In this lab, we added replication and caching to the existing project to improve request processing latency.
First,We add an in-memory cache to two servers(order and catalog).the cache takes request from user and if it has the item it returns to user,if not,it sends request to server to get the data item and when it takes it ,cache store the item and then send to user.
In another time,if the user sends a new request the cache takes the request and returns the item data.
Then,for replication we replicated the database table to make multi connection if the first server that takes a request is busy the second server takes a request, and if the server breaks down there is a backup.

To deal with this replication,we use load balancing on a per-request basis or a per user-session basis,when a user sends a request it starts a session on every request and saves a session before serving the response.

We implement load balancing on a per user-session basis,when a user sends a request to buy a book the session starts that takes an incoming buy request and sends it to cache,and when a buy operation is done the server checks if the value in the cache is the same.

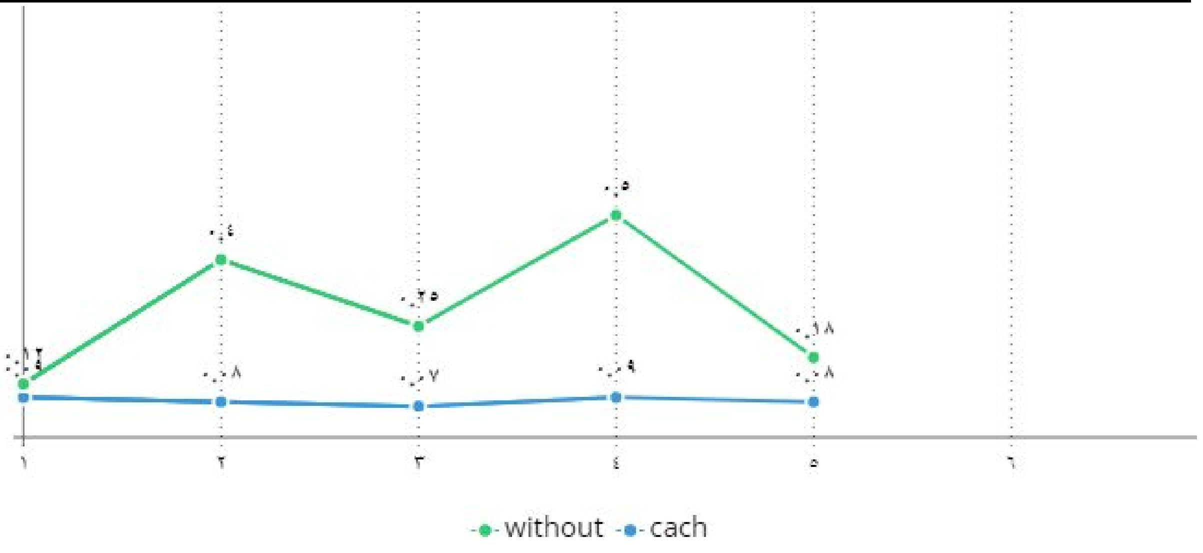
Experimental Evaluation and Measurements:

Compute the average response time (query/buy) :
We take 5 time for different buy request
with cache=(0.09+0.07+0.08+0.09+0.08)/5=0.082
Without cach(0.12+0.40+0.25+0.50+0.18)/5=.29
We note that response time with cache small compared to without cache.
overhead of cache=3.54
latency =0.08

Number operation	Time
1	0.09
2	0.07
3	0.08

4	0.09
5	0.08

Number operation(without cache)	Time
1	0.12
2	0.40
3	0.25
4	0.50
5	0.18



How it works :

1-To use cache:
First ,we will need to install the predis/predis package version (1.1) and illuminate/redis package version (~5.1) via Composer.
For predis/predis :
Use this command:composer require predis/predis
For illuminate/redis :
composer require illuminate/redis

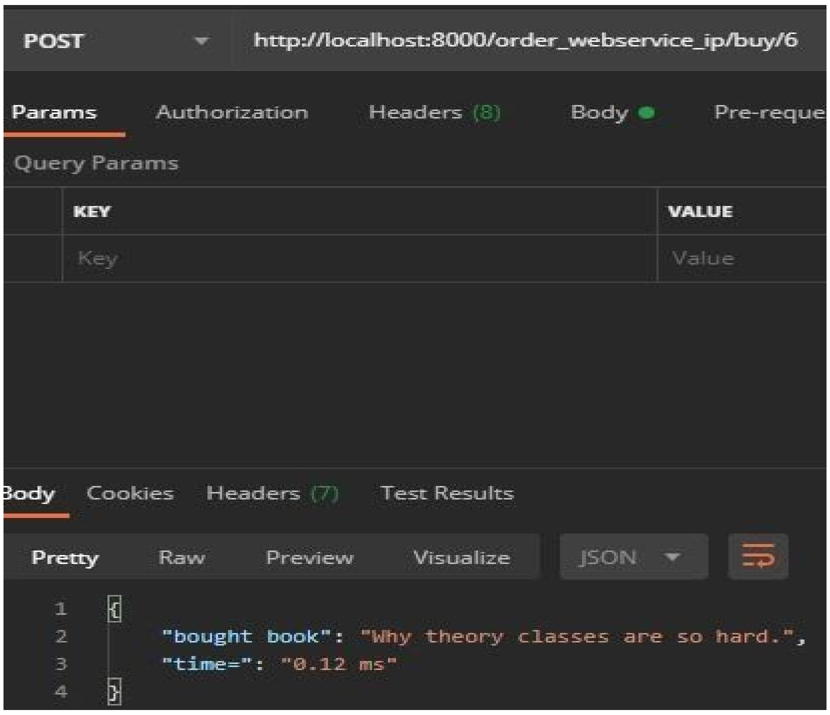
2-For the run project we used this command: php -S localhost:8000 -t public.

3-For the load balancing algorithm we used the round-robin algorithm,we downloaded this package:composer require illuminate/session(version 8.18).

4-Install a package to enable server push:
composer require jacobbennett/laravel-http2serverpush

Output

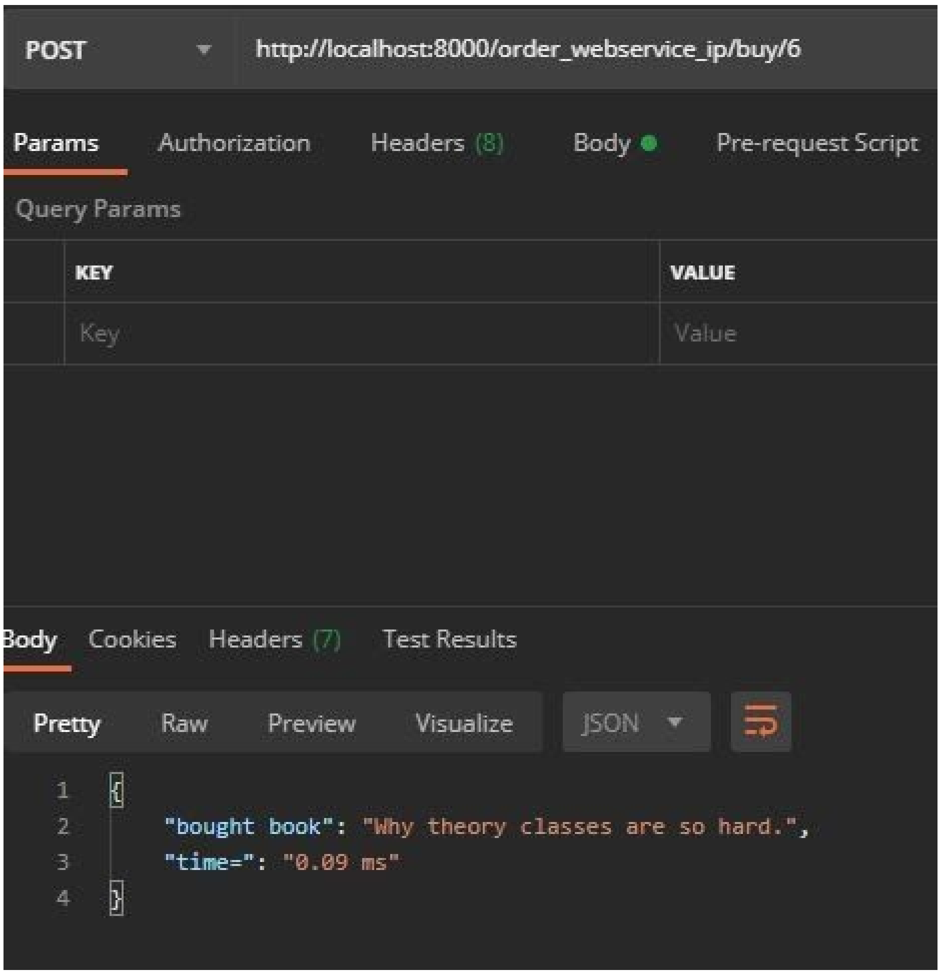
Buy server(without cache):



```
http://localhost:8000/order_webservice_ip/buy/6
```

```
{
    "bought book": "Why theory classes are so hard.",
    "time=": "0.12 ms"
}
```

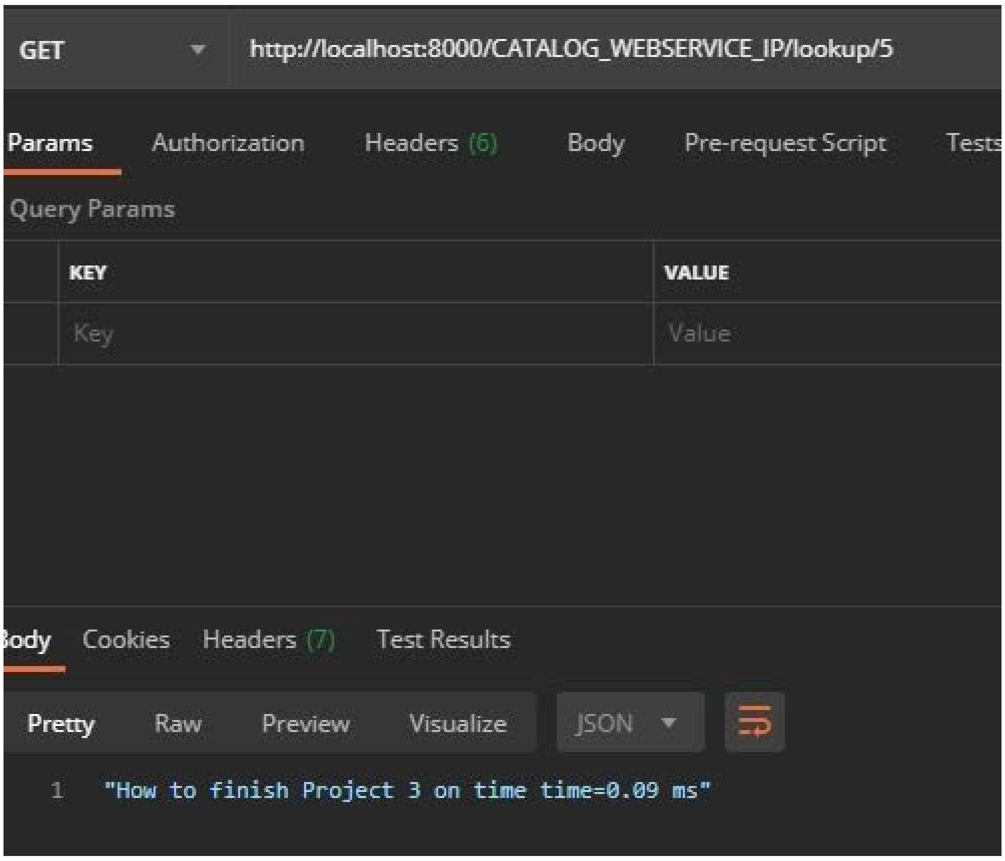
Buy with cache :



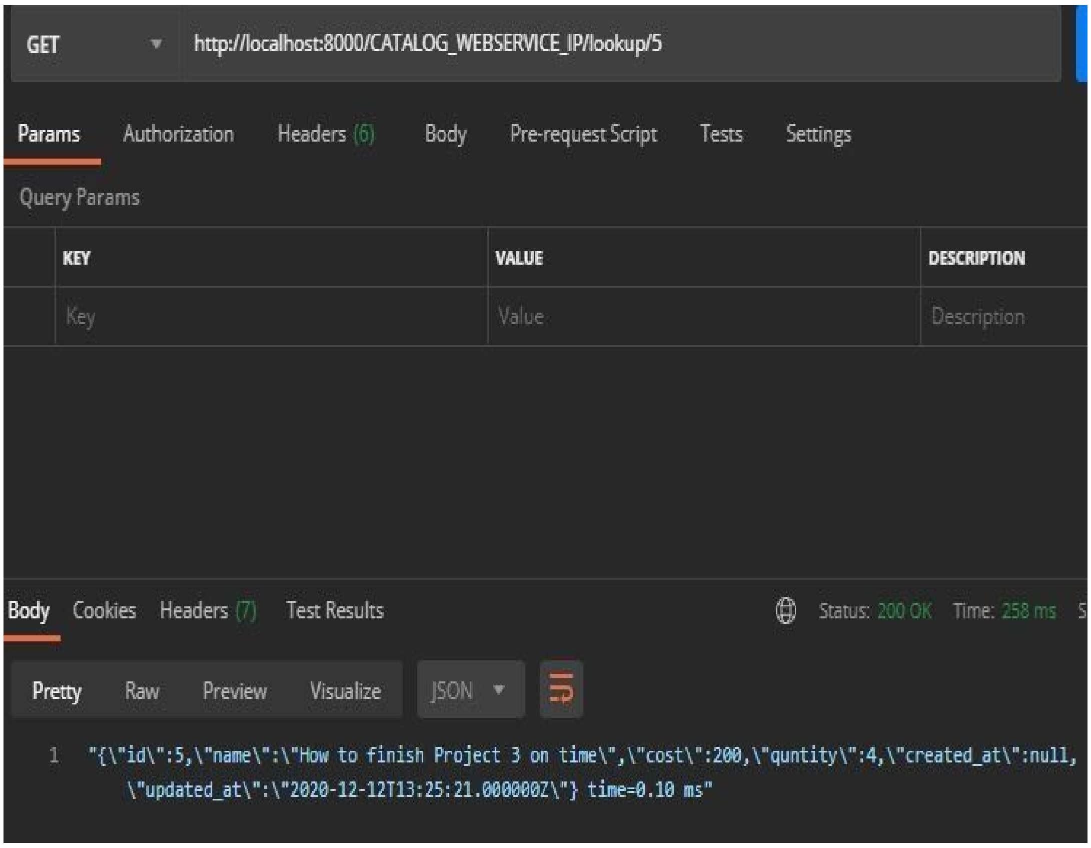
```
http://localhost:8000/order_webservice_ip/buy/6
```

```
{
    "bought book": "Why theory classes are so hard.",
    "time=": "0.09 ms"
}
```

Catalog server:
Lookup(without cache)

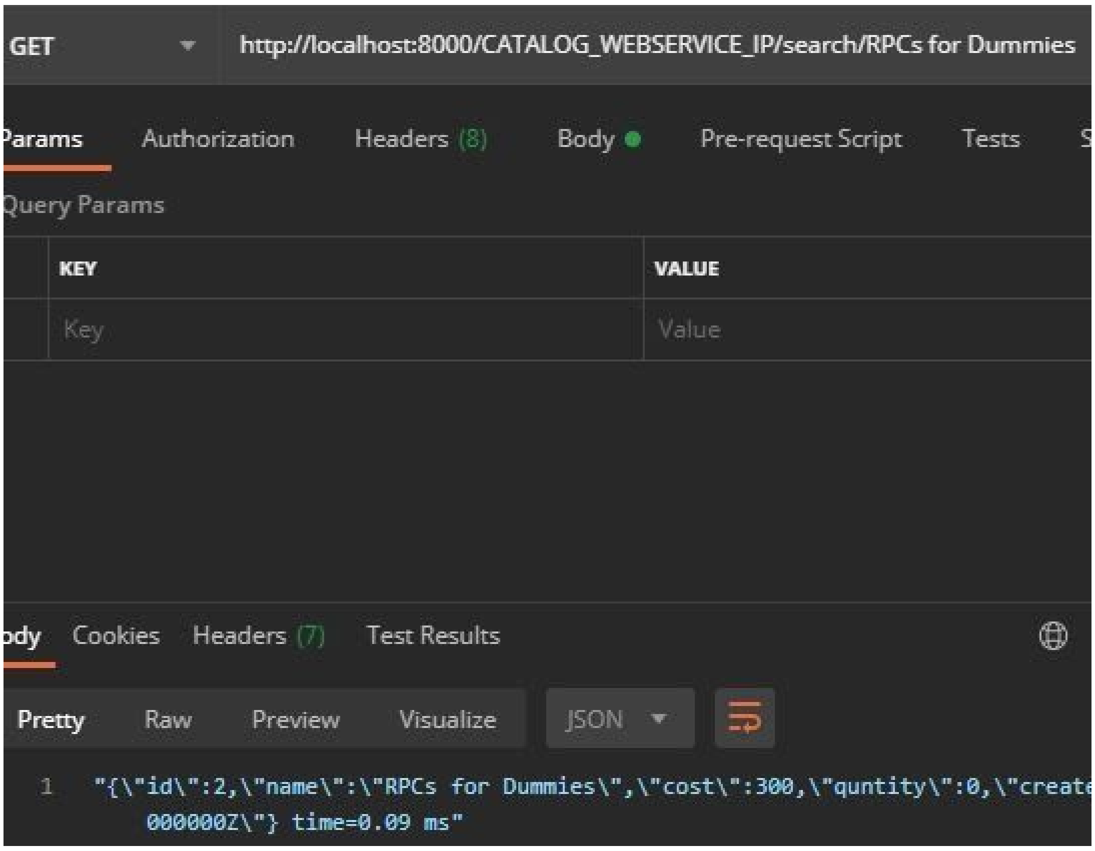

GET http://localhost:8000/CATALOG_WEBSERVICE_IP/lookup/5

```
"How to finish Project 3 on time time=0.09 ms"
```


GET http://localhost:8000/CATALOG_WEBSERVICE_IP/lookup/5

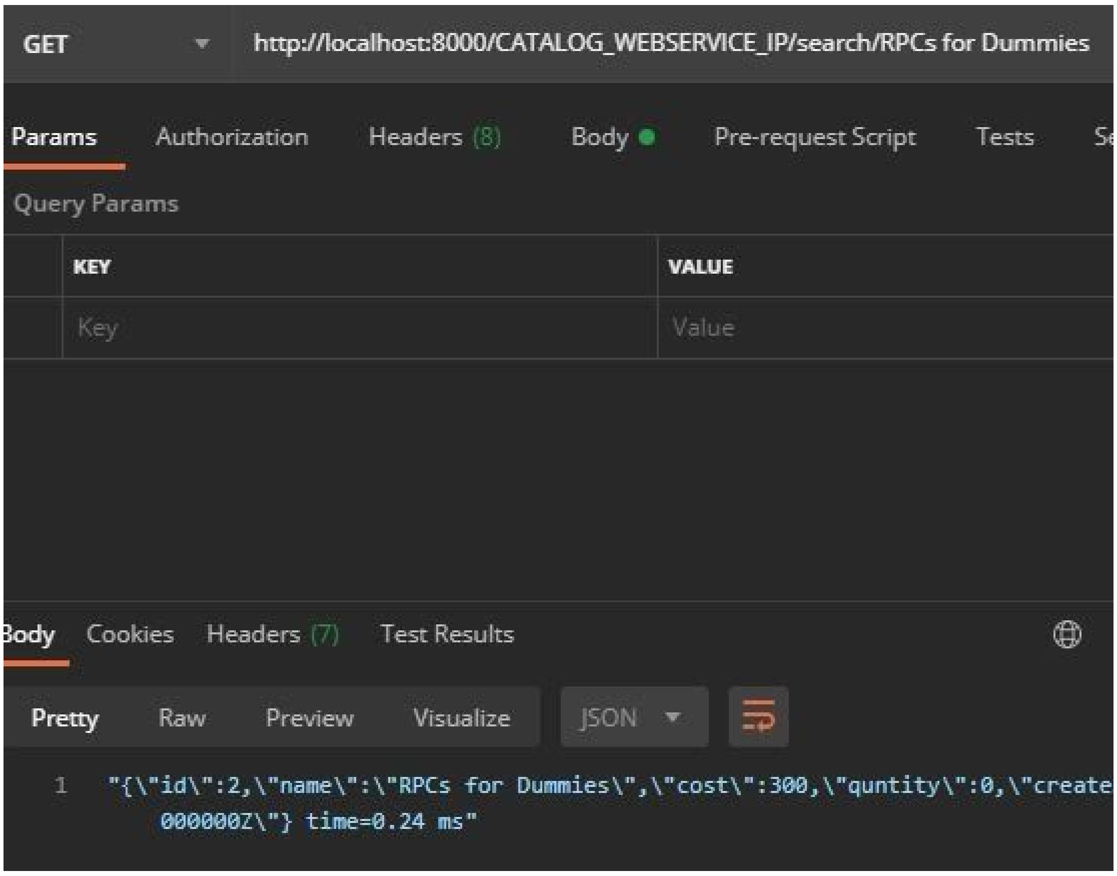
Status: 200 OK Time: 258 ms

```
"{\"id\":5,\"name\":\"How to finish Project 3 on time\",\"cost\":200,\"quntity\":4,\"created_at\":null,
   \"updated_at\":\"2020-12-12T13:25:21.000000Z\"} time=0.10 ms"
```



GET http://localhost:8000/CATALOG_WEBSERVICE_IP/search/RPCs for Dummies

```
"{\"id\":2,\"name\":\"RPCs for Dummies\",\"cost\":300,\"quntity\":0,\"create
    000000Z\"} time=0.09 ms"
```



GET http://localhost:8000/CATALOG_WEBSERVICE_IP/search/RPCs for Dummies

```
"{\"id\":2,\"name\":\"RPCs for Dummies\",\"cost\":300,\"quntity\":0,\"create
    000000Z\"} time=0.24 ms"
```

Github:
github.com/ruyaatatreh99/DosPro