Universidade Federal do Rio Grande do Sul - Instituto de Informática
INF01151 – Sistemas Operacionais II - Prof. Alberto Egon Schaeffer Filho

Nome: Eduardo Dalmás Faé Cartão: 334087 Data: 01/12/2023
Nome: Giordano Souza de Paula Cartão: 308054
Nome: João Pedro Kuhn Braun Cartão: 325265

Atividade de Programação Guiada 2 - Relatório

1. Implementação dos Requests e Replies

No arquivo calculator.proto foi realizada a implementação das mensagens com a definição dos requests e replies para cada novo procedimento.

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

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

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

message MaxReply {
  double s = 1;
```

```
message DivisionRequest {
  int64 a = 1;
  int64 b = 2;
}

message DivisionReply {
  int64 q = 1;
  int64 r = 2;
```

2. Adição dos serviços

No arquivo calculator.proto foram adicionados os novos procedimentos.

```
service Calculator {
  rpc Sum (SumRequest) returns (SumReply);
  rpc Multiply (MultiplyRequest) returns (MultiplyReply);
  rpc Max (MaxRequest) returns (MaxReply);
  rpc Division (DivisionRequest) returns (DivisionReply);
}
```

3. Implementação dos procedimentos

No arquivo calculator_server.py foram definidos 3 novos procedimentos, um que realiza a multiplicação, outro que retorna o máximo entre 3 números, e outro que retorna o quociente e o resto de uma divisão entre 2 números.

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

def Max(self, request: MaxRequest, context: ServicerContext) -> MaxReply:
    return MaxReply(s=max(request.a, request.b, request.c))

def Division(self, request: DivisionRequest, context: ServicerContext) -> DivisionReply:
    return DivisionReply(q=int(request.a / request.b), r=request.a % request.b)
```

4. Testes

No arquivo calculator_integration_test.py foram implementados testes para cada um dos novos procedimentos criados. Todos seguindo o modelo abaixo.

```python
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_max(calculator_client):
    from calculator_pb2 import MaxRequest

    # given
    a = 256.5
    b = 128.8
    c = 64.4

    expected = max(a, b, c)

    # when
    result = calculator_client.Max(MaxRequest(a=a, b=b, c=c))

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

```python
def test_division(calculator_client):
    from calculator_pb2 import DivisionRequest

    # given
    a = 256
    b = 128

    expected_q = int(a / b)
    expected_r = a % b

    # when
    result = calculator_client.Division(DivisionRequest(a=a, b=b))

    # then
    assert result.q == expected_q and result.r == expected_r
```

5. Resultados Obtidos

```
[joao-pedro@fedora PG2]$ python -m pytest
============================================= test session starts ==============================================
platform linux -- Python 3.11.6, pytest-7.4.3, pluggy-1.3.0
rootdir: /home/joao-pedro/Downloads/SisOp2_PGs-main/PG2
collected 4 items

calculator_integration_test.py ....                                                                    [100%]

============================================== 4 passed in 0.91s ===============================================
```

Link para o código: https://github.com/eduardofae/SisOp2_PGs/tree/main/PG2