ACS - Assignment 1

Rikke Tophøj Heinemann
Marie Elkjær Rødsgaard
Rasmus Pallisgaard

December 3, 2024

1

a

The implementation of the `rateBooks` method address the all-or-nothing semantics, This can be seen since if on of the books the the book rating set does not have a valid ISBN number then the function throws and exception. The same if an invalid rating is given then it again throws an exception. So either all the books are rated or none.

The implementation of the `getTopRatedBooks` method adresses the all-or-nothing semantic by either getting the k number of top rated books or no books. The k must be smaller than the number of books in the given set and must be positive. by sorting the ratings and getting the k top books a list is found.

The implementation of the `getBooksInDemand` method addresses the all-or-nothing semantic trivially, as obtaining the list of books in demand is constructed via a `filter->map->collect` into the bookMap variable after checking for potential errors due to null references. We tested the ability to find in demand books, whether there be none, all the books in the store or a subset. We tested the method both locally and using the RPC api and our tests were successful.

b

We tested the service using an extended testing suite including tests of the new functionality implemented previously. For `rateBooks` we rated both whether we could rate books successfully and if potential errors were caught properly, totalling 4 tests. For `getTopRatedBooks` we tested multiple negative outcomes, including if no books were rated, if fewer books were rated than queried, and if a negative number of books were queried. We also tested if the method successfully extracted the books queried, totalling 5 tests. For `getBooksInDemand` we tested whether the method could correctly extract the books in demand when zero, some, or all books were in demand, totalling 3 tests.

These tests and the rest of the suite was tested using both the local version and with the server running over HTTP to ensure correct functionality.

2

a

The architecture is strongly modular in the sense that we have separated client and server sides explicitly and limited possible communication channels between them. This modularity is strong because we also ensure that no errors in one module will directly cause an error in another module.

b

The server, `BookStoreHTTPServer` is responsible for dealing with requests from the bookstore service and distributing them, thus providing a layer of security. The `BookStoreHTTPProxy` and the `StockManagerHTTPProxy` serves as some sort of isolation between the clients and the bookstore service and controls the access to some functionality.

c

The enforced modularity is compromised when running clients and services locally in the same JVM, as the components will need / gets access to functionalities they usually cannot access.

3

a

Yes, there is - it is the stockManager. The stockManager has an accompanying http message handler which handles naming of requests sent between HTTP proxies for the book store and the stock manager.

b

The naming mechanism used utilizes specific path namses in http requests to specify communications. The proxy code send messages between clients and services by adding a named action to the server address when sending a message.

4

This depends on the way the HTTPS handles retries and how the components of the architecture has implemented error-handling strategies. One could also check if the system has idempotent methods. One would strive to have the exactly-once semantic, but this is very difficult, and most systems implements the at-least-once.

5

a

Yes.

b

Though the use of proxy servers can introduce potential extra latency, when they are placed between the client and server, they can be used for caching calls from the clients repeatedly addressing the same books. When scaling the architecture horizontally (e.g. adding more servers or distributing the workload between more parts of the architecture), it can be beneficial to use proxies. Furthermore proxies can enhance security between the servers and the external clients. The proxy servers could deployed with the StockManagerHTTPProxy and BookStoreHTTPProxy which already is the components between the server and clients.

6

a

Yes.

b

The BookStoreHTTPServer could be a bottleneck if it receives too many request to distribute throughout the components in the architecture.

7

a

In the case where the proxies are deployed as per question 5b, the clients would still experience disruptions, though the proxies could be implemented with failover mechanisms to handle such events.

b

It could mask failures to some extend but would not serve as a complete failover mechanism. It might not have the recently updated of the data ready.

c

Cached data might not always represent real-timed values in the case of A Certain Bookstore, especially when the number of books are frequently updated. Since the architecture is not only read-heavy, but also rather write-heavy, caching might not be a good choice to implement, as it might introduce stale data, which would change the experience for the client, e.g. if the stock of books is updated to miss one, but this is not cached.