COMMUNICATION BETWEEN WEB APP AND CLIENT APP

1) PLAYER INITIALIZATION

a) for the creation of a new player, insert the username

[POST request to the server, using the username as parameter, the server will return the player's ID. Key: 'name'] url = PlayerInit/

b) player's name update

[POST request to the server, using as parameter the new username and the player's ID, the server will return an 'OK'. Keys: 'name', 'id'] url = PlayerUpdate/

2) LOBBY MANAGEMENT

a) to create a new lobby and being part of it insert new lobby's name

[POST request to the server using as parameters the lobby's name and the player's ID. Lobby's name will be a primary key in the database. The server will return an 'NOK' if lobby's name already exists, 'OK' otherwise. If the player is already in another lobby, will be removed from it. Keys: 'name', 'id'] url = CreateLobby/

b) join existing lobby (url = JoinLobby/ (for both))

- scroll existing lobbies

[GET request to the server (no parameters), the server will return a list with the existing lobbies (JSON)]

#possibility to add a search based on initial letters

- join lobby

[POST request to the server using as parameters lobby's name and player's ID. The server will return an 'OK'. If the player is already in another lobby, will be removed from it. Keys: 'name', 'id']

3) GAME CREATION

To create a new game chose the number of checkpoint for every single player, the play range, and the start coordinates.

[POST request to the server using as parameters the number of checkpoints, the play range, start coordinates, the difficulty and the lobby the player belongs to. The server will return the maximum number of players for the created game. Keys: 'lobby', 'first' (number of checkpoints), 'second' (play range), 'third' (start coordinates as a string), 'fourth' (difficulty).] url = SearchForCheckpoints/

#the maximum number of players depends on the fact that for every player must be available 1,5 times the number of checkpoints chosen by the game's creator, hence the application can manage which checkpoint provide on the fly, so the number of checkpoints in the play range must be enough to satisfy all requests.

4) GET CHECKPOINTS

press a certain button to get checkpoints

[POST request to the server using as param the player's ID. The server will return a number of randomly choosen checkpoints equals to N\*2 where N is the selected number of checkpoints in the creation of the game (JSON). Key: 'id']

url = GetCheckpoints/

#possibility to add a function returning an error when the maximum number of players is reached, in case the participant players add on the fly

5)RANKING

[POST request to the server using as params the player's ID and the number of checkpoints completed. The server returns a list of tuples, containing the players participating to the game and the number of checkpoints completed, sorted by the incrising number of checkpoints completed; where the number is the same, they're alphabetically ordered (JSON). Keys: 'id', 'checked'] url = Ranking/

6)GAME STATUS

[POST request with the player's ID. Returns the status of the game the player is associated with, as True if it's in progress, False if ended or not even started. Key: 'id'] url = GameStatus/

7) CHECK TIME

[POST request with the player's ID as param. Returns the time delta since the beginning of the game in the following format h:m:s.us. Key: 'id'] url = CheckTime/

8) BEGIN/FINISH GAME

[POST request using as parameters the player's ID and a boolean variable which is False if the player has completed his checkpoints. In this case every player finishing the game can use this function. On the other hand the same function called with True as parameter value should be a prerogative of the creator, which in this way starts the game. The server return an 'OK'. Keys: 'yes\_no', 'id'] url = BeginFinish/

+ URLs are meant to be appended to http://proper.router.ip:8000/