AN-NAJAH NATIONAL UNIVERSITY

FACULTY OF ENGINEERING AND INFORMATION TECHNOLOGY

Computer Engineering Department

DOS (Distributed Operating Systems)

BAZAR.COM

Students
Maya Yacoub
Aya Zitawe

Instructor
Dr.Samer Arandi

Palestine, July 2022

Contents

List of Figures

1

Introduction

1.1 Problem Statement

Design a Bazar.com - the World's smallest book store. A two-tiered web design will be deployed using micro-services for both the front-end and the back-end with the following requirements:

- front-end tier has one service
  - The user requests will be handled by this service and forwarded to the back-end services.
- back-end tier with two servers
  - **Catalog server** consists of the catalog, database of the books providing its information like book unique number, name, topic, number of items in stock, and the cost. In addition to handling the requests passed to it like searching for books in a topic, getting information on specific book, checking if a book is available, and updating the book number of items when purchasing as will be described later.
  - **Order server** for the main purchasing process and send requests to the catalog server.
  - **Database** represented in a CSV file. A file is specified for the catalog table which store all details of the books. Another file used for the orders table that store the information related to the purchase operation like the book ID, cost and ordering time.

1.2 Framework & Tools

- Framework
  - Node framework used which is a lightweight technology tool that supports the use of Node.js and JavaScript language for developing microservices. Gunaratee the **productivity**, **scalability**, and **speed**.
- API platform
  - **Postman** API platform used for building and using APIs.
- Editor
  - Visual studio code editor used. In addition to install Node.js source code to be used in the developing process.
- Npm libraries used in Node.js projects
  - **express** node module on top of Node.js used to manage servers and routes.
  - **body-parser** used to parse the request body in order to get the body using **req.body**.
  - **axios** used to help in requests to back-end services from the front-end service.
  - **fs** used to interact with file system.
  - **fast-csv** used to interact with the database represented as csv file.
  - **csv-writer** to write in the csv file after the updating of the items number after purchasing.
  - **json2csv** used for converting the json format of the data into csv format.

2

Methodology

2.1 How to run the System

The system is distributed on a Linux virtual machine and 2 Windows laptops since one laptop could not run and handle 2 virtual machines at the same time. On each platform needed to install the Node.js source code and the npm libraries needed within each server.

- Installing Node.js source code
  - https://nodejs.org/en/
- Create a folder for each service in visual studio code in each device. Then add a file .js in each one in order to implement the code of the services in. Install the needed packages (npm install library_name) for each project as described in the in-line comments in the code.
- To run the services, in each service terminal type (node file_name.js) and it will start running on the specified port.

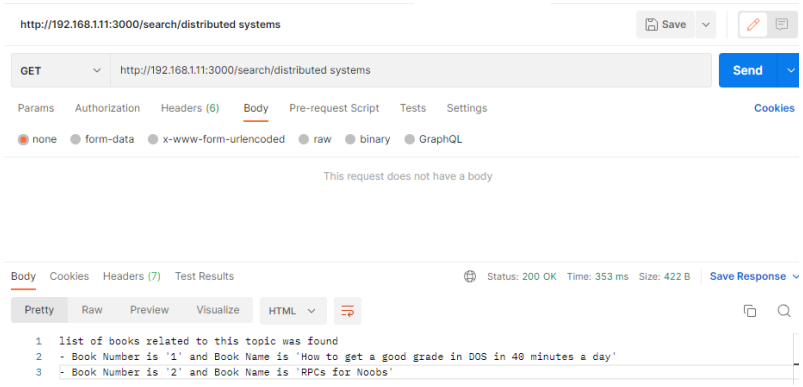
3

Results & Discussion

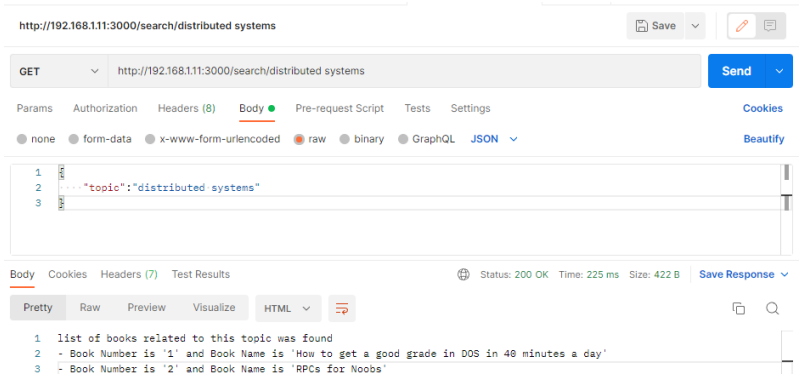
We run the services in 3 machines. We use windows machine for the client and purchase services. and use ubuntu as a virtual machine to run the catalog services. Each machine has its own ip address and port. then we used postman to test and debug the rest API

3.1 Project Outcomes

As mentioned before, the postman tool was used to debug and test the rest API. All requests will be handled by the front-end tier then perform the initial processing. The front-end server supports three operations:
Search by topic get request with parameter in the url.JSON object which will return as shown below:

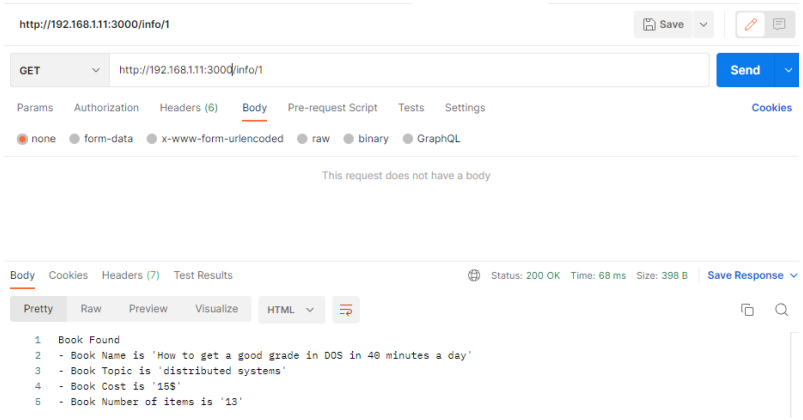


(a) Topic in Header

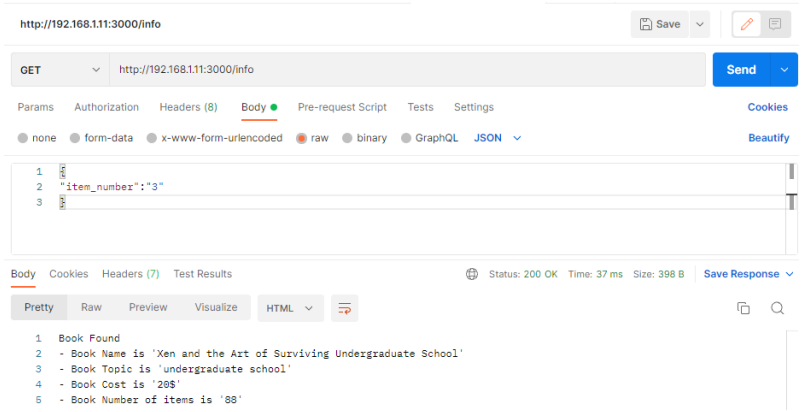


(b) Topic in body

The second operation is the **info**. And the same as the search operation the item number can be passed on the header or in the bode as shown below:

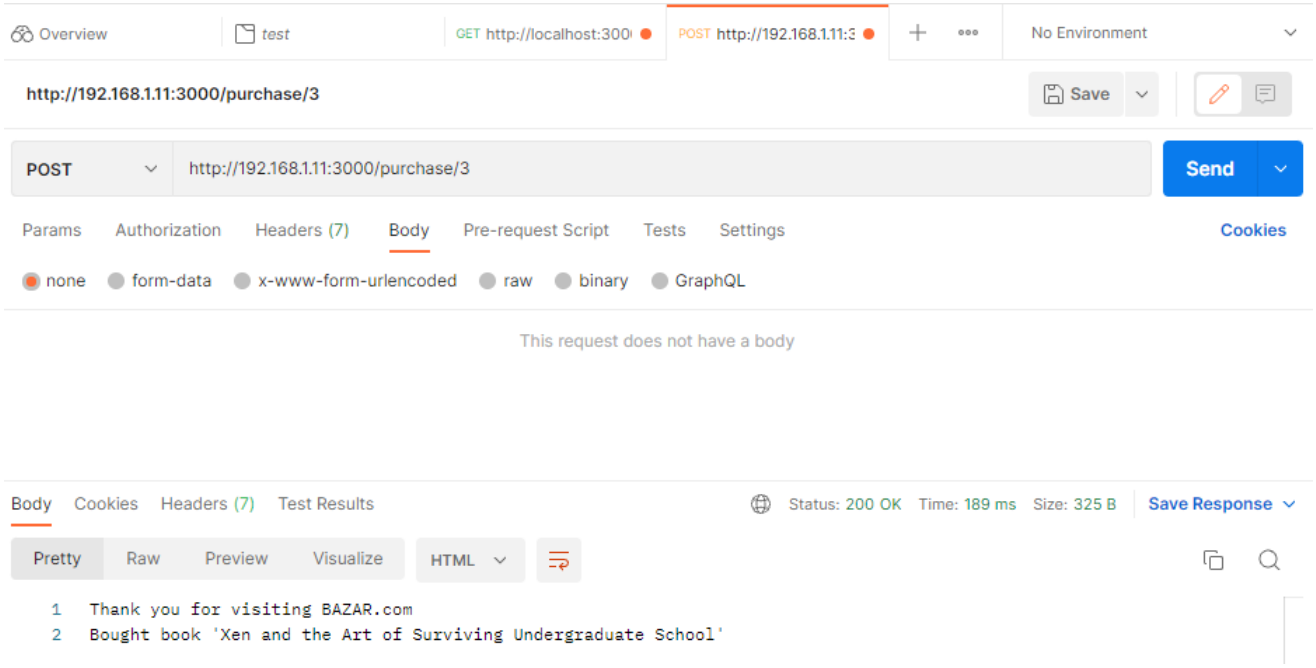


(c) Item number in Header

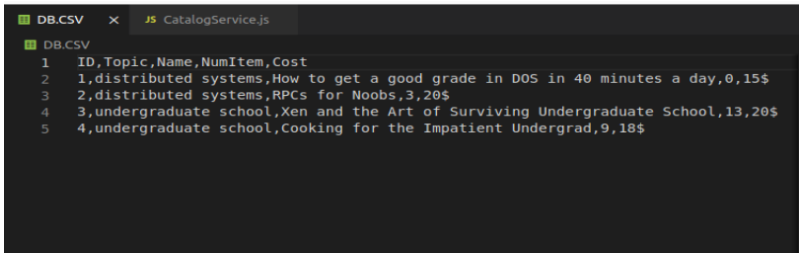


(d) Item number in body

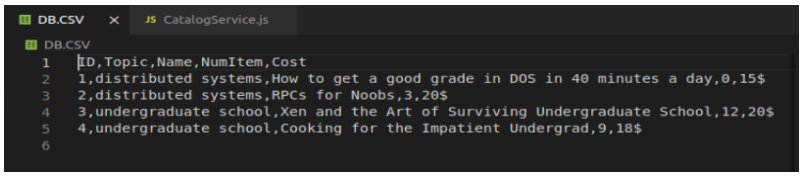
The last operation is the **purchase**. If the number of item in stock not zero then the amount of the database will be decrement by one.



These images show the database before the purchase and after of item number '3':

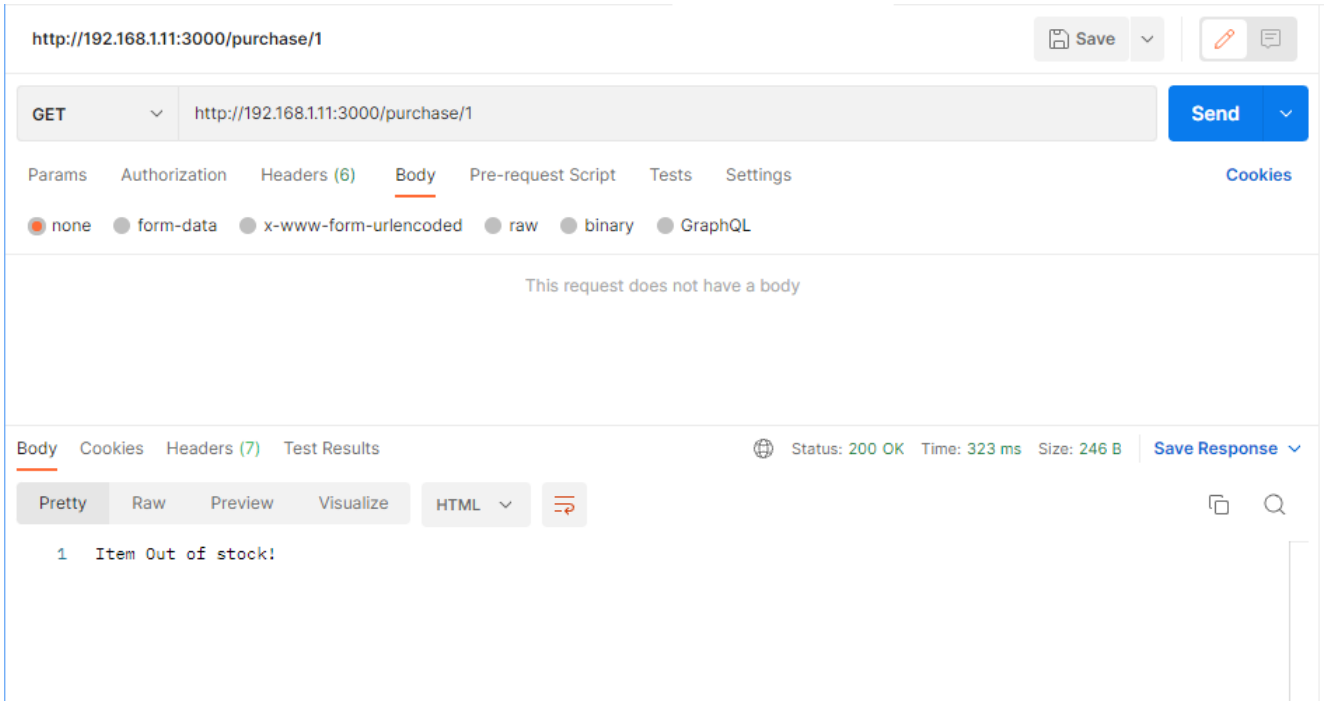


(e) Catalog database before the purchase



(f) Catalog database after the purchase

If the number of the item in stock equal zero as the item of number '1' then the purchase operation will not happen and will send Item Out of stock!



When the purchase is done the order details will be stored in the orders database which specified in a csv file as shown below: