Sequence Diagram

Communication Protocol

List of all messages exchanged between client and server, divided in those sent by the server and those sent by the client. We decided to implement in the communication protocol two advanced functionalities: chat and multigame. Note that all updates are meant to be received asynchronously by the client.

Server messages:

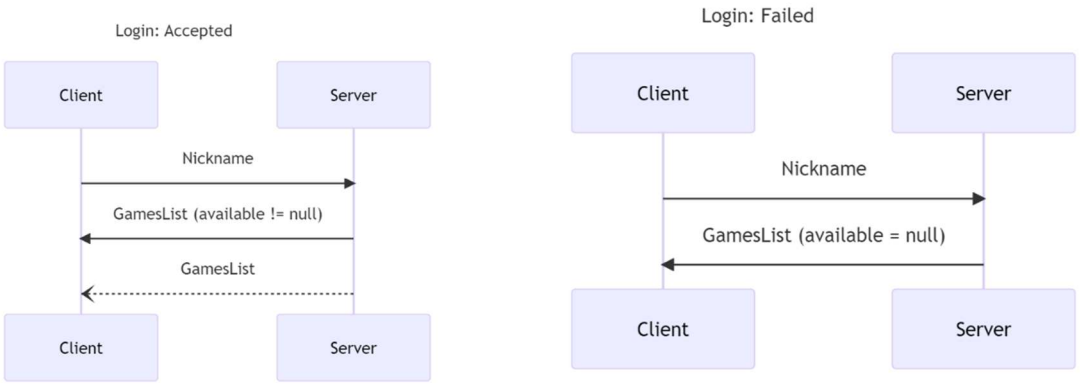
AcceptedInsertion	`{accepted : boolean}`
BookshelfUpdate	`{owner : String, bookshelf : Map<Position, Item>}`
ChatAccepted	`{accepted : boolean}`
ChatUpdate	`{sender : String, text : String, receiver : String}`
CommonGoalCardsUpdate	`{cards : Map<Integer, Integer>}`
EndGameUpdate	`{winner : String}`
EndingTokenUpdate	`{owner : String}`
GameData	`{numberPlayers : int, connectedPlayers : Collection<String>, numberCommonGoalCards : int, livingRoomRows : int, livingRoomColumns : int, bookshelvesRows : int, bookshelvesColumns : int}`
GamesList	`{available : [{id : int, numberPlayers : int, numberCommonGoals : int}]}`
LivingRoomUpdate	`{livingRoom : Map<Position, Item>}`
PersonalGoalCardUpdate	`{id : int}`
ScoresUpdate	`{scores : Map<String, Integer>}`
SelectedItems	`{items : List<Item>}`
StartTurnUpdate	`{currentPlayer : String}`
WaitingUpdate	`{nickname : String, missing : int, isConnected : boolean}`
Reconnection	`{reconnectedPlayer : String}`
Disconnection	`{disconnectedPlayer : String}`

Client messages:

BookshelfInsertion	`{column : int, permutation : List<Integer>}`
ChatMessage	`{text : String, receiver : String}`
GameInitialization	`{numberPlayers : int, numberCommonGoalCards : int}`
GameSelection	`{id : int}`
LivingRoomSelection	`{positions : List<Position>}`
Nickname	`{nickname : String}`

Login

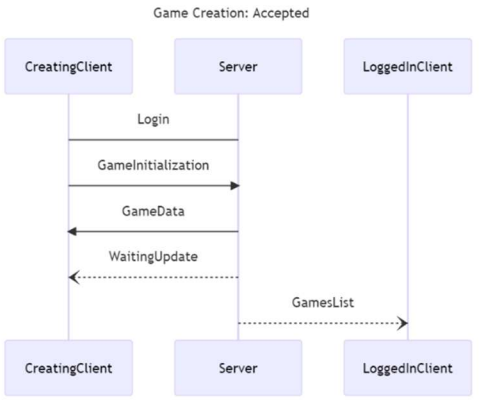
The login is necessary for the client to perform any other type of action. The client sends a Nickname message to the server: if the nickname provided has not already been chosen by another logged in client, the client is logged in; else, the login attempt is rejected, and the client has to try again.

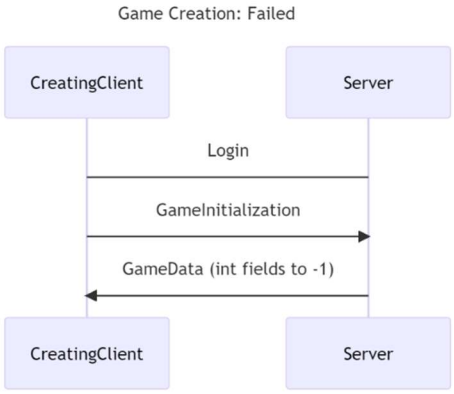


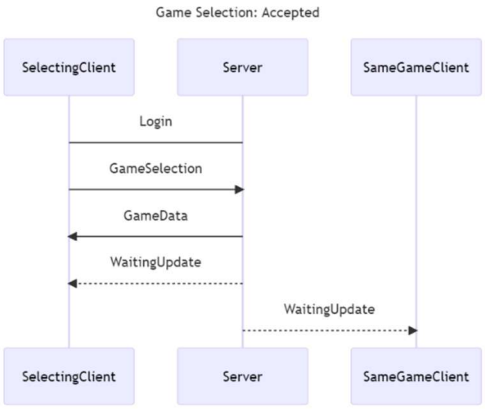
Game Creation/Selection

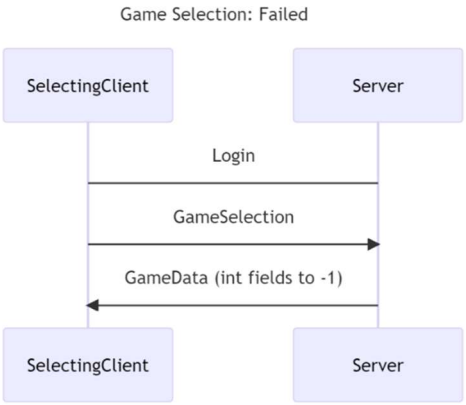
After the Login, the client can choose either to create a new game with the desired parameters or select an already existing game. If the client fails to set correctly parameters for game creation or selects an invalid game id, the server will notify the client that will need to try again.

Note: if a game that had not yet started is deleted by the server, all clients not in game are notified with GamesList message with `available` containing only one element with the id of the deleted game and all other fields to -1.









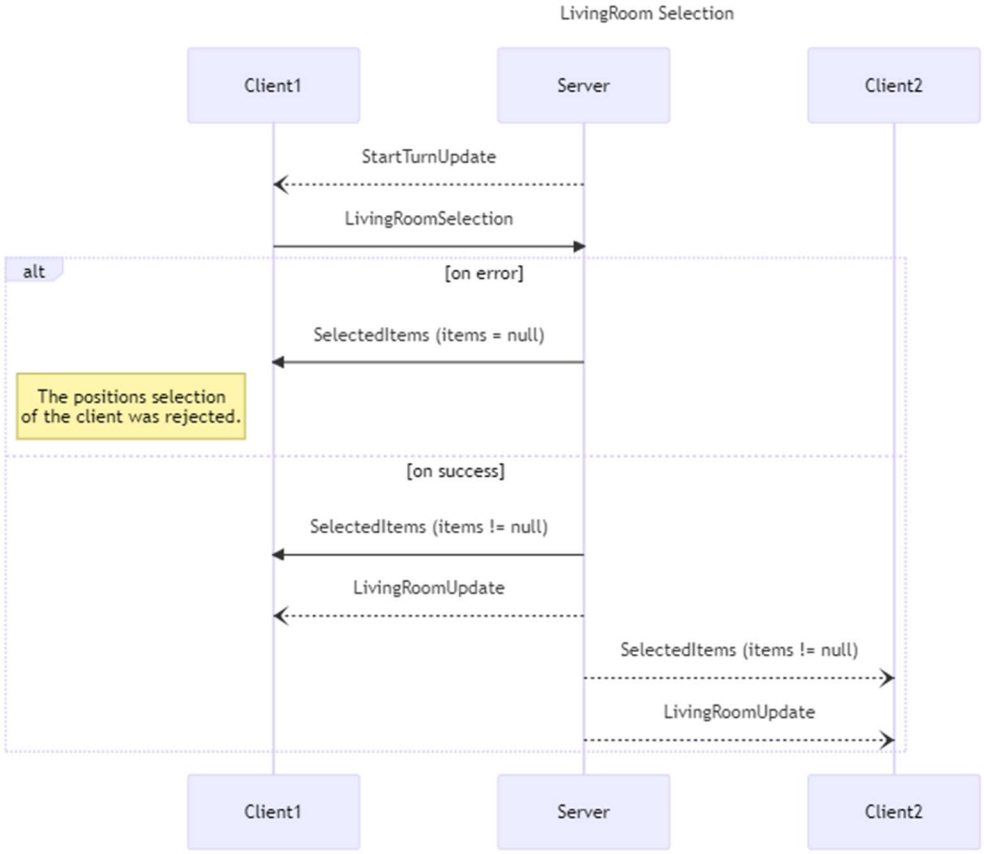
Game Beginning

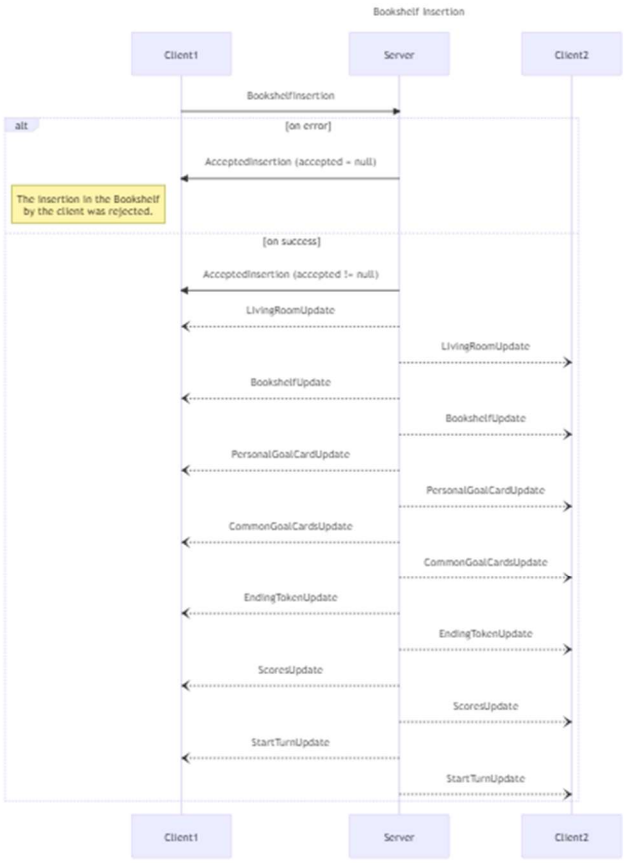
At the beginning of a game, all clients involved in the game receive asynchronously the following updates: BookshelfUpdate, CommonGoalCardsUpdate, EndingTokenUpdate, LivingRoomUpdate, PersonalGoalCardUpdate, ScoresUpdate, StartTurnUpdate.

Turn Phases

First, the server notifies the start of turn, then the current client selects the tiles in the Livingroom. On success the client needs to select the column and the order of the selected tiles to put in the Bookshelf. On success the server starts the next turn. Note: for the sake of clarity, the following

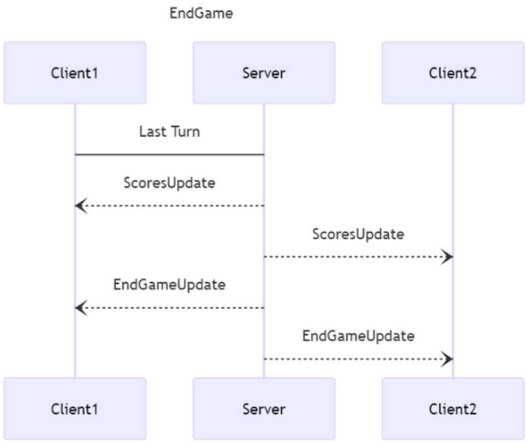
sequence diagrams show only two clients.

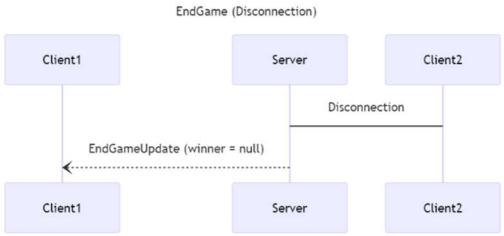




Endgame

All clients receive a ScoresUpdate message, notifying the final scores, and an EndGameUpdate with the nickname of winner. The EndGameUpdate message (with a null winner) is also used to signal to the clients that the game ended because one of the clients disconnected. Once an EndGameUpdate is sent to the clients, no other message will be sent and the server will not accept any other message.

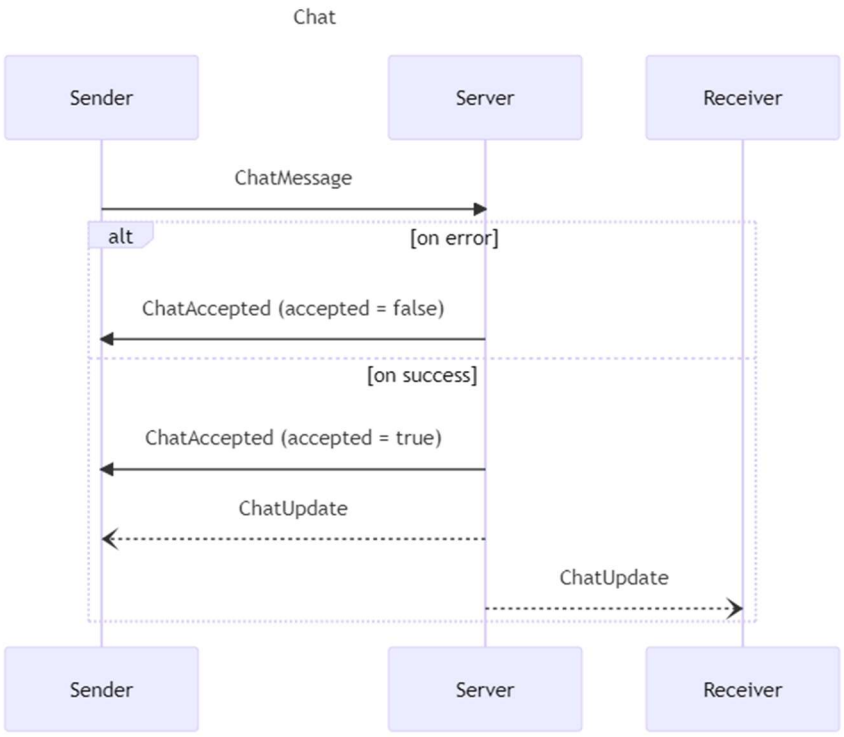




Chat

The chat messages can either be broadcast or unicast: the client sending the message specifies either the nickname of another client for unicast messages or “all” for broadcast messages as receiver in ChatMessage. After a ChatMessage is sent, the client will receive a ChatAccepted

message, and if the message was sent successfully, all involved clients will receive asynchronously a ChatUpdate.



Disconnection & Reconnection

When a disconnection occurs in game, all clients in that game get notified of the disconnection through a Disconnection message sent by the server with the nickname of the player that just disconnected. When a client connects to the Lobby with the same nickname of a player that disconnected from a game (still on going), the client is joined into that game and everybody (together with the client that just reconnected) receives a Reconnection message bearing the name of the reconnected player.