**Distributed Systems**

**Continuous Assessment (CA) Type: Project**

**Smart Home**

**Due Date: 19th April 2024**

**Lecture: Mark Cudden**

Repository:

<https://github.com/sergio-oliveira-br/CA1-DistributedSystems>

Sergio Vinicio da Silva Oliveira

[**x23170981@student.ncirl.ie**](mailto:x23170981@student.ncirl.ie)

Table of Contents

1. Domain Description ........................................................................................................................... 3

2. Service Definition ................................................................................................................................ 4

3. Service Implementations................................................................................................................... 6

4. Command Line & GUI ....................................................................................................................... 7

5. GitHub...................................................................................................................................................... 9

Smart Home v2.0 – GUI by JavaFX**A screenshot of a computer

Description automatically generated**

**1. Domain Description**

To apply the concepts presented in the classroom, using the idea of a smart home as a basis, I chose to approach the four gRPC services. Also, in an attempt to improve and reinforce my concepts in gRPC services, and assuming that practice leads to perfection, more than one version was built to explore the Java Swing and JavaFX topics.

The first two versions differ only in the graphical presentation, where one addresses command line concepts, and the second addresses the GUI graphical user interface through the Java Swing features.

**Please consider the third version as the main one to be evaluated. It includes GUI addressing JavaFX.**

**2. Service Definition**

As mentioned earlier, the project consists of three files and the first two differ only in the graphical presentation.

For these named versions of Smart Home v1.0, and Smart Home GUI v1.1, I am presenting the services listed below:

In Unary RPC, the user must send only one request and get a single response from the Server. The application of this concept is associated with the command on and off lights, and the opening and closing of doors.

Client Streaming RPC suggests the client send a sequence of messages to the Server, that replies with only one message. This concept is connected to the connection verification procedure in this project, where the user sends his data to the Server to confirm his connection.

Bidirectional Streaming RPC suggests continuous interaction between Client and Server. This project is being addressed in the topic of Smart Thermostats, where the customer requests ambient temperature adjustment and weather forecast, obtaining the answers as confirmation of the ambient temperature adjustment and the weather forecast of the next day.

As mentioned earlier, the project consists of three files and the first two differ only in the graphical presentation.

For these named versions of Smart Home v1.0, and Smart Home GUI v1.1, I am presenting the services listed below:

In Unary RPC, the user must send only one request and get a single response from the Server. The application of this concept is associated with the command on and off lights, and the opening and closing of doors.

Client Streaming RPC suggests the client send a sequence of messages to the Server, that replies with only one message. This concept is connected to the connection verification procedure in this project, where the user sends his data to the Server to confirm his connection.

Bidirectional Streaming RPC suggests continuous interaction between Client and Server. This project is being addressed in the topic of Smart Thermostats, where the customer requests ambient temperature adjustment and weather forecast, obtaining the answers as confirmation of the ambient temperature adjustment and the weather forecast of the next day.

Regarding the proposal for the Smart Home v2.0, in the file, there are five Proto Buffers identified.

This final release only has three of them: Domestic Utilities, Environment Management, and Home Security.

Domestic Utilities proposes a single service with two methods present, which can be summarized in the client’s request for power consumption monitoring, obtaining as a response streaming from the Server. The customer has the option to request the interruption of streaming services and get a confirmation of cancellation as a response.

Environment Management also has a single service, but with three methods, which address the concepts of Unary RPC and Server-Side Streaming RPC.

The proposal of this service does not differ much from ta one mentioned earlier, the server sends multiple messages indicating that the air control system is configured according to the client's request for an air temperature adjustment.

Besides being accessible in the Terminal and text field on the GUI screen, identifying this information is also possible through the chart on the GUI screen.

Environment Management also includes a third method that addresses the unary concept, in which the user requests the weather forecast from the server, which uses time to return a single message symbolizing the requested forecast.

Home Security offers two services, one of which focuses on unary concepts, where the client makes a request using two built-in methods and receives a response from the server. The other proposes the server streaming the information that the alarm was activated as requested by the Client.

**3. Services Implementations**

The services mentioned were implemented using Proto Buffers, specifically the "proto 3" syntax.

Once the services and methods have been defined, the Java compiler generates the necessary files with the classes for each message, including getters and setters field values, serializing your messages into an output stream and analyzing your messages from an input stream.

The implementation structure is based on the extension of the classes generated through Proto Buffers, where we initially build a response object to return to the Client or Server as declared in the service definition. In addition, we have included the observer response methods, which through the onNext( ), onError( ) and onCompleted ( ), we can track the current status of the service.

Besides carrying out the methods mentioned earlier, the implementation and initialization of the gRPC Server are essential for this communication between Client and Server.

In a nutshell, this server implementation is done by specifying an address and a port for communication, making it possible to include the service through the addService( ). Finally, launch the RPC server through the build( ) and start( ) methods.

Regarding the Client-Side, it is essential to create a gRPC channel for Stub, which is through the ManagedChannelBuilder( ) method that we specify the port or address of the server to be used for communication with the server.

After the Channel is established, the newStub() and newBlockingStub( ) methods provided through Proto Buffers allow association with stubs, which can be used synchronously or asynchronously.

The client and server deployment is complete and ready to use the declared service methods once this is finished.

**4. Command-Line and GUI**

Due to the challenges proposed by the project, the Smart Home v1.0 version was developed to interact with the user through the command line, concentrating solely on the development and implementation of gRPC services and not on anything else.

During the development of the initial version, I realized that the time available for completion of this project allowed the increment of new challenges, and it was there that I chose to migrate the initial version to the GUI using the Java Swing concepts.

In this version of Smart Home GUI v1.1, I attempted to reinforce the knowledge acquired in the gRPC implementation and apply efforts in learning Java Swing.

Despite having the time to send the project, I decided to take on a new challenge, which led to Smart Home v2.0.

I am dedicating myself once more to strengthening gRPC concepts and providing code that is cleaner, better, and more organized. At this stage, my aim is to promote new learning by introducing JavaFX concepts and gRPC features to provide a closer experience of the reality of software development.

Smart Home v1.0 – Command Line

A screenshot of a computer

Description automatically generated

Smart Home v1.1 – GUI by Java Swing

A screenshot of a computer

Description automatically generated

Smart Home v2.0 – GUI by JavaFX

A screenshot of a computer

Description automatically generated

**5. GitHub**

This project was stored in a public repository on GitHub and can be accessed through the link, <https://github.com/sergio-oliveira-br/CA1-DistributedSystems>.

More than 180 commits were made due to approximately 20 days of work.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated