Guia Prático – gRPC – POC: Calculadora Simples

Este é um guia prático para criar e utilizar a POC (Proof of Concept) criada pelo nosso grupo. Você será capaz de construir uma calculadora simples, utilizando gRPC, com comunicação entre servidor e cliente, a partir da linguagem de programação Python.

Antes de iniciar, algumas observações:

- Ter o ambiente Python baixado no computador
- Não utilizar Jupyter Notebook (.ipynb), mas sim .py
- Todos os arquivos criados nesse projeto precisam estar na mesma pasta

1) Instalar bibliotecas do Python utilizadas:

> terminal

```
pip install grpcio
pip install futures
```

2) Criação de um arquivo “.proto”

- Criar o arquivo, como por exemplo, “calculator.proto”

> protobuf

```
syntax = "proto3";

package calculator;

service Calculator {
    rpc Add(AddRequest) returns (AddResponse);
}

message AddRequest {
    int32 a = 1;
    int32 b = 2;
}

message AddResponse {
    int32 result = 1;
}
```

3) Compilação do arquivo “.proto”

- No terminal, navegar até o diretório com o arquivo e executar o seguinte código:

> terminal

```
python -m grpc_tools.protoc -I. --python_out=. --grpc_python_out=.
calculator.proto
```

4) Criação e implementação do servidor

- Criar o arquivo do servidor, como por exemplo, “server.py”

> python

```
import grpc
import calculator_pb2
import calculator_pb2_grpc
from concurrent.futures import ThreadPoolExecutor

class CalculatorServicer(calculator_pb2_grpc.CalculatorServicer):
    def Add(self, request, context):
        result = request.a + request.b
        return calculator_pb2.AddResponse(result=result)

def serve():
    server = grpc.server(ThreadPoolExecutor(max_workers=10))
    calculator_pb2_grpc.add_CalculatorServicer_to_server(CalculatorServic
er(), server)
    server.add_insecure_port('[::]:50051')
    server.start()
    server.wait_for_termination()

print('Servidor Iniciado... \n ctrl + c para sair')

if __name__ == '__main__':
    serve()
```

5) Criação e implementação do cliente

- Criar o arquivo do cliente, como por exemplo, “client.py”

> python

```
import grpc
import calculator_pb2
import calculator_pb2_grpc

def run():
    with grpc.insecure_channel('localhost:50051') as channel:
        stub = calculator_pb2_grpc.CalculatorStub(channel)
        a = int(input("Digite o primeiro número: "))
        b = int(input("Digite o segundo número: "))
        response = stub.Add(calculator_pb2.AddRequest(a=a, b=b))
    print("Result:", response.result)
```

```
if __name__ == '__main__':
    run()
```

6) Executando o servidor

- No terminal, navegar até o diretório onde estão todos os arquivos. Executar o seguinte comando:

> terminal

```
python server.py
```

7) Executando o cliente

- Em um novo terminal, navegar até o diretório onde estão todos os arquivos. Executar o seguinte comando:

> terminal

```
python client.py
```

- Será solicitado a inserção de dois números. Você, usuário, poderá escolher qual, para somar.
- Ao final será apresentado o resultado da soma dos dois números escolhidos.
- Como mostrado na imagem abaixo: