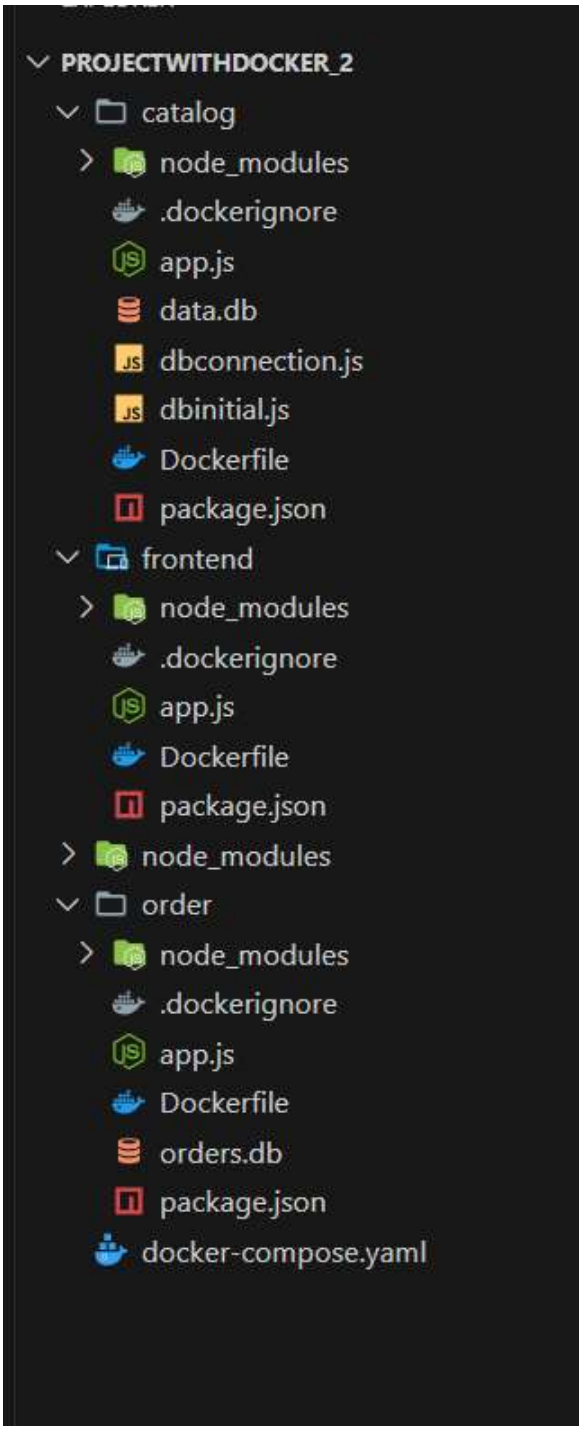
Micro-webservice Project

We made 3 microservices: catalog server, order or purchase server and frontend server with this structure shown below:



1- Catalog Server

This server serves requests that related to Books information and crud operation about it.

2- Order Server

This server serves requests that related to anything about ordering and purchasing

3- Frontend Server

That is the server that interact with clients directly and communicate with other servers to serve client requests.

We used “Sqlite” package to store the books and orders information and “axios” package to interact with other servers.

Frontend server has these functionalities:

1- Search for books under specific topic
`URL/search/{topic}`

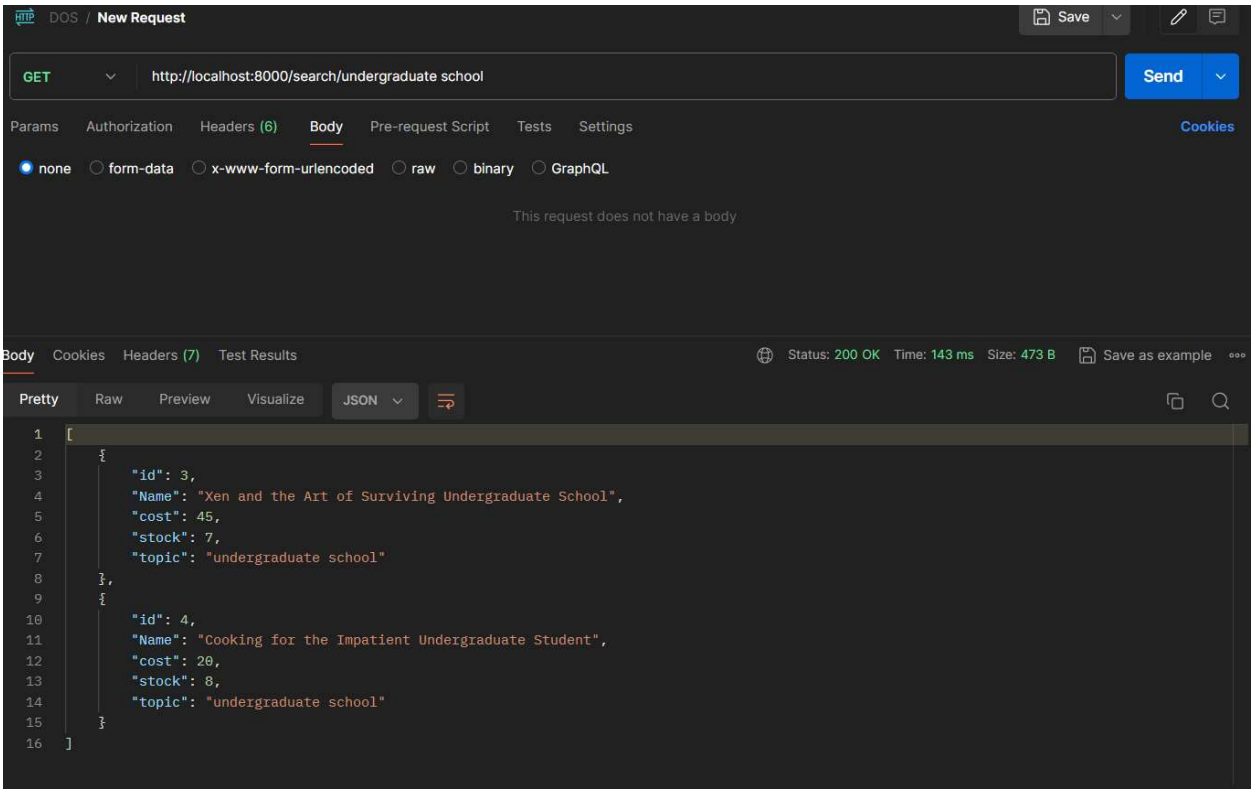
2- Get information of specific book
`URL/info/{bookId}`

3- Purchase a book
`URL/purchase/{bookId}`

API Testing

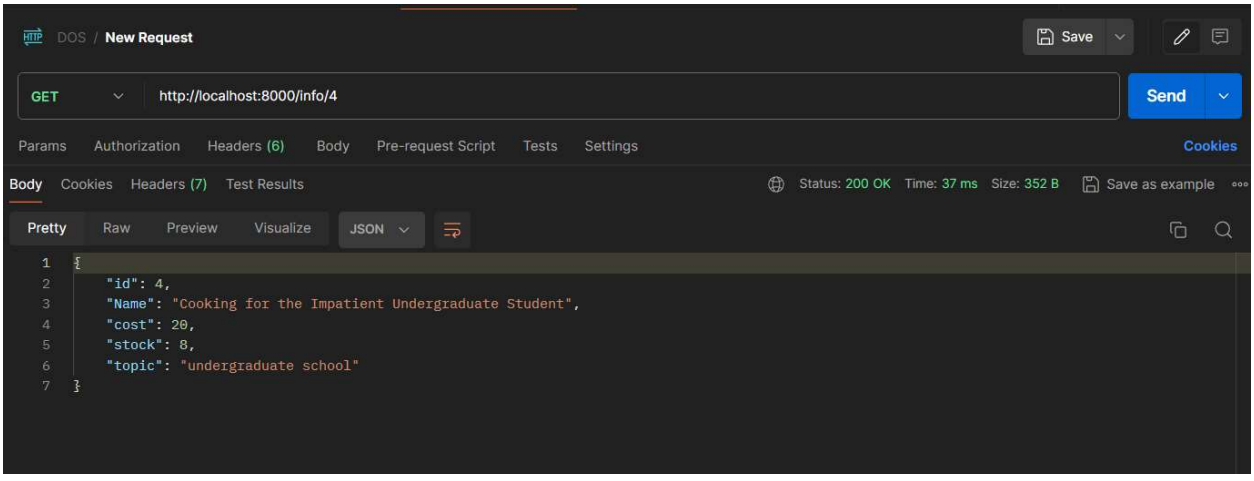
1- Search for specific topic

I will search for books that is under topic “undergraduate school”

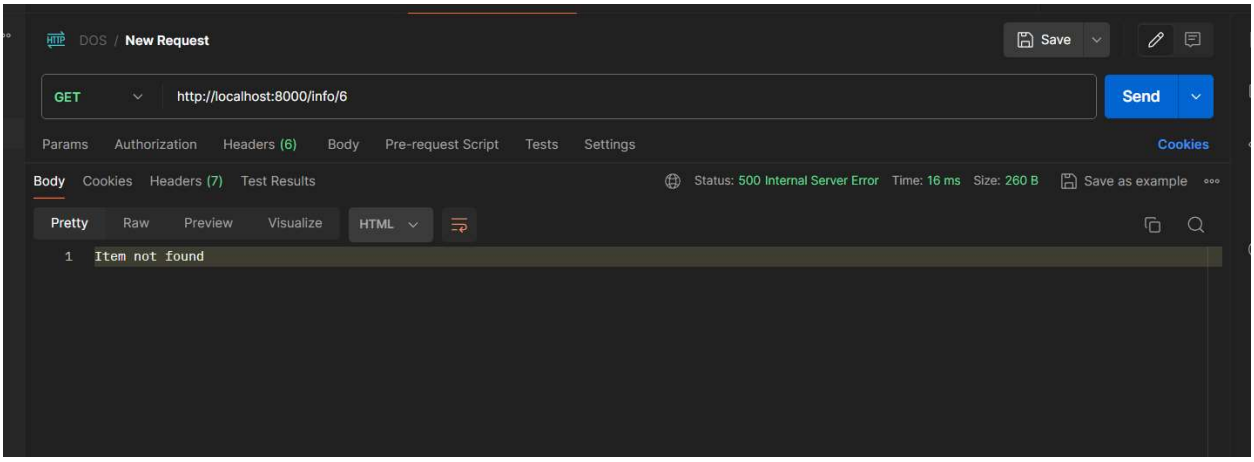


2- GET a book information

a. Let’s get details about book with id “4”

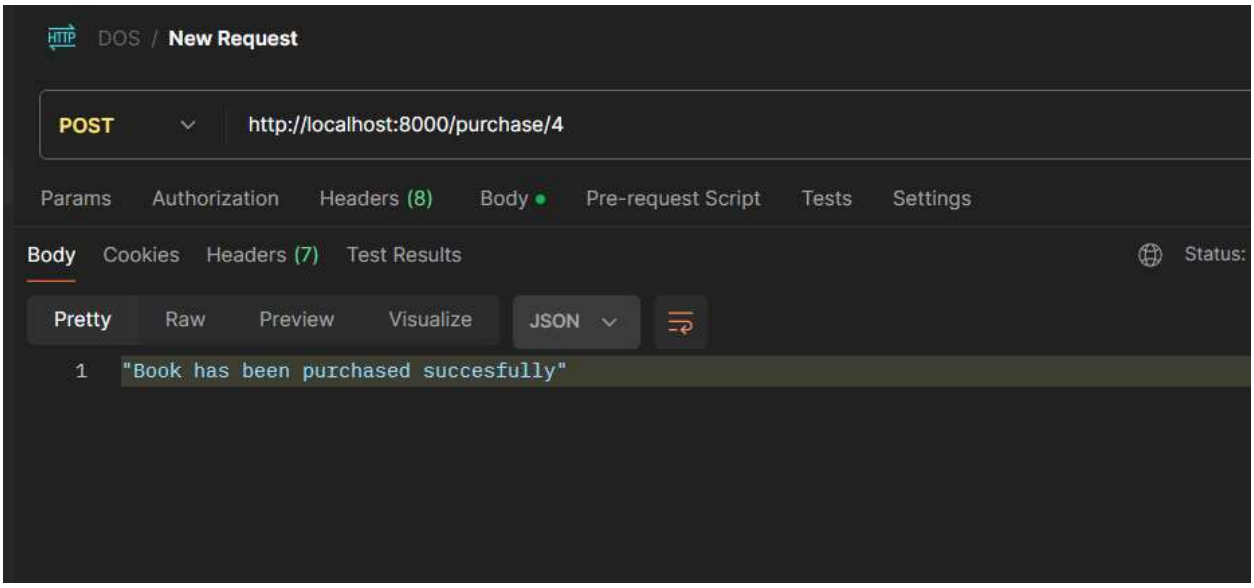


b. And if we request a book that don’t exists that will happen:



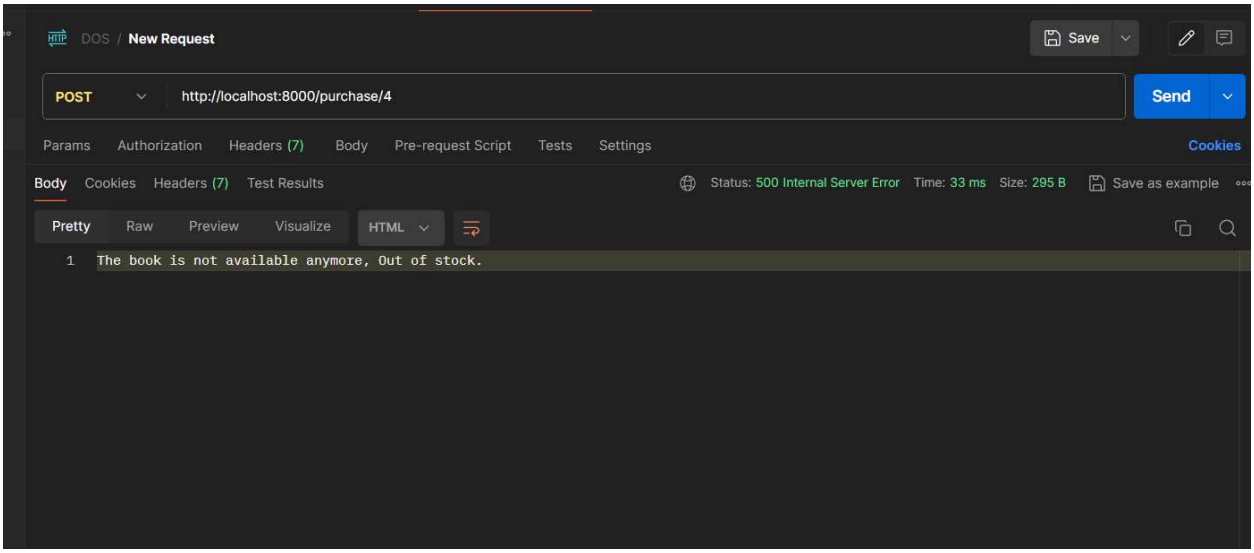
3- Purchase a book

a. Now we will buy a copy of book with id 4



As shown the book has purchased successfully…

b. If we trying to purchase book that out of stock that will happen:



Now there is some cases that will make the system fail.

1- Purchase and book delete conflict:
As I said before, the catalog server has crud operations about books information, now when the client purchases a book The order server will GET the book information, and make the order, after that it will update the book stock by telling the catalog server to decrease the book stock…

now the **conflict** will happen if the order server **GETs** the information and the catalog server **deletes** that book before order server **updates** the stock

2- More than client purchasing same book conflict:
   This case will happen if more than client purchasing a specific book at the same time and the book stock is 1, this caused due to order system, because order **GETs the information** then **make the order** then **update the book stock**