**Lab1 – Bazar.com: A multi-tier online book store**

In this lab, we have designed a book store as multi-tier architecture. While the concept has been applied by building three servers: Catalog server: Which contains the DB (SQLite), initially the DB has only one table with 4 rows of data as described in the problem. The order server: maintains a list of all orders received for the books. The frontend is implemented as a single microservice, it has the URLs of the main APIs provided to the user to access through other servers and get data.

The frontend supports three main operations: search (topic), info (item-number) and purchase (item-number). The first two operations trigger queries on the catalog server directly, while the last operation triggers the request to the order server.

The catalog server supports only two operations: query and update, query is mainly two types: query-by-subject and query-by-item. The update operation allows the cost of an item to be updated or the number of items in the stock to be increased or decreased.

***How to tackle the problem?***

We used Flask in python web framework for the backend for each server and SQLite DB.

First of all, we create new virtual machines on VirtualBox, with Ubuntu OS. Each machine is for one server. The machines’ network is connected as bridged network. Machines can communicate with each other by URL, specifying the appropriate API, IP addresses and port number of the process.

***Install the requirements:***

On the machine holds the catalog server, we need SQLite DB:

***sudo apt install SQLite3***

On each machine, install python3 virtual environment:

***Sudo apt install python3-venv***

Then we install git:

***Sudo apt install git***

Create a directory on each machine for creating project and writing code:

***Mkdir Catalog\_server // Machine 1***

***Mkdir Frontend\_server // Machine 2***

***Mkdir Order\_server // Machine 3***

Initializing virtual environment in each directory:

***Python3 -m venv Catalog-Venv***

***Python3 -m venv Catalog-Venv***

***Python3 -m venv Catalog-Venv***

Activate the venv:

***. “name-of-venv/bin/activate”***

After creating python virtual environment, we can use pip for python to install flask:

***Pip install flask // On each machine***

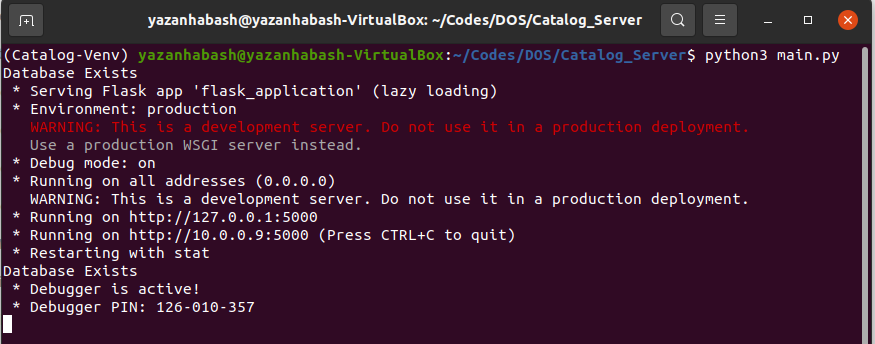
After creating files and writing code for each server, we have files called “flask\_application.py” which contains the IP addresses and port numbers for needed another server on other machine. i.e., if the frontend needs to send request to order server, it needs the IP address and port number of the process running the app, etc.

In the main.py, we can specify by the argument **host** that the app can be accessed by any machine, we give it subnet mask as following below:

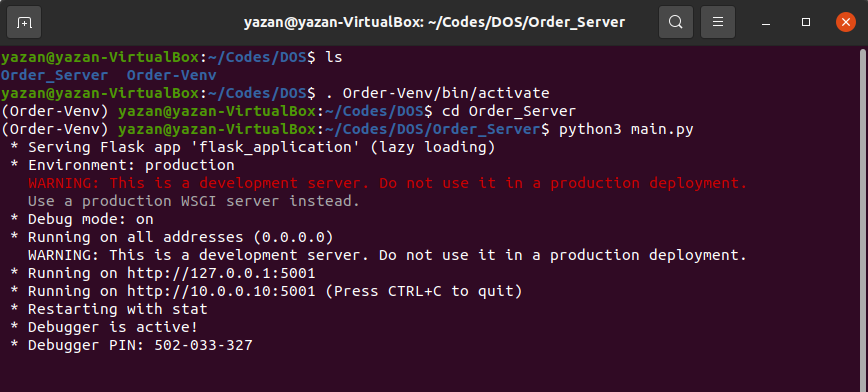
*If \_\_name\_\_ == “\_\_main\_\_”:*

*app.run(host=”0.0.0.0”, debug=True, port=”some port number”)*

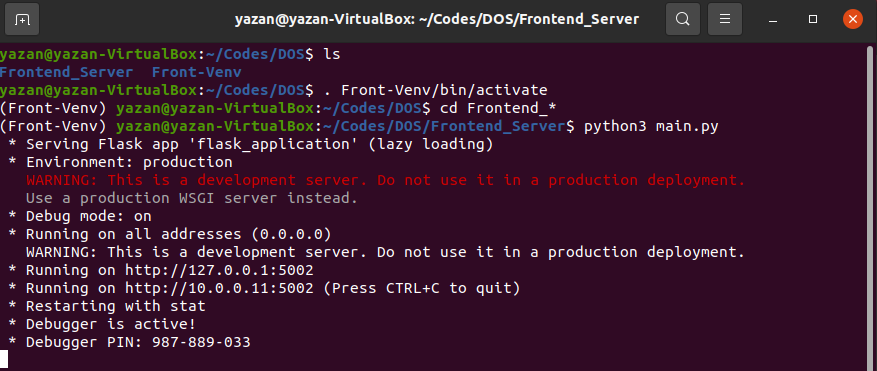
1. Run the catalog server; 1st machine:



1. Run the order server; 2nd machine:



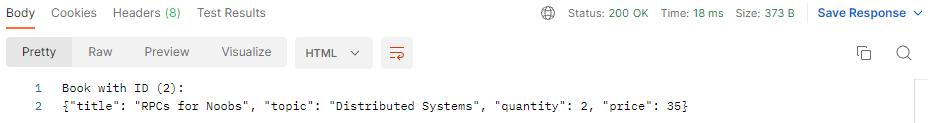
1. Run the frontend server; 3rd machine:



We will test all frontend APIs:

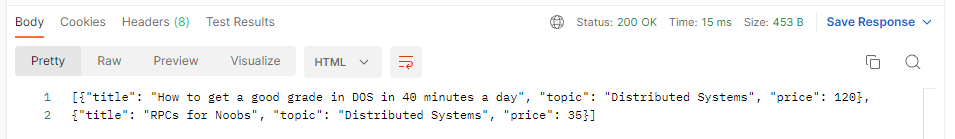
1. Get book by ID, assume client from Postman (Windows host OS will send to frontend server):





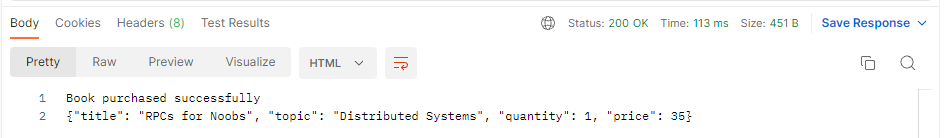
1. Get books by subject (Distributed Systems):





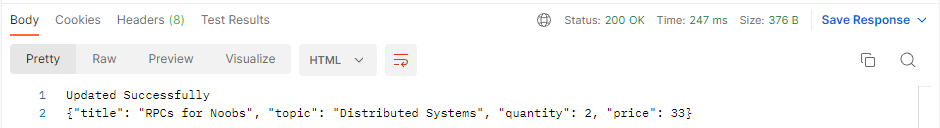
1. Purchase Book by ID:

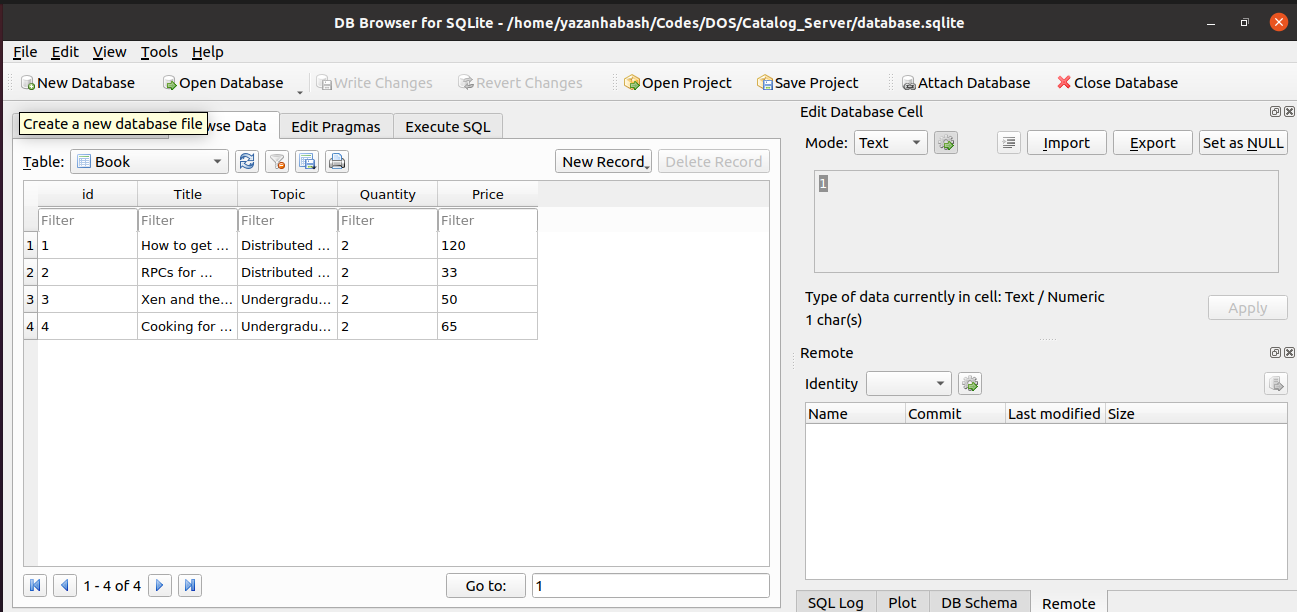




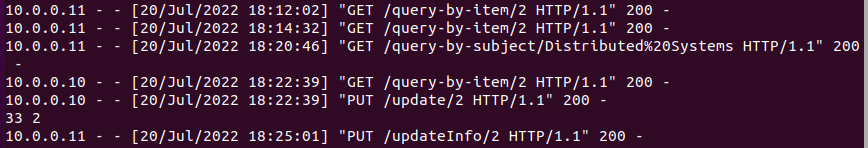
1. Edit the quantity and price of a book:







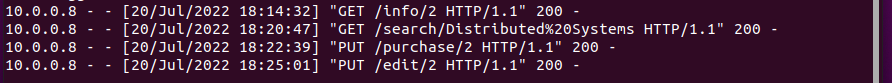
Requests accepted from catalog server (previous steps):



Requests accepted from order server (previous steps):



Requests accepted from the frontend server (previous steps):



By: Yazan Habash & Ashraf Hab-Rumman