Construindo sua Primeira API gRPC

1. Preparação do Ambiente.

Passo a Passo:

1. Crie a pasta do projeto:
   - Crie uma pasta chamada loja-grpc.
2. Ambiente Virtual (Recomendado):
   python -m venv venv

3. Instalação das Dependências:
   - grpcio: O pacote principal.
   - grpcio-tools: Ferramentas para gerar código a partir do .proto.

   pip install grpcio grpcio-tools

2. Definindo o Contrato (.proto)

Crie um arquivo chamado loja.proto na raiz do projeto.

1. A versão do protocolo (proto3).
2. Um serviço chamado Estoque com um método RPC BuscarProduto.
3. As mensagens de entrada e saída.

Estrutura:

```
syntax = "proto3";

// Define o serviço
service Estoque {
  rpc BuscarProduto (ProdutoRequest) returns (ProdutoResponse);
}

// Mensagem de entrada
message ProdutoRequest {
   int32 id = 1;
}

// Mensagem de saída
message ProdutoResponse {
  int32 id = 1;
  string nome = 2;
  float preco = 3;
```

```
  int32 quantidade_estoque = 4;
}
```

3. Compilação

Agora precisamos traduzir esse contrato para Python.

Ação: Execute o comando abaixo no terminal. Tente entender o que cada flag faz:

python -m grpc_tools.protoc -I. --python_out=. --grpc_python_out=. loja.proto

- Verifique se os arquivos loja_pb2.py e loja_pb2_grpc.py foram gerados.
- Nunca edite esses arquivos gerados manualmente.

4. Implementando o Servidor

Crie o arquivo servidor.py.

Passo A: Importações e Banco de Dados (Mock) Copie esta estrutura inicial:

```
import grpc
from concurrent import futures
import loja_pb2
import loja_pb2_grpc

# Nosso "Banco de Dados"
PRODUTOS_DB = {
   1: {"nome": "Notebook Gamer", "preco": 4500.00, "qtd": 5},
   2: {"nome": "Mouse Sem Fio", "preco": 120.00, "qtd": 20},
   3: {"nome": "Teclado Mecânico", "preco": 350.00, "qtd": 8},
}
```

Passo B: Lógica do Serviço: Você precisa herdar da classe gerada pelo gRPC e implementar o método.

```
class EstoqueService(loja_pb2_grpc.EstoqueServicer):

   # Dica: O método precisa ter a assinatura (self, request, context)
   def BuscarProduto(self, request, context):
      1. Pegue o ID vindo da request (request.id)
```

print(f"Buscando ID: {request.id}")

2. Busque no dicionário PRODUTOS_DB

3. Se achar: Retorne um objeto loja_pb2.ProdutoResponse preenchido

4. Se NÃO achar:
Use context.set_code(grpc.StatusCode.NOT_FOUND)
Use context.set_details('...')
Retorne um ProdutoResponse vazio
pass

Passo C: Inicialização do Servidor Adicione o código padrão para levantar o servidor na porta 50051:

```
def main():
    server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
    loja_pb2_grpc.add_EstoqueServicer_to_server(EstoqueService(), server)
    server.add_insecure_port('[::]:50051')
    print("Servidor rodando na porta 50051...")
    server.start()
    server.wait_for_termination()

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

5. Implementando o Cliente

Crie o arquivo cliente.py para consumir sua API.

Desafio: Tente implementar a chamada RPC seguindo os passos comentados.

```
import grpc
import loja_pb2
import loja_pb2_grpc

def run():
    1. Crie um canal inseguro para 'localhost:50051'
    use grpc.insecure_channel(...) com 'with'

        2. Crie o Stub (o cliente) usando loja_pb2_grpc.EstoqueStub(channel)
```

print("--- Buscando Produto 1 ---")

3. Chame o método **stub.BuscarProduto**
Você precisa passar um objeto **loja_pb2.ProdutoRequest(id=1)**

4. Imprima o resultado **(response.nome, response.preco)**

5. (Opcional) Tente buscar um ID inexistente e trate o erro com **try/except grpc.RpcError**

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

Final

1. Inicie o servidor: python servidor.py
2. Inicie o cliente: python cliente.py

Se tudo der certo, você verá os dados do Notebook Gamer aparecendo no cliente!