Il progetto Escape Room è un'applicazione web multiplayer dove i giocatori collaborano per risolvere puzzle e quiz entro un limite di tempo prestabilito

Tecnologie Utilizzate

Frontend: React per l'interfaccia utente

Backend: Node.js con Express

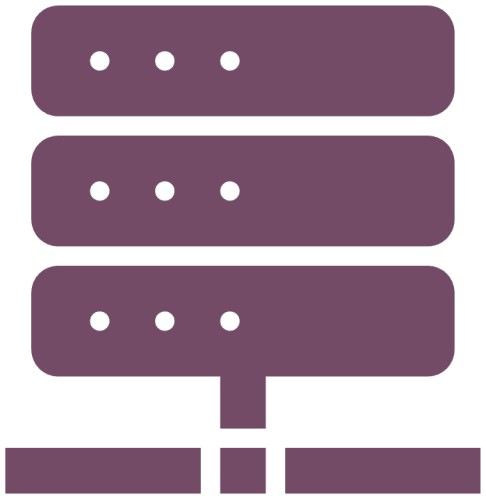
Comunicazione Real-time: Socket.io

Rendering del gioco: Phaser.js per i puzzle

ESCAPE ROOM

Daniel Meco

ARCHITETTURA CLIENT-SERVER

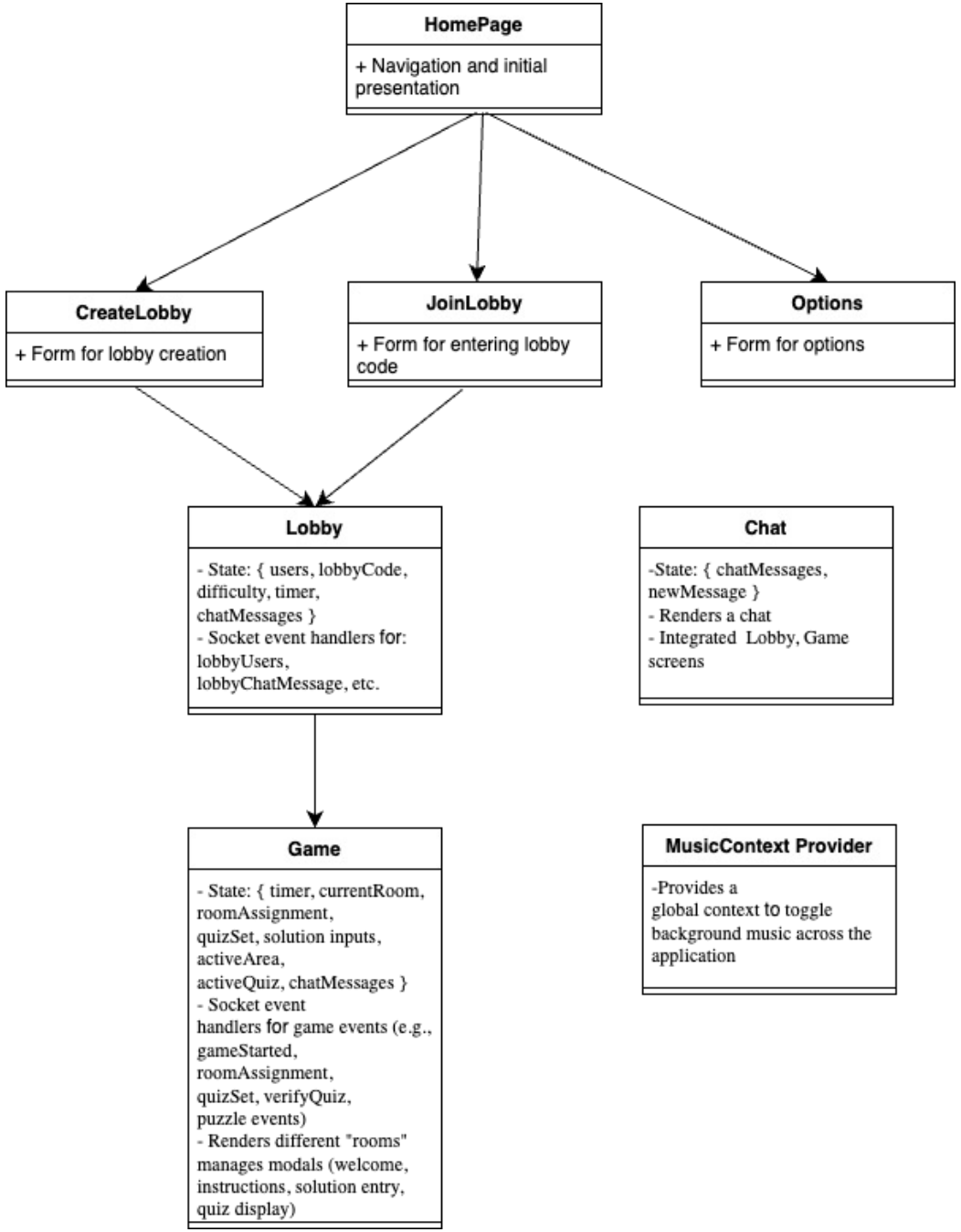


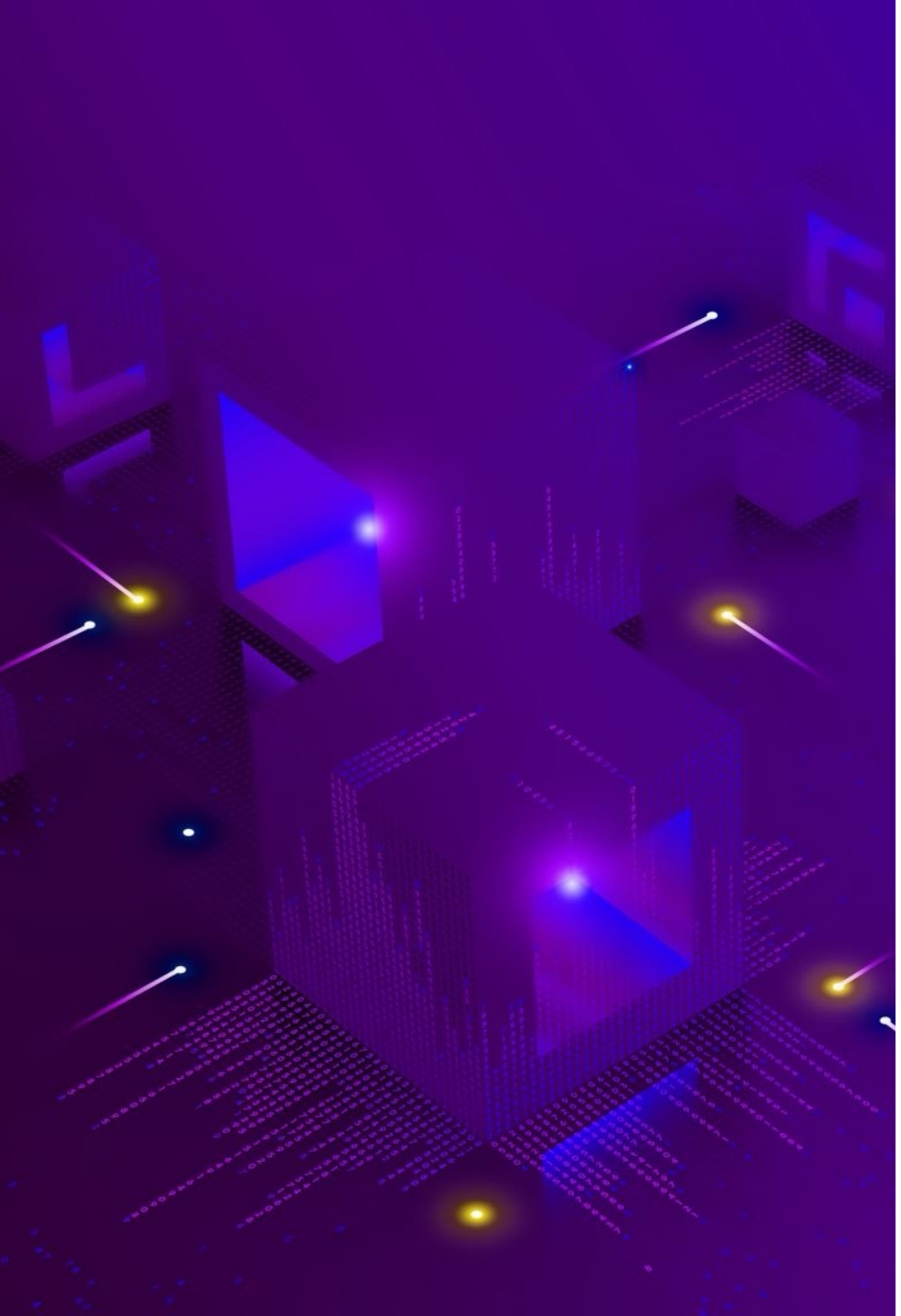
Il progetto utilizza un'architettura basata su WebSocket attraverso Socket.io che permette comunicazioni bidirezionali in tempo reale tra client e server.

Il punto di partenza per tutta la comunicazione real-time nell'applicazione Escape Room è il file socket.js:

1. Il file crea un'unica istanza Socket.io che viene riutilizzata in tutta l'applicazione
2. Socket.io stabilisce la connessione appena il modulo viene importato
3. Tutti i messaggi scambiati tra client e server sono in formato JSON

SCHEMA DELLE CLASSI CLIENT-SIDE

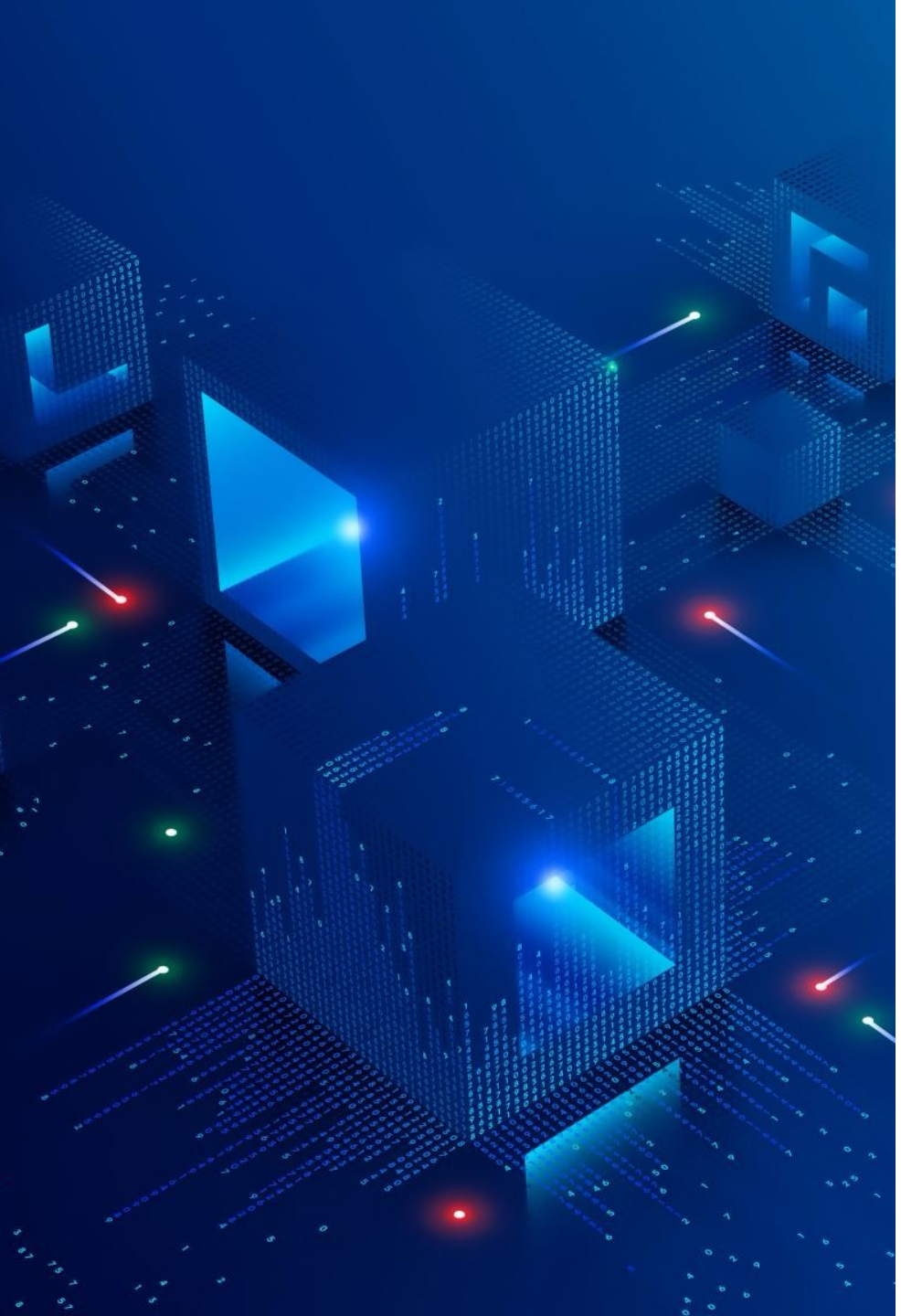




Il componente CreateLobby.jsx rappresenta la pagina attraverso la quale un utente può creare una nuova lobby di gioco nell'applicazione Escape Room.

1. Il server è responsabile della generazione del codice lobby e della creazione effettiva della sala.
2. I dati della lobby appena creata vengono inviati dal server al client, che li trasferisce alla pagina successiva attraverso React Router:

Il componente JoinLobby.jsx gestisce l'interfaccia e la logica per permettere a un utente di unirsi a una lobby esistente nell'applicazione Escape Room.



La Lobby è un componente che funge da sala d'attesa prima dell'inizio effettivo del gioco. In questa fase, i giocatori possono comunicare, visualizzare i partecipanti e segnalare la loro prontezza per avviare la sessione.

Game.jsx è il componente principale dell'esperienza di gioco, rappresentando un'escape room virtuale collaborativa dove i giocatori devono risolvere una serie di enigmi entro un tempo limite.

Il componente è strutturato in diverse "stanze" virtuali con meccaniche di gioco specifiche:

- **Room1**: Stanza iniziale/hub da cui si accede alle altre stanze
- **Room2**: Stanza con il safe (cassaforte) dove inserire soluzioni
- **Room4**: Stanza con puzzle e quiz visivi

DIAGRAMMA DI SEQUENZA LOBBY

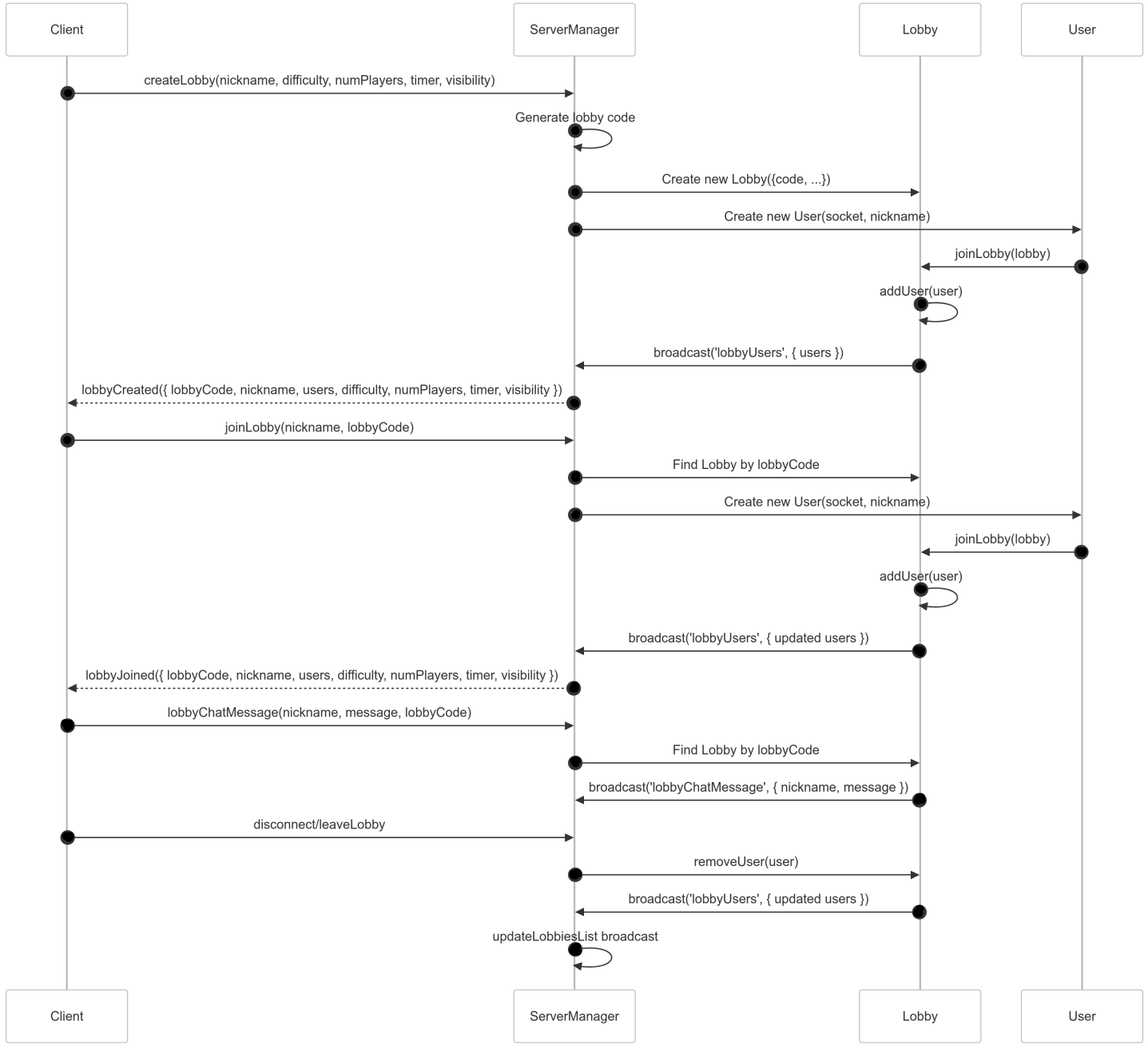
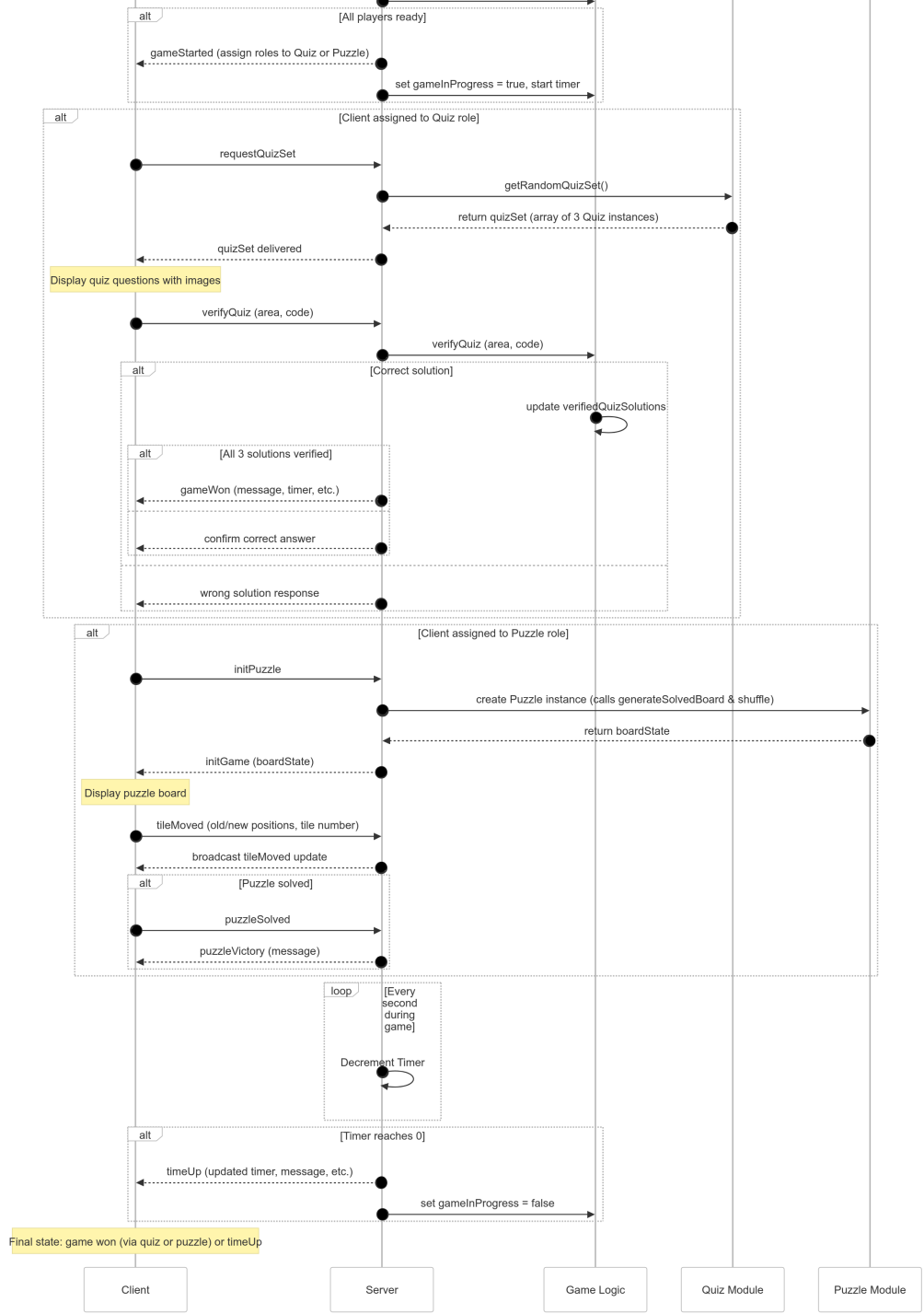


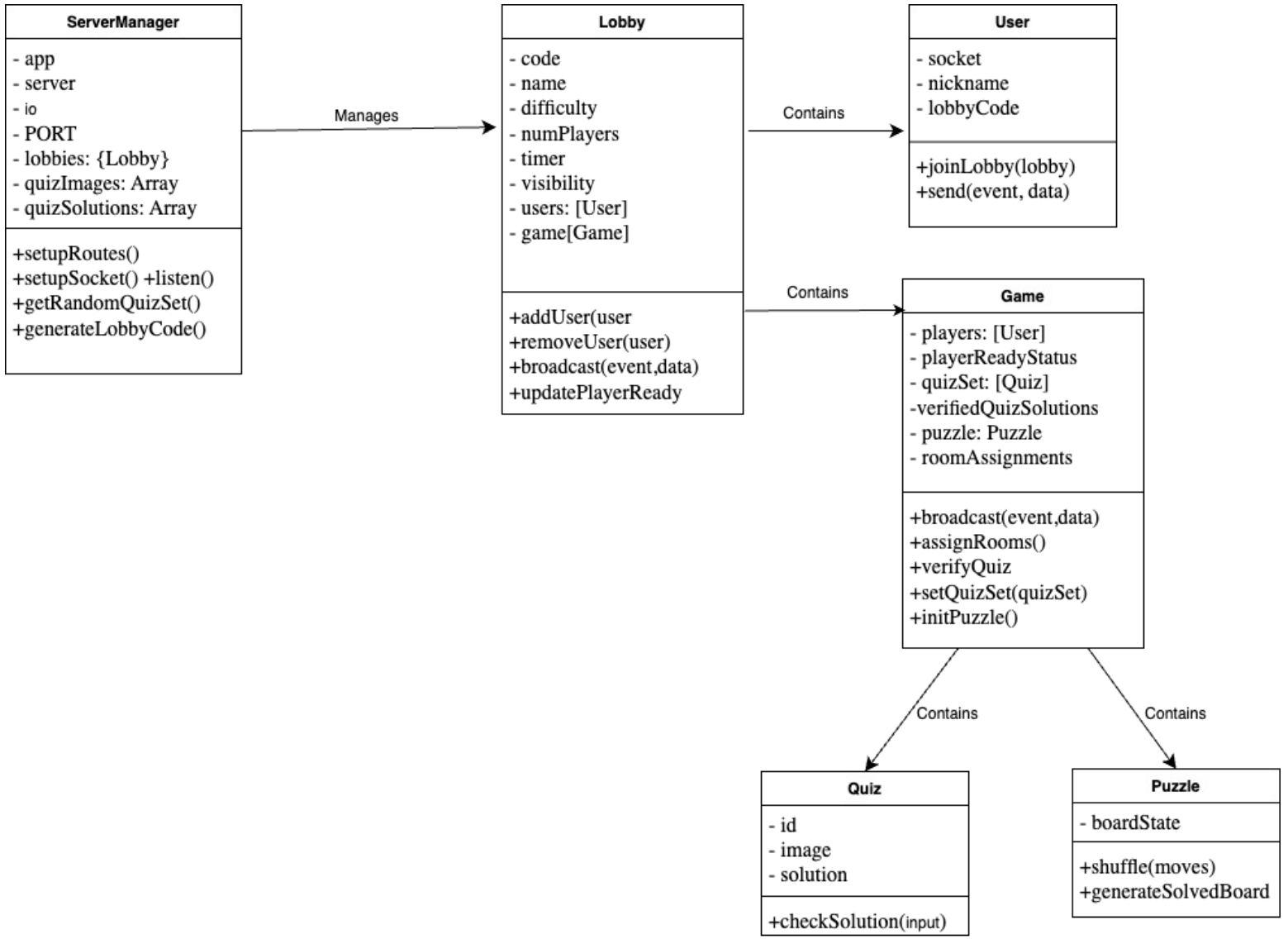
DIAGRAMMA DI SEQUENZA - GAME



PATTERN UTILIZZATI

- **Event-Driven Pattern:** è il pattern dominante nel sistema
- **Request-Response Pattern:** presente sia nelle interazioni HTTP tradizionali (gestite da Express) che in alcune comunicazioni Socket.io (callback)
- **Time-Based Pattern:** usato nel limite di tempo della partita e il sistema di riconnessione.
- **Observer Pattern:** implementato attraverso il meccanismo di broadcasting
- **State Management Pattern:** il gioco mantiene e gestisce diversi stati

SCHEMA DELLE CLASSI SERVER-SIDE



Il file server.js implementa un server Node.js che gestisce l'intera logica di backend dell'Escape Room, rappresentando il nucleo dell'architettura distribuita del sistema.

- **Struttura Modulare e OOP**

```
class User {...} // Gestisce un utente connesso

class Quiz {...} // Rappresenta un singolo quiz con
  soluzione

class Puzzle {...} // Gestisce il puzzle sliding

class Game {...} // Gestisce la logica di gioco

class Lobby {...} // Rappresenta una stanza virtuale

class ServerManager {...} // Orchestratore
dell'intero sistema
```

Gestione dello Stato di Gioco

Stato Distribuito

- Lo stato del gioco viene mantenuto sul server e sincronizzato con i client:

Sistema di Riconnessione e Persistenza

- L'applicazione implementa un meccanismo avanzato per gestire disconnessioni temporanee

Sincronizzazione in Tempo Reale

- L'architettura permette sincronizzazioni immediate tra i client
- Verifica delle soluzioni dei quiz