Faculty of Engineering and Information Technology

Computer Engineering department

Distributed Operation Systems

10636456

Dr. Samer Arandi

First Semester, 2023/2024

Multi-Tier Online Book Store Lab2

Team Members:

Noor Aldeen Muneer Abu Shehadeh

Momen Hasan Owad Odeh

Multi-Tier Online Book Store Program Design Lab2

Overview

The Multi-Tier Online Book Store is designed to provide users with an online platform for add book, update book information, purchasing, get information, and searching for books. The program consists of three main components: the Catalog Server, the Order Server, and the Frontend Server. These components work together to create a functional online bookstore.

How It Works

1. Catalog Server

The Catalog Server is responsible for managing book information and providing it to the system. Key functionalities include:

- Query book information: Users can request book details by book id. The Catalog Server retrieves this information from an SQLite database and returns it as a JSON response.

- Search books by Topic: Users can search for books by specifying a topic. The Catalog Server queries the database for books with matching topics and returns the results.

- Purchase book: When users decide to purchase a book, the Frontend Server communicates with the Order Server, which then communicates with the Catalog API to check book availability and update the database accordingly.

- Add new book: User can insert new book to the library by enter book information.

- Update book information: User can update a book information by enter book id.

2. Order Server

The Order Server handles book purchases and coordinates with the Catalog Server to ensure that books are available. Its primary functionality is:

- Purchase books: The Order Server receives purchase requests from the Frontend Server. It verifies book availability and updates the database by send request to Catalog Server, then save order in Orders database in Order Server, and returns the purchase status to the Frontend.

3. Frontend Server

The Frontend Server serves as the user interface for the online book store. Users can interact with the system through a web-based application that offers the following:

- Query book information: Users can look up book details by book id, which triggers a GET request to the Catalog Server.

- Search books by topic: Users can search for books by topic, and the Frontend Server communicates with the Catalog Server to retrieve matching book data.

- Purchase books: When users choose to purchase a book, the Frontend Server communicates with the Order Server to initiate the purchase.

- Add new book: User can insert new book to the library by enter book information then the Frontend Server send POST request to Catalog Server then return the response that received from Catalog Server.

- Update book information: User can update a book information by enter book id then the Frontend Server send PUT request to Catalog Server then return the response that received from Catalog Server.

Replication:

We create replicas for each backend server (catalog and order) to reduce the load on the servers. We distribute requests among the servers using a load balancing algorithm (round-robin).

Consistency:

The replicated backend servers should be consistent (have the same data). Therefore, when a server receives a request that changes the data, it should update the data on the other replicas.

Caching:

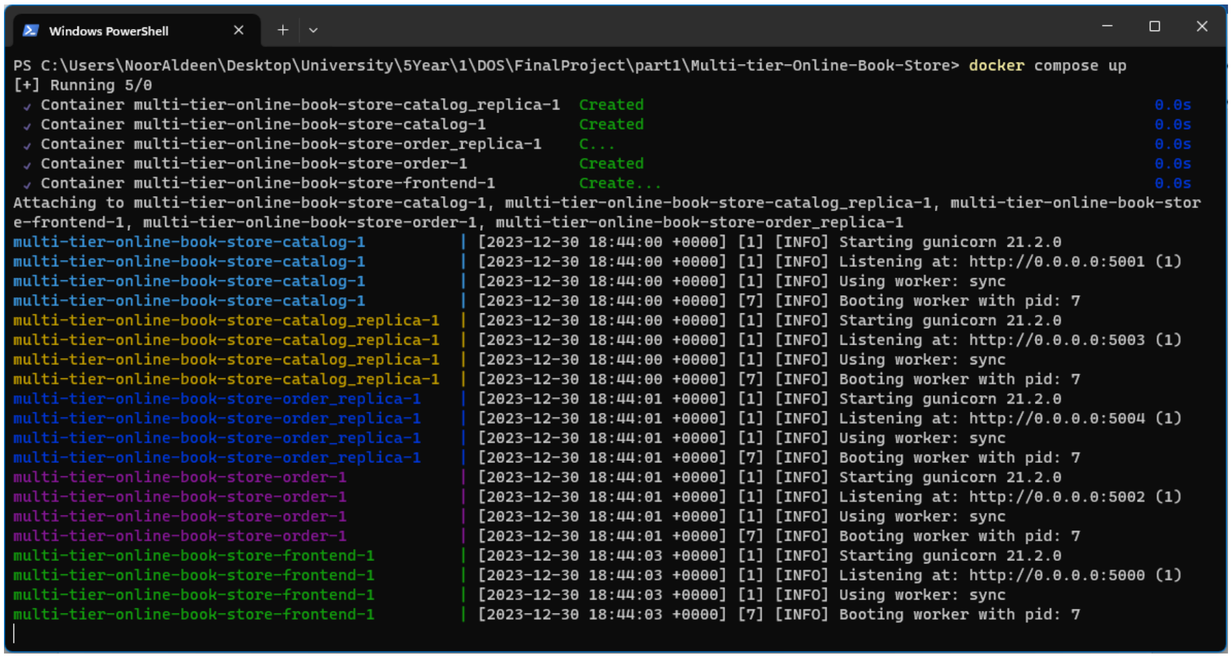
To enhance the performance of GET requests, we have implemented a caching mechanism that stores data based on a key. When a user sends a GET request, we check the cache for the corresponding information. If the information is found, we return it directly to the user. If it doesn't exist in the cache, we send a request to the backend, then when receive response from backend save the data to the cache, and then return it to the user.

Design tradeoffs:

1. Consistency Challenges: Achieving and maintaining consistency among replicated copies of data can be challenging. Synchronization delays between replicas result in eventual consistency.
2. Increased Complexity: Replication introduces additional complexity to the system. Managing and coordinating updates across multiple replicas require sophisticated mechanisms, and this complexity can affect system design, implementation, and maintenance.
3. Network Overhead: Maintaining consistency between replicas often involves communication and coordination among distributed components. This can lead to increased network traffic and latency, especially in scenarios where frequent updates are being made.
4. Storage Overhead: Replicating data across multiple nodes requires additional storage space. While this may improve read performance, it comes at the cost of increased storage requirements, which may be a concern in resource constrained environments.

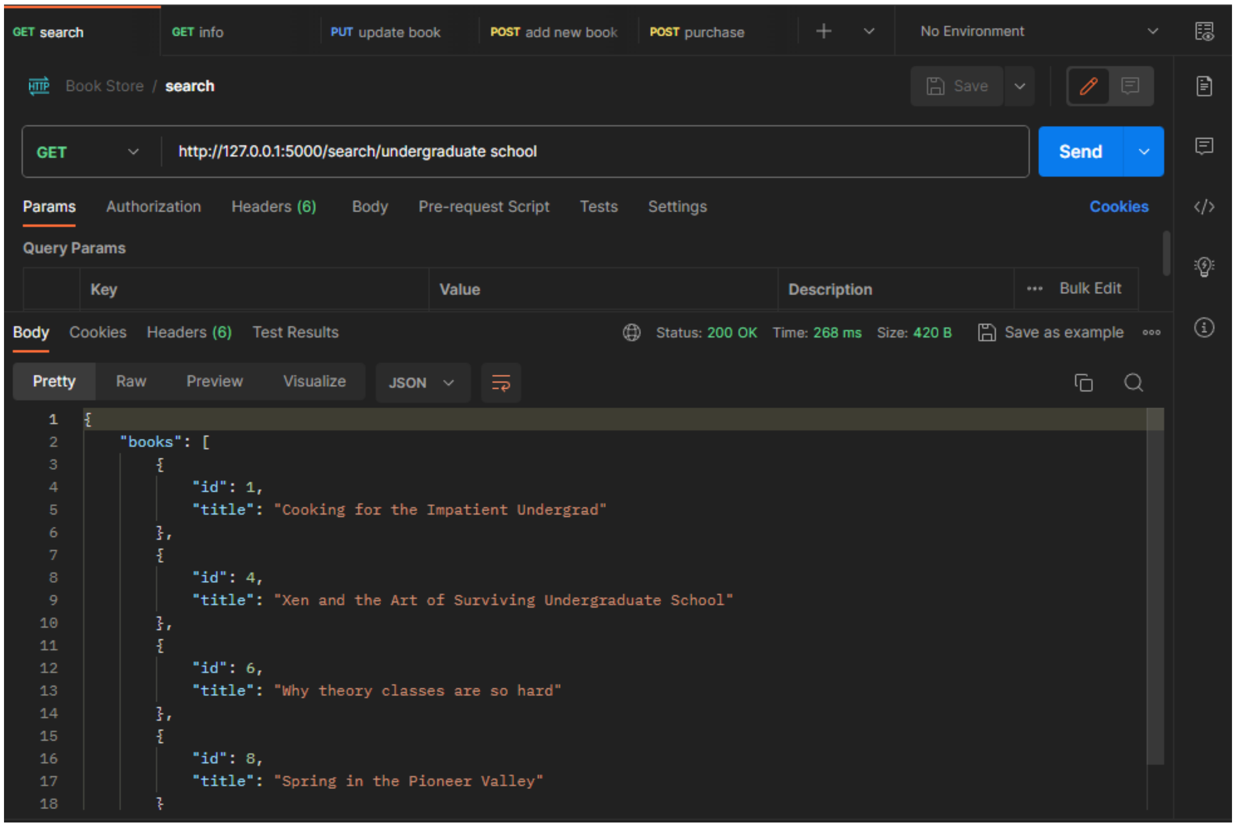
How run it and what is the output:

We write a docker files for each server then write docker compose file to manage all containers then when need to run servers write docker compose up.



```
PS C:\Users\NoorAldeen\Desktop\University\5Year\1\DOS\FinalProject\part1\Multi-tier-Online-Book-Store> docker compose up
[+] Running 5/0
 ✓ Container multi-tier-online-book-store-catalog_replica-1  Created                                           0.0s
 ✓ Container multi-tier-online-book-store-catalog-1          Created                                           0.0s
 ✓ Container multi-tier-online-book-store-order_replica-1    C...                                              0.0s
 ✓ Container multi-tier-online-book-store-order-1            Created                                           0.0s
 ✓ Container multi-tier-online-book-store-frontend-1         Create...                                         0.0s
Attaching to multi-tier-online-book-store-catalog-1, multi-tier-online-book-store-catalog_replica-1, multi-tier-online-book-stor
e-frontend-1, multi-tier-online-book-store-order-1, multi-tier-online-book-store-order_replica-1
multi-tier-online-book-store-catalog-1          | [2023-12-30 18:44:00 +0000] [1] [INFO] Starting gunicorn 21.2.0
multi-tier-online-book-store-catalog-1          | [2023-12-30 18:44:00 +0000] [1] [INFO] Listening at: http://0.0.0.0:5001 (1)
multi-tier-online-book-store-catalog-1          | [2023-12-30 18:44:00 +0000] [1] [INFO] Using worker: sync
multi-tier-online-book-store-catalog-1          | [2023-12-30 18:44:00 +0000] [7] [INFO] Booting worker with pid: 7
multi-tier-online-book-store-catalog_replica-1  | [2023-12-30 18:44:00 +0000] [1] [INFO] Starting gunicorn 21.2.0
multi-tier-online-book-store-catalog_replica-1  | [2023-12-30 18:44:00 +0000] [1] [INFO] Listening at: http://0.0.0.0:5003 (1)
multi-tier-online-book-store-catalog_replica-1  | [2023-12-30 18:44:00 +0000] [1] [INFO] Using worker: sync
multi-tier-online-book-store-catalog_replica-1  | [2023-12-30 18:44:00 +0000] [7] [INFO] Booting worker with pid: 7
multi-tier-online-book-store-order_replica-1    | [2023-12-30 18:44:01 +0000] [1] [INFO] Starting gunicorn 21.2.0
multi-tier-online-book-store-order_replica-1    | [2023-12-30 18:44:01 +0000] [1] [INFO] Listening at: http://0.0.0.0:5004 (1)
multi-tier-online-book-store-order_replica-1    | [2023-12-30 18:44:01 +0000] [1] [INFO] Using worker: sync
multi-tier-online-book-store-order_replica-1    | [2023-12-30 18:44:01 +0000] [7] [INFO] Booting worker with pid: 7
multi-tier-online-book-store-order-1            | [2023-12-30 18:44:01 +0000] [1] [INFO] Starting gunicorn 21.2.0
multi-tier-online-book-store-order-1            | [2023-12-30 18:44:01 +0000] [1] [INFO] Listening at: http://0.0.0.0:5002 (1)
multi-tier-online-book-store-order-1            | [2023-12-30 18:44:01 +0000] [1] [INFO] Using worker: sync
multi-tier-online-book-store-order-1            | [2023-12-30 18:44:01 +0000] [7] [INFO] Booting worker with pid: 7
multi-tier-online-book-store-frontend-1         | [2023-12-30 18:44:03 +0000] [1] [INFO] Starting gunicorn 21.2.0
multi-tier-online-book-store-frontend-1         | [2023-12-30 18:44:03 +0000] [1] [INFO] Listening at: http://0.0.0.0:5000 (1)
multi-tier-online-book-store-frontend-1         | [2023-12-30 18:44:03 +0000] [1] [INFO] Using worker: sync
multi-tier-online-book-store-frontend-1         | [2023-12-30 18:44:03 +0000] [7] [INFO] Booting worker with pid: 7
```

- **Search by topic:**

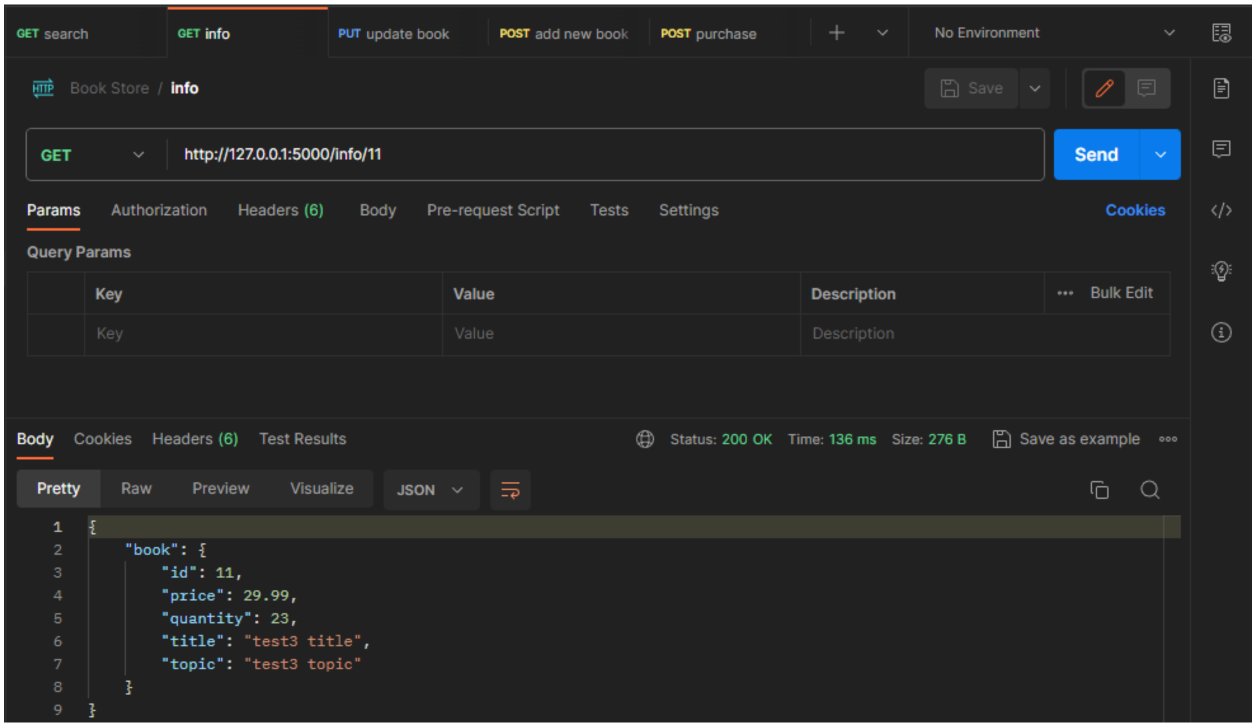


GET http://127.0.0.1:5000/search/undergraduate school

Status: 200 OK Time: 268 ms Size: 420 B

```
{
    "books": [
        {
            "id": 1,
            "title": "Cooking for the Impatient Undergrad"
        },
        {
            "id": 4,
            "title": "Xen and the Art of Surviving Undergraduate School"
        },
        {
            "id": 6,
            "title": "Why theory classes are so hard"
        },
        {
            "id": 8,
            "title": "Spring in the Pioneer Valley"
        }
```

- **Info by book id:**

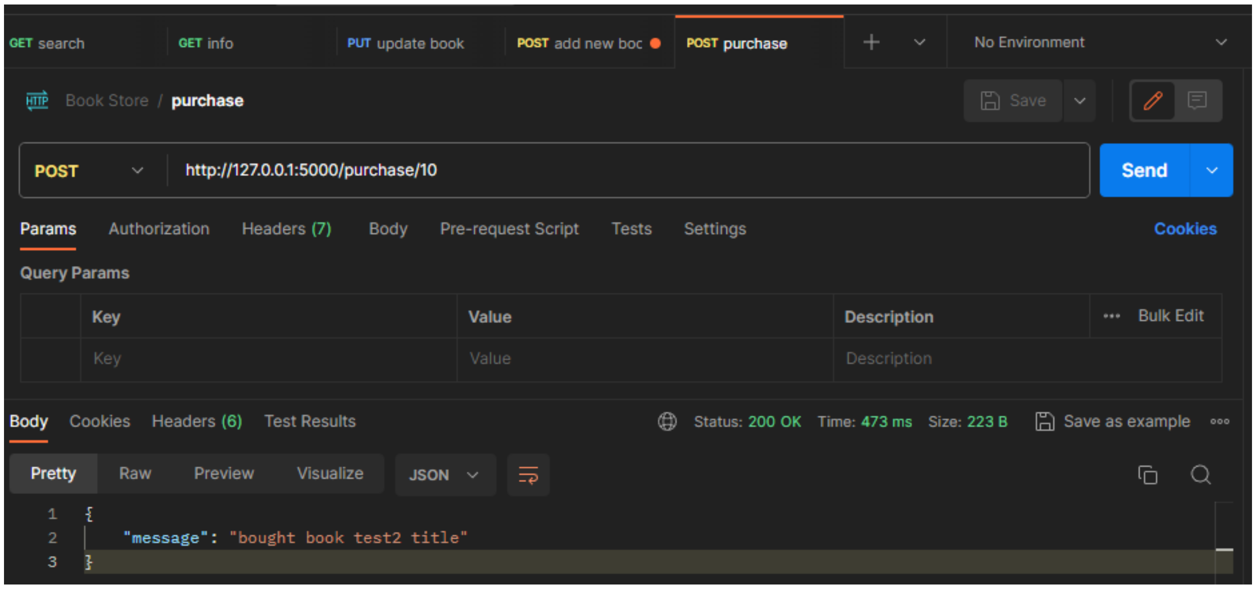


GET http://127.0.0.1:5000/info/11

Status: 200 OK Time: 136 ms Size: 276 B

```
{
    "book": {
        "id": 11,
        "price": 29.99,
        "quantity": 23,
        "title": "test3 title",
        "topic": "test3 topic"
    }
}
```

- **Purchase by book id:**

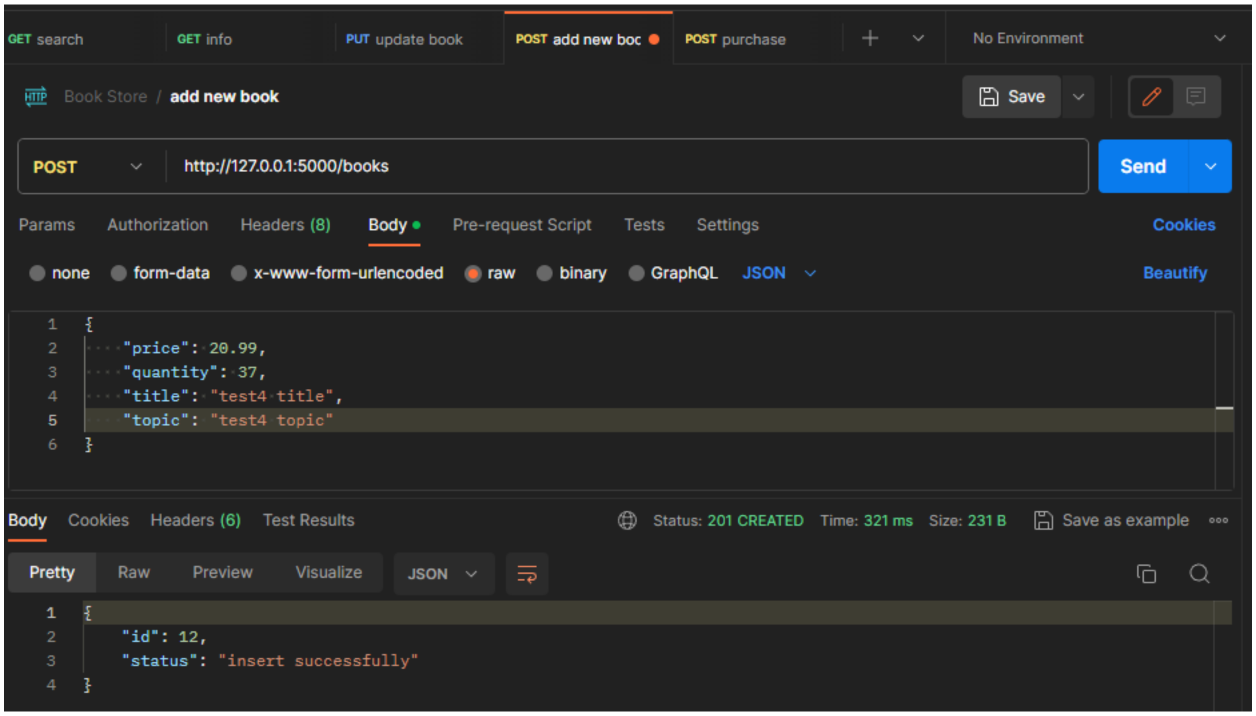


POST http://127.0.0.1:5000/purchase/10

Status: 200 OK Time: 473 ms Size: 223 B

```
{
    "message": "bought book test2 title"
}
```

- **Add new book:**



POST http://127.0.0.1:5000/books

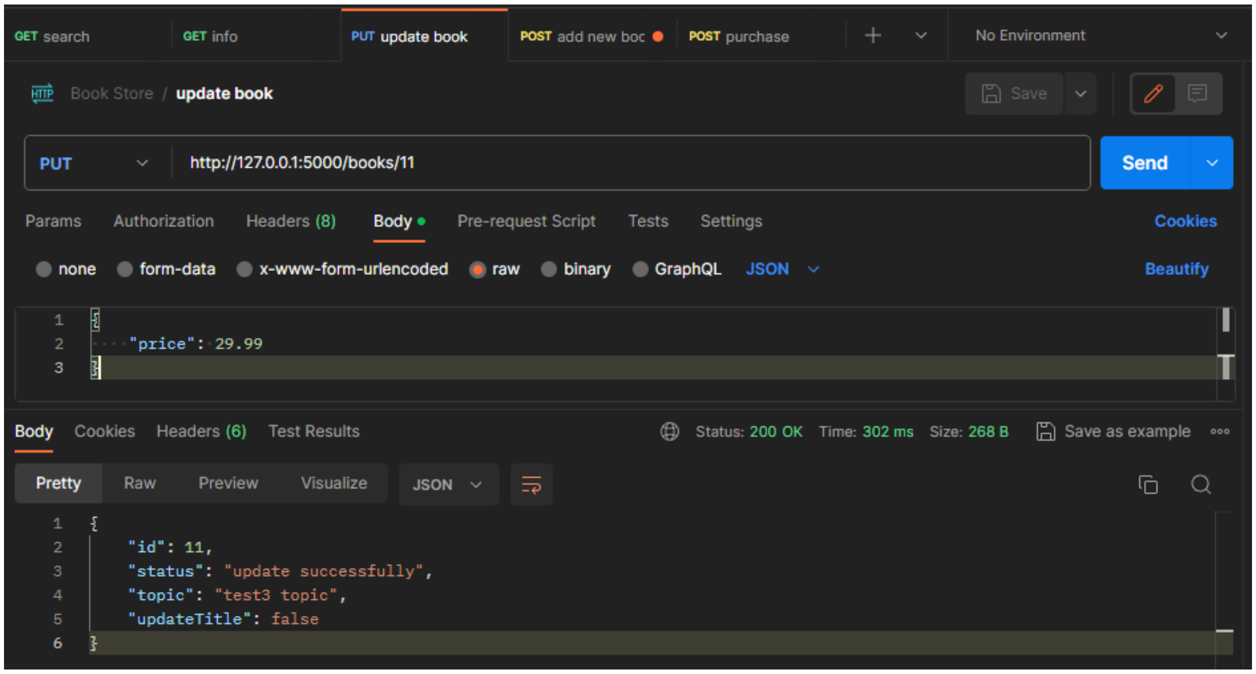
Body: raw JSON

```
{
    "price": 20.99,
    "quantity": 37,
    "title": "test4 title",
    "topic": "test4 topic"
}
```

Status: 201 CREATED Time: 321 ms Size: 231 B

```
{
    "id": 12,
    "status": "insert successfully"
}
```

- **Update book info:**



PUT http://127.0.0.1:5000/books/11

Body: raw JSON

```
{
    "price": 29.99
}
```

Status: 200 OK Time: 302 ms Size: 268 B

```
{
    "id": 11,
    "status": "update successfully",
    "topic": "test3 topic",
    "updateTitle": false
}
```

Experimental Evaluation and Measurements:

	search without cache	search with cache	Info without cache	Info with cache	Purchase	Add	Update
	268	49	136	15	473	321	302
	72	26	40	18	472	226	149
	83	22	87	13	316	130	234
	72	26	81	26	448	222	172
	54	22	33	20	417	169	100
	68	18	70	18	245	157	244
	71	13	41	23	268	218	86
	48	18	69	10	266	163	68
	64	9	89	28	259	154	138
	27	16	29	11	434	255	196
Average	**82.7**	**21.9**	**67.5**	**18.2**	**359.8**	**198.5**	**166.9**

- **How much caching help?** Reduce the response time for get requests.

- **What is the overhead of cache consistency operations?** Communication overhead includes the invalidation messages among cache nodes, exerting additional pressure on the server.

- **What is the latency of a subsequent request that sees a cache miss?** The latency of a subsequent request encountering a cache miss is higher than that of a cache hit. This is because, in the case of a cache miss, the system must request a data from server, resulting in increased retrieval time.