Assignment 4: Blanket Fort Game REST API

1. General

- All ID's are assigned by the back-end and are implementation specific. Therefore, you should not worry about trying to match the IDs that appear in the sample URLs below. The UI will adjust to work with whatever IDs you pass it.
- All commands return HTTP 200 (OK) unless stated otherwise.
- Any end-point accepting an ID (be it in the path, query string, or in the body) must return an HTTP 404 error if the ID does not exist.

GET /api/about

- Return your name as a string.

2. Games

GET /api/games

- Return a list of `ApiGameDTO` objects for the games that the system knows about. Note that once a game is created, the server does not need to ever delete that game: it can exist until the server is restarted.
- Each `ApiGameDTO` stores an ID to uniquely identify the game. This ID is assigned by the backend.
  - *TIP: Make the ID the game's index into the list which you use to store the games.*

POST /api/games

- Create a new game with 5 opponentss by POSTing to this end point. You need not post any data (no body).
- Returns one fully populated `ApiGameDTO` object instance.
- Returns HTTP status code 201 (Created) when successful.

GET /api/games/1

- Return `ApiGameDTO` object for game 1 (where 1 is the game ID).
- Error Handling:
  - Return 404 (File Not Found) if the requested game does not exist.

3. Board

GET /api/games/1/board

- Return the current state of the board as one `ApiBoardDTO` object (in JSON) for game 1 (change to the ID you need).
- See `ApiBoardDTO` file for field info.
- Error Handling:
  - Return 404 (File Not Found) if the requested game does not exist.
- Strings for 2D `cellState` array, used as `cellState[row][col]`:
  - "fog" – Fog: an unexplored cell
  - "hit" – Hit the fort: shows a red explosion
  - "fort" – Fort; shown when revealed the board
  - "miss" – A missed shot; shows muddy ground
  - "field" – Open field (when the game board is revealed)

Moves

POST /api/games/1/moves

- Make a move in game 1 (change to the ID you need).
- Body of POST message must be an `ApiLocationDTO` object stating which cell location the user is firing at. Cell row and column indexes start at 0.
- By posting here, the user makes their move which causes the back-end to perform the opponents firings.
- No data is returned by this request (empty body): client expected to query other endpoints to find out board state after a move has been made.
- Returns HTTP status code 202 (Accepted) when successful.
- Error Handling:
  - Return 404 (File Not Found) if the requested game does not exist.
  - Return 400 (Bad Request) if the location is invalid, such as shoot at such as (-1,0) or (3, 10).

Cheats

POST /api/games/1/cheatstate

- Activate a cheat state for the game.
- Body of POST message must be:
  - “SHOW_ALL”: Permanently change the game to make all cells visible for this game (no fog).
- Returns HTTP status code 202 (Accepted) when successful.
- Error Handling:
  - Return 404 (File Not Found) if the requested game does not exist.
  - Return 400 (Bad Request) if the body of the message is not the valid cheat string (“SHOW_ALL”).