

CHAMADA DE PROCEDIMENTO REMOTO - GRPC

**FABIANA CAMPOS
JOÃO ROBERTO**



O QUE É?

- Forma de comunicação principalmente entre serviços e microsserviços;
- Comunicação entre sistemas de forma mais rápida, simples e dinâmica;
- Executar uma função de forma remota em uma outra máquina;
- Protocol Buffers (protobuf) - serializa dados estruturados de maneira eficiente, compacta e é independente de linguagem.

GREETER_SERVICE .PROTO

```
1  syntax = "proto3";
2
3  // The greeter service definition.
4  service Greeter {
5      // Sends a greeting
6      rpc SayHello (HelloRequest) returns (HelloReply) {}
7  }
8
9  // The request message containing the user's name.
10 message HelloRequest {
11     string name = 1;
12 }
13
14 // The response message containing the greetings
15 message HelloReply {
16     string message = 1;
17 }
```

SERVER.PY

```
import grpc
from concurrent import futures
import greeter_service_pb2
import greeter_service_pb2_grpc

# Implementando a classe que contém a lógica do serviço
class Greeter(greeter_service_pb2_grpc.GreeterServicer):
    def SayHello(self, request, context):
        # Processa a requisição recebida e retorna a resposta
        message = f"Hello, {request.name}!"
        return greeter_service_pb2.HelloReply(message=message)

# Função para rodar o servidor
def serve():
    # Cria o servidor
    server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))

    # Registra o serviço Greeter no servidor
    greeter_service_pb2_grpc.add_GreeterServicer_to_server(Greeter(), server)

    # Define a porta onde o servidor irá escutar (por exemplo, 50051)
    server.add_insecure_port('[::]:50051')

    # Inicia o servidor
    print("Servidor gRPC rodando na porta 50051...")
    server.start()

    # Aguarda indefinidamente (mantém o servidor rodando)
    server.wait_for_termination()

# Rodar o servidor
if __name__ == '__main__':
    serve()
```

CLIENT.JS

```
const grpc = require('@grpc/grpc-js');
const protoLoader = require('@grpc/proto-loader');

// Carregar o arquivo .proto
const packageDefinition = protoLoader.loadSync('../proto/greeter_service.proto');
const greeterProto = grpc.loadPackageDefinition(packageDefinition).Greeter;

// Conectar ao servidor gRPC (Python)
const client = new greeterProto('localhost:50051', grpc.credentials.createInsecure());

// Enviar uma solicitação de saudação
const request = { name: 'Fabiiii' };

client.SayHello(request, (error, response) => {
  if (!error) {
    console.log('Mensagem recebida do servidor:', response.message);
  } else {
    console.error('Erro:', error);
  }
});
```

grpc Public

 Pin Unwatch 1 Fork 0 Star 0

main 1 Branch 0 Tags

Q Go to file t

Add file

<> Code ▾

About

No description, website, or topics provided.

 [Readme](#)

Activity

☆ 0 stars

1 watching

0 forks

Releases

No releases published

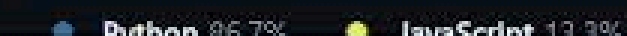
[Create a new release](#)

Packages

No packages published

[Publish your first package](#)

Languages



OBRIGADO!