An-Najah National University

Faculty of Engineering

Computer Engineering Department

DOS Course

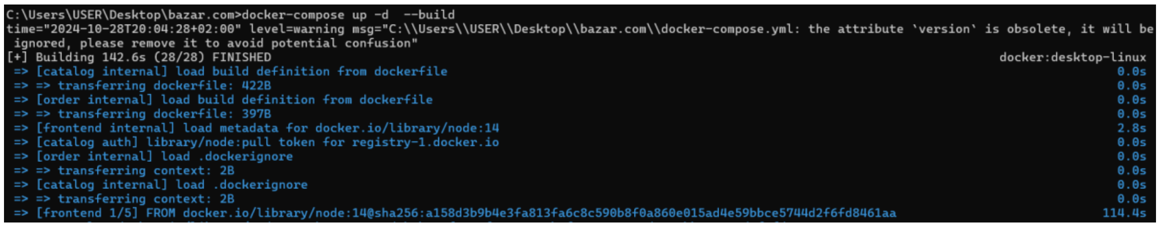
Project part1

Student’s name:

Samah Qaradeh(12028923)

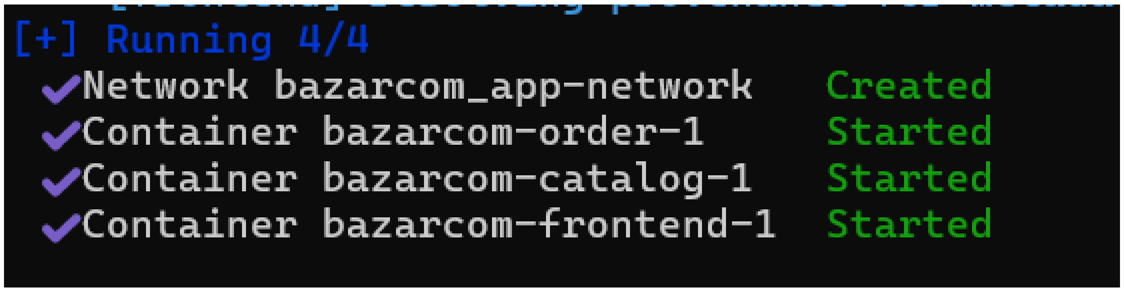
Muna khwaireh(12028473)

In this project we design a bazar.com which is the world's smallest book store, we have front end and back end, the front end has only 1 microservices, while the backend has 2(order and catalog), they talk with each other to do specific task, we make a composer to organize communication with them, once we start we build and run a composer in a docker

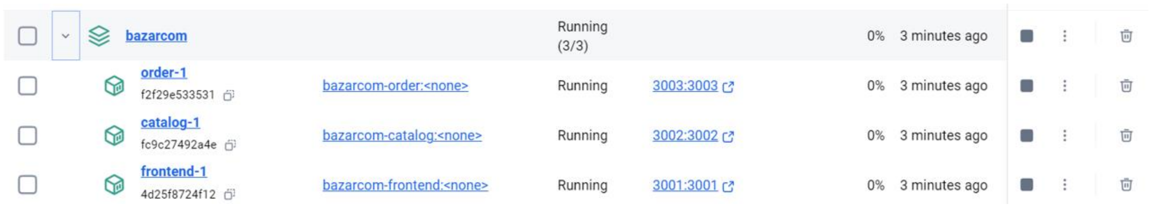


```
C:\Users\USER\Desktop\bazar.com>docker-compose up -d  --build
time="2024-10-28T20:04:28+02:00" level=warning msg="C:\\Users\\USER\\Desktop\\bazar.com\\docker-compose.yml: the attribute `version` is obsolete, it will be
 ignored, please remove it to avoid potential confusion"
[+] Building 142.6s (28/28) FINISHED                                                                                                  docker:desktop-linux
 => [catalog internal] load build definition from dockerfile                                                                                          0.0s
 => => transferring dockerfile: 422B                                                                                                                  0.0s
 => [order internal] load build definition from dockerfile                                                                                            0.0s
 => => transferring dockerfile: 397B                                                                                                                  0.0s
 => [frontend internal] load metadata for docker.io/library/node:14                                                                                   2.8s
 => [catalog auth] library/node:pull token for registry-1.docker.io                                                                                   0.0s
 => [order internal] load .dockerignore                                                                                                               0.0s
 => => transferring context: 2B                                                                                                                       0.0s
 => [catalog internal] load .dockerignore                                                                                                             0.0s
 => => transferring context: 2B                                                                                                                       0.0s
 => [frontend 1/5] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa                           114.4s
```

After build and run the composer , it will run these containers:



```
[+] Running 4/4
 ✔Network bazarcom_app-network    Created
 ✔Container bazarcom-order-1      Started
 ✔Container bazarcom-catalog-1    Started
 ✔Container bazarcom-frontend-1   Started
```



☐	bazarcom		Running (3/3)		0%	3 minutes ago
☐	order-1 f2f29e533531	bazarcom-order:<none>	Running	3003:3003	0%	3 minutes ago
☐	catalog-1 fc9c27492a4e	bazarcom-catalog:<none>	Running	3002:3002	0%	3 minutes ago
☐	frontend-1 4d25f8724f12	bazarcom-frontend:<none>	Running	3001:3001	0%	3 minutes ago

3 container communicate with each other using the network that the compose build. We write the composer, docker files and java script, we submit all these , we use node js.

We don't use any database, we just use a file csv for book,

Operations:

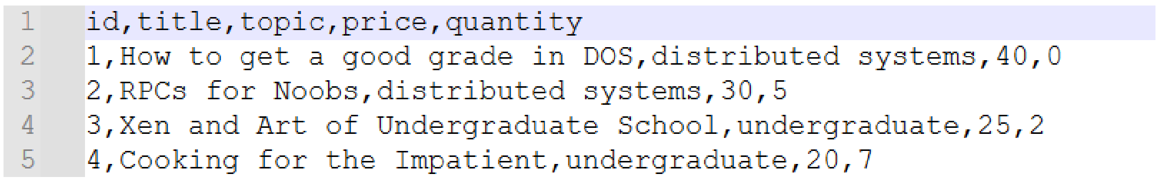
1-localhost:3001/search/topic

2- localhost:3001/info/item number

3- localhost:3003/purchase/ item number

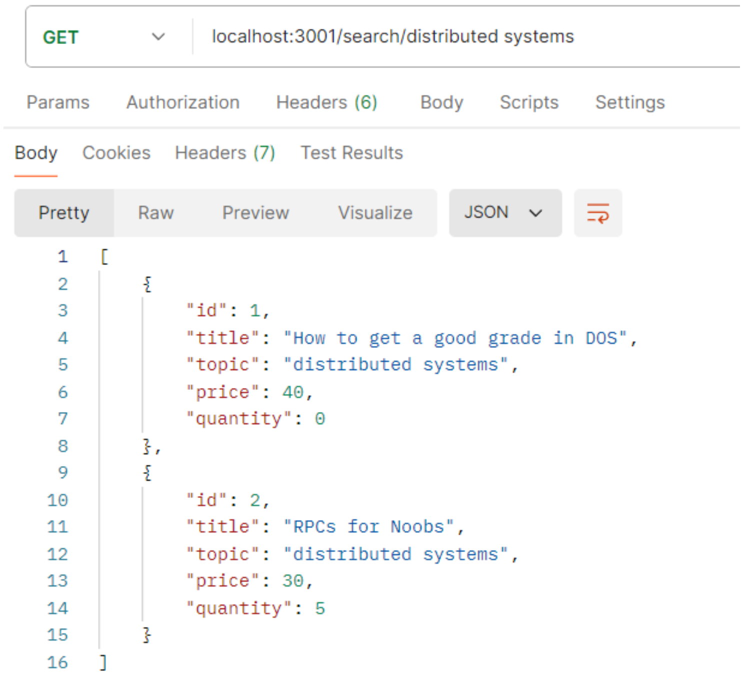
4- localhost:3003/update / item number

We will make a test in postman and see the results, initially we have this data before any query or update. (this is in a csv file)



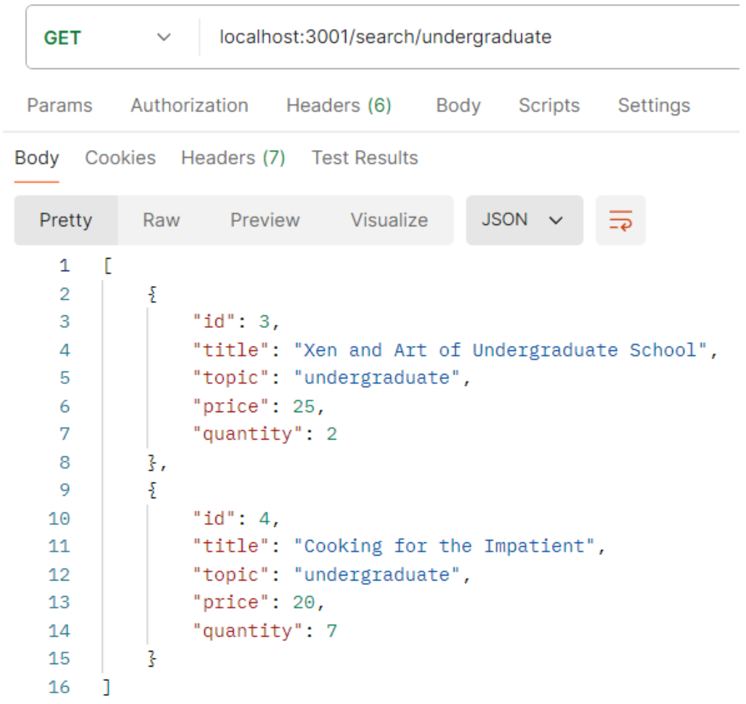
```
id,title,topic,price,quantity
1,How to get a good grade in DOS,distributed systems,40,0
2,RPCs for Noobs,distributed systems,30,5
3,Xen and Art of Undergraduate School,undergraduate,25,2
4,Cooking for the Impatient,undergraduate,20,7
```

We will try to make search depend on the topic, note that we use get method, also we use port 3001 which is front



```
localhost:3001/search/distributed systems
```

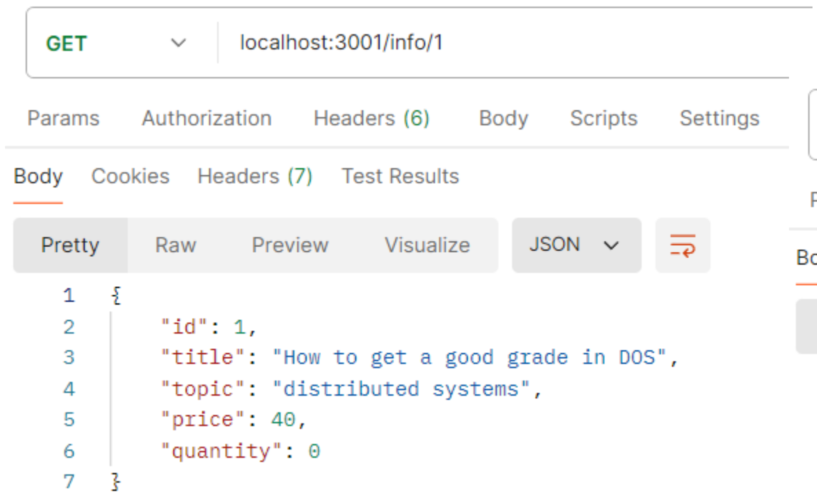
```
[
    {
        "id": 1,
        "title": "How to get a good grade in DOS",
        "topic": "distributed systems",
        "price": 40,
        "quantity": 0
    },
    {
        "id": 2,
        "title": "RPCs for Noobs",
        "topic": "distributed systems",
        "price": 30,
        "quantity": 5
    }
]
```



```
localhost:3001/search/undergraduate
```

```
[
    {
        "id": 3,
        "title": "Xen and Art of Undergraduate School",
        "topic": "undergraduate",
        "price": 25,
        "quantity": 2
    },
    {
        "id": 4,
        "title": "Cooking for the Impatient",
        "topic": "undergraduate",
        "price": 20,
        "quantity": 7
    }
]
```

After that we try to make a query just to have information a bout specific book, we search depend on the ID of it, we use GET method, port 3001 for front



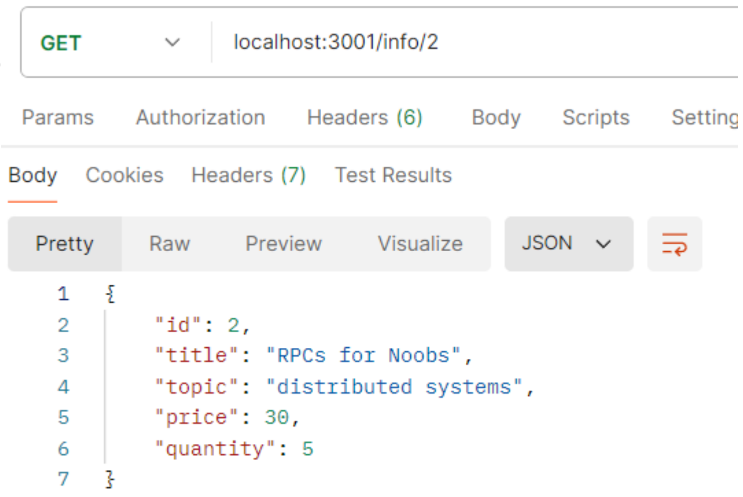
GET localhost:3001/info/1

Params Authorization Headers (6) Body Scripts Settings

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "id": 1,
    "title": "How to get a good grade in DOS",
    "topic": "distributed systems",
    "price": 40,
    "quantity": 0
}
```



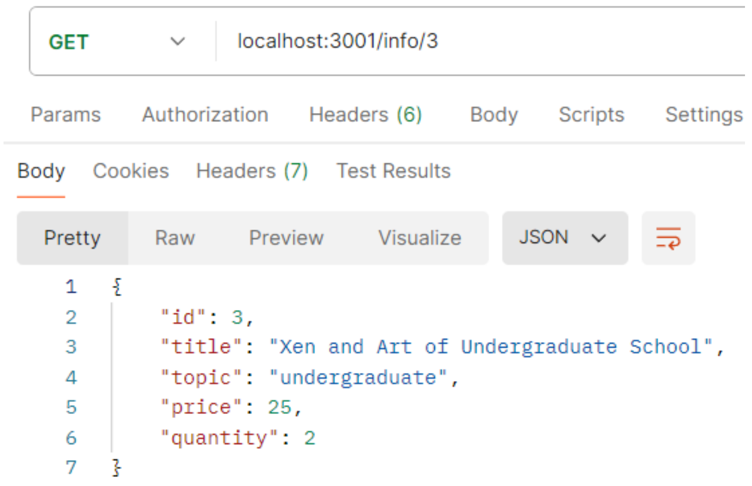
GET localhost:3001/info/2

Params Authorization Headers (6) Body Scripts Setting

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "id": 2,
    "title": "RPCs for Noobs",
    "topic": "distributed systems",
    "price": 30,
    "quantity": 5
}
```



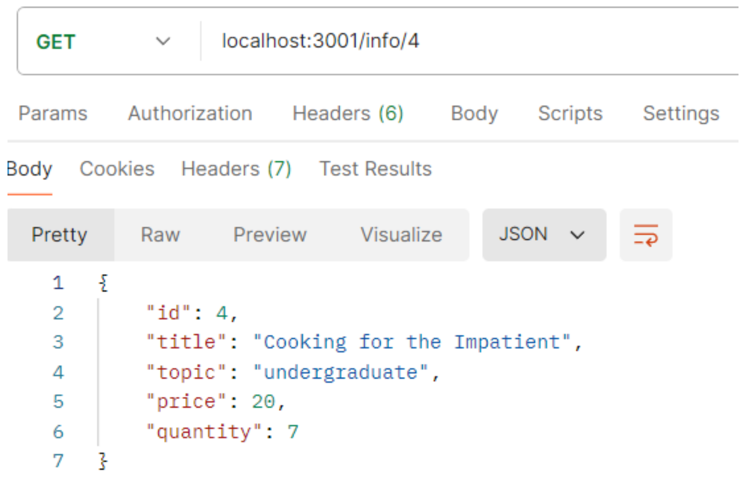
GET localhost:3001/info/3

Params Authorization Headers (6) Body Scripts Settings

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "id": 3,
    "title": "Xen and Art of Undergraduate School",
    "topic": "undergraduate",
    "price": 25,
    "quantity": 2
}
```



GET localhost:3001/info/4

Params Authorization Headers (6) Body Scripts Settings

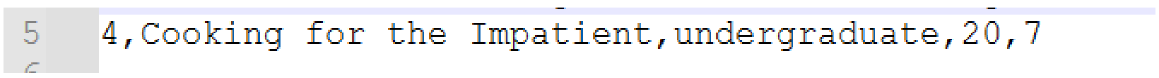
Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "id": 4,
    "title": "Cooking for the Impatient",
    "topic": "undergraduate",
    "price": 20,
    "quantity": 7
}
```

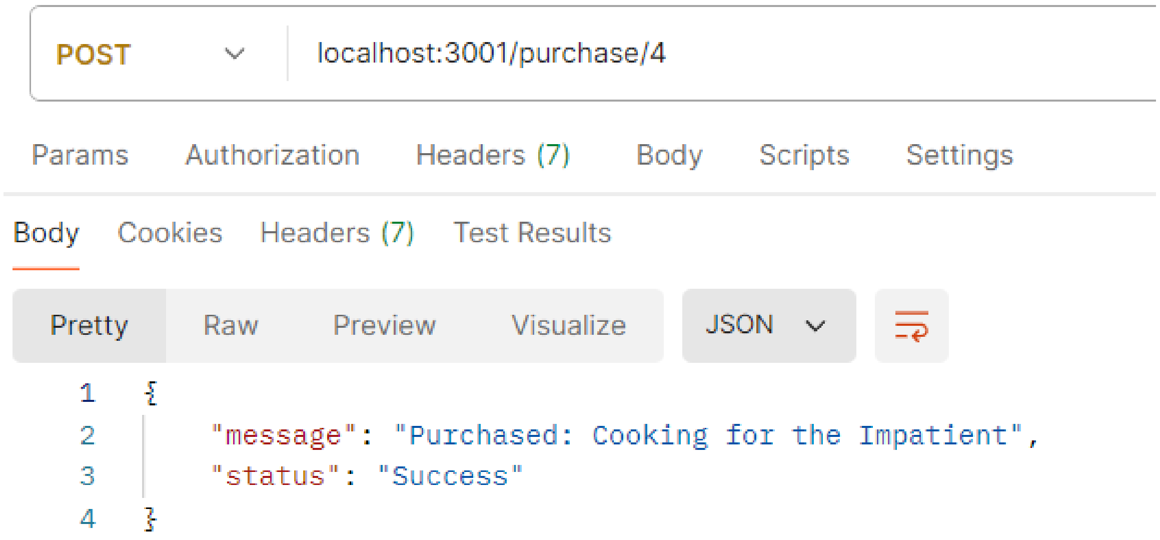
After that we try to make a purchase ,for book with id=4, we have quantity=7, we will purchase and the quantity should decrement by 1

Before:



```
4,Cooking for the Impatient,undergraduate,20,7
```

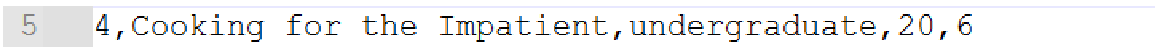
We use post method:



POST localhost:3001/purchase/4

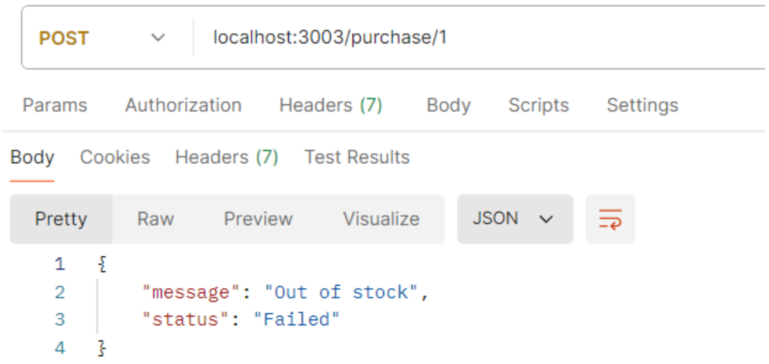
```
{
    "message": "Purchased: Cooking for the Impatient",
    "status": "Success"
}
```

After:



```
4,Cooking for the Impatient,undergraduate,20,6
```

The quantity become 6, also when we try to but when quantity =0



POST localhost:3003/purchase/1

```
{
    "message": "Out of stock",
    "status": "Failed"
}
```