

Distributed os

hw2



November 20, 2020

**Done by : Aida Bustami**

**supervised to : Dr. Samer Arandi**

**Summary:**

**I have implemented the front end layer in the same computer (192.168.1.50) in port 8000 and in secure https or Http .**

**I ran the catalog server in VirtualBox inside a virtual machine instance (ubuntu) running on static ip address which is 192.168.1.215 and I run the order server in VirtualBox inside a virtual machine instance (ubuntu) running on static ip address which is 192.168.1.251 .**

**I implemented each microservice using Restful flask – python and I used the sqlite database .**

**I made two instances of the database one for catalog server and one for order server.**

**named as database : for catalog server and database1 : for order server.**

**I also installed GitHub client for desktop use in my host machine and the other two virtual machines to be able to reach same remote repository from different machines.**

**\*I implemented the internal communication between virtual machines and my host by setting the network setting on both of the virtual machines to be bridged Network settings .**

**And I also used postman as the client app that will send the requests to the front end server which will then redirects the https(front\_end\_https) or http(front\_end\_http) request that has been send to it to http request related to a specific end point whether in order or catalog server.**

**I implemented 5 operations that can be accessed from the user point of view (I mean request goes to front end) but I also implemented internal operation that normal users can not reach from front end layer . examples of these functionalities are : update information or query things in both orders server or catalog server like :**

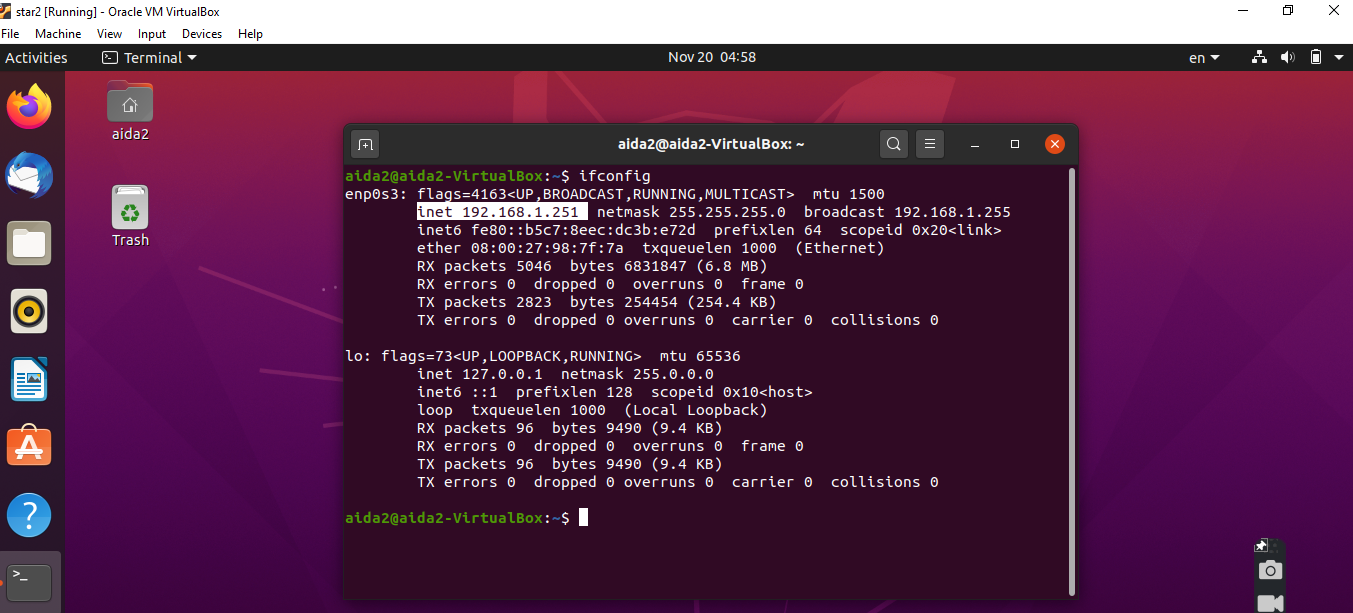
**getting all orders ,edit order information ,printing all books in book store, create new book ,edit book information ,delete a book ,search books by different parameters ,search order table by different parameters ,increasing number in stock for a book , decreasing number in stock for a book …**

**I implemented HTTPs only on the front end microservice using self-assigned certification and I edit postman settings to accept it.to run it you need to do the self assigned certification . I followed the steps in this article :** [**https://blog.miguelgrinberg.com/post/running-your-flask-application-over-https**](https://blog.miguelgrinberg.com/post/running-your-flask-application-over-https) **but if you prefer I will include two versions of front end one that works with https(front\_end\_https) and need self assigned certificates to work and other that works with http(front\_end\_http) and runs in a normal way.**

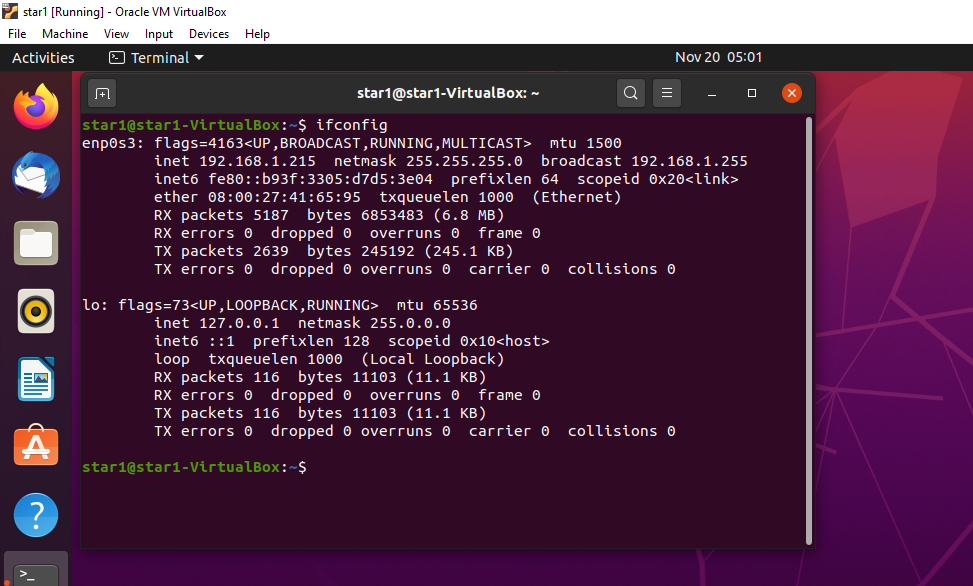
**You can run my code by running front end in host , catalog server in vm1 , order server on vm2 and send requests to front end using postman .**

**Some Results:**

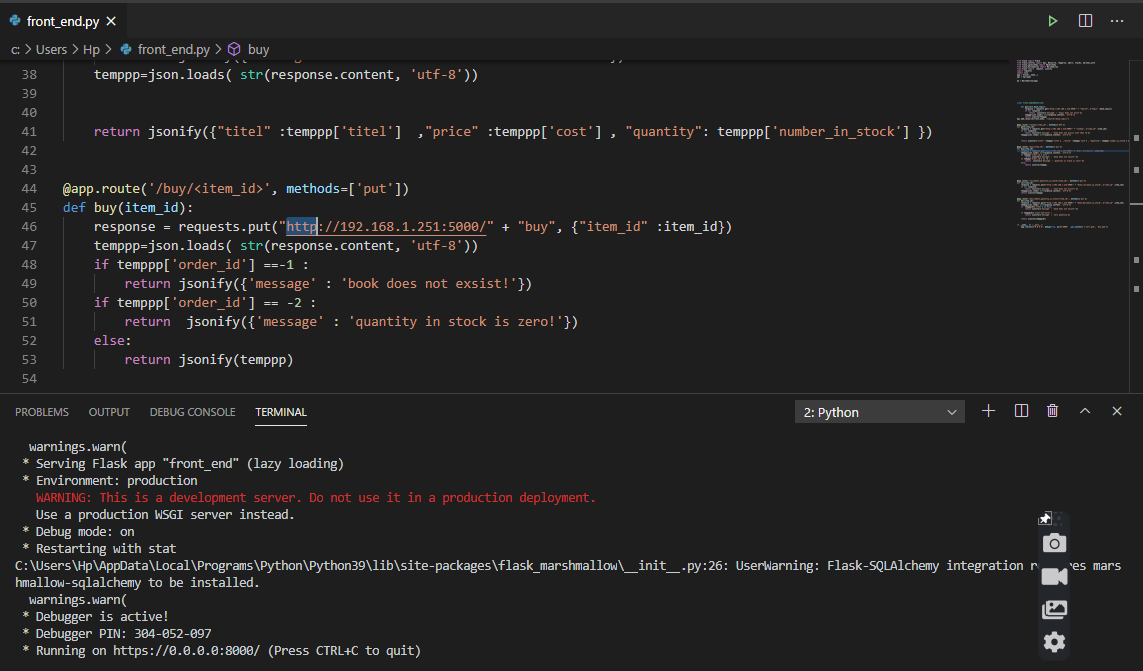
**\*running vm2 on ip : 192.168.1.251.**



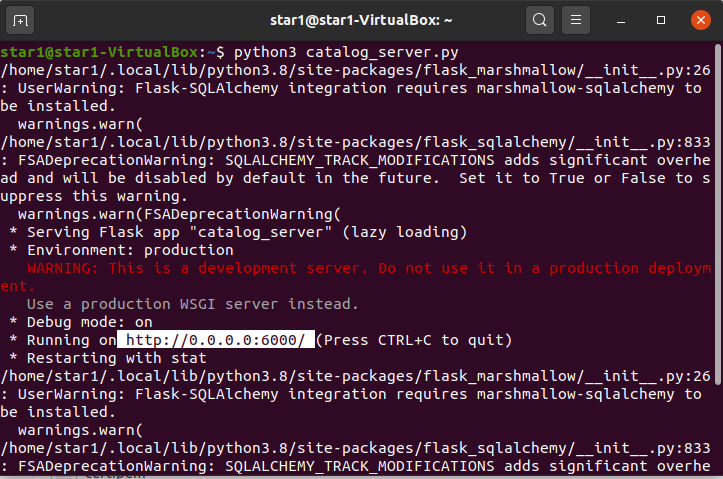
**running vm1 on ip : 192.168.1.215.**



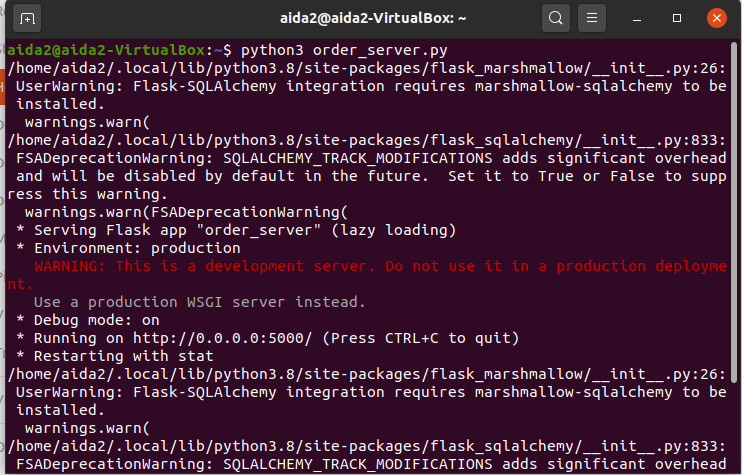
running front end server on ip 192.168.50.1 : 8000



**\*running catalog server on ip : 192.168.1.215:6000**

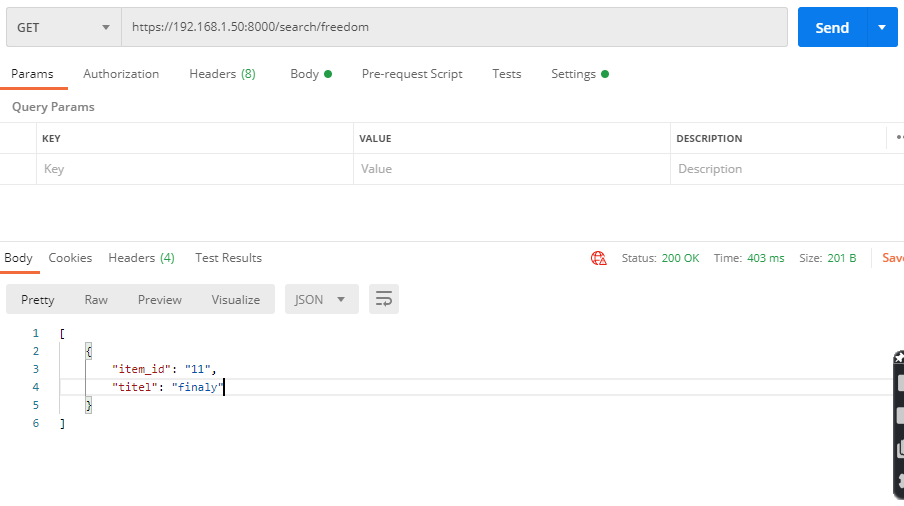


**running order server on ip 192.168.1.251 : 5000**

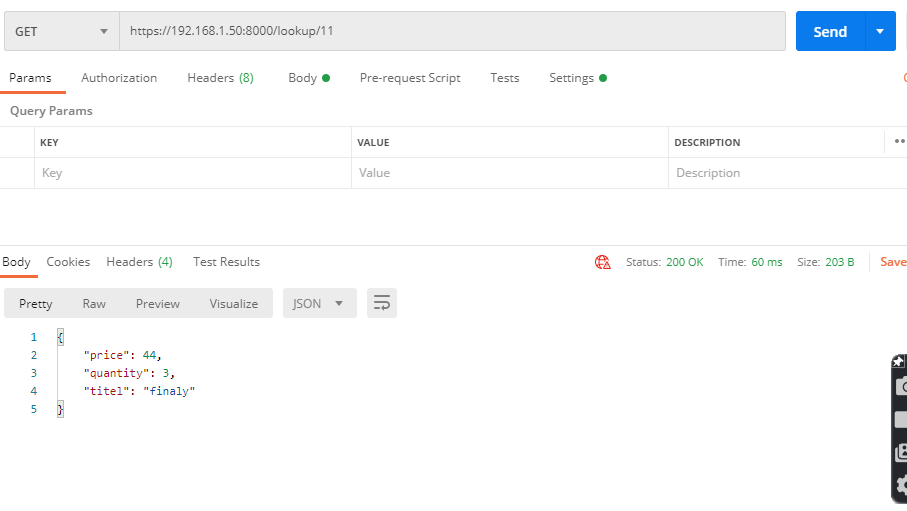


**Here is some of the result from executing the operations that the front end server can handle .**

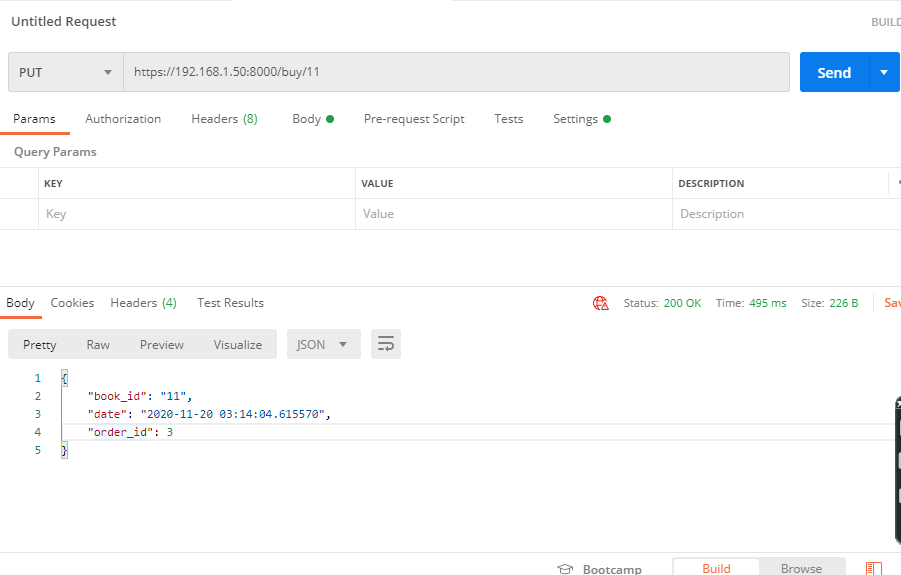
1-search



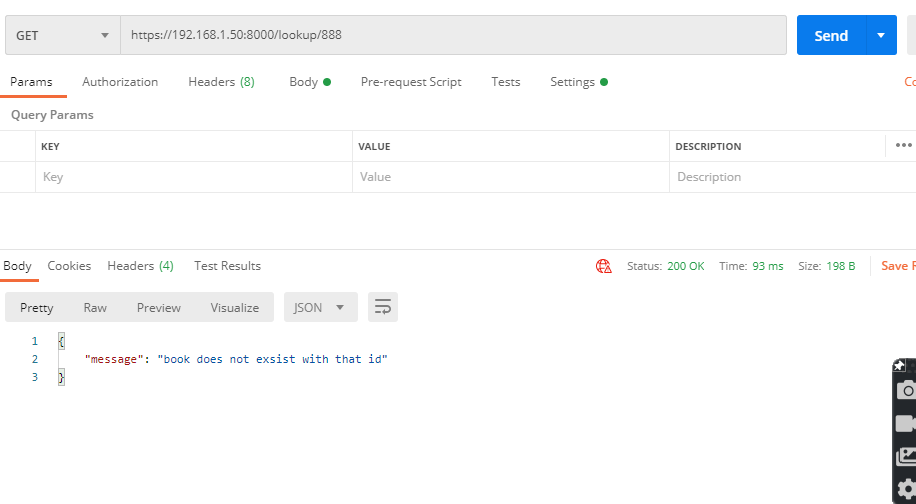
2- lookup



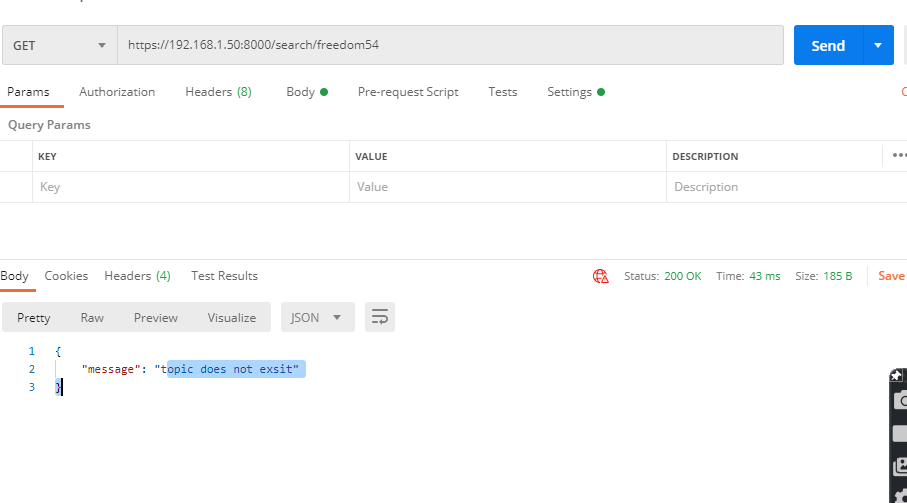
3- buy



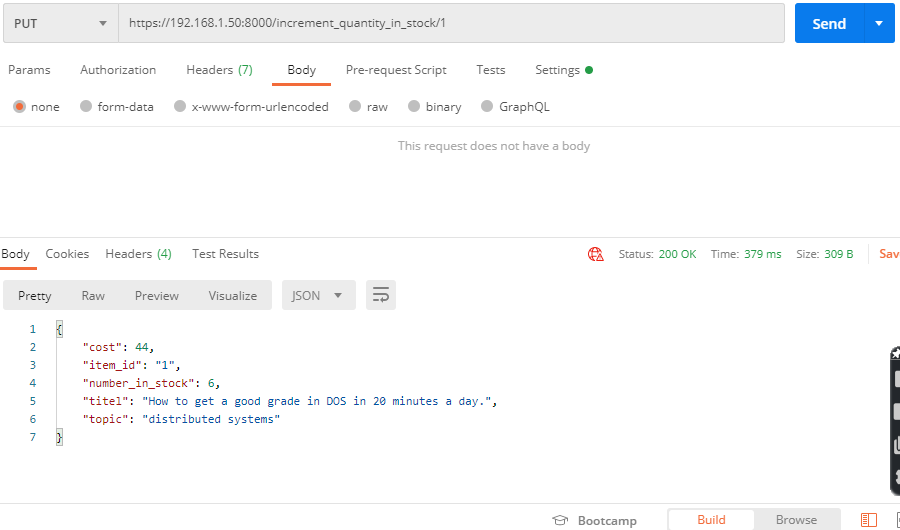
4- lookup with invalid id .



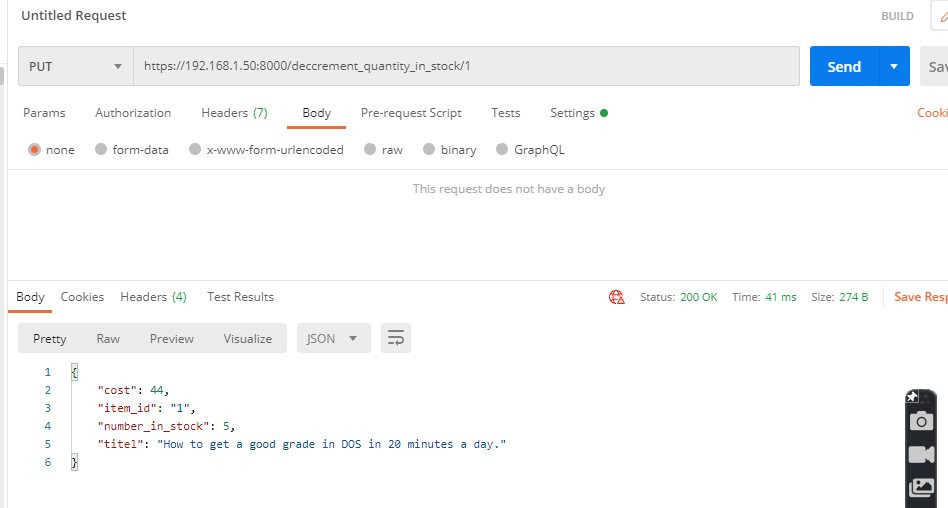
5- search with invalid topic



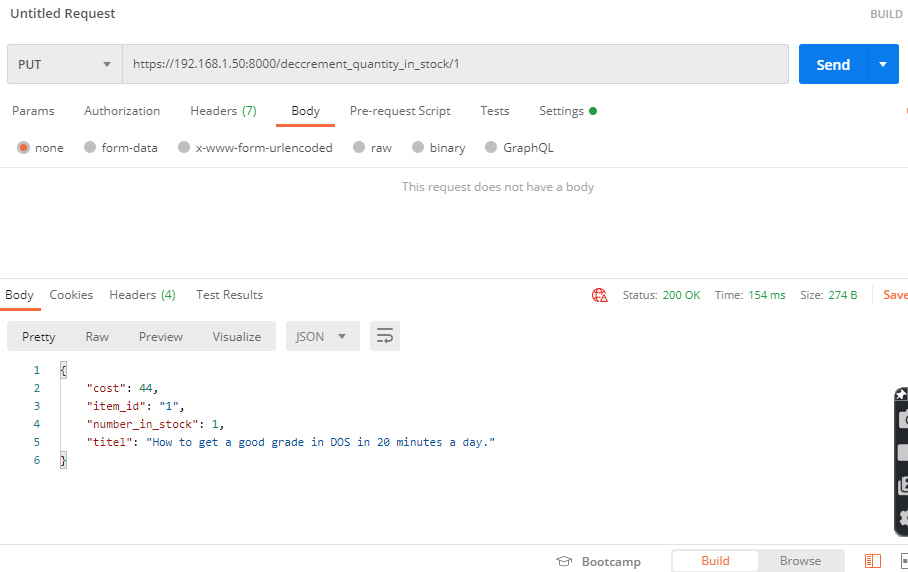
6- increasing quantity in stock .



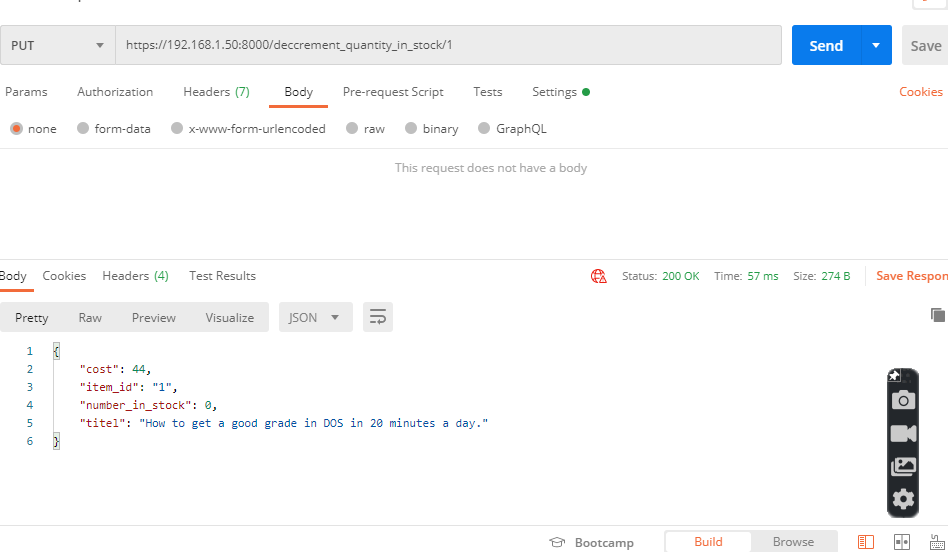
7- decreasing quantity In stock :



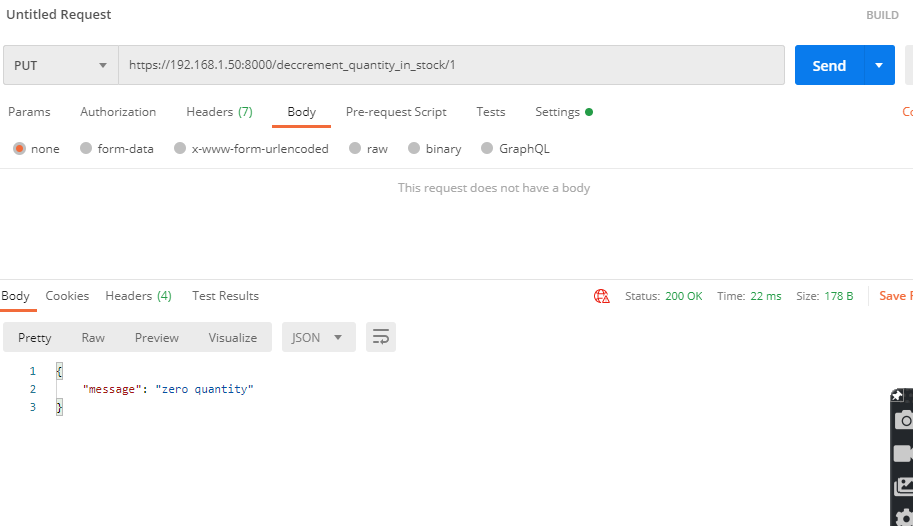
keep decreasing till quantity becomes 1 :



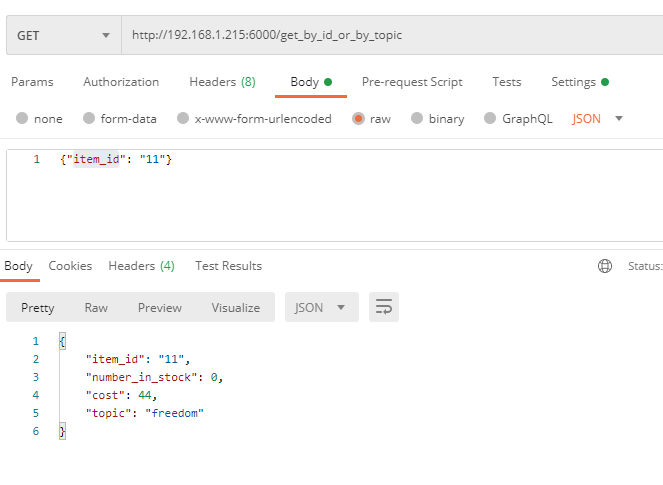
then decreasing another time:

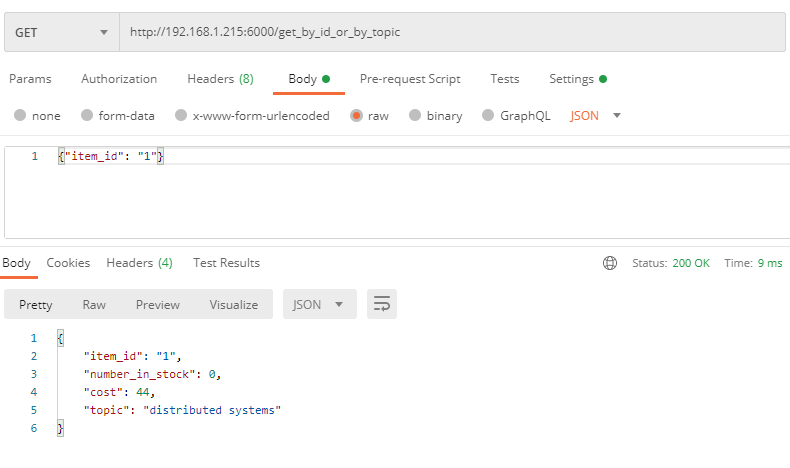


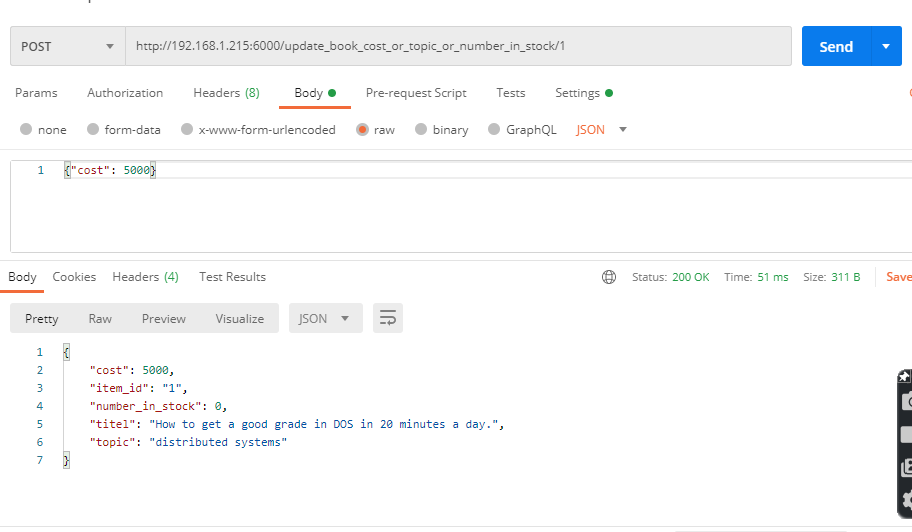
finally decrease another time:

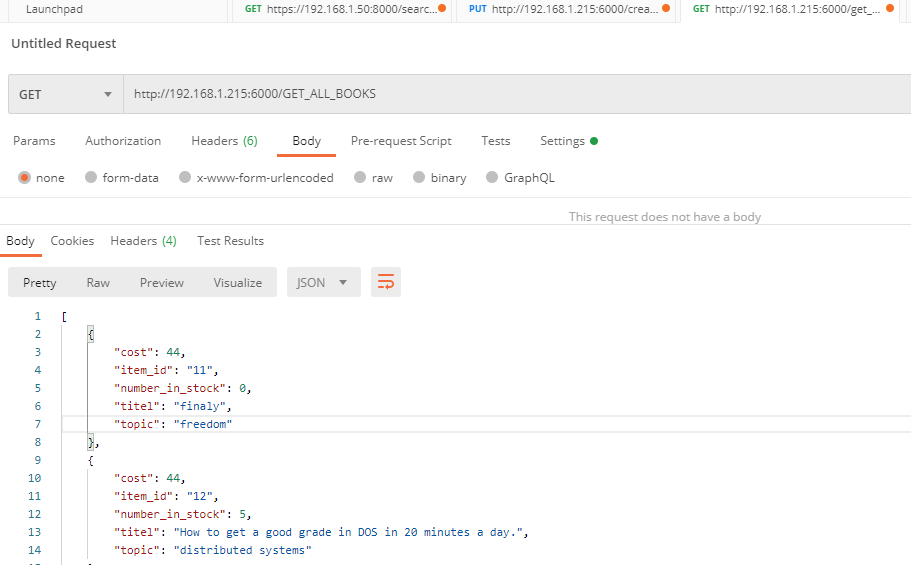


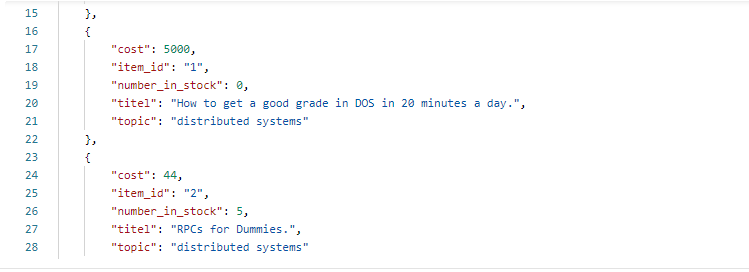
All functionality of front end has been viewed . next the is some of the test cases on functionality catalog server provides (normal users can NOT reach from front end server).

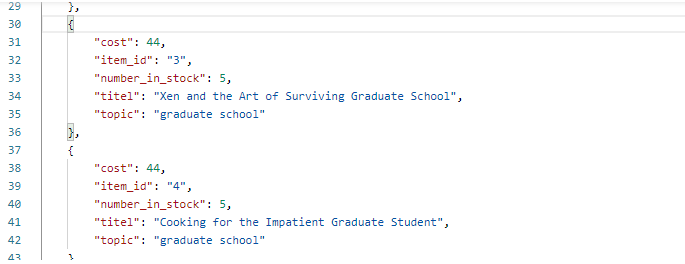


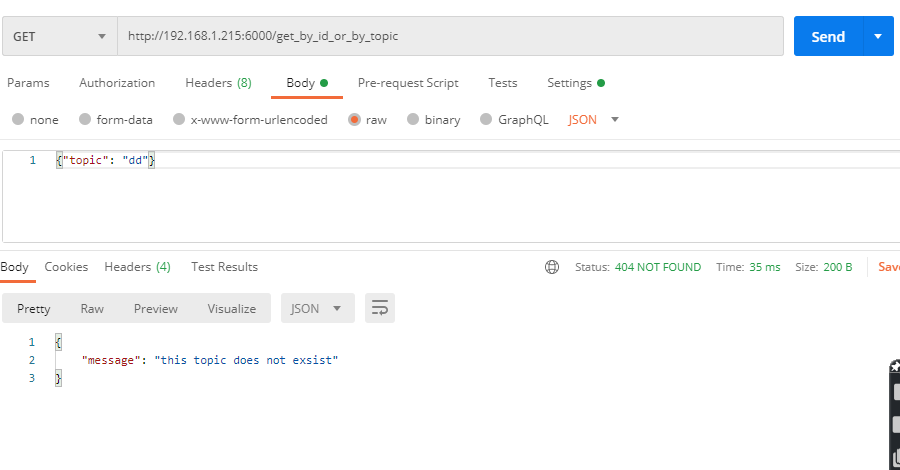


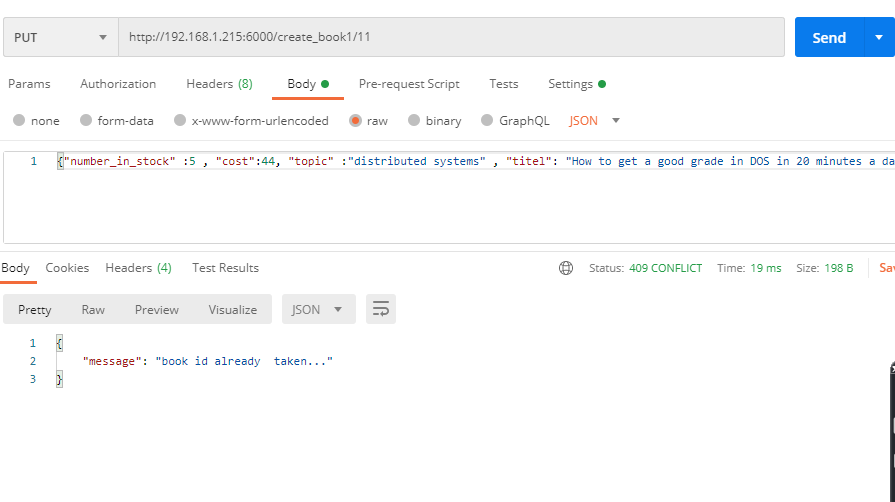


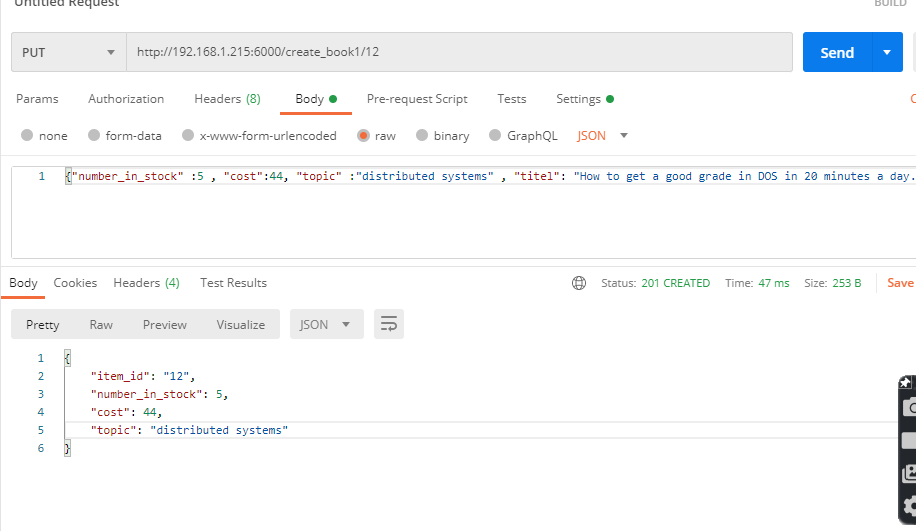


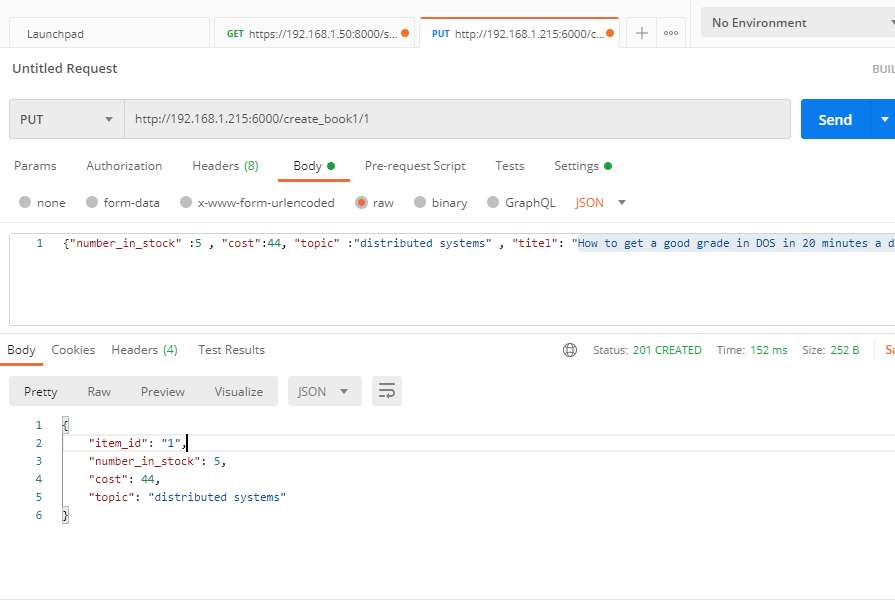


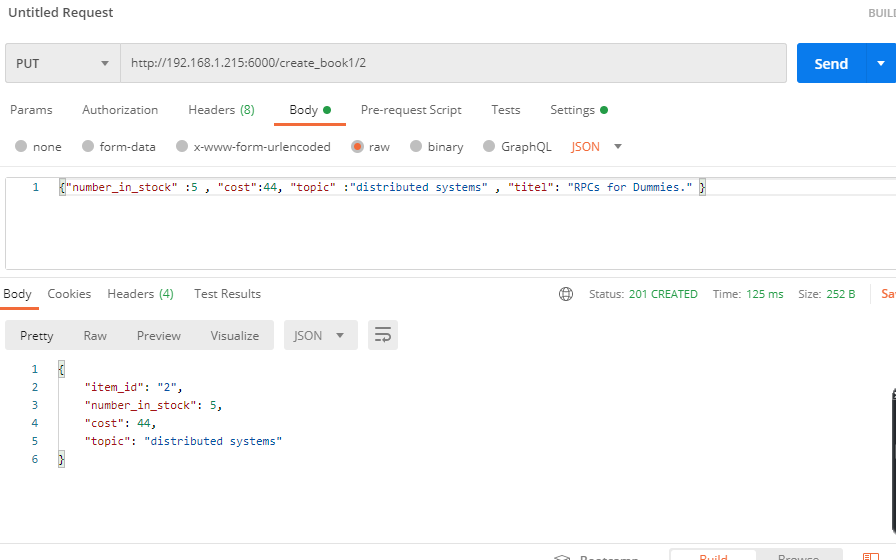


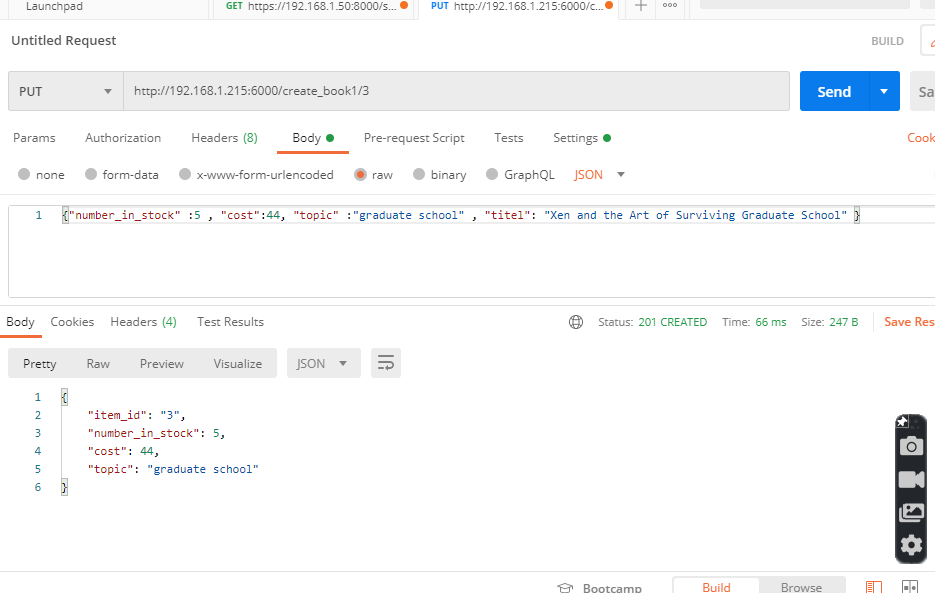


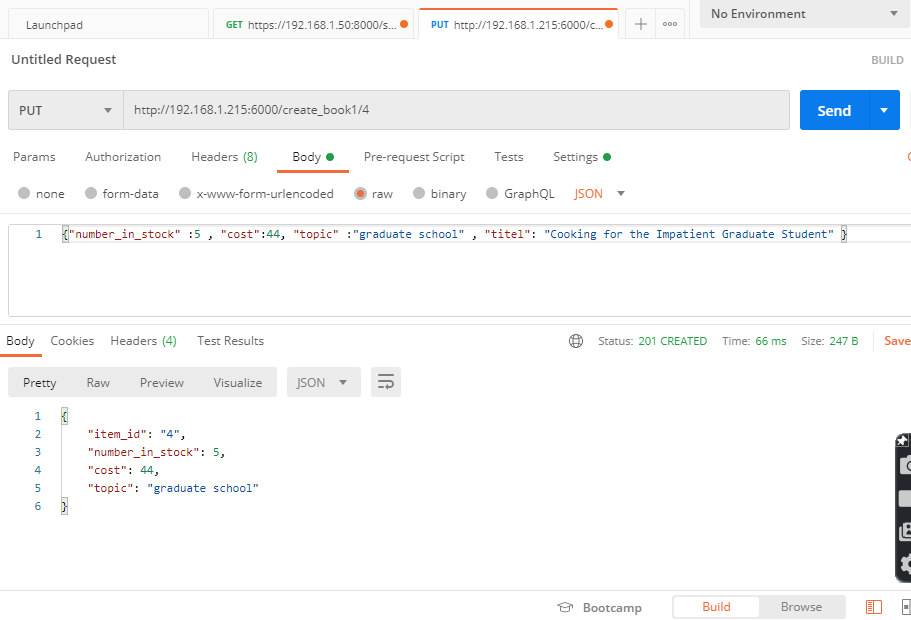






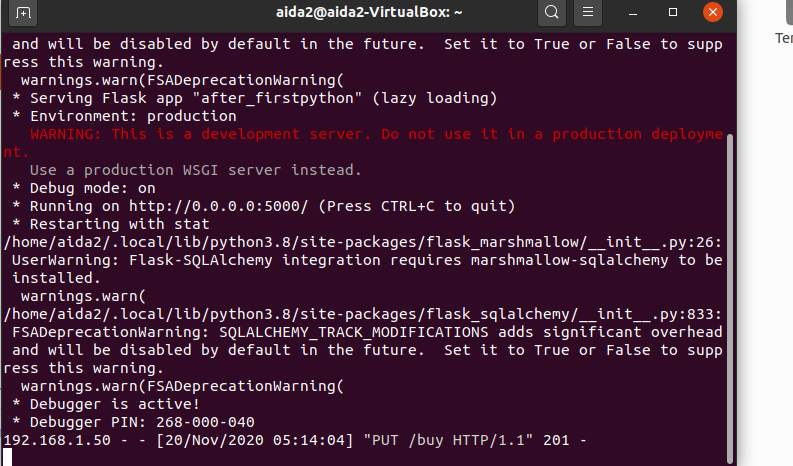






Many other end points are there working but I think this is enogh for demonstarion .

but here is example on what is printed on vm2 terminal when buy command:



here is example on what is printed on vm1 terminal when excuting some commands:

