Conner Pappas, CS493: Assignment 7 - More Authentication and Authorization
URL: https://cpcs493loginships.appspot.com

GET /ships

Description: Get all ships stored in the datastore. The request must include an ‘Accept’ header with either ‘application/json’ or ‘text/html’ as the value. The response is the collection of ships in the requested format with a 200 status code. Any other value for the ‘Accept’ header returns a 406 status code.

Example response:

```
Response code: 200
[
    {
        "name": "User1 Ship",
        "type": "1 of a kind",
        "length": 45,
        "owner": "user1",
        "id": "5630742793027584",
        "self": "https://cpcs493loginships.appspot.com/ships/5630742793027584"
    },
    {
        "name": "User2 Ship",
        "type": "2 is a pair",
        "length": 89,
        "owner": "user2",
        "id": "5639026912526336",
        "self": "https://cpcs493loginships.appspot.com/ships/5639026912526336"
    }
]
```

POST /ships

Description: Create a new ship containing the properties specified in the request body. The properties that can be specified by the user are the name of the ship, the type of ship, and the length of the ship. The user must be authenticated, and the request must contain a JWT in the ‘Authorization’ header. The request must contain a ‘Content-Type’ header with the value ‘application/json’ since the server only accepts json, else the server responds with a 415 status code. If a nonvalid JWT is provided the response code is 401. The response to a successful request is the id of the new ship with a 201 status code.

Example request:

```
{
	"name":"Test Ship 1",
	"type": "Ship 1",
	"length": 10
}
```

Example response:

```
Response code: 201
{"id": 5959639107633152}
```

GET /users/{userID}/ships

Description: Get all ships stored owned by the user specified by userID. The request must include a JWT in the ‘Authorization’ header. If the JWT and userID do not agree then the response code is 403 and no ships are returned. The request must include an ‘Accept’ header with either ‘application/json’ or ‘text/html’ as the value. The response is the collection of ships in the requested format with a 200 status code. Any other value for the ‘Accept’ header returns a 406 status code.

Example response:

```
Response code: 200
[
   {
      "name": "User1 Ship",
      "type": "1 of a kind",
      "length": 45,
      "owner": "user1",
      "id": "5630742793027584",
      "self": "https://cpcs493loginships.appspot.com/ships/5630742793027584"
   }
]
```

DELETE ships/{shipID}

Descriptions: Delete a ship, specified by shipID, from the datastore. The request must include a JWT in the ‘Authorization’ header. If the user does not own that ship the response code is 403 and the ship isn’t deleted.

Example response:

```
Response code: 204
```

POST /login

Description: Login as a user with username and password credentials in the body of the request with the ‘Content-Type’ header set to ‘application/json’. The response for correct credentials is a JWT that can be used for request that require authentication.

Example request:

```
{
        "username": "user1@cpcs493loginships.com",
        "password": "cpcs493!"
}
```

Example response:

```
{
   "access_token": "7C3 ….. E3w",
   "id_token": "eyJ0e ….. ZbF2A",
   "scope": "openid profile email address phone",
   "expires_in": 86400,
   "token_type": "Bearer"
}
```

GET /users/userid

Description: Get the name and id of an authenticated user with the JWT in the ‘Authorization’ header. This method is for user’s to find out their own information. Note that userid is a resource since it is not in brackets.

Example response:

```
{
   "id": "5bf115020f9b19740d1085d5",
   "name": "user1"
}
```

POST /signup

Description: Signup with a new account for the API. The body of the request should contain an email address and a password to be used as the credentials for the new account. The response is the id for the new account and the email address that was just used for the registration.

Example request:

```
{
        "email": "user1@cpcs493loginships.com",
        "password": "cpcs493!"
}
```

Example response:

```
{
   "_id": "5bf14467734e847e73896a94",
   "email_verified": false,
   "email": "bt2eaularbov3i2rjshkpb@cpcs493loginships.com"
}
```

Note: User accounts are stored on Auth0. Authentication between this application and Auth0 is done through the use of the Auth0 authentication API (https://auth0.com/docs/api/authentication). Users submit their credentials for their account using the '/login' route above. This grants them a JWT to use for all authenticated processes. The "under the hood" details use the functionality of the express-jwt library to validate the JWT. When validated the authorization process is finished and the intended processes can resume.