National College of Ireland

Higher Diploma in Science in Computing Information

Distributed Systems

Lecturer: Caitriona Nic Lughadha

Student: Laura Santana – 23169931

Continuous Assessment (CA)

1. Proposal

1 Introduction
This proposal outlines the guide for a Smart Cinema, trying to optimize the control of services and daily activities inside the desire environment. Seeking a better flux of the activities and time.

2 Service 1: ScreenControl
With this service, the idea is to have the control of the screens. The client would be able to ask the screen status – if the screens are ON or OFF - and in both cases, the client would be able to turn the screen ON or OFF, changing the ScreenStatus. The most important feature of this service is the ALLScreenStop method when the client can ask for all the screens to be turned off in case an emergency is found or at the end of the day, instead of having to do it manually with each screen individually.

The attributes are:
- ScreenID;
- ScreenStatus (it can be on/off);
- ScreenON;
- ScreenOFF;

2.1 Methods
2.1.1 RPC Method 1 – ScreenStatus ()
Using a Unary RPC Method. The client makes a call to this service asking for the ScreenStatus, specifying the ScreenID in the call.
The service gets back telling the ScreenStatus returning the value in the real-time between the two options: ScreenON or ScreenOFF.

rpc SayStatus (ScreenStatus) returns (ScreenON/ ScreenOFF) { }

2.1.2 RPC Method 2 – TurnScreenON ()
Using a Unary RPC Method. The client makes a call to the service asking to turn the ScreenStatus to ON, specifying the ScreenID in the call.
The service return ScreenON.

2.1.3 RPC Method 3 – ALLScreenStop()

Similar with the Method 2, Using a Unary RPC Method. The client makes a call to the service asking to turn the ScreenStatus to OFF, selecting all ScreenIDs.
The service return ALLScreenStop. 5

3 Service 2: StaffStatus

Whis this service, it would be able to keep a track on the StaffStatus. With the functions of ClockIn and ClockOut and keeping a track of the time for both functions.
It will also be able to keep a track of how many hours each Staff have done
The attributes are:

- StaffID;
- StaffStatus; (if the staff is “clock-in” or “clock-out”)
- time; (currently time for each activity)
- StaffHours; (the number of hours done by the day)
- StaffTotalHours (the number of hours done by the week)

3.1 Methods

3.1.1 RPC Method 1 – StaffStatus ()

Using a Unary RPC Method. The client makes a call to this service asking for the StaffStatus, specifying the StaffID in the call.
The service gets back telling the StaffStatus returning the value in the real-time between the two options: StaffIn or StaffOut.

3.1.2 RPC Method 2 - StaffHours ()

Using a Server streaming, the client makes a call to the server asking for StaffHours, specifying the StaffID.
The services get back telling the number of hours done by the staff in the day, calculating by the difference of time from the Clock-outa and Clock-in.

4 Service 3: DoorLocker

With this service, it will be possible to have a control of the access for different parts of the cinema, controlling the doors locks. Each door would have a DoorID that will have different styles of controls, depending on the StaffLevel and the DoorLevel, keeping the access restrict for different areas and also been able to lock or unlock any door remotely.
The attributes are:

- DoorID;
- DoorLevel; (will keep the control for each staff and who can get access to)
- StaffID; 6
- DoorStatus;
- DoorOpen;
- DoorClose; (even close, anyone can open)
- DoorLock; (only the staff with the right level/authorization can open)

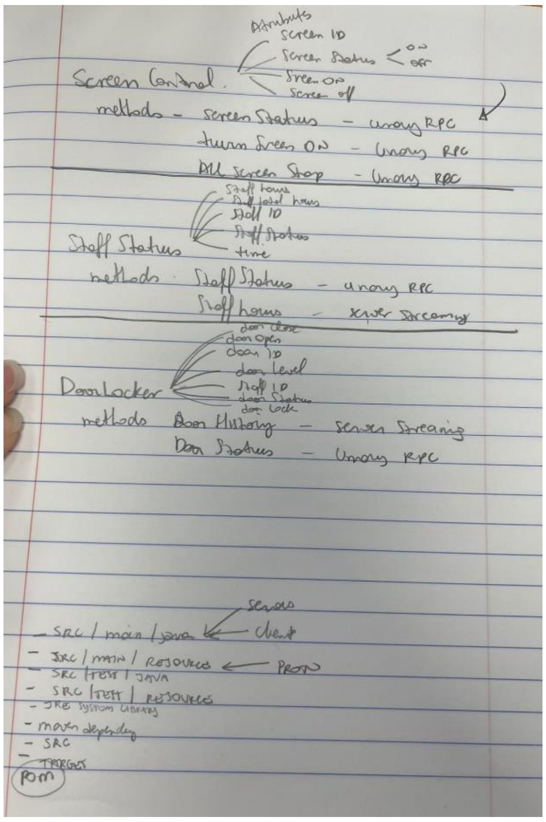
4.1 RPC Method 1 – DoorHistory ()

Using a Server streaming, the client makes a call to the server asking for the DoorHistory, specifying the DoorID in the call.
The serves gets back telling all the access that the specify door has in the certain amount of time, including the last one, showing the DoorLevel and the StaffIDs.

4.1.1 RPC Method 2 – DoorStatus ()

Using a Unary RPC Method. The client makes a call to this service asking for the DoorStatus, specifying the DoorID in the call.
The service gets back telling the DoorStatus returning the value in the real-time between the two options: DoorOpen, DoorClose or DoorLock.

2. About the code – structure



At the beginning of the second part of the assignment, I was having issues with what to do first. Them after checking and having some help from my colleagues, I realized that the first thing that I had to create was a project and create the pom file, coping from the one that was used in class.

After that, I had to create a proto file for each service chosen and write the methods used in each one. I went back to the proposal for that.

The third stage was creating the Server and Client class for each of the methods and in the eclipse, all the other classes would be built with the maven.
There was a lot of coming and backing from each class and proto once all was being changed with the code.

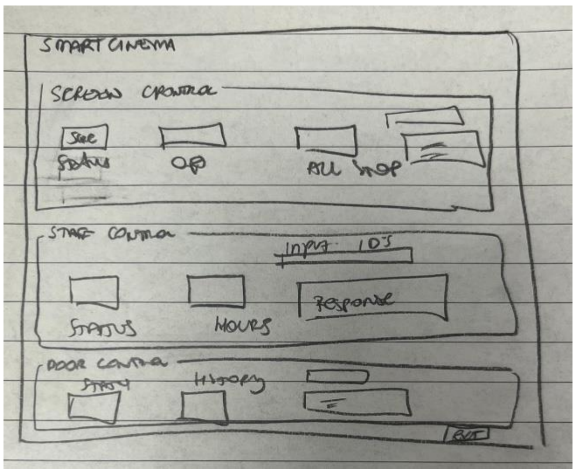
I structure the Eclipse project separating the com. Packages for each server, it helps to keep track of what was doing and where to go. (com.DoorControl/ com.ScreenControl/ com.StaffControl).

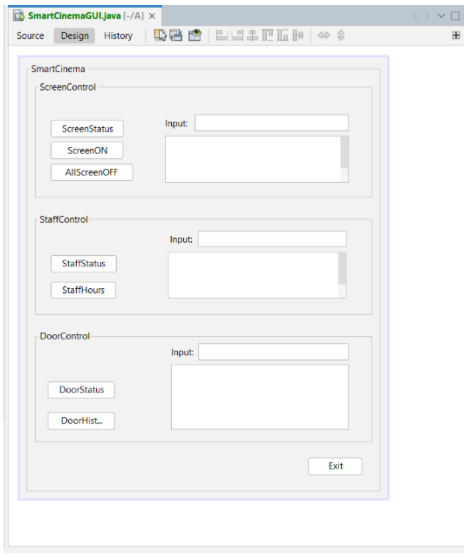
3. Naming Services.

It was the though part for me, once I wasn't in class when the subject was done.
I had issues with the pom file after doing that, changing some code from it.
I had help with colleges from class to understand what I had to do, that was create a Registration and Discovery class for each of the serves. I used examples done in class to help with that.

4. Graphical user Interface

We had a small introduction to the GUI system in the Data Structure's classes where we created an interface. I decided to use that as a base when creating mine.
I did a little scratch about the layout and start doing in NetBeans.





After done with the layout in NetBeans, I copy the code from the eclipse in the designed area and once all the methods are in the right place, I copy the file in a package on eclipse. And fix with the suggestions from the eclipse itself.

5. Github

The link for the repository is: https://github.com/lauracsantana/SmartCinema.git

References:

I used this channel to help me to understand the steps of GRPC.

[Implementing GRPC Service in Java (youtube.com)]