An-Najah National University

Computer Engineering Department

Distributed Operating System (DOS)

BookStore MicroServices System

Name of Student :

Afaf Nasr & Yasmine Saad

Dr .Samer Arandi

Introduction

This project demonstrates a simple bookstore system based on microservices using Node.js, Express, SQLite, and Docker. It is designed to illustrate inter-service communication, isolation, and deployment using container-based architecture

Technologies Used

Node.js → Backend runtime

Express.js → Web framework

SQLite →Lightweight database per service

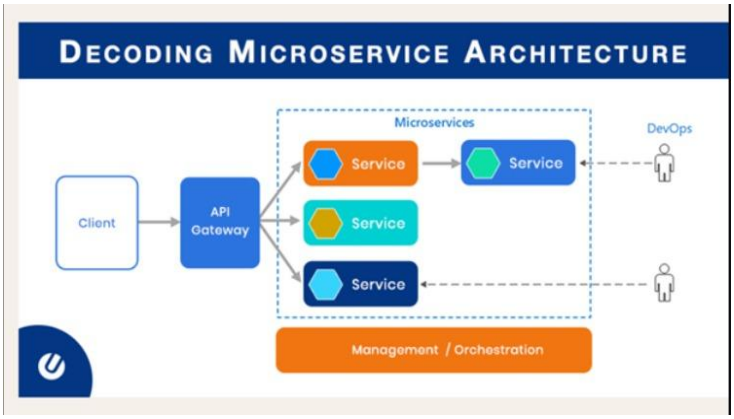
Docker → Containerization

Postman→ Testing APIs

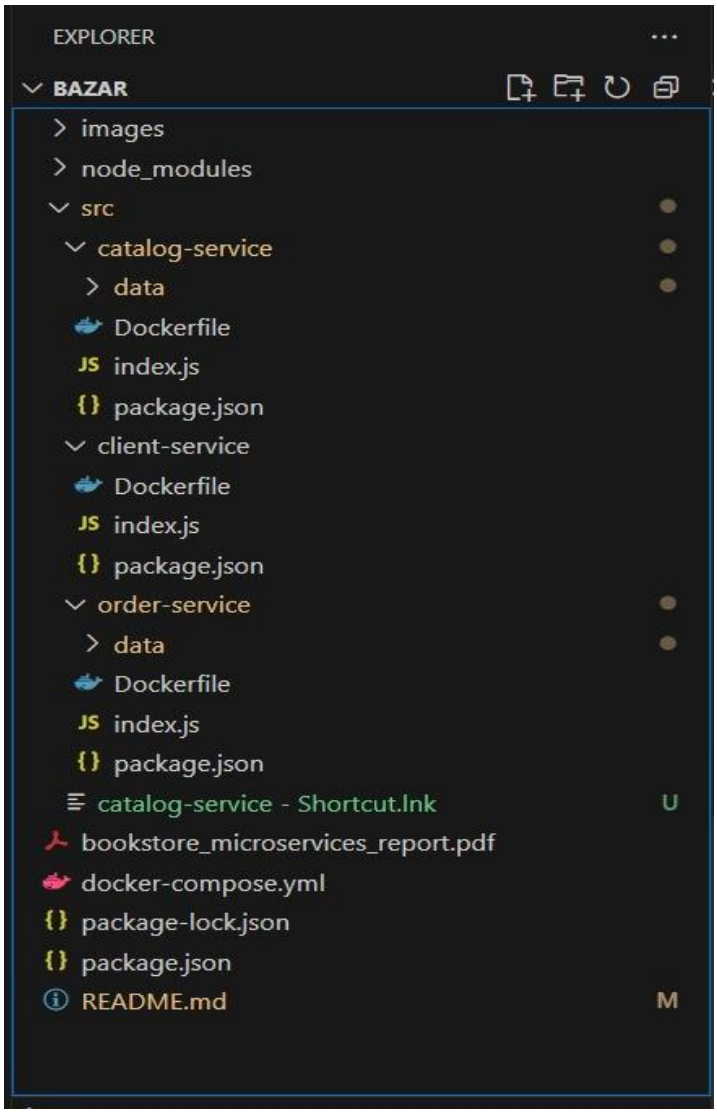
Git/GitHub→ Version control

System Architecture

The client interacts with the system via the front-end (client service). Requests are routed to either the catalog or order services using REST API calls. Each service handles its own database and logic independently



Project Structure



src/order-service *:* Implements the Order Service — handles book purchases and order tracking

src/order-service/index.js : Main entry point; processes /purchase/:id and records orders

src/order-service/Dockerfile : Builds the Docker image for the Order Service

src/order-service/data : Holds the SQLite database for order management

src/order-service/data/orders.db : Stores order history and purchased book records

src/client-service : Implements the Client Service — acts as API gateway and entry point for users

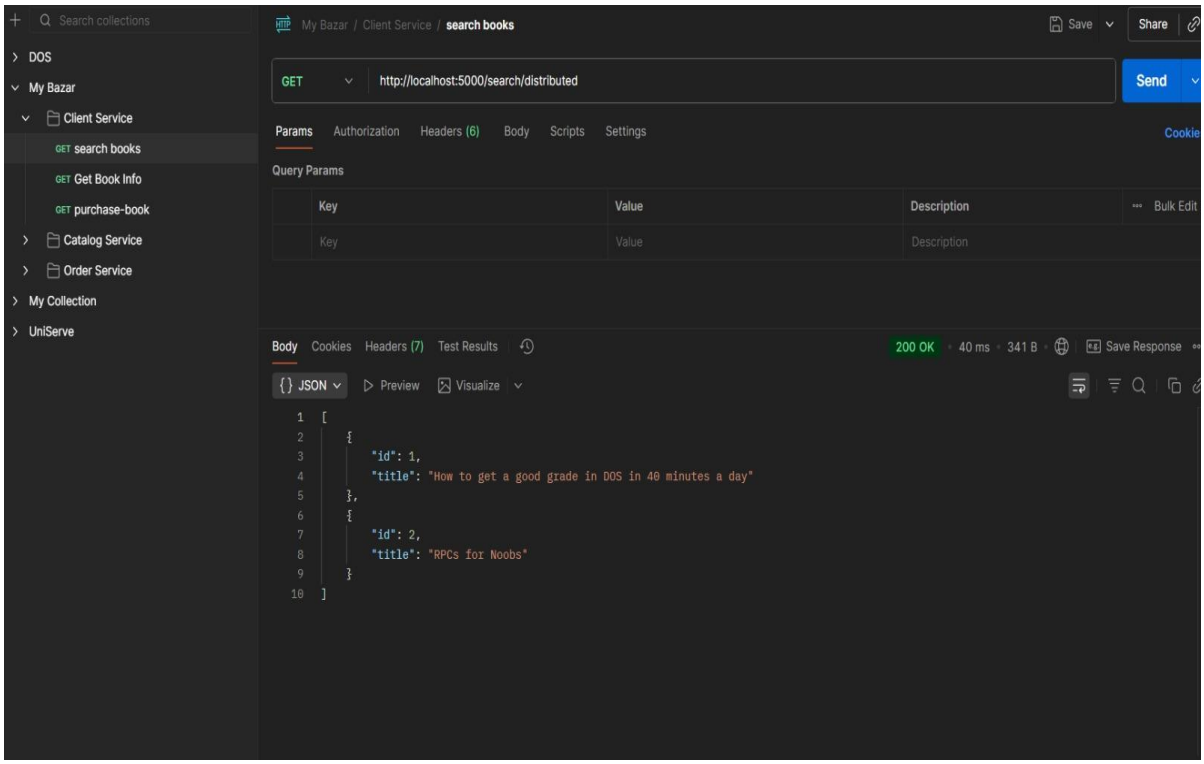
src/catalog-service : Manages book data: search, details, and catalog browsing

src/client-service : Acts as an API gateway or frontend for interacting with users

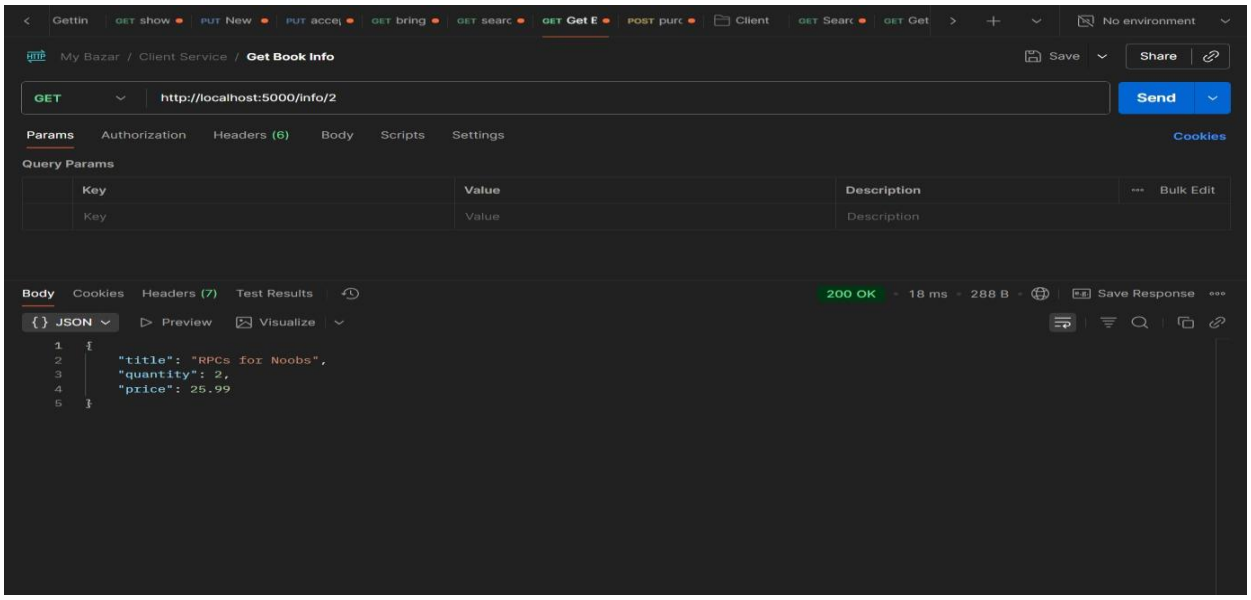
API Endpoints & Screenshots

Catalog Service — http://localhost:5001:

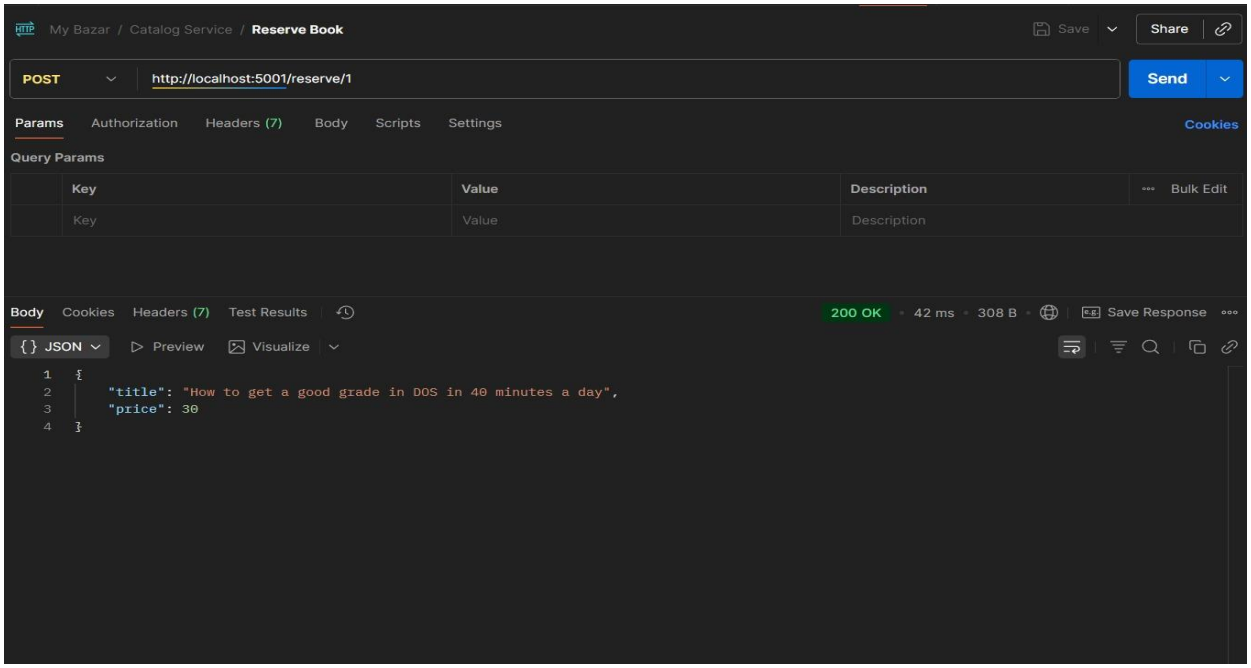
GET /search/:topic → Search books by topic



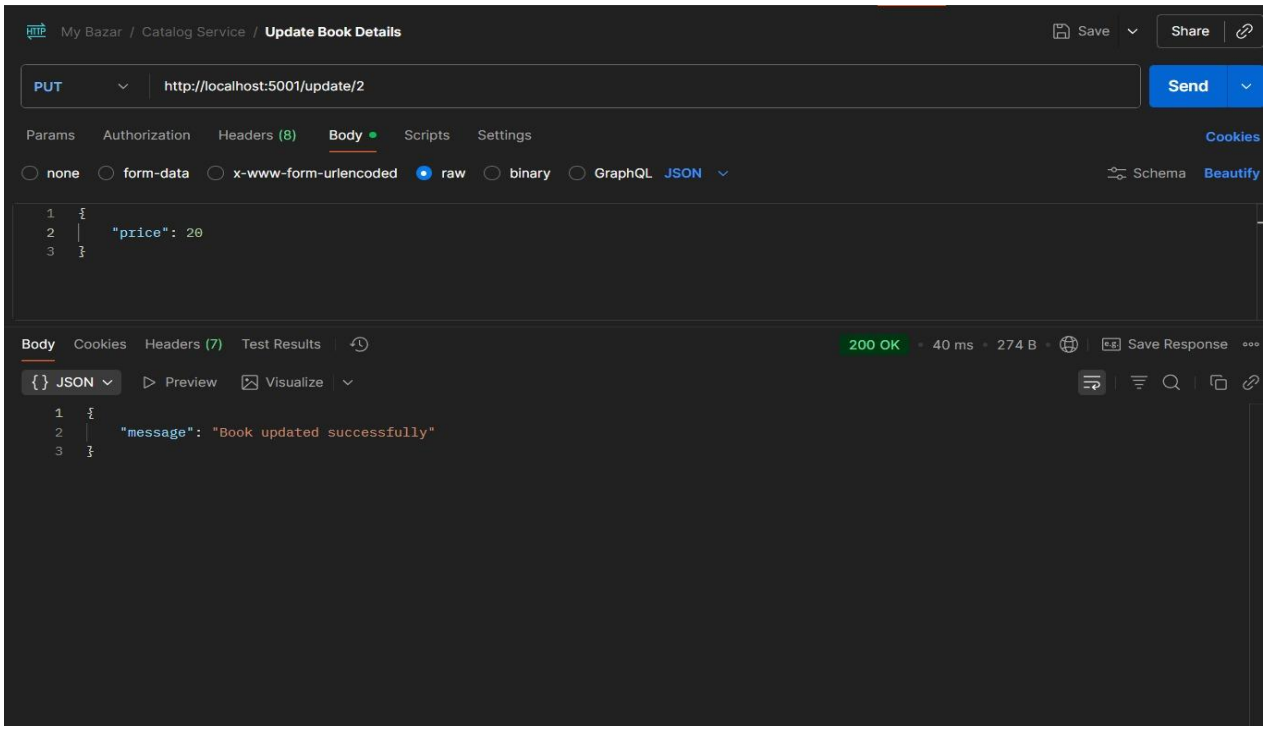
GET /info/:id → Get book details



POST /reserve/:id → Reserve a book

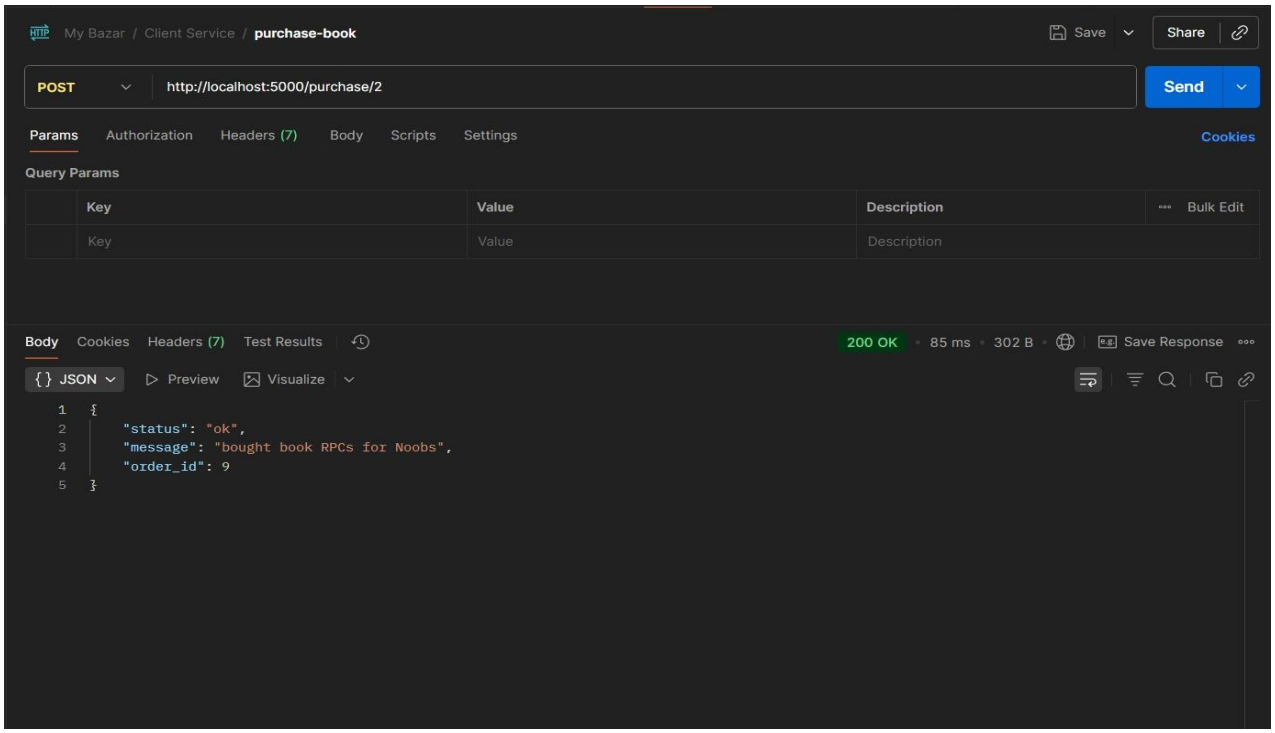


PUT /update/:id → Update book price/quantity

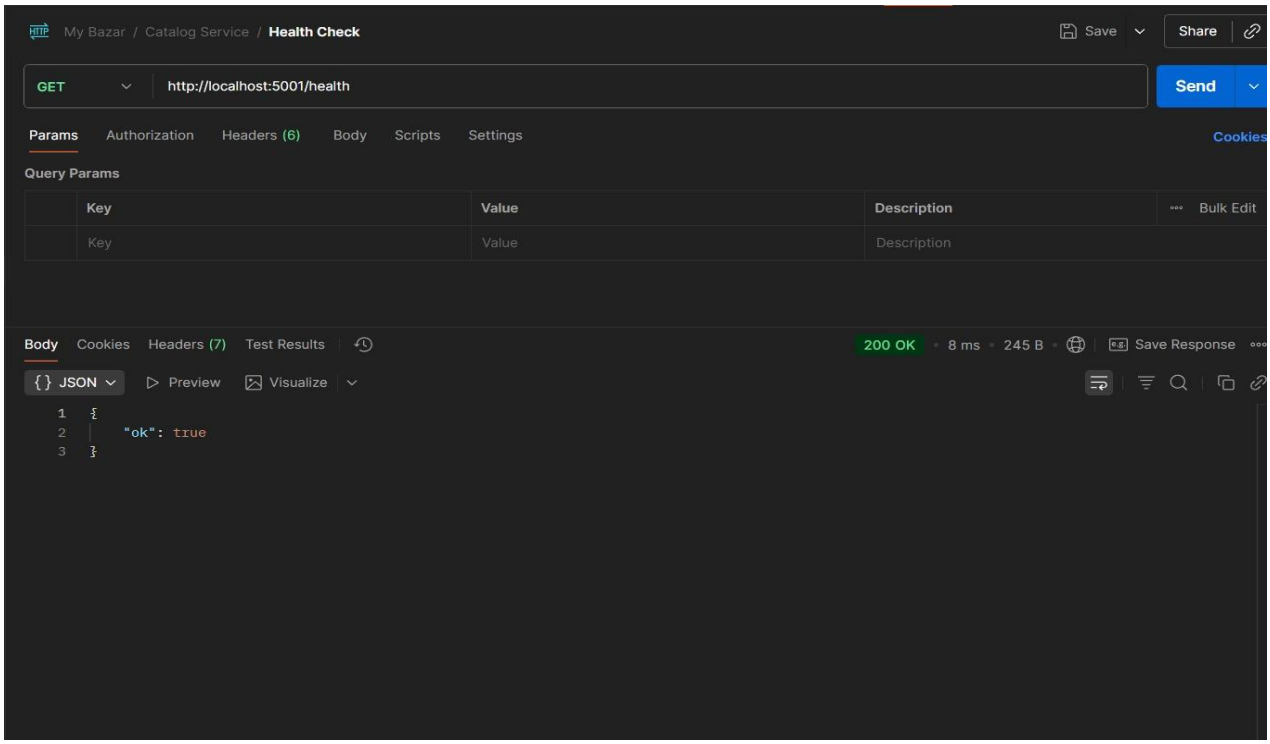


Order Service — http://localhost:5002

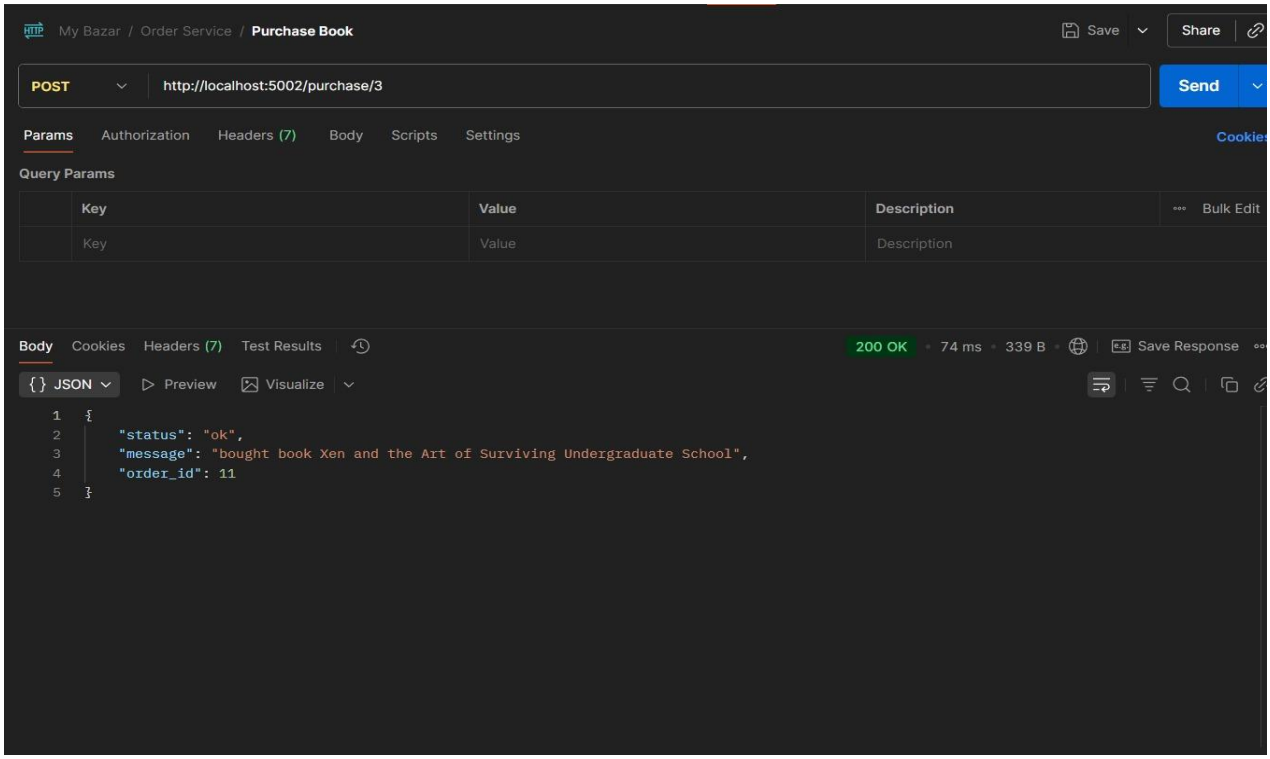
POST /purchase/:id → Purchase and store order



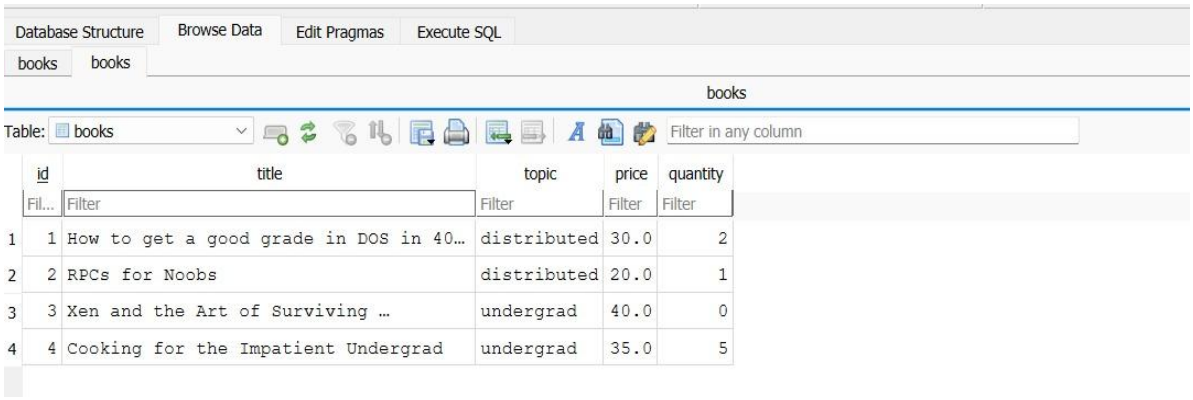
GET /health → Health check



If the user want to purcash a book :

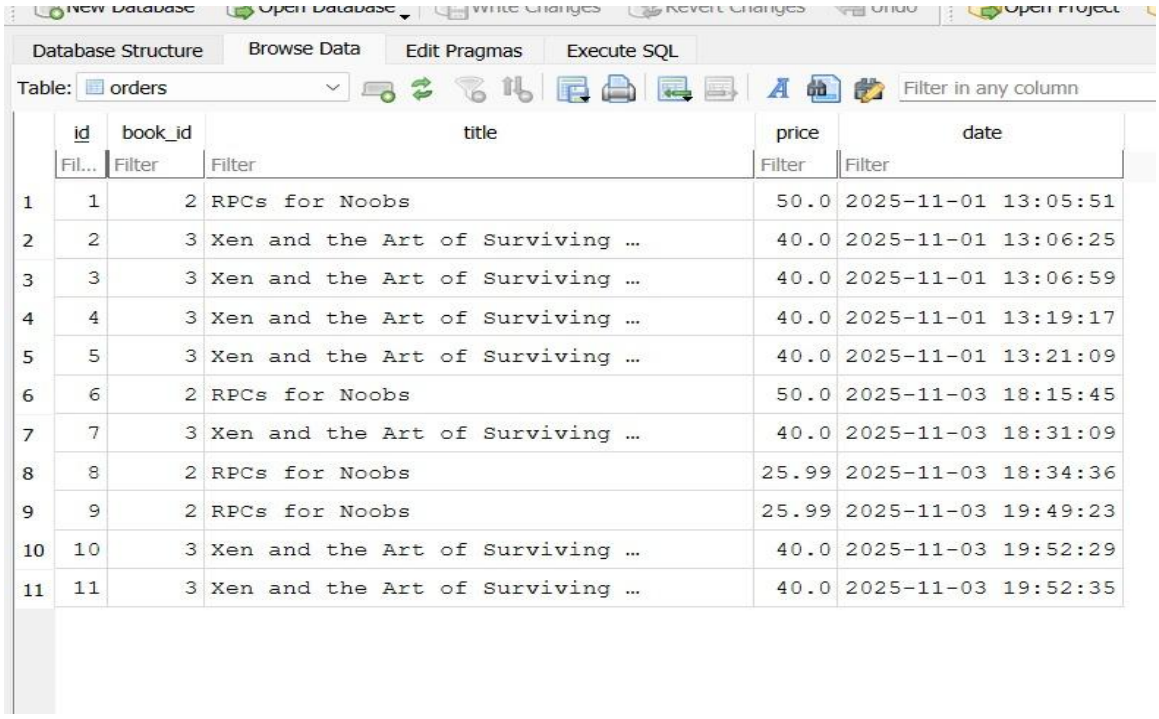


The quantity decrease by 1 :



	id	title	topic	price	quantity
	Fil...	Filter	Filter	Filter	Filter
1	1	How to get a good grade in DOS in 40…	distributed	30.0	2
2	2	RPCs for Noobs	distributed	20.0	1
3	3	Xen and the Art of Surviving …	undergrad	40.0	0
4	4	Cooking for the Impatient Undergrad	undergrad	35.0	5

And this is the order table that contains all the order :



	id	book_id	title	price	date
	Fil...	Filter	Filter	Filter	Filter
1	1	2	RPCs for Noobs	50.0	2025-11-01 13:05:51
2	2	3	Xen and the Art of Surviving …	40.0	2025-11-01 13:06:25
3	3	3	Xen and the Art of Surviving …	40.0	2025-11-01 13:06:59
4	4	3	Xen and the Art of Surviving …	40.0	2025-11-01 13:19:17
5	5	3	Xen and the Art of Surviving …	40.0	2025-11-01 13:21:09
6	6	2	RPCs for Noobs	50.0	2025-11-03 18:15:45
7	7	3	Xen and the Art of Surviving …	40.0	2025-11-03 18:31:09
8	8	2	RPCs for Noobs	25.99	2025-11-03 18:34:36
9	9	2	RPCs for Noobs	25.99	2025-11-03 19:49:23
10	10	3	Xen and the Art of Surviving …	40.0	2025-11-03 19:52:29
11	11	3	Xen and the Art of Surviving …	40.0	2025-11-03 19:52:35

Note : When the user want to purcash book and the quantity for the book = 0 the user can’t do this and the message “ Item out of stock “ appear for him

Conclusion

This project deepened our understanding of microservices, inter-process communication, and container orchestration. It also provided hands-on experience with backend design and API testing.