Introduction

In this document, we describe the communication protocol used in our implementation of “Maestri del Rinascimento”. We decided to use JSON (the GSON library) to serialize the messages and send them as string in the input and output streams. We made this choice to make the debugging of the messages sent from client to server and vice versa easier for us.
The decision to use a thin client approach has led us to develop server messages to the client in order to start each phase of the game, in addition to the client request and server response messages.
As required our server is a multi-client server, so each client is managed by a dedicated thread. We also decided to implement the advanced feature “Multiple games”: to do this we designed a server based on lobbies (GameHandlers), each managing a different game (from 1 to 4 players). When a user connects to the server, he is addressed by the lobbies’ manager to the first free lobby (if he chose to play a multiplayer game, otherwise a single player lobby is created).
The first parameter of the message represents its type:

- “Setup”: this type of message is sent by the client at the beginning, once he connects to the server. The client sends in this message the username chosen and the preferred number of players (if 1, a single player game will start, otherwise the player will be added to the current lobby for a multiplayer game).
- “Disconnect”: a DisconnectionMessage is sent by the client when he wants to disconnect from the server.
- “Action”: the action message is a message containing the info about a specific game action (“buy”, “produce”, etc). This message is sent by the client when he wants to do a game action.
- “Update/Answer”: this message is sent by the server to the client(s) when there has been a change to the game (new client connected, client disconnected, change in the model, etc).

In the next chapters we will describe in detail each different communication phase.

Setup Connection

When a client connects to the server, the latter will instantiate a ClientHandler which will be in charge of sending and receiving messages from the client. The client will then send a message to the server in which it specifies his nickname and the preferred number of players (a number between 1 and 4). The server validates the received message: if there is something wrong (typos or duplicate nickname), it will send back a ErrorAnswer message containing the error and closes the connection with the client. If everything goes fine, it sends back a StartedAnswer message to the client. If the player selected sologame (by sending a number of players equal to 1) the game will immediately start, otherwise the client will wait until the number of players is reached.

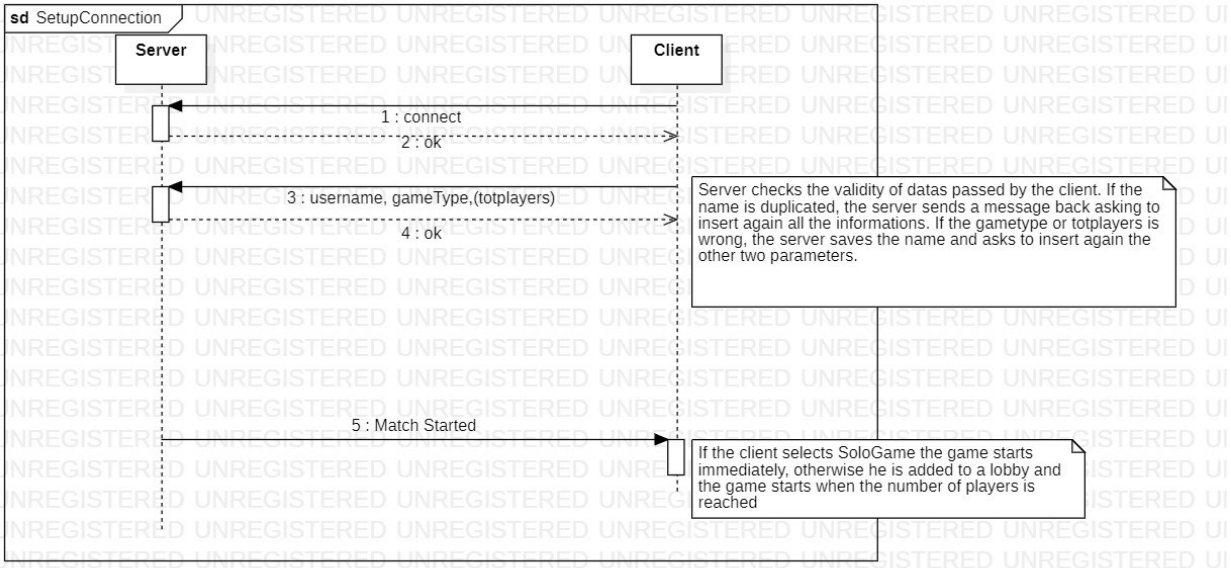

Figure 1 SetupConnection sequence diagram

Leader Phase

Once the last client for the current game has connected to the server, the match will start. First of all, the server sends to each client a ChooseLeadersAnswer message, containing the initial state of the Develop Decks and of the Market, then it sends to each client (one at a time, clockwise) an update message with the 4 Leader Cards from which the player must choose 2 and asks the client to choose his 2 leaders. Once the client sends back the indexes of the chosen leaders (two numbers between 1 and 4), the server validates the received message: if there is a problem (the values for the indexes are out of bounds, for example "10" is out of bounds) the server sends back an ErrorAnswer message asking the client to insert again the indexes; if everything goes fine, the server sends a OkLeadersAnswer message to the client sending him the cards he chose, then sends to the next player the update message containing his 4 Leader Cards and he has to do the same actions as the previous client. Once every client in the game correctly chose his leaders, the game evolves in the Resource Phase.

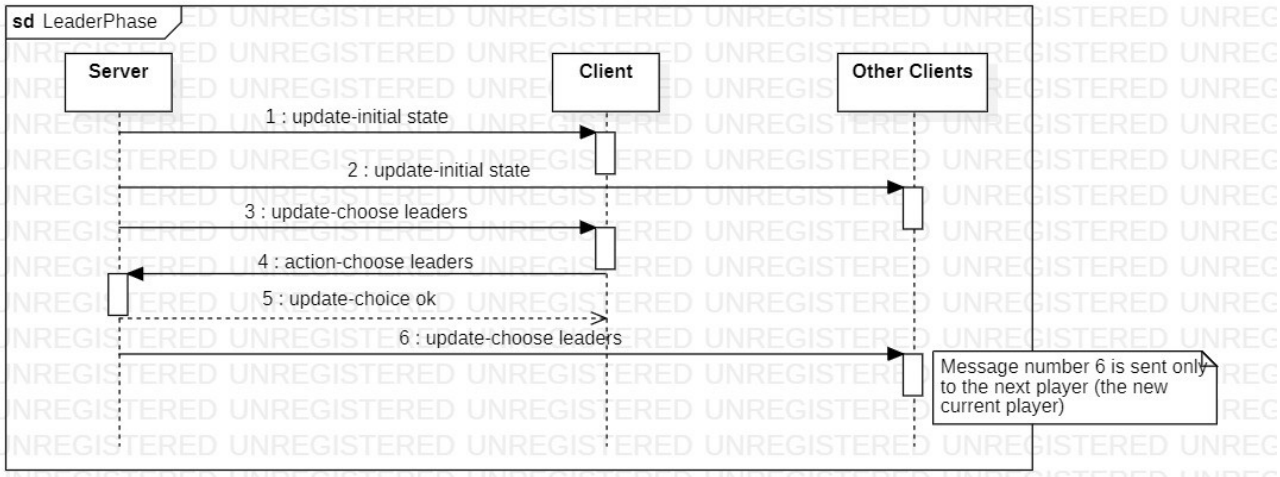

Figure 2 Leader Phase sequence diagram

Resource Phase

The server sends ChooseResourcesAnswer to each player that has the right to choose an initial resource, asking him which color he wants and where he wants to put it in. The player has to reply with the following syntax:

- “resX”: the player types the color that he chooses for the resource X.
- “posX”: the player types the deposit [“small”, “mid”, “big”, “sp1”, “sp2”] that he chooses to put the resource X in.

The server validates the received message: if there is a problem, the server sends back an ErrorAnswer message asking the client to insert the infos again; if everything goes fine, the server sends an OkResourcesAnswer message to the client showing him the chosen resources and, eventually, his new increased position on the Faith Track. Then it goes to the next player. Once every client in the game has correctly chosen his initial resources, the Initial Phase ends.

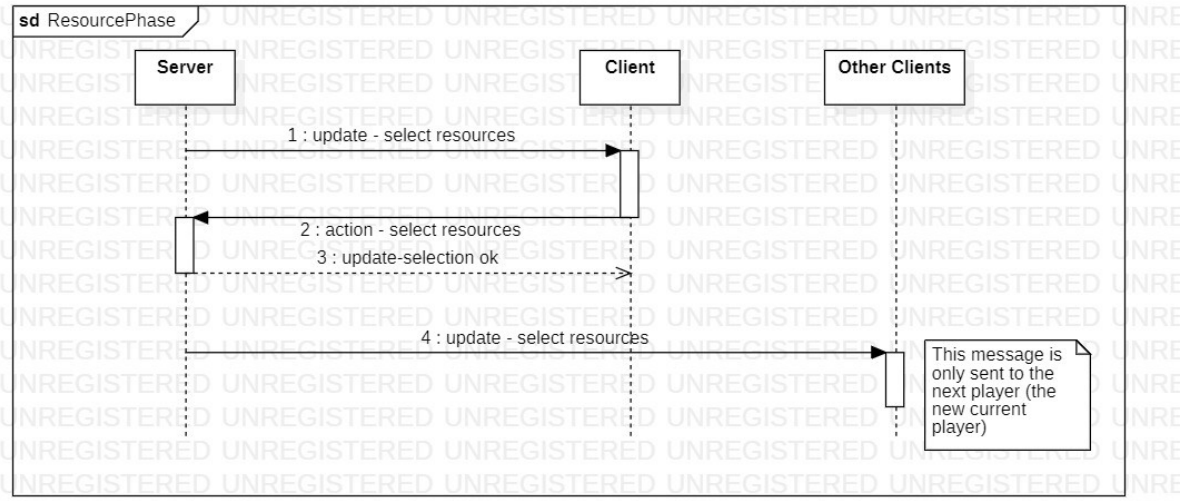

Figure 3 Resource Phase sequence diagram

Full Game Phase: activate/discard leader

When the current player decides to activate or discard one of his leaders, the client sends a LeaderActionMessage message to the server containing the type of action he wants to do (“activate” or “discard” in this case) and the index of the selected leader (0 or 1). The server validates the message received: if there is a problem (such as index out of bounds) he sends back to the client an ErrorAnswer message asking him to do the action again (or to do another action); if everything goes fine, he sends back a LeaderActionAnswer message containing the new state for the selected leader and the new value of the player’s position on the Faith Track (if the action was “discard”). If the activated leader is a deposit leader or a production leader, the client changes his view according to the leader type (adds a deposit or a new production).

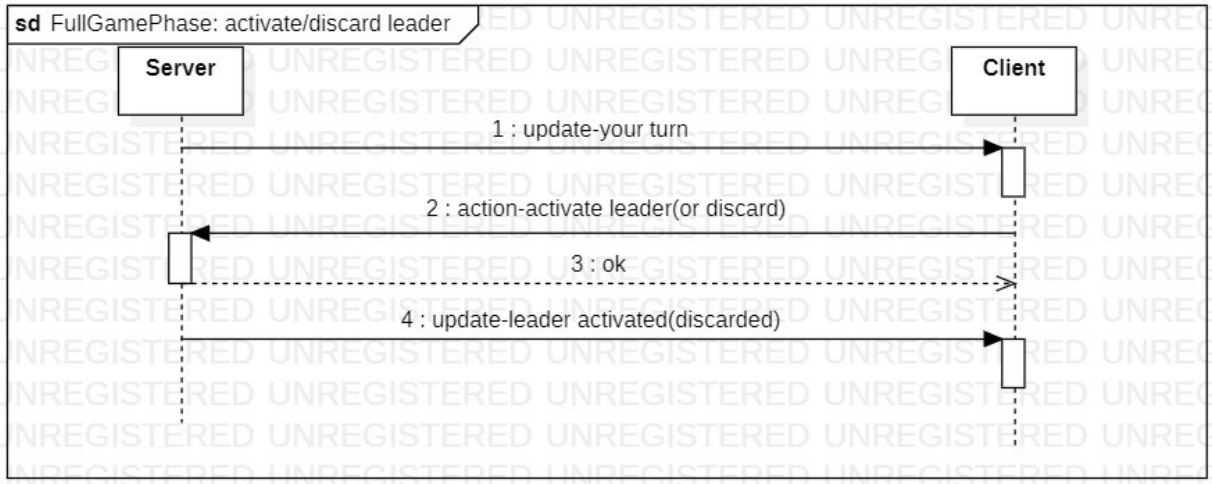

Figure 4 Full Game Phase: Activate/Discard Leader sequence diagram

Full Game Phase: Produce

When the current player decides to activate the production, the client sends a ProductionMessage message to the server containing all the informations about the productions the player wants to activate. These informations are:

- “prodX”: for each production the player has (“X” is the number of the production), he sends a value (“yes” or “no”) indicating if he wants to activate it or not.
- “posXY”: for each input resource (“Y”) requested by the production the player wants to activate (“X”), the player sends a value indicating where to take the resource (correct values are: “small”, “mid”, “big”, “sp1”/“sp2” when he has active deposit leaders, “strongbox”).
- “in01”, “in02”: for the first production (the base one, if the player wants to activate it), the player also sends the type of resource he wants to use as input.
- “outX”: for the first (the base production) and the special productions (if a production leader is active), if the player wants to activate them, he sends the type of resource he wants to produce.

The server validates the message received and tries to update the model consequently: if everything goes fine, the server sends back an ProduceAnswer message sending the new state of the deposits and strongbox; if there is a problem (the player does not have a production he wants to activate, there is no active leader that gives a special production the player wants to activate or some resource is not in the place indicated by the player) the server sends back an ErrorAnswer message telling what went wrong and asking the client to do the action again (or to do another one).

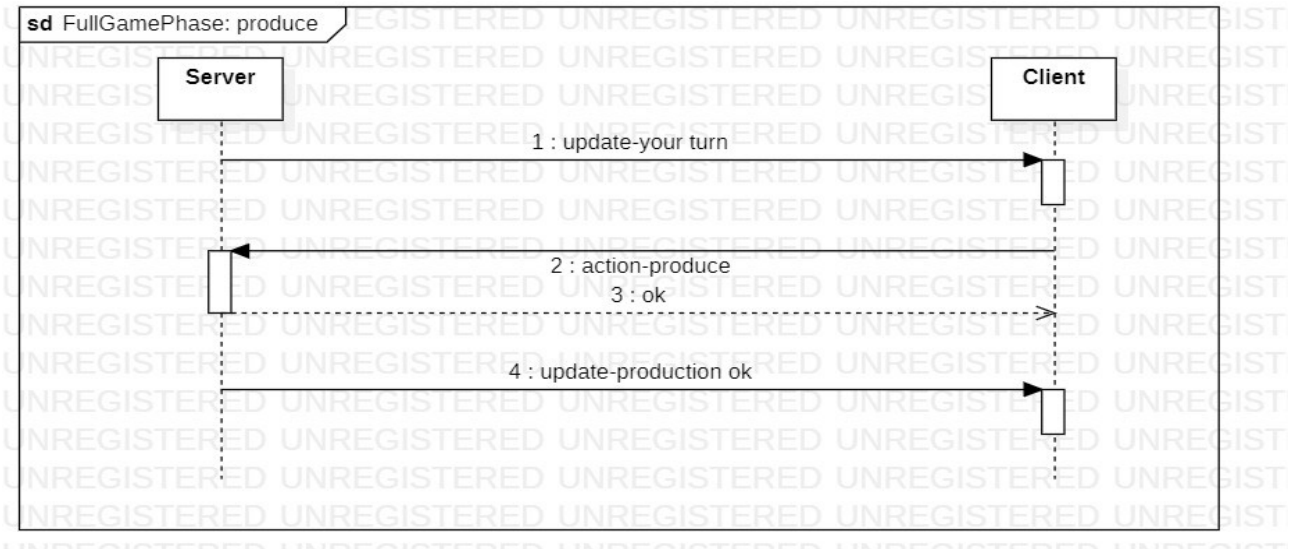

Figure 5 Full Game Phase: Produce sequence diagram

Full Game Phase: Market

When one of the clients makes a fromMarket action, it sends a MarketMessage message to the server containing the following information:

- “row” / “col”: the player types the index of the row [1-3] or column [1-4] that he chooses.
- “resX”: the player types the deposit [“small”, “mid”, “big”, “sp1”, “sp2”] that he chooses to put the resource X in.

The server replies with a MarketAnswer or an ErrorAnswer message and then. In the first case, it updates all the clients on the new situation of the market and, eventually, on their new position in the Faith Track. In the latter case, the player can redo the action.

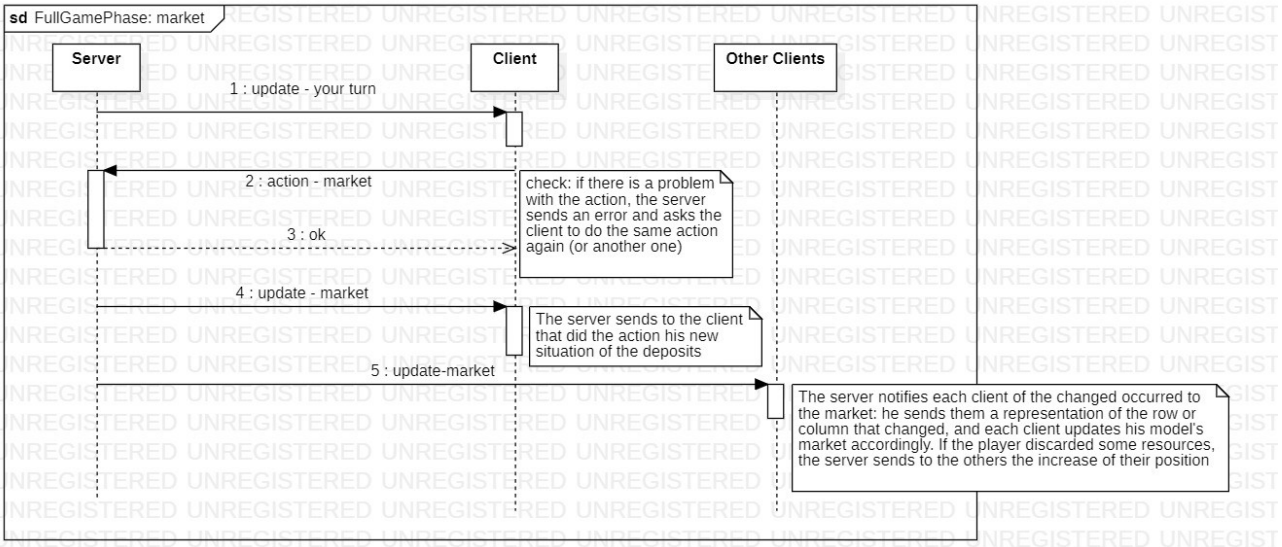

Figure 6 Full Game Phase: Market sequence diagram

Full Game Phase: Buy

When the current player decides to buy a develop card, the client sends a BuyMessage message to the server containing all the information about the buy move. It includes:

- "row": a number between 0 and 2, is the row of the develop deck in which the card is located.
- "column": a number between 0 and 3, is the column of the develop deck in which the card is located.
- "ind": a number between 0 and 2, is the player's personal board's slot in which the develop card will be positioned.
- "resX": for each resource needed for buying the develop card, the player will specify from where it has to be taken (big, small, mid, sp1, sp2, strongbox).

The server validates the received message: if there is something wrong (for example a typo) it will send the client an ErrorAnswer message asking him to repeat the action (or to do another one). If everything goes fine, it sends an BuyAnswer message just to the current player's client, updating him the new state of his deposits and strongbox. Then it sends to each client a BuyAnswer message containing the new state of that specific develop deck.

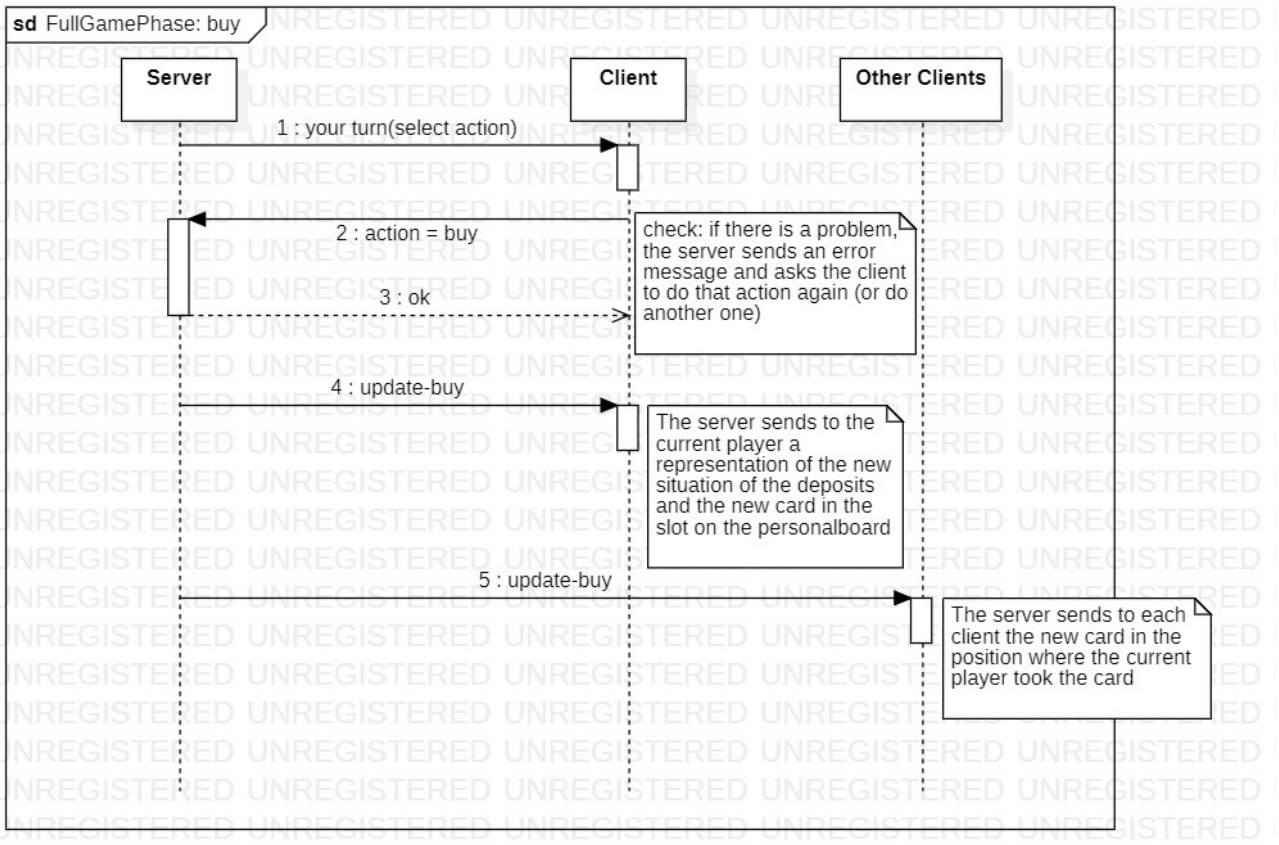

Figure 7 Full Game Phase: Buy sequence diagram

Full Game Phase: swapDeposits

When the current player decides to swap the resources of two deposits, the client sends a SwapMessage message to the server, containing the “source” and “destination” which are strings that represent the deposits a player wants to swap (small, medium, big, sp1, sp2). The server validates the received message: if there is something wrong (for example a typo) it will send the client an ErrorAnswer message asking him to repeat the action (or to do another one). If everything goes fine, it will send the client a SwapAnswer message containing the new state of the player’s deposits.

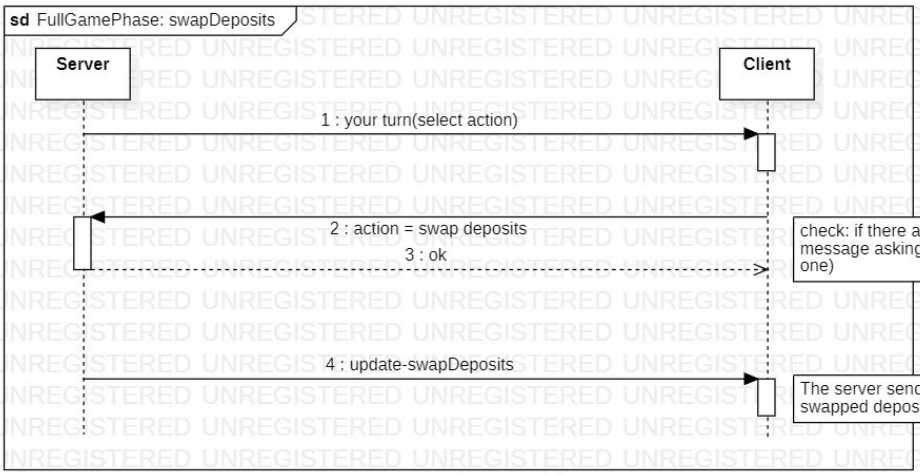

Figure 8 Full Game Phase: SwapDeposits sequence diagram

End Turn

When one of the users ends his turn, it sends an EndTurnMessage message to the server. The server checks if the player has done the mandatory action in his turn. If he has, then it replies with an EndTurnAnswer to each player, otherwise with an ErrorAnswer message to the current player. The EndTurnAnswer message contains information about each player's Favor Tile, and it updates him on the new value of the that. It also contains the name of the player that ended the turn and the name of the new current player. After this operation, the next player to have to play his turn is warned.

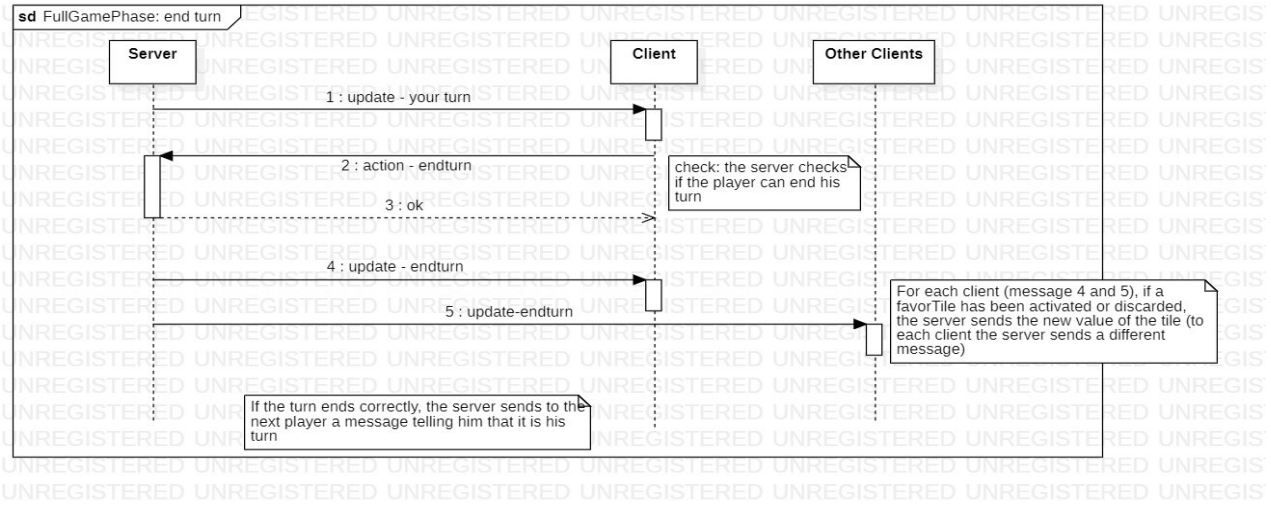

Figure 9 End Turn sequence diagram

End Game Phase: endgame

When a player correctly buys his seventh develop card or reaches the end of the faith track, the game phase will be changed to isEndgame. All players will then play their last turn and once the last player ends his turn the game will end, and the winner will be selected. The server will then send a EndGameAnswer message to each client telling them if they won or lost and closes the connection with each of them.

For the SoloGame: When the player buys the seventh card or reaches the end of the Faith Track, the server sends him an EndGameAnswer message telling him that he won and the number of points he made. Instead, if the black cross reaches the end of the Faith Track before the player or a column of the develop decks is completely empty, the server sends to the player an endgame message telling him that he lost and the number of points he made.

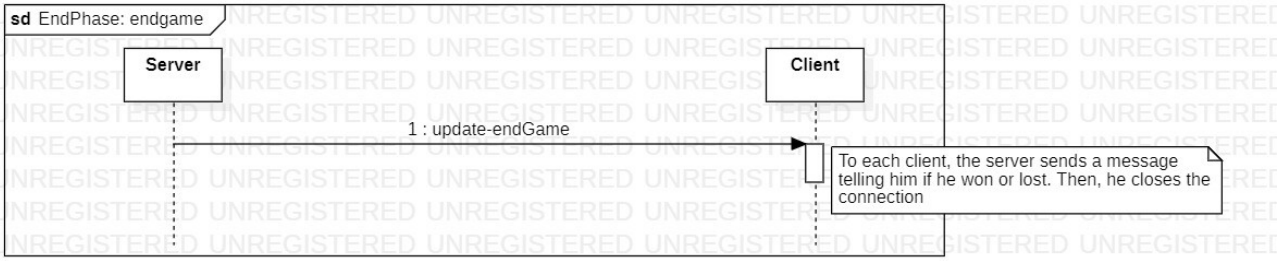

Figure 10 End Game sequence diagram