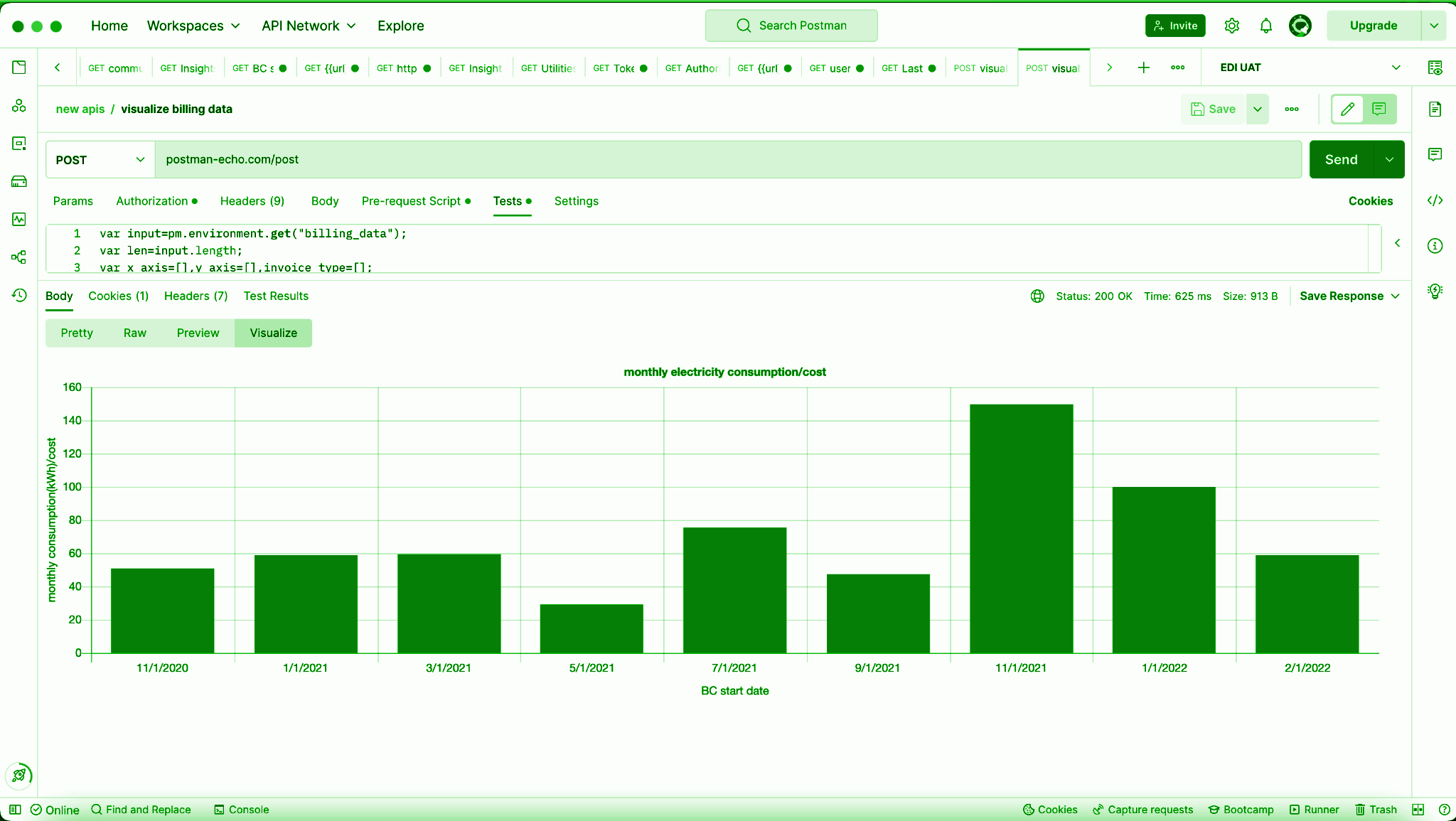
Postman Data Visualization



11.10.2022

**NAVNEET NIPU**

[**INTRODUCTION**](#_c47k4rr0t6td) **2**

[**Basics of POSTMAN VISUALIZATION**](#_q74guzl2b2lz) **3**

[**Automate access token renewal**](#_5b0ij69vylv3) **4**

[**Automate user endpoint creation**](#_tol6s2n8vx1z) **6**

[**Generate web dashboard URL**](#_eeg5bfm5n823) **7**

[**Billing Data visualization**](#_y2jo3nch6kgj) **8**

[**Itemization Data visualization**](#_59u047pd0gaq) **11**

[**Raw data visualization**](#_as4nf734ong0) **13**

[**Disagg data visualization**](#_hqfxpmqewzoc) **15**

[**Weather data visualization**](#_a7yso0o0tnrk) **16**

[**Utility notifications generation timeline**](#_bnt240o8tjnw) **18**

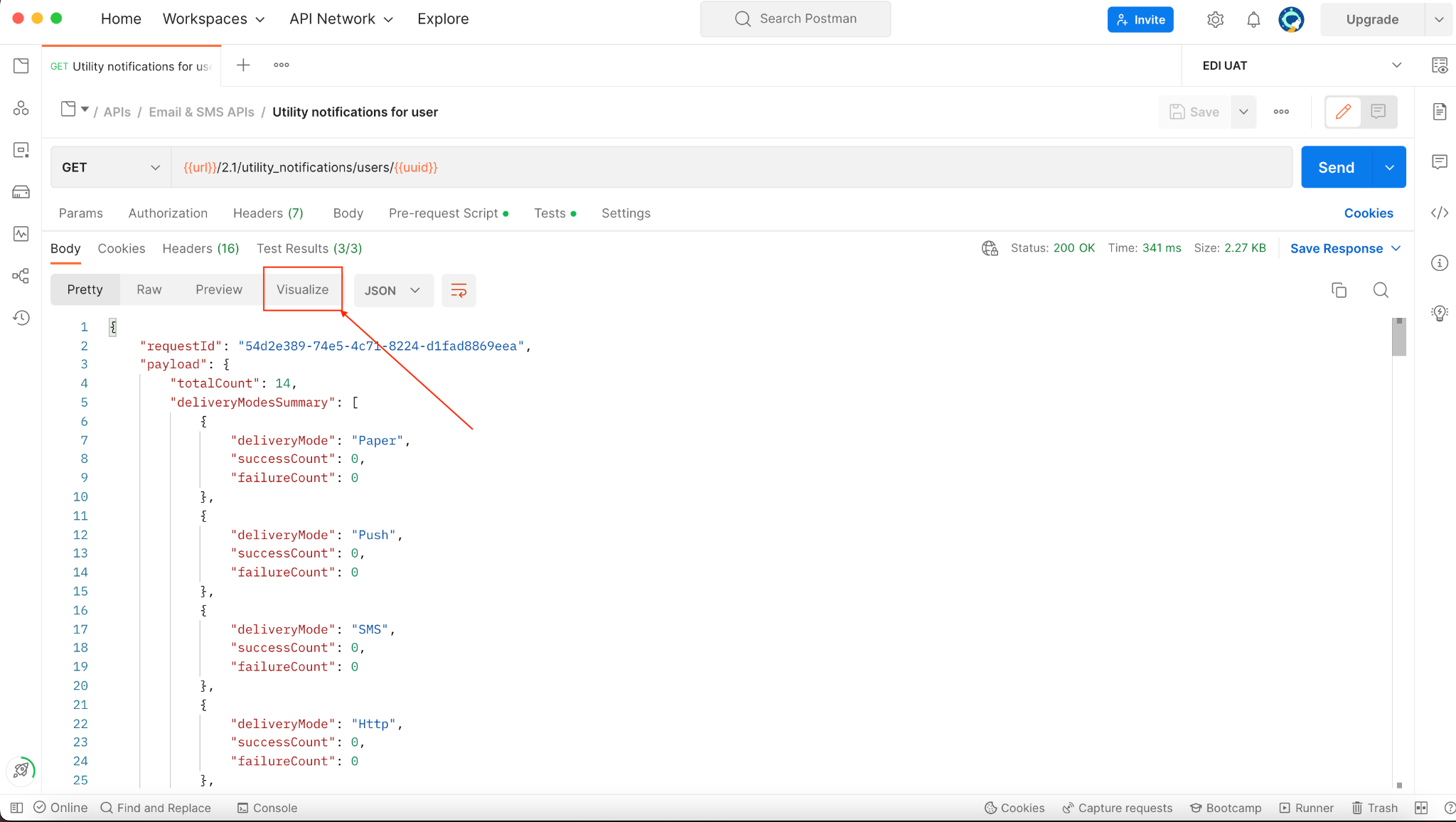
# 

# 

# 

# INTRODUCTION

This project is basically about API tool POSTMAN where we can visualize our backend API data using the VISUALIZE tab in the BODY section.



Using this feature we can visualize our backend API data to quickly inspect the API data.

Here we are going to visualize some of our basic API data like billing data, itemization, raw data, disagg data, weather data, utility notification timeline, etc.

# Basics of POSTMAN VISUALIZATION

Postman provides an option to visualize API data using JAVASCRIPT code.

A tutorial about postman visualization can be found below URL:

<https://learning.postman.com/docs/sending-requests/visualizer/>

# 

# 

# 

# 

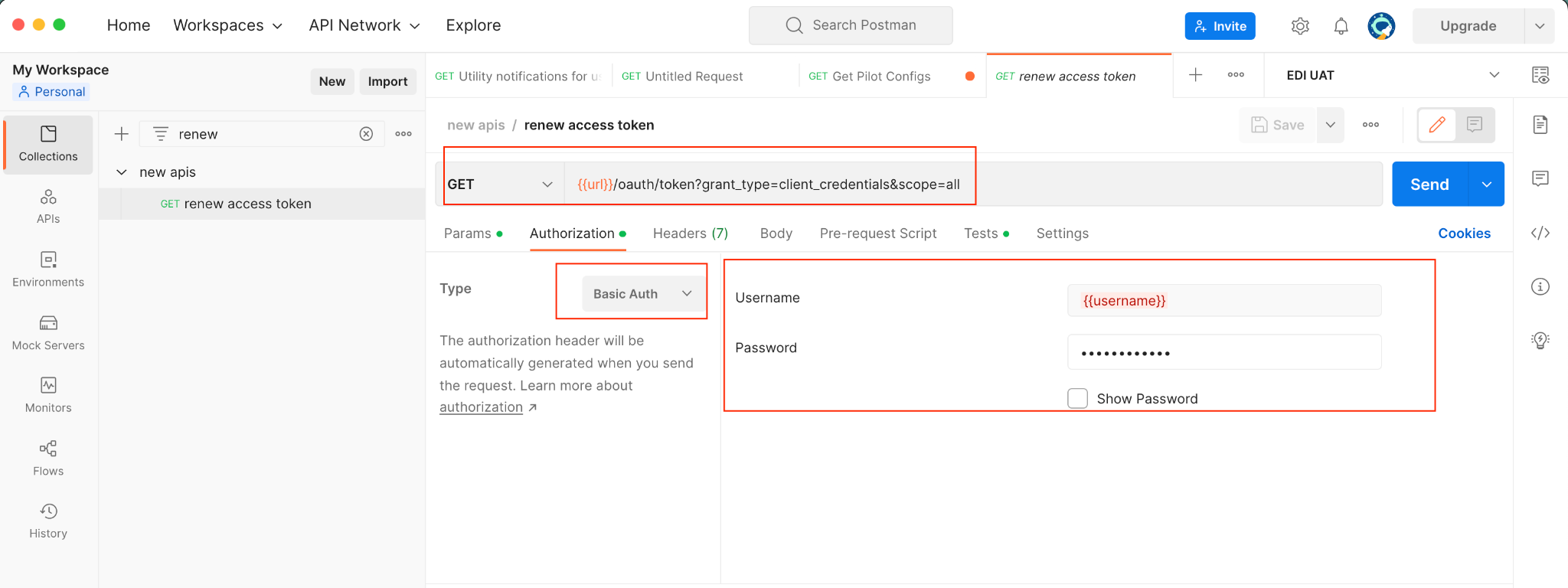
# 

# 

# 

# Automate access token renewal

We can use the below method to renew the access token for an Environment by just hitting the access token URL.

1. Set the below URL and make this GET or POST request:
   1. URL: **{{url}}/oauth/token?grant\_type=client\_credentials&scope=all**
2. In the Authorization tab of the postman, use basic Auth and then enter the username and password associated with the particular environment while client registration(one-time process) like below:
   1. ****
   2. Both username and password can be taken from environment variables which can vary from environment to environment.
3. Use the below-linked code in the **Tests** section which will perform some basic checks and store the renewed access token as **“accessToken”** in the particular environment:
   1. <https://github.com/navneetnipu-bidgely/PROJECTS/blob/postman_data_visualizer/POSTMAN%20DATA%20VISUALIZER/renew%20access%20token/test.js>
4. Now just hit the send link and the access token will be renewed and set into the Environment.
5. Just one step is needed to renew the token.
6. We can also use this as a pre-request script in POSTMAN to get run everytime before we hit any api that will automatically renew the token before sending the api request so that we will never get access token expired error but this is not appropriate to create access token every time we use an API.

# Automate user endpoint creation

We can actually run a script in the pre-request script in postman to populate anything which requires to be populated every time we change the user or environment.

**Whenever we send an API request in postman, it will first run the pre-request script then send the API request, and then performs the Tests script.**

So in this manner, we can use the code linked below in the **pre-request script** to populate the endpoint for a user for APIs which require a user endpoint:

<https://github.com/navneetnipu-bidgely/PROJECTS/blob/postman_data_visualizer/POSTMAN%20DATA%20VISUALIZER/renew%20endpoint/pre_request_script.js>

**Before executing the code, please fill in the required variables ( pm.variables.get() ) for endpoint API in the environment variables as per the code.**

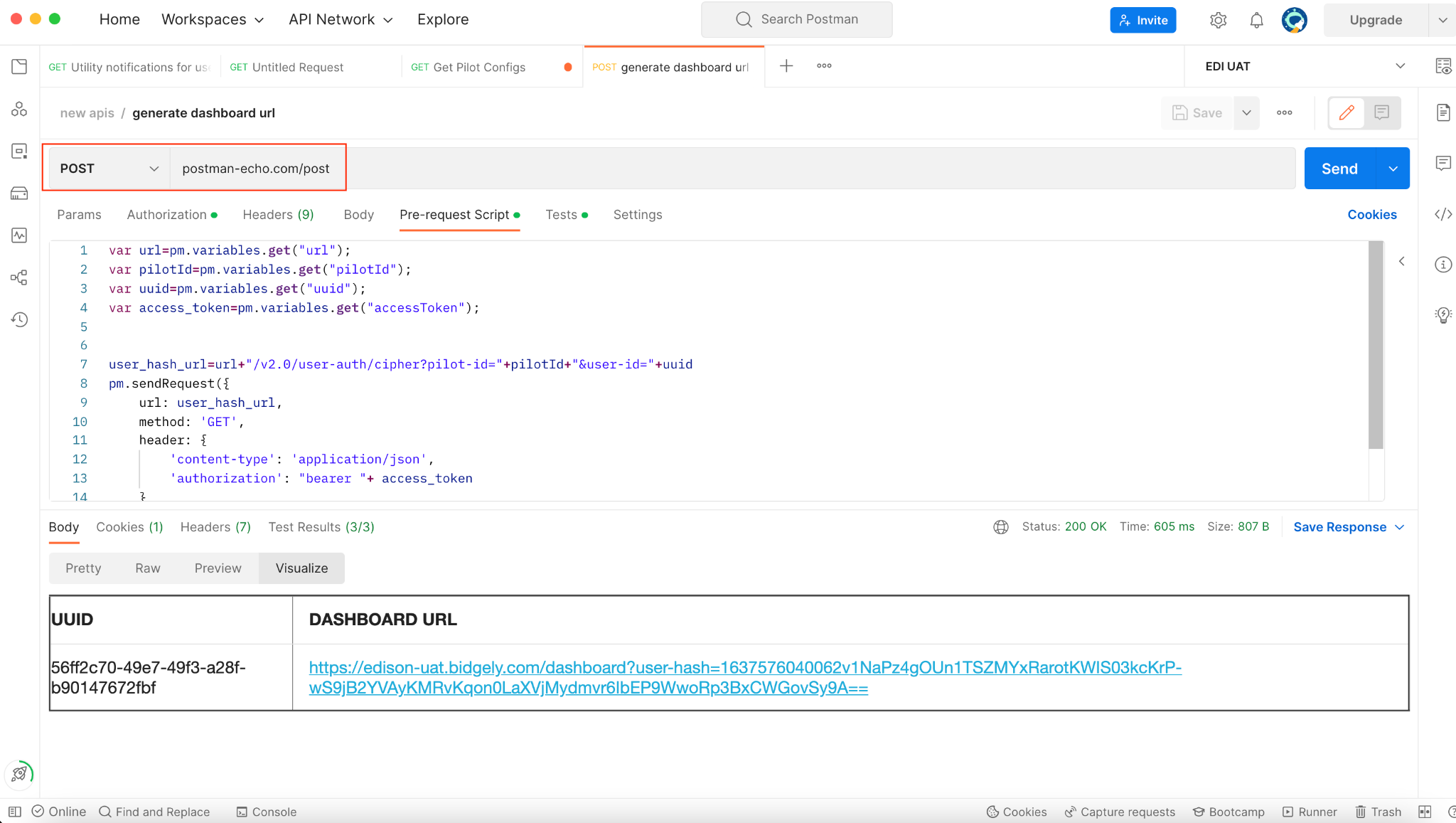
# 

# Generate web dashboard URL

Use the below codes in the pre-request and tests script in postman respectively to generate a user hash based web dashboard URL directly by hitting the send API button.

<https://github.com/navneetnipu-bidgely/PROJECTS/tree/postman_data_visualizer/POSTMAN%20DATA%20VISUALIZER/dashboard_url>

The above code will first populate the user hash for the user in the environment variables using the pre-request script and then attach the base URL with the user hash to generate the URL like below:

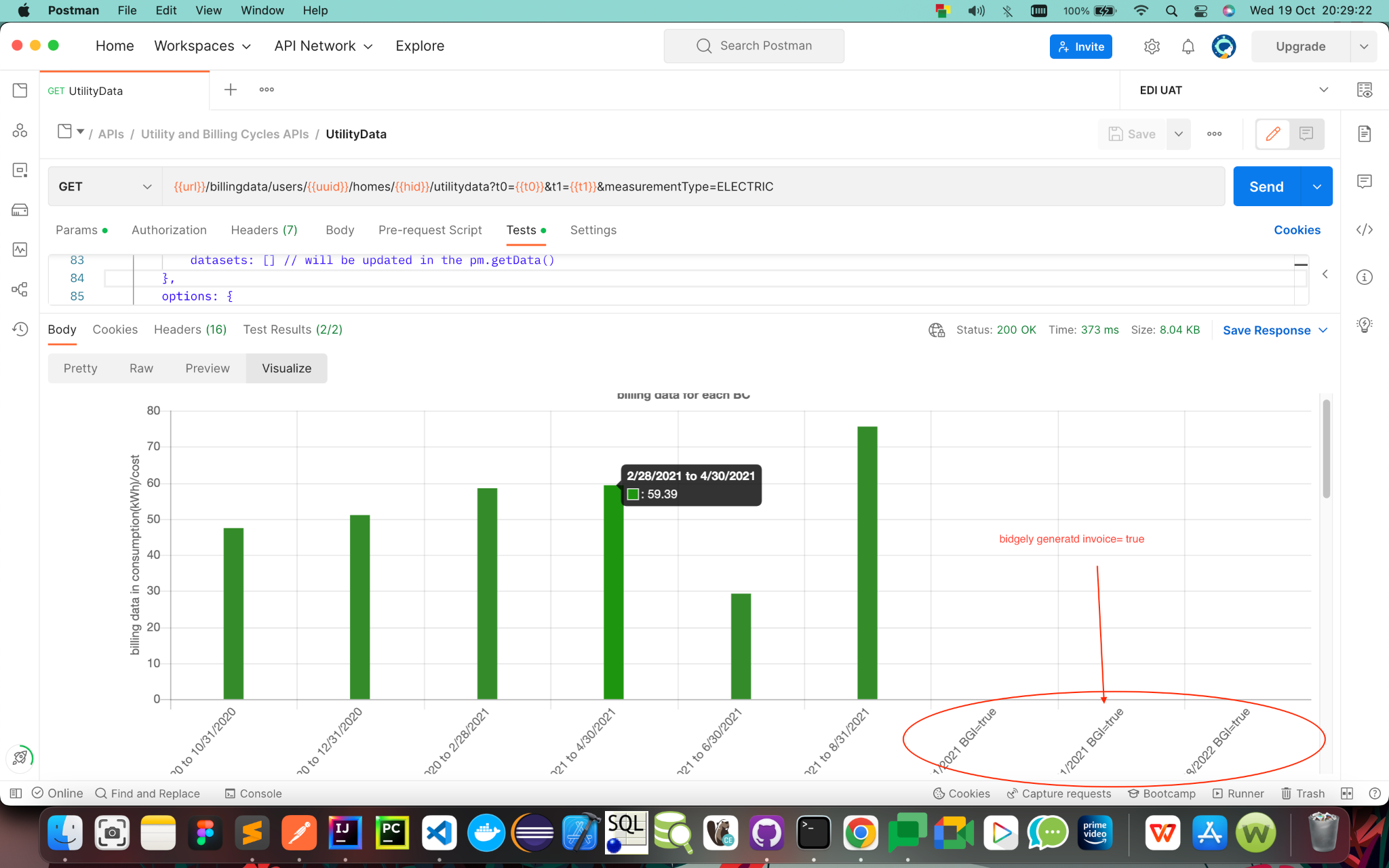
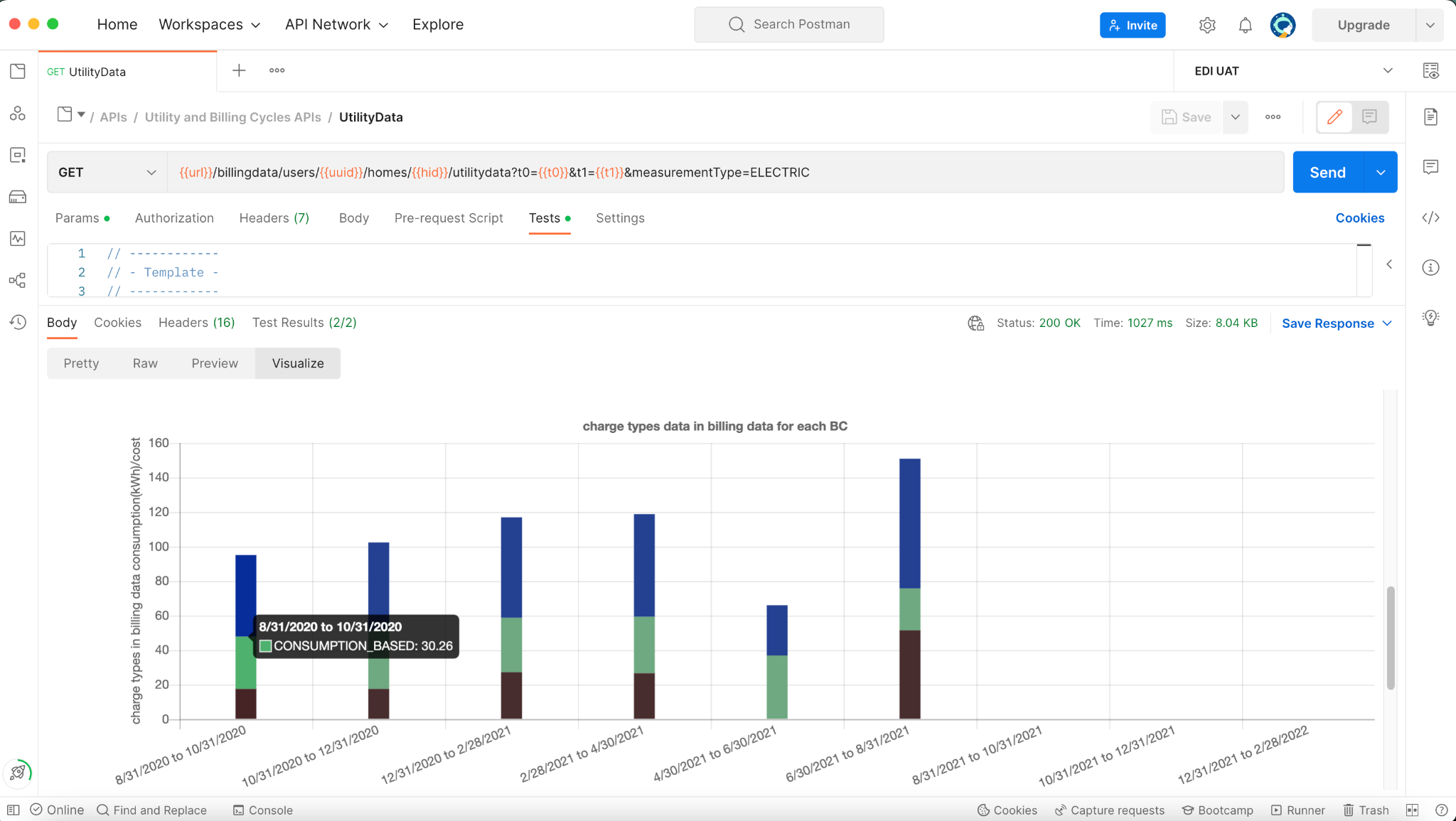


# Billing Data visualization

**CODE URL**: <https://github.com/navneetnipu-bidgely/PROJECTS/tree/postman_data_visualizer/POSTMAN%20DATA%20VISUALIZER/billing_data>

The above code will be used in Tests script in postman for **utility data API** to create the visualization of the response.

This will create two plots:

1. **One for billing data vs billing cycle**
   1. ****
2. **Second one for different charge types vs billing cycle**
   1. ****

# Itemization Data visualization

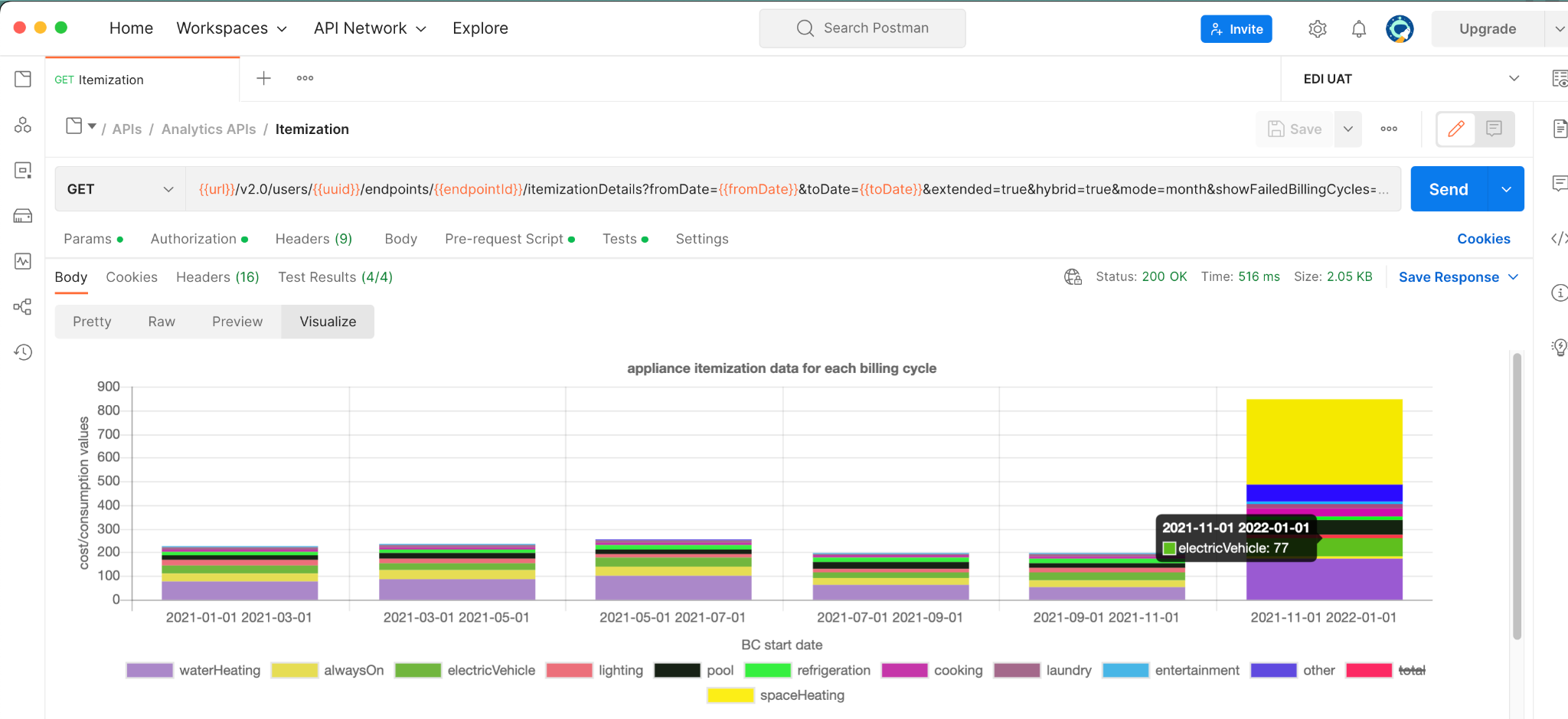
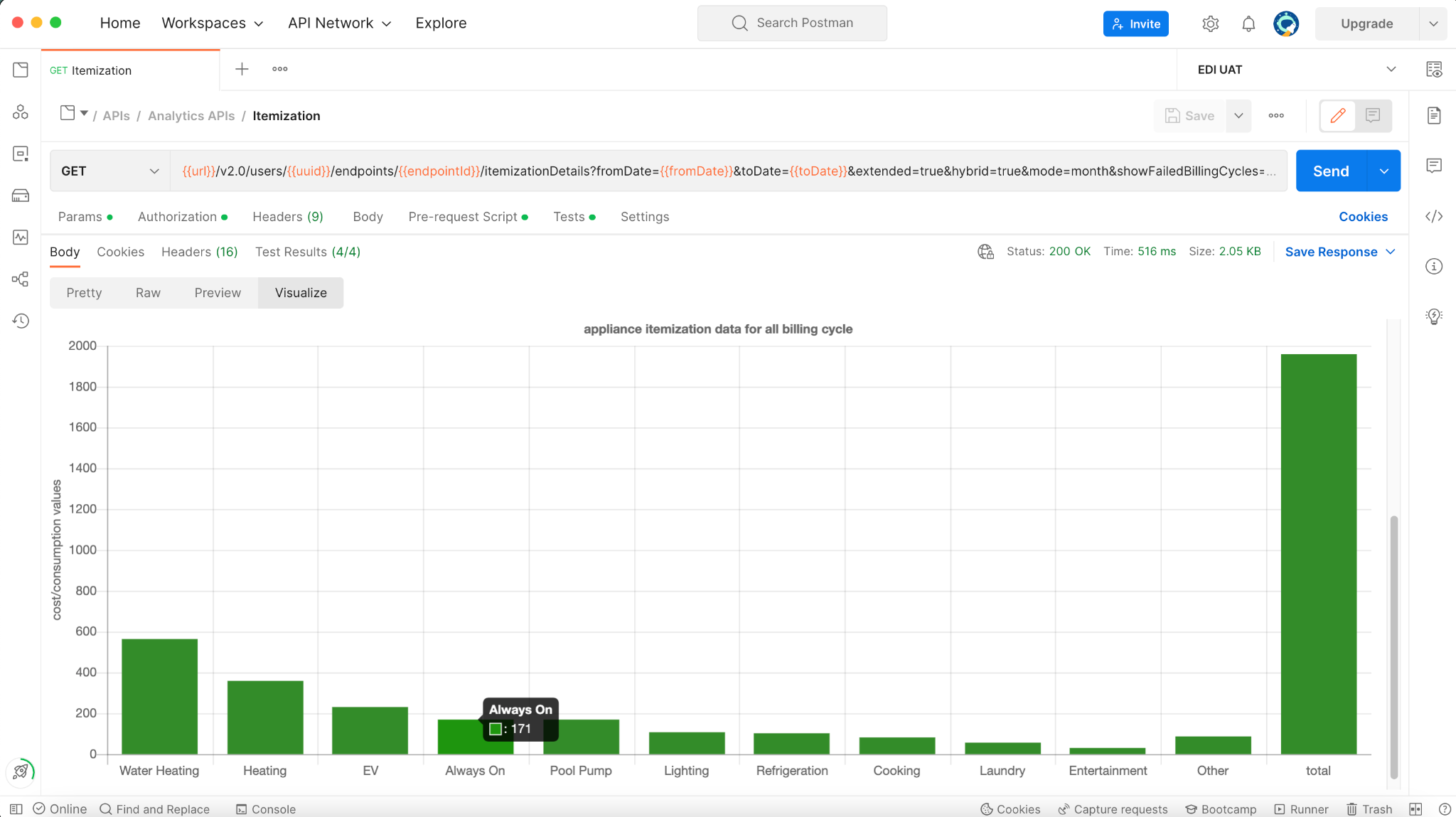
**CODE URL:** <https://github.com/navneetnipu-bidgely/PROJECTS/tree/postman_data_visualizer/POSTMAN%20DATA%20VISUALIZER/itemization>

Using the above code we can generate the visualization of itemization data that we get from the itemization API below:

**{{url}}/v2.0/users/{{uuid}}/endpoints/{{endpointId}}/itemizationDetails?fromDate={{fromDate}}&toDate={{toDate}}&extended=true&hybrid=true&mode=month&showFailedBillingCycles=true&showPercentage=true&round=true&measurementType=ELECTRIC**

The codes should be copied to respective pre-request and Tests script in POSTMAN in itemization API.

This will generate two plots given below:

1. The first plot will show appliance itemization plot vs billing cycles like below:
   1. 
   2. **If there is an error code present in the itemization, then the error code will be shown in the X-axis label along with the billing cycle.**
2. The second plot contains aggregated itemization ( for the overall billing cycle for which the itemization is present) vs appliances.
   1. 

# Raw data visualization

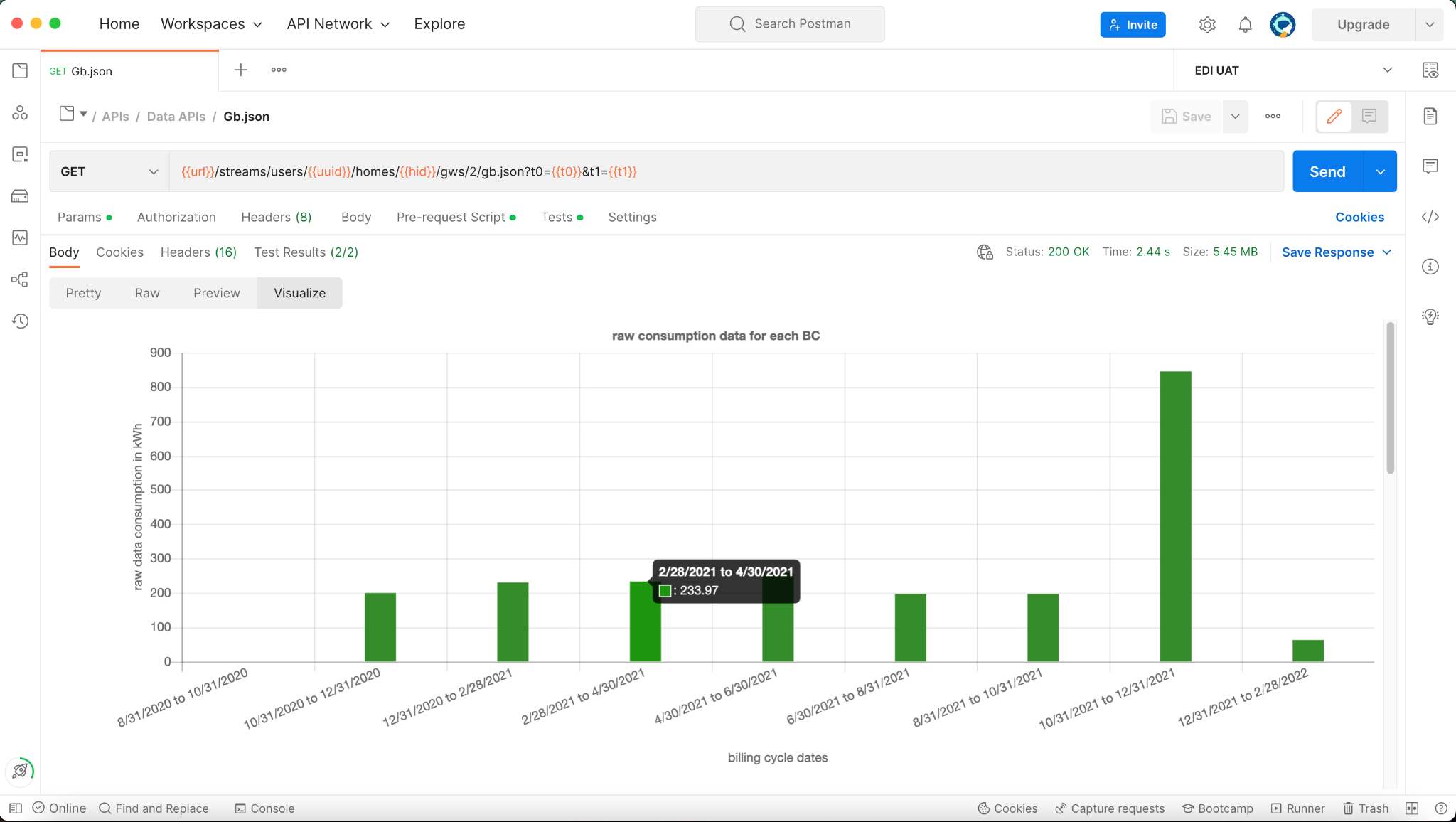
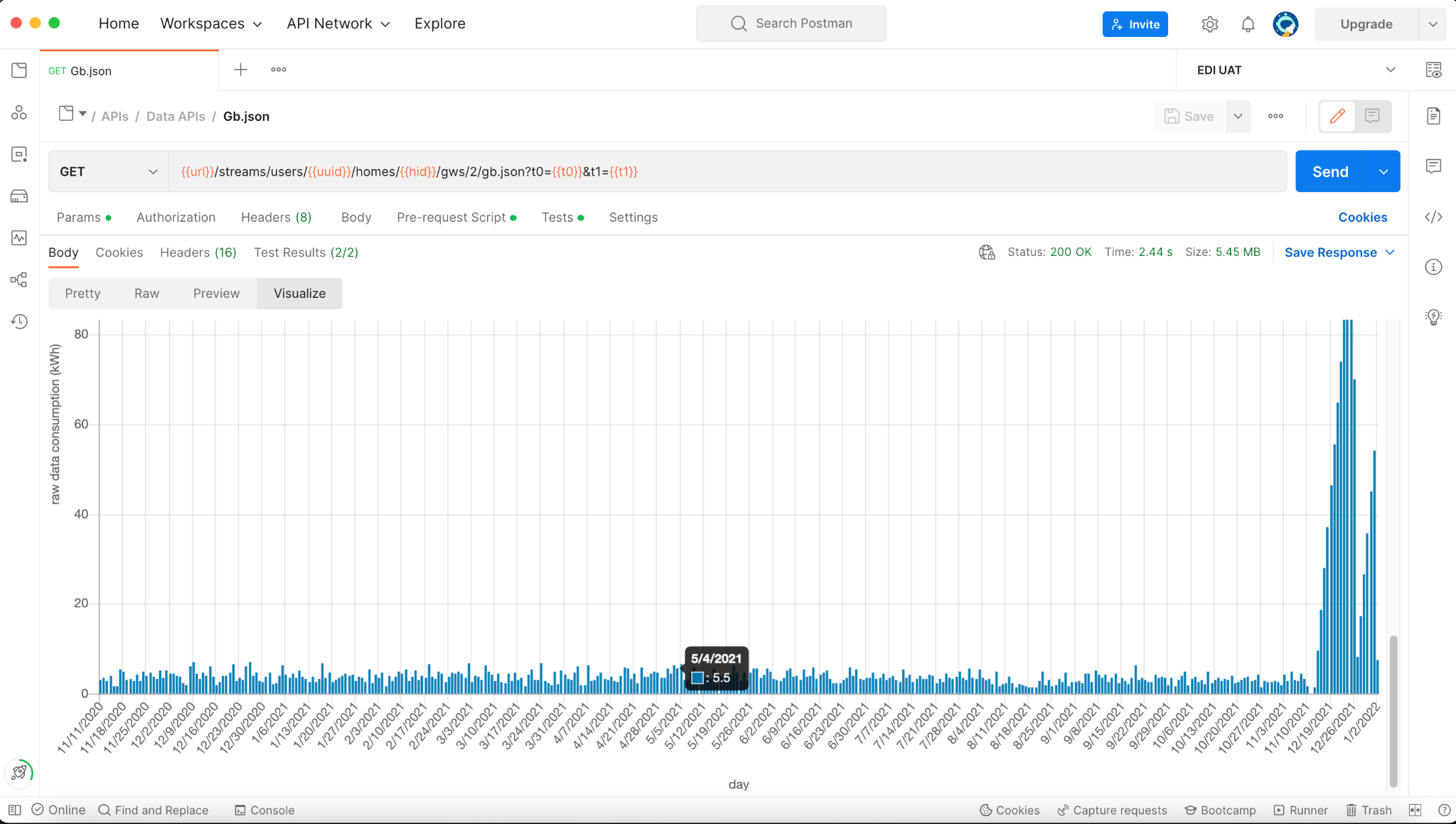
**CODE URL:**

<https://github.com/navneetnipu-bidgely/PROJECTS/tree/postman_data_visualizer/POSTMAN%20DATA%20VISUALIZER/raw%20data>

The above **pre-request script** code will fetch the billing cycles and store it as an environment variable.

Then the postman will s**end the gb.json API** and get the JSON response which will be processed by the **Tests script** post-API execution.

This will generate two plots highlighted below:

1. The total raw data for a billing cycle vs all available billing cycles:
   1. 
2. The second plot will be for raw data for each available day:
   1. 
3. These plots will help us to quickly check the raw data if any issue linked directly with the data.

# Disagg data visualization

**CODE URL:**

<https://github.com/navneetnipu-bidgely/PROJECTS/tree/postman_data_visualizer/POSTMAN%20DATA%20VISUALIZER/disagg>

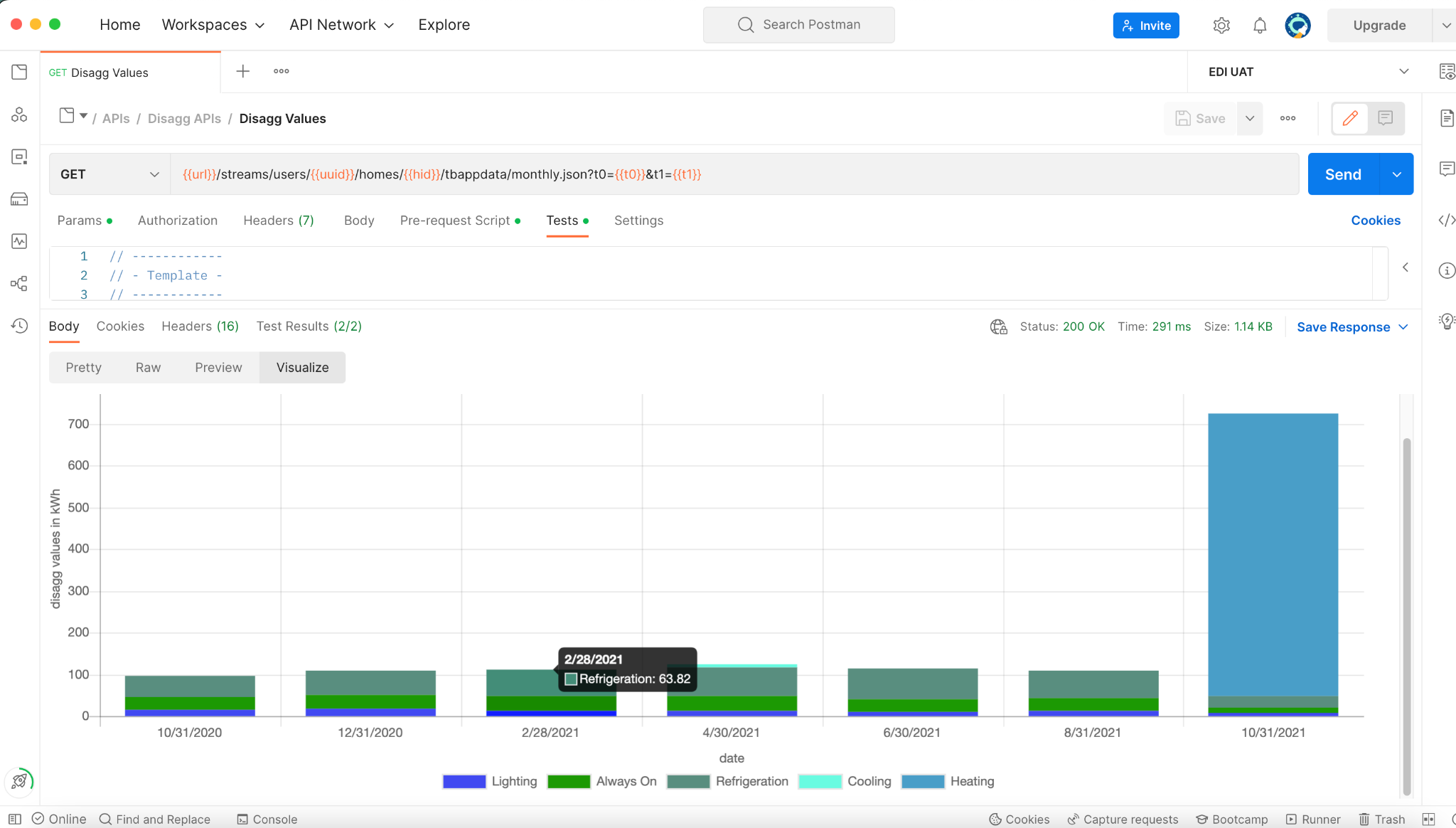
This code will be copied to respective pre-request script and Tests script for disagg values API mentioned below:

**{{url}}/streams/users/{{uuid}}/homes/{{hid}}/tbappdata/monthly.json?t0={{t0}}&t1={{t1}}**

The pre-request script will fetch the appliance mappings for the pilot and store it as environment variables.

Then the Tests script will generate the plot by processing the JSON data received from the disagg API.

This will generate the plot for consumption vs billing cycles as below:



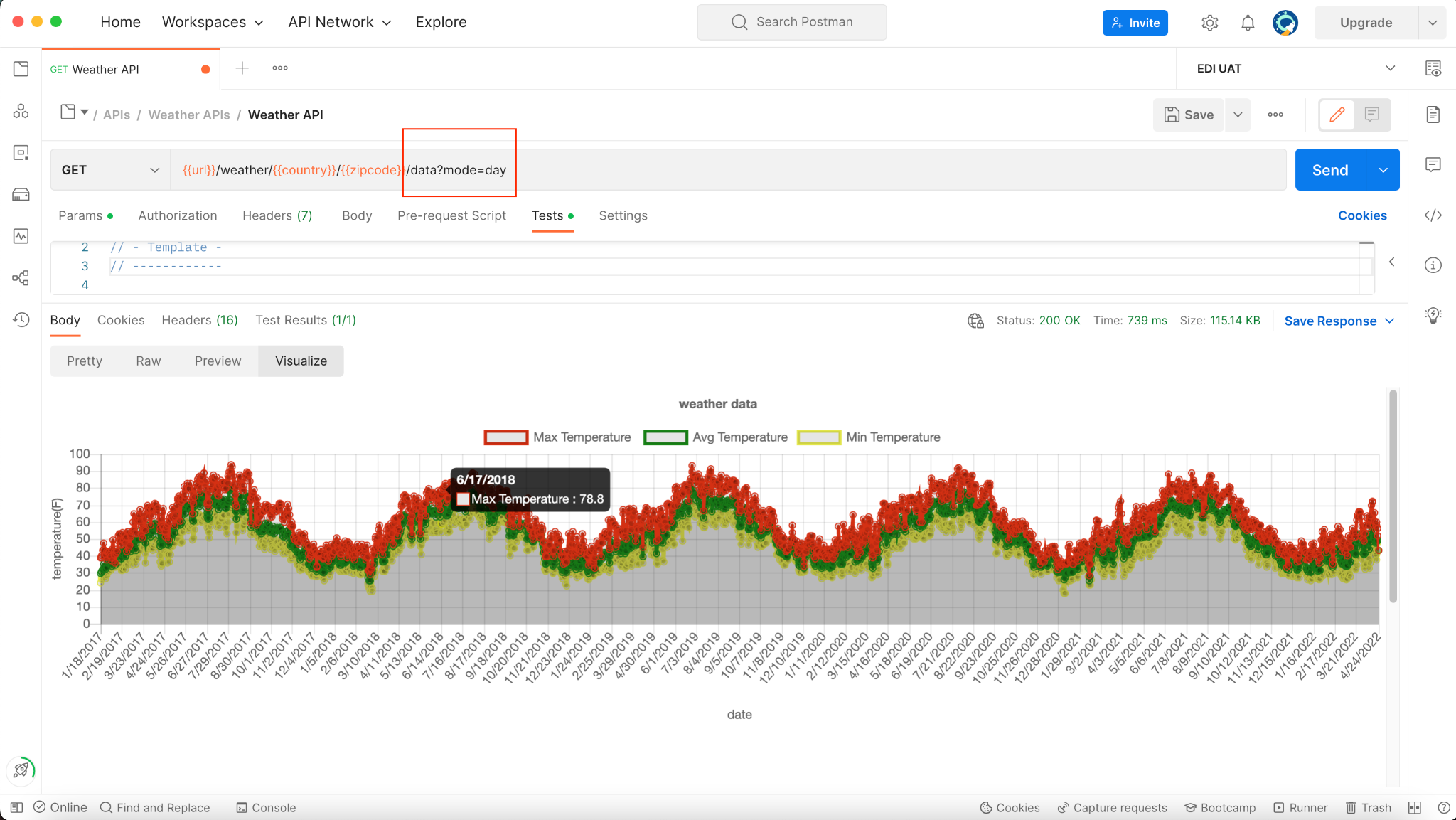
# Weather data visualization

**CODE URL:**

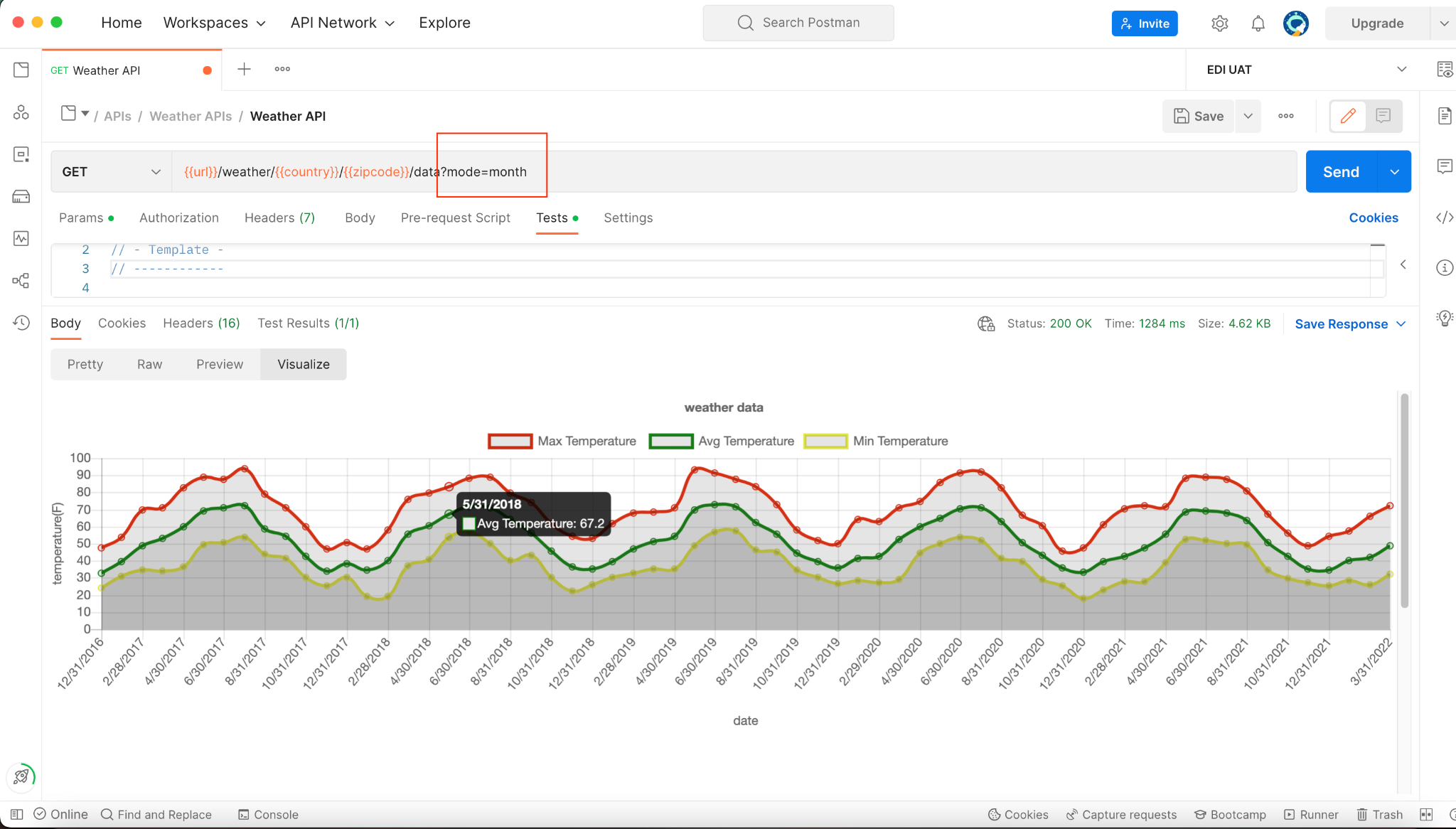
<https://github.com/navneetnipu-bidgely/PROJECTS/tree/postman_data_visualizer/POSTMAN%20DATA%20VISUALIZER/weather%20data>

This will generate the plot for **weather data (max, min, and avg temperature) vs day or month or year** depending upon the weather data API mentioned below:

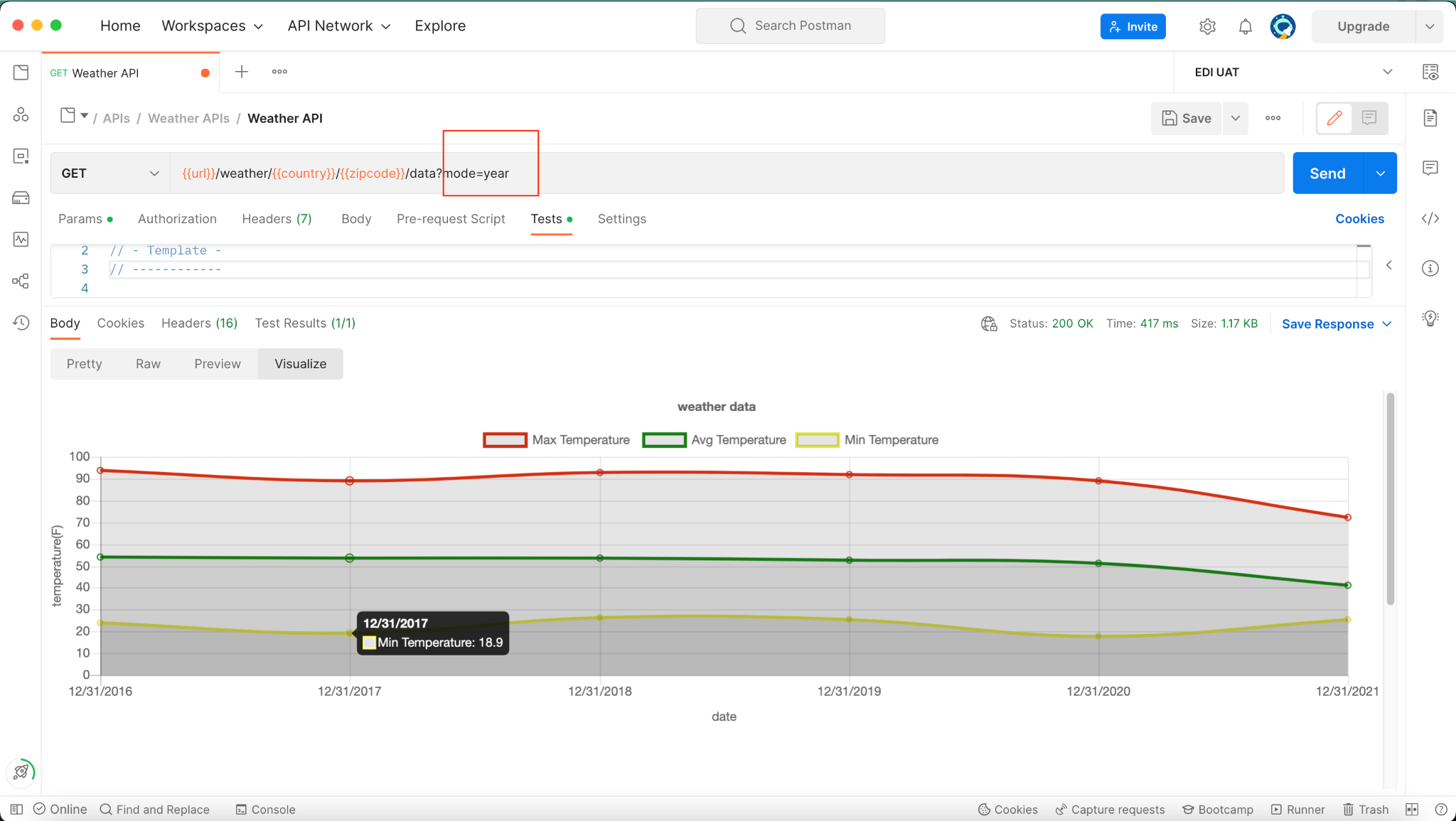
**DAY MODE:**



**MONTH MODE:**

****

**YEAR MODE:**

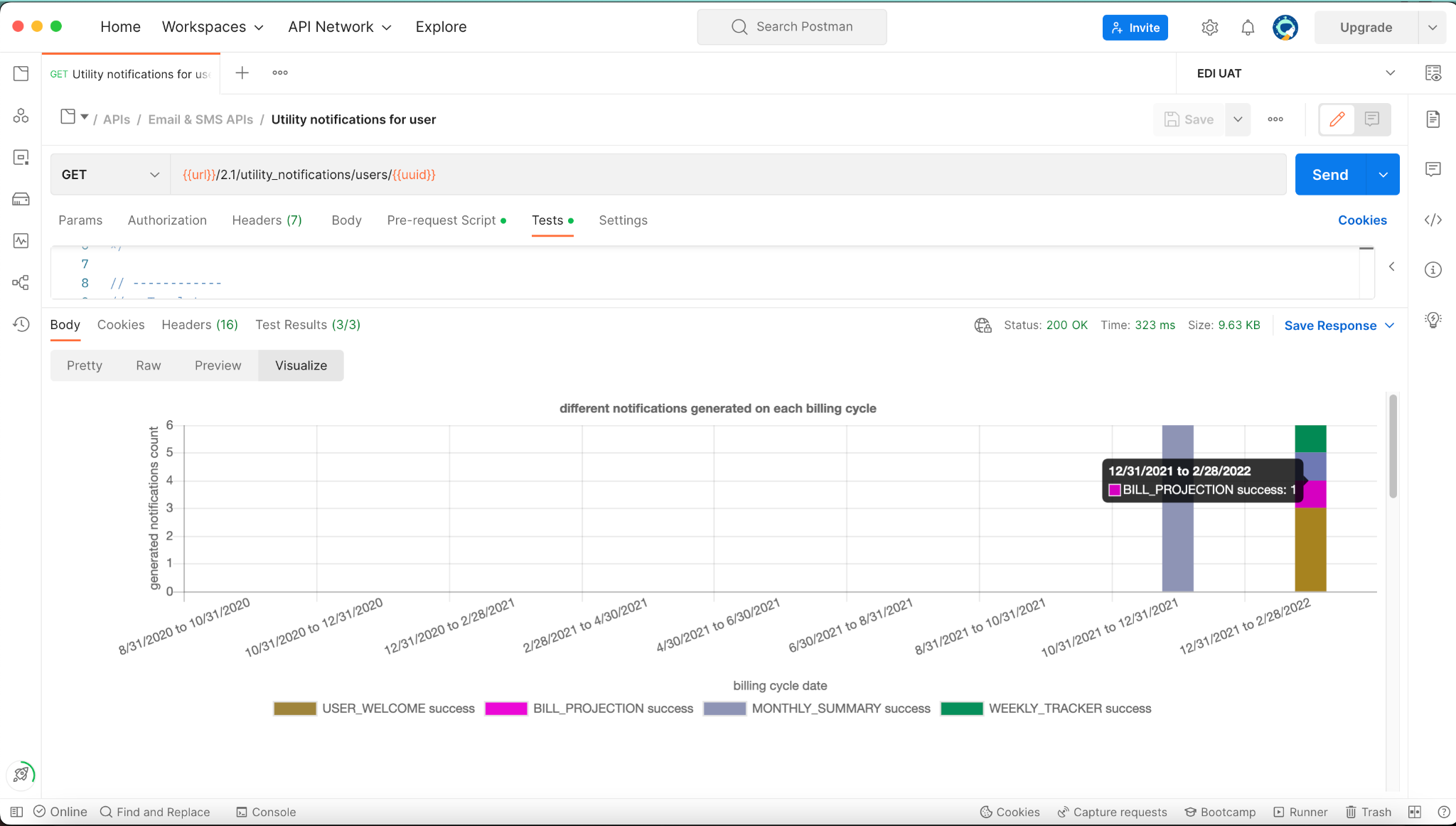
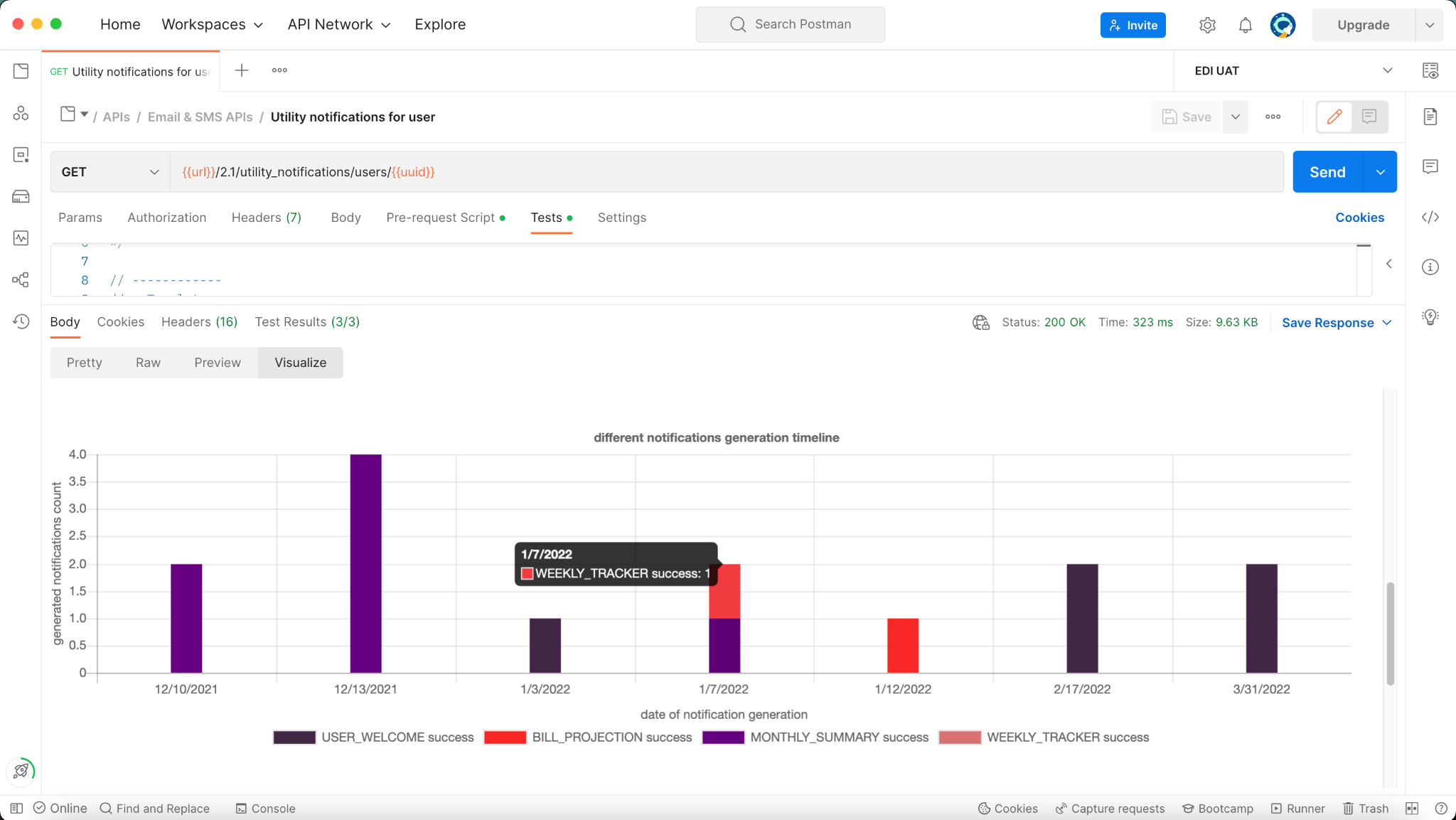
****

# Utility notifications generation timeline

**CODE URL:**

<https://github.com/navneetnipu-bidgely/PROJECTS/tree/postman_data_visualizer/POSTMAN%20DATA%20VISUALIZER/utility_notification_data>

The above code will generate two plots given below:

1. The utility notification for a billing cycle vs all available billing cycles**. This contains a stacked plot showing the number of each notification type generated for a user for a particular billing cycle and also highlighting the failure or success status for that** particular notification type:
   1. 
2. The utility notification generation timeline which shows which notification type was generated on which day:
   1. 
   2. This gives insights about what's happening with the notifications whether they are generated on time or not whether they are sent or get failed etc.