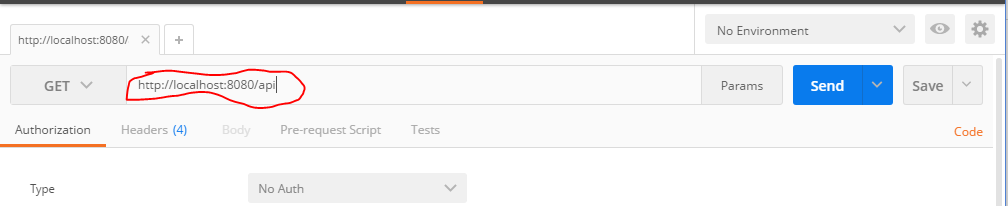
Hi, I thought it may be helpful to use postman to test things out and also to see what is in our database. Again this is totally optional, if you are having issues with this you can completely forget about it.

The main goal of this readme is for you to have a sneak peak at what is in the database and also to see what you are receiving when you send a post or a get using http request. Btw when we say we are using a GET request, that means we are primarily asking the database what to return to us(JSON objects). When we are using a POST request, we are feeding the database with data(JSON objects) and it is giving us data(json objects) in return

# Getting Started

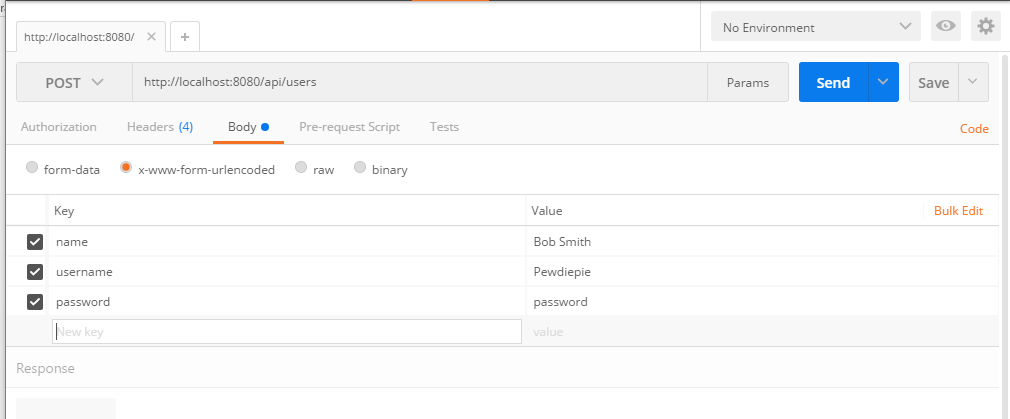
1. Download postman <https://www.getpostman.com/>
2. Open up node, and type node server.js in your project folder since we will want the server to run while using postman.
3. Once you open postman, you will notice how your command line will display a message stating “Welcome to our app!”
4. Next you will want to find the searchbar right by a button that usually either says “GET” or “POST”. In this search bar we will be writing our HTTP request, the request you make with angular. We will always start by writing http://localhost:8080/api , and we will add endpoints after /api to go to whatever we are routing to, whether that’s users, login, or items etc.



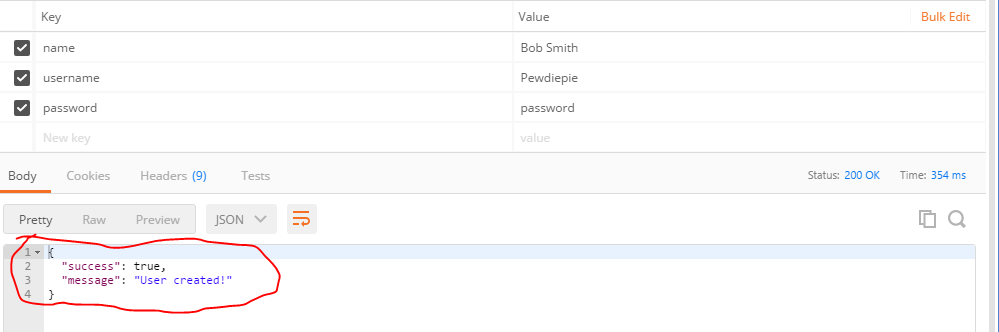
# Users Endpoint: Sign up

Lets start by looking at our USERS endpoint. Lets have a user by the name of Bob sign up.

1. If not already, change your request button on the left of your searchbar to “POST”. The reason why it should be on POST is because we will be FEEDING the db with user info in JSON object format.
2. Next type <http://localhost:8080/api/users> in the searchbar. This will change our endpoint to users.
3. Click the body tab below the search bar and and you will see there are two column fields, “Key” and “Vaule”.
   1. Note we are sending JSON objects, and each object contains KEY and VALUE pairs.
4. For each key you want a string like “Username”, and value would just be the value to that string!
5. Lets say our user is Bob Smith, and he wants his username to be Pewdiepie and his password is password(not recommended). We would type our key and values as shown below:



1. Now hit send and notice what you receive back? A json object with keys and values

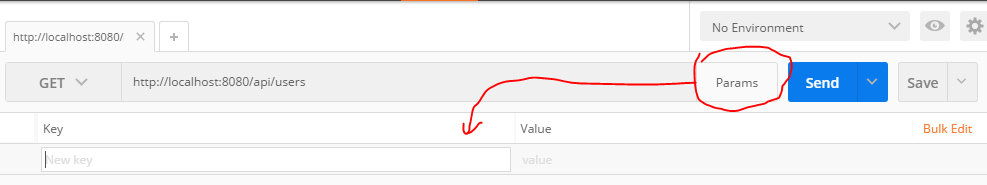


This is what you will be receiving when you type your angular script. One JSON object with two key-value pairs. Looks like our signup was a success!

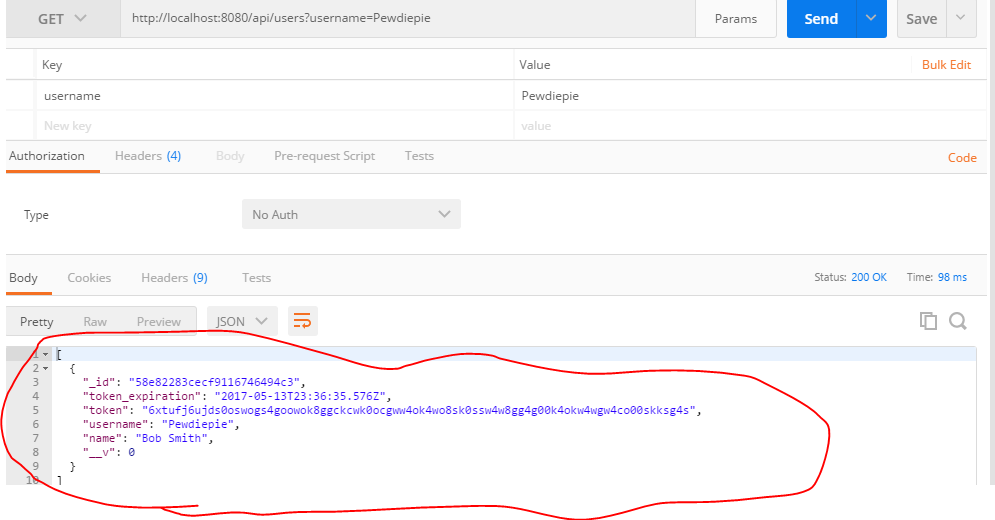
# Users Endpoint: find user

Now lets find our user to make sure our user is in the database. Now what you want to do is change the http request button to “GET”.

1. Now to the right of the search bar you will need to click the PARAMS tab. This will enable parameters so you can tell the server what JSON object to look for based on its key-values.



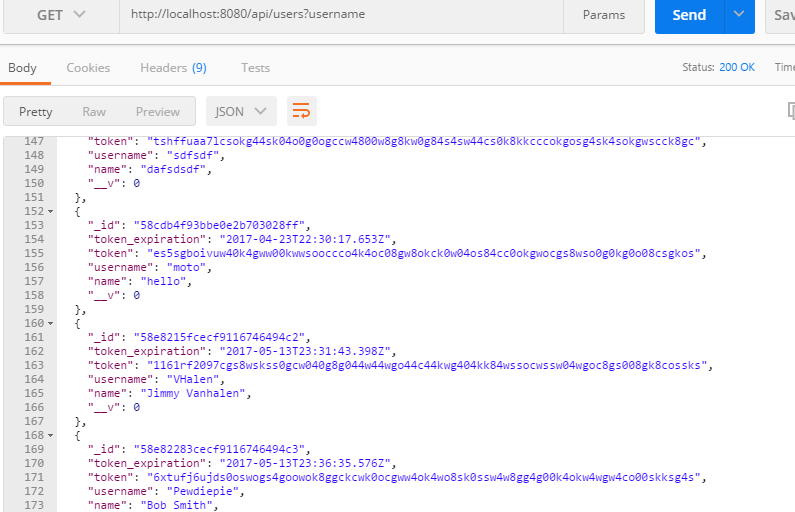
1. Type “username” for the key and type “Pewdiepie” for the value, and click send and you should see the Json object returned with “key-value” object for our user Pewdiepie!



* 1. You may ponder what are all the extra keys we didn’t type in? token\_expiration is when the user will automatically log off from the app(ex a user from cougar courses logs off when inactive). “Token” along with “ \_v”S, usually are built in for every thing we post.

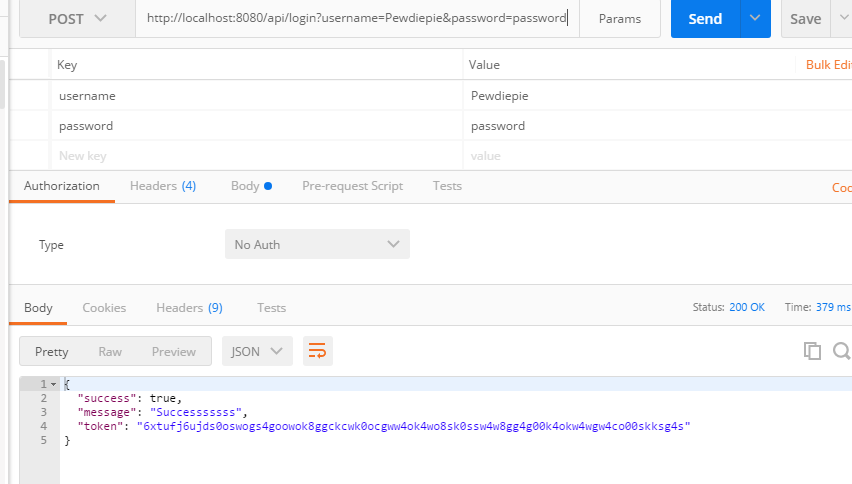
# User Endpoint: find all users

We can also just look at all the users in the database, by not passing in any params. Click params again to disable parameters, make sure your http link is back to <http://localhost:8080/api/users> Click send.

And here are all the users in our db. 

# Login Endpoint:

1. Now that our user is registered, lets Login as him. Go ahead change your endpoint from “/users” to :”/login” like “<http://localhost:8080/api/users>” and change your GET button to a POST button. Again click Params to enable params and type the username and password in the key and value fields and click send and see what you get back.

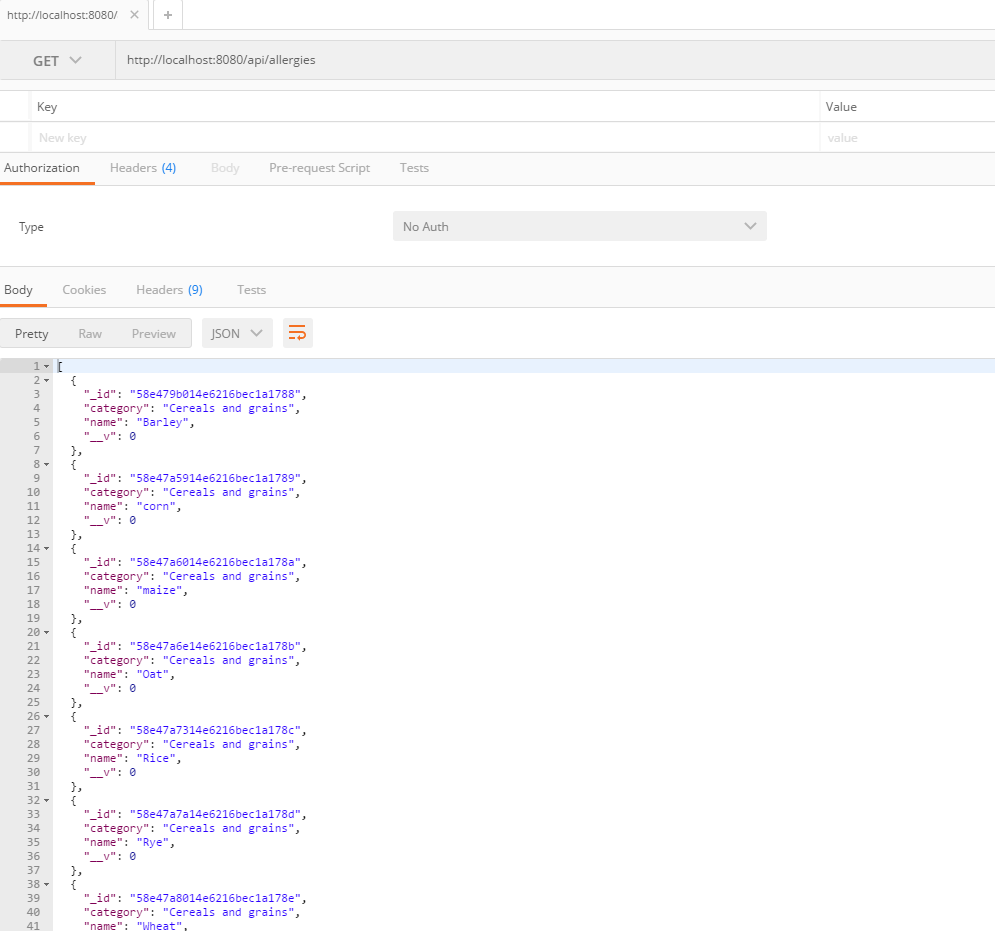


It send us back a JSON object with keys success, message, and token. You will see this if you successfully login.

Now using the techniques from our tutorial, ill briefly take you through finding all the allergens and items.

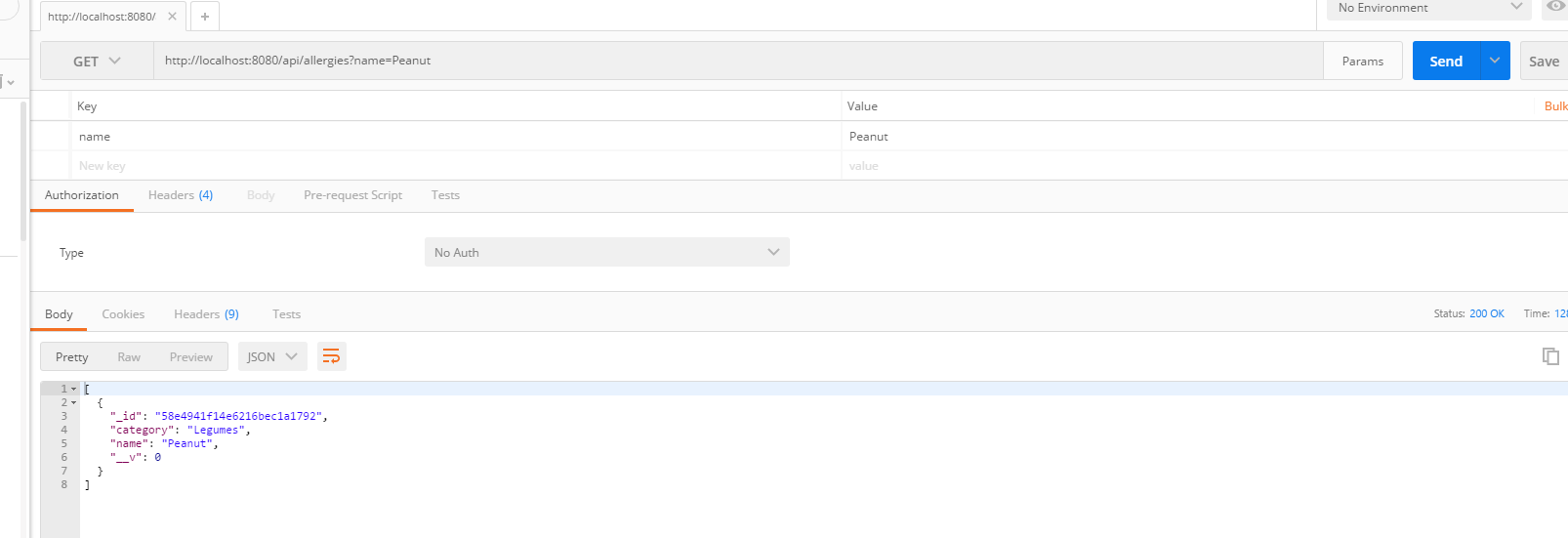
# Allergies Endpoint: List ALL allergens

1. Change the endpoint in the url to “/allergies” like http://localhost:8080/api/allergies
2. Change the POST button to GET, as we want to be receiving from the database.
3. Just click send, and you will receive all the allergens in the database, don’t put any params just yet

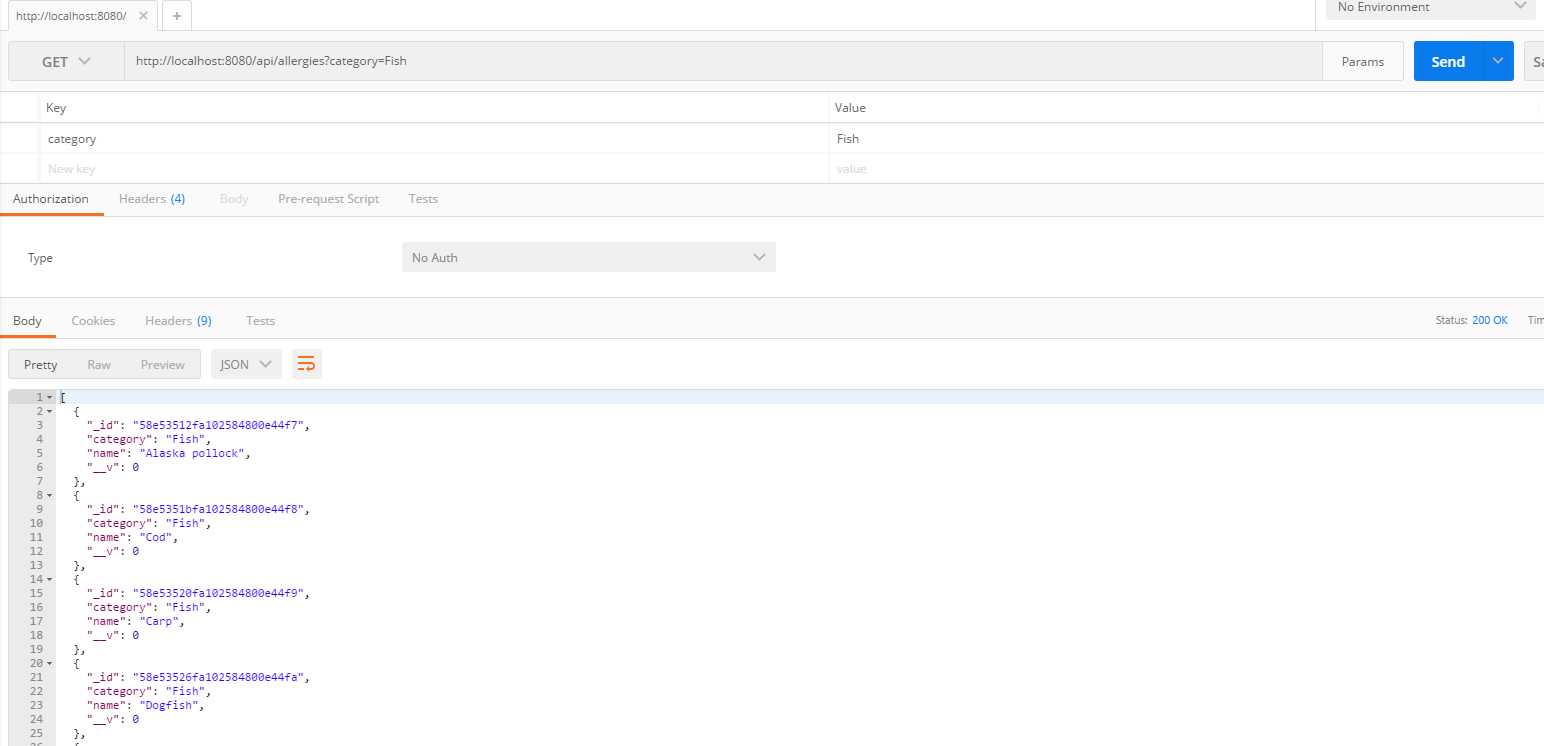
This will typically return category, name, and ID.

# Allergies Endpoint: Find allergins by name( or category)

1. Now lets search a specific allergy. Enable params and type “name” for key and try typing “Peanut”. Now click send



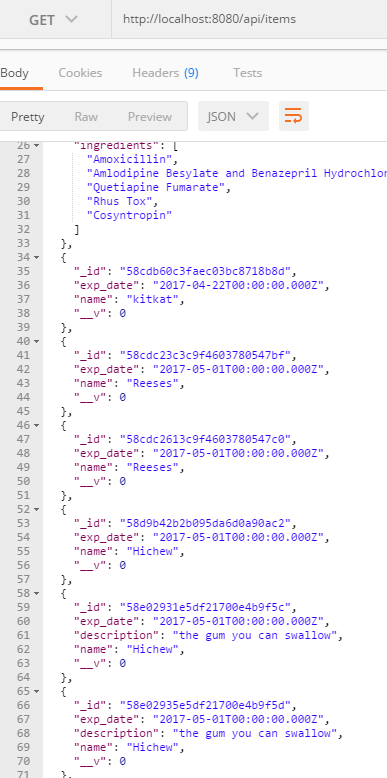
1. You receive back a json object with key-values of \_id, category, and name
2. Now change the name key to category and try “Fish” for a value!

5. You get a list of JSON objects that have key-value pairs of all the type of allergens that fall under the category Fish. You will almost always have \_id, category, and name returned. Though category is optional in the DB, but most likely you will always see it

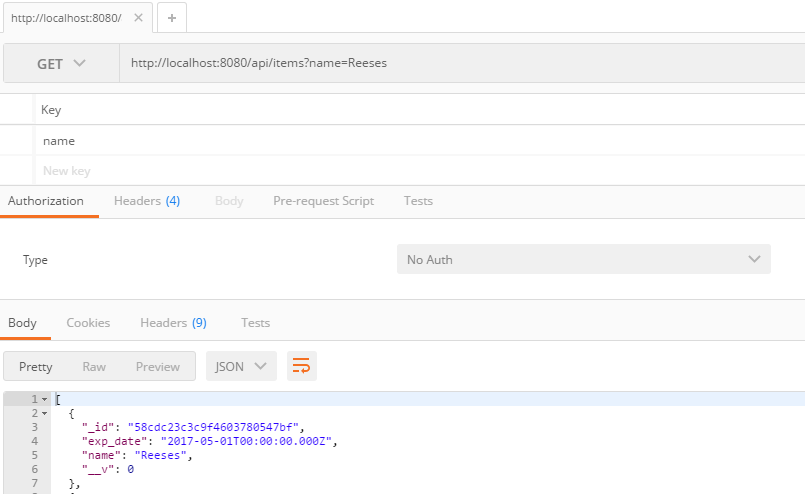
# Items Endpoint: Find all items

Last but not least, if you have been wondering what is in the item database, and what key-values will be returned, now is your chance to see

1. Now change the endpoint to “/items” for example <http://localhost:8080/api/items>.
2. Set the http request button to “GET” to receive all items.
3. Don’t add any parameters just simply click send and you will see a list of items



* As you may have noticed, some items have a description key and some others don’t. That is because Description is OPTIONAL, it is not always required for our database, although recommended.
* Using techniques above, you can search a specific item by typing the name of it in the params.



## Any questions, ask in person☺