



Computer Engineering Department	
Course Name: Distributed Operating Systems	Number: 10636456/1394
Report	
Academic Year: 2022/2023	Semester: 1 st
Homework 2	
Topic: Microservices (Bazar.com - A Multi-tier Online Book Store)	
Instructor Name:	
Dr. Samer Arandi	
Student:	
Name: Gharam Jamil Sarsour	ID: 11819668



Testing the API using different methods

- **How does it work?**

- 1- It started with a simple python API for the catalog on the host operating system, because it is the server that both the order and the frontend send requests to, then a csv file was created to store the data and used the catalog server to manipulate the data inside it, after that the server was run on the Ip of the device and tested to make sure there was no errors.
- 2- Next the order server was created on a virtual machine the same way, and it was tested on the device Ip, then the redirect for the catalog server was added and tested by putting the Ip and port of the catalog server (host os).
- 3- Finally, it was time to create the frontend server, which was done on a second virtual machine, and after adding the redirecting for the catalog with its Ip and port, and adding the redirecting for the order with its Ip and port, the front end was tested using postman API platform, all virtual machines was using bridge network adapter which allowed for the http requests to be directed from one server to another, and ending the chain of request with a response to the frontend server.

In Summary: the frontend API receives http requests from the user then redirects the requests to the catalog and order servers, if the request is search or info, it sends a query request to the catalog server with the parameters it has, and it takes the response from the catalog and sends it as a response to the user, if the request is a purchase, the frontend requests a purchase from the order with the item number as a parameter, the order request takes the item number and requests an update request to the catalog to decrease the stock, then it takes the response, sends it to the frontend, and the frontend sends it to the user.

- **Design Trade off!**

The application is relatively small witch means there is more complexity than needed, at the same time if we want to scall some part of the application and update, there is no need to shut all the services to update it, only the needed one.

- **Possible Improvement?**

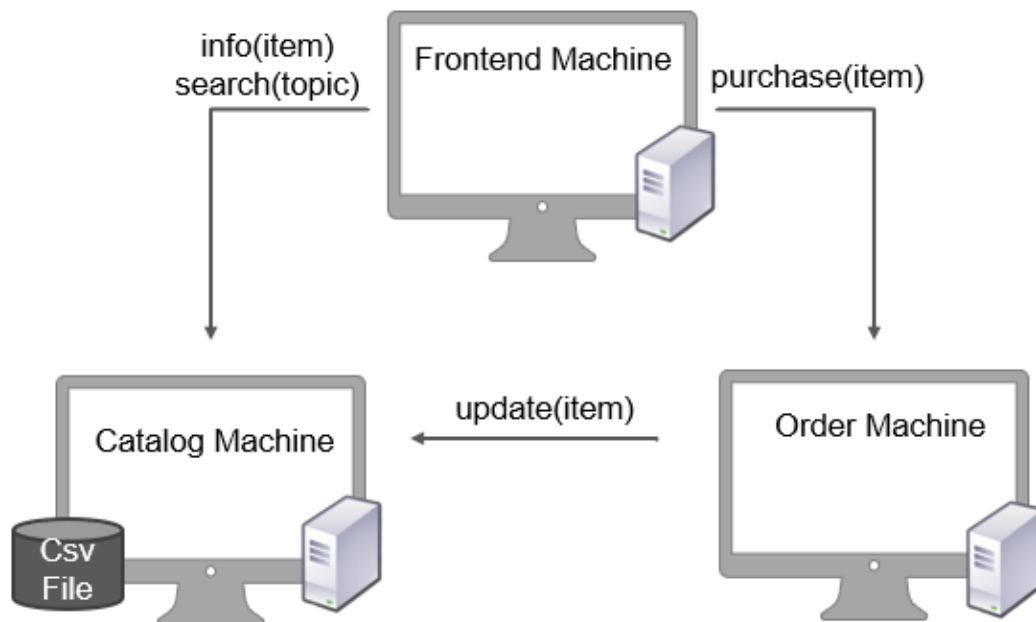
Adding an operation to edit the prices and stock of the catalog would be useful to make the book store more efficient.



- **How to run the program?**

- 1- First, we need three machines or rather 3 operating systems, we need to ensure that they are all running on a bridge network adapter, we also need to have flask working successfully on all the machines.
- 2- Then, we put each of the codes in a machine, then we get the Ip of each machine and use them in the right codes, we put the catalog Ip in both frontend and order servers, we put the order Ip in the frontend server.
- 3- After we save the codes with the new Ips, we run each server with the command (flask run -h server_ip -p server_port).
- 4- Finally we have all the servers working and we can start sending requests to the frontend server and get responses in return.

- **The Design**





- The output
the csv file with data:

```

app.py data.csv X
catalog > data.csv
1 item_number,title,topic,stock,cost
2 1,How to get a good grade in DOS in 40 minutes a day,distributed systems,20,35
3 2,RPCs for Noobs,distributed systems,30,30
4 3,Xen and the Art of Surviving Undergraduate School,undergraduate school,44,50
5 4,Cooking for the Impatient Undergrad,undergraduate school, 25,30
6

```

the catalog server running on 192.168.1.101 and port 5000:

```

app.py X data.csv
catalog > app.py > update
72 header = next(csvreader)
73 for row in csvreader:
74     rows.append(row)
75 for row in rows:
76     print(row)
77 for row in rows:
78     if row[0] == item:
79         title=row[1]
80         stock=row[3]

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```

['4', 'Cooking for the Impatient Undergrad', 'undergraduate school', ' 25', '30']
45
['item_number', 'title', 'topic', 'stock', 'cost']
[['1', 'How to get a good grade in DOS in 40 minutes a day', 'distributed systems', '20', '35'], ['2', 'RPCs for Noobs', 'distributed systems', '30', '30'], ['3', 'Xen and the Art of Surviving Undergraduate School', 'undergraduate school', '45', '50'], ['4', 'Cooking for the Impatient Undergrad', 'undergraduate school', ' 25', '30']]
192.168.1.106 - - [22/Oct/2022 17:25:43] "PUT /update?item=3 HTTP/1.1" 201 -
PS C:\Users\Asus\Desktop\catalog> flask run -h 192.168.1.101 -p 5000
* 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 http://192.168.1.101:5000/ (Press CTRL+C to quit)

```



the order server running on a different machine with ip 192.168.1.106

```

1 from flask import Flask, request
2
3 import requ
4
5 my_app=Flas
6
7 192.168.1.104 - - [22/Oct/2022 16:58:09] "PUT /purchase?item=3 HTTP/1.1" 201 -
8 192.168.1.104 - - [22/Oct/2022 16:59:05] "PUT /purchase?item=3 HTTP/1.1" 201 -
9 @my_app.rou192.168.1.104 - - [22/Oct/2022 16:59:55] "PUT /purchase?item=3 HTTP/1.1" 201 -
10 def purchas192.168.1.104 - - [22/Oct/2022 17:04:44] "PUT /purchase?item=3 HTTP/1.1" 201 -
11 item=re192.168.1.104 - - [22/Oct/2022 17:08:37] "PUT /purchase?item=3 HTTP/1.1" 201 -
12 ip="192192.168.1.104 - - [22/Oct/2022 17:09:36] "PUT /purchase?item=3 HTTP/1.1" 201 -
13 port="s192.168.1.104 - - [22/Oct/2022 17:10:50] "PUT /purchase?item=3 HTTP/1.1" 201 -
14 URL = "192.168.1.104 - - [22/Oct/2022 17:11:29] "PUT /purchase?item=3 HTTP/1.1" 201 -
15 PARAMS 192.168.1.104 - - [22/Oct/2022 17:12:13] "PUT /purchase?item=3 HTTP/1.1" 201 -
16 r = req192.168.1.104 - - [22/Oct/2022 17:13:34] "PUT /purchase?item=3 HTTP/1.1" 201 -
17 data=r.^Cgharam@gharam-VirtualBox:~$ flask run -h 192.168.1.106 -p 4000
18 * Debug mode: off
19 WARNING: This is a development server. Do not use it in a production deployment.
20 Use a production WSGI server instead.
21 * Running on http://192.168.1.106:4000
22 Press CTRL+C to quit
23 @my_app.err192.168.1.104 - - [22/Oct/2022 17:25:43] "PUT /purchase?item=3 HTTP/1.1" 201 -
24 def handlerCgharam@gharam-VirtualBox:~$ flask run -h 192.168.1.106 -p 4000
25 * Debug mode: off
26 WARNING: This is a development server. Do not use it in a production deployment.
27 Use a production WSGI server instead.
28 * Running on http://192.168.1.106:4000
29 Press CTRL+C to quit
30 if __name__
31     main()

```

the front end server running on a different machine with ip 192.168.1.104

```

P/1.1" 201 -
192.168.1.101 - - [22/Oct/2022 17:03:18] "GET /search/undergraduate%20school HTTP
P/1.1" 201 -
192.168.1.101 - - [22/Oct/2022 17:04:44] "PUT /purchase/3 HTTP/1.1" 201 -
192.168.1.101 - - [22/Oct/2022 17:08:37] "PUT /purchase/3 HTTP/1.1" 201 -
192.168.1.101 - - [22/Oct/2022 17:09:36] "PUT /purchase/3 HTTP/1.1" 201 -
192.168.1.101 - - [22/Oct/2022 17:10:50] "PUT /purchase/3 HTTP/1.1" 201 -
192.168.1.101 - - [22/Oct/2022 17:11:29] "PUT /purchase/3 HTTP/1.1" 201 -
192.168.1.101 - - [22/Oct/2022 17:12:13] "PUT /purchase/3 HTTP/1.1" 201 -
192.168.1.101 - - [22/Oct/2022 17:13:34] "PUT /purchase/3 HTTP/1.1" 201 -
^Cgharam2@gharam2-VirtualBox:~$ flask run -h 192.168.1.104 -p 3000
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Running on http://192.168.1.104:3000
Press CTRL+C to quit
192.168.1.101 - - [22/Oct/2022 17:25:43] "PUT /purchase/3 HTTP/1.1" 201 -
^Cgharam2@gharam2-VirtualBox:~$ flask run -h 192.168.1.104 -p 3000
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Running on http://192.168.1.104:3000
Press CTRL+C to quit

```



sending a search request to the frontend server using postman

http://192.168.1.104:3000/search/undergraduate school

GET http://192.168.1.104:3000/search/undergraduate school

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results 201 CREATED 115 ms 333 B Save Response

Pretty Raw Preview Visualize HTML

```
1 [{"item_number": "3", "title": "Xen and the Art of Surviving Undergraduate School"}, {"item_number": "4", "title": "Cooking for the Impatient Undergrad"}]
```

ubu3 [Running] - Oracle VM VirtualBox

```
192.168.1.101 - - [22/Oct/2022 17:11:29] "PUT /purchase/3 HTTP/1.1" 201 -
192.168.1.101 - - [22/Oct/2022 17:12:13] "PUT /purchase/3 HTTP/1.1" 201 -
192.168.1.101 - - [22/Oct/2022 17:13:34] "PUT /purchase/3 HTTP/1.1" 201 -
^Cgharam2@gharam2-VirtualBox:~$ flask run -h 192.168.1.104 -p 3000
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production server instead.
* Running on http://192.168.1.104:3000
Press CTRL+C to quit
192.168.1.101 - - [22/Oct/2022 17:25:43] "PUT /purchase/3 HTTP/1.1" 201 -
^Cgharam2@gharam2-VirtualBox:~$ flask run -h 192.168.1.104 -p 3000
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production server instead.
* Running on http://192.168.1.104:3000
Press CTRL+C to quit
192.168.1.101 - - [22/Oct/2022 17:34:47] "GET /search/undergraduate%20school HTTP/1.1" 201 -
```

The request in the frontend side

```
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 http://192.168.1.101:5000/ (Press CTRL+C to quit)
192.168.1.104 - - [22/Oct/2022 17:34:47] "GET /queryTopic?topic=undergraduate+school HTTP/1.1" 201 -
```

The request in the catalog side



sending an info to the frontend server using postman

http://192.168.1.104:3000/info/3

GET http://192.168.1.104:3000/info/3

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results 201 CREATED 453 ms 269 B Save Response

Pretty Raw Preview Visualize HTML

```
1 {"title": "Xen and the Art of Surviving Undergraduate School", "stock": "45", "cost": "50"}
```

ubu3 [Running] - Oracle VM VirtualBox

```
192.168.1.101 - - [22/Oct/2022 17:12:13] "PUT /purchase/3 HTTP/1.1" 201 -
192.168.1.101 - - [22/Oct/2022 17:13:34] "PUT /purchase/3 HTTP/1.1" 201 -
^Cgharam2@gharam2-VirtualBox:~$ flask run -h 192.168.1.104 -p 3000
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production
server instead.
* Running on http://192.168.1.104:3000
Press CTRL+C to quit
192.168.1.101 - - [22/Oct/2022 17:25:43] "PUT /purchase/3 HTTP/1.1" 201 -
^Cgharam2@gharam2-VirtualBox:~$ flask run -h 192.168.1.104 -p 3000
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production
server instead.
* Running on http://192.168.1.104:3000
Press CTRL+C to quit
192.168.1.101 - - [22/Oct/2022 17:34:47] "GET /search/undergraduate%20school HTTP/1.1" 201 -
192.168.1.101 - - [22/Oct/2022 17:39:12] "GET /info/3 HTTP/1.1" 201 -
```

The request in the frontend side

```
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 http://192.168.1.101:5000/ (Press CTRL+C to quit)
192.168.1.104 - - [22/Oct/2022 17:34:47] "GET /queryTopic?topic=undergraduate+school HTTP/1.1" 201 -
3
192.168.1.104 - - [22/Oct/2022 17:39:12] "GET /queryItem?item=3 HTTP/1.1" 201 -
```

The request in the catalog side



sending a purchase request to the frontend server using postman

http://192.168.1.104:3000/purchase/3

PUT http://192.168.1.104:3000/purchase/3

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results

201 CREATED 1022 ms 238 B Save Response

Pretty Raw Preview Visualize HTML

The response when the item is in stock

1 purchased Xen and the Art of Surviving Undergraduate School

ubu3 [Running] - Oracle VM VirtualBox

* Debug mode: off

WARNING: This is a development server. Do not use it in a production deployment. Use a production server instead.

* Running on http://192.168.1.104:3000

Press CTRL+C to quit

192.168.1.101 - - [22/Oct/2022 17:34:47] "GET /search/undergraduate%20school HTTP/1.1" 201 -

192.168.1.101 - - [22/Oct/2022 17:39:12] "GET /info/3 HTTP/1.1" 201 -

192.168.1.101 - - [22/Oct/2022 17:41:20] "PUT /purchase/3 HTTP/1.1" 201 -

The request in the frontend side

ubu2 [Running] - Oracle VM VirtualBox

* Debug mode: off

WARNING: This is a development server. Do not use it in a production deployment. Use a production server instead.

* Running on http://192.168.1.106:4000

Press CTRL+C to quit

192.168.1.104 - - [22/Oct/2022 17:41:20] "PUT /purchase?item=3 HTTP/1.1" 201 -

The request in the order side

30 main()

['2', 'RPCs for Noobs', 'distributed systems', '30', '30']

['3', 'Xen and the Art of Surviving Undergraduate School', 'undergraduate school', '45', '50']

['4', 'Cooking for the Impatient Undergrad', 'undergraduate school', '25', '30']

44

['item_number', 'title', 'topic', 'stock', 'cost']

[[['1', 'How to get a good grade in DOS in 40 minutes a day', 'distributed systems', '20', '35'], ['2', 'RPCs for Noobs', 'distributed systems', '30', '30'], ['3', 'Xen and the Art of Surviving Undergraduate School', 'undergraduate school', '45', '50'], ['4', 'Cooking for the Impatient Undergrad', 'undergraduate school', '25', '30']]]

192.168.1.106 - - [22/Oct/2022 17:41:20] "PUT /update?item=3 HTTP/1.1" 201 -

The request in the catalog side



http://192.168.1.104:3000/purchase/4

PUT http://192.168.1.104:3000/purchase/4

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (4) Test Results

Status: 201 CREATED Time: 282 ms Size: 210 B

Pretty Raw Preview Visualize HTML

The response when the item is out of stock

```
1 Cooking for the Impatient Undergrad is out of stock
```