بسم الله الرحمن الرحيم

AN-NAJAH NATIONAL UNIVERSITY

FACULTY OF ENGINEERING AND INFORMATION TECHNOLOGY

Computer Engineering Department

Online Book Store (Bazar.com)

Part 2

Samah Sheeha & Eba Khalil

16 November 2024

Abstract

This report explains the upgrades we made to Bazar.com’s online bookstore to handle requests faster and more efficiently. The main goals were to add features like replication, caching, and consistency to reduce delays and handle more users at once. We split the system into smaller parts, called microservices, and used containers to make everything easier to deploy and manage. We tested our changes by measuring response times, especially how caching helped, and found that these adjustments made a big difference.

Introduction

As Bazar.com’s online bookstore grew in popularity, the system started to slow down when handling customer requests, leading to frustration for users. Our project focused on re-designing the system to make it faster and better at handling a large number of requests. To do this, we added techniques like replication (where we make multiple copies of servers to share the load), caching (storing frequently used data temporarily for quick access), and consistency (keeping all the data in sync).

We also containerized each part of the system using Docker, which makes deployment smoother and scaling easier if demand increases. We set up REST APIs to help the different parts of the system communicate smoothly, as these are widely used in web development. Once everything was in place, we tested the system to see how much the upgrades boosted performance and analyzed the results to understand the improvements.

Project Overview and Key Components

1. **Replication and Caching**
   - **Replication**: We created copies of the order and catalog servers so that requests could be spread across them. This way, no single server is overwhelmed, and response times improve.
   - **Caching**: By adding an in-memory cache, we could store frequently requested information, like book details, locally instead of querying the catalog server every time. This reduces the time it takes to serve repeat requests and speeds up response times for users.
2. **Load Balancing**
   A load balancer was added to distribute incoming requests evenly across the replicated servers. We used methods like round-robin (where requests take turns going to each server) or choosing the server with the least load, so requests are handled quickly and efficiently.

3. **Cache Consistency**
   To keep cached data accurate, we added a way for the system to update or delete outdated information from the cache whenever there's a change in the database. This ensures that users see the latest data, even if they're getting it from the cache.

4. **Docker Containerization**
   Each part of the system, including the frontend server, cache, and backend servers, is packed into a Docker container. This setup makes it easier to deploy, run, and scale the system on different machines or cloud environments.

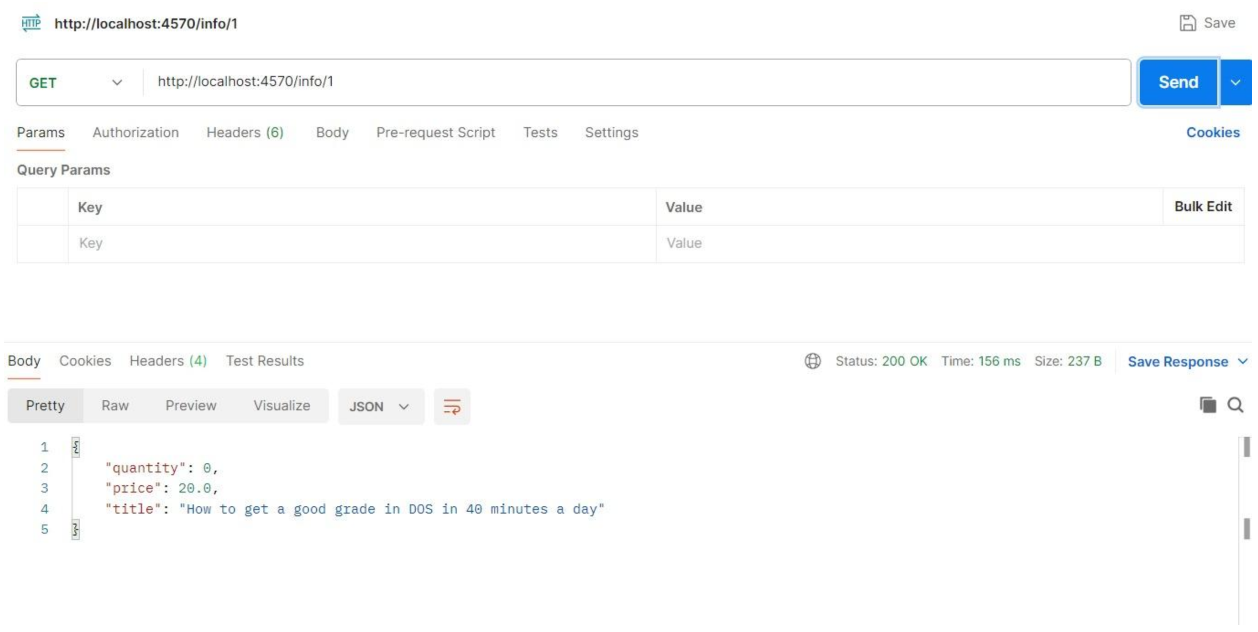
5. **Performance Testing**
   We ran experiments to check how well the system performs by:

   - Measuring the average response time for common actions like searching or buying books, both with and without caching, to see how much caching helps.

   - Testing how cache consistency affects performance by triggering cache invalidation (deleting old data) and measuring the response time when the system has to handle cache misses.

Practical steps

This is a screenshot when do " get info " for the first time

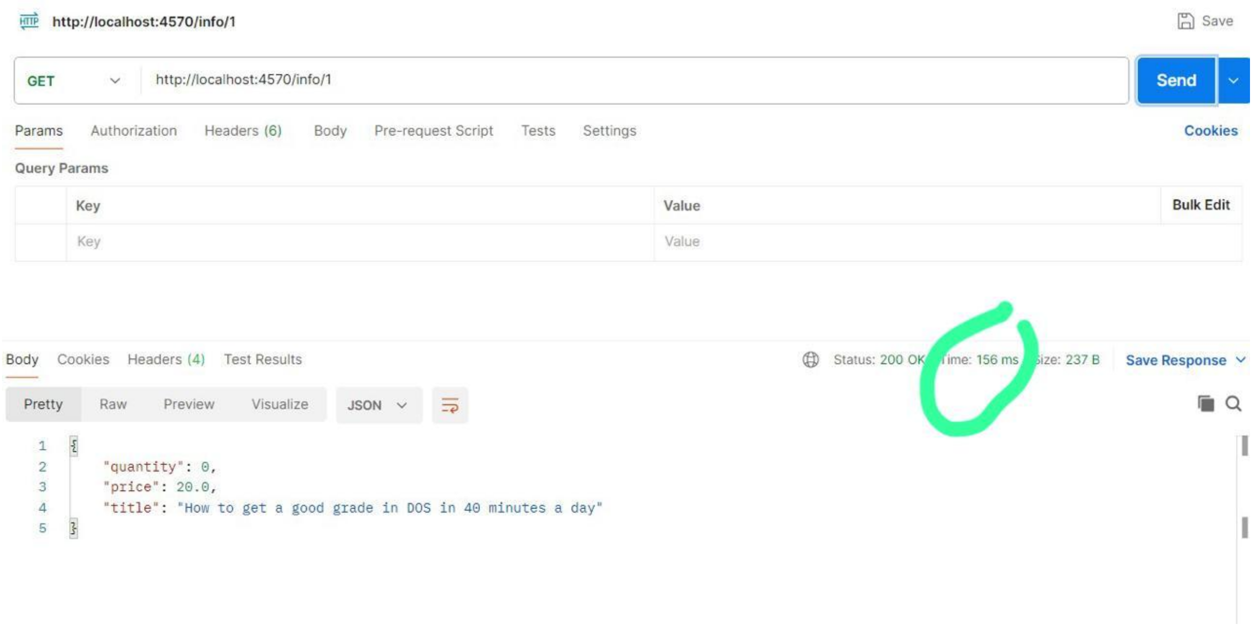


GET http://localhost:4570/info/1

Status: 200 OK Time: 156 ms Size: 237 B

```
{
    "quantity": 0,
    "price": 20.0,
    "title": "How to get a good grade in DOS in 40 minutes a day"
}
```

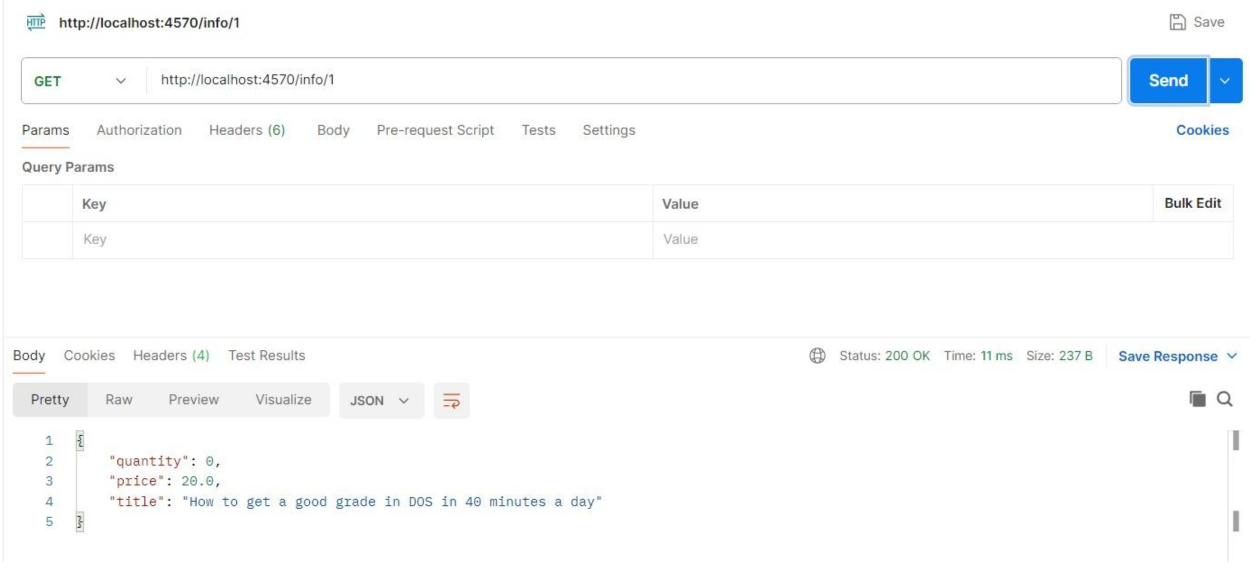
The time :-


GET http://localhost:4570/info/1

Status: 200 OK Time: 156 ms Size: 237 B

```
{
    "quantity": 0,
    "price": 20.0,
    "title": "How to get a good grade in DOS in 40 minutes a day"
}
```

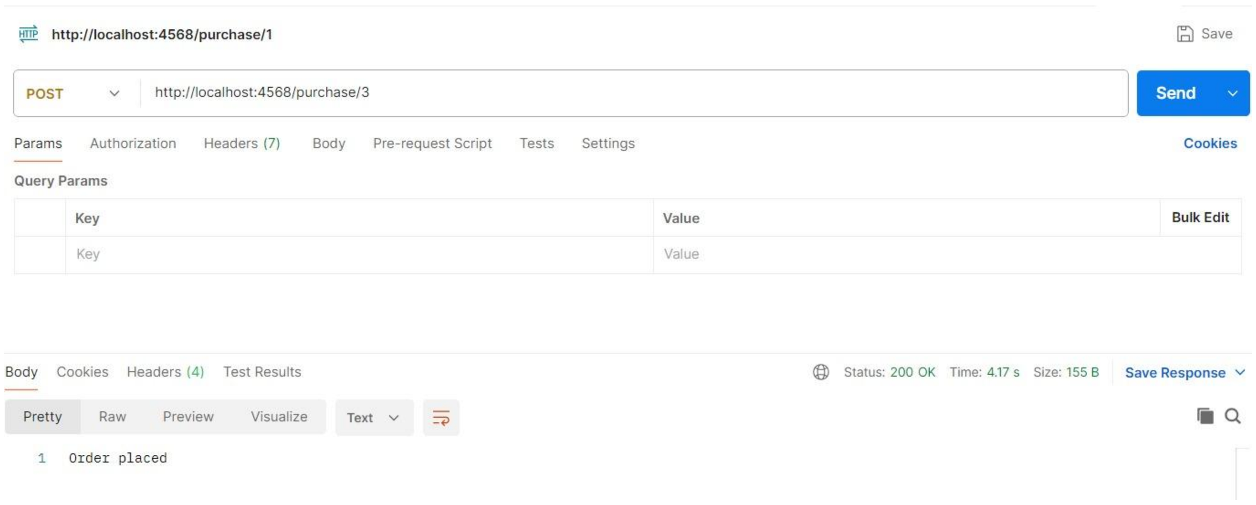
Screenshot of the second time I did " get info ", it took less time


GET http://localhost:4570/info/1

Status: 200 OK Time: 11 ms Size: 237 B

```
{
    "quantity": 0,
    "price": 20.0,
    "title": "How to get a good grade in DOS in 40 minutes a day"
}
```

This screenshot of the purchase takes a good amount of time compared to the first part of the project.



http://localhost:4568/purchase/3

Status: 200 OK Time: 4.17 s Size: 155 B

```
Order placed
```

In the first part, the purchase took 150 ms, while in the second part here it became 4.1 s

Summary: Here in the second part we reduced the time for the search and Get Information process, but the purchase will take longer because it will be modified on all replica.

Conclusion

This project showed that adding replication, caching, and consistency measures really helped Bazar.com's online bookstore handle more traffic and respond to users faster. The system now uses load balancing and caching to reduce delays and manage resources better. By containerizing the parts of the system, we also made it easier to deploy and scale if traffic increases. Our testing confirmed that these upgrades significantly improved performance, especially when it came to loading popular items from the catalog. In the future, we could consider expanding the cache dynamically based on demand and experimenting with other load-balancing methods to make the system even more resilient.