https://a4-loyami.uw.r.appspot.com/

Assignment 4: Intermediate Rest API Specification

CS 493: Cloud Application Development

Michelle Alexandra Loya

Data Model

The app stores two kinds of entities in Datastore, Boats and Loads.

Boats

Property	Data Type	Notes
id	Integer	The id of the boat. Datastore automatically generates it. Don't add it yourself as a property of the entity.
name	String	Name of the boat.
type	String	Type of the boat. E.g., Sailboat, Catamaran, etc.
length	Integer	The length of the boat in feet.
loads	List	List of dictionary objects where each object contains the **id**, **volume**, **item**, **creation_date**, and **self** link of the load.
self	String	The direct URL or link to access the boat. Automatically generated upon creation.

Slips

Property	Data Type	Notes
id	Integer	Automatically generated upon creation. Unique identifier for each load.
volume	Integer	Represents the size or magnitude of the load.
carrier	Dictionary	Contains details about the boat carrying the load (e.g., **id**, **name**, and **self**). If the load is not assigned to any boat, this would be set to None.
item	String	Describes what the load contains (e.g., "LEGO Blocks").
creation_date	String/Date	The date the load was created.
self	String	The direct URL or link to access the load. Automatically generated upon creation.

Create a Boat

Allows you to create a new boat.

POST /boats

Request

Path Parameters
None

Request Body
Required

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": 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 that is not listed above).

Response Examples
Datastore will automatically generate an ID and store it with the entity being created. Along with this ID, the app will also initialize and store an empty loads list. In addition to the ID, loads list, and input attributes, a self-attribute URL must be generated on the fly and sent back with all the values in the response body as shown in the example.

Success

```
Status: 201 Created
{
   "id": 123,
```

```
    "name": "Sea Witch",
     "type": "Catamaran",
    "length": 28
    "loads": [],
     "self": "http://example.com/boats/123"
}
```


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

Path Parameters

Name	Description
boat_id	ID of the boat

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

Self-attributes are not stored in Datastore and must generated on the fly (for both boats and loads).

Success

```
Status: 200 OK
{
   "id": 123,
   "name": "Sea Witch",
    "type": "Catamaran",
   "length": 28
   "loads": [],
    "self": "http://example.com/boats/123"
}
```

Failure

```
Status: 404 Not Found
{
"Error":  "No boat with this boat_id exists"
}
```

List all Boats

List all the boats.

```
GET /boats
```

Request

Path Parameters

None

Query Parameters

Name	Data Type	Description
offset	integer	Specifies where in the list of boats to start returning results. Default is 0.

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	

Response Examples

The API supports pagination, where a user can specify an offset query parameter to determine the starting point for the list of boats returned. The default return limit is set to 3. If there are more boats available than the limit, a "next" link will be provided in the response to fetch the next set of boats. Self-attributes are not stored in Datastore and must generated on the fly (for both boats and loads).

Success

```
Status: 200 OK
{
[
{
  "id": 123, "name": "Sea Witch",
  "type": "Catamaran",
  "length": 28,
  "loads": [],
  "self": "http://localhost:8080/boats/123"
},
{
  "id": 456,
  "name": "Adventure",
  "type": "Sailboat",
  "length": 50, "loads": [],
  "self": "http://localhost:8080/boats/456"
},
{
  "id": 789,
  "name": "Hocus Pocus",
  "type": "Sailboat",
  "length": 100,
  "loads": [],
  "self": "http://localhost:8080/boats/789"
}
],
"next": "http://localhost:8080/boats?offset=3"
}
```

Delete a Boat

Allows you to delete a boat. Deleting a boat will unload any loads that were loaded on to it.

```
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	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. All newly created loads begin unassigned to any boat.

POST /loads

Request

Path Parameters

None

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
volume	The volume of the load.	Yes
item	Description of the item being loaded.	Yes
creation_date	The date when the load was created.	Yes

Request Body Example

```
{
  "volume": 1200,
  "item": "Containers",
  "creation_date": "2023-10-30"
}
```

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. Along with this ID, your app will also initialize an empty carrier attribute for the entity. Your app must send back these values, along with generated a self-link, in the response body as shown in the example.

Success

```
Status: 201 Created
{
   "id": 456,
   "volume": 1200,
  "item": "Containers",
  “carrier”: None
  "creation_date": "2023-10-30",
  "self": " http://localhost:8080/loads/456"
}
```

Failure

```
Status: 400 Bad Request
{
"Error":  "The request object is missing at least one of the required attributes”
}
```

Get a Load

Allows you to get an existing load.

```
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	
Failure	404 Not Found	No load with this load_id exists

Response Examples

Self-attributes are not stored in Datastore and must be generated on the fly (for both boats and loads).

Success

```
Status: 200 OK
{
   "id": 456,
   "volume": 1200,
   "item": "Containers",
   “carrier”: None
   "creation_date": "2023-10-30",
   "self": " http://localhost:8080/loads/456"
}
```

Failure

```
Status: 404 Not Found
{
"Error":  " No load with this load_id exists "
}
```

List all Loads

List all the loads.

GET /loads

Request

Path Parameters

None

Query Parameters

Name	Data Type	Description
offset	integer	Specifies where in the list of loads to start returning results. Default is 0.

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	

Response Examples

The API supports pagination, where a user can specify an offset query parameter to determine the starting point for the list of loads returned. The default return limit is set to 3. If there are more loads available than the limit, a "next" link will be provided in the response to fetch the next set of loads. Self-attributes are not stored in Datastore and must be generated on the fly (for both boats and loads).

Success

```
Status: 200 OK
{
[
{
   "id": 456,
   "volume": 1200,
  "item": "Containers",
  “carrier”: None
  "creation_date": "2023-10-30",
  "self": " http://localhost:8080/loads/456"
}
,
{
   "id": 789,
   "volume": 1200,
  "item": "Containers",
  “carrier”: None
  "creation_date": "2023-10-30",
  "self": " http://localhost:8080/loads/789"
}
,
{
   "id": 123,
   "volume": 1200,
  "item": "Containers",
  “carrier”: None
  "creation_date": "2023-10-30",
  "self": " http://localhost:8080/loads/123"
}
],
"next": "http://localhost:8080/loads?offset=3"
}
```

Delete a Load

Allows you to delete a Load. If the load being deleted was on a boat, the load is removed from the boat's loads.

```
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
"
}
```

View All Loads for a Specific Boat

Retrieve all loads for a specific boat.

```
GET /boats/:boat_id/loads
```

Request

Path Parameters

Name	Description
boat_id	ID of the boat

Request Body

None

Response

No body

Response Body Format

Success: JSON

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 Ok	Returns the specified boat with all its associated loads.
Failure	404 Not Found	No boat with the given boat_id exists.

Response Examples

Self-attributes are not stored in Datastore and must be generated on the fly (for both boats and loads).

Success

```
Status: 200 Ok

{
  "id": "boat1234",
  "name": "Boaty",
  "type": "Sailboat",
  "length": 28,
  "loads": [
    {
      "id": "load1234",
      "volume": 10,
      "item": "Cargo A",
      "creation_date": "2023-10-10",
      "self": "http://example.com/loads/load1234"
    },
    {
      "id": "load5678",
      "volume": 20,
      "item": "Cargo B",
      "creation_date": "2023-10-15",
      "self": "http://example.com/loads/load5678"
    }
  ],
```

```
  "self": "http://example.com/boats/boat1234"
}
```

Failure

```
Status: 404 Not Found
{
"Error":  " No boat with this boat_id exists"
}
```

Assign a Load to a Boat

Assign a specified load to a specified boat.

```
PUT /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

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 and load with matching ids exist and if load is not already assigned to another boat
Failure	404 Not Found	The specified boat and/or load does not exist
Failure	403 Forbidden	The load is already loaded on another boat

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 404 Not Found
{
"Error":  " The specified boat and/or load does not exist "
}
```

```
Status: 403 Forbidden
{
"Error":  " The load is already loaded on another boat "

}
```

Delete a Load from a Boat

Remove a specified load from a specified boat. The load's carrier attribute will be cleared, and the load will be removed from the boat's list of loads. The load, itself, will not be deleted

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

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 and load with matching ids exist and if load is assigned to the specified boat
Failure	404 Not Found	The specified boat and/or load does not exist or the load is not loaded on this boat.

Response Examples

Success

Status: 204 No Content

Failure

```
Status: 404 Not Found
{
"Error":  " The specified boat and/or load does not exist "
}
```