Battleship API

Creating a new game

`POST /new`
Creates a new game of battleship.

Example Response (201)

```
{
    "gameId": "dq5lqs",
    "playerId": "3ah310",
}
```

HTTP Status Codes

Status Code	Description
201	The game has successfully been created

Joining an existing game

`POST /join`
Joins an existing game of battleship.

Query Parameters

`gameId` (string)

The id of the game to be joined.

Example Response (200)

```
{
    "playerId": "e4ublr",
}
```

HTTP Status Codes

Status Code	Description
200	The game has successfully been joined
403	The game is already full
404	No game with matching ID found

Setting battleships and starting the game

`POST /ready`
If game status is `pendingStart` (see status), places the battleships on the board and readies the server to begin the game.

Query Parameters

`gameId` (string)

The id of the active game.

`playerId` (string)

The id of the requesting player.

Body Parameters

`battleships` (object)

The starting configuration of battleships. See `/battleshipTemplate` for the specific schema.

HTTP Status Codes

Status Code	Description
204	The battleship configuration was accepted
403	Configuration was attempted after the game was started
404	Either the game or the player couldn't be found
422	The battleship configuration was invalid (overlapping ships, ship not on grid)

Launching Attacks

`POST /shoot`
If it's the requesting player's turn, launches an attack at a specified target cell. Responds with the result of the attack.

Query Parameters

`gameId` (string)

The id of the active game.

`playerId` (string)

The id of the attacking player.

Body Parameters

`target` (Object)

The target grid cell to be attacked.

`target.row` (number)

The target row

`target.column` (number)

The target column

Example Response (200)

```
{
    "isHit": "true",
    "sunkShips": [
        { "row": 3, "column": 2 },
        { "row": 4, "column": 2 },
        { "row": 5, "column": 2 },
    ],
}
```

HTTP Status Codes

Status Code	Description
200	The target was successfully attacked
403	An attack can't be made at this time (game not started, out of turn, game over)
404	Either the game or the player couldn't be found
422	The target was invalid (target already attacked, target not on grid)

Get an empty board

`GET /boardTemplate`
Returns a 10x10 2D array of `false`'s, representing the board with no battleships.

Example Response (200)

```
{
    "board": [...],
}
```

HTTP Status Codes

Status Code	Description
200	Success

Get the template for battleships

`Get /battleshipTemplate`
Returns an example of a valid battleship configuration (see battleship configuration). Returned object contains a 5-grid-cell array, a 4-grid-cell array, two 3-grid-cell arrays, and a 2-grid-cell array.

Example Response (200)

```
{
    "battleships":[
        [
            {"row":0,"column":0},
            {"row":0,"column":1},
            "..."
        ],
        [
            {"row":1,"column":0},
            {"row":1,"column":1},
            "..."
        ],
        "..."
    ]
}
```

HTTP Status Codes

Status Code	Description
200	Success

Get the game status

`GET /status`
Returns the current status of the game (`pendingStart` , `hostTurn` , `guestTurn` , `gameOver`).

Query Parameters

`gameId` (string)

The id of the active game.

Example Response (200)

```
{
    "status"; "pendingStart",
}
```

HTTP Status Codes

Status Code	Description
200	Success
404	No game with matching ID found

Get your game data

`GET /data`
Returns all game data relevant to the requesting player.
Includes:

- the location of all battleships belonging to the player
- the game board with every occupied cell
- every guess the player has made
- every guess their opponent has made
- every ship sunk belonging to the player
- every ship sunk belonging to their opponent.

Query Parameters

`gameId` (string)

The id of the active game.

`playerId` (string)

The id of the requesting player.

Example Response (200)

```
{
    "data": {
        "ships": [...],
        "board": [...],
        "myGuesses": {
            "{\"row\":0,\"column\":1}": true,
            "{\"row\":0,\"column\":2}": true,
            "..."
        },
        "opponentGuesses": {
            "{\"row\":4,\"column\":2}": false,
            "{\"row\":2,\"column\":3}": false,
            "..."
        },
        "shipsIveLost": [],
        "shipsIveSunk": [],
}
```

HTTP Status Codes

Status Code	Description
204	The battleship configuration was accepted
404	Either the game or the player couldn't be found

Project Questions

What is the intended purpose of your application?

My application allows pairs of users to play multiplayer battleship online using their web browser.

What data will be stored and delivered by the API?

The data for each game session of battleship will be stored by the API and delivered to the client to enable gameplay.

What work has been completed for this prototype?

The game logic is fully functional. The API is nearly fully functional. The client interface is barely functional with severely throttled gameplay.

What work is left, and how do you plan to complete it?

The client interface needs to be basically entirely built. I already have it designed, it just needs to be coded.

Do you have a plan for going above and beyond? If so, what is it?

I don't know if typescript qualifies for that. That's not the reason I used it, but I did use it for the game logic so if that works then I plan to go above and beyond. If not, then I don't plan to go above and beyond.

If you used any borrowed code or code fragments, where did you get them from?

I used the webpack and typescript documentation to help configure my webpack settings. Those came from their respective documentation pages and are contained in my webpack.config.js and tsconfig.json files, respectively.