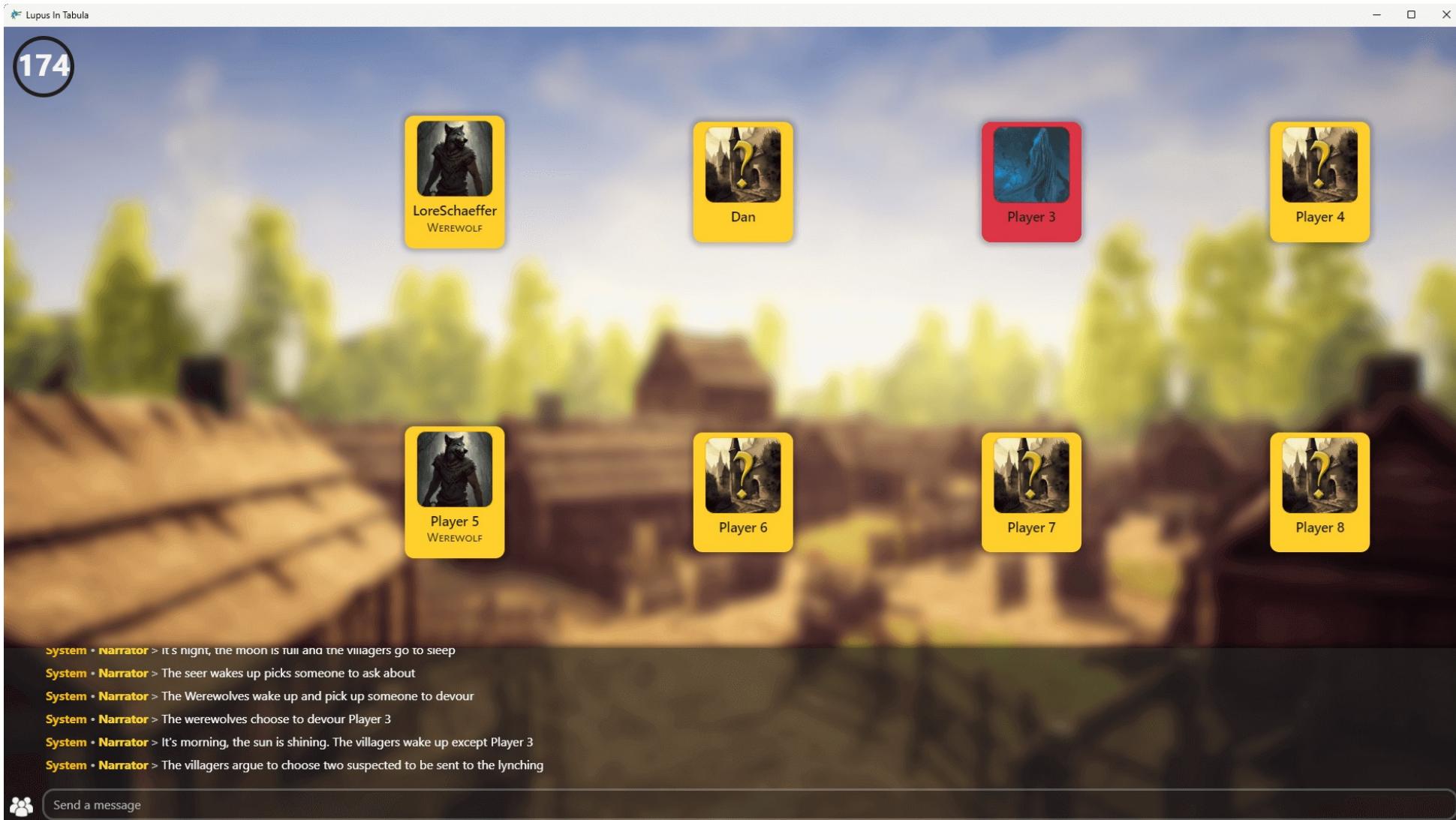
# DEMO











# Testing

- Durante lo sviluppo il corretto funzionamento del codice è stato testato costantemente con test pratici di esecuzione del programma e delle sue funzionalità. Questo processo è stato aiutato particolarmente dalla scelta di inserire all'interno del codice un gran numero di messaggi di debug (attivabili mediante il parametro di avvio -d) che mostrano in molti momenti lo stato del model, e le azioni che vengono svolte da giocatori e dal server.



# Testing

- Una volta scritto lo scheletro del codice e ottenuta una versione più o meno definitiva/funzionante del gioco sono stati implementati dei test automatici mediante JUnit per poter verificare il corretto funzionamento del codice anche in presenza di casi limite. La scrittura di questo tipo di test è, tuttavia, stata resa più complessa dalla necessità di simulare l'interazione tra client e server. Questo non è stato possibile per il rispetto della deadline e quindi è stato programmato per il futuro.

