Distributed Computing Project 2

Ben Lenihan – T00200674

1. Describe a redesign of your short message service as a microservice using RPC or REST as an interface – explain why you choose the interface method you chose (REST/RPC) and describe the microservice principles you will apply in your design. Using pseudo-code describe the implementation of the four methods of your service: **15 marks**

In this Project I have used ONC RPC as an interface. While REST is better at getting information RPC is better for calling functions. This means that functions like login and logout should work better with RPC, whereas upload and download functions would be more suitable for REST.

Very little refactoring is needed in turn reducing the amount of time and effort required to redesign the project. RPC is easy to implement for this type of application.

RPC is lightweight allowing for a more efficient implementation. It also allows preservation of the business logic of the application.

ECHOPROG contains a method login\_1() which takes two strings: uName, a username and pWord, a password.

Pseudo Code:

Login:

OnClickLogin()

If Username and Password in Database

Return 101 Username already in Database, Login Successful

If Database is not found

Generate Database and add Username and password

Return 102 Username didn’t exist previously

If Username not in Database

Return 102 Username didn’t exist previously

Else

Return 103 Error

Display Login screen

Send Message:

onClickSendMessage()

If text is empty

Return error

If text is !empyty

Send Message to server

Download Messages:

onClickDownloadMessages()

download messages from server

Display received messages

Logout:

onClickLogout()

User logged out

Display Login Screen

1. Implement **one** of the four methods (note you don’t have to host it on Docker): **15 marks** . Look at the labs on RPC (ONC RPC/Thrift/gRPC) or REST(IntelliJ or DropWizard) to help you.

For this I chose to implement the login method using RPC. First I created an IDL file, used to generate code which will be used.

Graphical user interface

Description automatically generated

Figure 2.1 IDL file Echo.x

With this login methods were generated.

Text

Description automatically generated

Figure 2.2 Login method in ECHOPROG



Figure 2.3 Login method in ECHOPROGServer

EchoClient and EchoServer were refactored from the previous project.

Text

Description automatically generated

Figure 2.4 EchoClient

Text

Description automatically generated

Figure 2.5 EchoServer

First EchoServer is run

Text

Description automatically generated

Figure 2.6 Server is Running

Then the client can be run.

The following are screenshots taken of the client UI.

Graphical user interface

Description automatically generated

Figure 2.7 Username UI

Graphical user interface

Description automatically generated

Figure 2.8 Password UI

Graphical user interface, text, application

Description automatically generated

Figure 2.9 Login Successful

Graphical user interface, application, website

Description automatically generated

Figure 2.10 User Login Successful

In the event a username and/or password are not entered

Graphical user interface, application

Description automatically generated

Figure 2.11 User Login Failed

1. Compare and contrast your original protocol-based solution with the new microservices solution **20 marks (**make four points supported with examples**)**

The new microservice based approach is much simpler to implement when compared to the previous protocol-based one. While the protocol-based solution required datagrams and sockets whereas the RPC based microservice did not require these. This made the microservice significantly simpler to implement as it used an IDL file. This IDL file generated two classes with given methods and parameters allowing for a faster and easier setup.

The protocol-based solution would be more difficult to upscale over the RPC microservice. The protocol-based solution would require a significant amount of redesigning both the code and design documentation. This would cost a lot of time and effort that could be used elsewhere. The RPC microservice does not have this issue. The IDL file would be the only major change in this case where it would be redesigned to generate any new classes.

The protocol-based solution required far more code. This approach had far more classes, each having a significant amount of code within them. This made debugging more difficult as you had to sort through more code. They could also become more difficult to read through with the number of classes and methods. The RPC microservice had fewer classes, with each class being shorter with fewer methods than comparable classes from the protocol-based solution. Several of these classes were autogenerated. This approach saved time by lessening the amount of code which had to be written and kept track of.

The protocol-based solution is much lower level requiring more setup. I first designed the protocol that would be implemented and then I designed and implemented the four methods. The RPC microservice is layer higher than the protocol approach. It requires less direct interaction with the computer as it does not use datagrams and sockets unlike the protocol approach.