Name: Willy Seah Wee Hung

Title: Sellgo Backend Interview Test Report

Link to Github: <https://github.com/WillySeahh/sellgobackend>

Instructions to set up:

1. Pull my repository from github at the link provided above.

2. Ensure that postgres is on your computer and create a database called ‘dbforsellgo’

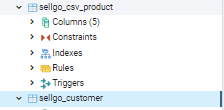
3.Type python manage.py makemigrations sellgo into Terminal and click Enter

4. Type python manage.py migrate sellgo into Terminal and click Enter.

5. Type python manage.py runserver 8080into Terminal and click Enter.

DB Tables

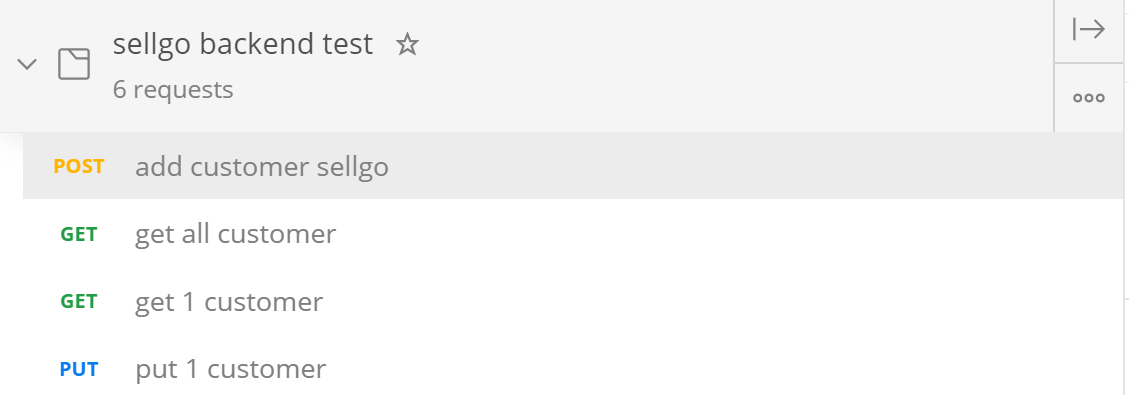
2 tables are created as seen below:



Endpoints

Customer

The Customer table supports the common CRUD operations such as POST,PUT,DELETE,GET.



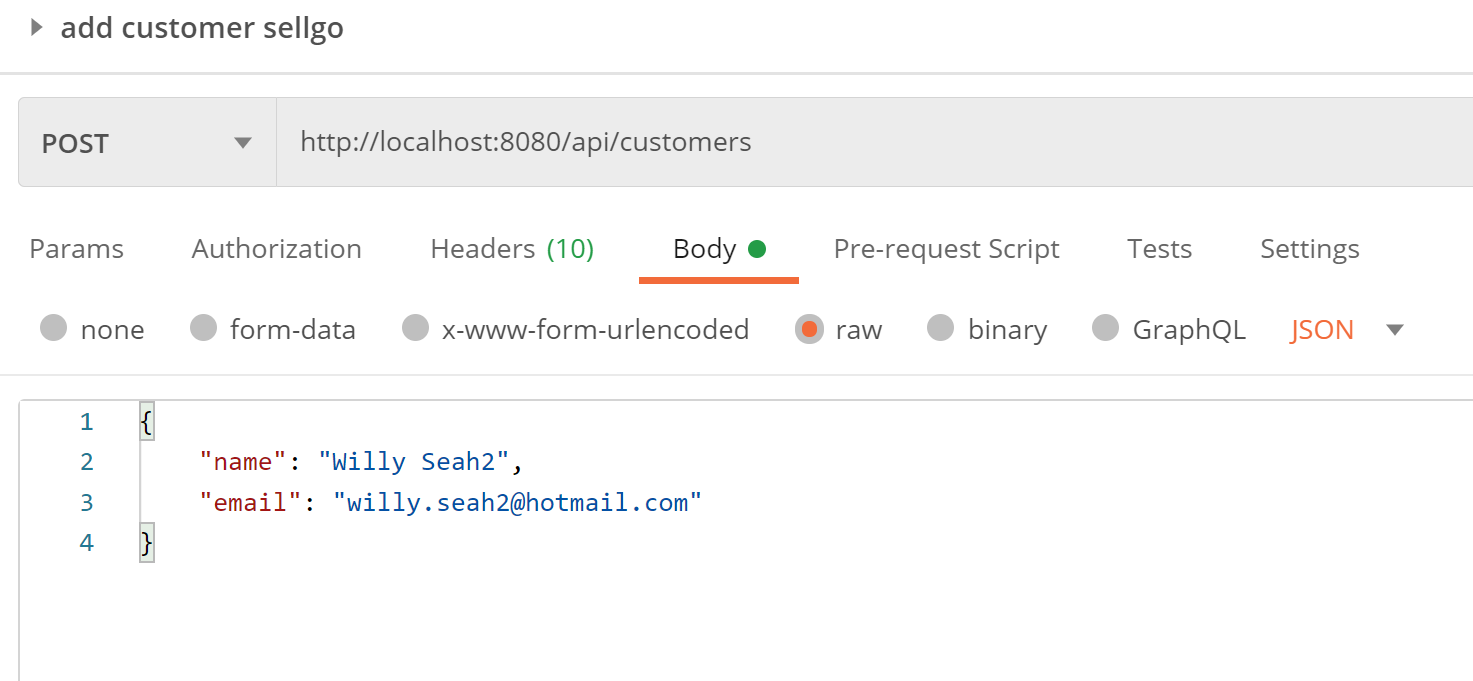
For Post, GET all and Delete all the url is as follows:

<http://localhost:8080/api/customers>

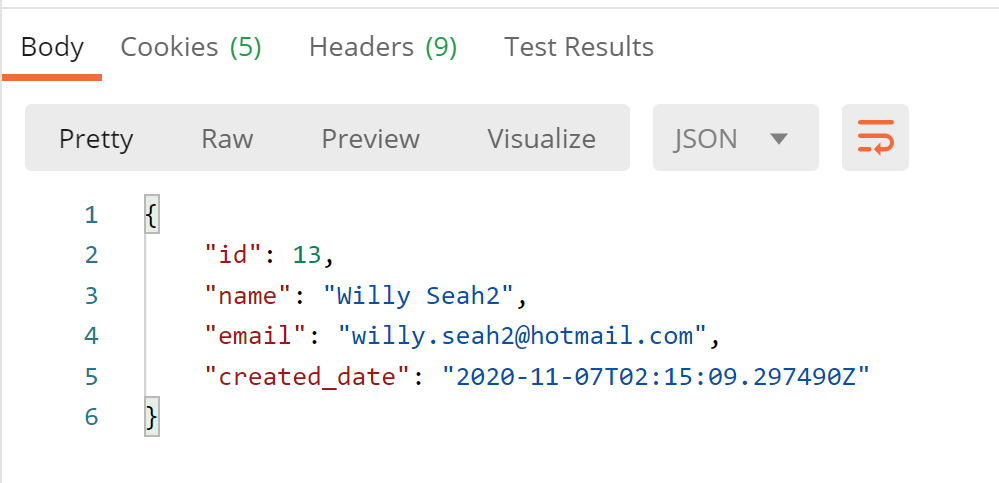
For Get one, Put, Delete one the url is as follows:  
[http://localhost:8080/api/customers/{id}](http://localhost:8080/api/customers/%7bid%7d)

Here are some examples of testing the Customer endpoints using Postman.

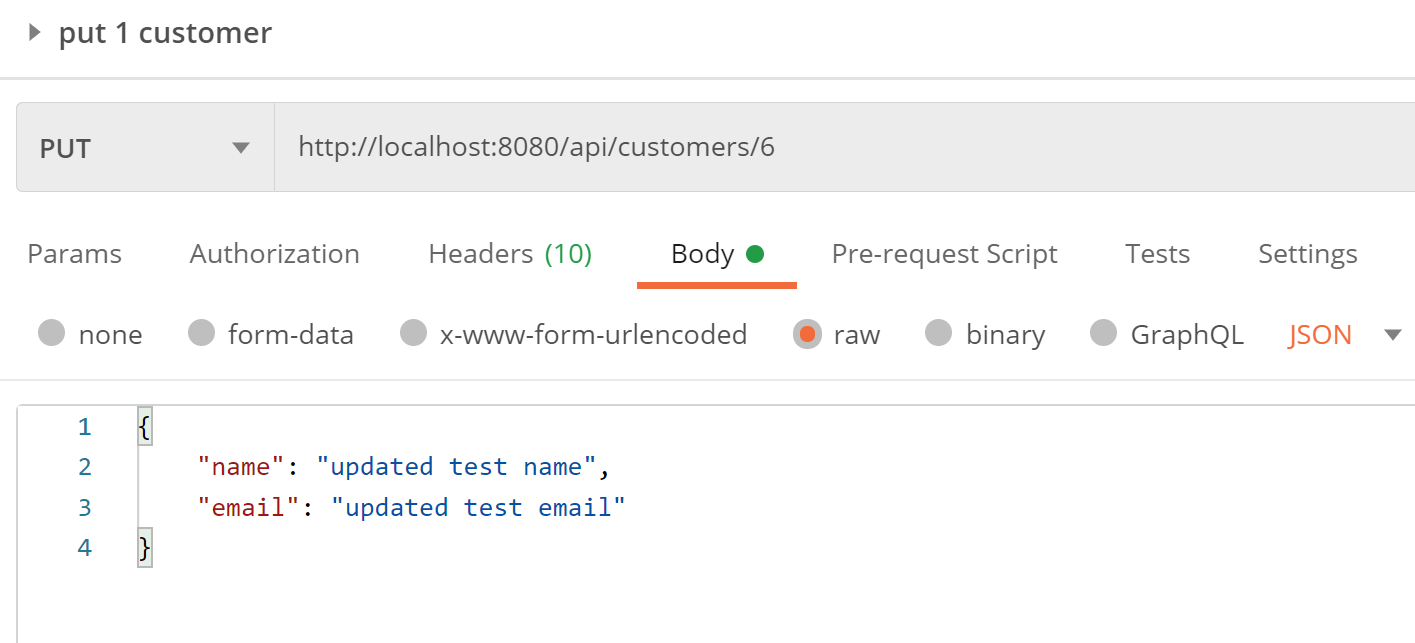
Creating 1 customer:



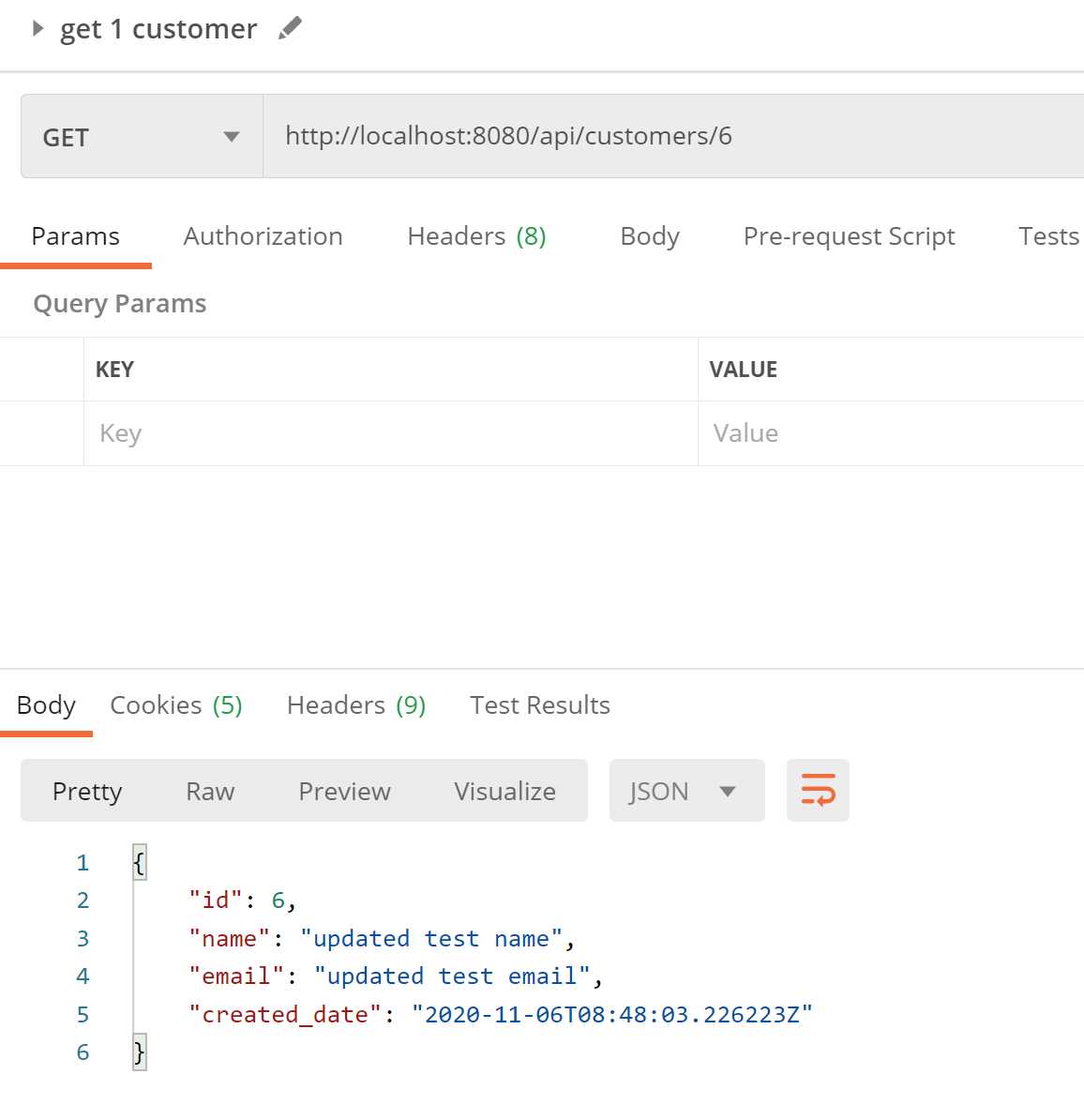
Success message of creating 1 customer:



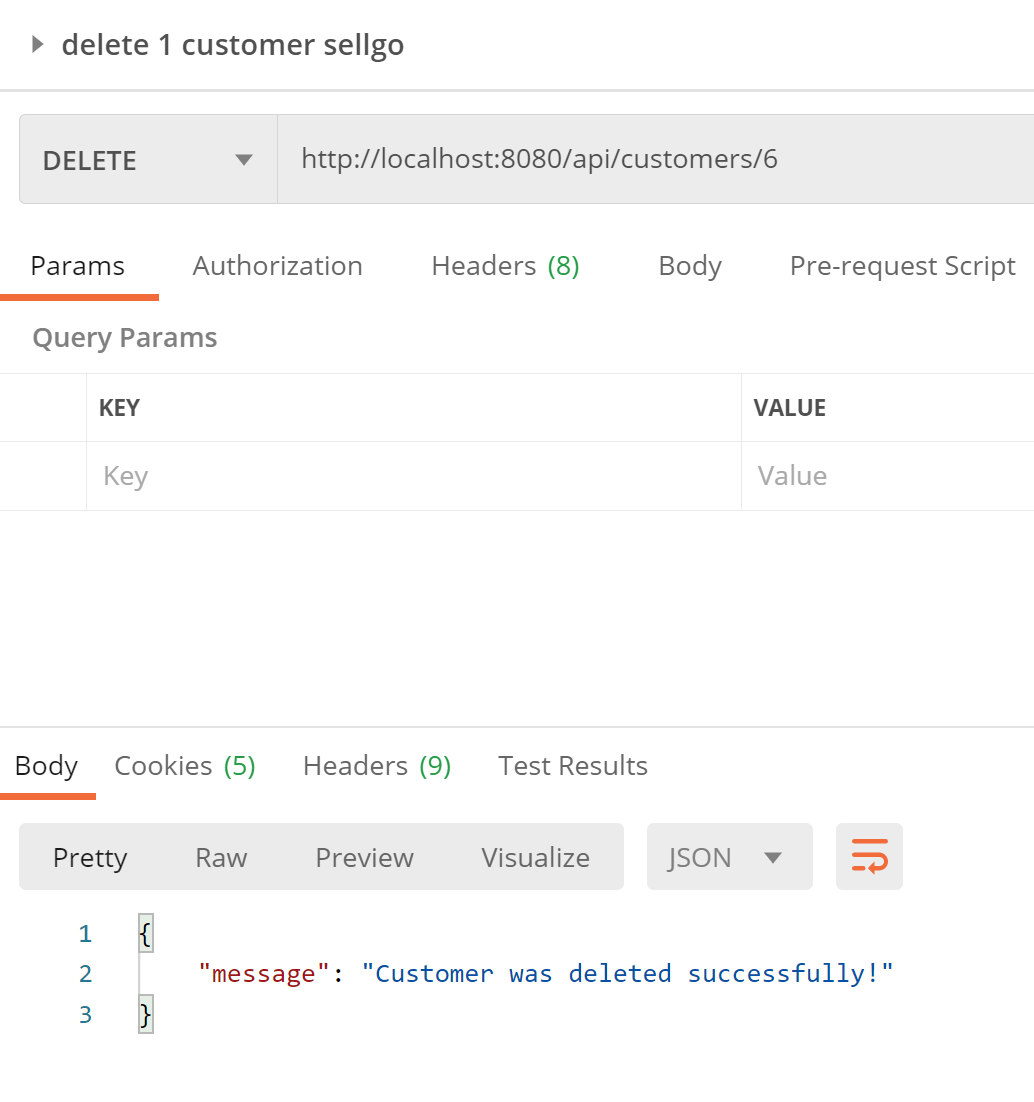
Updating the details of 1 customer:



Getting 1 customer and its success message:



Delete 1 customer and its response message:



Csv\_Product

When taking the content of the .csv file with the customer id which will be passed in the URL.

POST URL: http://localhost:8080/api/csv/{customer\_id}

It will create 3 rows in the csv\_product table every time the POST command is ran.

It adds all the items in the CSV, parses them accordingly and before adding each item into the database.

The table will store all uploaded data historically in the sense where previous data is persistent and will not be deleted.

If there are more than 3 items in the csv, all will be added into the table as well.

Snippet of the csv\_product table after I called POST on the above URL:



Combination of both of the above queries

I interpreted the question as given 1 customer id return all the latest product associated with the customer. For this I had to query 2 different tables, manipulate their output and convert them into 1 single JSON that is returned.

Since the only input is customerid, calling it once will return all 3 product with their latest title, price and uploaded date.

GET URL: [http://localhost:8080/api/csv/{customer\_id}](http://localhost:8080/api/csv/%7bcustomer_id%7d)

Sample of JSON returned:

