Bazar.com: A Multi-tier Online Book Store

Name:mustafa Ramahi

Id:11923890

Overall Program Design:

The program consists of three components: a Catalog Server, an Order Server, and a Front-End Server. The Catalog Server manages a collection of books, allowing users to search, retrieve information, and update stock levels. The Order Server handles the purchase process, coordinating with the Catalog Server to update stock and generating order IDs. The Front-End Server acts as a proxy, forwarding requests from clients to either the Catalog or Order Server.

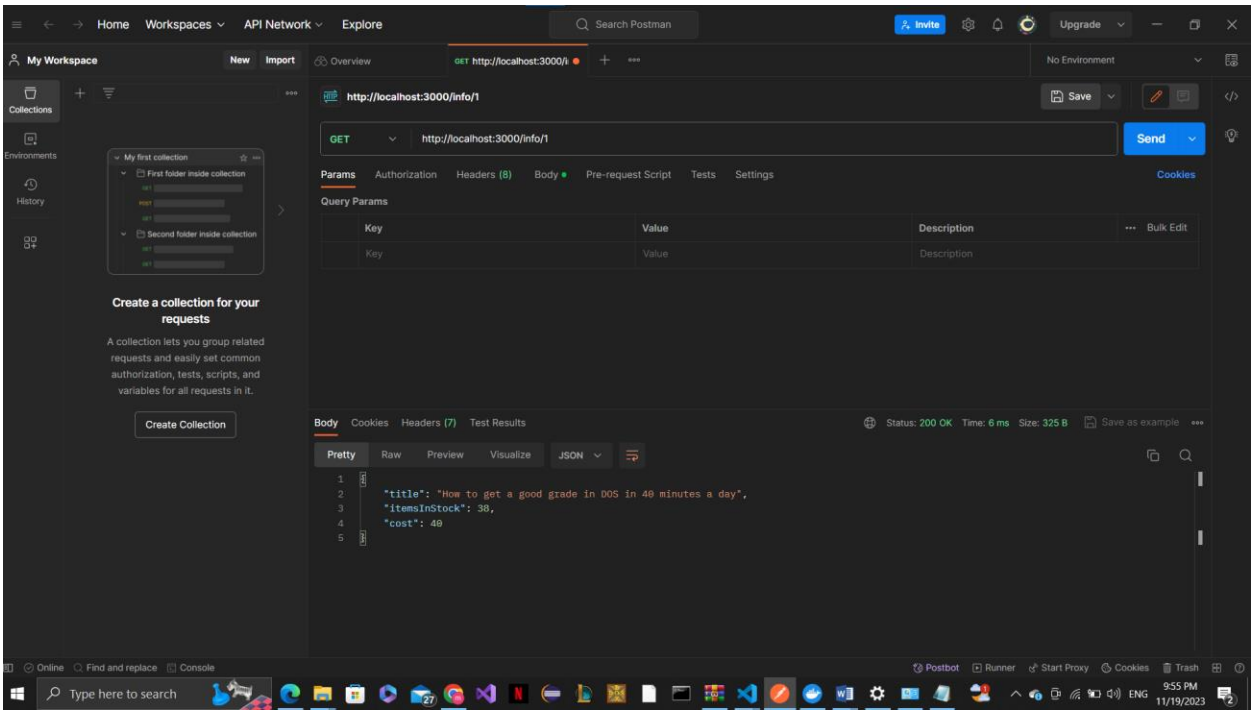
How It Works:

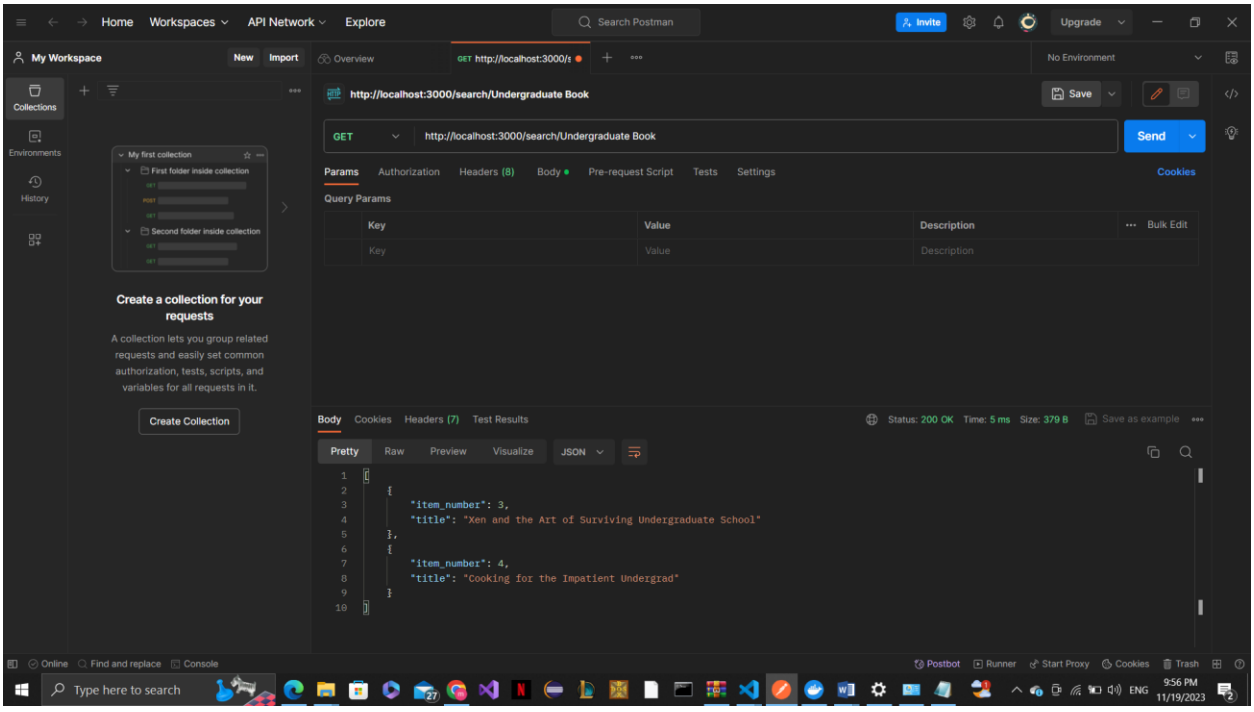
Catalog Server:

Exposes endpoints for searching (/search/:topic), retrieving book information (/info/:item_number), and updating book details (/update/:item_number).

Handles these requests by interacting with a books module, which likely encapsulates book-related logic and data.

That it work on localhost:3000


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
GET http://localhost:3000/i
No Environment
Collections
Environments
History
http://localhost:3000/info/1
Save
GET
http://localhost:3000/info/1
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 6 ms
Size: 325 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"title": "How to get a good grade in DOS in 40 minutes a day",
"itemsInStock": 38,
"cost": 40
}
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash
Type here to search
ENG
9:55 PM
11/19/2023


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
GET http://localhost:3000/s
No Environment
Collections
Environments
History
http://localhost:3000/search/Undergraduate Book
Save
GET
http://localhost:3000/search/Undergraduate Book
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 5 ms
Size: 379 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
[
{
"item_number": 3,
"title": "Xen and the Art of Surviving Undergraduate School"
},
{
"item_number": 4,
"title": "Cooking for the Impatient Undergrad"
}
]
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash
Type here to search
ENG
9:56 PM
11/19/2023

Order Server:

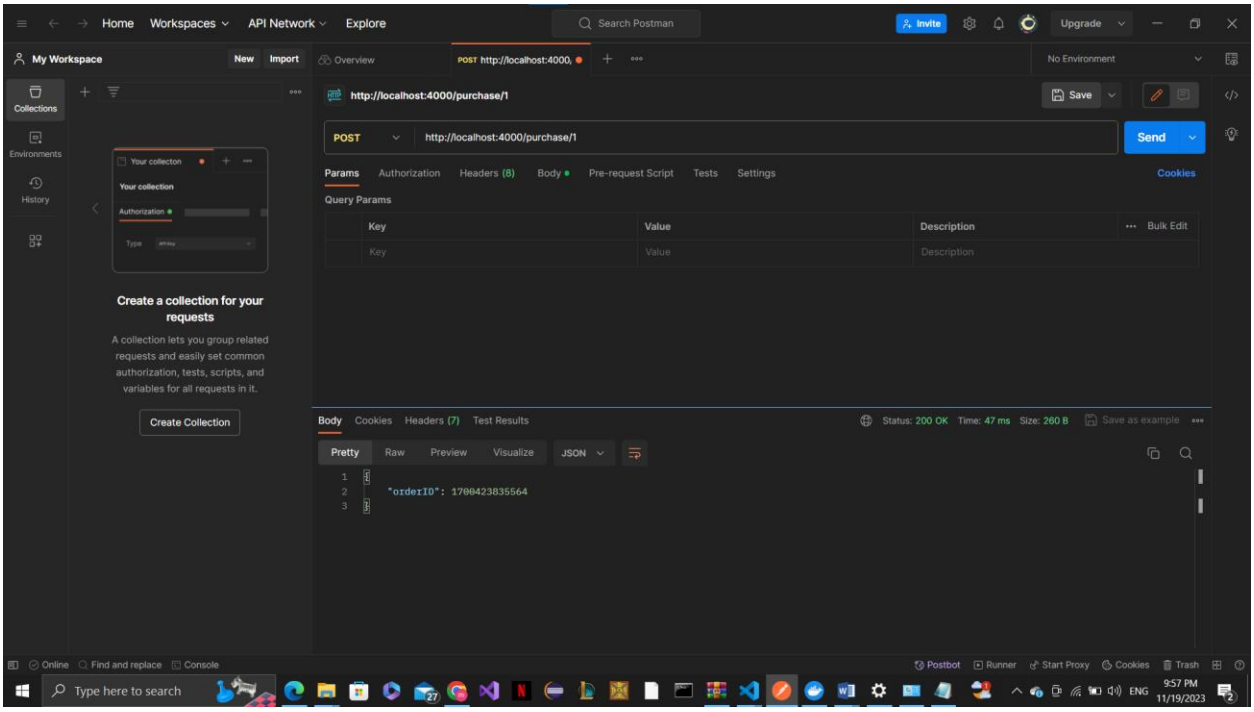
Listens for purchase requests (/purchase/:item_number).

Sends a GET request to the Catalog Server to check if the item is available.

If available, sends a PUT request to the Catalog Server to update stock and adds an order using the orders module.

Returns the order ID to the client.

That it work on localhost:4000



Front-End Server:

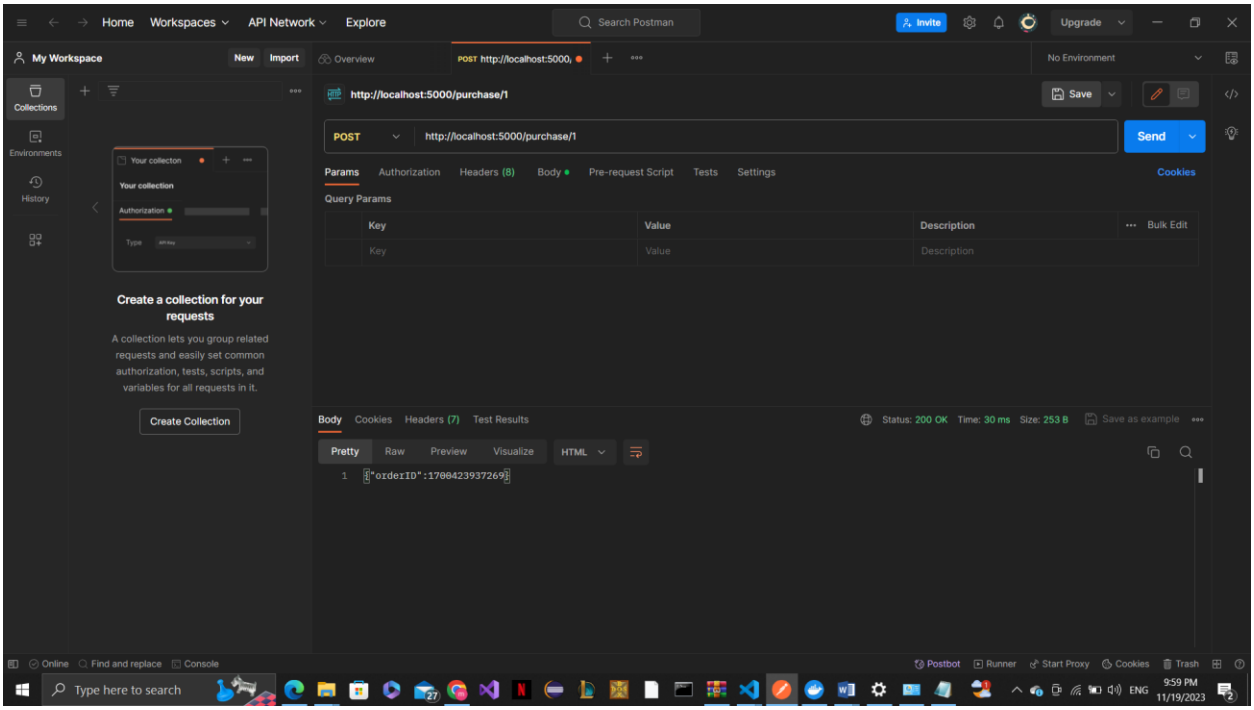
Acts as a middleware between the client and other servers.

Forwards requests to either the Catalog or Order Server based on the endpoint.

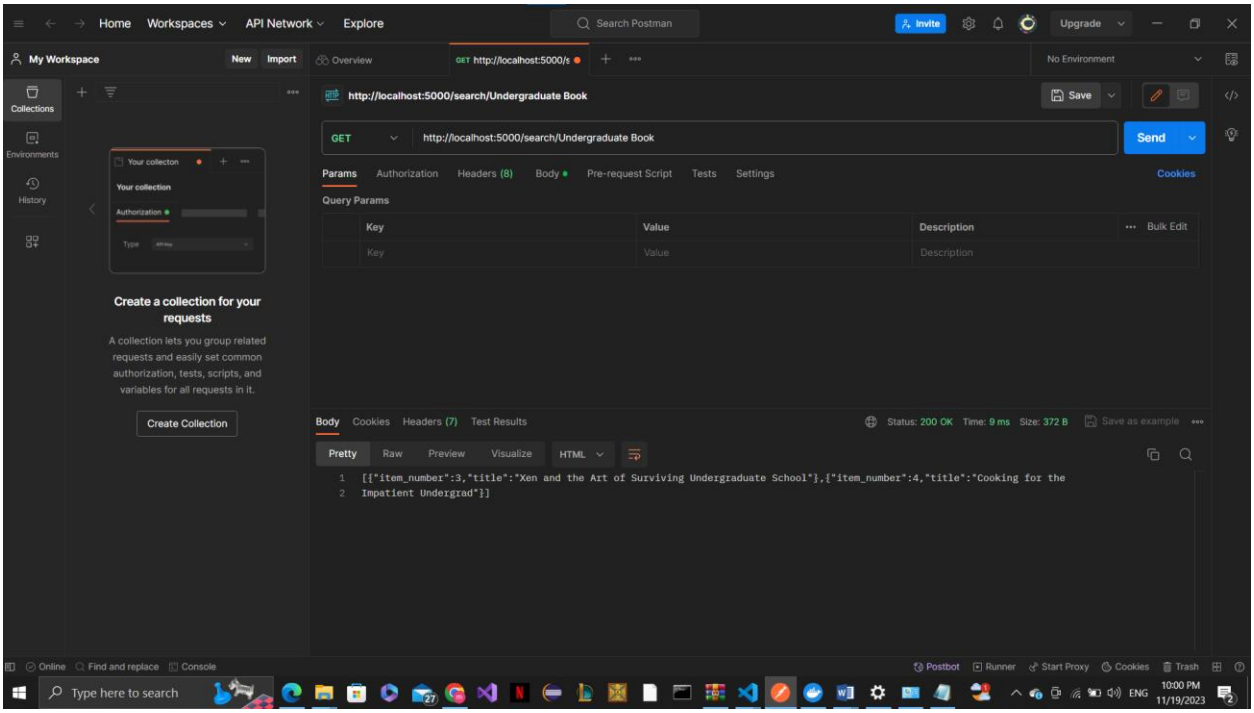
Uses the request module to make HTTP requests.

That it work on localhost:5000

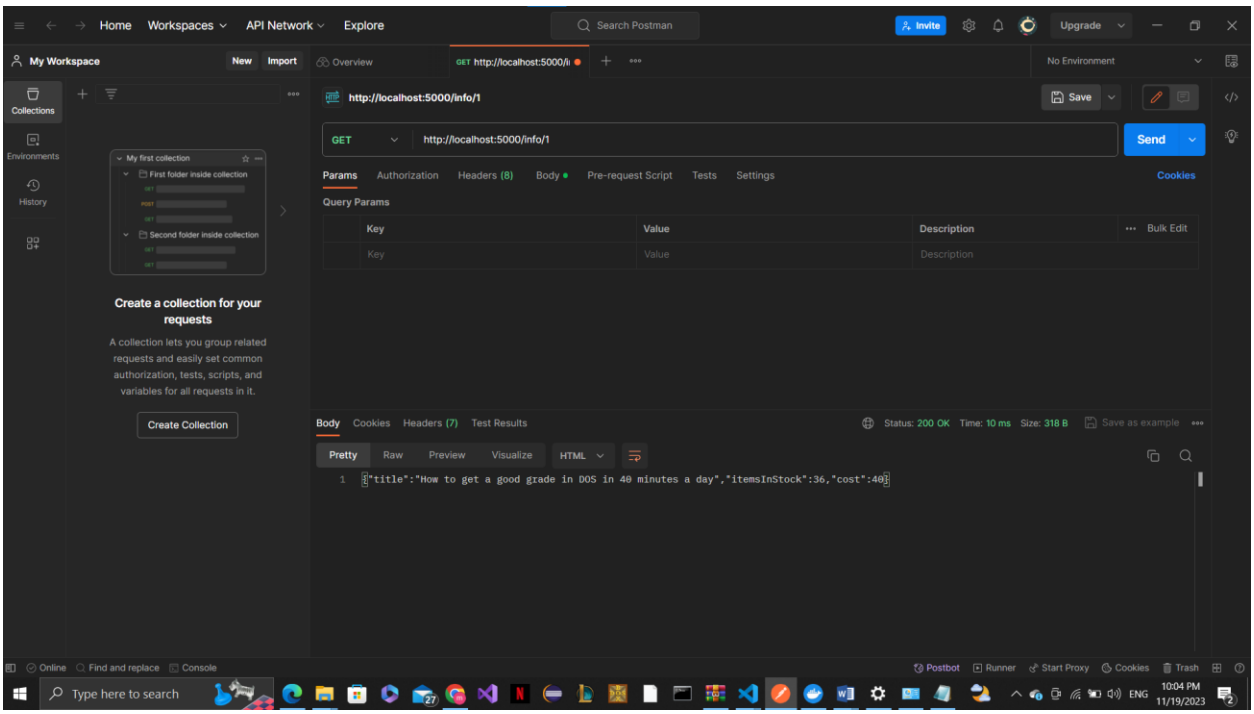
In this I make purchase from front end server.



In this I make search from front end server.



In this I make info from front end server.



Built from the front-end server's Dockerfile.

Docker Compose:

A Docker Compose file (docker-compose.yml) is provided to orchestrate the deployment of the three containers. It defines the services, their configurations, and the network they share. This simplifies the deployment process and allows running the entire application stack with a single command.

Running the Program with Docker:

Build Docker Images:

Navigate to each server's directory containing the Dockerfile.

Build the images using the command: docker build -t <image-name>:<tag> .

Docker Compose:

In the project root directory, run docker-compose up.

This command starts the defined services, creating containers for the Catalog, Order, and Front-End Servers.

Accessing Services:

The Catalog Server is accessible at http://localhost:3000.

The Order Server is accessible at http://localhost:4000.

The Front-End Server is accessible at http://localhost:5000.

Docker Containers:

The program has been containerized using Docker to simplify deployment and ensure consistent environments across different systems. There are three main containers:

Catalog Server Container:

Hosts the Catalog Server.

Exposes the necessary ports to communicate with the Front-End Server.

Built from the catalog server's Dockerfile.

Order Server Container:

Hosts the Order Server.

Exposes the necessary ports to communicate with the Front-End Server.

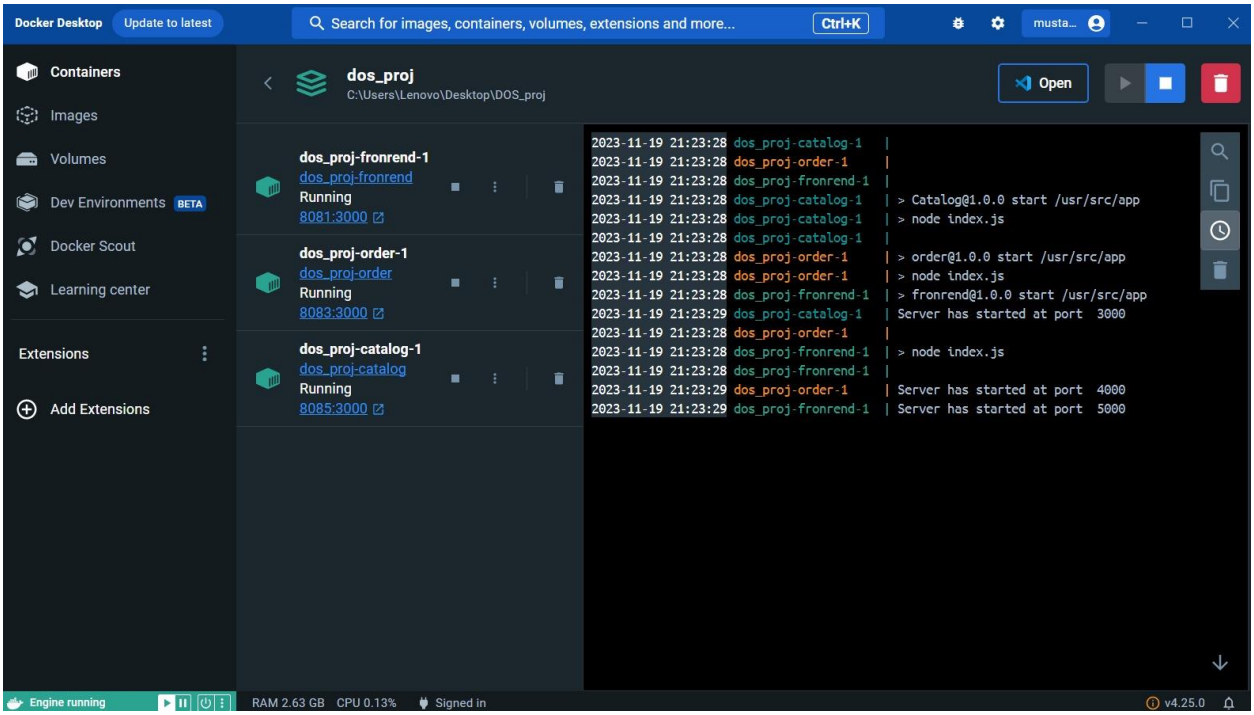
Built from the order server's Dockerfile.

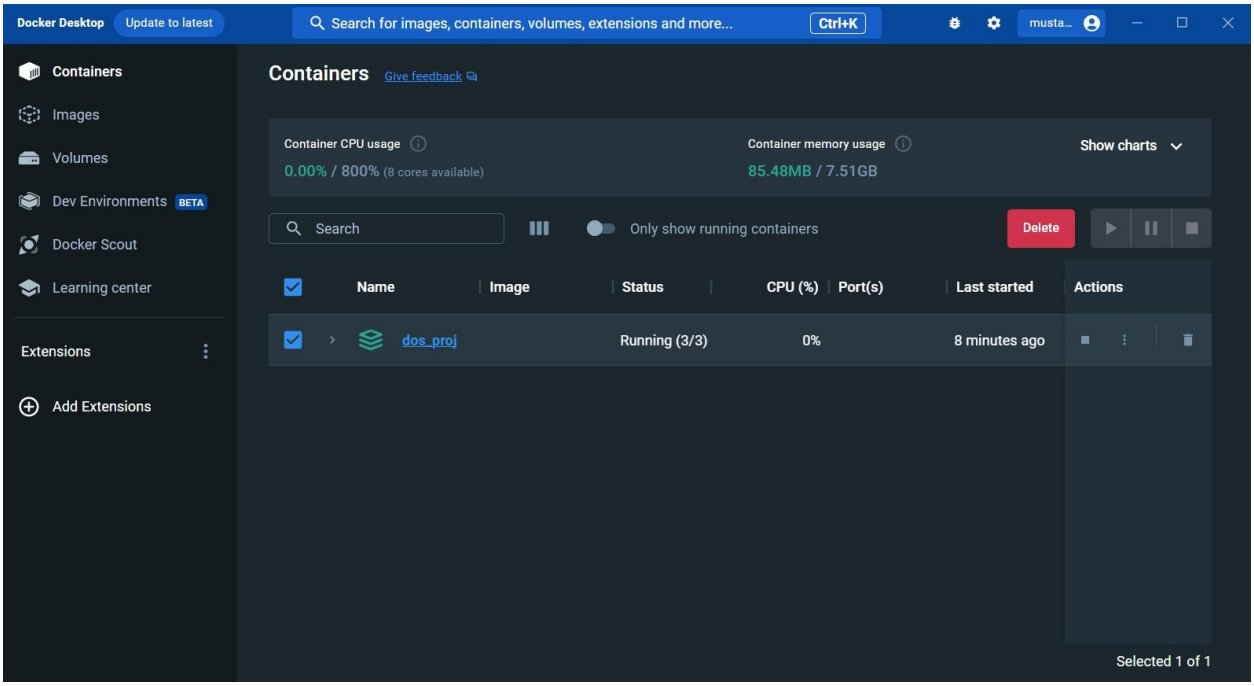
Front-End Server Container:

Hosts the Front-End Server.

Exposes the necessary ports to accept client requests.

Built from the front-end server's Dockerfile.





I make three containers and images