Online book store – DOS project

Masa alsayyed

Meera saymeh

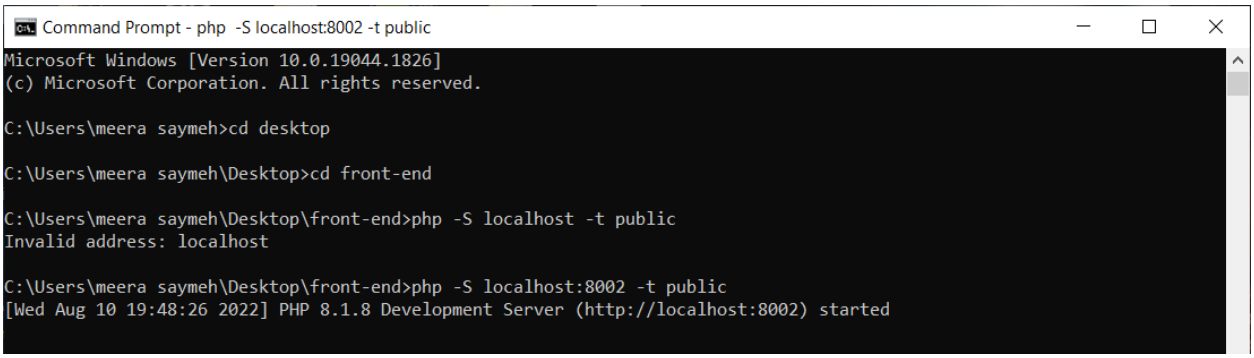
To build our online book store we used Laravel php, Xamp and mySQL database ,we build three micro web services one for the front-end where the user can search for a book, look for information about a book and purchase books, and two services in the server side one for the orders and the other for the books catalog.

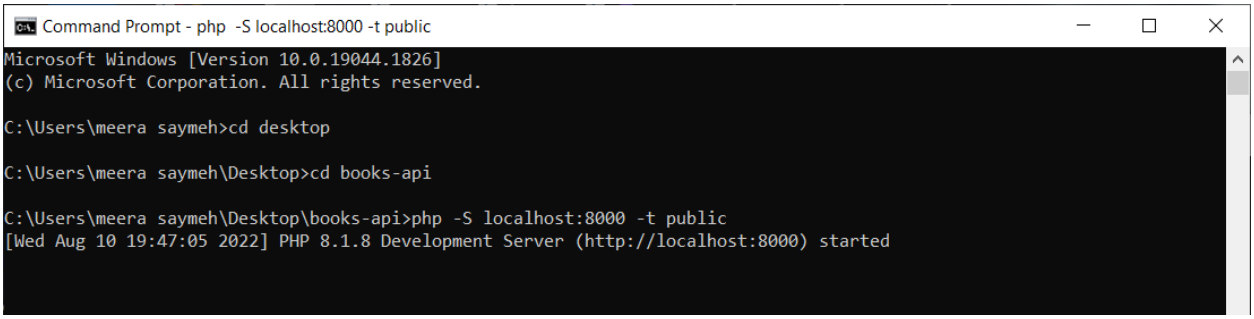
We faced a problem trying to run the services on different virtual machines and finally decided to work using the localhost and give each service a different port.

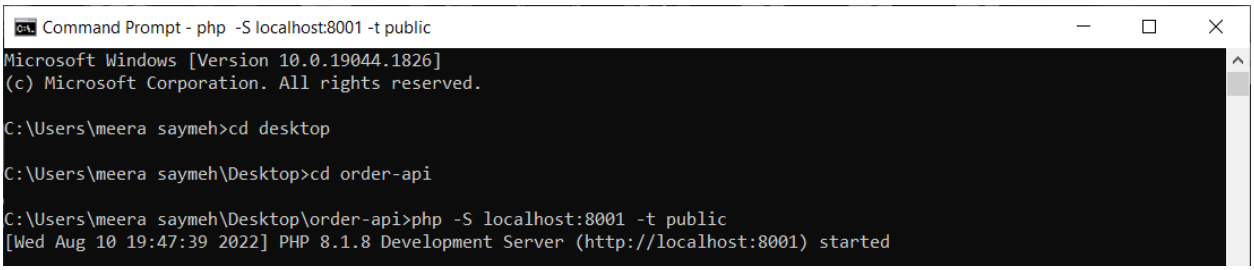
Front-end -> localhost:8002

Order -> localhost:8001

Catalog -> localhost8000







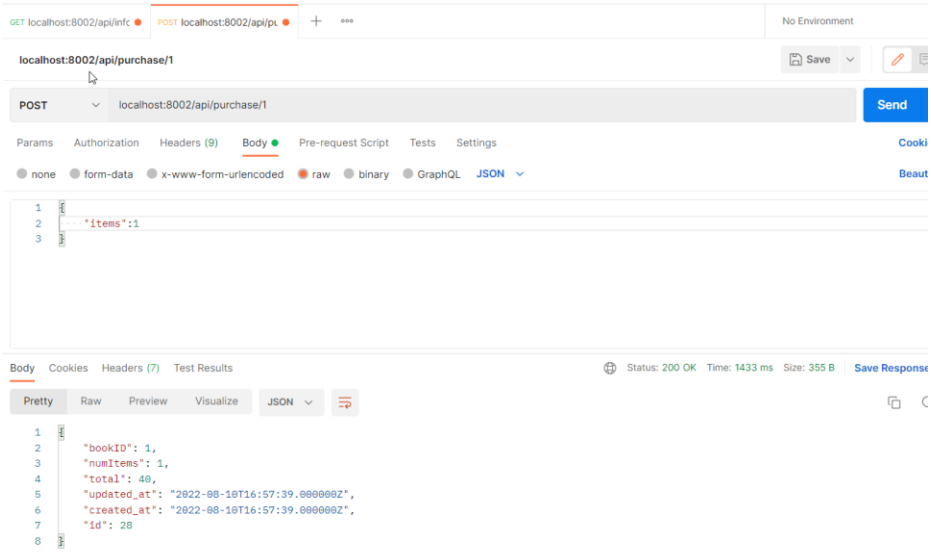
Purchase :

When sending a purchase request from the front-end the user have to send the id for the book in the URL request and the number of books (items) in the Jason , the request will be sent to the orderIP.

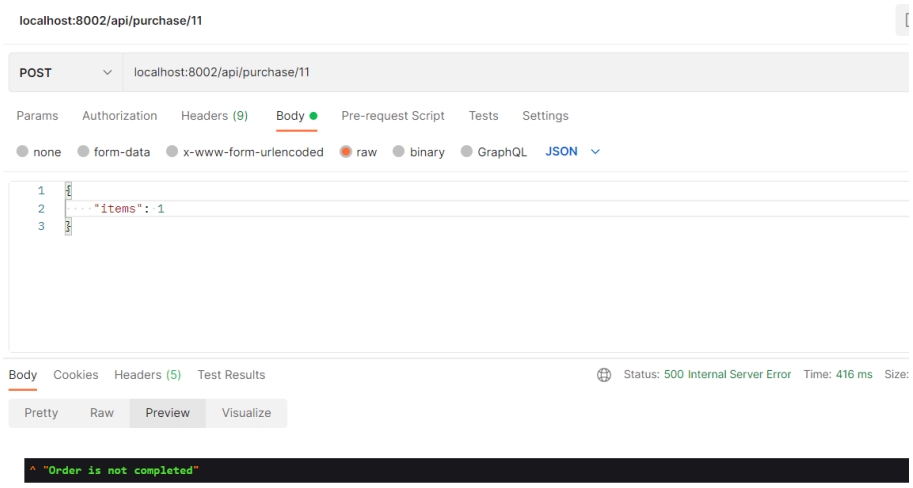
Then in the orderIP if the request was valid, we'll get the info about the pouched book from the catalogIP and send it back as a Jason, the user will see the information about his purchase and the total cost.

in the database the order will be added to the order table, and the number of items for the book will get decreased in the book table. If the book does not exist or if it's out of stock the purchase won't get completed.

Example:



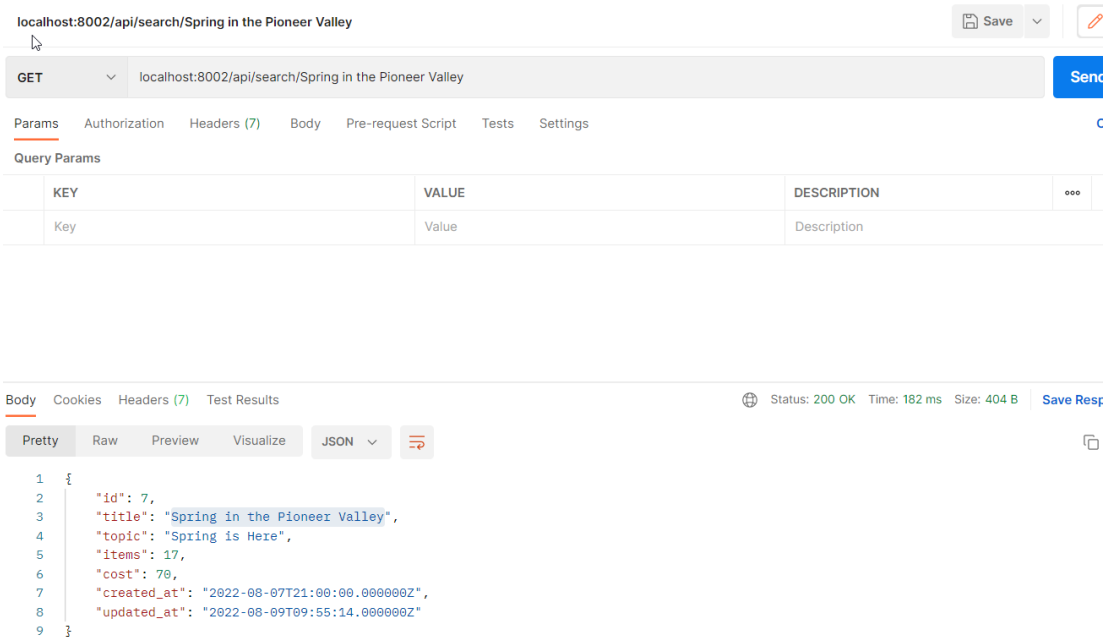
In case you made a purchase for unexacting book -> "order is not completed" massage will appear



Search:

When sending a search request we need to send the book name in the URL, the request will go the catalogIP, we'll get the info about the book and send it back as a Jason .

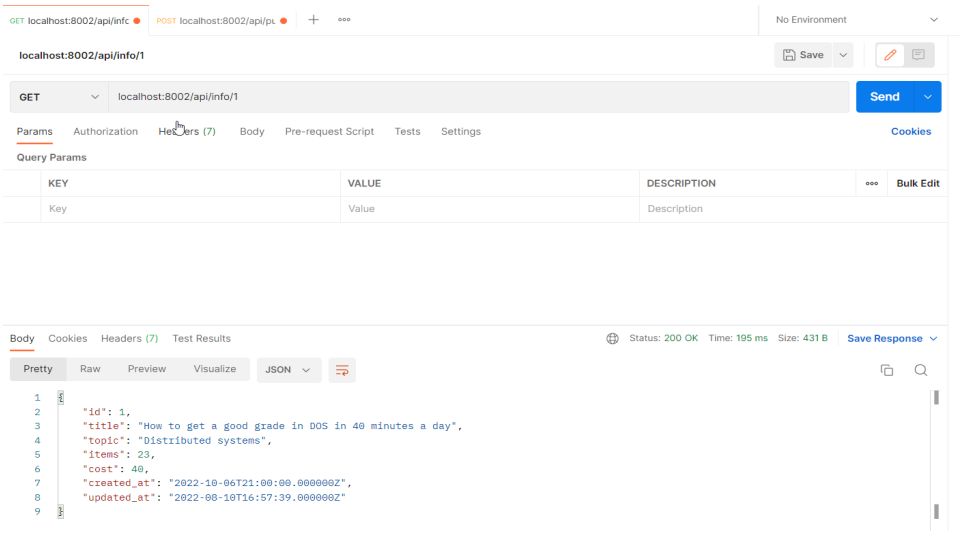
Example:



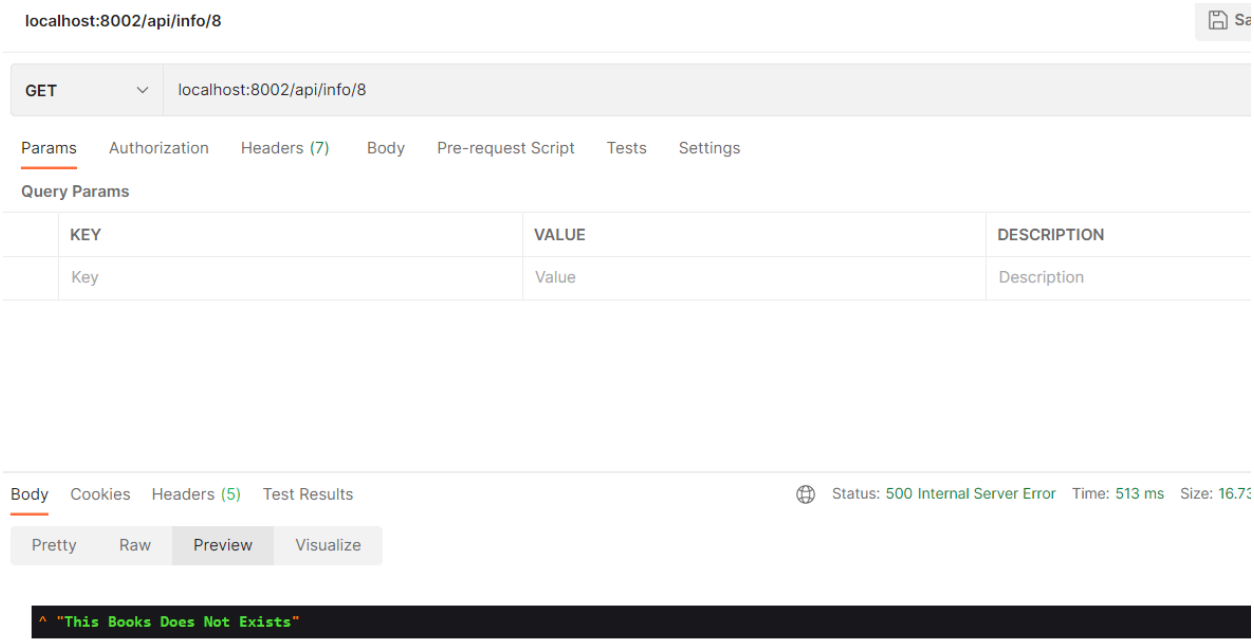
Info:

When sending an info request from the from-end we need to send the book id in the URL, it will go to the catalogIP as in the search function and send back the results as Jason.

Example:



And If the book is not in store -> “this book does not exists” massage will appear



In the second part of the project we were asked to made replicas for the server side. the catalog, the order and the database. And to guarantee consistence between the replicas.

We added another catalog and order server and we used round-robin to choose which replica the request will go to by sitting a flag that will change value each time a request will get sent.

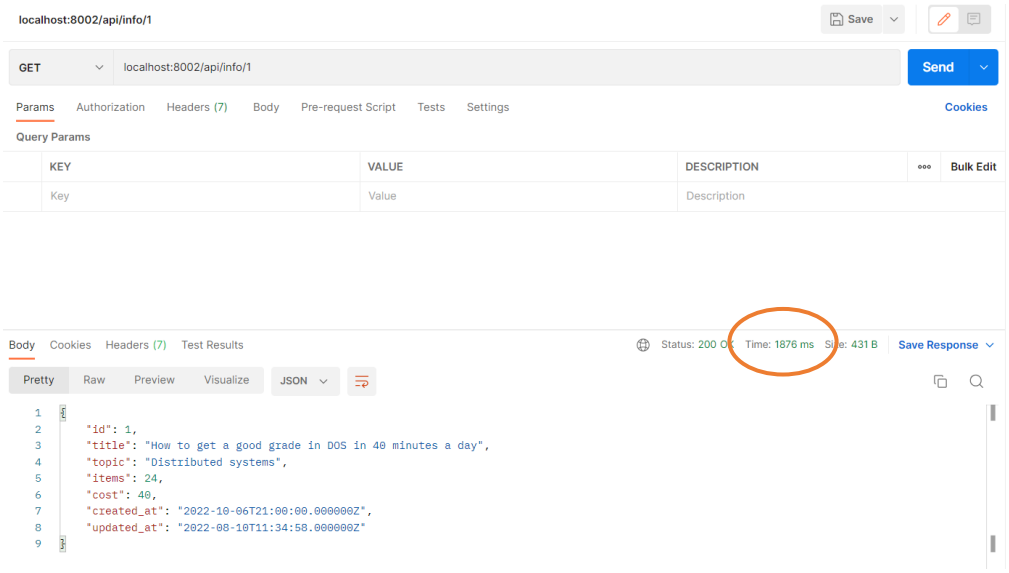
Also we were asked to have a cache system for the request to enhance the performance, we did our research on Laravel and there’s a build in cache system that can be use in different ways, the default way if the file, a file will be created for each server (catalog, order) and this way is a per-request cache.

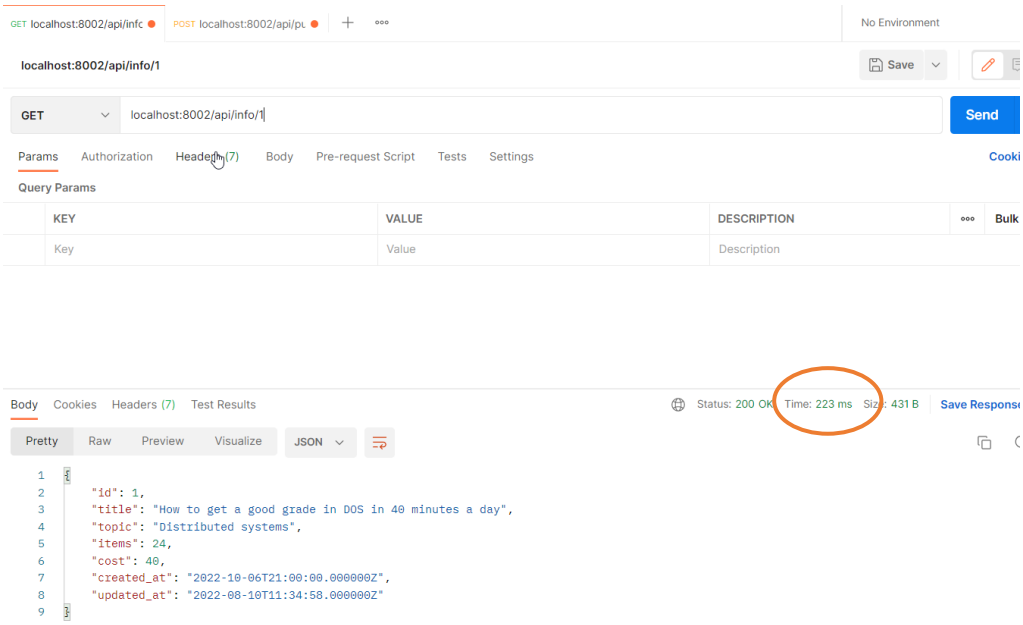
The other way is the array , an array will be created for out project and it will cache the requests, we found this way has a slightly better performance than the file one. Also there is popular caching backends that larval supports like Memcached and DynamoDB.

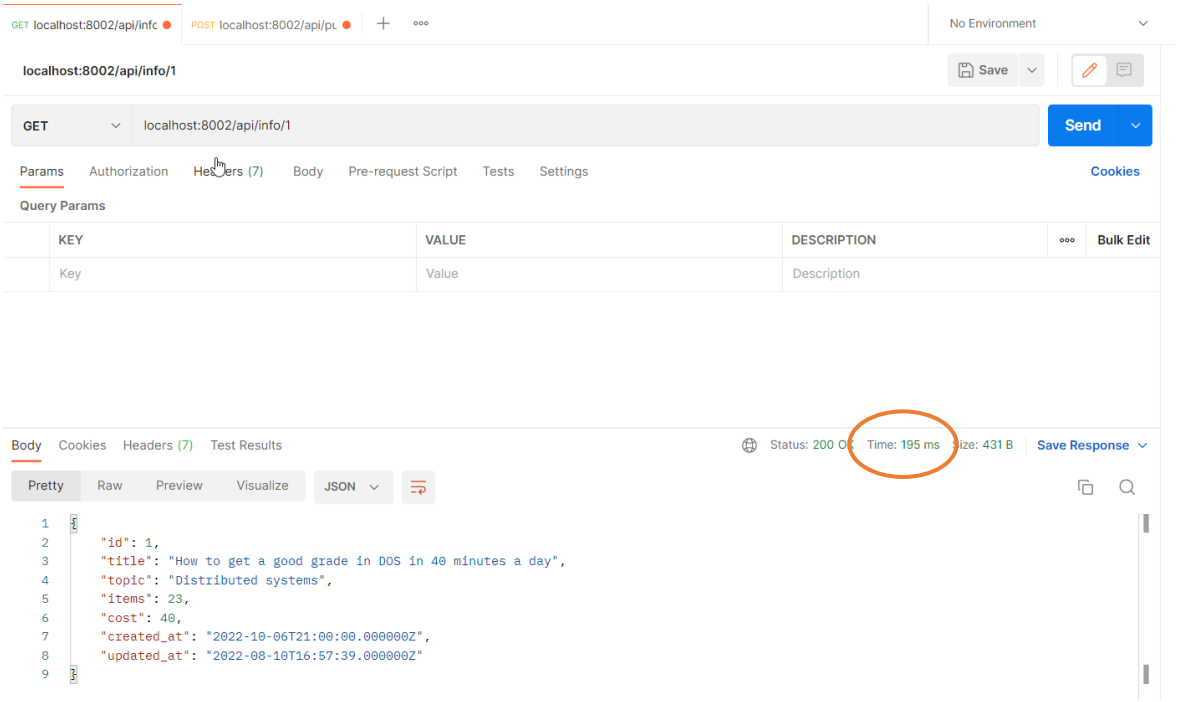
As much as we wanted to have an external database cache like Redis as the performance would be much better, we didn’t have the time to set it up and decided to go with the array approach.

Repeating a request we can notice the time of respond is faster each time.

-First time we will get the data from the database, next time is faster since the data is cached .







- In order to run our project, we need to open the xamp and every server on their port, we used postman while working since we can see more details about the request, time and the status