UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
DEPARTAMENTO DE INFORMÁTICA APLICADA

INF01151 – SISTEMAS OPERACIONAIS II N
SEMESTRE 2021/2
ATIVIDADE DE PROGRAMAÇÃO GUIADA: REMOTE PROCEDURE CALLS

Nome: Gabriel Martins dos Santos Cartão: 00275617

Para esse exercício de programação guiada, após ler os materiais complementares e seguir as instruções do documento disponibilizado.

Gerei as seguintes mudanças para implementar todos os exercícios propostos.

Mudanças no arquivo **calculator.proto**:

```
message MultiplyRequest {
  double a = 1;
  double b = 2;
}

message MultiplyReply {
  double s = 1;
}

message GreaterRequest {
  double a = 1;
  double b = 2;
  double c = 3;
}

message GreaterReply {
  double s = 1;
}
```

```
message QuotientAndRemainerRequest {
  double a = 1;
  double b = 2;
}

message QuotientAndRemainerReply {
  double s = 1;
  double t = 2;
}

service Calculator {
  rpc Sum (SumRequest) returns (SumReply);
  rpc Multiply (MultiplyRequest) returns (MultiplyReply);
  rpc Greater (GreaterRequest) returns (GreaterReply);
  rpc QuotientAndRemainer (QuotientAndRemainerRequest) returns (QuotientAndRemainerReply);
}
```

Mudanças no arquivo **calculator_server.py**:

```
def Multiply(self, request: MultiplyRequest, context: ServicerContext) -> MultiplyReply:
    return MultiplyReply(s=request.a * request.b)

def Greater(self, request: GreaterRequest, context: ServicerContext) -> GreaterReply:
    return GreaterReply(s=self.greaterBetween(request.a,request.b,request.c))

def QuotientAndRemainer(self, request: QuotientAndRemainerRequest, context: ServicerContext) -> QuotientAndRemainerReply:
    return QuotientAndRemainerReply(s=request.a / request.b, t=request.a % request.b)

def greaterBetween(self, a, b, c):

    if(a >= b and a >= c):
        return a

    if(b >= a and b >= c):
        return b

    if(c >= a and c >= b):
        return c
```

Criação de novos testes no arquivo **calculator_integration_test.py**:

```
def test_multiply(calculator_client):
    from calculator_pb2 import MultiplyRequest

    # given
    a = 256.5
    b = 128.8

    expected = a * b

    # when
    result = calculator_client.Multiply(MultiplyRequest(a=a, b=b))

    # then
    assert result.s == expected
```

```python
def test_greater(calculator_client):
    from calculator_pb2 import GreaterRequest

    # given
    a = 256.5
    b = 128.8
    c = 74.4

    expected = a

    # when
    result = calculator_client.Greater(GreaterRequest(a=a, b=b, c=c))

    # then
    assert result.s == expected
```

```python
def test_quotientAndRemainer(calculator_client):
    from calculator_pb2 import QuotientAndRemainerRequest

    # given
    a = 256.5
    b = 128.8

    expected1 = a / b
    expected2 = a % b

    # when
    result = calculator_client.QuotientAndRemainer(QuotientAndRemainerRequest(a=a, b=b))

    # then
    assert result.s == expected1
    assert result.t == expected2
```

Após executar todos os testes utilizando os comandos

python3 -m grpc_tools.protoc -I. --python_out=. --grpc_python_out=. Calculator.proto

Seguido de **python3 -m pytest**

```
-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
========================================== 4 passed, 2 warnings in 0.12s ==========================================
gabriel@gabriel-VirtualBox:~/Downloads/PG2/PG2-RPC$
```