**Robust Server Structure: URL shortener service assignment**

**Instructions**

Your task is to build a URL shortener service API using Node.js and Express. It should allow users to submit a URL and receive a "shortened" code, or ID, that can be used to retrieve the original URL later. It should also keep track of how often each shortened URL is retrieved so you can calculate the most popular URL's.

**What is a URL Shortener?**

The e-commerce company that you work for sells many different products under different categories. For example: www.shoppingsite.com/category/shoe/product/nike132032.

If a customer wants to share a link to the product on Twitter, they may run into restrictions on the text length.

A URL shortener service overcomes this issue by shortening www.shoppingsite.com/category/shoe/product/nike/132032 to www.shoppingsite.com/8d13lk2k.

**Existing files**

You will only need to edit the src folder and to follow code organization principles you learned in this module.

Use the existing data files located in src/data for the responses. Feel free to add or remove data from the files as necessary, but keep the same shape of the data.

**Tasks**

**Create routes and handlers to create, read, update, delete, and list short urls**

You will need to create the following API endpoints for the urls resources:

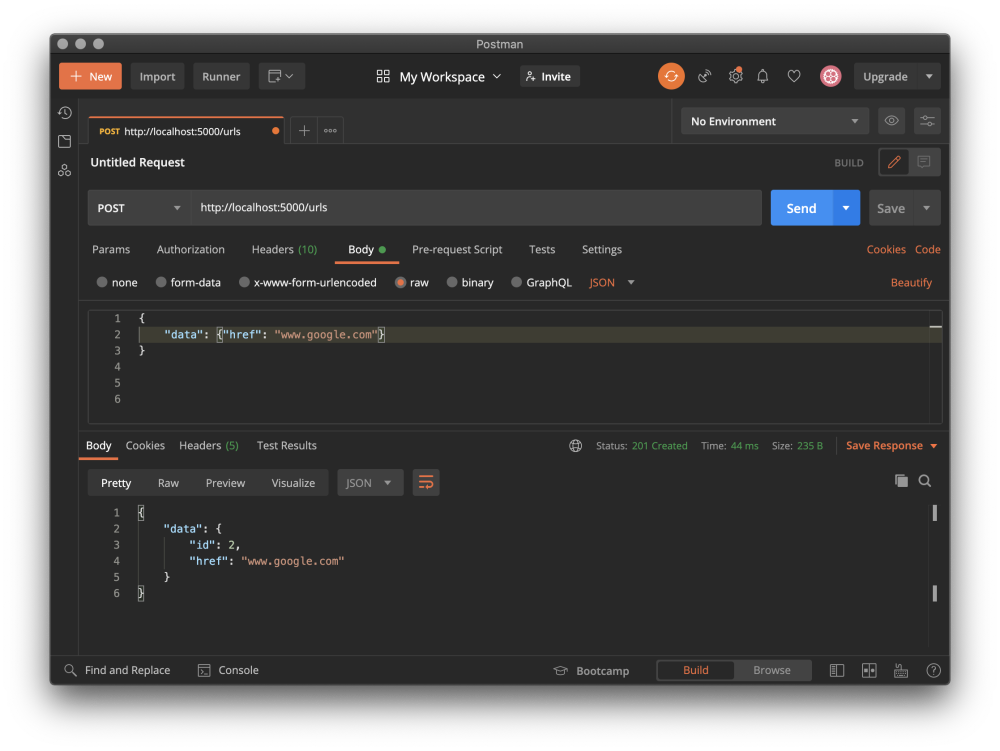
| **HTTP Verb** | **Path** | **Description** |
| --- | --- | --- |
| POST | /urls | create a new short url |
| GET | /urls/:urlId | retrieve a short url by id |
| PUT | /urls/:urlId | update a short url by id |
| GET | /urls | retrieve a list of all short url's |
| GET | /urls/:urlId/uses | retrieve a list of use metrics for a given short url id |
| GET | /urls/:urlId/uses/:useId | retrieve a use metric by id for a given short url id |

Short URL's cannot be deleted once created, because this would break existing links.

**Create**

The following Postman screen shot shows the data posted to /urls, and the response from the server.

1:1



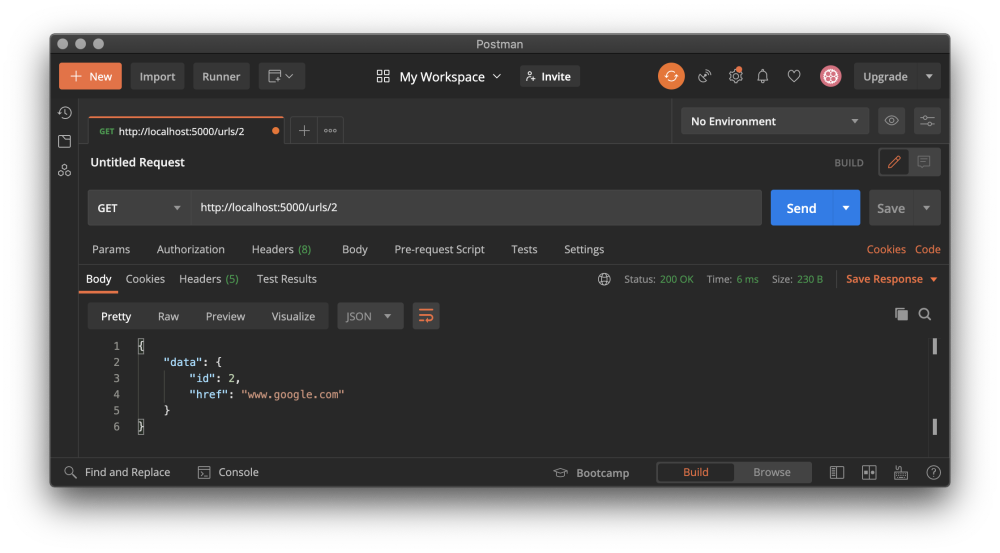
POST { data: {"href":"www.some-url.com"} } to /urls should assign an id to the object, save it, and return the saved object as a response to the client.

**Read**

The following Postman screen shot shows a GET request to /urls/:urlId, and the response from the server.

Additionally, use records are created as a side-effect of a GET request to /urls/:urlId. Each use record contains an id , a urlId which corresponds to id of the URL being tracked by the use metric, and a time property (set to Date.now()) indicating when the use metric was recorded.

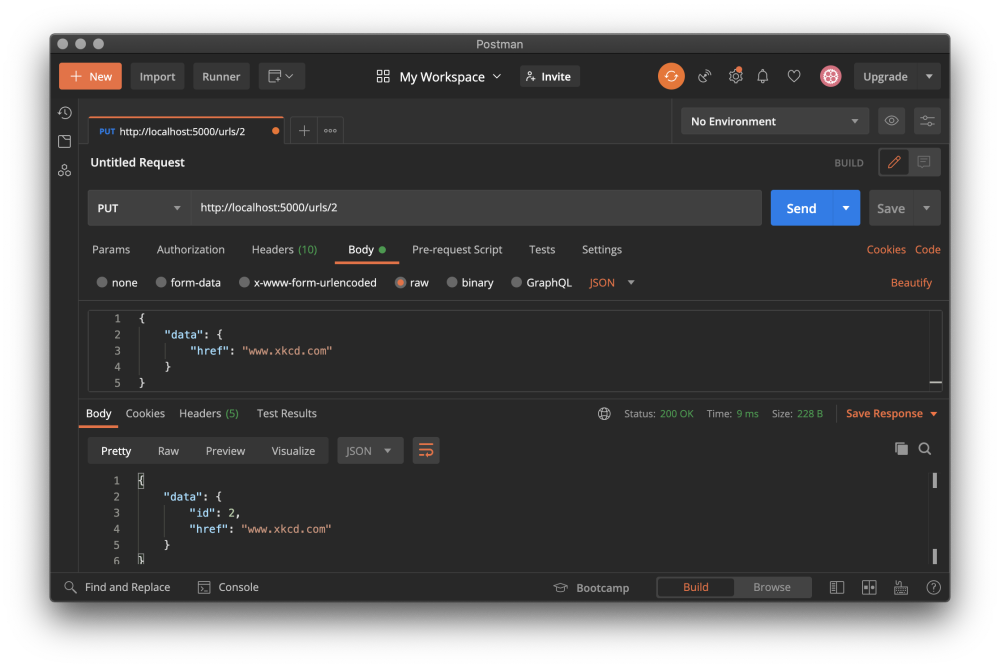
1:1



**Update**

The following Postman screen shot shows a PUT request to /urls/:urlId, and the response from the server.

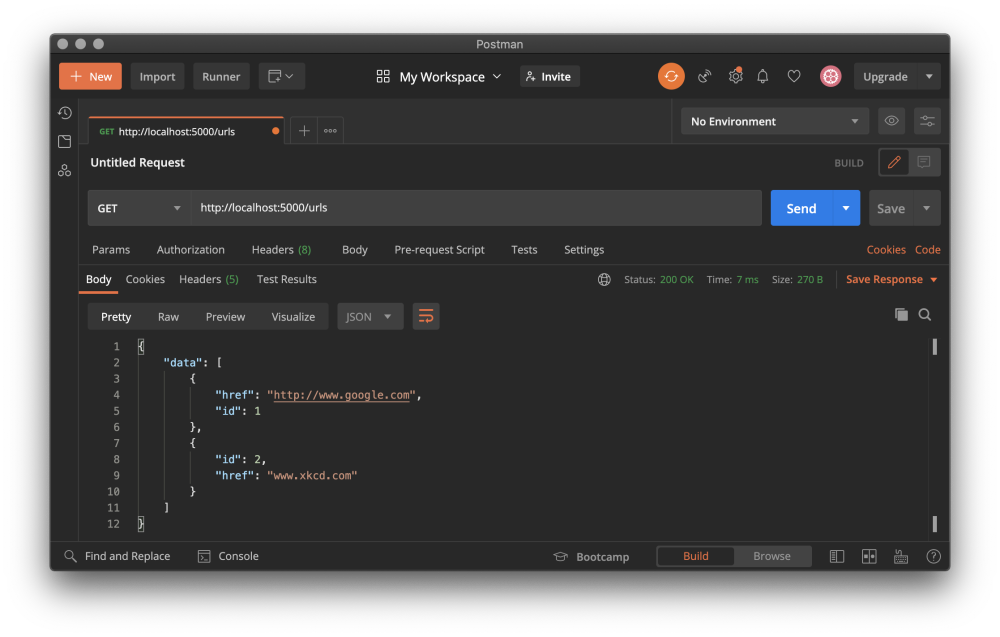
1:1



**List**

The following Postman screen shot shows a GET request to /urls, and the response from the server.

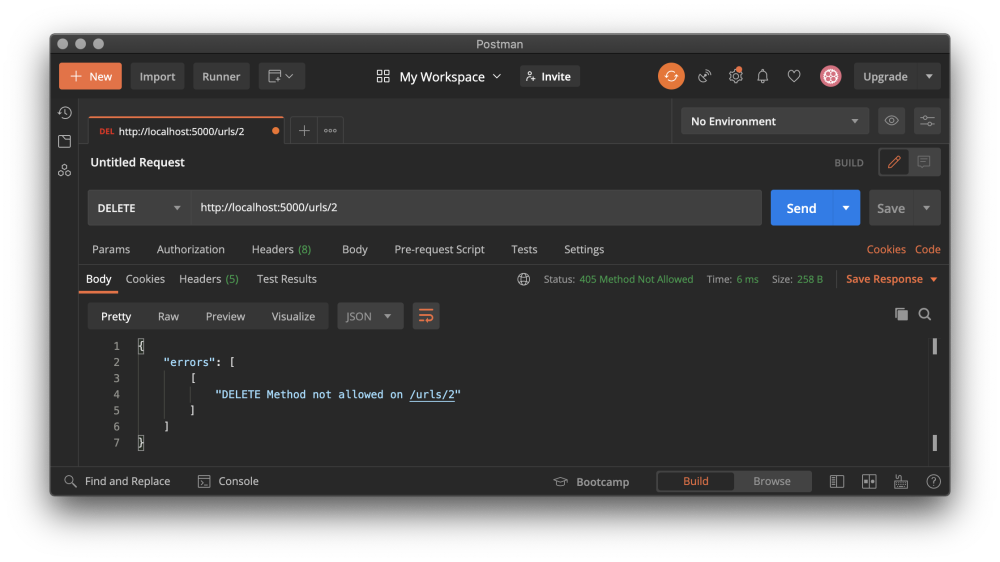
1:1



**Delete**

The following Postman screen shot shows a DELETE request to /urls/:urlId, and the response from the server.

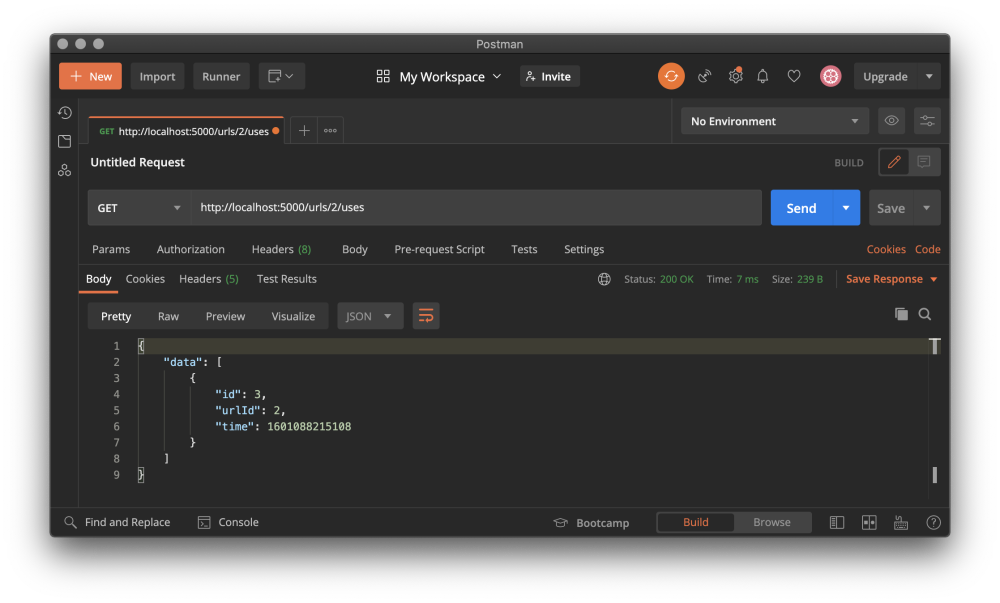
1:1



**List Short URL Uses**

The following Postman screen shot shows a GET request to /urls/:urlId/uses, and the response from the server.

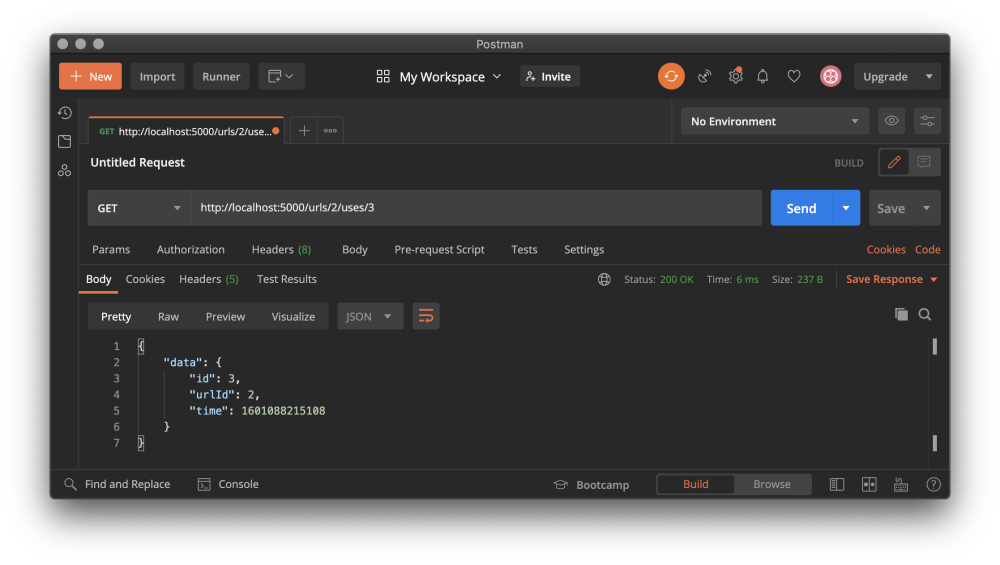
1:1



**Read Short URL Use**

The following Postman screen shot shows a GET request to /urls/:urlId/uses/:useId, and the response from the server.

1:1



The service should return a 404 error if the :urlId and :useId are mis-matched. For example, if you send a GET request to /42/uses/79 and useId 79 is NOT associated with urlId 42 the server should respond with 404.

**Create routes and handlers to create, read, update, delete, and list use metrics related to short urls**

You will need to create the following API endpoints for the uses resources:

| **HTTP Verb** | **Path** | **Description** |
| --- | --- | --- |
| GET | /uses/:useId | retrieve a use metric by id |
| DELETE | /uses/:useId | delete a use metric by id |
| GET | /uses | retrieve a list of all use metrics |

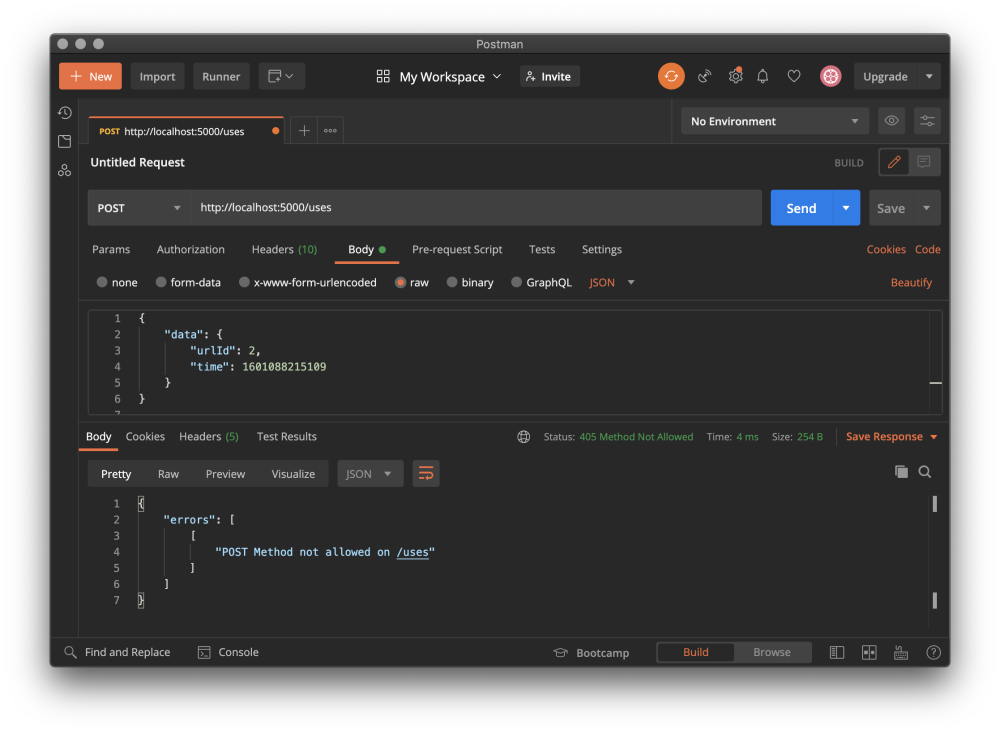
The uses resources have a path of /uses and are a record of every GET request for a specific short url.

**Create**

Creating use records through the API is not allowed. Use records are created as a side-effect of a GET request to /urls/:urlId.

The following Postman screen shot shows the data posted to /urls/:urlId, and the response from the server.

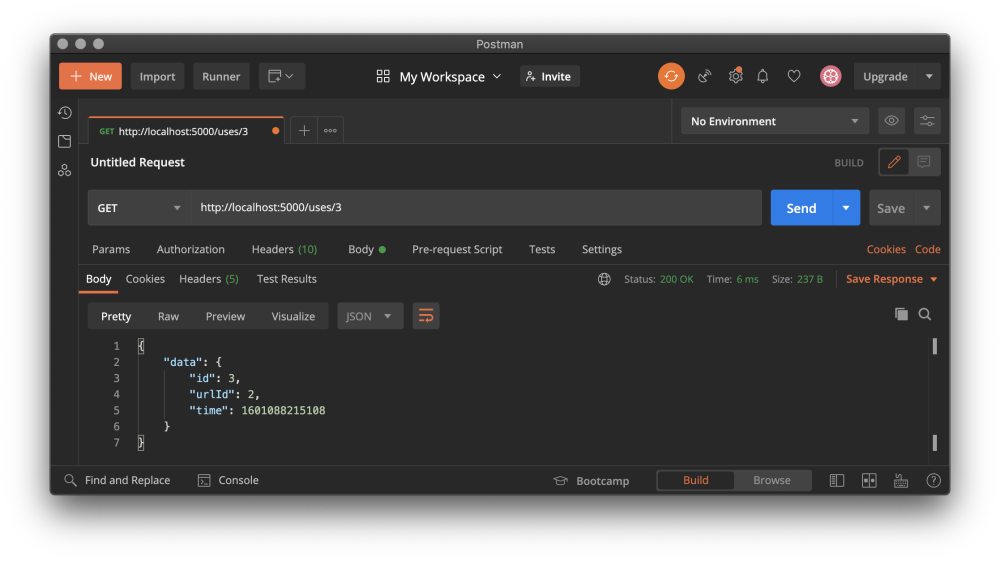
1:1



**Read**

The following Postman screen shot shows a GET request to /uses/:useId, and the response from the server.

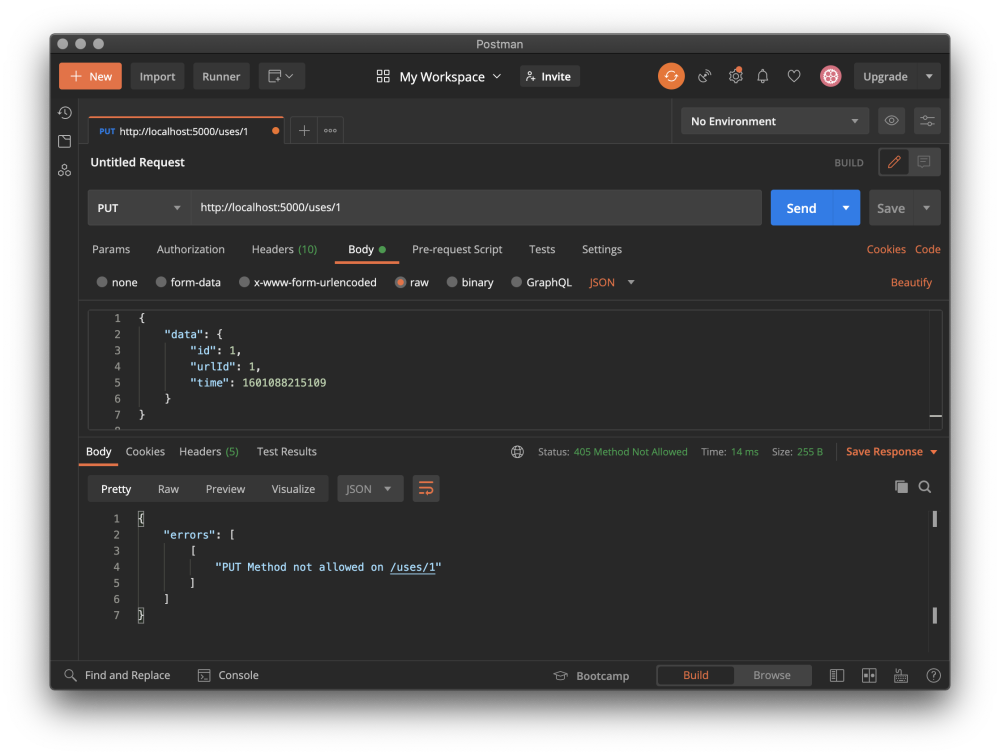
1:1



**Update**

The following Postman screen shot shows a PUT request to /uses/:useId, and the response from the server.

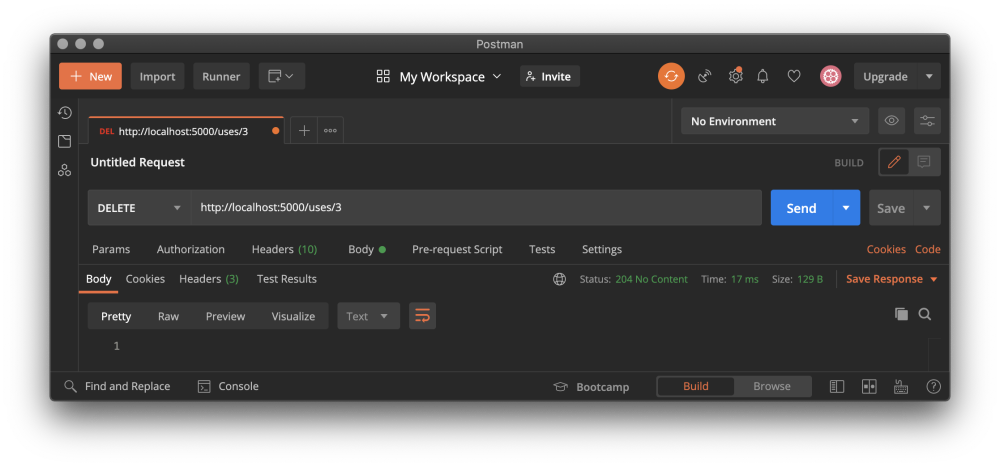
1:1



**Delete**

The following Postman screen shot shows a DELETE request to /uses/:useId, and the 204 response from the server.

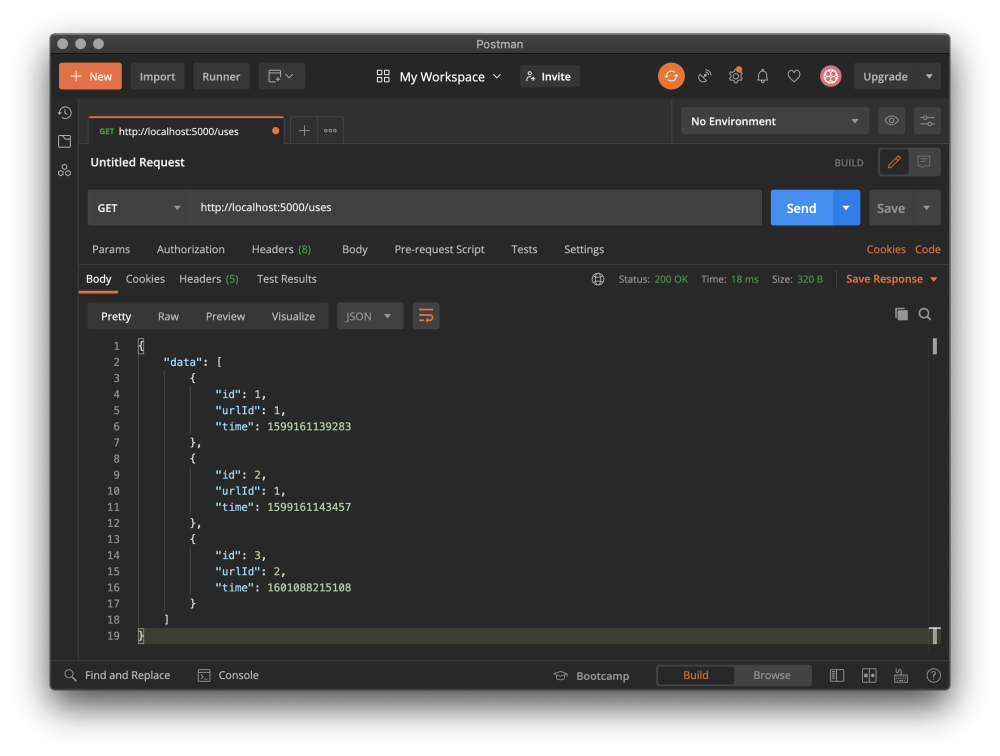
1:1



**List**

The following Postman screen shot shows a GET request to /uses, and the response from the server.

1:1



**Handle errors properly**

* Return a 404 error for any non-existent path or resource
* Methods that are not allowed should return 405 (e.g., a DELETE request sent to /urls/:urlId)

**Saving data**

There is no database in use for this project. All changes are stored in-memory.

The short url data is exported from /src/data/urls-data.js.

The use data is exported from /src/data/uses-data.js.

There is some existing data in each file to give you a starting place.

Add and remove data from the arrays using .push() and .splice() respectively.

When you restart your server, any changes made to these arrays will be lost.

**Assigning ID's**

ID's are often assigned by the database. Since your API is not connected to a database, you can use array.length + 1 to assign ID's, as follows:

const newUrlId = urls.length + 1;

const newUseId = uses.length + 1;

However, note that this method of assigning ID's to your database records is **NOT** recommended in practice and is only used in this assignment for simplicity so you can focus on building the API. Later on in the backend module, you will learn about industry-standard databases and better ways to assign ID's to database records.

**Assigning time property**

Use Date.now() to assign the time property of uses.