DOS – Project Part 1

Bazar Project

Distributed Operating System

Mai Fahed 11820357

Ola Abdallah 11820341

DOS project part 1:

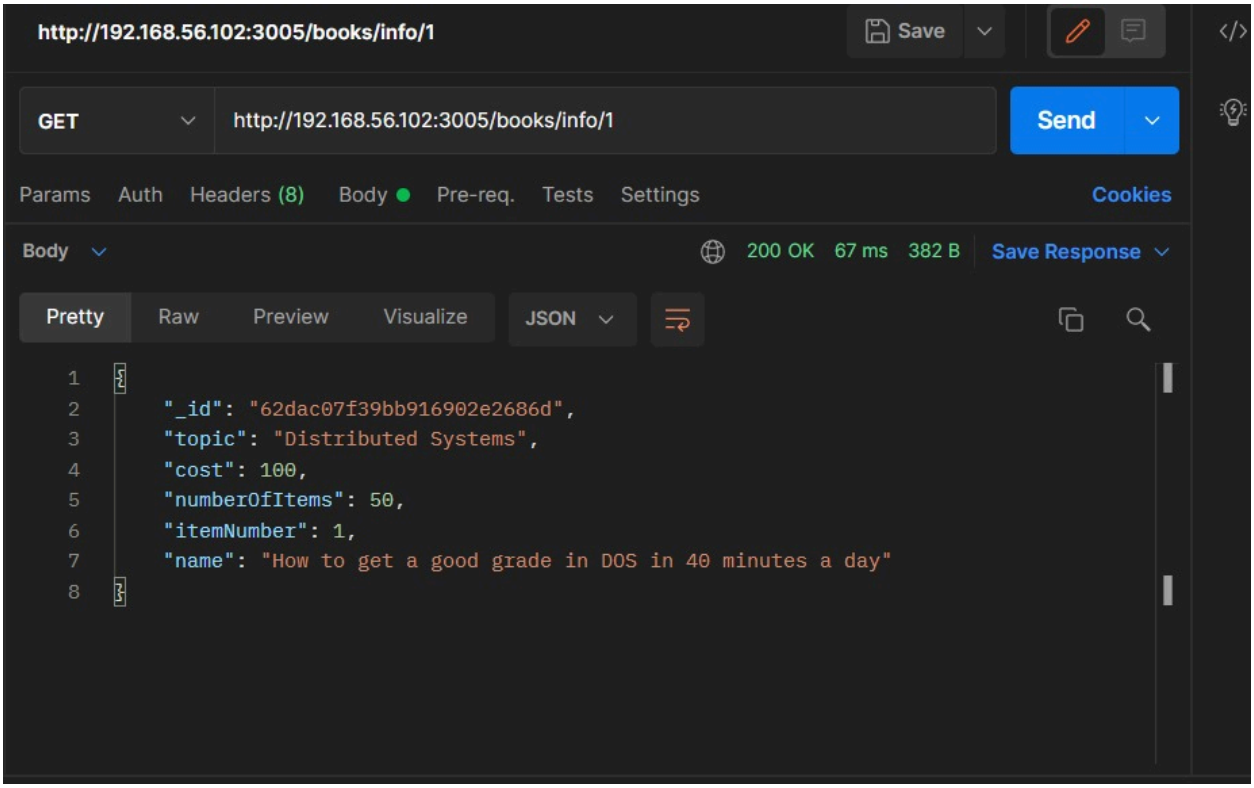
we were asked to implement three servers, catalog server, purchase server and frontend server. Each one of these servers should run on a different virtual machine as separate service, so we used to implement this virtual box machine and we had two virtual machines in addition to the original machine which is the frontend server, the first machine is catalog server with its database and the second machine is purchase server with its database.

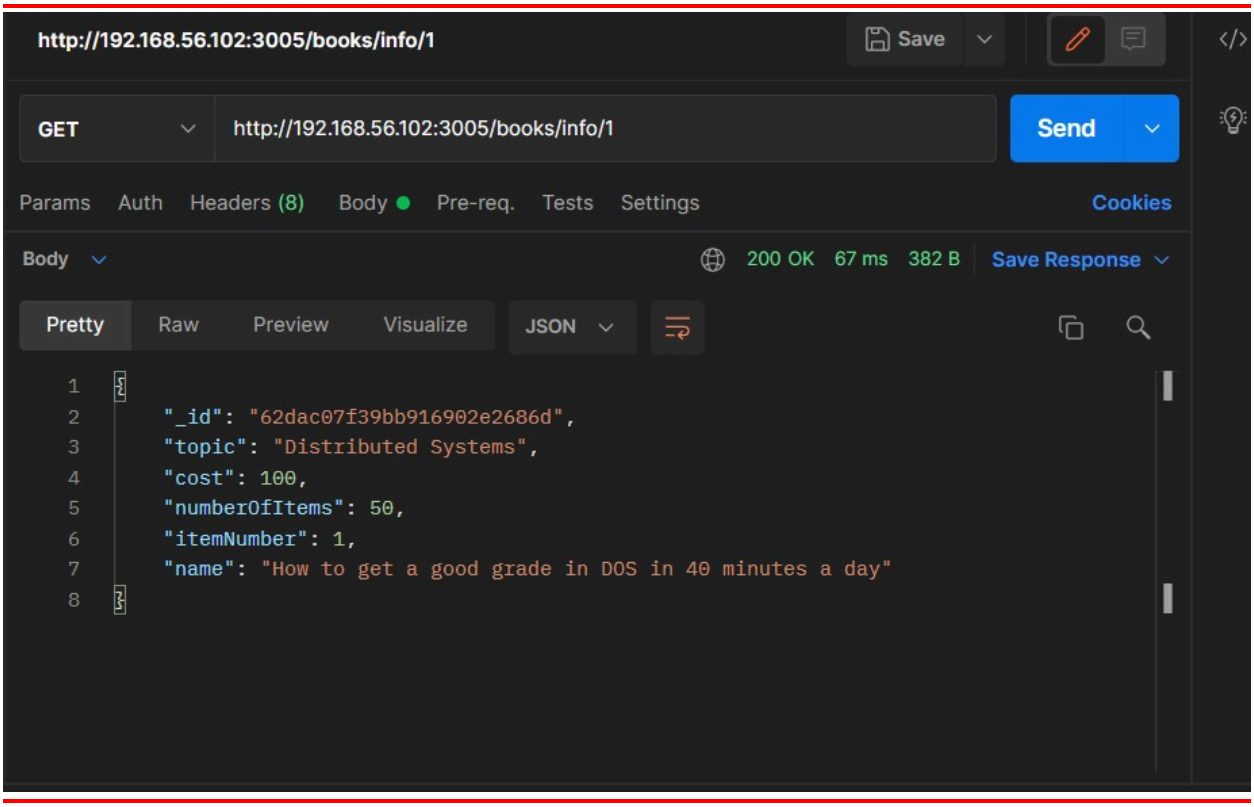
We used MongoDB to store the books, for each book there is a record that includes: itemNumber, name, cost, topic and numberOfItems, and the technologies we used is Node.js, express, Mongodb and virtual machines.

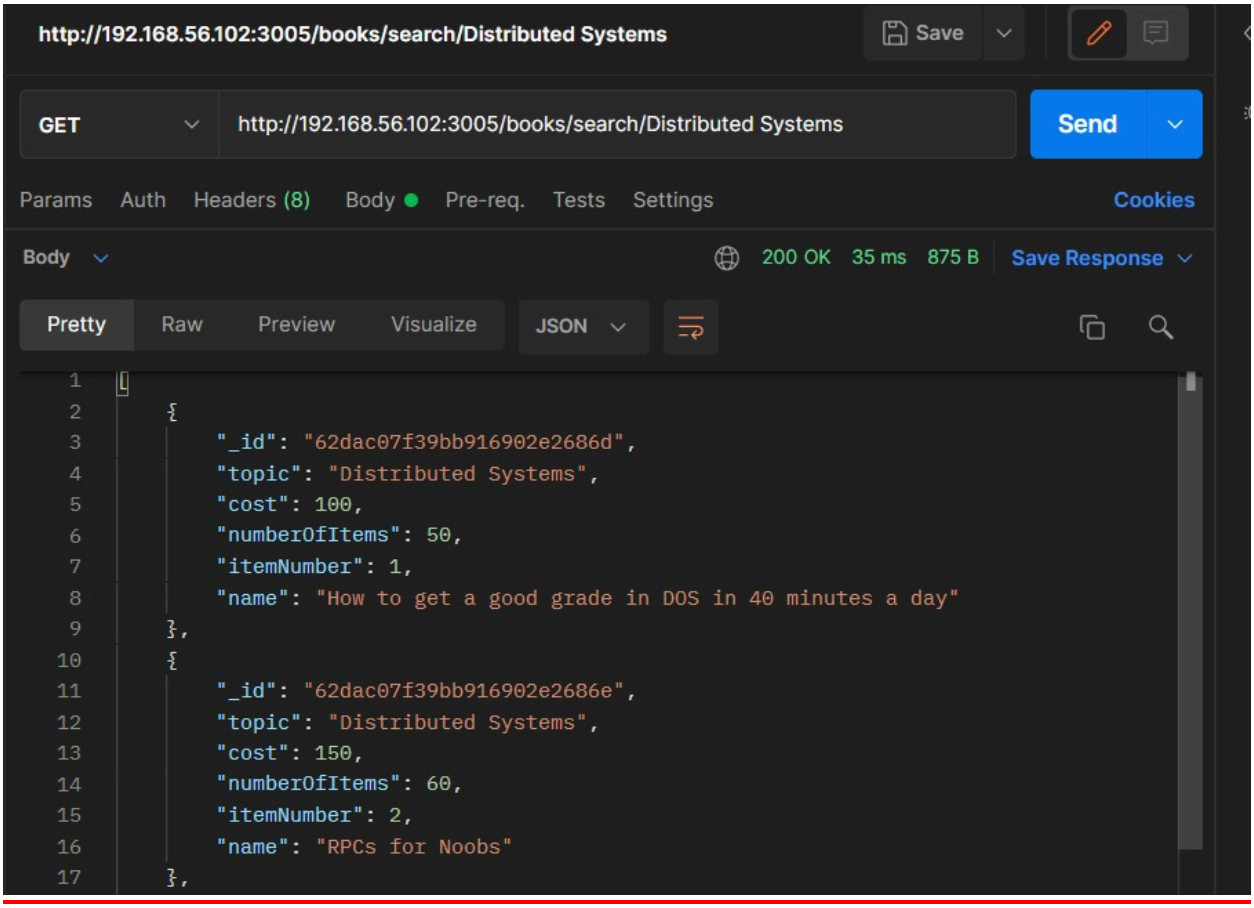
How it works:

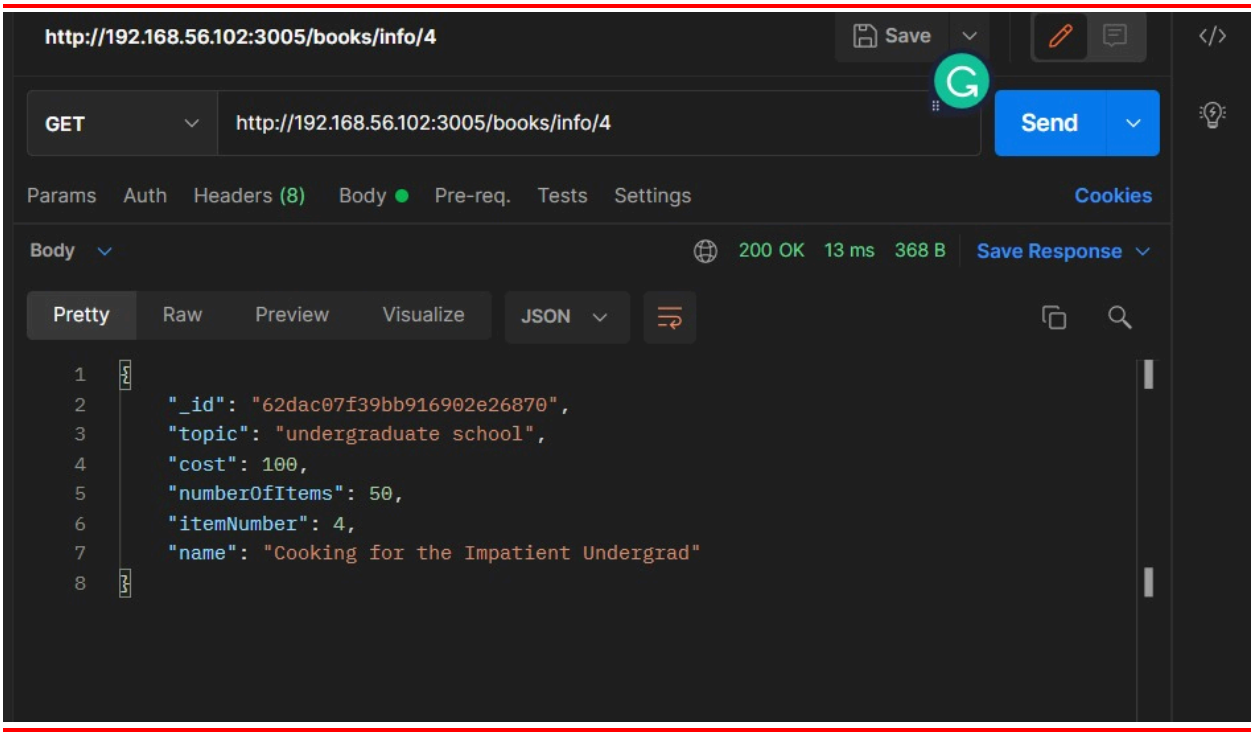
A request is sent by the client which can be the browser, Postman, etc. The client only talks to the frontend server (deployed on my machine). So, the clients will send requests to localhost. When the frontend server receives a request it redirects this request to one of the two servers depending on the request type; catalog server and order server. Catalog server is the only server that is allowed to read or modify on the database and it handles three types of requests: get a book by its id, get books by their topic and modify the cost or number of items for a book. And Order server only handles the purchase requests, it first communicates with the catalog server to check whether this item is available, if not then the operation isn't valid, otherwise, the order server needs to contact the catalog server again to decrement the quantity of that item.

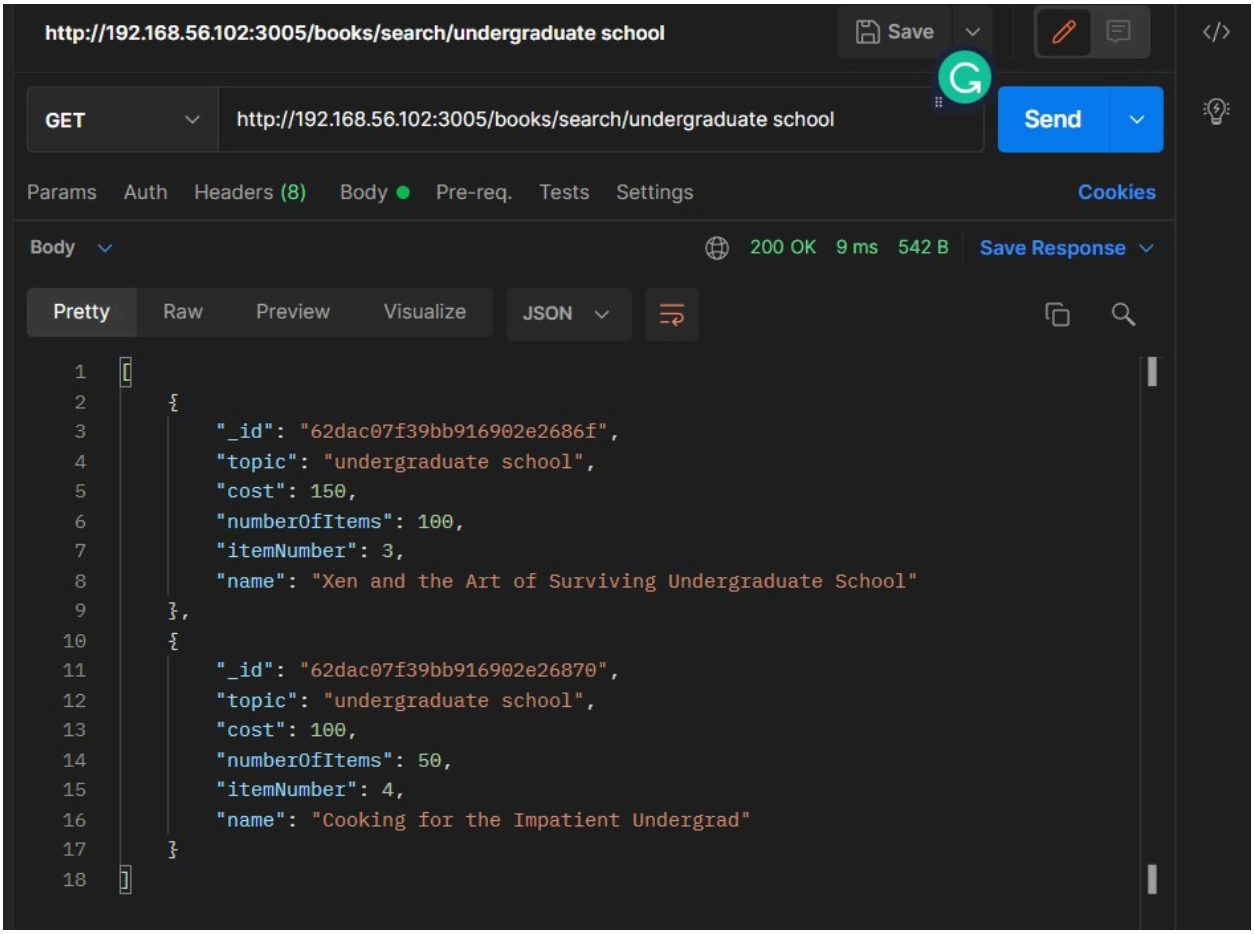
Outputs:

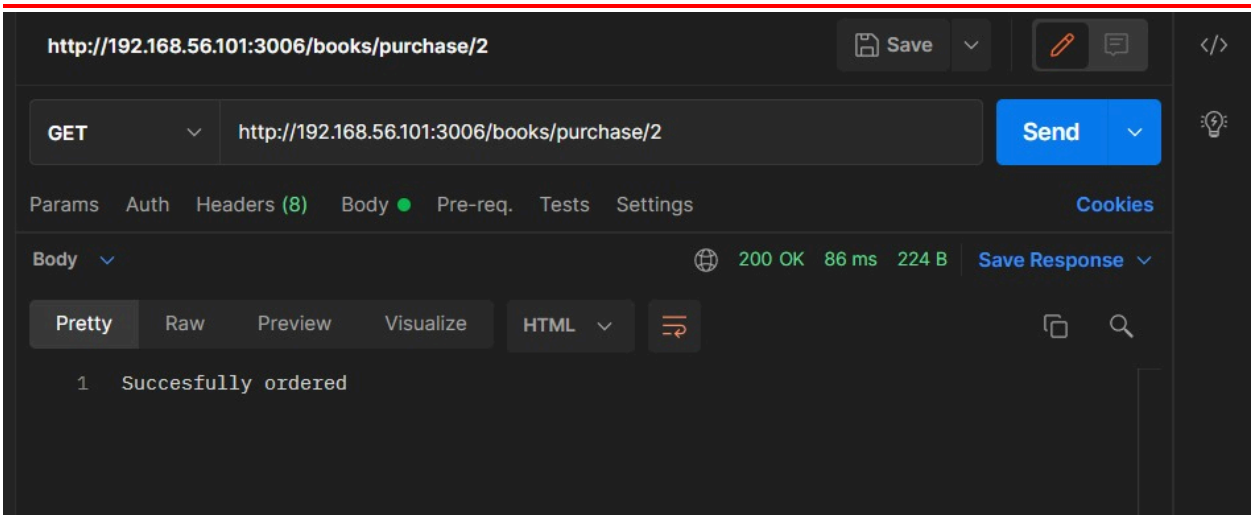

http://192.168.56.102:3005/books/info/1
Save
GET
http://192.168.56.102:3005/books/info/1
Send
Params Auth Headers (8) Body Pre-req. Tests Settings
Cookies
Body
200 OK 67 ms 382 B
Save Response
Pretty Raw Preview Visualize JSON
{
"_id": "62dac07f39bb916902e2686d",
"topic": "Distributed Systems",
"cost": 100,
"numberOfItems": 50,
"itemNumber": 1,
"name": "How to get a good grade in DOS in 40 minutes a day"
}


http://192.168.56.102:3005/books/info/1
Save
GET
http://192.168.56.102:3005/books/info/1
Send
Params Auth Headers (8) Body Pre-req. Tests Settings
Cookies
Body
200 OK 67 ms 382 B
Save Response
Pretty Raw Preview Visualize JSON
{
"_id": "62dac07f39bb916902e2686d",
"topic": "Distributed Systems",
"cost": 100,
"numberOfItems": 50,
"itemNumber": 1,
"name": "How to get a good grade in DOS in 40 minutes a day"
}


http://192.168.56.102:3005/books/search/Distributed Systems
Save
GET
http://192.168.56.102:3005/books/search/Distributed Systems
Send
Params Auth Headers (8) Body Pre-req. Tests Settings
Cookies
Body
200 OK 35 ms 875 B
Save Response
Pretty Raw Preview Visualize JSON
[
{
"_id": "62dac07f39bb916902e2686d",
"topic": "Distributed Systems",
"cost": 100,
"numberOfItems": 50,
"itemNumber": 1,
"name": "How to get a good grade in DOS in 40 minutes a day"
},
{
"_id": "62dac07f39bb916902e2686e",
"topic": "Distributed Systems",
"cost": 150,
"numberOfItems": 60,
"itemNumber": 2,
"name": "RPCs for Noobs"
},

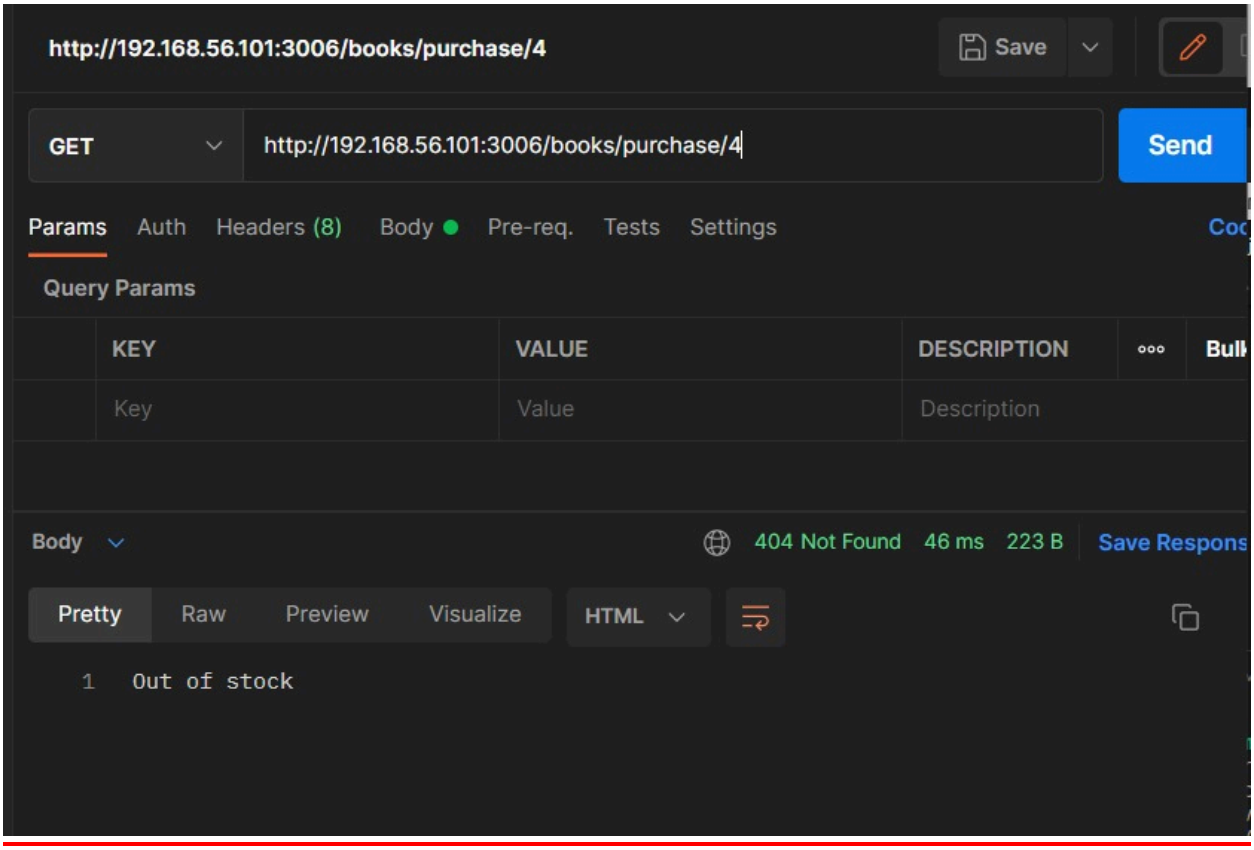

http://192.168.56.102:3005/books/info/4
Save
GET
http://192.168.56.102:3005/books/info/4
Send
Params Auth Headers (8) Body Pre-req. Tests Settings
Cookies
Body
200 OK 13 ms 368 B
Save Response
Pretty Raw Preview Visualize JSON
{
"_id": "62dac07f39bb916902e26870",
"topic": "undergraduate school",
"cost": 100,
"numberOfItems": 50,
"itemNumber": 4,
"name": "Cooking for the Impatient Undergrad"
}


http://192.168.56.102:3005/books/search/undergraduate school
Save
GET
http://192.168.56.102:3005/books/search/undergraduate school
Send
Params Auth Headers (8) Body Pre-req. Tests Settings
Cookies
Body
200 OK 9 ms 542 B
Save Response
Pretty Raw Preview Visualize JSON
[
{
"_id": "62dac07f39bb916902e2686f",
"topic": "undergraduate school",
"cost": 150,
"numberOfItems": 100,
"itemNumber": 3,
"name": "Xen and the Art of Surviving Undergraduate School"
},
{
"_id": "62dac07f39bb916902e26870",
"topic": "undergraduate school",
"cost": 100,
"numberOfItems": 50,
"itemNumber": 4,
"name": "Cooking for the Impatient Undergrad"
}
]

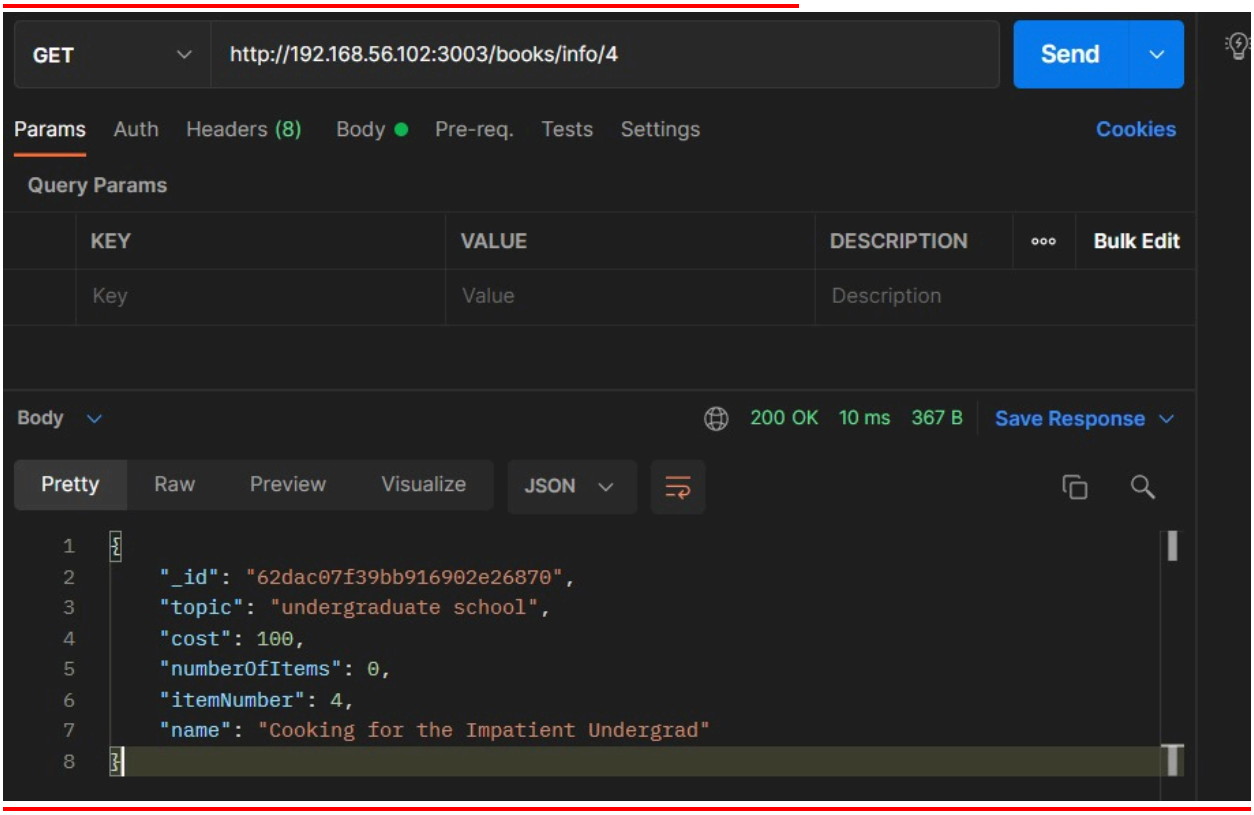

http://192.168.56.101:3006/books/purchase/2
Save
GET
http://192.168.56.101:3006/books/purchase/2
Send
Params Auth Headers (8) Body Pre-req. Tests Settings
Cookies
Body
200 OK 86 ms 224 B
Save Response
Pretty Raw Preview Visualize HTML
Succesfully ordered

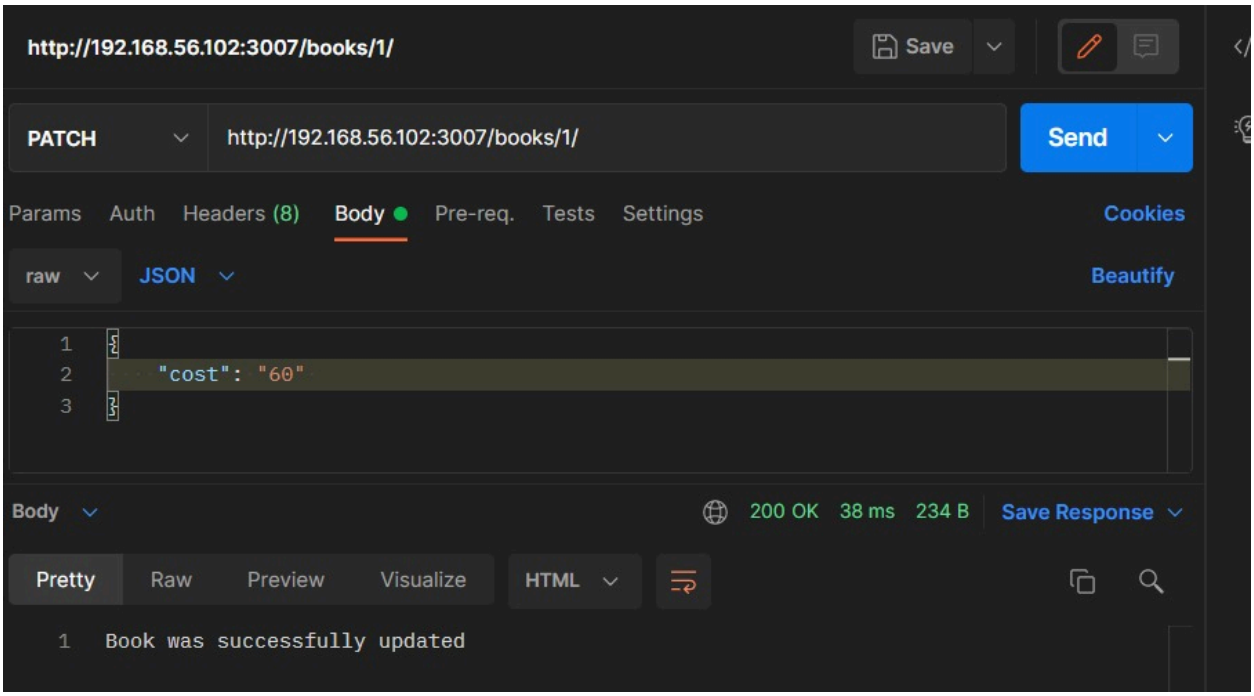
```
server is listening
connected
received a request, host:192.168.56.101
path:/books/purchase/2
method:GET
**********************************************************
**
{ msg: 'available', count: 56 }
Succesfully ordered
```

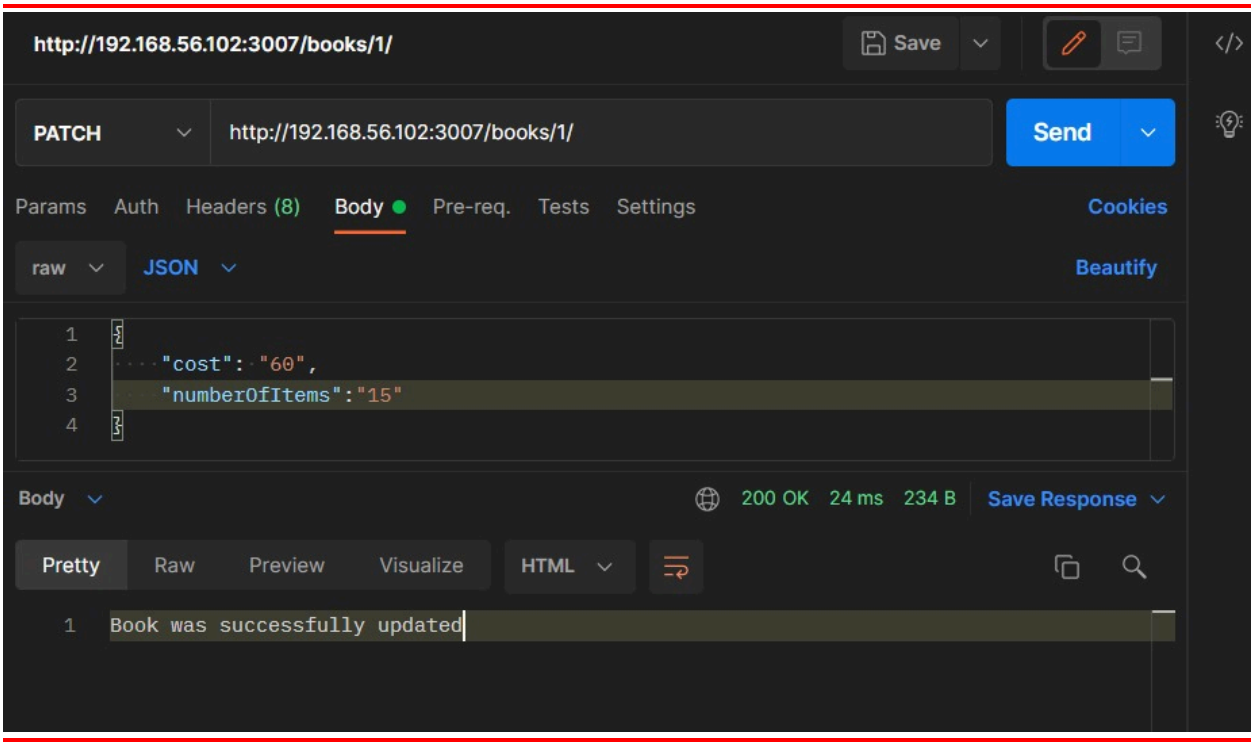
```
{ msg: 'available', count: 56 }
Succesfully ordered
received a request, host:192.168.56.101
path:/books/purchase/2
method:GET
**********************************************************
**
{ msg: 'available', count: 55 }
Succesfully ordered
```

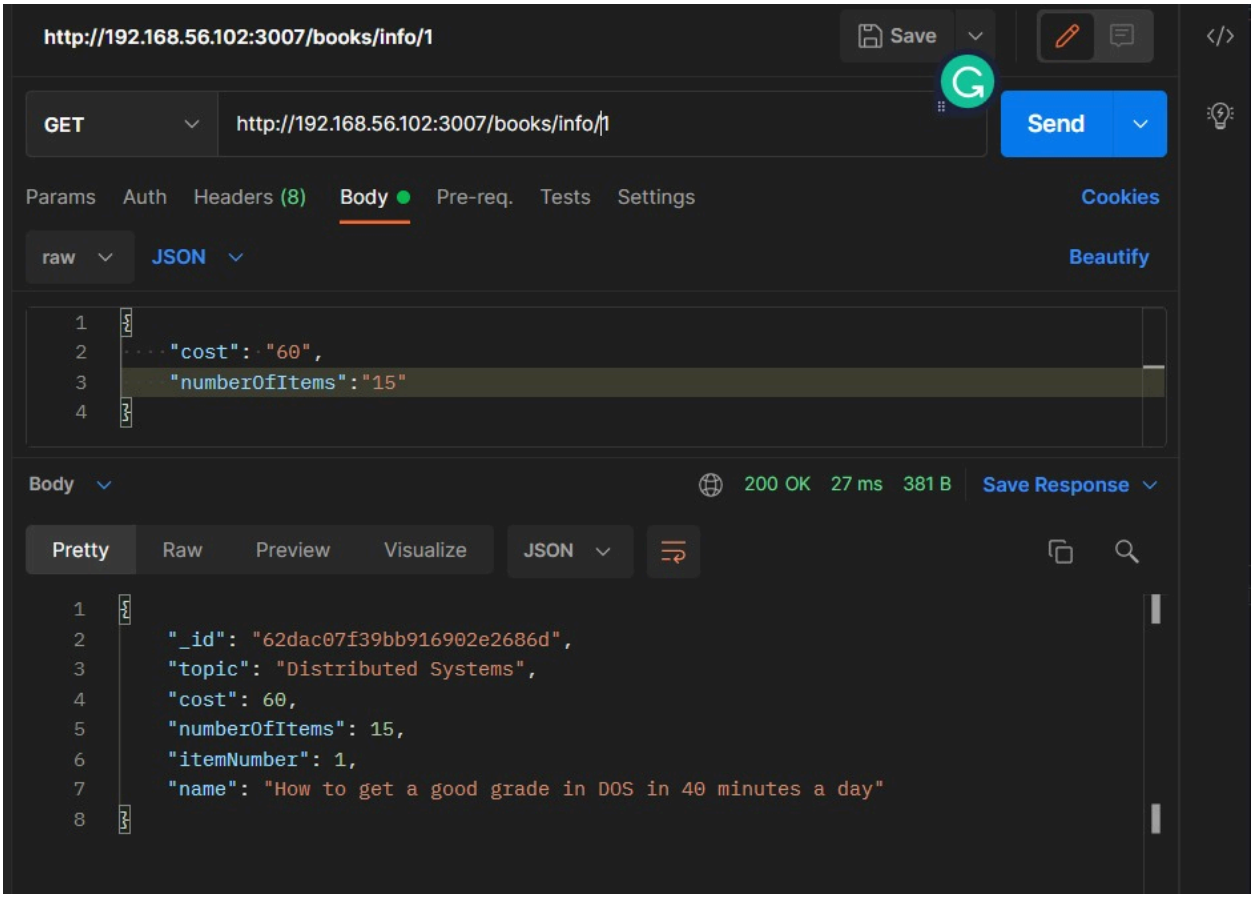


```
{ msg: 'available', count: 1 }
Succesfully ordered
received a request, host:192.168.56.101
path:/books/purchase/4
method:GET
**********************************************************
```




http://192.168.56.102:3007/books/1/
Save
PATCH
http://192.168.56.102:3007/books/1/
Send
Params Auth Headers (8) Body Pre-req. Tests Settings
Cookies
raw JSON
Beautify
1 {
2 "cost": "60"
3 }
Body
200 OK 38 ms 234 B
Save Response
Pretty Raw Preview Visualize HTML
1 Book was successfully updated


http://192.168.56.102:3007/books/1/
Save
PATCH
http://192.168.56.102:3007/books/1/
Send
Params Auth Headers (8) Body Pre-req. Tests Settings
Cookies
raw JSON
Beautify
1 {
2 "cost": "60",
3 "numberOfItems":"15"
4 }
Body
200 OK 24 ms 234 B
Save Response
Pretty Raw Preview Visualize HTML
1 Book was successfully updated



Trade-offs:

The client knows nothing about what happens behind the scene and only communicates with one server which is the frontend server. However, there may be an overhead due to the communication between servers.

Improvments and extensions:

If we wanted to make the system larger, there could be a level of authentication on the user level to protect the data, also there could be an admin who can add new books to the database.

Cases which don’t work corectly:

We implemented all the required cases.

How to run the program:

You need to have a tool like Postman to send requests. On your machine you have to install Node.js. On catalog server, you need to install MongoDB and Node.js, and fill the database with books with the attributes, you also may need to give this virtual machine IP address. Then you can run catalog server code. On the second VM (order server), you need to install Node.js and Mongodb, give this the machine the IP, then you can run the order server code. Now, you can start sending requests to localhost.

Catalog Server: https://github.com/OlaAbdallah1/Catalog-Server

Order Server: https://github.com/OlaAbdallah1/Orders-Server

Frontend Server: https://github.com/OlaAbdallah1/Frontend-Server