Bazar.com Project Report

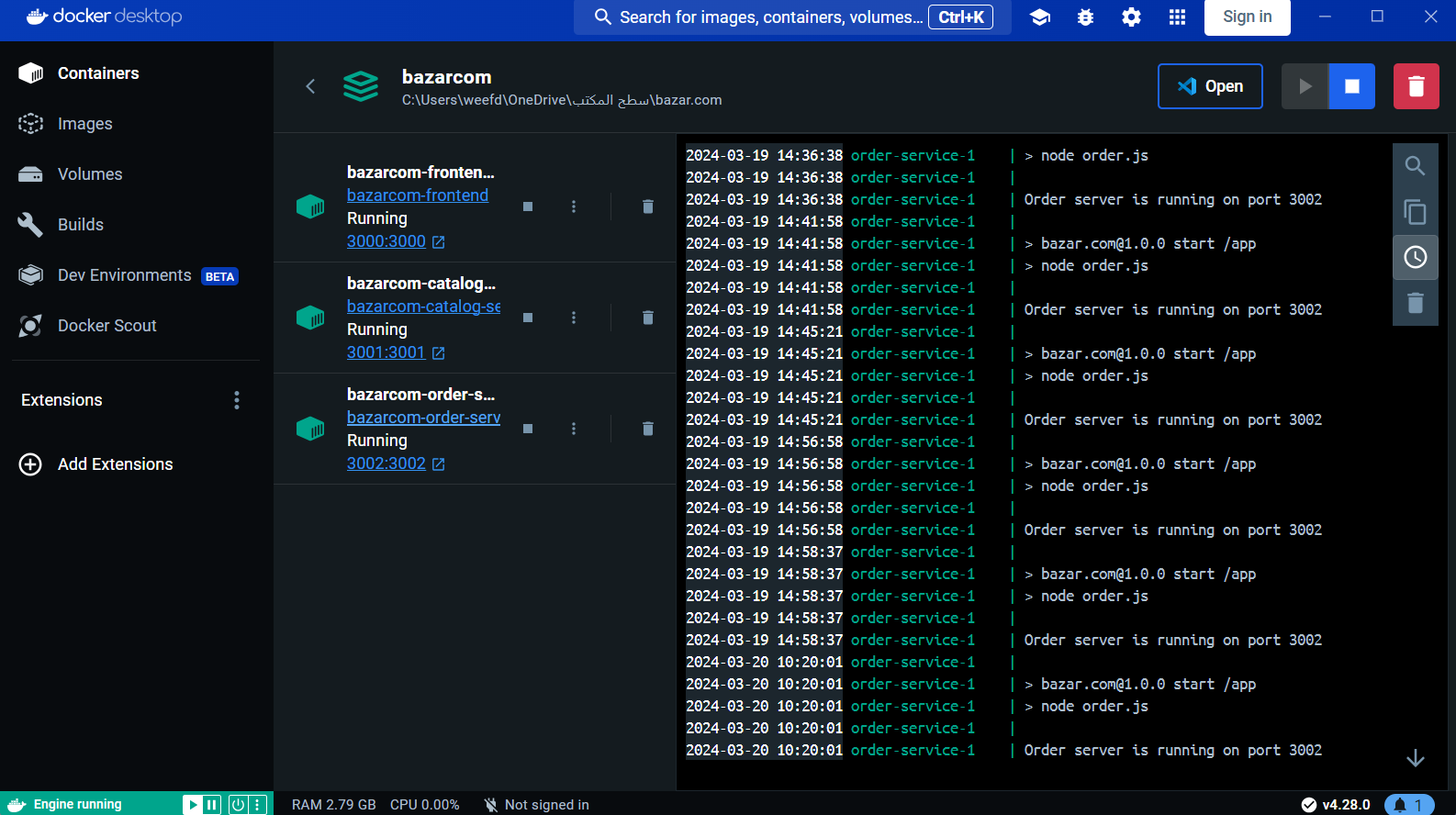
So first of all Lets talk about Docker…

I create image called bazarcom with 3 containers :

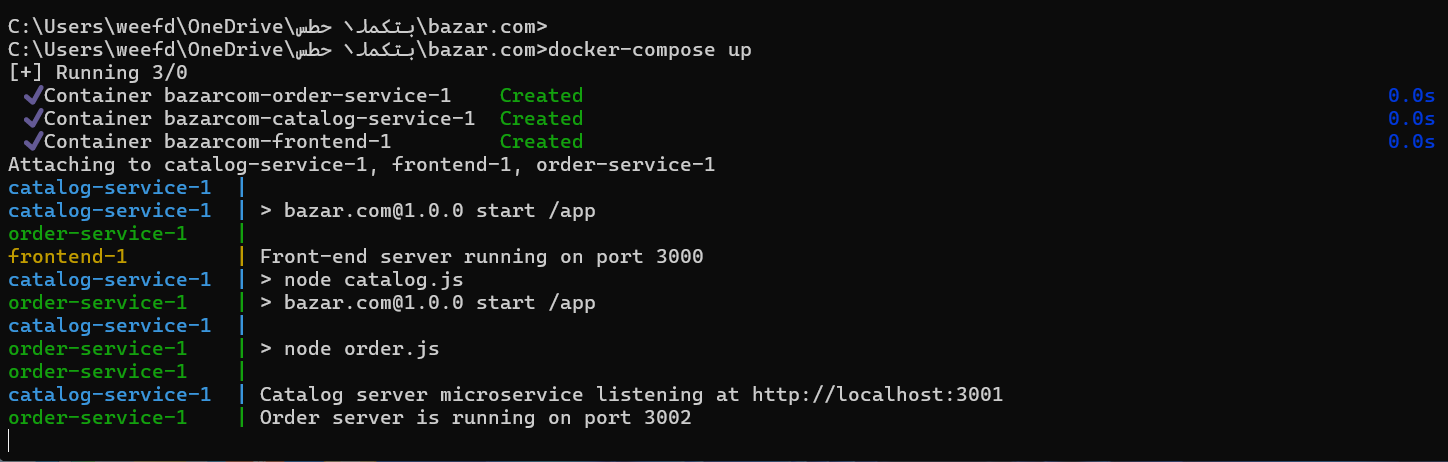
1.bazarcom-catalog-service

2.bazarcom-frontend

3.bazarcom-order-sevice



The image above shows that we have a container of image bazarcom and a container for frontend , catalog and order .



This photo show how to run the program.

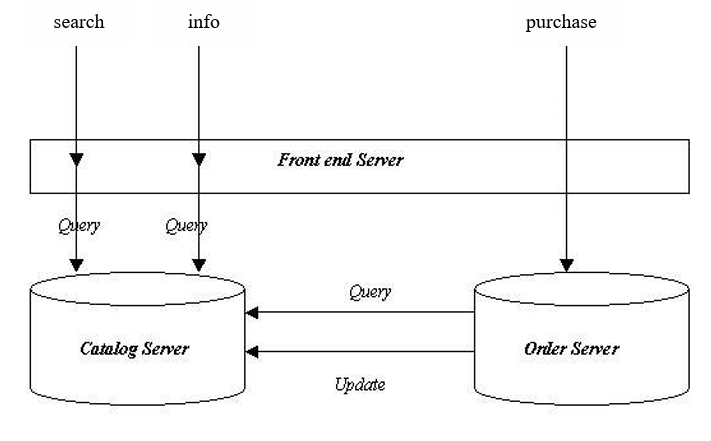
To run the program we use the command

-docker-compose build on the path of bazar.com file as above photo shows.

And then we write the following command to run it:

-docker-compose up .

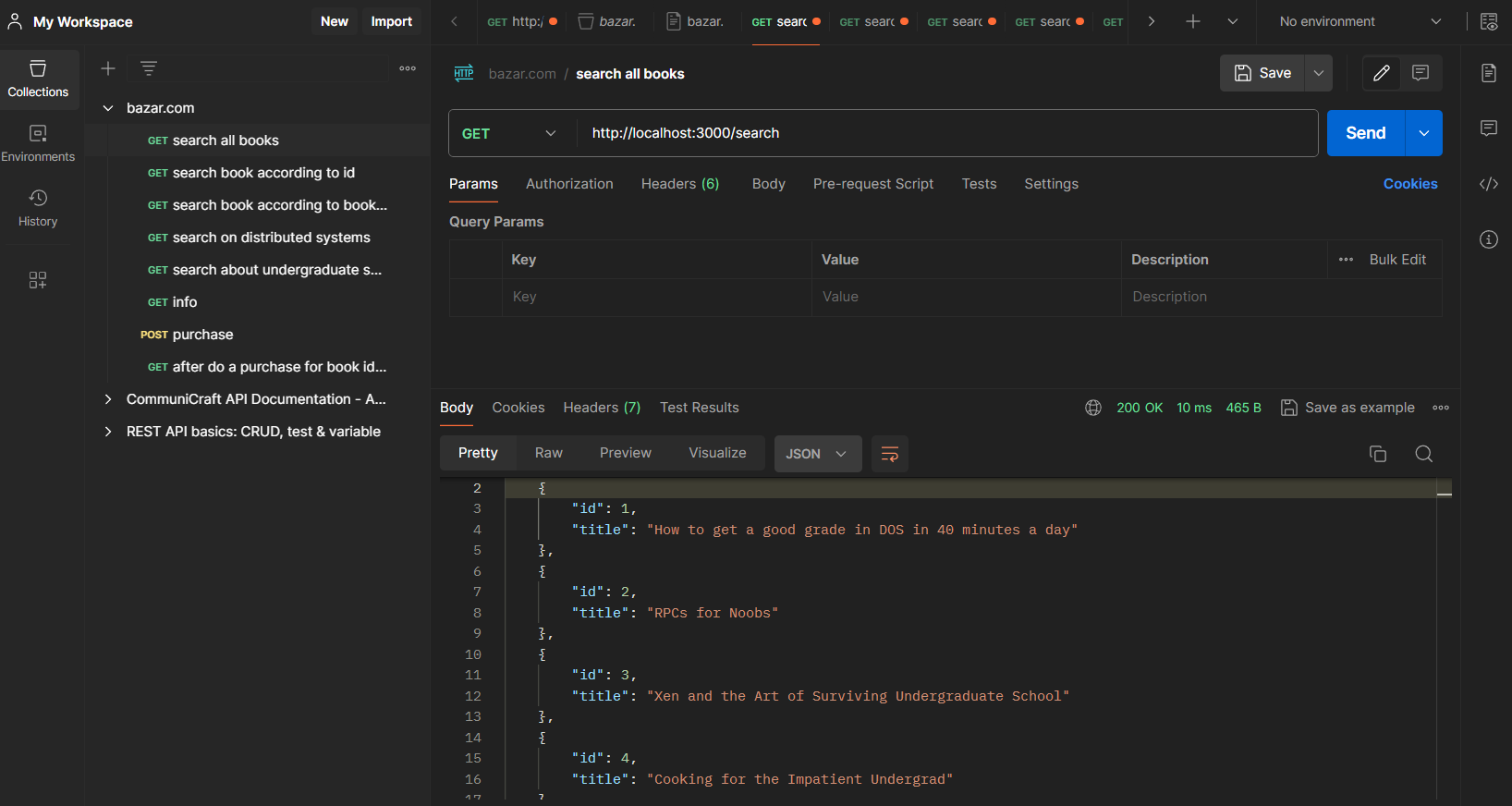
To Describe the program :



This photo give us a great Describe for the program so lets explain…

We have 3 servers each server is a container in docker the servers communication between each other as in the photo above It is determined by yml file .

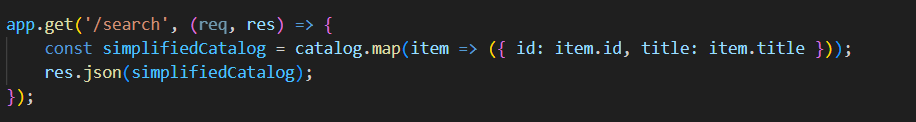
First of all a request for the frontend server to search about all books you will get the following result:



So what happened here:

1.If you see our code the frontend server has a port 3000 so the request goes to frontend as in the code then frontend will forward the request to the catalog server .

2.catalog server take the request Specifies the link that matches the link in the request ooh it find the link matches the request its here :



So it returns all books (catalogs) to frontend server and then we will get the above result.

* You can see more details in input output file in Git hub.

-Ok now lets talk about purchase :

The request will take from frontend server the front end will pass it to order server .. then order server will send a get request to catalog server to see if the book exist or not.. if not exist it will return not exist .. if exist it will do an update request (put) to the data of the same id by decreasing the quantity by one.

-design tradeoffs:

* Micro services allow for greater flexibility, scalability, and independent development of components. However, they introduce complexities in deployment, testing, and monitoring.
* Monolithic architectures simplify deployment and testing but may become harder to maintain and scale as the application grows.
* The program's performance, dependability, and usefulness can all be enhanced. A more scalable database system should be used, along with caching techniques to improve response times, authentication and authorization for increased security, and stringent data validation to ward off vulnerabilities.

- Using fault tolerance strategies, asynchronous processing, and container orchestration can also boost scalability and resilience. Enhancements to API versioning, logging, and monitoring can ensure backward compatibility and provide more meaningful data regarding system performance. Implementing load balancing, integration testing, and following containerisation best practices will further improve the system's efficacy and dependability. All things considered, these enhancements and improvements are required to have a reliable and efficient programme that may change to meet the needs and preferences of users.