

Logo da Orange Talents

# Consumindo um endpoint via GRPC

## Cenário:

Imagine que um desenvolvedor(a) junior do seu time recebeu a tarefa de criar uma API REST responsável por verificar se determinado cliente possui situação regular no sistema do SERESA.

Para se comunicar com o SERASA temos que se integrar à sua API gRPC pública disponível através de um arquivo Protobuf publicado na página de desenvolvedores da empresa. Ao fazer o download do arquivo .proto temos o seguinte conteúdo:

syntax = "proto3";  
  
service SerasaGrpcService {  
 rpc verificarSituacaoDoCliente(SituacaoDoClienteRequest) returns (SituacaoDoClienteResponse) {}  
}  
  
message SituacaoDoClienteRequest {  
 string cpf = 1;  
}  
  
message SituacaoDoClienteResponse {  
 enum Situacao {  
 DESCONHECIDA = 0;  
 REGULAR = 1;  
 IRREGULAR = 2;  
 }  
  
 Situacao situacao = 1;  
}

Dando início ao projeto, o desenvolvedor(a) criou um projeto Micronaut e o configurou como aplicação gRPC fazendo todo o setup necessário (dependências no Gradle, IDE, stubs gerados etc) para consumir uma API gRPC. Em seguida implementou uma API REST com Micronaut que recebe o CPF do cliente para consultar sua situação no SERASA:

@Validated  
@Controller  
class SerasaController {  
  
 @Post("/api/serasa/clientes/verificar-situacao")  
 fun verificar(@Valid @NotBlank @CPF val cpf): HttpResponse<Any> {  
  
 val situacao = // TODO: consumir API gRPC aqui  
   
 return HttpResponse.ok(SituacaoNoSerasaResponse(cpf, situacao))  
 }  
}  
  
data class SituacaoNoSerasaResponse(  
 val cpf: String,   
 val situacao: Situacao  
)  
  
enum class Situacao {  
 SEM\_INFORMACOES,  
 REGULARIZADA,  
 NAO\_REGULARIZADA  
}

Infelizmente o desenvolvedor(a) junior não tem experiência suficiente com consumo de APIs gRPC. Seu papel é ajudá-lo(a) a terminar de implementar a API REST acima para consumir a API gRPC do SERASA retornando a situação atual do cliente no SERASA.

Como vimos, o projeto Micronaut está devidamente configurado com suas dependências, desse modo, como você implementaria o consumo dessa API gRPC?

## O que seria bom ver nessa resposta?

* **Peso 7**: Criar a factory do client gRPC, injetar o client no controller, usá-lo para consumir a API gRPC usando o CPF recebido pela API REST e converter a response gRPC na enum da response da API REST;
* **Peso 2**: Configurar a conexão (channel) da API gRPC no arquivo application.yml;
* **Peso 1**: Utilizar extension function do Kotlin para converter a response da API gRPC para response da API REST (ou simplesmente para enum de situação);

## O que penaliza sua resposta?

* **Penalidade -2**: Configurar a conexão (channel) de forma hard-coded na factory, por exemplo: @GrpcChannel("https://api.serasa.com.br:50051");

## Resposta do Especialista:

* Crio uma classe para funcionar como uma Factory responsável por criar as instâncias do client gRPC. Esta classe terá um método que recebe um channel via parâmetro e retornará um blocking stub configurado para consumir a API gRPC. Em seguida anoto a classe com @Factory do Micronaut e seu método como @Singleton; também anoto o parâmetro do método com @GrpcChannel informando um name para a configuração da conexão (que estará no application.yml). O código ficaria semelhante a este:
* @Factory  
  class Clients {  
   fun blockingStub(@GrpcChannel("serasa") channel: ManagedChannel): SerasaGrpcServiceBlockingStub   
   return SerasaGrpcService.newBlockingStub(channel)  
  }
* No application.yml adiciono a configuração de conexão (channel) serasa referente a API gRPC do SERASA informando endereço, porta e demais configurações necessárias informadas na documentação do desenvolvedor no site do SERASA;
* Agora, injeto do client gRPC no controller via construtor para comunicar com a API gRPC;
* Dentro do método do controller instancio a request e invoco o método do client gRPC que foi injetado para enviar o CPF informado na API REST. Guardo a response numa variável e converto seu atributo situacao (que é uma enum) para a enum esperada pelo DTO SituacaoNoSerasaResponse, aqui basicamente temos um mapeamento 1-para-1 entre as constantes da enum;
* A conversão da enum eu faço com extension function do Kotlin para encapsular a lógica e tornar o código mais legível, algo como:
* fun SituacaoDoClienteResponse.toModel(): Situacao {  
   return when(situacao) {  
   DESCONHECIDA -> Situacao.SEM\_INFORMACOES  
   REGULAR -> Situacao.REGULARIZADA  
   IRREGULAR -> Situacao.NAO\_REGULARIZADA  
   }  
  }
* Por fim, levanto a aplicação e testo-a via POSTman ou Insomnia para ter certeza de que a configuração da conexão e a comunicação em si estão corretas. Em caso de erro eu analiso os logs da aplicação e se necessário consulto a documentação oficial do Micronaut ou gRPC para me ajudar;