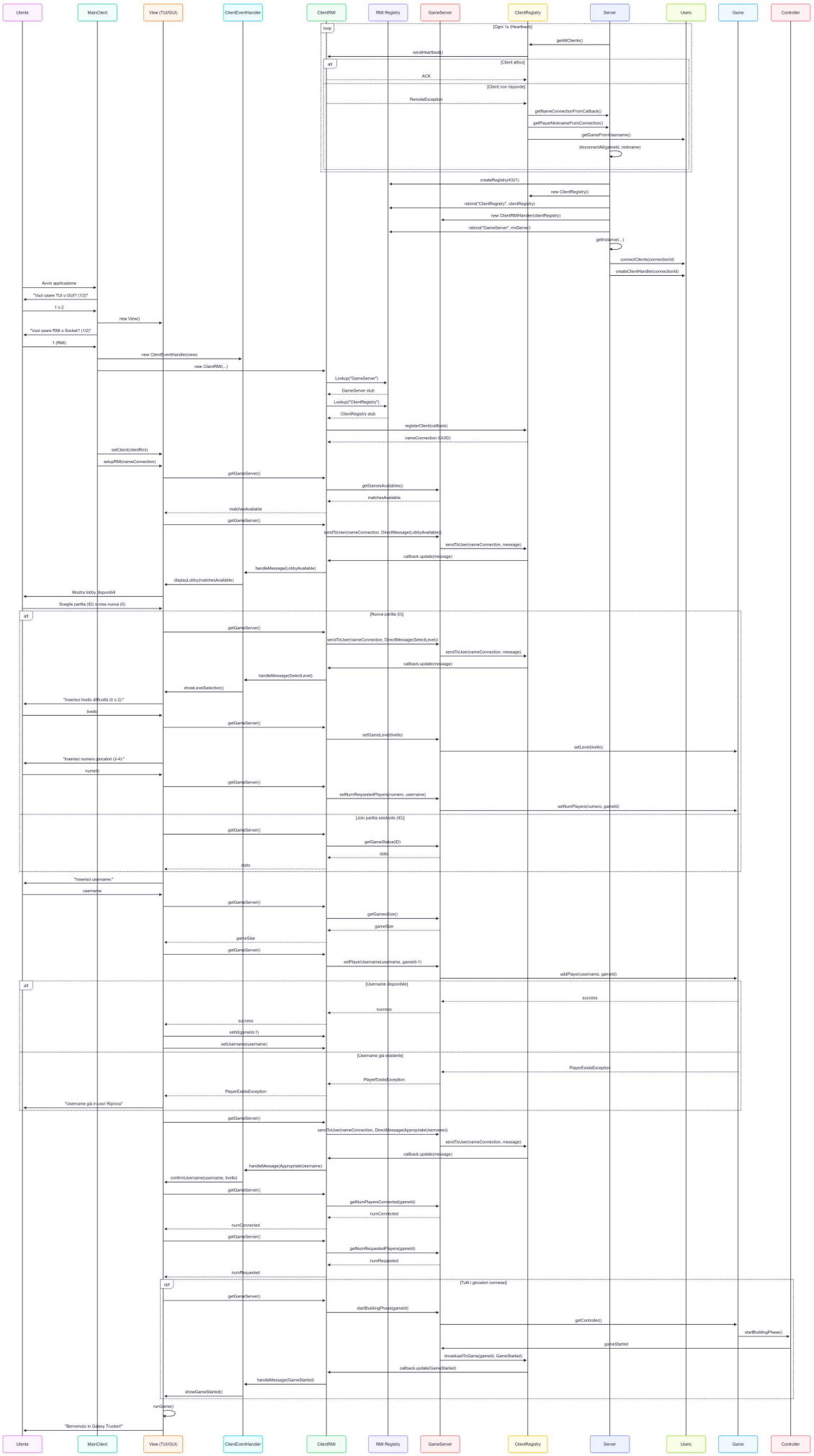
Accesso al gioco di un giocatore con protocollo RMI



Description

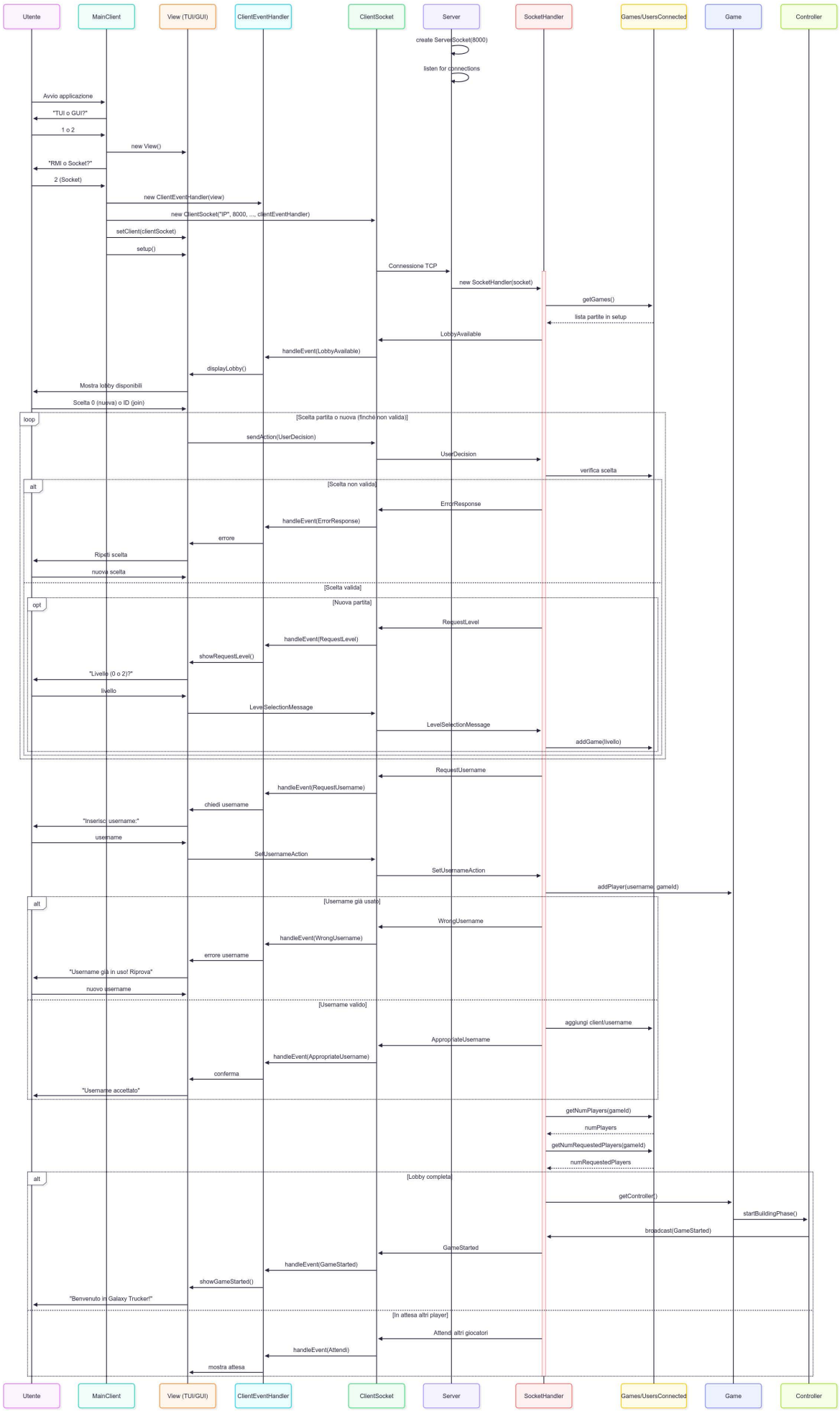
The system implements a multiplayer game where clients can choose either a text-based (TUI) or graphical (GUI) interface and connect to the server using the RMI (Remote Method Invocation) protocol. After starting the client and selecting the interface and connection type, the user interacts with the server, which manages the state of active games and available lobbies.

A key aspect is the heartbeat (ping) mechanism: the server periodically sends heartbeat messages to all connected RMI clients to verify their availability. If a client fails to respond, it is considered disconnected and the server removes it from the game,notifying the other participants.

Upon connecting, the user can choose to join an existing lobby or create a new one. When creating a new game, the user is prompted to select the difficulty level and the number of players. In both scenarios, the client must provide a username that is unique within network; the server checks for uniqueness and, in case of a conflict, asks the user to provide a different name.

All communication between client and server—including callbacks for notifications and status updates—takes place via RMI, and from the user's perspective, these are handled like normal method calls. Network errors, exception handling, and details of the user interface are not depicted in the diagram, which focuses solely on the logical flow of user access and game setup operations.

Accesso al gioco di un giocatore con protocollo Socket



Description

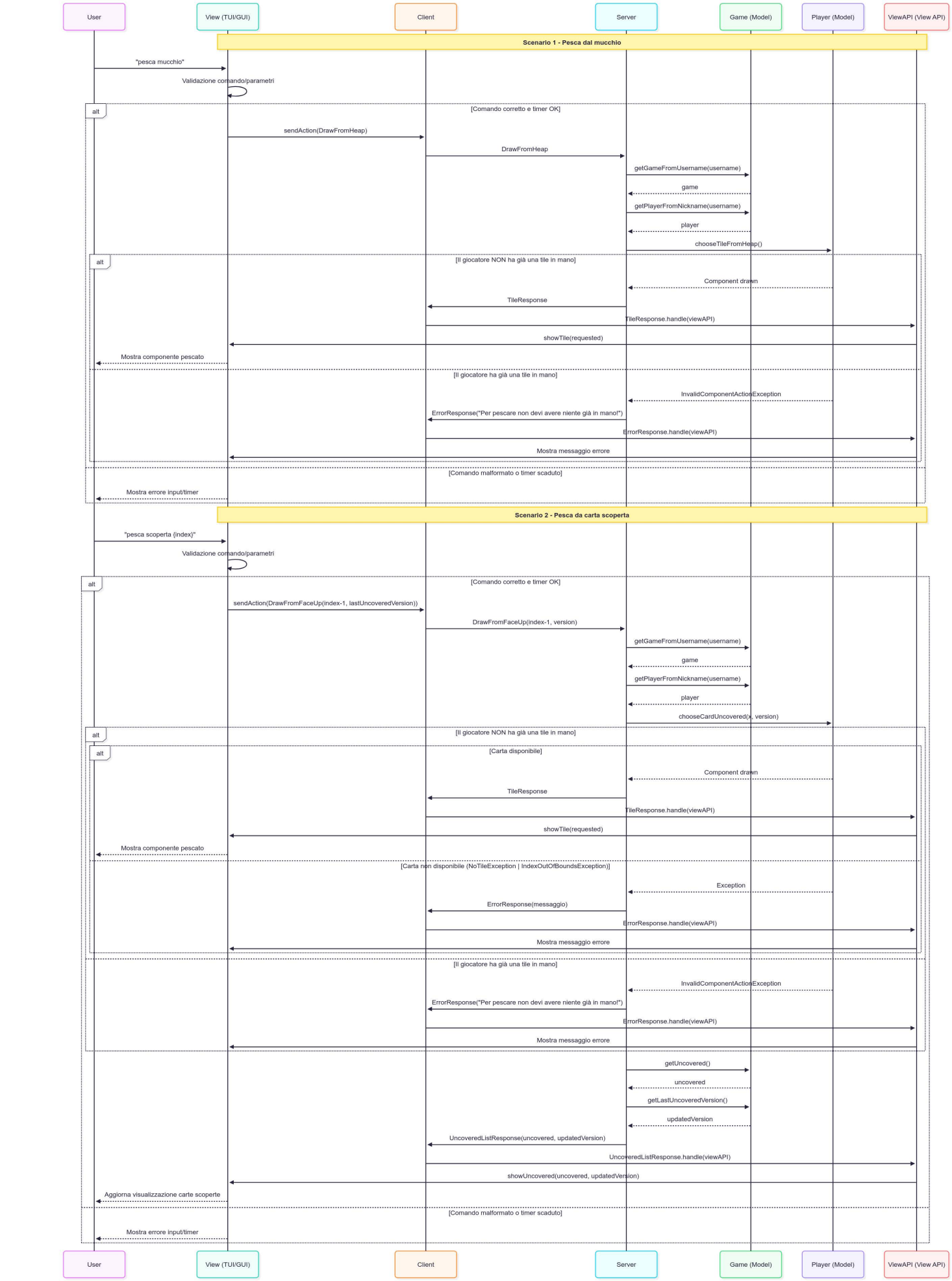
This sequence diagram illustrates the login and lobby access flow for a player connecting to the multiplayer Galaxy Trucker game using the socket protocol, which is an alternative to RMI. After starting the client, the user selects both the user interface type (textual or graphical) and the network protocol, then the client establishes a TCP socket connection to the server.

Upon connection, the server immediately sends the client the list of available lobbies (games not yet started), allowing the user to either join an existing game or create a new one. The client's choice is sent to the server over the socket as a serialized message. If the choice is invalid (for example, trying to join a lobby that no longer exists or is already full), the server notifies the client, and the process repeats until a valid selection is made.

If the player chooses to create a new game, the server requests the difficulty level, and the client responds accordingly. In both cases (join or create), the server asks the client for a username, which must be unique within network. The server validates the username; if it is already taken, the client is prompted to enter a different one, repeating this process until an acceptable username is provided.

After the username is confirmed, the server checks whether the lobby has reached the required number of players. If not, the client is notified and enters a waiting state. Once the lobby is full, the server starts the game by broadcasting a message to all connected clients.

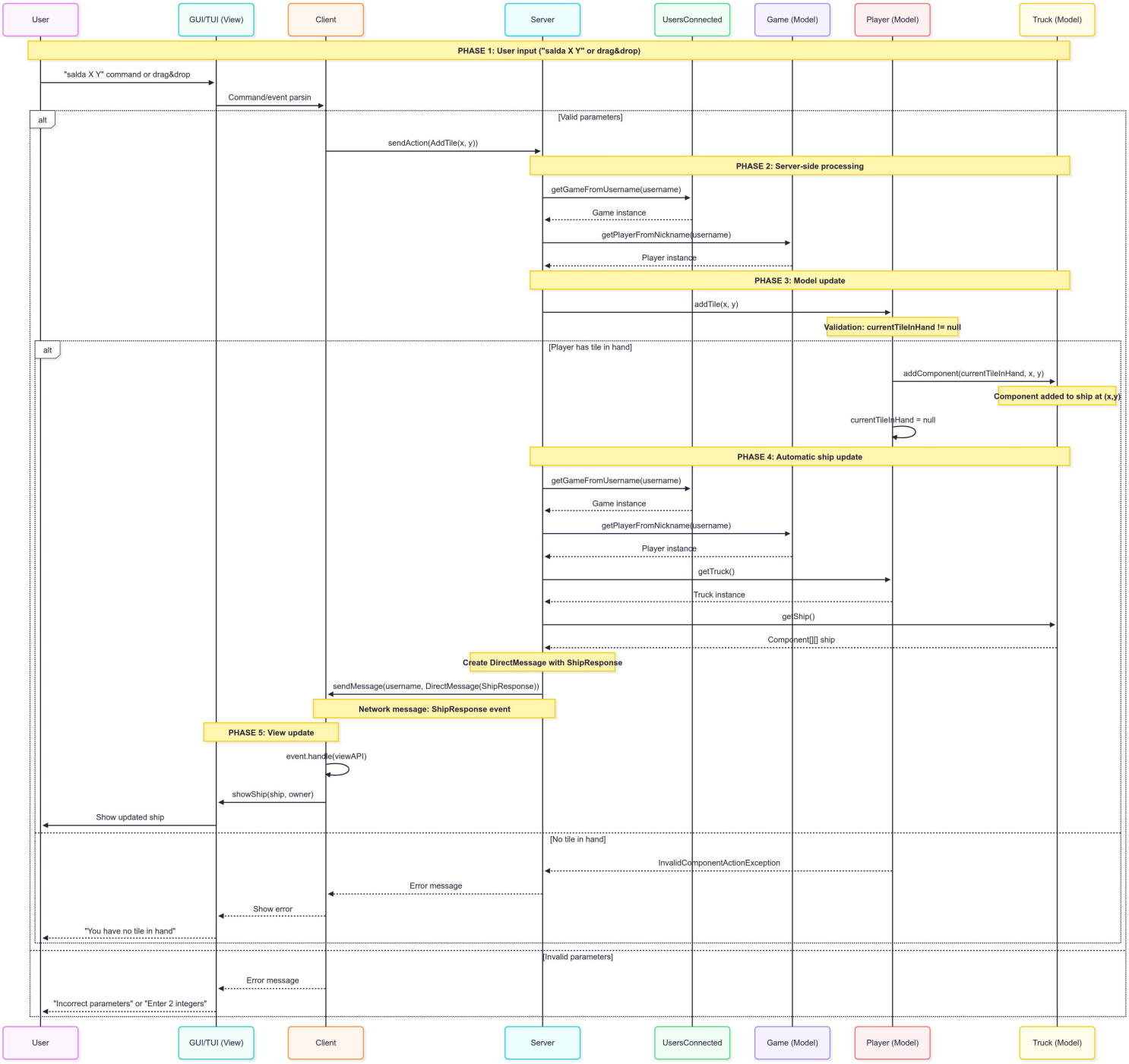
Azione 'pescare una carta componente dal tavolo'



Description

When a player wants to draw a component card, the client sends an action to the server, which processes the request by first checking if the player already holds a tile in their hand. If the hand is empty, the server will either draw a tile from the main heap or from the set of uncovered cards, depending on the player’s request. The selected tile is then set as the player’s current tile in hand and sent back to the client as a response. If the player tries to draw a card while already holding one, the server interrupts the action and sends back an error message. For draws from the face-up cards, the server also sends the updated list of available uncovered cards after every attempt, so the client can keep the display in sync with the game state. On the client side, these responses are managed by updating the interface to show either the newly drawn component or an appropriate error message to the user.

Azione 'agganciare una carta componente'



Description

This sequence diagram describes the process by which a player attaches a component tile to their spaceship, either by entering a command (in the TUI) or by performing a drag-and-drop action (in the GUI). The interaction starts with user input, which is parsed and validated by the client. If the parameters are valid, the client sends an action request to the server, which retrieves the appropriate game and player instances. The server then checks if the player actually holds a tile in hand; if so, the tile is added to the ship at the chosen coordinates and the player’s hand is cleared. The server automatically updates the game state and sends the new ship configuration back to the client, which updates the user interface accordingly. If the player tries to attach a component without holding a tile, or if the input parameters are invalid, an error message is displayed to the user. The diagram clearly illustrates the division of responsibilities between the View (GUI/TUI), Controller (Client/Server), and Model (game state and entities).