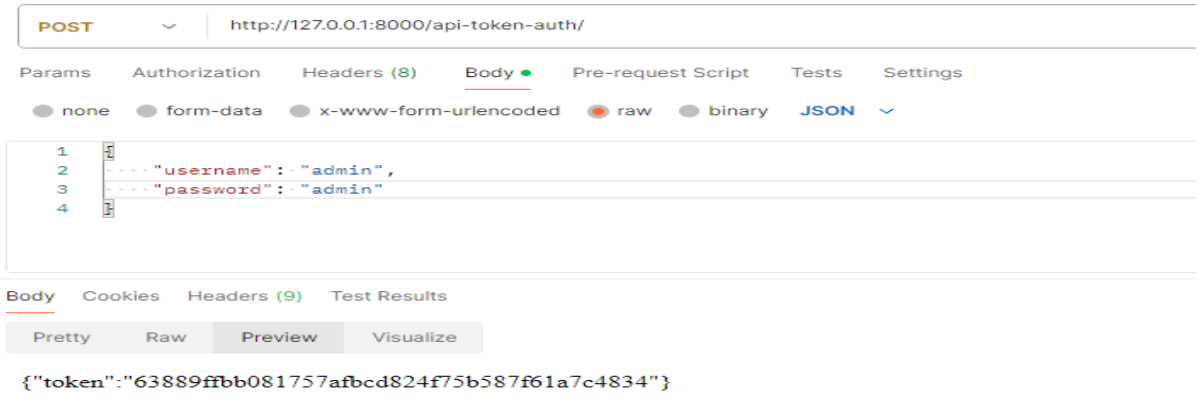
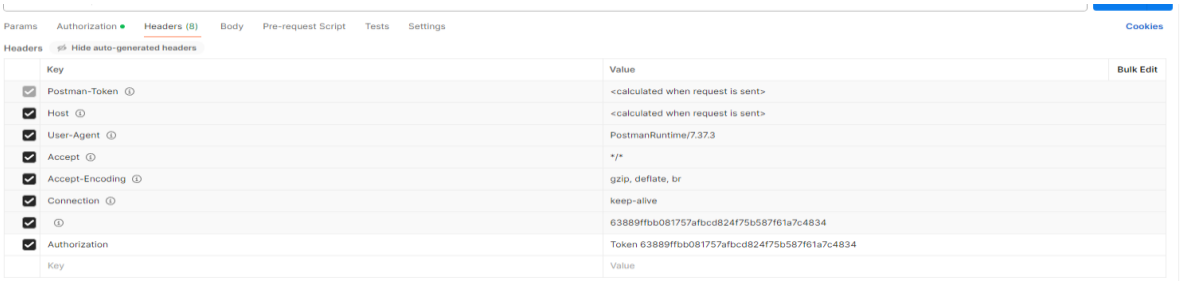
Testing Endpoints Step by Step

Authorization: We are using token based authentication. So first of all we have to get a token to make requests to all of our endpoints. We have to make sure our Django server is running. Then open postman.

We have to put a POST request at - http://127.0.0.1:8000/api-token-auth/ , in body we are sending username and password. In response it will give us back a token that we can use.

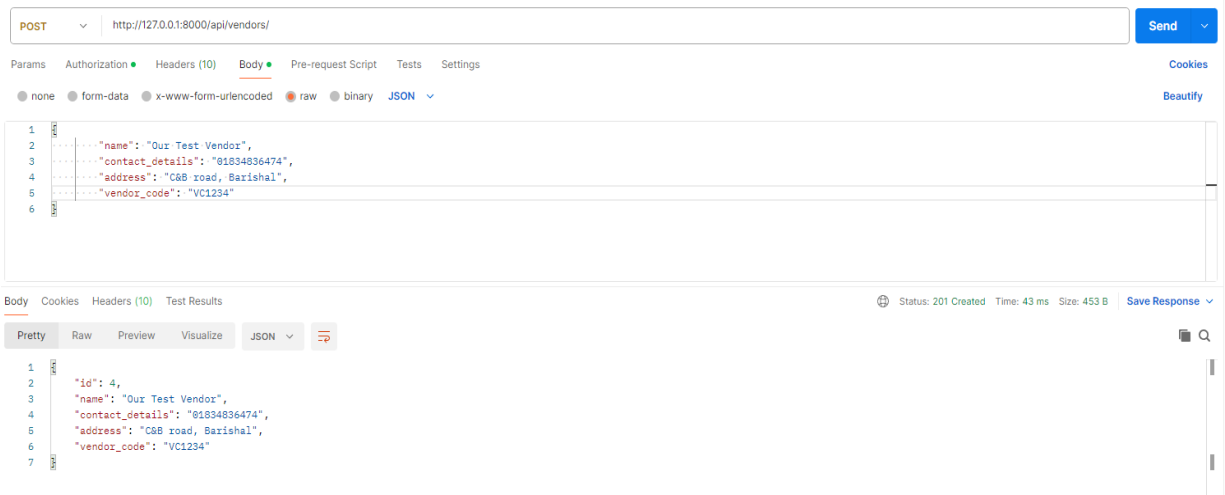


Now we have to go to header section on Postman, and put authorization and token like the below example. We are all set up to make requests to our endpoints now!



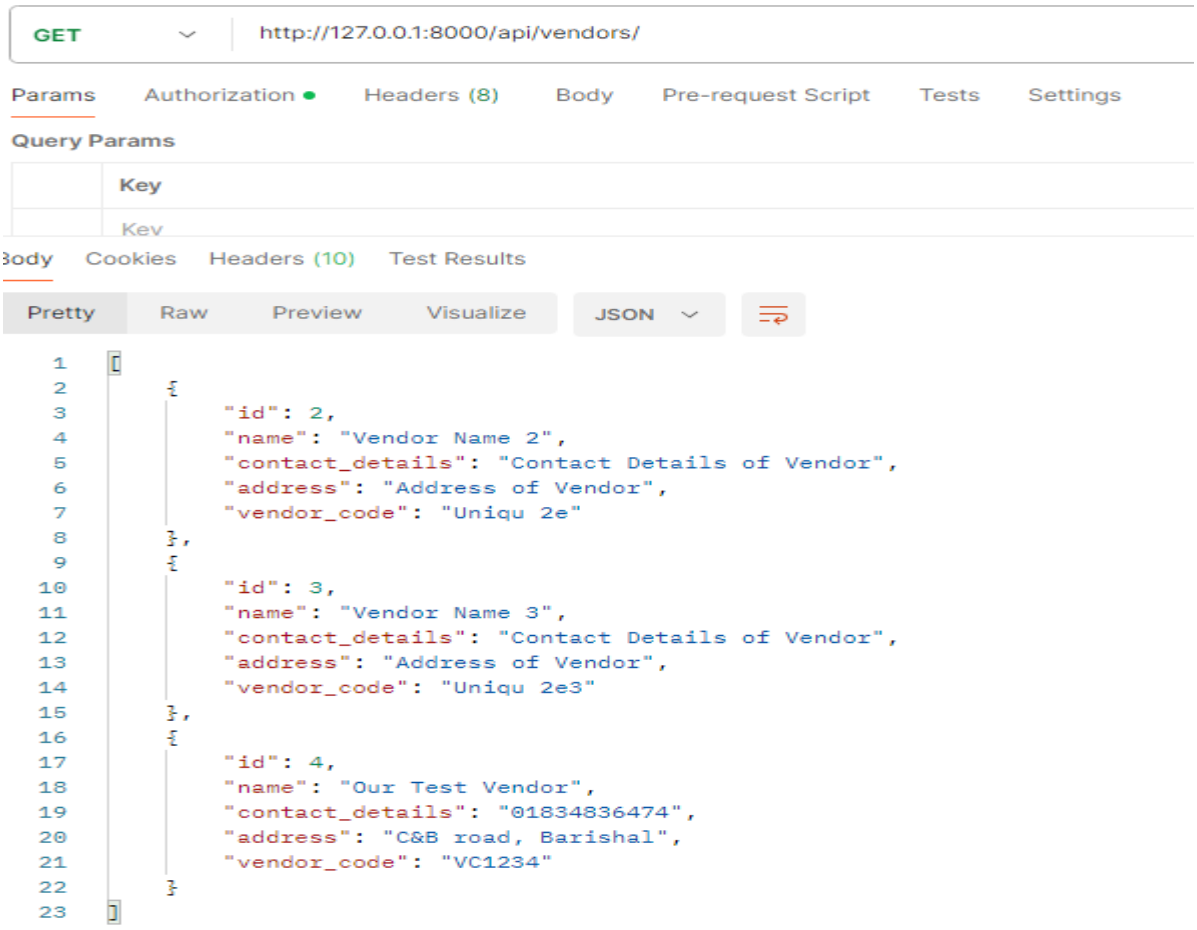
Endpoints

1. Create a new vendor



If we make a POST request at http://127.0.0.1:8000/api/vendors/ it with the data in body like above, it will create a vendor for us.

2. List all vendors:



We send a GET request at http://127.0.0.1:8000/api/vendors/ and it gives us all the vendors.

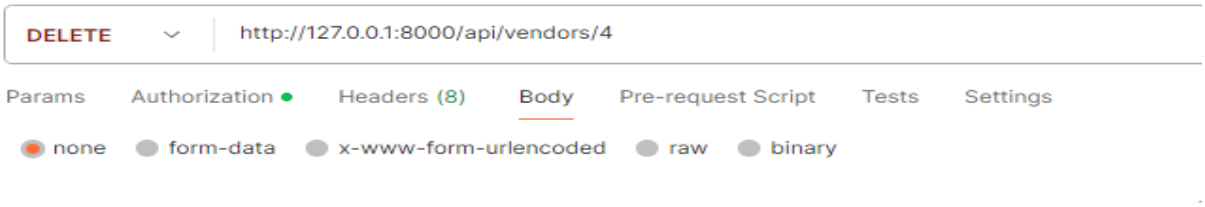
3. Retrieve a specific vendor's details:

We make a GET request at http://127.0.0.1:8000/api/vendors/4 and get details of the vendor with id 4.

4. Update a vendor's details:

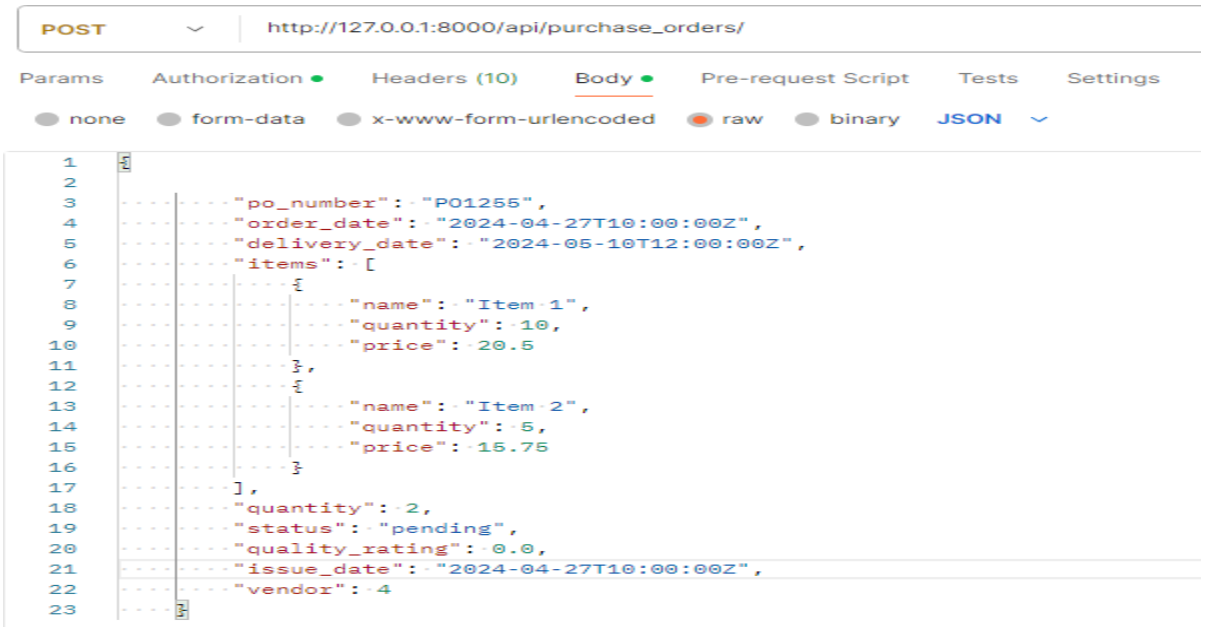
We made a PUT request at http://127.0.0.1:8000/api/vendors/4 and we are just updating the address of the vendor. It updates successfully.

5. Delete a vendor:

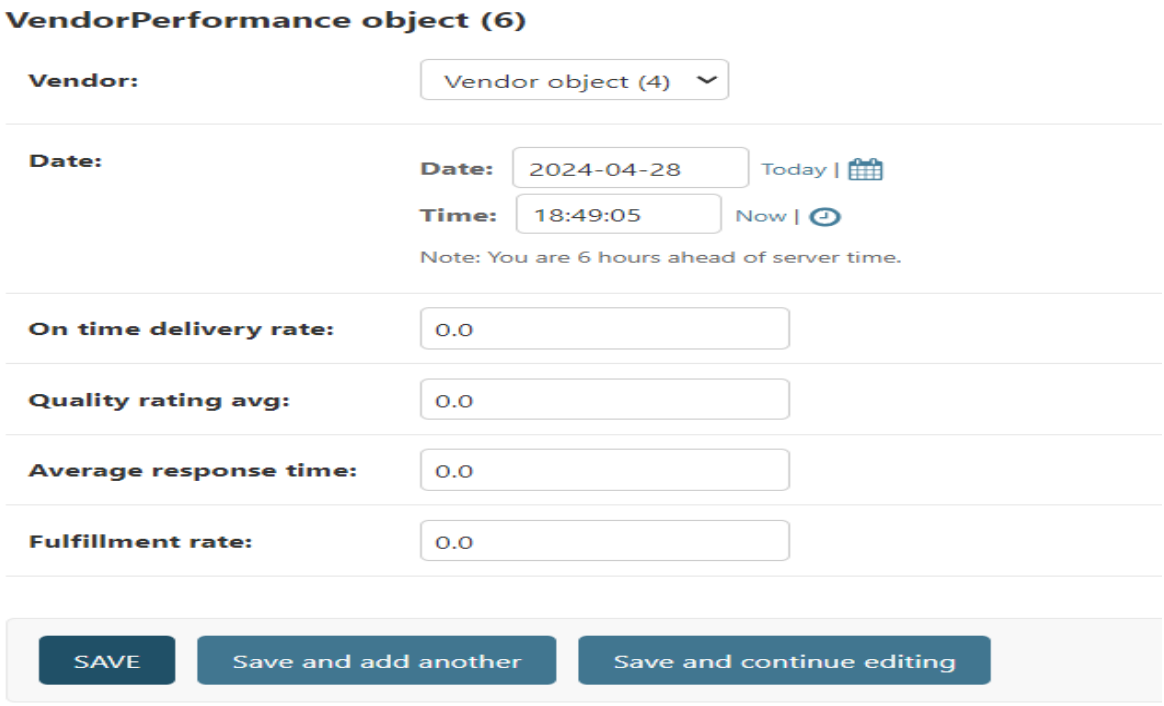


If we put a DELETE request, it will delete the vendor. For now, we are not deleting it for testing next endpoints.

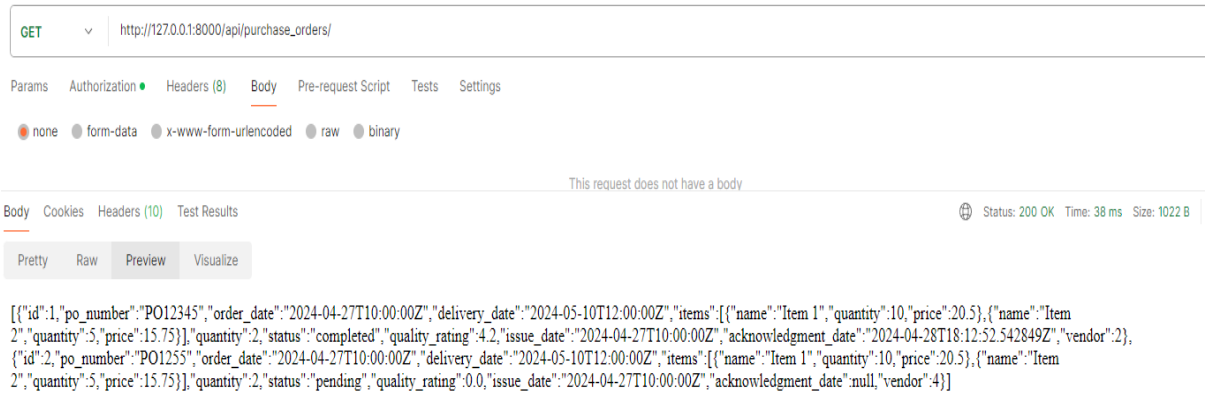
6. Create a purchase order:



We send a POST request at http://127.0.0.1:8000/api/purchase_orders/ and make a PO for vendor 4
After the PO object is created, Django signal creates an object of VendorPerfromance for vendor 4 and initializes it.(Prove from Django-admin)



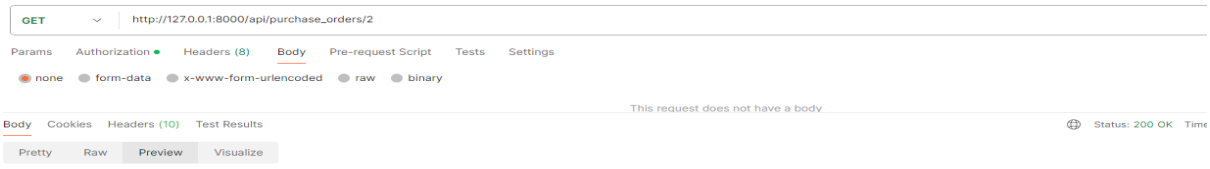
7. List all purchase orders with an option to filter by vendor



We send a GET request at http://127.0.0.1:8000/api/purchase_orders/ and it returned all the POs

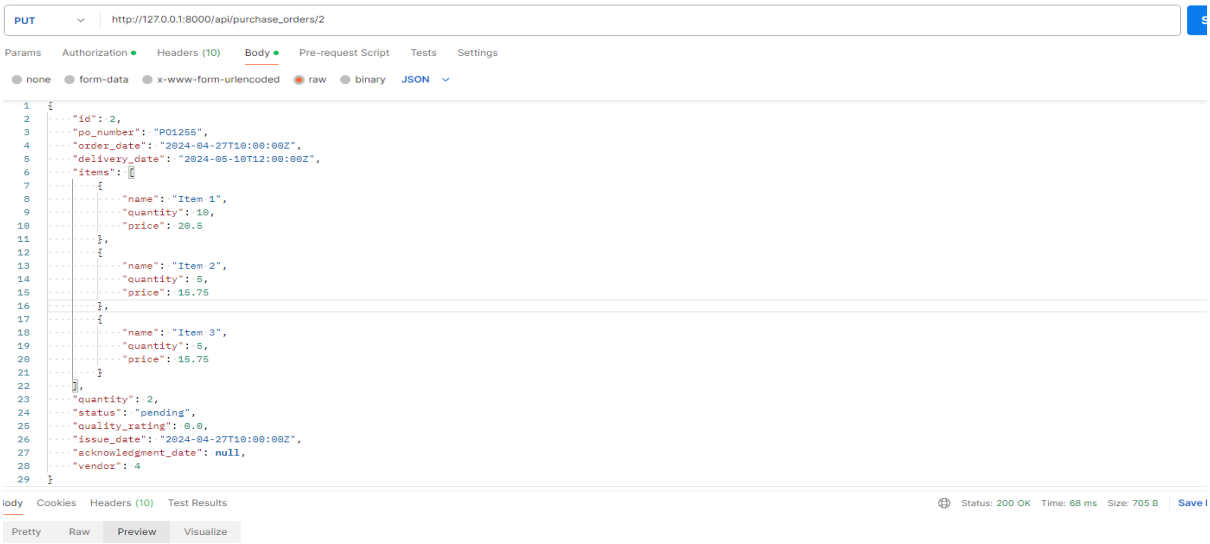
We send GET request at http://127.0.0.1:8000/api/purchase_orders?vendor=4 and it returned all the POs for the vendor 4. So the option to filter by vendor also works.

8. Retrieve details of a specific purchase order:



We send a GET request at http://127.0.0.1:8000/api/purchase_orders/2 to get the PO with the id 2.

9. Update a purchase order:



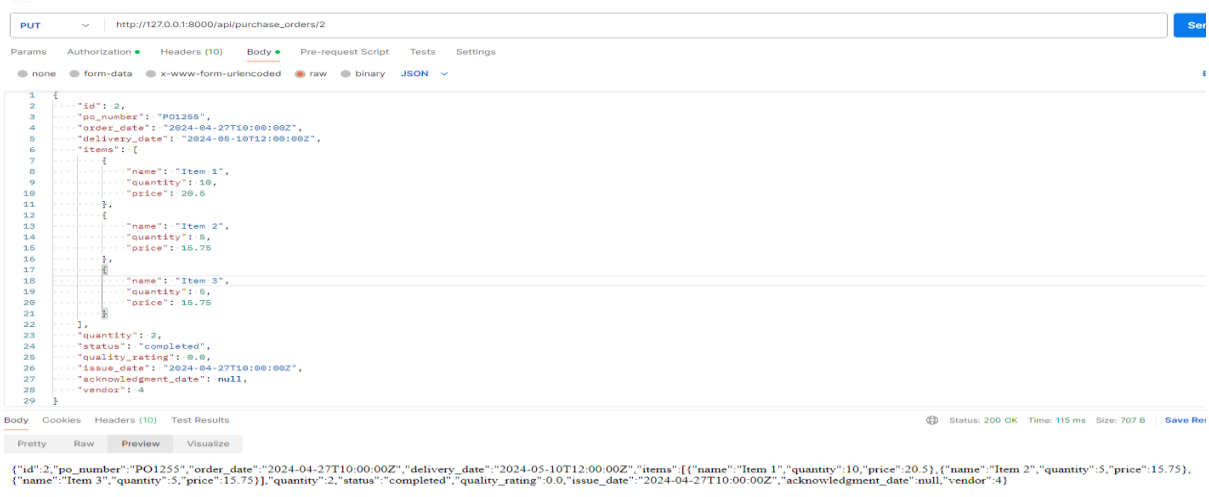
We send a PUT request at http://127.0.0.1:8000/api/purchase_orders/2 to add a new item to the PO with id 2. It is successfully updated.

10. Delete a purchase order:

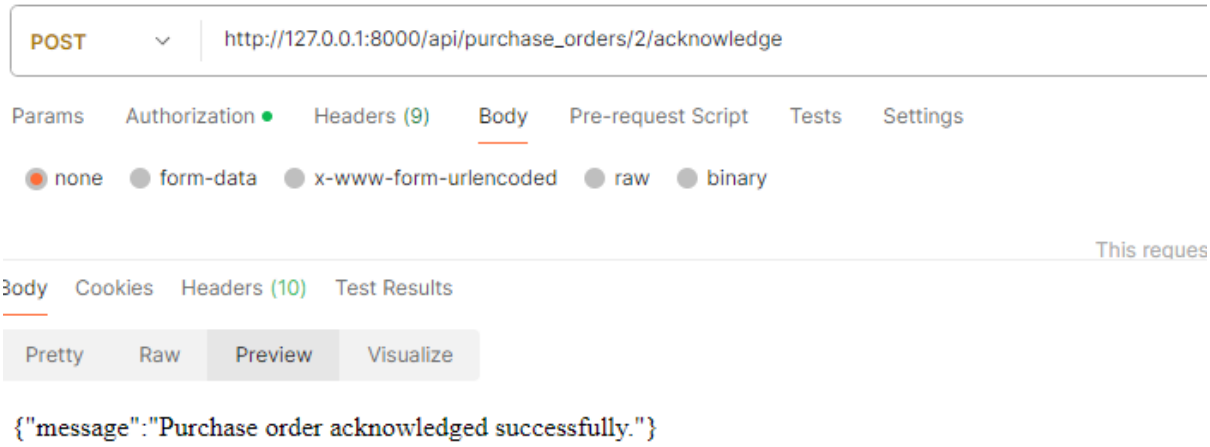
For the sake of next endpoints testing we are not deleting.

Performance

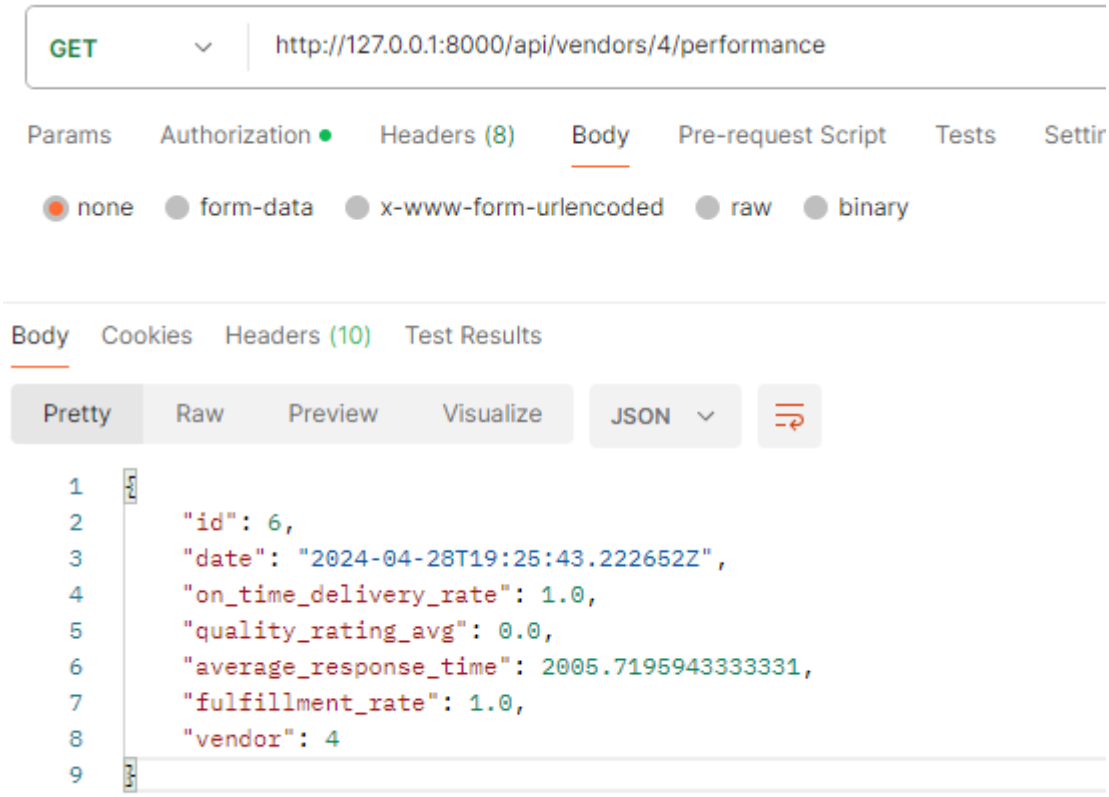
First let us set the status of our recently created PO with id 2 and make it completed. It has some consequences. Any change to this PO will trigger Django signals and it will calculate the metrics.



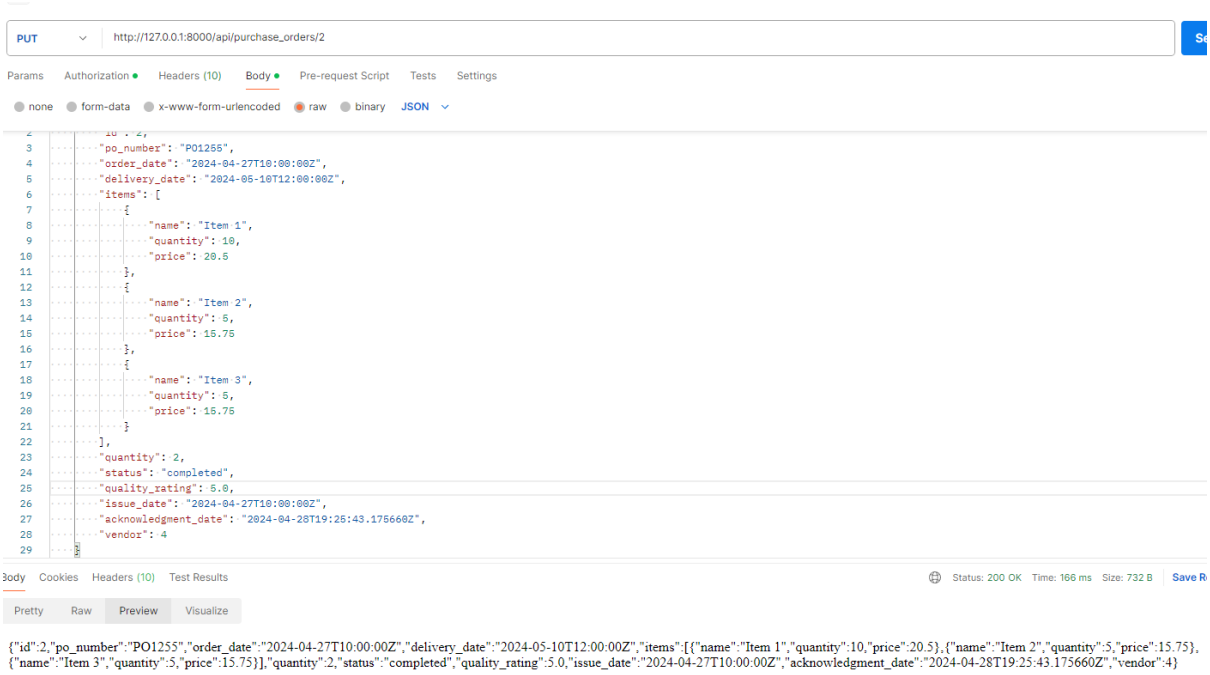
Now let us put a POST request for acknowledgement.



Now let's check the performance of vendor 4. And his recently created PO id is 2.

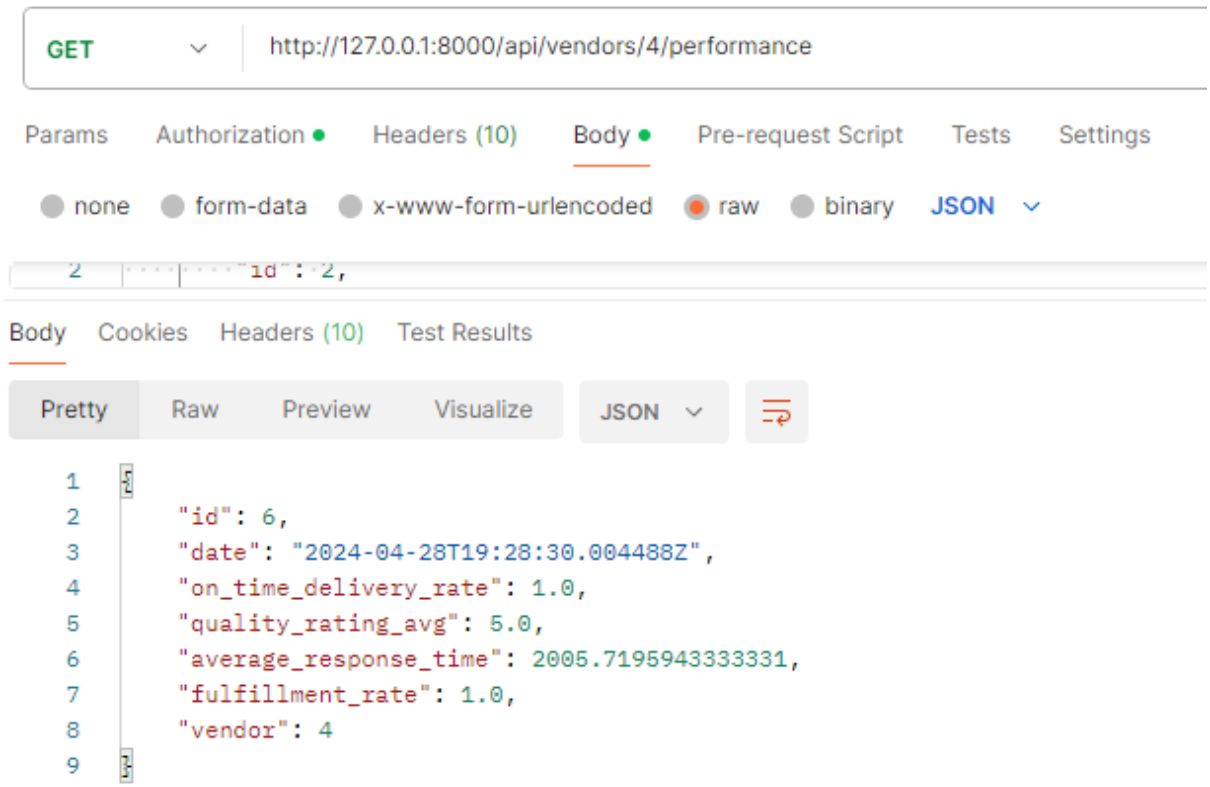


So we can see the performance of the vendor 4. But his average rating is 0 because he has not received any rating yet. So let us give him a rating.



{"id":2,"po_number":"PO1255","order_date":"2024-04-27T10:00:00Z","delivery_date":"2024-05-10T12:00:00Z","items":[{"name":"Item 1","quantity":10,"price":20.5},{"name":"Item 2","quantity":5,"price":15.75},{"name":"Item 3","quantity":5,"price":15.75}],"quantity":2,"status":"completed","quality_rating":5.0,"issue_date":"2024-04-27T10:00:00Z","acknowledgment_date":"2024-04-28T19:25:43.175660Z","vendor":4}

Let us check his performance again:



So now we see that the quality rating is updated.
Note: On_time_delivery_rate and fulfilment_rate is 1.0 because we only have one PO for vendor 4. We are calculating average_response_time in seconds.

From now on this performance calculation will be updated every time a new PO is created or updated for vendor 4.