Portfolio Project: Final Project

Relationships

Owner – Boats:

- One owner can have many boats.
- A boat can only be owned by a single owner.

Boat – Loads:

- Boat can have many loads.
- A load can only be loaded onto a single boat.

Data Model

The app stores one kind of entity in Datastore, Boats, for this assignment.

Owners – User Entity

Property	Data Type	Required	Notes
id	Integer		The id of the owner. This property will be generated automatically by Datastore.
sub	String	✓	The unique ID of each owner that will be generated by Auth0 when logging in for the first time.
nickname	String		The nickname given by Auth0 in the JWT.
email	String	✓	The email of the owner (the name property of Auth0's JWT).
boats	Object		The boat that the owner owns – it may be null if the owner does not have any boats. This object will have the boat's id and the boat's self attribute.
self	String		The url of the owner.

Boats

Property	Data Type	Required	Notes
id	Integer		The id of the boat. This property will be generated automatically by Datastore.
name	String	✓	Name of the boat – unique.
type	String	✓	Type of the boat. E.g., Sailboat, Catamaran, etc.
length	Integer	✓	The length of the boat in feet.
owner	Object	✓	The owner of the boat – has the owner's id and self-attribute. This is a required property but does not have to be filled in the request's body because the JWT will have that covered.
public	Boolean	✓	Whether or not the boat is a public or private boat.
loads	Object		Information of the load the boat is carrying – has the load's id and self attribute.
self	String		The url of the boat.

Loads

Property	Data Type	Required	Notes
id	Integer		The id of the load. This property will be generated automatically by Datastore.
item	String	✓	What the load is – eg. Toys, water bottles, pens.
volume	Integer	✓	The volume of items each load is.
origin	String	✓	The country of where the item originated.
carrier	Object		The boat that is carrying this load. The load may or may not be on a boat. If it is not on a boat, this property will be null.
self	String		The url of the load.

Comments

- When each object is displayed, it will have a “self” attribute that is **not** stored within Datastore, but generated as each request comes in. Specifically, for GET, POST, PATCH, PUT.
- When creating a new boat (POST /boats) or updating a property through PUT, these four properties are required and will have these unique constraints. New properties can be added (for example, “color”, “size”), however, they will not be validated. Only the required attributes will be validated. Specifically:
    - name – upper-case, lower-case letters (alphabet) and numbers (0-9) only – 50 characters limit. The attribute must also be unique.
    - type – upper-case, lower-case letters (alphabet) and numbers (0-9) only – 50 characters limit.
    - length – numbers (0-9) only – 10 characters limit.
    - public – Boolean.
- A new owner will be created when someone logs in for the first time. They will be created regardless of being a boat owner or not. Each new owner must have a email and the “sub” attribute given from Auth0 upon first log in. Therefore, only successful log in’s will be an “owner”.
    - sub – a unique id given from Auth0 - they will all be unique.
    - email – from Auth0.
- For new loads, the item, volume, and origin properties must be given, otherwise a successful creation will not happen.
    - item – must be letters(lower/uppercase) with a 50 characters limit.
    - volume – integer with 10-character limit.
    - origin – must be letters(lower/uppercase) with a 50 characters limit.
- User Entity Description:
    - The Owners unique identifier will be the “sub” property from the JWT given by Auth0.
    - If a request needs to supply a user identifier, the request must be done with a valid JWT.
    - The application maps a supplied JWT to the identifier of a user by using the “req.auth.sub” property.

Create a Boat – PROTECTED

Creation of a new boat. To have a successful post, the “name”, “type”, “length”, and “public” attributes must all be considered valid. To access, a valid JWT must be given in the request.

POST /boats – Protected Endpoint
Content-Type: application/json
Accepts: application/json
Authorization: Bearer Token – from JWT

Request

Path Parameters

None

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
name	The boat’s name – each boat name must be unique with only valid characters (upper- and lower-case letters, numbers, and spaces) and 50 characters allowed.	Yes
type	The type of boat. Only valid characters (upper- and lower-case letters, numbers, and spaces) and 50 characters allowed.	Yes
length	Length of the boat in feet – numbers and decimal character only.	Yes
public	States whether the boat is public or private information.	Yes

Request Body Example

Required Attributes:

```
{
  "name": "Odyssey",
  "type": "Yacht",
  "length": 99,
  "public": true
}
```

Extra Attributes:

```
{
  "name": "The Ship",
  "type": "Yacht",
  "length": 99,
  "public": false,
  "color": "Red",
  "location": null,
  "weight": 1
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	If extra attributes are given, they will be added – but they won't be checked for validation. All load attributes will be initialized to null. Loads will be assigned at a different endpoint. The owner attribute information will be added from the given JWT.
Failure	400 Bad Request	If the request does not have all 4 of the required attributes, has invalid values, or an empty body, the request will not create a new boat.
Failure	401 Unauthorized	If a valid JWT is not given. Token Expired.
Failure	403 Forbidden	If the request tries to create a new boat with a name already in use.
Failure	406 Not Acceptable	The client requests a media type the server can't provide.
Failure	415 Unsupported Media Type	If the request does not get sent with application/json.

Response Examples

Success – Status: 201 Created

Required Attributes:


```
{
    "owner": {
        "sub": "auth0|6557c6dec92e12d600a448c3",
        "id": "741",
        "self": "https://appspot.com/owners/741"
    },
    "public": true,
    "name": "Odyssey",
    "length": 99,
    "loads": [],
    "type": "Yacht",
    "id": "123",
    "self": "https://appspot.com/boats/123"
}
```


Extra Attributes:


```
{
    "owner": {
        "sub": "auth0|6552682f7409f13021869b75",
        "id": "741",
        "self": " https://appspot.com/owners/741"
    },
    "public": true,
```

```
    "color": "Red",
    "name": "The Ship",
    "length": 99,
    "loads": [],
    "weight": 1,
    "location": null,
    "type": "Yacht",
    "id": "123",
    "self": "https://appspot.com/boats/123"
}
```

Failure – Status 400 Bad Request

Missing Attribute

```
{
    "Error": "INVALID REQUEST: The request object is missing an attribute."
}
```

Character Limit/Invalid Attribute Value

```
{
    "Error": "INVALID REQUEST: An invalid attribute value detected."
}
```

Empty Body

```
{
    "Error": "INVALID REQUEST: Empty body is not allowed."
}
```

Failure – Status: 401 Unauthorized

NO AUTH

```
{
    "Error": "INVALID REQUEST: Unauthorized User."
}
```

EXPIRED TOKEN

```
{
    "Error": "INVALID TOKEN: Please renew token by logging in again. "
}
```

Failure – Status 403 Forbidden

```
{
    "Error": "INVALID REQUEST: The request object does not have a unique name value."
}
```

Failure – Status: 406 – Not Acceptable

```
{
    "Error": "INVALID REQUEST: Not an Acceptable Media Type."
}
```

Failure – Status 415 Unsupported Media Type

```
{
    "Error": "INVALID REQUEST: The server accepts 'application/json' requests only."
}
```

Get a Boat – PROTECTED

Allows you to get an existing boat.

GET /boats/:boat_id – Protected Endpoint
Content-Type: application/json
Accepts: application/json, text/html
Authorization: Bearer Token – from JWT

Request

Path Parameters

Name	Description
boat_id	ID of the boat

Request Header

Accepts: application/json or text/html

Request Body

None

Response

Response Body Format

JSON, HTML

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	JSON: returns a JSON object. HTML: returns boat information in html format with the "id" of each list element as the attribute in the "class" of "boat-attribute".
Failure	401 Unauthorized	A valid JWT was not given or the user does not own the boat they are trying to retrieve. Expired token.
Failure	404 Not Found	No boat with this boat_id exists
Failure	406 Not Acceptable	The server doesn't offer this MIME type.

Response Examples

Success – Status: 200 OK

JSON


```
{
    "owner": {
        "sub": "auth0|6557c6dec92e12d600a448c3",
        "id": "5751860518453248",
        "self": "http://localhost:8080/owners/5751860518453248"
    },
    "public": true,
    "name": "Odyssey",
    "length": 99,
    "loads": [],
```

```
    "type": "Yacht",
    "id": "5097238783066112",
    "self": "http://localhost:8080/boats/5097238783066112"
}
```

HTML

```
<li id="owner" class="boat-attribute">owner: [object Object]</li>
<li id="public" class="boat-attribute">public: true</li>
<li id="name" class="boat-attribute">name: Odyssey</li>
<li id="length" class="boat-attribute">length: 99</li>
<li id="loads" class="boat-attribute">loads: </li>
<li id="type" class="boat-attribute">type: Yacht</li>
<li id="id" class="boat-attribute">id: 5097238783066112</li>
<li id="self" class="boat-
attribute">self: http://localhost:8080/boats/5097238783066112</li>
```

Failure – Status: 401 Unauthorized

No JWT - Unauthorized Request

```
{
    "Error": "INVALID REQUEST: Unauthorized User."
}
```

Valid JWT, Wrong Owner

```
{
    "Error": "INVALID REQUEST: This owner_id does not own this boat."
}
```

EXPIRED TOKEN

```
{
    "Error": "INVALID TOKEN: Please renew token by logging in again. "
}
```

Failure – Status: 404 Not Found

```
{
    "Error": "INVALID REQUEST: No boat with this boat_id exists."
}
```

Failure – Status: 406 Not Acceptable

```
{
    "Error": "INVALID REQUEST: Not an Acceptable Media Type."
}
```

Edit a Boat – PATCH - PROTECTED

Allows you to edit a boat – allows updating any attribute while the other attributes remain untouched. The boat's boat_id is not allowed to be edited.

PATCH /boats/:boat_id
Content-Type: application/json
Accepts: application/json
Authorization: Bearer Token – from JWT

Request

Path Parameters

Name	Description
boat_id	ID of the boat – cannot be edited.

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
name	The name of the boat.	No
type	The type of the boat. E.g., Sailboat, Catamaran, etc.	No
length	Length of the boat in feet.	No
public	If the boat's information is available publicly.	No

Request Comments:

- None of these attributes are required when doing a PATCH request unless that attribute needs to be updated.
- Empty PATCH requests will send back an Error message.
- Only send attributes that need to be updated.
- When one of the main attributes (name, type, length, public) is sent in for an update, the attributes will be checked to see if they are valid. If they are not one of the main attributes, they will not be checked for validity.
- New attributes cannot be added with a patch, only pre-existing attributes will be allowed.

Request Body Example

PATCH – full main attributes:

```
{
  "name": "Sea Boat",
  "type": "Galleon",
  "length": 111,
  "public": false
}
```

PATCH – 1 attribute only:

```
{
    "public": false
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	400 Bad Request	If the request doesn't have at least one attribute to edit. If the request tries to edit a non-existing attribute. If the request attempts to edit the id.
Failure	401 Unauthorized	Invalid JWT or the user and the boat has no relation. Expired token.
Failure	403 Forbidden	The request violates the uniqueness constraint for the "name" attribute.
Failure	404 Not Found	No boat with this boat_id exists.
Failure	406 Not Acceptable	The client requests a media type the server cannot return.
Failure	415 Unsupported Media Type	PATCH requests will only accept JSON requests.

Response Examples

Success – Status: 200 OK


```
{
    "owner": {
        "sub": "auth0|6552686fdd42eb641c88f7bb",
        "id": "5722597228544000",
        "self": "http://localhost:8080/owners/5722597228544000"
    },
    "public": false,
    "name": "Patched Boat",
    "length": 111,
    "loads": [],
    "type": "Galleon",
    "id": "5743317224521728",
    "self": "http://localhost:8080/boats/5743317224521728"
}
```


Failure – Status: 400 Bad Request

EMPTY BODY


```
{
    "Error": "INVALID REQUEST: Empty body is not allowed."
}
```


INVALID ID PATCH


```
{
    "Error": "INVALID REQUEST: The id cannot be edited."
}
```

Failure – Status: 401 Unauthorized

NO AUTH

```
{
    "Error": "INVALID REQUEST: Unauthorized User."
}
```

VALID JWT, WRONG OWNER/BOAT HAS NO OWNER

```
{
    "Error": "INVALID REQUEST: This owner_id has no relation with this boat_id."
}
```

EXPIRED TOKEN

```
{
    "Error": "INVALID TOKEN: Please renew token by logging in again. "
}
```

Failure – Status: 403 Forbidden

```
{
    "Error": "INVALID REQUEST: The request object does not have a unique name value."
}
```

Failure – Status: 404 Not Found

```
{
    "Error": "INVALID REQUEST: No boat with this boat_id exists."
}
```

Failure – Status: 406 Not Acceptable

```
{
    "Error": "INVALID REQUEST: Not an Acceptable Media Type."
}
```

Failure – Status: 415 Unsupported Media Type

```
{
    "Error": "INVALID REQUEST: The server accepts 'application/json' requests only."
}
```

Comments:

- When adding new attributes to a PATCH request, they will not be considered. The request will go through, but the new attribute will be ignored and only the pre-existing attributes will be updated (if they needed to be).
- The “name” attribute will “allow” same names in the PATCH request, if it is for the boat that originally had that name.

Edit a Boat – PUT - PROTECTED

Allows you to edit a boat – updates the whole boat. The boat's boat_id will not be allowed to be edited. PUT requests will allow additions of new attributes but they will not be checked for validation. The "name" attribute uniqueness constraint must not be violated.

PUT /boats/:boat_id
Content-Type: application/json
Accepts: application/json
Authorization: Bearer Token – JWT

Request

Path Parameters

Name	Description
boat_id	ID of the boat – cannot be edited.

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
name	The name of the boat. The name attribute can be the same if updating the same boat.	Yes
type	The type of the boat. E.g., Sailboat, Catamaran, etc.	Yes
length	Length of the boat in feet.	Yes
public	Whether the boat's information is public or not.	Yes

Request Body Example

PUT – main attributes:

```
{
    "name": "Sea Rocks",
    "type": "Tanker Boat",
    "length": 600,
    "public": true
}
```

PUT – new attribute additions:

```
{
    "name": "Multiple Attributes Boat",
    "type": "Multiple Attributes Test",
    "length": 150,
    "color": "Black",
    "weight": 1,
    "edition": 1,
    "public": false
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	
Failure	400 Bad Request	If the request doesn't have at least one attribute to edit. If the request attempts to edit the ID.
Failure	401 Unauthorized	Invalid JWT or the boat and user have no relation. Expired token.
Failure	403 Forbidden	The request attempts to violate the unique name constraint.
Failure	404 Not Found	No boat with this boat_id exists.
Failure	406 Not Acceptable	The client requests a media type the server can't return.
Failure	415 Unsupported Media Type	PUT only accepts JSON requests.

Response Examples

Success – Status: 200 OK

```
{
    "owner": {
        "sub": "auth0|6557c6dec92e12d600a448c3",
        "id": "5750599274463232",
        "self": "http://localhost:8080/owners/5750599274463232"
    },
    "public": true,
    "name": "Sea Rocks",
    "length": 600,
    "loads": [
        {
            "id": "6046718076387328",
            "self": "http://localhost:8080/loads/6046718076387328"
        }
    ],
    "type": "Tanker Boat",
    "id": "5129794635169792",
    "self": "http://localhost:8080/boats/5129794635169792"
}
```

Failure – Status: 400 Bad Request

MISSING ATTRIBUTES

```
{
    "Error": "INVALID REQUEST: The request object is missing an attribute."
}
```

INVALID ATTRIBUTE VALUES

```
{
```

```
    "Error": "INVALID REQUEST: An invalid attribute value detected."
}
```

INVALID ID EDIT

```
{
    "Error": "INVALID REQUEST: The id cannot be edited."
}
```

EMPTY BODY

```
{
    "Error": "INVALID REQUEST: Empty body is not allowed."
}
```

Failure – Status: 401 Unauthorized

NO AUTH

```
{
    "Error": "INVALID REQUEST: Unauthorized User."
}
```

VALID JWT, WRONG OWNER

```
{
    "Error": "INVALID REQUEST: This owner_id has no relation with this boat_id."
}
```

EXPIRED TOKEN

```
{
    "Error": "INVALID TOKEN: Please renew token by logging in again. "
}
```

Failure – Status: 403 Forbidden

```
{
    "Error": "INVALID REQUEST: The request object does not have a unique name value."
}
```

Failure – Status: 404 Not Found

```
{
    "Error": "INVALID REQUEST: No boat with this boat_id exists."
}
```

Failure – Status: 406 Not Acceptable

```
{
    "Error": "INVALID REQUEST: Not an Acceptable Media Type."
}
```

Failure – Status: 415 Unsupported Media Type

```
{
    "Error": "INVALID REQUEST: The server accepts 'application/json' requests only."
}
```

Delete a Boat – PROTECTED

Allows you to delete a boat.

DELETE /boats/:boat_id
Accept: application/json
Authorization: Bearer Token – JWT

Request

Path Parameters

Name	Description
boat_id	ID of the boat

Request Body

None

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	
Failure	401 Unauthorized	No Authorization – No JWT. Expired token.
Failure	403 Forbidden	The user does not own the boat.
Failure	404 Not Found	No boat with this boat_id exists
Failure	406 Not Acceptable	When a client requests for media type the server can't return.

Response Examples

Success

Status: 204 No Content

Failure – Status: 401 Unauthorized

NO AUTH

```
{
    "Error": "INVALID REQUEST: Unauthorized user."
}
```

EXPIRED TOKEN

```
{
    "Error": "INVALID TOKEN: Please renew token by logging in again. "
}
```

Failure – Status: 403 Not Forbidden

```
{
    "Error": "INVALID REQUEST: This owner does not own this boat."
```

```
}
```

Failure – Status: 404 Not Found

```
{
    "Error": "INVALID REQUEST: No boat with this boat_id exists."
}
```

Failure – Status: 406 Not Acceptable

```
{
    "Error": "INVALID REQUEST: Not an Acceptable Media Type."
}
```

Comments:

- If a boat has no owner and is being deleted, the boat will be deleted without the need of re-assigning an owner.
- Loaded boats will be deleted and the updates will be made to Load objects that their carrier is now “null” because the boat has been deleted.

Invalid Requests

Some possible requests that a user might do that are considered invalid.

PUT /boats
DELETE /boats
DELETE /loads
GET /boats/:boat_id/loads/:load_id

Request

Path Parameters

None

Request Body

JSON

Request Body Format

JSON

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Failure	405 Method Not Allowed	These endpoints don't support these methods.

Failure – Status: 405 Not Allowed

```
{
    "Error": "INVALID REQUEST: This method is not supported at this endpoint."
}
```

Get all Boats – ALLOWS AUTHORIZATION

This endpoint will return data differently depending on if the request is sent with a valid JWT. If a valid JWT is sent, the endpoint will return all boats related to the user of the JWT. If there is no JWT, then the endpoint will return all public boats. This endpoint is also paginated with a limit of 5 boats per page. The "next" attribute will naturally show up if there are more results.

GET /boats
Authorization: Bearer Token – JWT (for all boats of an owner)

Request

Path Parameters

Name	Description
cursor	Pagination is set to return 5 boats at a time. If there are currently more than 5 total public boats, then the "next" button will be shown.

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	401 Unauthorized	Expired token

Response Examples

Success – Status: 200 OK

NO AUTH – MORE THAN 5 BOATS:


```
{
    "total": 14,
    "boats": [
        {
            "owner": {
                "sub": "auth0|6557c6dec92e12d600a448c3",
                "id": "5750599274463232",
                "self": "http://localhost:8080/owners/5750599274463232"
            },
            "public": true,
            "name": "This Boat Item Is Sending Exactly Fifty Characters",
            "length": 99,
            "loads": [],
            "type": "Yacht",
            "id": "4870139568717824",
            "self": "http://localhost:8080/boats/4870139568717824"
        },
        {
```

```
            "owner": {
                "sub": "auth0|6557c6dec92e12d600a448c3",
                "id": "5750599274463232",
                "self": "http://localhost:8080/owners/5750599274463232"
            },
            "public": true,
            "name": "FILLER BOAT2",
            "length": 100,
            "loads": [],
            "type": "Ship",
            "id": "5071094914482176",
            "self": "http://localhost:8080/boats/5071094914482176"
        },
        {
            "owner": {
                "sub": "auth0|6552682f7409f13021869b75",
                "id": "5761168853434368",
                "self": "http://localhost:8080/owners/5761168853434368"
            },
            "public": true,
            "name": "Single Patch",
            "length": 100,
            "loads": [],
            "type": "Ship",
            "id": "5084627349798912",
            "self": "http://localhost:8080/boats/5084627349798912"
        },
        {
            "owner": {
                "sub": "auth0|6552686fdd42eb641c88f7bb",
                "id": "5720442564247552",
                "self": "http://localhost:8080/owners/5720442564247552"
            },
            "public": true,
            "name": "Before Patch",
            "length": 100,
            "loads": [],
            "type": "Ship",
            "id": "5360978162089984",
            "self": "http://localhost:8080/boats/5360978162089984"
        },
        {
            "owner": {
                "sub": "auth0|6557c6dec92e12d600a448c3",
                "id": "5750599274463232",
                "self": "http://localhost:8080/owners/5750599274463232"
            },
```

```
            "public": true,
            "name": "Loaded Patch",
            "length": 100,
            "loads": [],
            "type": "Ship",
            "id": "5432790015279104",
            "self": "http://localhost:8080/boats/5432790015279104"
        }
    ],
    "next": "http://localhost:8080/boats?cursor=CigSImoMcH5wYWthLW9hdXRochILEgVCb2F0cxiAgID4%2F6LTCQwYACAA"
}
```


AUTH – LESS THAN 5 BOATS:


```
{
    "total": 4,
    "boats": [
        {
            "owner": {
                "sub": "auth0|6552682f7409f13021869b75",
                "id": "5761168853434368",
                "self": "http://localhost:8080/owners/5761168853434368"
            },
            "public": true,
            "name": "Single Patch",
            "length": 100,
            "loads": [],
            "type": "Ship",
            "id": "5084627349798912",
            "self": "http://localhost:8080/boats/5084627349798912"
        },
        {
            "owner": {
                "sub": "auth0|6552682f7409f13021869b75",
                "id": "5761168853434368",
                "self": "http://localhost:8080/owners/5761168853434368"
            },
            "public": false,
            "color": "Red",
            "name": "Voyage",
            "length": 99,
            "loads": [],
            "type": "Yacht",
            "id": "5924708121837568",
            "self": "http://localhost:8080/boats/5924708121837568"
        },
        {
```

```
            "owner": {
                "sub": "auth0|6552682f7409f13021869b75",
                "id": "5761168853434368",
                "self": "http://localhost:8080/owners/5761168853434368"
            },
            "public": false,
            "color": "Red",
            "name": "Voyage",
            "length": 99,
            "loads": [],
            "type": "Yacht",
            "id": "6263446555328512",
            "self": "http://localhost:8080/boats/6263446555328512"
        },
        {
            "owner": {
                "sub": "auth0|6552682f7409f13021869b75",
                "id": "5761168853434368",
                "self": "http://localhost:8080/owners/5761168853434368"
            },
            "public": false,
            "color": "Red",
            "name": "The Ship",
            "length": 99,
            "loads": [],
            "weight": 1,
            "location": null,
            "type": "Yacht",
            "id": "6284323518939136",
            "self": "http://localhost:8080/boats/6284323518939136"
        }
    ]
}
```


Failure – Status: 401 Unauthorized

EXPIRED TOKEN


```
{
    "Error": "INVALID TOKEN: Please renew token by logging in again. "
}
```


Comments:

- The “total” in the response returns the total number of all boats – all created boats (regardless of who owns them and whether or not they are private or public).

Create a Load – NOT PROTECTED

Creating a new load entry. Extra attributes besides the three required attributes will not be considered.

POST /loads
Content-Type: application/json
Accepts: application/json

Request

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
item	What the load is.	Yes
volume	The amount/number of the load.	Yes
origin	Where the load is coming from.	Yes

Request Body Example

Required Attributes:

```
{
  "volume": 1,
  "item": "Laptops",
  "origin": "France"
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	Displays the newly created load with its attribute. The carrier property will be initialized as null, the id property is given by Datastore, and the self property is made on the fly.
Failure	400 Bad Request	If the request is missing any of the 3 required attributes, the load will not be created and the 400 status code will be returned.
Failure	406 Not Acceptable	The client requests a media type the server can't provide.
Failure	415 Unsupported Media Type	If the request does not get sent with application/json.

Response Examples

Success – Status: 201 Created – Required/Extra Attributes


```
{
    "id": "9632",
    "volume": 4,
    "carrier": null,
    "item": "Toys",
    "origin": "Russia",
    "self": "https://appspot.com/loads/9632"
}
```


Failure – Status: 400 Bad Request


```
{
    "Error": "INVALID REQUEST: The request object is missing an attribute."
}
```


Failure – Status: 406 Not Acceptable


```
{
    "Error": "INVALID REQUEST: Not an Acceptable Media Type."
}
```


Failure – Status: 415 Wrong Media Type


```
{
    "Error": "INVALID REQUEST: The server accepts 'application/json' requests only."
}
```


Comments

- Initial “carrier” attributes will have the “null” attribute because loads are not assigned to a carrier when created.
- The “self” attribute will be created “on the fly” and not stored within Datastore.
- Extra added attributes will not be considered.

GET a Load - UNPROTECTED

Allows you to get a load with the given load_id.

GET /loads/:load_id

Request

Path Parameters

Name	Description
load_id	ID of the load

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	Displays the contents of the given load_id
Failure	404 Not Found	No load with this load_id exists

Response Examples

Success – Status: 200 OK

NO CARRIER:


```
{
    "id": "5151614545428480",
    "volume": 5,
    "carrier": null,
    "item": "LEGO Blocks",
    "origin": "America",
    "self": "http://localhost:8080/loads/5151614545428480"
}
```


CARRIER:


```
{
    "id": "4920818169544704",
    "volume": 5,
    "carrier": {
        "id": "5202293146255360",
        "name": "Odyssey",
        "self": "http://localhost:8080/boats/5202293146255360"
    },
    "item": "LEGO Blocks",
    "origin": "France",
    "self": "http://localhost:8080/loads/4920818169544704"
}
```

Failure – Status: 404 Not Found

```
{
    "Error": "INVALID REQUEST: No load with this load_id exists."
}
```

Comments

- In a non-empty “carrier” attribute, the “self” value is generated “on the fly” and not stored within Datastore.

GET all Loads – UNPROTECTED

Lists all existing loads with cursor-based pagination. Retrieves 5 results at a time. The result will have a "next" attribute with the next page of results unless it is the last page of results. The total returns the total count of created loads.

GET /loads

Request

Path Parameters / Request Body

cursor	Pagination with 5 items per page.

Response

Response Body Format

JSON

Response Statuses

Outcome	**Status Code**	**Notes**
Success	200 OK	Displays all the loads with pagination of 5 items at a time.

Response Examples

Success – 200 OK


```
{
    "total": 1,
    "loads": [
        {
            "id": "5659117611909120",
            "volume": 5,
            "carrier": null,
            "item": "LEGO Blocks",
            "origin": "France",
            "self": "https://paka-oauth.ue.r.appspot.com/loads/5659117611909120"
        }
    ]
}
```

Edit a Load – PATCH – UNPROTECTED

Allows you to edit a load – allows updating any attribute while the other attributes remain untouched. The load's load_id is not allowed to be edited. New attributes for the load cannot be added with a PATCH – they will be ignored.

PATCH /loads/:load_id
Content-Type: application/json
Accepts: application/json

Request

Path Parameters

Name	Description
load_id	ID of the load – cannot be edited.

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
volume	The volume/amount of the load (an integer).	No
Item	What the load is.	No
origin	Where the item was created/originated.	No

Request Comments:

- None of these attributes are required when doing a PATCH request unless that attribute needs to be updated.
- Empty PATCH requests will send back an Error message.
- Only send attributes that need to be updated.
- When one of the main attributes is sent in for an update, the attributes will be checked to see if they are valid.

Request Body Example

PATCH – full main attributes:

```
{
  "volume": 6,
  "item": "Water Bottles",
  "origin": "Netherlands"
}
```

PATCH – 1 attribute only:

```
{
  "volume": 20
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	400 Bad Request	If the request doesn't have at least one attribute to edit. If the request tries to edit a non-existing attribute. If the request attempts to edit the id.
Failure	404 Not Found	Non-existent load.
Failure	406 Not Acceptable	The client requests a media type the server cannot return.
Failure	415 Unsupported Media Type	PATCH requests will only accept JSON requests.

Response Examples

Success – Status: 200 OK

```
{
    "volume": 20,
    "carrier": null,
    "item": "Stuffed Animals",
    "origin": "Russia",
    "id": "5742856555724800",
    "self": "http://localhost:8080/loads/5742856555724800"
}
```

Failure – Status: 400 Bad Request

EMPTY BODY

```
{
    "Error": "INVALID REQUEST: Empty body is not allowed."
}
```

INVALID ID Patch

```
{
    "Error": "INVALID REQUEST: The id cannot be edited."
}
```

Failure – Status: 404 Not Found

```
{
    "Error": "INVALID REQUEST: No load with this load_id exists."
}
```

Failure – Status: 406 Not Acceptable

```
{
    "Error": "INVALID REQUEST: The id cannot be edited."
}
```

Failure – Status: 415 Unsupported Media Type

```
{
    "Error": "INVALID REQUEST: The server accepts 'application/json' requests only."
}
```

Edit a Load – PUT – UNPROTECTED

Allows you to edit a load. The load's load_id is not allowed to be edited. New attributes cannot be added with a PUT request – they will be ignored.

PUT /loads/:load_id
Content-Type: application/json
Accepts: application/json

Request

Path Parameters

Name	Description
load_id	ID of the load – cannot be edited.

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
volume	The volume/amount of the load (an integer).	Yes
Item	What the load is.	Yes
origin	Where the item was created/originated.	Yes

Request Comments:

- Empty PUT requests will send back an Error message.
- The main attributes will be checked to see if they are valid.

Request Body Example

```
{
  "volume": 6,
  "item": "Water Bottles",
  "origin": "Netherlands"
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	
Failure	400 Bad Request	If the request doesn't have all of the required attributes to edit. If the request tries to edit a non-existing attribute. If the request attempts to edit the id.
Failure	404 Not Found	Non-existent load.

Failure	406 Not Acceptable	The client requests a media type the server cannot return.
Failure	415 Unsupported Media Type	put requests will only accept JSON requests.

Response Examples

Success – Status: 201 Created


```
{
    "volume": 20,
    "carrier": null,
    "item": "Stuffed Animals",
    "origin": "Russia",
    "id": "5742856555724800",
    "self": "http://localhost:8080/loads/5742856555724800"
}
```


Failure – Status: 400 Bad Request

EMPTY BODY


```
{
    "Error": "INVALID REQUEST: Empty body is not allowed."
}
```


INVALID ID PUT


```
{
    "Error": "INVALID REQUEST: The id cannot be edited."
}
```


INVALID INPUTS


```
{
    "Error": "INVALID REQUEST: An invalid attribute value detected."
}
```


MISSING REQUIRED ATTRIBUTES


```
{
    "Error": "INVALID REQUEST: The request object is missing an attribute."
}
```


Failure – Status: 404 Not Found


```
{
    "Error": "INVALID REQUEST: No load with this load_id exists."
}
```


Failure – Status: 406 Not Acceptable


```
{
    "Error": "INVALID REQUEST: The id cannot be edited."
}
```


Failure – Status: 415 Unsupported Media Type


```
{
    "Error": "INVALID REQUEST: The server accepts 'application/json' requests only."
}
```

DELETE a Load – UNPROTECTED

Delete a load. Deletion of a load will also be reflected in the boat's "loads" attribute if the load is currently loaded onto a carrier.

DELETE /loads/:load_id

Request

Path Parameters

Name	Description
load_id	ID of the load

Request Body

None

Response

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	
Failure	404 Not Found	No load with this load_id exists

Response Examples

Success

Status: 204 No Content

Failure – Status: 404 Not Found

```
{
    "Error": "INVALID REQUEST: No load with this load_id exists."
}
```

Assign a Load to a Boat - PROTECTED

Allows you to put a load onto a boat.

PATCH /boats/:boat_id/loads/:load_id

Request

Path Parameters

Name	Description
boat_id	ID of the boat
load_id	ID of the load

Request Body

None

Response

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	Succeeds only if a valid boat_id and load_id is given with a valid JWT and if the load does not already have a carrier.
Failure	401 Unauthorized	An invalid JWT was given. The JWT has no relation to the boat. Expired token.
Failure	403 Forbidden	The load is already loaded on another boat
Failure	404 Not Found	The specified boat and/or load does not exist

Response Examples

Success

Status: 204 No Content

Failure – Status: 401 Unauthorized

INVALID JWT

```
{
    "Error": "INVALID REQUEST: Unauthorized User."
}
```

OWNER AND BOAT HAVE NO RELATION

```
{
    "Error": "INVALID REQUEST: This owner_id has no relation with this boat_id."
}
```

EXPIRED TOKEN

```
{
    "Error": "INVALID TOKEN: Please renew token by logging in again. "
}
```

Failure – Status: 403 Forbidden

```
{
    "Error": "INVALID REQUEST: The load is already loaded on another boat."
}
```

Failure – Status: 404 Not Found

```
{
    "Error": "INVALID REQUEST: The specified boat and/or load does not exist."
}
```

Comments

- A load cannot be assigned to multiple boats. However, a boat can carry multiple loads.
- If a boat has no owner and is being deleted, the boat will be deleted without the need of re-assigning an owner.

- Loaded boats will be deleted and the updates will be made to Load objects that their carrier is now “null” because the boat has been deleted.

Remove a Load off a Boat – PROTECTED

Load has been deleted off a carrier/boat.

DELETE /boats/:boat_id/loads/:load_id

Request

Path Parameters

Name	Description
boat_id	ID of the boat
load_id	ID of the load

Request Body

None

Response

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	Succeeds only if a boat exists with this boat_id, a load exists with this load_id, and if this load is loaded onto this boat. A valid JWT must be used.
Failure	401 Unauthorized	A valid JWT was not given. Expired token.
Failure	403 Forbidden	Boat has no owner
Failure	404 Not Found	No boat with this boat_id is loaded with the load with this load_id. Boat/Load doesn't exist.

Response Examples

Success

Status: 204 No Content

Failure – Status: 401 Unauthorized

NO AUTH

```
{
    "Error": "INVALID REQUEST: Unauthorized User."
}
```

VALID JWT, WRONG OWNER

```
{
    "Error": "INVALID REQUEST: This valid user has no relation with this boat."
}
```

EXPIRED TOKEN

```
{
    "Error": "INVALID TOKEN: Please renew token by logging in again. "
}
```

Failure – Status: 403 Forbidden

```
{
    "Error": "INVALID REQUEST: This boat has no owner."
}
```

Failure – Status: 404 Not Found

BOAT AND LOAD EXISTS, BUT NO RELATION

```
{
    "Error": "INVALID REQUEST: No boat with this boat_id is loaded with the load with this load_id."
}
```

BOAT/LOAD DOES NOT EXIST

```
{
    "Error": "INVALID REQUEST: The specified boat and/or load does not exist."
}
```

Comment

- A load cannot be assigned to multiple boats.
- However, a boat can carry multiple loads.
- Trying to remove a load that is not found on the given boat will return a 404 Request error.

Assign a Load onto a Boat – PROTECTED

Load has been assigned to a carrier/boat.

PATCH /boats/:boat_id/loads/:load_id

Request

Path Parameters

Name	Description
boat_id	ID of the boat
load_id	ID of the load

Request Body

None

Response

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	Succeeds only if a boat exists with this boat_id, a load exists with this load_id. A valid JWT must be used.
Failure	401 Unauthorized	A valid JWT was not given. Expired token.
Failure	403 Forbidden	Boat has no owner. Authorized user doesn't own the boat.
Failure	404 Not Found	No boat with this boat_id is loaded with the load with this load_id. Boat/Load doesn't exist.

Response Examples

Success

Status: 204 No Content

Failure – Status: 401 Unauthorized

NO AUTH

```
{
    "Error": "INVALID REQUEST: Unauthorized User."
}
```

EXPIRED TOKEN

```
{
    "Error": "INVALID TOKEN: Please renew token by logging in again. "
}
```

Failure – Status: 403 Forbidden

VALID JWT, WRONG OWNER

```
{
    "Error": "INVALID REQUEST: This valid user has no relation with this boat."
```

```
}
```
NO OWNER
```
{
    "Error": "INVALID REQUEST: This boat has no owner."
}
```
Failure – Status: 404 Not Found

BOAT AND LOAD EXISTS, BUT NO RELATION
```
{
    "Error": "INVALID REQUEST: No boat with this boat_id is loaded with the load with this load_id."
}
```
BOAT/LOAD DOES NOT EXIST
```
{
    "Error": "INVALID REQUEST: The specified boat and/or load does not exist."
}
```

Comment

- A load cannot be assigned to multiple boats.
- However, a boat can carry multiple loads.
- Trying to add a non-existent load will return a 404 Request error. Same for using a non-existent boat.

Get an Owner – PROTECTED

Allows you to get an existing owner. Users can only get their information – they cannot retrieve information on other existing owners.

GET /owners/:owner_id
Content-Type: application/json
Accepts: application/json, text/html
Authorization: Bearer Token – JWT

Request

Path Parameters

Name	Description
boat_id	ID of the boat

Request Header

Accepts: application/json or text/html

Request Body

None

Response

Response Body Format

JSON, HTML

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	JSON: returns a JSON object. HTML: returns boat information in html format with the "id" of each list element as the attribute in the "class" of "boat-attribute".
Failure	401 Unauthorized	To retrieve a specific user, a valid JWT is necessary. Expired token.
Failure	403 Forbidden	When a valid JWT/user, wrong owner.
Failure	404 Not Found	No boat with this boat_id exists
Failure	406 Not Acceptable	The server doesn't offer this MIME type.

Response Examples

Success – Status: 200 OK


```
{
    "sub": "auth0|6552686fdd42eb641c88f7bb",
    "nickname": "janedoe",
    "boats": [
        {
            "id": "5738356268859392",
            "self": "http://localhost:8080/boats/5738356268859392"
        },
        {
            "id": "5689069036109824",
```

```
            "self": "http://localhost:8080/boats/5689069036109824"
        },
        {
            "id": "4919906462072832",
            "self": "http://localhost:8080/boats/4919906462072832"
        },
        {
            "id": "5743317224521728",
            "self": "http://localhost:8080/boats/5743317224521728"
        }
    ],
    "id": "5722597228544000",
    "email": "janedoe@example.com",
    "self": "http://localhost:8080/owners/5722597228544000"
}
```


Failure – Status: 401 Unauthorized

NO AUTH


```
{
    "Error": "INVALID REQUEST: Unauthorized User."
}
```


EXPIRED TOKEN


```
{
    "Error": "INVALID TOKEN: Please renew token by logging in again. "
}
```


Failure – Status: 403 Forbidden


```
{
    "Error": "INVALID REQUEST: This verified user is not authorized for this request."
}
```


Failure – Status: 404 Not Found


```
{
    "Error": "INVALID REQUEST: This owner_id does not exist."
}
```


Failure – Status: 406 Not Acceptable


```
{
    "Error": "INVALID REQUEST: Not an Acceptable Media Type."
}
```

Get all Owners – UNPROTECTED

An unprotected endpoint – returns all users/owners. Has no pagination.

GET /owners

Request

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	Returns the owner/user's "sub", "nickname", "email", "boats", "id", and "self" attributes.

Response Example

Success – Status: 200 OK


```
{
    "total": 4,
    "owners": [
        {
            "sub": "auth0|657362269fa97e13efcc17bb",
            "nickname": "new_user",
            "boats": [],
            "email": "new_user@example.com",
            "id": "4798028208668672",
            "self": "http://localhost:8080/owners/4798028208668672"
        },
        {
            "sub": "auth0|6552686fdd42eb641c88f7bb",
            "nickname": "janedoe",
            "boats": [
                {
                    "id": "5414971135492096",
                    "self": "http://localhost:8080/boats/5414971135492096"
                },
                {
                    "id": "5995739968700416",
                    "self": "http://localhost:8080/boats/5995739968700416"
                },
                {
                    "id": "4915556767498240",
                    "self": "http://localhost:8080/boats/4915556767498240"
                },
                {
```

```
                "id": "5360978162089984",
                "self": "http://localhost:8080/boats/5360978162089984"
            }
        ],
        "id": "5720442564247552",
        "email": "janedoe@example.com",
        "self": "http://localhost:8080/owners/5720442564247552"
    },
    {
        "sub": "auth0|6557c6dec92e12d600a448c3",
        "nickname": "wallace",
        "boats": [
            {
                "id": "6193765677006848",
                "self": "http://localhost:8080/boats/6193765677006848"
            },
            {
                "id": "4870139568717824",
                "self": "http://localhost:8080/boats/4870139568717824"
            },
            {
                "id": "6303023672328192",
                "self": "http://localhost:8080/boats/6303023672328192"
            },
            {
                "id": "5432790015279104",
                "self": "http://localhost:8080/boats/5432790015279104"
            },
            {
                "id": "6609668029808640",
                "self": "http://localhost:8080/boats/6609668029808640"
            },
            {
                "id": "5071094914482176",
                "self": "http://localhost:8080/boats/5071094914482176"
            }
        ],
        "id": "5750599274463232",
        "email": "wallace@cheese.com",
        "self": "http://localhost:8080/owners/5750599274463232"
    },
    {
        "sub": "auth0|6552682f7409f13021869b75",
        "nickname": "johndoe",
        "boats": [
            {
                "id": "5924708121837568",
```

```
                    "self": "http://localhost:8080/boats/5924708121837568"
                },
                {
                    "id": "6263446555328512",
                    "self": "http://localhost:8080/boats/6263446555328512"
                },
                {
                    "id": "6284323518939136",
                    "self": "http://localhost:8080/boats/6284323518939136"
                },
                {
                    "id": "5084627349798912",
                    "self": "http://localhost:8080/boats/5084627349798912"
                }
            ],
            "id": "5761168853434368",
            "email": "johndoe@example.com",
            "self": "http://localhost:8080/owners/5761168853434368"
        }
    ]
}
```


Comments:

- The “sub”, “nickname”, and “email” attributes are given by Auth0.
- The “id” attribute is generated by Datastore with each new member added.
- The “self” attribute is generated on the fly when the data is requested and not stored within Datastore.
- The “boats” attribute is not paginated and holds record of the boat’s id. The “self” attribute is generated on the fly as well.
- The “total” returns the total number of owners that are recorded in Datastore.

Assign a Boat to an Owner – PROTECTED

Allows you to assign a boat to an owner. An owner can have multiple boats but a boat will be owned by a single owner.

PATCH /owners/:owner_id/boats/:boat_id
Authorization: Bearer Token – JWT

Request

Path Parameters

Name	Description
owner_id	ID of the owner
boat_id	ID of the boat

Request Body

None

Response

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	Succeeds only if a valid boat_id and owner_id is given and if the boat does not already have a carrier.
Failure	401 Unauthorized	No valid JWT or the owner of the boat doesn't match the user making the request. Expired token.
Failure	403 Forbidden	The boat is already assigned to another owner.
Failure	404 Not Found	The specified boat and/or owner does not exist

Response Examples

Success

Status: 204 No Content

Failure – Status: 401 Unauthorized

NO AUTH

```
{
    "Error": "INVALID REQUEST: Unauthorized user."
}
```

VALID JWT, WRONG OWNER

```
{
    "Error": "INVALID REQUEST: JWT does not match the given user_id."
}
```

EXPIRED TOKEN

```
{
    "Error": "INVALID TOKEN: Please renew token by logging in again. "
}
```

Failure – Status: 403 Forbidden

```
{
    "Error ": "INVALID REQUEST: This boat already has an owner."
}
```

Failure – Status: 404 Not Found

INVALID OWNER ID

```
{
    "Error": "INVALID REQUEST: This owner_id does not exist."
}
```

INVALID BOAT ID

```
{
    "Error": "INVALID REQUEST: No boat with this boat_id exists."
}
```

Comment

- A load cannot be assigned to multiple boats. However, a boat can carry multiple loads.

Unassign a Boat from an Owner – PROTECTED

Allows you to unassign a boat from an owner. An owner can have multiple boats but a boat will be owned by a single owner.

DELETE /owners/:owner_id/boats/:boat_id
Authorization: Bearer Token – JWT

Request

Path Parameters

Name	Description
owner_id	ID of the owner
boat_id	ID of the boat

Request Body

None

Response

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	Succeeds only if a valid boat_id and owner_id is given and if the boat is owned by this owner.
Failure	401 Unauthorized	No valid JWT or the owner of the boat doesn't match the user making the request. Expired token.
Failure	403 Forbidden	The boat has no owner.
Failure	404 Not Found	The specified boat and/or owner does not exist.

Response Examples

Success

Status: 204 No Content

Failure – Status: 401 Unauthorized

NO AUTH

```
{
    "Error": "INVALID REQUEST: Unauthorized user."
}
```

VALID JWT, WRONG OWNER

```
{
   "Error": "INVALID REQUEST: This authorized user has no relation with this boat_id."
}
```

EXPIRED TOKEN

```
{
    "Error": "INVALID TOKEN: Please renew token by logging in again. "
}
```

Failure – Status: 403 Forbidden

NO OWNER

```
{
    "Error": "INVALID REQUEST: This boat has no owner."
}
```

Failure – Status: 404 Not Found

INVALID OWNER ID

```
{
    "Error": "INVALID REQUEST: Non-existent owner_id given."
}
```

INVALID BOAT ID

```
{
    "Error": "INVALID REQUEST: No boat with this boat_id exists."
}
```

Comment

- A load cannot be assigned to multiple boats. However, a boat can carry multiple loads.