Problem Statement

Title: Mini Twitter Coding Challenge

Create the backend for a mini messaging service, inspired by Twitter. It should have a RESTful API where all endpoints require HTTP Basic authentication and generate output in JSON format. Implement the the following basic functionality:

- An endpoint to read the message list for the current user (as identified by their HTTP Basic authentication credentials). Include messages they have sent and messages sent by users they follow. Support a search= parameter that can be used to further filter messages based on keyword.
- Endpoints to get the list of people the user is following as well as the followers of the user.
- An endpoint to start following another user.
- An endpoint to unfollow another user.

In addition to the basic functionality above, choose ONE of the following extra features to implement:

- An endpoint that returns the current user's "shortest distance" to some other user. The shortest distance is defined as the number of hops needed to reach a user through the users you are following (not through your followers; direction matters). For example, if you follow user B, your shortest distance to B is 1. If you do not follow user B, but you do follow user C who follows user B, your shortest distance to B is 2.
- An endpoint that returns a list of all users, paired with their most "popular" follower. The more followers someone has, the more "popular" they are. Hint: this is possible to do with a single SQL query!
- Write a small web interface for your API in JavaScript. Don't use server-side templates (like JSPs). Instead, write a single-page application that interacts with your API via AJAX calls. Feel free to use any modern JavaScript frameworks (e.g., AngularJS, ReactJS, etc.).

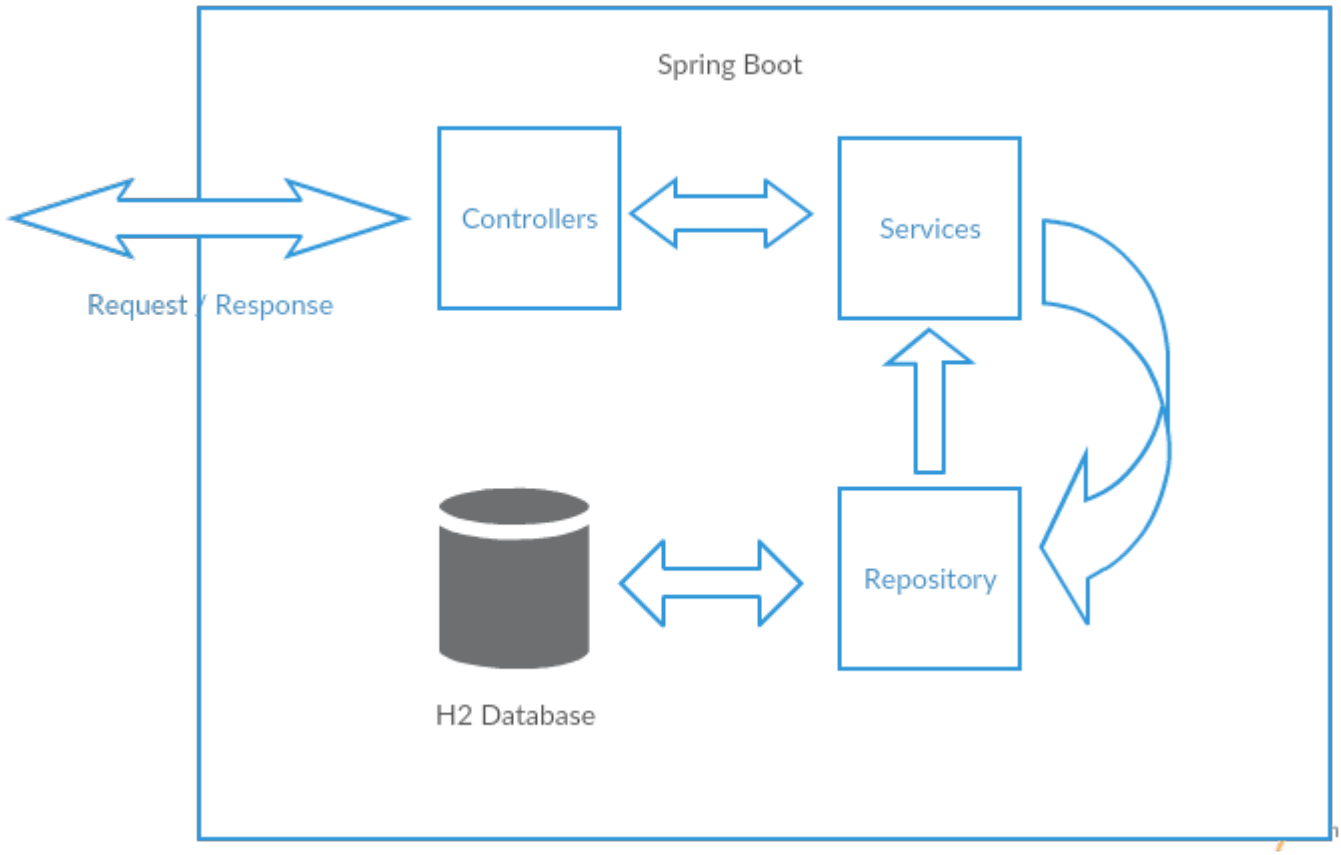
Prologue

I tried my best to complete this project despite of a busy week. I have written comments on most of the methods and I have used eclemma to test the code coverage. The code coverage I have got is *86%* . I took the liberty to change the project build tool from gradle to Maven (I kinda like Maven over Gradle). The SQL files are unchanged. I hope you enjoy my solution.

Solution

The code can be found here. To run the project, clone the project from github and import the project to any IDE and run it. Define your goal as ***clean install spring-boot:run***.A basic authentication has been implemented. The credentials to login are the person's handle name as user name and password. For ex. the login details would be batman/batman.

Architecture



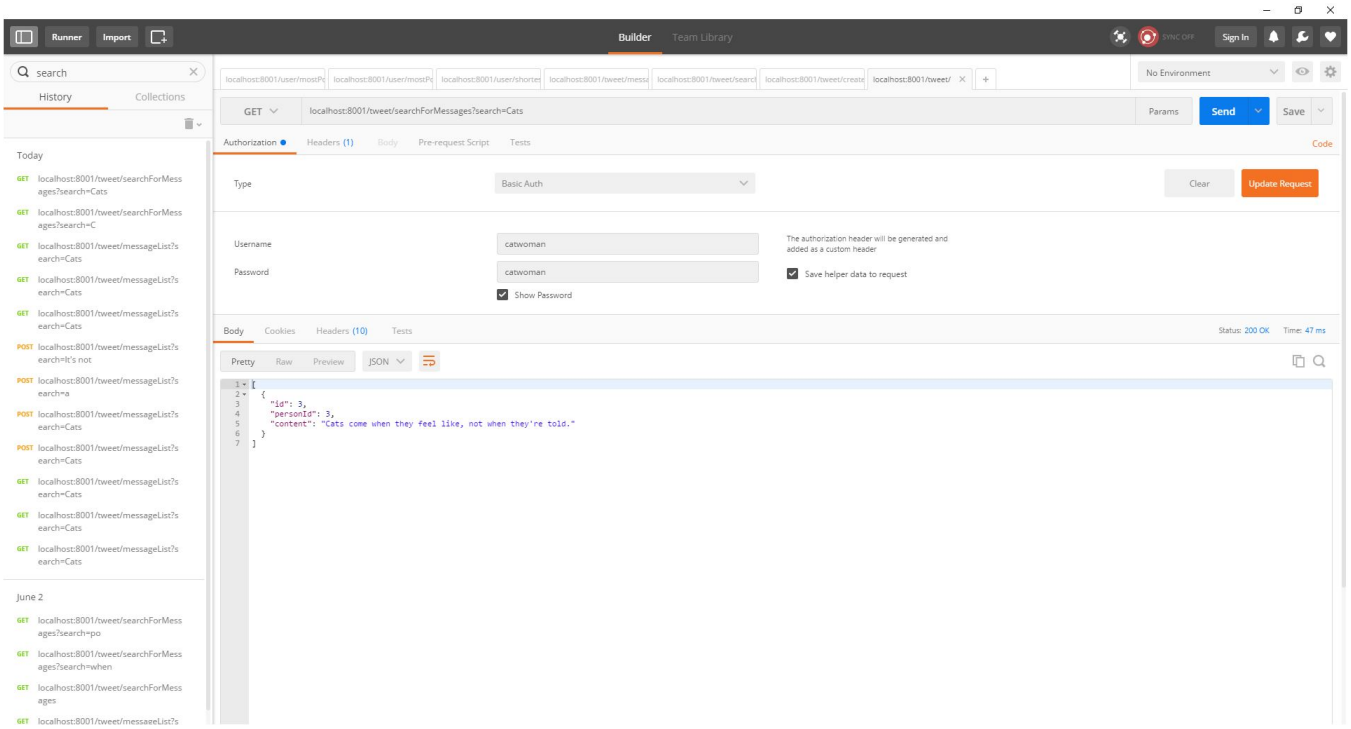
End Points

I have written test cases for the success scenarios, and the test cases for the exceptions has not been tested. The exceptions are database exceptions. Since, we are using H2 database(Sql scripts) the exceptions might not occur. A global exception handler is created to wrap the exceptions.

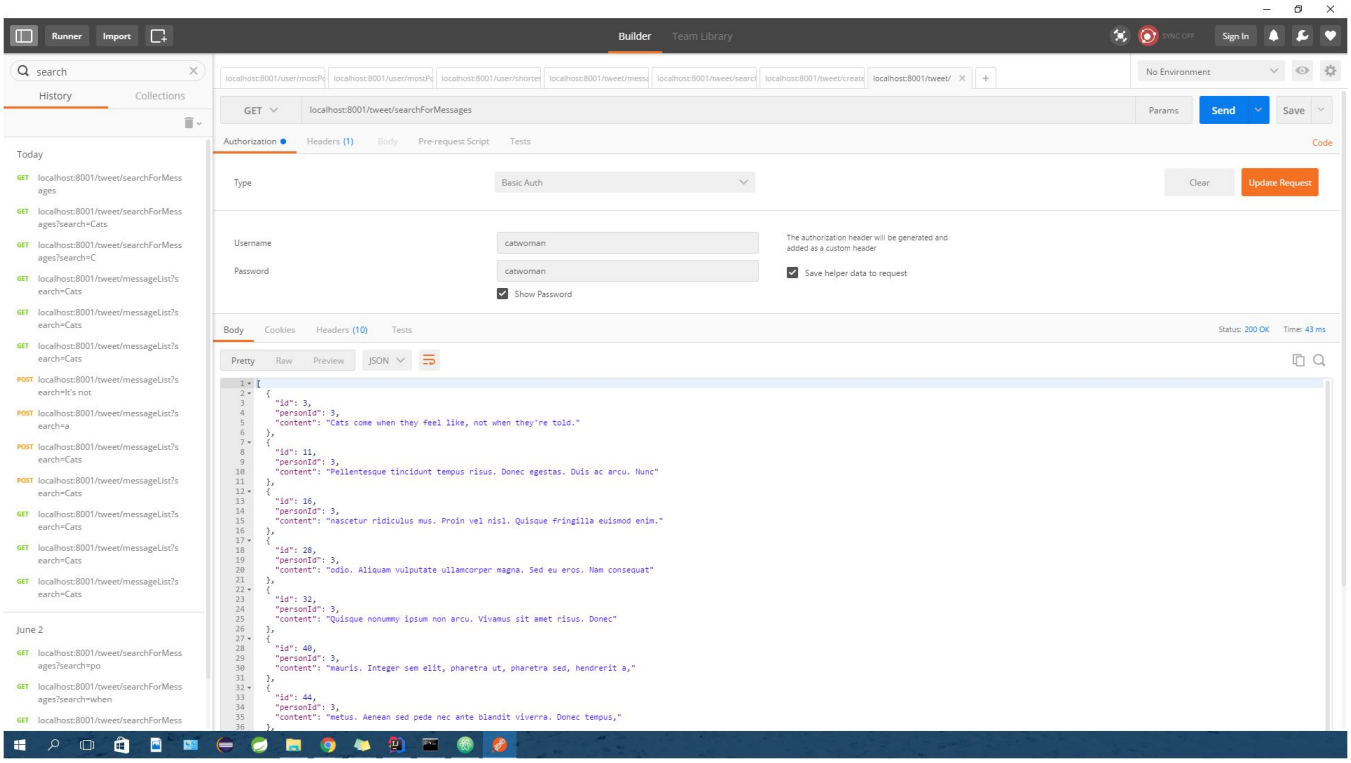
The Server port is *8001*. You can change the port number in *application.properties* file under the resources folder.

Search Parameter

The end point is `http://localhost:8001/tweet/searchForMessages?search=[SearchKeyword]`.
The request type is POST.
Here is a screen shot from postman when a search parameter is specified



When a search word is not mentioned it will display all the messages sent/received by the user.

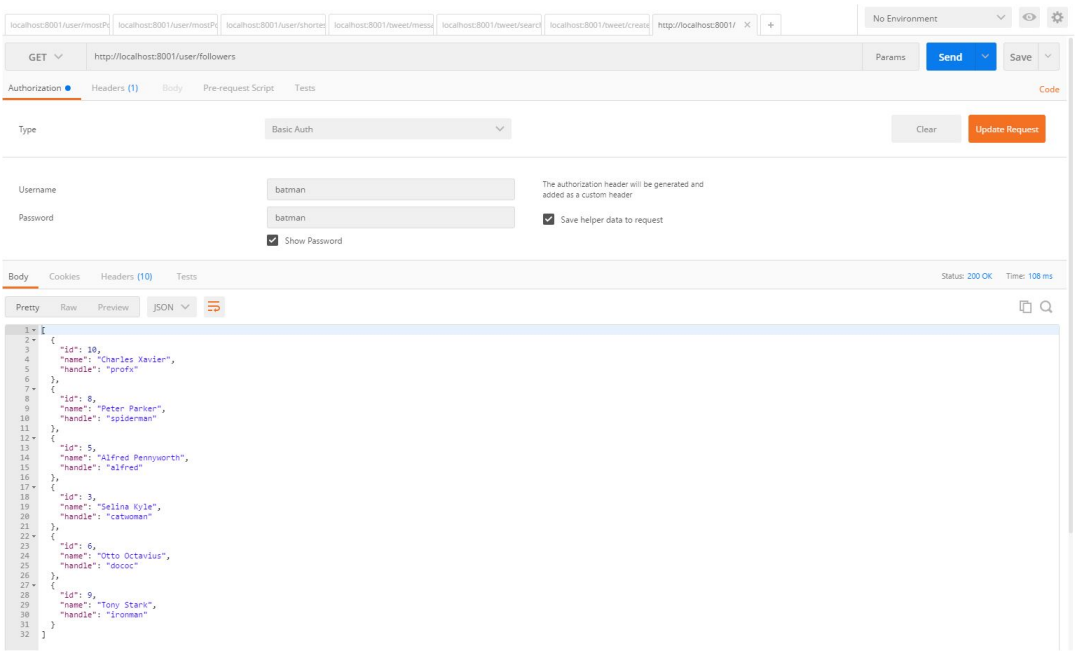


Followers and Following

This end point will get the information related to the followers of the Person and the Persons who he/she is following.

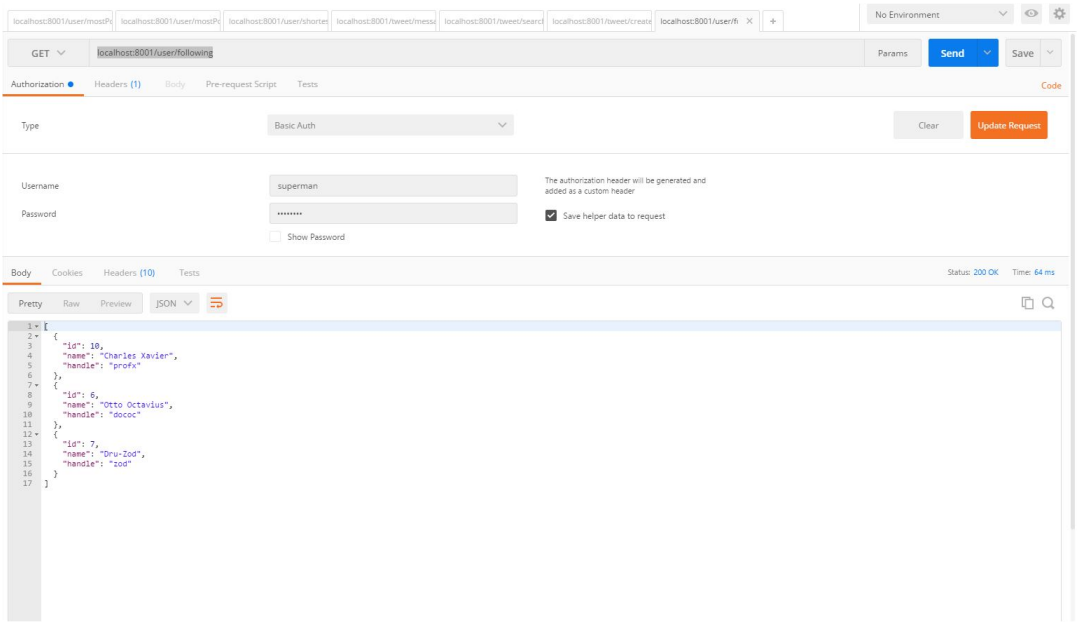
Followers

The end point is `http://localhost:8001/user/followers`. It displays the followers of the person. The request type is GET.



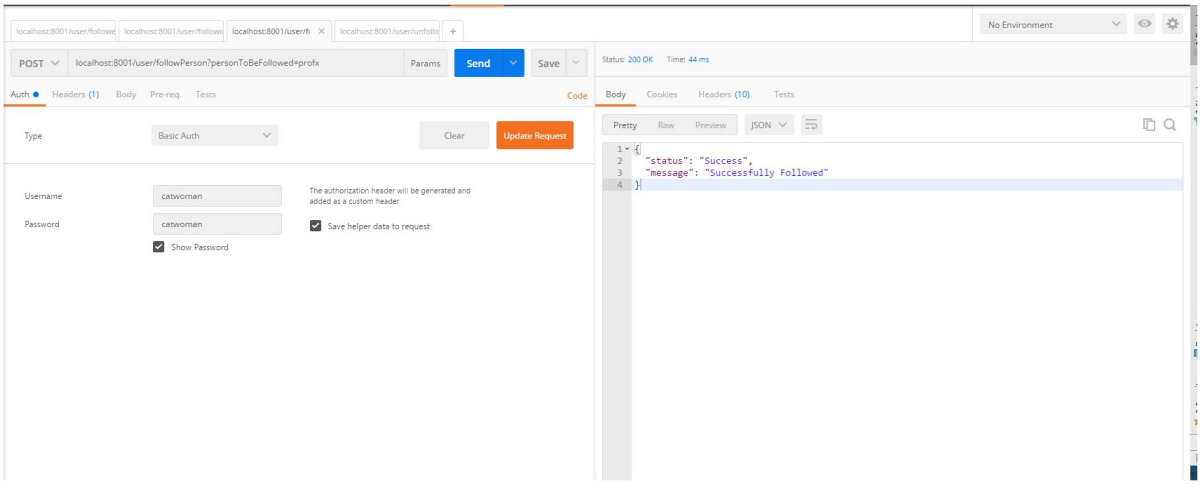
Following

The end point is `http://localhost:8001/user/following`. This will fetch the information about the followers of the person.The request type is GET.



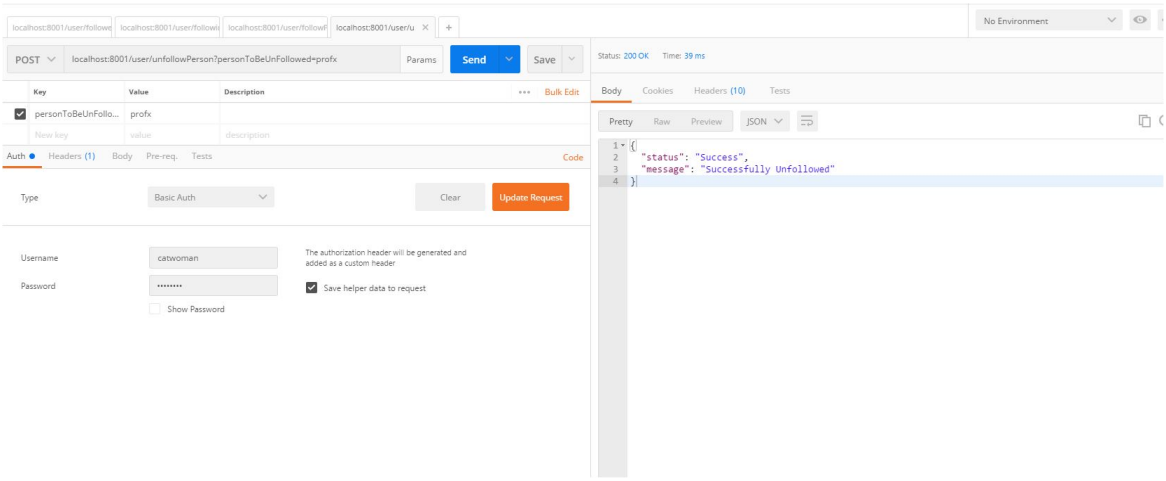
Follow a Person

The end point is `http://localhost:8001/user/followPerson?personToBeFollowed=[Person'sHandleName]`. The request is POST.



Unfollow a person

The end point is `http://localhost:8001/user/unfollowPerson?personToBeUnFollowed=[Person'sHandleName]`. The request type is POST.

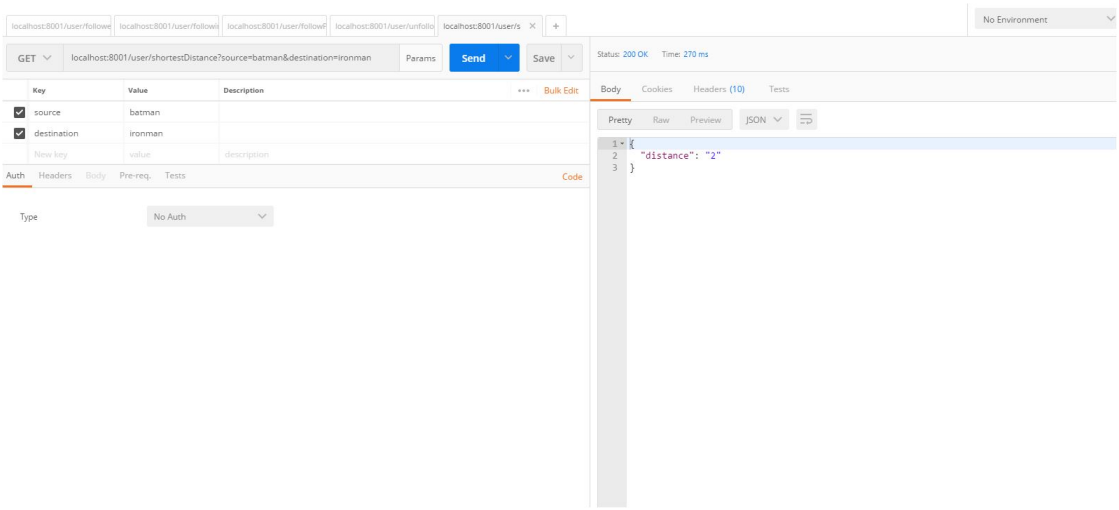


Extra Features

Shortest Distance

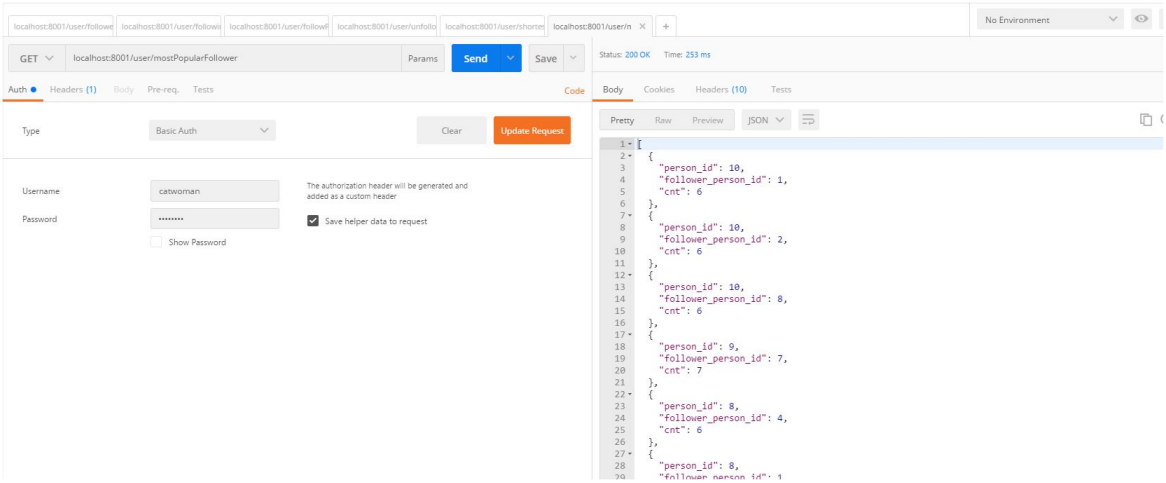
I have used BFS to find the shortest distance between two users'. The code is self-explanatory. The end point is `http://localhost:8001/user/shortestDistance?source=`

`[Source'sHandle]&destination=[Destination'sHandle]`.



Most Popular Follower

The end point is `http://localhost:8001/user/mostPopularFollower`



Assumption

- I have assumed that the handle is unique for every user.