**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 ............................................................................................................ ?

2. Service Definition ....................................................................................................................... ?

3. Service Implementations................................................................................................................... ?

4. Advanced Features ................................................................................................. ?

5. Command Line & GUI ................................................................................................................... ?

6. GitHub.......................................................................................................................... ?

**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.

The third version is no longer directly associated with the main idea of the Smart Home context. However, through this, I strengthened my understanding of the gRPC services approach and was introduced to 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.

Concerning the proposal for the Smart Oven v2.0 version, I implemented Unary RPC for an individualized greeting and effortless access, and Server Streaming RPC sending fictitious data that simulates the temperature of an oven.

Como mencionado anteriormente, o projeto consiste em três arquivos na qual os dois primeiros se diferem apenas na apresentação gráfica.

Para estas versões nomeadas Smart Home v1.0 e Smart Home GUI v1.1 proponho os seguintes serviços.

Unary RPC, propõe ao usuário envia uma única solicitação, e obtém uma única resposta do servidor. A aplicação deste conceito está associada ao comando de ligar e desligar luzes, e a abertura e fechamento de portas.

Client Streaming RPC, sugere o cliente enviar uma sequência de mensagens ao servidor, que por sua vez responde com uma única mensagem. Neste projeto este conceito está sendo associado no procedimento de verificação de conexão, onde o usuário envia seus dados ao Servidor para confirmar sua conexão.

Bidirectional Streaming RPC: Este serviço propõe uma interação contínua entre cliente e o servidor. E neste projeto está sendo abordado no tópico Smart Thermostats, onde o cliente solicita ajuste de temperatura ambiente e a previsão do tempo, obtendo as respostas como confirmação do ajuste de temperatura ambiente e a previsão do tempo do dia seguinte.

Referente a versão nomeada SmarOven v2.0 implementei os seguintes serviços.

Unary RPC para uma simples saudação e personalização de acesso ao disposto Smart Oven.

Server Streaming RPC, enviando dados fictícios que simulam a temperatura de um forno.

**3. Services Implementations**

Para implementar os serviços mencionados, fora utilizado Proto Buffers, especificamente a sintaxe proto 3.

Uma vez que os serviços e métodos foram definidos, o compilador Java gera os arquivos necessários com as classes para cada mensagem incluindo obter e definir valores de campo, serializar suas mensagens em um fluxo de saída e analisar suas mensagens de um fluxo de entrada.

A estrutura de implementação tem com base a extensão das classes geradas através do Proto Buffers, onde inicialmente construímos um objeto de resposta para retornar ao cliente ou servidor conforme declarado na definição de serviço. Além disso, temos incluso os métodos de resposta do observador, que por meio dos metodos onNext( ), onError( ) e onCompleted ( ) podemos acompanhar o status atual do serviço.

Além da implementação dos métodos mencionados anteriormente, a implementação e inicialização do servidor gRPC é fundamental para que ocorram essa comunicação entre cliente e servidor.

De modo resumido, esta implementação do Servidor é construída por meio da especificação de um endereço e porta para efetivar a comunicação, sendo possível adicionar este serviço ao construtor por meio do método addservice( ), e por fim, iniciando o servidor RPC por meio dos métodos build( ) e start( ).

Referente ao Client-Side, é fundamental a criação de um canal gRPC para o Stub, que é por meio do método ManagedChannelBuilder( ) que especificamos a porta ou endereço do servidor a ser utilizado para a comunicação com o servidor.

Uma vez que o canal foi criado, os métodos newStub( ) e newBlockingStub fornecidos através do Proto Buffers permitem a associação com o Stub, que neste caso utilizei o de bloqueio/síncrono, na qual a chamada RPC aguarda a resposta do servidor, retornando uma resposta ou uma exceção.

Feito isso, implementação do cliente e servidor estão concluídas e prontas para chamar os métodos de serviços declarados.

**4. Testing & Responsiveness**

Talk about website responsiveness here along with **Optimisation, SEO and Testing**. You can talk about how you made your website responsive. You could also mention how you used JavaScript: loops, if statements, etc. How you deployed your website?

**5. Command-Line and GUI**

Devido aos desafios que incialmente não estava claro, a versão Smart Home v1.0 foi desenvolvida para interagir com o usuário por meio de command-line, não contemplando nada de especial, e focando exclusivamente no desenvolvimento e implementação dos serviços gRPC.

Durante o desenvolvimento da versão inicial, percebi que o tempo disponível para conclusão deste projeto possibilitava a incrementação de novos desafios, e foi ai que optei em migrar a versão inicial para o GUI aplicando os conceitos Java Swing.

Nesta versão Smart Home GUI v1.1, busquei reforçar os conhecimentos adquiridos na implementação gRPC, e aplicar esforços no aprendizado Java Swing.

Ainda com tempo disponível para o envio do projeto, decide me aventurar em um novo desafio, e assim surgiu a Smart Oven v 2.0. Nesta versão me dedico mais uma vez em reforçar os conceitos gRPC focando em entregar um código mais limpo, mais objetivo e mais organizado, além de promover um novo aprendizado nos conceitos JavaFX.

Vale apena ressaltar que proposta Smart Oven não está integralmente conectado com o projeto. A ideia principal era introduzir a mim mesmo o JavaFX, e diante da oportunidade do projeto, optei em unir JavaFX e gRPC para fins acadêmicos.

**6. 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 140 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