

An-Najah National University
Computer Engineering Department
Distributed Operating Systems - 10636456

Microservices Lab-Bazar.com



Dr.Samer Arandi
Aya Farhan Suwadeh-11717238

➤ **System Design:**

This section describes the system design and how all its components work all together smoothly, and exchange data and requests responses across the three microservices,

First of all this system (bazar.com) consists of 3 components or microservices, in that each microservice runs on a separate machine and responsible of small set of operations, servers are front server which responsibility is routing and redirecting the coming request into the corresponding responsible backend server, two backend servers, one make the purchase operation (order server), and the other do the info, search and update operations on books (catalog server), and as mentioned each runs separately on different machines, by in my case i run front server on one virtual machine and the other two backends on another virtual machine, and i hit request on the front server from the host machine.

➔ **Implementation and how it works:**

All microservices scripts are written in flask framework(python language), which is a lightweight framework that supports microservices and they all are rest apis, that they interact with each other in a rest pattern over http,

First server-front server, takes the coming request from the client (postman for example) and according to the operation and the requested resource in the uri it redirects the request to one of the backends, purchase to one server and info or search to the other server, the database used in this system is sqlite as a proof of concept, each server or microservice, manage its own database, as required and needed.

➔ **System tradeoffs in implementation and designing:**

This system design follows the microservice architecture, which is good for modularity, reusability, and heterogeneity in implementations and frameworks but in the same time i requires a good network infrastructure for a good communication, as also the interaction between the processes and services is more complex, requires more efforts in management, and for my design i implemented the 3 servers on 2 machine to reduce the overhead on my computer, and to make faster and less heavy for the computer.

➔ **Improvements and extensions:**

when scaling up the system, for better performance it's better to separate the Backend servers, and also we can make a centralised database in case the data was not too huge, this would reduce communication overhead and storage.

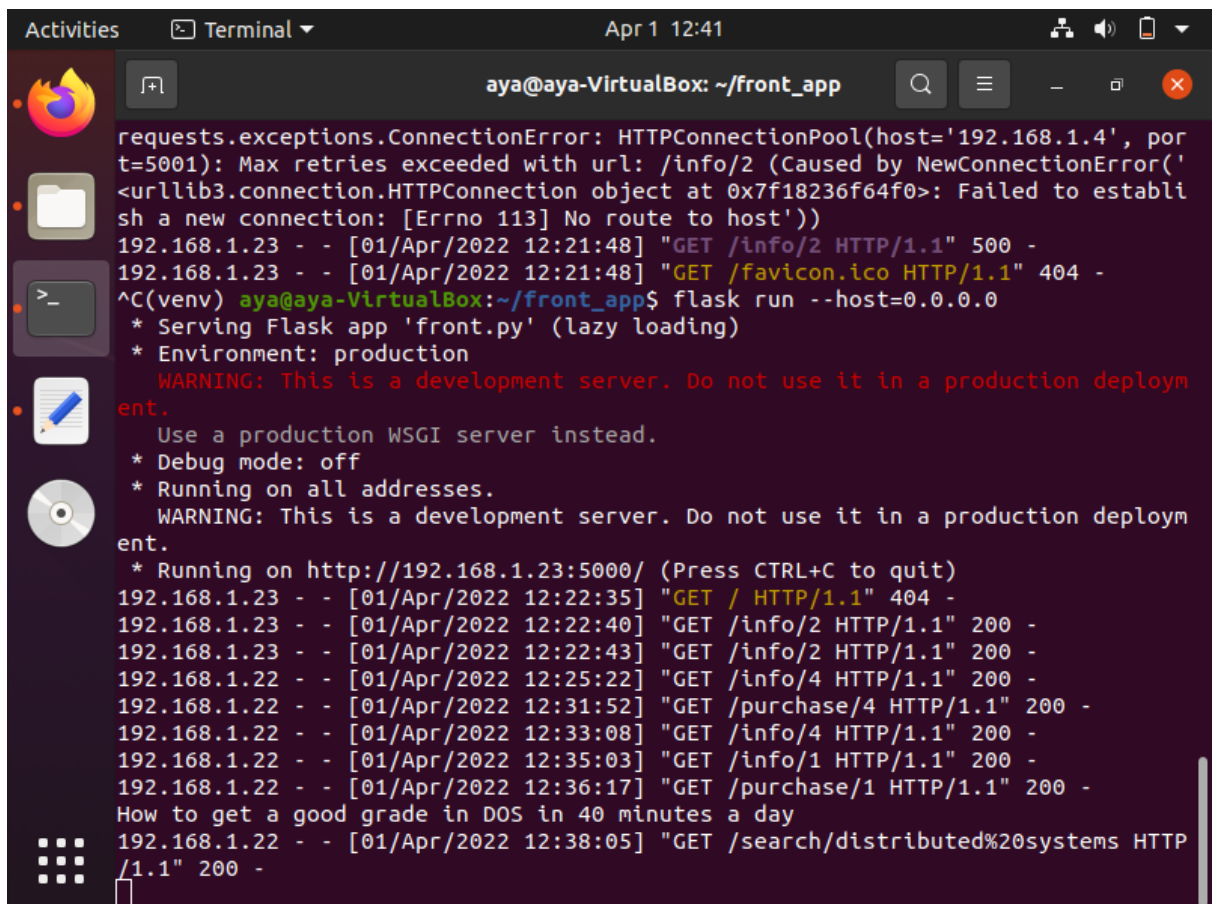
→ Malfunctioning cases :

To handle concurrent requests we need to implement threads, for that I rely on the framework-flask technique, so testing this case was not introduced, also my IPs for servers are dynamic each run has a different IP which requires exposing this IP again into the interconnected servers, and if a server on a machine went down we will lose the communication and the service on that machine until it works again, and for purchase operation we need to specify the method to post otherwise it wouldn't work.

➤ Run the system:

This system requires installing python, flask framework and sqlite packages, as it also needs json http request response packages for implementation.

1. Turn on each server on each machine separately,



```
aya@aya-VirtualBox: ~/front_app
requests.exceptions.ConnectionError: HTTPConnectionPool(host='192.168.1.4', port=5001): Max retries exceeded with url: /info/2 (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f18236f64f0>: Failed to establish a new connection: [Errno 113] No route to host'))
192.168.1.23 - - [01/Apr/2022 12:21:48] "GET /info/2 HTTP/1.1" 500 -
192.168.1.23 - - [01/Apr/2022 12:21:48] "GET /favicon.ico HTTP/1.1" 404 -
^C(venv) aya@aya-VirtualBox:~/front_app$ flask run --host=0.0.0.0
* Serving Flask app 'front.py' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://192.168.1.23:5000/ (Press CTRL+C to quit)
192.168.1.23 - - [01/Apr/2022 12:22:35] "GET / HTTP/1.1" 404 -
192.168.1.23 - - [01/Apr/2022 12:22:40] "GET /info/2 HTTP/1.1" 200 -
192.168.1.23 - - [01/Apr/2022 12:22:43] "GET /info/2 HTTP/1.1" 200 -
192.168.1.22 - - [01/Apr/2022 12:25:22] "GET /info/4 HTTP/1.1" 200 -
192.168.1.22 - - [01/Apr/2022 12:31:52] "GET /purchase/4 HTTP/1.1" 200 -
192.168.1.22 - - [01/Apr/2022 12:33:08] "GET /info/4 HTTP/1.1" 200 -
192.168.1.22 - - [01/Apr/2022 12:35:03] "GET /info/1 HTTP/1.1" 200 -
192.168.1.22 - - [01/Apr/2022 12:36:17] "GET /purchase/1 HTTP/1.1" 200 -
How to get a good grade in DOS in 40 minutes a day
192.168.1.22 - - [01/Apr/2022 12:38:05] "GET /search/distributed%20systems HTTP/1.1" 200 -
```

This is the run environment of the front server, I used `--host=0.0.0.0` with **run flask** command to expose the server to the public, so we can see it took **92.168.1.23** IP address, so when calling this server from Postman for example, the IP address in the URL will be this,

```

aya@aya-VirtualBox:~/catalog_app$ source venv/bin/activate
(venv) aya@aya-VirtualBox:~/catalog_app$ export FLASK_APP=query_update.py
(venv) aya@aya-VirtualBox:~/catalog_app$ flask run --host 0.0.0.0 --port 5001
* Serving Flask app 'query_update.py' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://192.168.1.24:5001/ (Press CTRL+C to quit)
192.168.1.24 - - [01/Apr/2022 12:17:37] "GET / HTTP/1.1" 404 -
192.168.1.24 - - [01/Apr/2022 12:17:38] "GET /favicon.ico HTTP/1.1" 404 -
192.168.1.24 - - [01/Apr/2022 12:17:53] "GET /info/2 HTTP/1.1" 200 -
127.0.0.1 - - [01/Apr/2022 12:18:10] "GET /info/2 HTTP/1.1" 200 -
127.0.0.1 - - [01/Apr/2022 12:18:59] "GET /query/2 HTTP/1.1" 200 -
127.0.0.1 - - [01/Apr/2022 12:18:59] "PUT /update/2/amount_of_items/-1 HTTP/1.1" 200 -
127.0.0.1 - - [01/Apr/2022 12:19:07] "GET /info/2 HTTP/1.1" 200 -
192.168.1.23 - - [01/Apr/2022 12:22:40] "GET /info/2 HTTP/1.1" 200 -

```

Using the commands shown in the figure above i could run the catalog server, its public ip = **192.168.1.24** and binded it to **5001** port because i run 2 servers on this machine so each server binded to a different port, which make this server resources reachable from out this machine and server,

```

aya@aya-VirtualBox:~$ cd order_app
aya@aya-VirtualBox:~/order_app$ source venv/bin/activate
(venv) aya@aya-VirtualBox:~/order_app$ export FLASK_APP=purchase.py
(venv) aya@aya-VirtualBox:~/order_app$ flask run --host=0.0.0.0 --port 5000
* Serving Flask app 'purchase.py' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://192.168.1.24:5000/ (Press CTRL+C to quit)
^C(venv) aya@aya-VirtualBox:~/order_app$ flask run --host=0.0.0.0 --port 5000
* Serving Flask app 'purchase.py' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://192.168.1.24:5000/ (Press CTRL+C to quit)
192.168.1.24 - - [01/Apr/2022 12:18:51] "GET / HTTP/1.1" 404 -
192.168.1.24 - - [01/Apr/2022 12:18:51] "GET /favicon.ico HTTP/1.1" 404 -
192.168.1.24 - - [01/Apr/2022 12:18:59] "GET /purchase/2 HTTP/1.1" 200 -
192.168.1.23 - - [01/Apr/2022 12:31:52] "GET /purchase/4 HTTP/1.1" 200 -
192.168.1.23 - - [01/Apr/2022 12:36:17] "GET /purchase/1 HTTP/1.1" 200 -

```

This is how i run the order server and expose it to the public through the ip address **192.168.1.24** port **5000**,

2. Start testing the endpoints by requesting them

