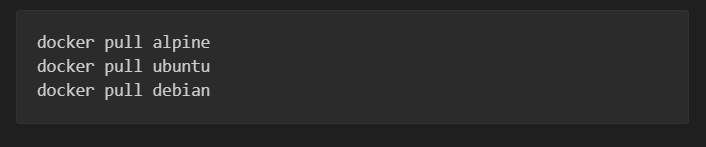
1. Pull docker linux images that are used in this lap for the 3 tiers



2-define tiers > catalog server,order server and frontend tier

And will implement them and connect them based on micro services approach

3- minimizing project

Because docker is used there is no need for virtual environments or packages installed

This process is going to be executed in the image it self when it is being built

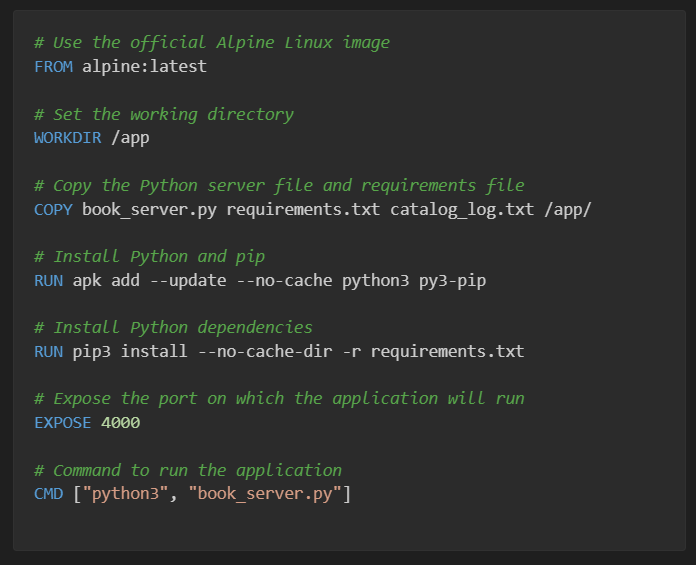
So all required is to list required packages|libraries so that container will know which packages to install

To do that we make a requirements.txt file using below command

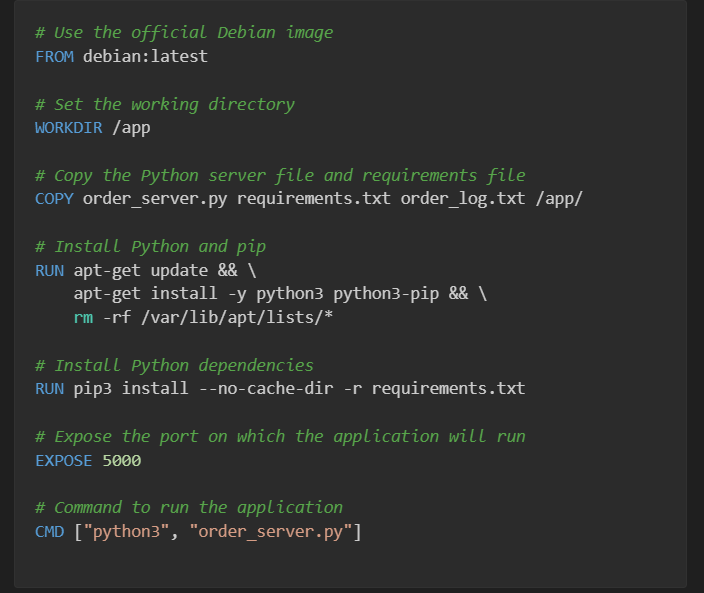
**Pip freeze > requirements.txt**

3- build the Dockerfile for each tier

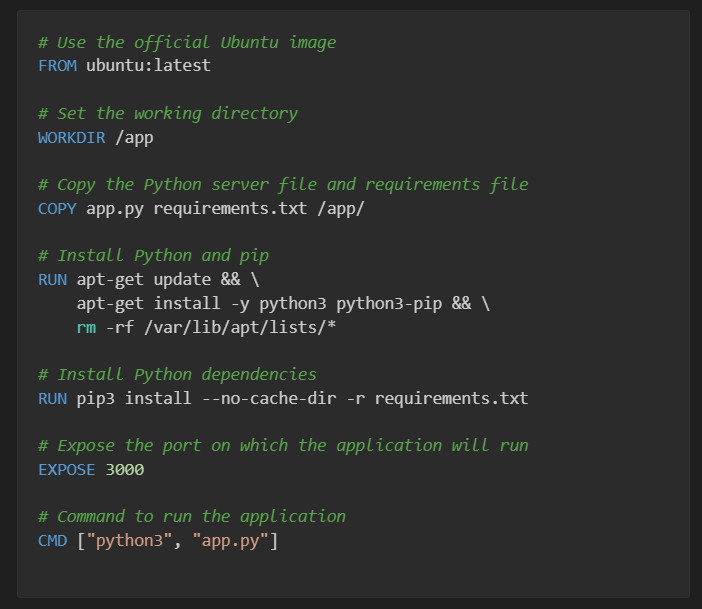
Catalog server tier:



Order server tier:



Finally frontend tier



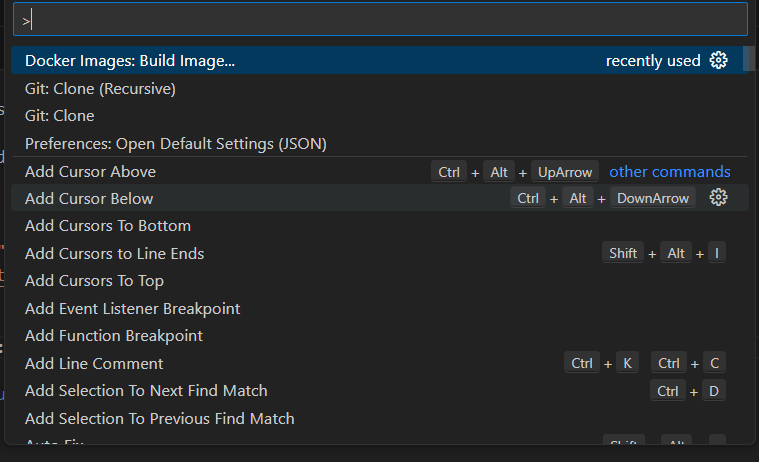
will use these Dockerfile to build the docker images of linux

Every docker image will have it’s own port exposed to the tier server whether catalog,order or frontend

4-building Dockerfile and creating images

Using vscode facilities the process of building image

Using “ctrl + shift + p” and selecting “Docker Images: build image” option



5- creating new docker volume for the images|containers

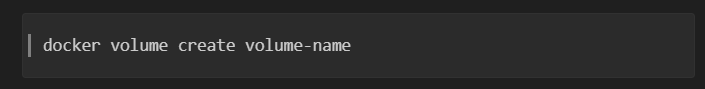
Because each one of the tiers is using the same Database and in this lap sqlite was chosen

And because sqlite is a local database so it means only localhost of the docker container can access it

But other containers wont’t

So volume was created to be shared between all containers

Using below command:



And relative path of sqlite for container is

app.config["SQLALCHEMY\_DATABASE\_URI"] = "sqlite:////data/project.db"

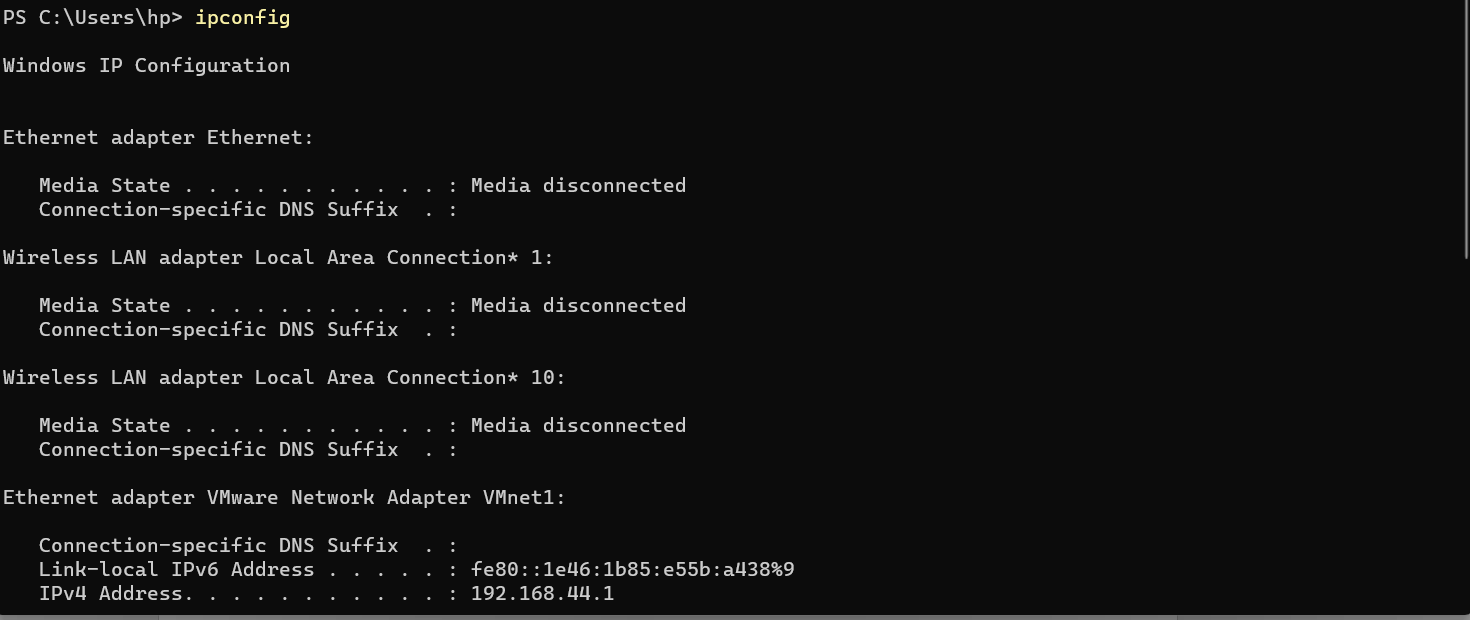
6- Containers Communication

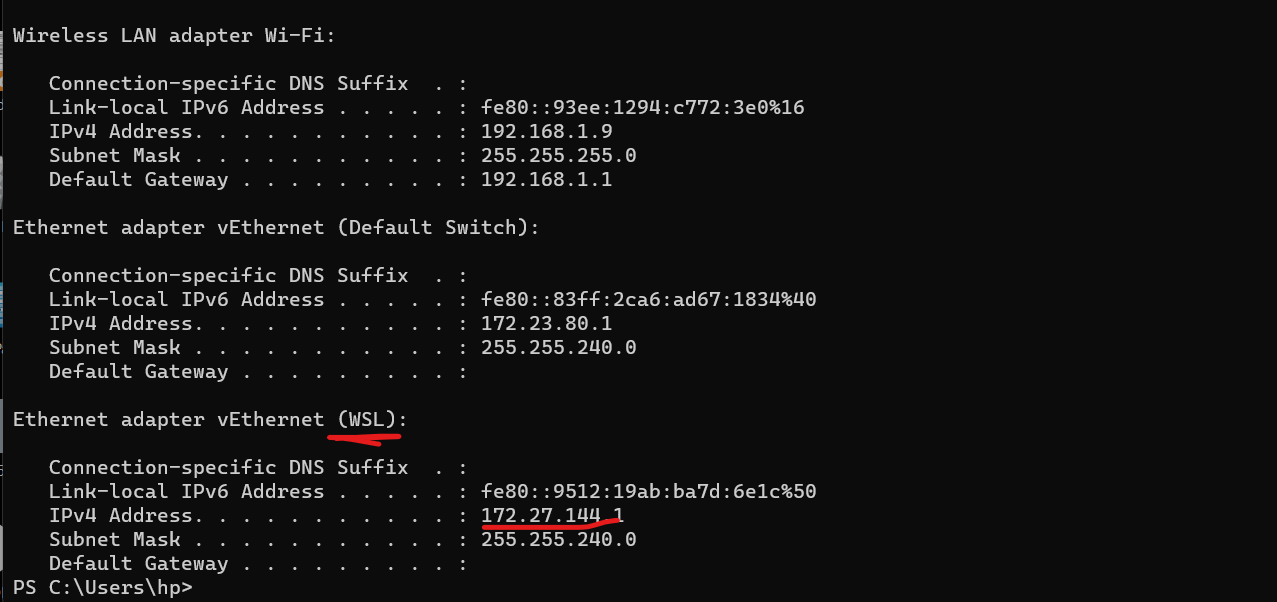
As tiers are in different containers so that they will need to communicate with each other

For this simple a basic communication way was used, because all containers are at same local machine (PC | Host) so they will the same Host gateway

The Host gateway is the ipv4 address, but docker is a subsystem of windows so it has it’s own different ipv4 addres

To gain it use below command

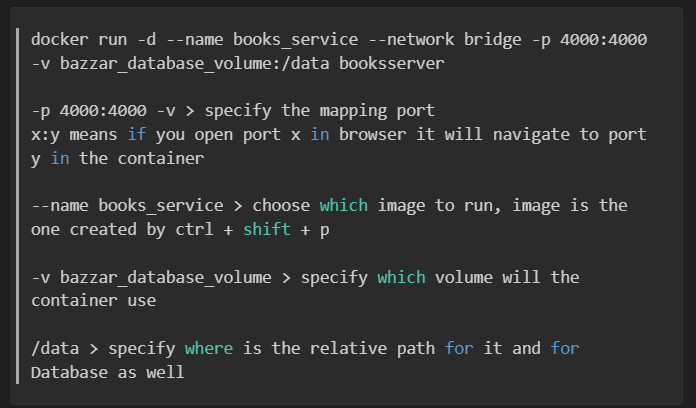




That will be used when communicating with between containers

As <http://172.27.144.1:port/>endpoint

7-finally make instances of images and run it using below command



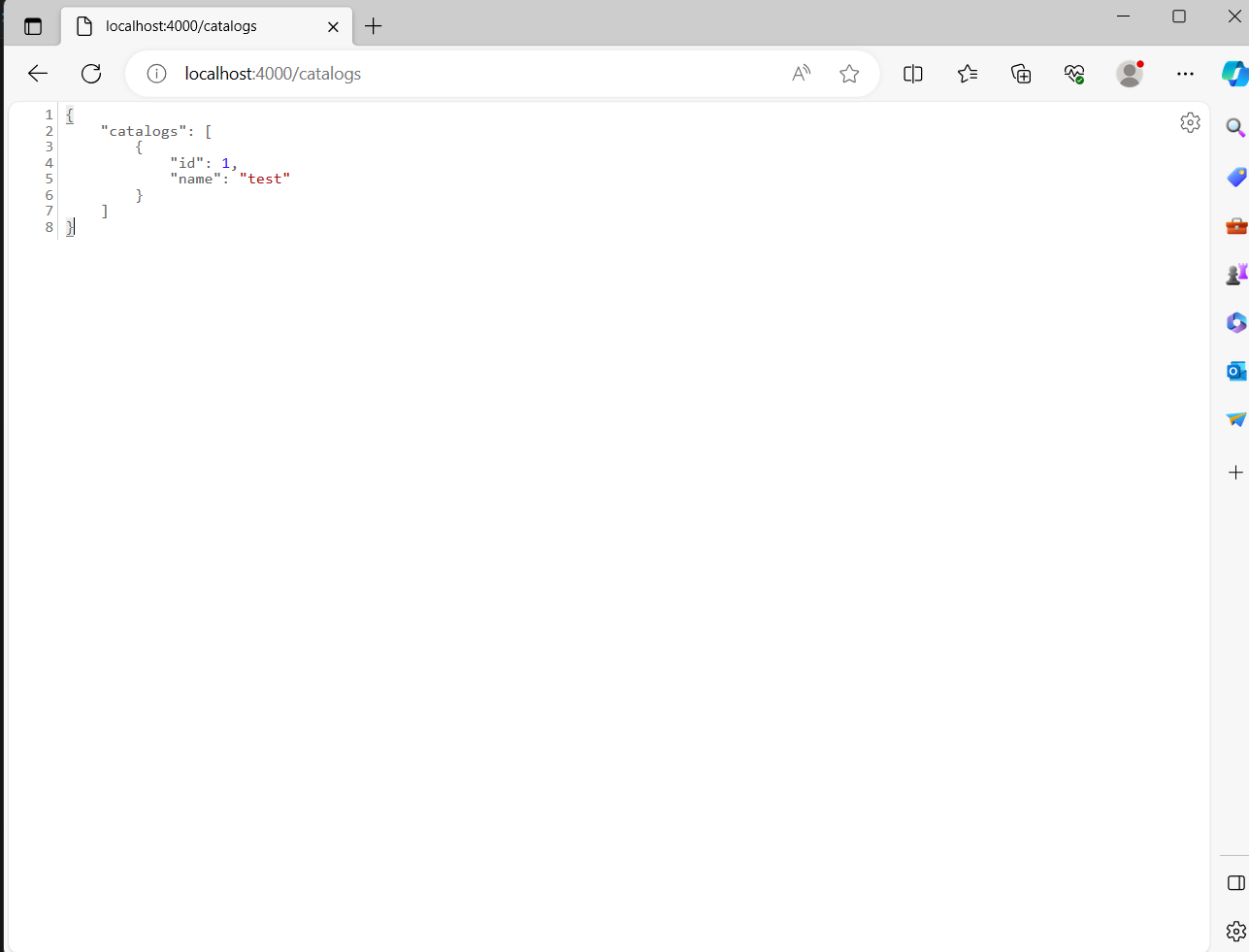
-p 4000:4000 -v > specify the mapping port

x:y means if you open port x in browser it will navigate to port y in the container

--name books\_service > choose which image to run, image is the one created by ctrl + shift + p

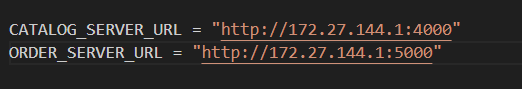
-v bazzar\_database\_volume > specify which volume will the container use

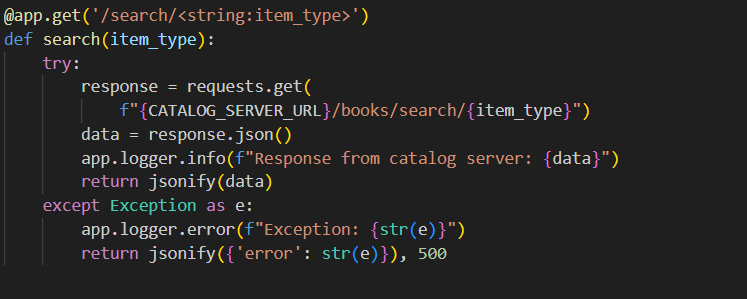
/data > specify where is the relative path for it and for Database as well

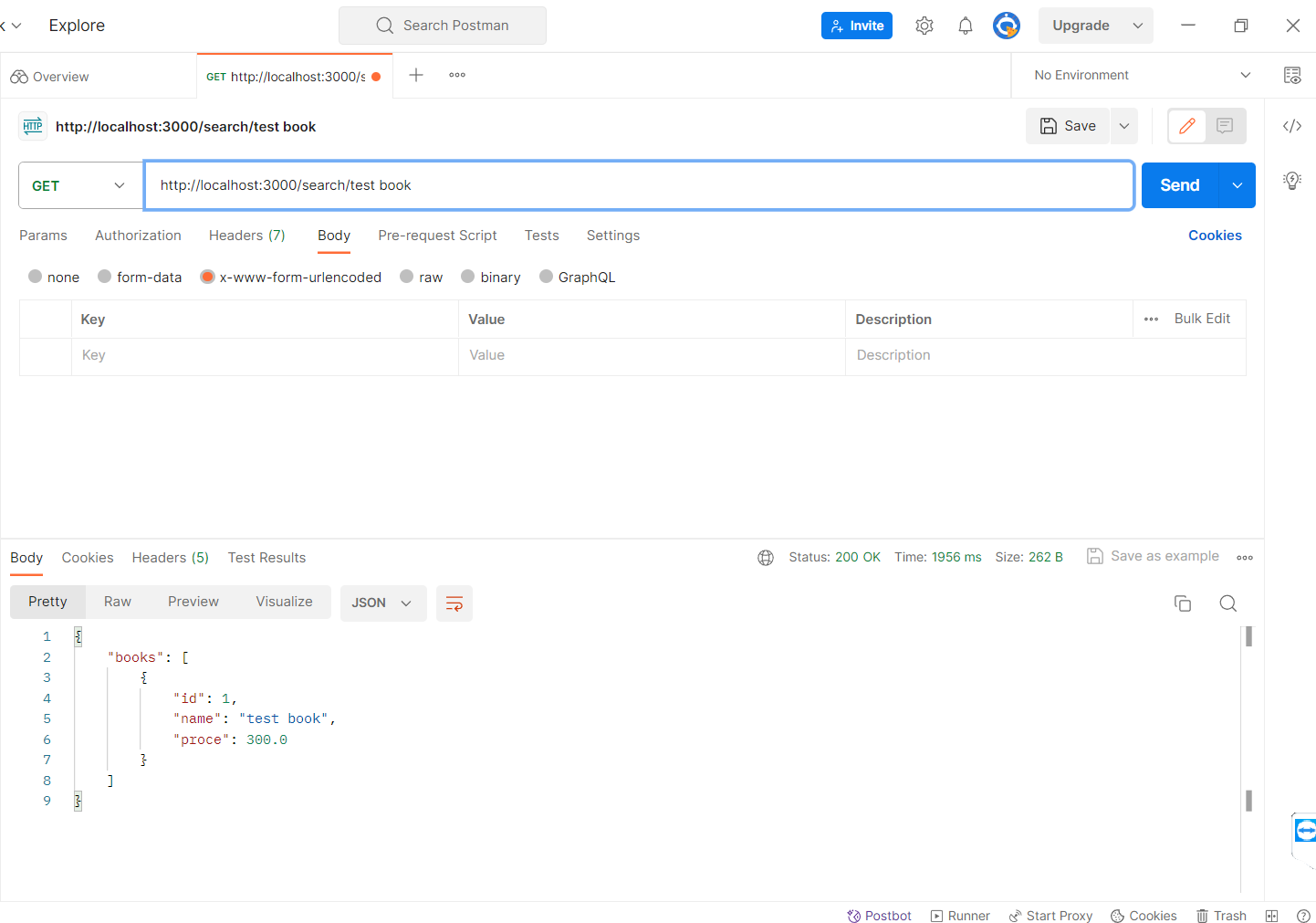


Front tier at port 3000 communicates with vatalogs at port 4000 to search a book

Testing it:







All Works Fine……