

Dos project report

Part 1 and 2

**Dr.Samer Arandi**

**Anwar Salman**

**Thara’a Ali Abuhamed**

In the **first part** we have implemented three servers

1-The front end server supports three operations:

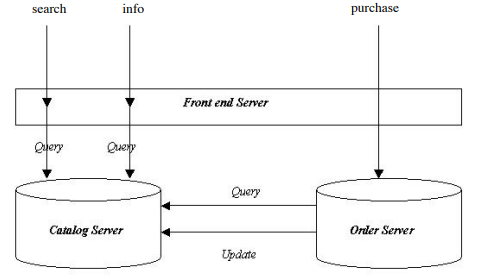
a-search(topic)

b- info(item\_number)

c- purchase(item\_number)

2- The catalog server supports two operations: query and update

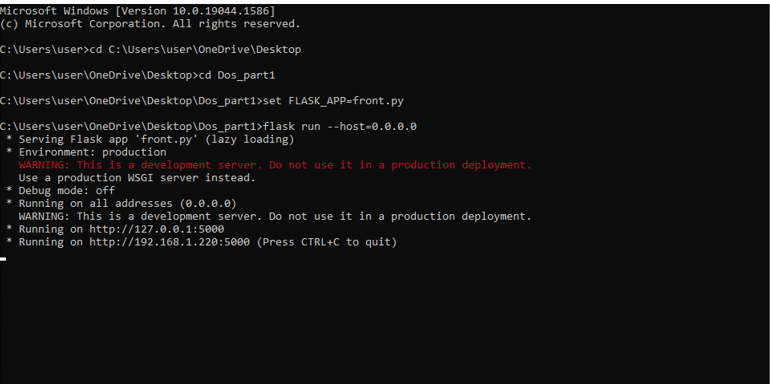
3- The order server supports a single operation: purchase(item\_number).



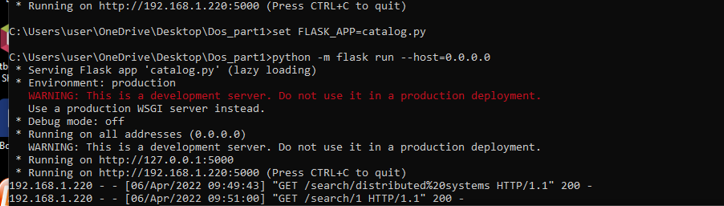
How we run the program:

We run the three servers at the same time and start send requests from postman to each server depend on each operation with require parameters.( all servers run on port 5000)

Frontend server run on ip:192.168.1.220

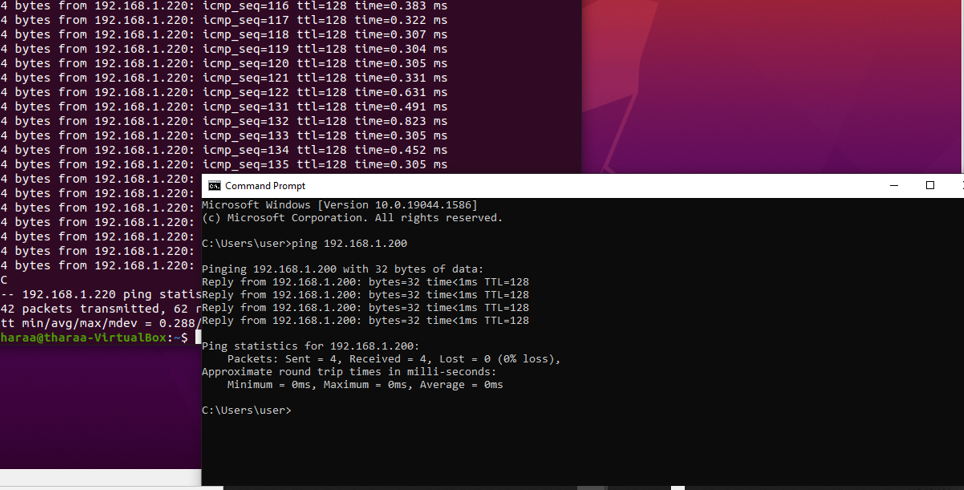


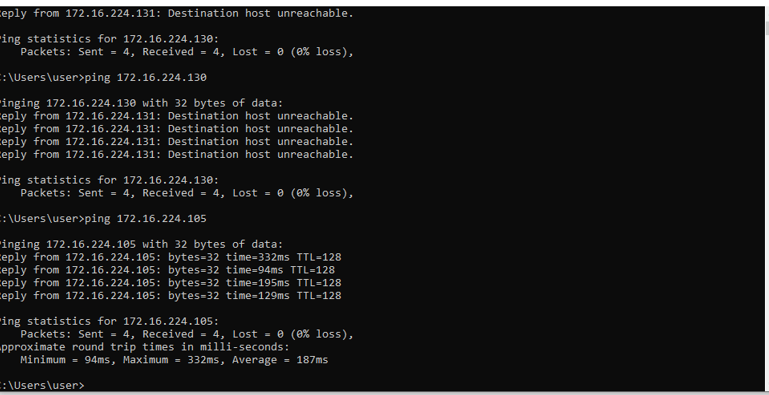
Catalog server run on ip: 192.168.1.220



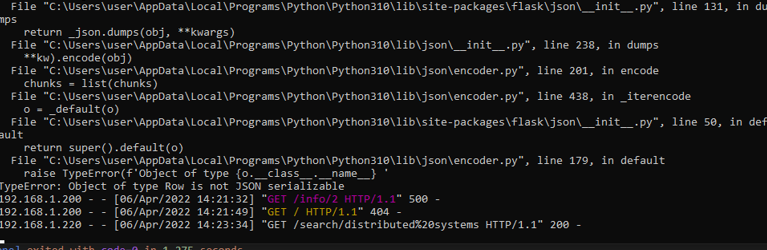
Order server run on ip: 192.168.1.220

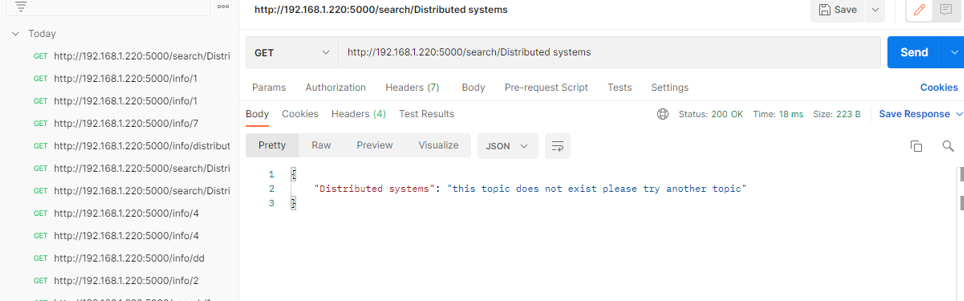
Pending between machines in order to connect between them

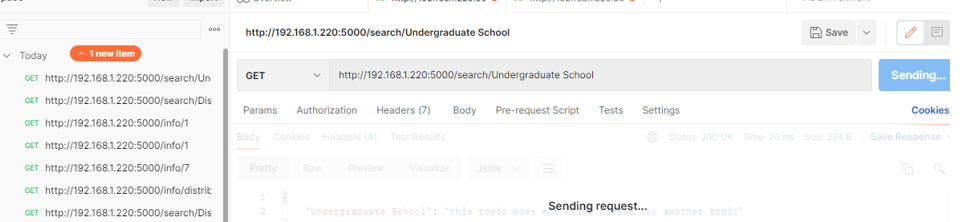




Results:







Part 2 of the project

**Overview**

In this part of the project, we will improve the first part that was sent by adding replication to the servers except for the front end that receives requests from clients where it first checks cache before redirecting a request file to the catalog server where it receives read requests and writes it is sent to the catalog must be processed by request servers or catalog instead of cache.

**How it works?**

In order for the program to run, each server must be placed individually, and the client can use the postman to send requests to the Front End, who will check the process using Reset services.

If the request already exists, it uses the cache server; if it does not exist, it sends it back to the catalog server.

**Load balance:**

We used a round robin algorithm to distribute requests between the servers that we copied. We made a counter whose value ranges between zero and one. If the counter is zero, it sends the request to the first catalog and increases the counter to (1) amount. If the counter is 1, it sends the request to the second catalog and returns the counter 0.

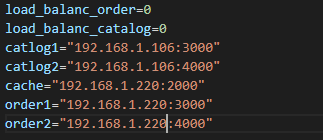
**Replication and Caching**

We made two replicas of the catalog server and the order server (two order servers and two catalog servers), except for Front End, giving each server a different paper port from the other.

We set a maximum size of 10 for cash in the event of a new request, the oldest unused requests are removed using LRU.

* Here we create two identical copies of the catalog server and order server then give them different port numbers

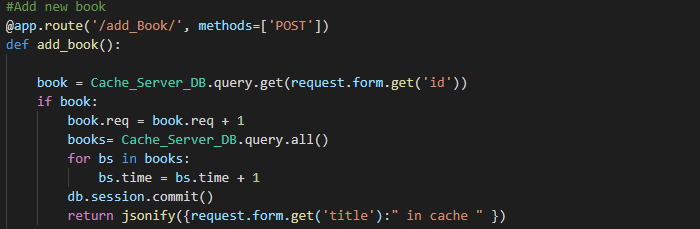


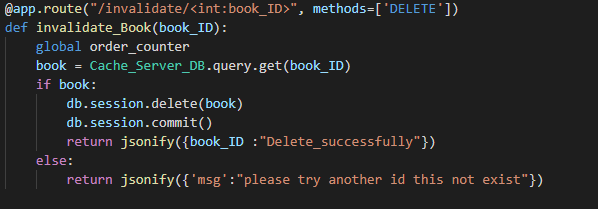


Here we set a maximum size of 10 for cashe ,in addition we create counter

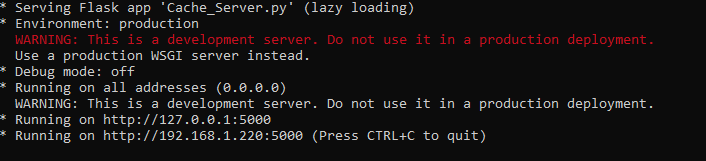


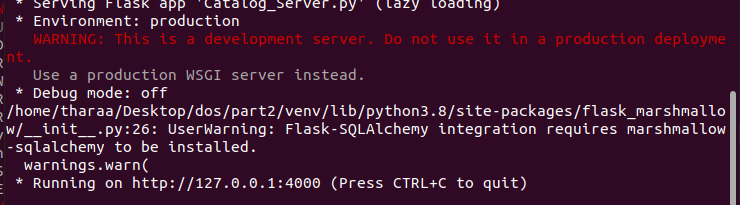
Cashe\_server code :

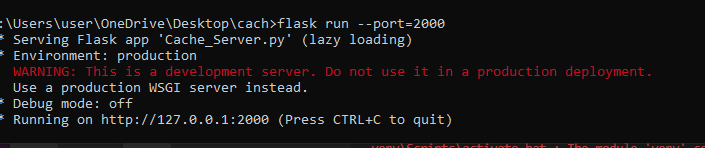


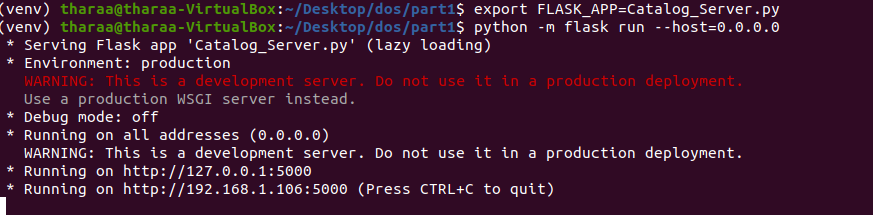


* Because of my knowledge of the cache material, it takes less time than I sent a request to the server and waited for it to return the data or repository to add the cache.I have to, if I have a specific item and I want to do some edit from it like edit price, I have to show a certain one, the server have to send to the cash that I have this information for the book and it invalidate (deleted it).
* The attached images below show how to run each of the files on its port, where the first and second catalogs were run on port 3000 and 4000 respectively, front on 5000, cache on 2000 and order on 3000

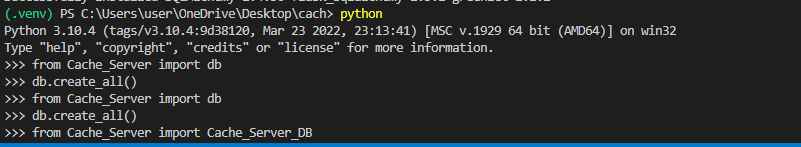




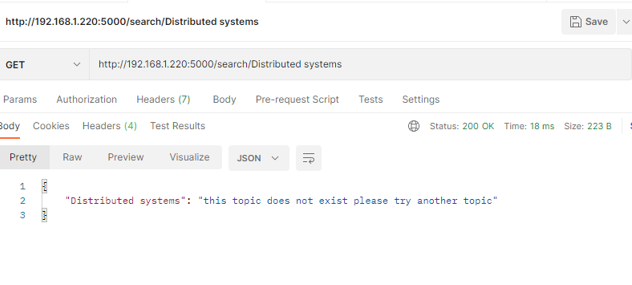
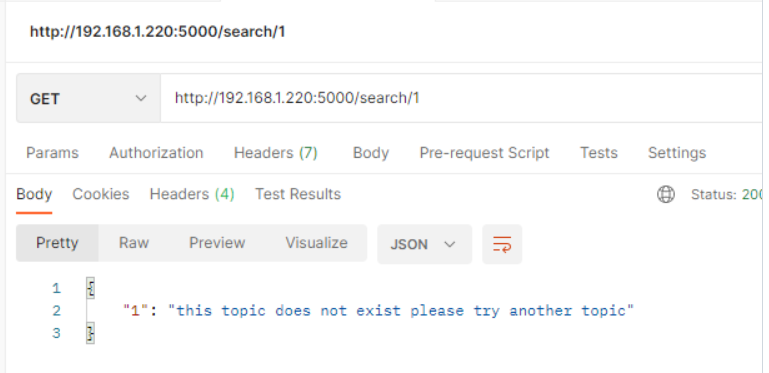




And we created the data base for the cache as follows:



Some of result :



**Unfortunately, we encountered technical problems with our devices and we could not get the results as required, but with the presence of the cache, it is expected that the performance will become better when the request is sent more than once, and therefore it takes less time to respond.**