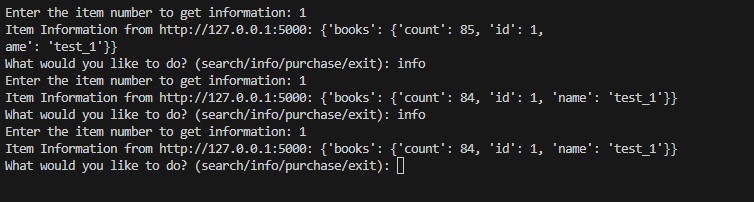
**Output part 2**

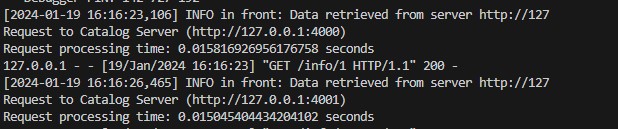
Here when user try to get info:



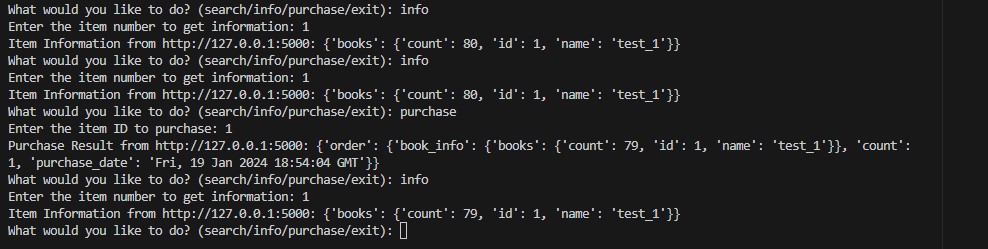
In the front server in the first time not find the value in the cache so get the data from server, then the value will be on the cache:



Here when the data should retrieve from server, using origin for the first time then using the replica in the second time (using round robin algorithm)



Here when the user choose purchase:



The data will be invalid on the cache then get the data from the server:



The response time before cache and after using it:

|  |  |  |
| --- | --- | --- |
|  | With cache | Without cache |
| search | Cache miss:  0.01784205436706543 seconds cache hit: 0.0 seconds | 0.0 seconds |
| info | First time (not be on the cache) “cache miss” : 0.012956380844116211 seconds  when be on the cache “cache hit”: 0.0010454654693603516 seconds | 0.005999088287353516 seconds |
| purchase | 0.0462191104888916 seconds | 0.05619502067565918 seconds |

utilizing a cache improves the performance of search and info operations significantly by reducing the time for cache hits. However, the impact on the purchase operation is less pronounced, with a moderate improvement in execution time.

But in our project the search operation without cache consistently takes 0.0 seconds. Therefore, the cache seems to introduce a small overhead in the case of a cache miss for the search operation. However, the benefit of faster cache hits can outweigh this overhead, especially when the data is frequently accessed and can be retrieved from the cache.