Portfolio Assignment: Final Project

CS 493: Cloud Application Development

Fall 2021

Oregon State University

Kevin Macandog

URL: https://portfolio-macandok.wl.r.appspot.com

Login: https://portfolio-macandok.wl.r.appspot.com/login

Data Model

The app stores two kinds of non-user entities in Datastore, Boats and Loads, and one user entity, Users.

The User entity is related to Boats and is protected. The Loads is only directly related to Boats and is unprotected.

Users

Property	Required	Data Type	Notes
userID	It's pre-made, so No.	String	The userID is made upon login or upon the creation of an account at /login, because a user entity has just been made then. Note that the userID is the same as the "sub", which was taken along with the JWT at /login. This is used as the unique identifier for the user. So use this for endpoints that needs a user ID. E.g., "auth0|619fe521af07b300718a9e0d". This is stored into the Datastore.
email	It's pre-made, so No.	String	Just like the userID, the email came along from the same source as the userID. E.g., "kevinmagnus_001@yahoo.com". This is stored into the Datastore.

boats	No.	object	The boat(s) that the user owns. It is pre-made and it's empty by default. When a boat is created, along with a valid user JWT, the boat that has just been created is then assigned here. An example of a boats array that will be stored in Datastore below, and note that the name of the boat is also stored into Datastore! But the self-link is not stored, only generated for display.
			`"boats":`
			`[`
			`{`
			`"boat_id": 1234,`
			`"self": https://<your-app>/loads/1234,`
			`"boat_name": "Titanic"`
			`}`
			`]`

Boats

Property	Required	Data Type	Notes
id	It's automatically-generated, so No.	Integer	The id of the boat. Datastore automatically generates it. Don't add it yourself as a property of the entity.
self	It's automatically-generated, so No.	String	The link that points to the canonical representation of the entity. It is generated on the fly based of the "id".
name	Yes.	String	Name of the boat. E.g., "Sea Witch", "Pirate Ship", etc.
type	Yes.	String	Type of the boat. E.g., "Sailboat", "Catamaran", etc.
length	Yes.	Integer	The length of the boat in feet. E.g., 100
userID	It's automatically-generated, so No.	String	This corresponds to the relationship with the user who owns the boat. The userID is the same as the user's "sub". E.g., "auth0|619fe521af07b300718a9e0d". This is stored into the Datastore.
userEmail	It's automatically-generated, so No.	String	Just like userID, this also corresponds to the relationship with the owner, but with the owner's email instead. This easily helps identify who the owner is. E.g., "kevinmagnus_001@yahoo.com". This is stored into the Datastore.
loads	No.	Array (of objects)	The load(s) being carried by the boat. A boat can have one or more loads at a time. Note: when creating a boat, do not bother to include loads, because all newly created boats are not supposed to have any loads by default. The "load" is empty upon the boat's creation. You can assign one using the "Add a relationship between a boat and a load" endpoint. An example of loads array that will be stored in Datastore below, and note that the content of the load is also stored into Datastore! But the self-link is not stored, only generated for display.

```
"loads":

[
    {

        "id": 5678,

        "self": https://<your-app>/loads/5678,

         "content": "Santa's Gifts"

    },
    {

        "id": 1234,
```

			<pre>"self": https://<your-app>/loads/1234,
			"content": "RTX 3090 Graphics Card"
			}
			]</pre>

Loads

Property	Required	Data Type	Notes
id	It's automatically-generated, so No.	Integer	The id of the load. Datastore automatically generates it. Don't add it yourself as a property of the entity.
self	It's automatically-generated, so No.	String	The link that points to the canonical representation of the entity. It is generated on the fly based of the "id".
content	Yes.	String	The content of the load. E.g., "drugs", "Amazon packages", etc.
volume	Yes.	Integer	The volume size of the load in feet. E.g., 10.
creation_date	Yes.	String	Date the load was created. In a string format of "MM/DD/YYYY". E.g., "11/29/2021".

carrier	No.	object	The boat carrier of the load (a load can only be carried by one boat at a time). Note: when creating a load, do not bother to include a carrier for it, because all newly created loads are supposed to be unassigned by default. The "carrier" is empty upon the load's creation. You can assign one using the "Add a relationship between a boat and a load" endpoint.
			An example of a carrier object that will be stored in Datastore below, and note that the name of the carrier is also stored into Datastore! But the self-link is not stored, only generated for display.
			<pre>"carrier": {
			"id": 1234,
			"self": https://<your-app>/loads/1234,
			"name": "Titanic"
			}</pre>

Other Information:

- “Users” have a one-to-many relationship with “Boats”. And “Boats” have a one-to-many relationship with “loads”.
- The “Boats” entity is directly related to “Users”, so it is a protected resource. “Loads” is unprotected.
- The JWT of the user needs to be taken from the webpage at /login. It’s used for the protected resource, the “Boats”.

View all Users

The users that will be displayed is unprotected so no JWT is needed so anyone can access this endpoint.

GET /users

Request

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	Returns an empty array if no users exist

Response Examples

Success

```
Status: 200 OK

[
  {
    "boats": [],
    "email": "ragnakevmac@gmail.com",
    "userID": "auth0|619feda983a20e006925e2be"
  },
  {
    "boats": [
      {
        "boat_id": 6295879195557888,
        "boat_name": "U1's Boat 1",
        "self": "https://portfolio-macandok.wl.r.appspot.com/boats/6295879195557888"
      }
    ],
    "email": "kevinmagnus_001@yahoo.com",
    "userID": "auth0|619fe521af07b300718a9e0d"
  }
]
```

Create a Boat

Allows you to create a new boat. All newly created boats begin without any loads. But all created boats are automatically assigned their owner's userID and userEmail. This is a protected resource so the user's JWT must be used, especially since the boat will be assigned the user's ID and Email upon creation.

POST /boats

Request

Path Parameters

None

Request Header

Accept: application/json

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
name	The name of the boat. E.g., "Sea Witch", "Pirate Ship", etc	Yes
type	The type of the boat. E.g., "Sailboat", "Catamaran", etc.	Yes
length	Length of the boat in feet. E.g., 100	Yes

Request Body Example

```
{
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 100
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	
Failure	400 Bad Request	If the request is missing any of the 3 required attributes, the boat must not be created, and 400 status code must be returned.
Failure	400 Bad Request	`Request cannot be empty`
Failure	406 Not Acceptable	`Accept header must have application/json`
Failure	403 Forbidden	`If the name of the boat already exists`
Failure	401 Unauthorized	`If missing or Invalid JWT`

Response Examples

- Datastore will automatically generate an ID and store it with the entity being created. Your app must send back this value in the response body as shown in the example.
- The `self` attribute will contain the live link to the REST resource corresponding to this boat. In other words, this is the URL to get this newly created boat. You must not store the `self` attribute in Datastore.
- Upon the creation of your new boat, an empty "loads" array will automatically be generated and will be stored in the Datastore.
- Upon the creation of the new boat, the owner's userID and userEmail are added to the properties and are then stored in the Datastore.

Success

Status: 201 Created

```
{
  "id": 123,
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 28,
  "loads": [],
  "userID": "auth0|619fe521af07b300718a9e0d",
  "userEmail": "kevinmagnus_001@yahoo.com",
  "self": "https://<your-app>/boats/123"
}
```

Failure

Status: 406 Not Acceptable

```
{
"Error":  "Accept header must have application/json"
}
```

Failure

Status: 400 Bad Request

```
{
"Error":  "Request cannot be empty"
}
```

Failure

Status: 400 Bad Request

```
{
"Error":  "The request object is missing at least one of the required attributes"
}
```

Failure

Status: 403 Forbidden

```
{
"Error": "The name of the boat already exists"
}
```

Failure

Status: 401 Unauthorized

```
{
"Error": "Missing or Invalid JWT"
}
```

View a Boat

Allows you to get an existing boat. The boat that will be displayed is exclusive to the user's JWT because this is a protected resource.

GET /boats/:boat_id

Request

Path Parameters

Name	Description
boat_id	ID of the boat

Request Body

None

Request Header

Accept: application/json

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	404 Not Found	If no boat with this boat_id exists
Failure	403 Forbidden	If `boat_id is owned by someone else`
Failure	401 Unauthorized	If `missing or Invalid JWT`

Response Examples

Success

Status: 200 OK

```
{
  "id": 123,
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 28,
  "loads": [
      {
          "id": 456,
          "content": "Christmas gifts",
          "self": "https://<your-app>/loads/456"
      },
      {
          "id": 789,
          "content": "Valentine chocolates",
          "self": "https://<your-app>/loads/789"
      }

  ],
```

```
  "userID": "auth0|619fe521af07b300718a9e0d",
  "userEmail": "kevinmagnus_001@yahoo.com",
  "self": "https://<your-app>/boats/123"
}
```

Failure

Status: 404 Not Found

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

Failure

Status: 403 Forbidden

```
{
"Error": "boat_id is owned by someone else"
}
```

Failure

Status: 401 Unauthorized

```
{
"Error": "Missing or Invalid JWT"
}
```

View all Boats

The boats that will be displayed are exclusive to the user's JWT because this is a protected resource. This then lists 5 items per page. There will be a "next" link at the bottom to show more results if there are more items. The last page will not show a "next" link. There will also be "Number of Total Items", available on each page, to indicate the total number of items in the collection corresponding only to the user (and not to other users).

GET /boats

Request

Path Parameters

None

Request Header

Accept: application/json

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	An empty array is returned if no boats are found.
Failure	406 Not Acceptable	`Accept header must have application/json`

Response Examples

Success

```
Status: 200 OK

{
  "items": [
    {
      "name": "boat 05",
      "type": "Angel Ship",
      "loads": [],
      "userID": "auth0|619fe521af07b300718a9e0d",
      "userEmail": "kevinmagnus_001@yahoo.com",
      "length": 55,
      "id": 555,
      "self": "https://<your-app>/boats/555"
    },
    {
      "name": "boat 04",
      "loads": [],
      "userID": "auth0|619fe521af07b300718a9e0d",
```

```
    "userEmail": "kevinmagnus_001@yahoo.com",
    "type": "Pirate Ship",
    "length": 44,
    "id": 444,
    "self": "https://<your-app>/boats/444"
  },
  {
    "name": "boat 03",
    "type": "Ghost Ship",
    "loads": [],
    "userID": "auth0|619fe521af07b300718a9e0d",
    "userEmail": "kevinmagnus_001@yahoo.com",
    "length": 33,
    "id": 333,
    "self": "https://<your-app>/boats/333"
  },
  {
    "name": "boat 02",
    "loads": [],
    "userID": "auth0|619fe521af07b300718a9e0d",
    "userEmail": "kevinmagnus_001@yahoo.com",
    "type": "Sea Witch",
    "length": 22,
    "id": 222,
    "self": "https://<your-app>/boats/222"
  },
  {
    "loads": [
      {
        "id": 736,
        "self": "https://<your-app>/loads/736",
        "content": "load 01"
      },
      {
        "id": 480,
        "self": "https://<your-app>/loads/480",
        "content": "load 02"
      },
      {
        "id": 992,
        "self": "https://<your-app>/loads/992",
        "content": "load 03"
      },
      {
        "id": 520,
        "self": "https://<your-app>/loads/520",
        "content": "load 04"
      }
```

```
        ],
        "userID": "auth0|619fe521af07b300718a9e0d",
        "userEmail": "kevinmagnus_001@yahoo.com",
        "length": 11,
        "type": "Catamaran",
        "name": "boat 01",
        "id": 111,
        "self": "https://<your-app>/boats/111"
      }
   ],
   "next": "https://<your-app>/boats?limit=5&offset=5"
}
```


Failure

Status: 406 Not Acceptable


```
{
"Error":  "Accept header must have application/json"
}
```

Edit a Boat [PATCH]

Allows you to edit a boat partially. The boat that will be displayed is exclusive to the user's JWT because this is a protected resource.

PATCH /boats/:boat_id

Request

Path Parameters

Name	Description
boat_id	ID of the boat

Request Body

Required

Request Header

Accept: application/json

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
name	The name of the boat. E.g., "Sea Witch", "Pirate Ship", etc	No, but at least one of these three.
type	The type of the boat. E.g., "Sailboat", "Catamaran", etc.	No, but at least one of these three.
length	Length of the boat in feet. E.g., 100	No, but at least one of these three.

Request Body Example

```
{
  "name": "Sea Witch",
  "type": "Catamaran"
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	400 Bad Request	If the request does not have any of the 3 attributes, the boat must not be updated, and 400 status code must be returned. If the request is empty with no attributes, then the boat must not be updated, and 400 status code must be returned.

		If the request contains an attribute other than name, type, or length, the boat must not be created, and 400 status code must be returned.
Failure	403 Forbidden	If the name of the boat already exists in Datastore, the boat must not be created, and 403 status code must be returned.
Failure	404 Not Found	No boat with this boat_id exists
Failure	406 Not Acceptable	Accept header must have application/json

Response Examples

- The `self` attribute will contain the live link to the REST resource corresponding to this boat. You must not store this attribute in Datastore.

Success

Status: 200 OK

```
{
  "id": 123,
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 1,
  "loads": [
    {
      "id": 736,
      "self": "https://<your-app>/loads/736",
      "content": "load 01"
    }
  ],
  "userID": "auth0|619fe521af07b300718a9e0d",
  "userEmail": "kevinmagnus_001@yahoo.com",
  "self": "https://<your-app>/boats/123"
}
```

Failure

Status: 400 Bad Request

```
{
"Error": "Request cannot be empty"
}
```

Status: 400 Bad Request

```
{
"Error": "We do not support attributes other than name, type, and length"
}
```

Status: 400 Bad Request

```
{
"Error": " The request object is missing at least one of the required attributes"
```

```
}
```

Failure

```
Status: 403 Forbidden

{
"Error":  "The name of the boat already exists"
}
```

Failure

```
Status: 404 Not Found

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

Failure

```
Status: 406 Not Acceptable

{
"Error":  " Accept header must have application/json"
}
```

Edit a Boat [PUT]

Allows you to edit a boat fully. The boat that will be displayed is exclusive to the user's JWT because this is a protected resource.

PUT /boats/:boat_id

Request

Path Parameters

Name	Description
boat_id	ID of the boat

Request Body

Required

Request Header

Accept: application/json

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
name	The name of the boat.	Yes
type	The type of the boat. E.g., Sailboat, Catamaran, etc.	Yes
length	Length of the boat in feet.	Yes

Request Body Example

```
{
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 99
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	400 Bad Request	If the request is missing any of the 3 required attributes, the boat must not be updated, and 400 status code must be returned. If the request is empty with no attributes, then the boat must not be updated, and 400 status code must be returned.

		If the request contains an attribute other than name, type, or length, the boat must not be created, and 400 status code must be returned.
Failure	403 Forbidden	If the name of the boat already exists in Datastore, the boat must not be created, and 403 status code must be returned.
Failure	403 Forbidden	If the boat_id is owned by someone else
Failure	404 Not Found	No boat with this boat_id exists
Failure	406 Not Acceptable	Accept header must have application/json

Response Examples

- The `self` attribute will contain the live link to the REST resource corresponding to this boat. You must not store this attribute in Datastore.

Success

Status: 200 OK

```
{
  "id": 123,
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 99,
  "loads": [
    {
      "id": 736,
      "self": "https://<your-app>/loads/736",
      "content": "load 01"
    }
  ],
  "userID": "auth0|619fe521af07b300718a9e0d",
  "userEmail": "kevinmagnus_001@yahoo.com",
  "self": "https://<your-app>/boats/123"
}
```

Failure

Status: 400 Bad Request

```
{
"Error": "The request object is missing at least one of the required attributes"
}
```

Status: 400 Bad Request

```
{
"Error": "Request cannot be empty"
}
```

Status: 400 Bad Request

```
{
```

```
"Error": "We do not support attributes other than name, type, and length"
}
```

Failure

Status: 403 Forbidden

```
{
"Error": "The name of the boat already exists"
}
```

Failure

Status: 404 Not Found

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

Failure

Status: 406 Not Acceptable

```
{
"Error": "Accept header must have application/json"
}
```

Delete a Boat

Allows you to delete a boat. Note that if the boat currently holds any loads, deleting the boat will make all of its loads unassigned. Also it will be unassigned from the corresponding user. This is a protected resource so the user must have a valid JWT.

```
DELETE /boats/:boat_id
```

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	403 Forbidden	If boat_id is owned by someone else
Failure	404 Not Found	If no boat with this boat_id exists
Failure	401 Unauthorized	If missing or invalid JWT

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 403 Forbidden

{
"Error":  "boat_id is owned by someone else"
}
```

Failure

```
Status: 404 Not Found

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

Failure

Status: 401 Unauthorized

```
{
"Error":  "Missing or Invalid JWT"
}
```

Create a Load

Allows you to create a new load. All newly created loads begin unassigned to any boat. This is an unprotected resource, so no JWT is needed so anyone can access this endpoint.

POST /loads

Request

Path Parameters

None

Request Header

Accept: application/json

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
content	The content of the load. E.g., “drugs”, “Amazon packages”, etc.	Yes
volume	The volume size of the load in feet. E.g., 10	Yes
creation_date	The date the load was created. In a string format of “MM/DD/YYYY”. E.g., “11/29/2021”.	Yes

Request Body Example

```
{
  "content": "Christmas Ugly Sweaters",
  "volume": 15,
  "creation_date": "11/29/2021"
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	
Failure	400 Bad Request	If the request is missing any of the 3 required attributes, the load must not be created, and 400 status code must be returned.
Failure	400 Bad Request	Request cannot be empty
Failure	406 Not Acceptable	Accept header must have application/json

Response Examples

- Datastore will automatically generate an ID and store it with the entity being created. This value needs to be sent in the response body as shown in the example.

- The value of the attribute `self` is a live link to the REST resource corresponding to this load. In other words, this is the URL to get this newly created load. You must not store this attribute in Datastore.
- Upon the creation of your new load, an empty "carrier" object will automatically be generated and will be stored in the Datastore.

Success

Status: 201 Created

```
{
  "id": 123,
  "content": "Halloween costumes",
  "volume": 15,
  "creation_date": "10/24/2021",
  "carrier": {}
  "self": "https://<your-app>/loads/123"
}
```

Failure

Status: 400 Bad Request

```
{
"Error":  "The request object is missing at least one of the required attributes"
}
```

Failure

Status: 400 Bad Request

```
{
"Error":  "Request cannot be empty
}
```

Failure

Status: 406 Not Acceptable

```
{
"Error":  "Accept header must have application/json"
}
```

View a Load

Allows you to get an existing load. This is an unprotected resource so no JWT is needed so anyone can access this endpoint.

GET /loads/:load_id

Request

Path Parameters

Name	Description
load_id	ID of the load

Request Body

None

Request Header

Accept: application/json

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	404 Not Found	No load with this load_id exists
Failure	406 Not Acceptable	Accept header must have application/json

Response Examples

Success

Status: 200 OK

```
{
  "id": 123,
  "content": "Halloween costumes",
  "volume": 15,
  "creation_date": "10/24/2021",
  "carrier": {
      "id": 456,
      "name": "Ghost Ship"
      "self": "https://<your-app>/boats/456"
  }
  "self": "https://<your-app>/loads/123"
}
```

Failure

Status: 404 Not Found

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

Failure

Status: 406 Not Acceptable

```
{
"Error":  "Accept header must have application/json"
}
```

View all Loads

The loads that will be displayed is unprotected so no JWT is needed so anyone can access this endpoint. This then lists 5 items per page. There will be a “next” link at the bottom to show more results if there are more items. The last page will not show a “next” link. There will also be “Number of Total Items”, available on each page, to indicate the total number of items in the collection.

GET /loads

Request

Path Parameters

None

Request Header

Accept: application/json

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	406 Not Acceptable	Accept header must have application/json

Response Examples

Success

Status: 200 OK

```
{
   "items": [
      {
         "content": "Star Wars 1st Movie DVDs",
         "volume": 5,
         "carrier": {
            "id": 008,
            "self": "https://<your-app>/boats/008",
            "name": "boat 01"
         },
         "creation_date": "10/18/2020",
         "id": 520,
         "self": "https://<your-app>/loads/520"
      },
      {
         "content": " Star Wars 2nd Movie DVDs",
         "volume": 5,
```

```
        "creation_date": "10/18/2020",
        "carrier": {
          "id": 008,
          "self": "https://<your-app>/boats/008",
          "name": "boat 01"
        },
        "id": 736,
        "self": "https://<your-app>/loads/736"
      },
      {
        "creation_date": "10/18/2020",
        "volume": 5,
        "carrier": {
          "id": 008,
          "self": "https://<your-app>/boats/008",
          "name": "boat 01"
         },
        "content": " Star Wars 3rd Movie DVDs",
        "id": 992,
        "self": "https://<your-app>/loads/992"
      }
   ],
   "next": "https://<your-app>/loads?limit=3&offset=3"
}
```


Failure


```
Status: 406 Not Acceptable

{
"Error":  "Accept header must have application/json"
}
```

Edit a Load [PATCH]

Allows you to edit a load partially. This is an unprotected resource so no JWT is needed so anyone can access this endpoint.

PATCH /loads/:load_id

Request

Path Parameters

Name	Description
load_id	ID of the load

Request Body

Required

Request Header

Accept: application/json

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
content	The content of the load. E.g., “drugs”, “Amazon packages”, etc.	No, but at least one of these three.
volume	The volume size of the load in feet. E.g., 10	No, but at least one of these three.
creation_date	The date the load was created. In a string format of “MM/DD/YYYY”. E.g., “11/29/2021”.	No, but at least one of these three.

Request Body Example

```
{
  "content": "Christmas Ugly Sweaters 2021",
  "volume": 15
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	400 Bad Request	If the request does not have at least 1 of the 3 attributes, the load must not be updated, and 400 status code must be returned. If the request is empty with no attributes, then the load must not be updated, and 400 status code must be returned.

		If the request contains an attribute other than content, volume, or creation_date, the load must not be created, and 400 status code must be returned.
Failure	404 Not Found	No load with this load_id exists
Failure	406 Not Acceptable	Accept header must have application/json

Response Examples

- The `self` attribute will contain the live link to the REST resource corresponding to this load. You must not store this attribute in Datastore.

Success

Status: 200 OK

```
{
  "id": 123,
  "content": "Christmas Ugly Sweaters 2021",
  "volume": 15,
  "creation_date": "12/25/1999",
  "carrier": {
      "id": 456,
      "name": "Santa's Ship"
      "self": "https://<your-app>/boats/456"
  }
  "self": "https://<your-app>/loads/123"
}
```

Failure

Status: 400 Bad Request

```
{
"Error":  "Request cannot be empty"
}
```

Status: 400 Bad Request

```
{
"Error":  "We do not support attributes other than content, volume, and
creation_date"
}
```

Status: 400 Bad Request

```
{
"Error":  " The request object is missing at least one of the required attributes"
}
```

Failure

Status: 404 Not Found

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

Failure

Status: 406 Not Acceptable

```
{
"Error":  " Accept header must have application/json"
}
```

Edit a Load [PUT]

Allows you to edit a load fully. This is an unprotected resource so no JWT is needed so anyone can access this endpoint.

PUT /loads/:load_id

Request

Path Parameters

Name	Description
load_id	ID of the load

Request Body

Required

Request Header

Accept: application/json

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
content	The content of the load. E.g., “drugs”, “Amazon packages”, etc.	Yes.
volume	The volume size of the load in feet. E.g., 10	Yes.
creation_date	The date the load was created. In a string format of “MM/DD/YYYY”. E.g., “11/29/2021”.	Yes.

Request Body Example

```
{
  "content": "Christmas Ugly Sweaters 2021",
  "volume": 15,
  "creation_date": "12/01/2021"
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	400 Bad Request	If the request is missing any of the 3 required attributes, the load must not be updated, and 400 status code must be returned.

		If the request is empty with no attributes, then the load must not be updated, and 400 status code must be returned. If the request contains an attribute other than content, volume, or creation_date, the load must not be created, and 400 status code must be returned.
Failure	404 Not Found	No load with this load_id exists
Failure	406 Not Acceptable	Accept header must have application/json

Response Examples

- The `self` attribute will contain the live link to the REST resource corresponding to this load. You must not store this attribute in Datastore.

Success

Status: 200 OK

```
{
  "id": 123,
  "content": "Christmas Ugly Sweaters 2021",
  "volume": 15,
  "creation_date": "12/01/2021",
  "carrier": {
      "id": 456,
      "name": "Santa's Ship"
      "self": "https://<your-app>/boats/456"
  }
  "self": "https://<your-app>/loads/123"
}
```

Failure

Status: 400 Bad Request

```
{
"Error":  "Request cannot be empty"
}
```

Status: 400 Bad Request

```
{
"Error":  "We do not support attributes other than content, volume, and
creation_date"
}
```

Status: 400 Bad Request

```
{
"Error":  " The request object is missing at least one of the required attributes"
}
```

Failure

Status: 404 Not Found

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

Failure

Status: 406 Not Acceptable

```
{
"Error": " Accept header must have application/json"
}
```

Delete a Load

Allows you to delete a load. If the load being deleted is assigned to a carrier, the boat carrier will now remove that load from its boat. This is an unprotected resource so no JWT is needed so anyone can access this endpoint.

DELETE /loads/:load_id

Request

Path Parameters

Name	Description
load_id	ID of the load

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	404 Not Found	No load with this load_id exists

Response Examples

Success

Status: 204 No Content

Failure

Status: 404 Not Found

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

Assign a Load to a Boat

Assign a load to a boat. A boat can have many loads. A load that is already assigned to another boat cannot be assigned to this boat. Upon succession, a load will now be in one of the boat's loads, and the boat will now be the carrier for that load. This is an unprotected resource so no JWT is needed so anyone can access this endpoint.

PUT /boats/:boat_id/loads/:load_id

Request

Path Parameters

Name	Description
load_id	ID of the load
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	Succeeds only if a boat exists with this boat_id, a load exists with this load_id, and the load is originally unassigned.
Failure	403 Forbidden	The load is already assigned to another boat.
Failure	404 Not Found	The specified boat and/or load does not exist

Response Examples

Success

Status: 204 No Content

Failure

Status: 403 Forbidden

```
{
"Error":  "The load is already assigned to another boat"
}
```

Status: 404 Not Found

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

Remove a Load from a Boat

Removes a certain load from a boat. Once removed, the boat will no longer carry the load, and the load will no longer be assigned to that boat carrier. This is an unprotected resource so no JWT is needed so anyone can access this endpoint.

```
DELETE /boats/:boat_id/loads/:load_id
```

Request

Path Parameters

Name	Description
load_id	ID of the load
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	Succeeds only if a boat exists with this boat_id, a load exists with this load_id, and the boat originally contains this load.
Failure	404 Not Found	`No load with this load_id is in the boat with this boat_id.`

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 404 Not Found
{
"Error":  " No load with this load_id is in the boat with this boat_id"
}
```