Game Server Documentation

The game server is a simple REST-like server. It has little opinion on games, but just stores the information. The client is allowed to add any properties it deems useful to both user and game information and the server will store and retrieve them faithfully.

Installing and running

The server code is at https://github.com/olehougaard/gameserver. Download the server code and run

```
npm install
```

You run the server with

```
npm start
```

The server stores the data in the file `data/data.json`. The file `data/default.json` is provided as a way of restoring the database.

Calling the server

The server listens for HTTP on port 9090.

All data in request and response bodies are in JSON format.

Creating users

POST /users

Creates a new user with the properties in the request body. The properties need to include username and password.

The user will receive and id and an admin Boolean property (default: false). Any id and admin properties in the request body will be overwritten.

Returns:

- 201 with the created user in the response body if the operations succeeded
- 400 if username and password are missing or are not strings, or if the username already exists.

Login and logout

POST /login

Attempts to log the user in with the username and password in the request body.

Returns:

- 200 with an object with token and userId properties if the login succeeded
- 403 if the login didn't succeed

POST /logout?token=<token>

Attempts to log out a logged in user. The token in the query must have been returned from a log in.

Returns:

- 200 if the log out succeeded
- 403 if the token is not from a current session

Other services

All other services require the user to be logged in and send a token in the request parameters (like for logout).

GET /users

For test purposes only. Only admins can access.

Returns:

- 200 with a list of all users
- 403 if unauthorized

GET /users/<id>

Returns the details of the user. Must be logged in as that user or admin.

Returns:

- 200 with the details of the user
- 403 if unauthorized

PATCH /users/<id>

Updates the user with the properties given in the request body. Any properties id, username, and admin will be ignored. Must be logged in as that user or admin.

Returns:

- 200 if successful
- 403 if unauthorized

GET /games

Returns a list of all games. Must be logged in.

Returns:

- 200 with a list of all games if successful
- 403 if unauthorized

POST /games

Starts a new game for the logged in user. The new game has the following properties:

- id: The id of the game
- user: The user id of the user playing the game
- score: The user score of the game (initially 0)
- completed: A Boolean indicating whether the game has been completed (initially false)

The request body is ignored.

Returns:

- 201 with the details of the new game if successful
- 403 if unauthorized

GET /games/<id>

Returns the details of the game. Must be logged in as the player.

Returns:

- 200 with the details of the game
- 403 if unauthorized

PATCH /games/<id>

Updates the game with the properties given in the request body. Any properties user and id will be ignored. Must be logged in as the player.

Returns:

- 200 if successful
- 403 if unauthorized