Assignment 4: API Spec

CS 493: Cloud Application Development

Fall 2020

Oregon State University

Nicolas Barraclough barracln@oregonstate.edu

Change log

Page	Change	Date
1	Initial version.	Tu, Oct 27

Create a Boat

Allows you to create a new boat.

POST /boats

Request

Request Parameters

None

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

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

Request Body Example

```
{
    "name": "Sea Witch",
    "type": "Catamaran",
    "length": 28
}
```

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. You don't need to validate the values of the attributes and can assume that if the request contains any of the listed attributes, then that attribute's value is valid. You can also assume that the request will not contain any extraneous attribute (i.e., the request JSON will never contain any attribute other than the ones that are listed).

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 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

Success

Status: 201 Created


```
{
    "id": "abc123",
    "name": "Sea Witch",
  "type": "Catamaran",
    "length": 28,
    "self": "https://<your-app>/boats/abc123"
}
```


Failure

Status: 400 Bad Request


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


Get a Boat

Allows you to get an existing boat

GET /boats/:boat_id

Request

Request Parameters

Name	Type	Description	Required?
boat_id	String	ID of the boat	Yes

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	404 Not Found	No boat with this boat_id exists

Response Examples

Success

Status: 200 OK


```
{
    "id": "abc123",
    "name": "Sea Witch",
   "type": "Catamaran",
    "length": 28,
    "self": "https://<your-app>/boats/abc123",
   “loads”: []
}
```


Failure

Status: 404 Not Found


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


List all Boats

List all the boats.

GET /boats

Request

Request Parameters

None

Request Body
None

Response
Response Body Format
JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	Self must be added to the response for each load and boat. Pagination is required. At most 3 items per page.

Response Examples
Success

```
Status: 200 OK

[
{
   "id": "abc123",
   "name": "Sea Witch",
   "type": "Catamaran",
   "length": 28,
   "self": "https://<your-app>/boats/abc123",
   “loads”: []
},
{
   "id": "def456",
   "name": "Adventure",
   "type": "Sailboat",
   "length": 50,
   "self": "https://<your-app>/boats/def456",
   “loads”: [
       {    "id": 5650604803815162,
            "self": "http://127.0.0.1:8080/loads/5650604803815162"
       }
    ]
},
{
   "id": "xyz123",
   "name": "Hocus Pocus",
   "type": "Sailboat",
   "length": 100,
   "self": "https://<your-app>/boats/xyz123",
   “loads”: []
}
]
```

Delete a Boat
Allows you to delete a boat. Note that if the boat currently has loads, deleting the boat makes the loads’ carrier nothing.

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

Request
Request Parameters
None

Request Body
None

Response
No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	All loads on the boat must be unloaded before deleting.
Failure	404 Not Found	No boat with this boat_id exists

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 404 Not Found

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

Create a Load

Allows you to create a new load.

```
POST /loads
```

Request

Request Parameters

None

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Type	Description	Required?
weight	Integer	Weight of the load in kg	Yes
content	String	Contents of the load	Yes
delivery_date	Integer	Date for the load to be delivered	Yes
carrier	Integer	Boat currently carrying the load	No

Request Body Example

```
{
  "weight": 5,
  "carrier": null,
  "content":"LEGO Blocks",
  "delivery_date": "10/18/2020"
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	

Failure	400 Bad Request	If the request is missing the number attribute, the load must not be created, and 400 status code must be returned. You don't need to validate the values of this attribute and can assume that if the number attribute is specified, then its value is valid. You can also assume that the request will not contain any extraneous attribute (i.e., the request JSON will never contain any attribute other than number).

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 carrier is the ID of the boat currently carrying this load. If there is no boat carrying this load, the value of carrier should be null
- 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

Success

```
Status: 201 Created

{
  “id”: 6738492038595401,
  "weight": 5,
  "carrier": null,
  "content":"LEGO Blocks",
  "delivery_date": "10/18/2020",
  “self”: “http://127.0.0.1:8080/loads/6738492038595401”
}
```

Failure

```
Status: 400 Bad Request

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

Get a Load

Allows you to get an existing load.

```
GET /loads/:load_id
```

Request

Request Parameters

Name	Type	Description	Required?
load_id	String	ID of the load	Yes

Request Body

None

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

Response Examples

Success

```
Status: 200 OK

{
  “id”: 6738492038595401,
  "weight": 5,
  "carrier": null,
  "content":"LEGO Blocks",
  "delivery_date": "10/18/2020",
  “self”: “http://127.0.0.1:8080/loads/6738492038595401”
}
```

Failure

```
Status: 404 Not Found

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

List all Loads

List all the loads.

```
GET /loads
```

Request

Request Parameters

None

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	Self must be added to the response for each load and boat. Pagination is required. At most 3 items per page.

Response Examples

Success

```
Status: 200 OK

{
   “loads”: [
  {
    “id”: 6738492038595401,
    "weight": 5,
    "carrier": {
          “id”: 5087093063687384,
          “name”: “boat1”,
          “self”: “http://127.0.0.1:8080/boats/5087093063687384”
     },
     "content":"LEGO Blocks",
     "delivery_date": "10/18/2020",
     “self”: “http://127.0.0.1:8080/loads/6738492038595401”
  },
  {
    "weight": 2,
    "carrier": null,
    "content":"Paper cranes",
    "delivery_date": "1/1/2021"
   }
 ],
```

```
"next": "http://127.0.0.1:8080/loads?limit=3&offset=3"
}
```

Delete a Load

Allows you to delete a load. If the load being deleted is carried by a boat, the load no longer exists in the boat's loads.

```
DELETE /loads/:load_id
```

Request

Request Parameters

None

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

```
PUT /boats/:boat_id/loads/:load_id
```

Request

Request Parameters

Name	Type	Description	Required?
load_id	String	ID of the load	Yes
boat_id	String	ID of the boat	Yes

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.
Failure	403 Forbidden	Load already has a carrier
Failure	404 Not Found	No boat with this boat_id exists, and/or no load with this load_id exits.

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 403 Forbidden

{
"Error":  "This load is already assigned to a boat"
}
```

```
Status: 404 Not Found

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

Comment

- A load can only have up to one carrier.
- Removing a load from a boat should not delete the load.

Remove Load from Boat

Unload the load from the boat.

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

Request

Request Parameters

Name	Type	Description	Required?
load_id	String	ID of the load	Yes
boat_id	String	ID of the boat	Yes

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.
Failure	403 Forbidden	Load must be on the boat
Failure	404 Not Found	No boat with this boat_id exists, and/or no load with this load_id exits.

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 403 Forbidden

{
  "Error":  "Load is not on this boat"
}
```

```
Status: 404 Not Found

{
  "Error":  "No boat/load exists with provided ID(s)"
}
```

Data Model

BOATS

Property Name	Notes
id	The ID of the boat (automatically generated by Datastore).
name (String)	Name of the boat.
type (String)	Type of the boat e.g. Sailboat, Catamaran, etc.
length (Integer)	Length of the boat in meters.

LOADS

Property Name	Notes
id	The ID of the load (automatically generated by Datastore).
weight (Integer)	Weight of the load in kg.
content (String)	Contents of the load.
delivery_date (String)	Date the load is to be delivered.
carrier	