

Logo da Orange Talents

# Introdução a protobuf

## Antes de começar

Para complementar o curso, recomendamos estudar a documentação oficial da Google sobre [Protocol Buffers: Language Guide (proto3)](https://developers.google.com/protocol-buffers/docs/proto3) caso você não tenho estudado ainda. Nela é possível ter uma idéia mais clara do poder e das possibilidades oferecidas pela tecnologia.

## Cenário:

Imagine que tenhamos que criar uma API gRPC para cadastrar um novo carro no nosso sistema. Para isso, nosso tech lead conversou com o cliente e levantou as informações na qual um carro precisa necessariamente ter:

* modelo (ex: Gol, Palio, Celta etc);
* placa (ex: HPX-1234, OIP-9876 etc);
* ano (ex: 1984, 2001, 2019 etc);
* nome e CPF do proprietário;
* tipo de combustível, nesse caso deve suportar 3 tipos: GASOLINA, ALCOOL e FLEX;

Além disso, ele identificou com o time do cliente que o retorno da API pode ser algo simples nesse momento, portanto nossa API gRPC deve retornar somente 2 campos:

* ID interno criado pelo sistema;
* data e hora de criação do carro no sistema;

Para facilitar nossa vida, nosso tech lead começou a escrever o arquivo .proto com a configuração inicial utilizada pela Zup:

syntax = "proto3";  
  
option java\_multiple\_files = true;  
option java\_outer\_classname = "CarrosGrpc";  
  
package br.com.zup.edu.carros;  
  
/\*\*  
 \* o código da sua API gRPC vai aqui  
 \*/

Com base nos requisitos e no código inicial do .proto acima, como você desenharia essa API gRPC com Protobuf?

## O que seria bom ver nessa resposta?

* **Peso 5**: Declarar um serviço com um método remoto que recebe uma mensagem de request e retorna uma mensagem de response. Além disso, declarar as mensagens de request e response de acordo com os requisitos da atividade;
* **Peso 2**: Declarar o campo tipo de combustível como enum em vez de outro tipo;
* **Peso 1**: Criar uma constante UNKNOWN ou NOT\_SPECIFIED como primeira constante da enum para representar um valor padrão quando não informado pelo usuário;
* **Peso 1**: Declarar os dados do proprietário como uma nova mensagem (tipo) contendo os campos nome e cpf;
* **Peso 1**: Declarar o campo data e hora de criação da resposta como Timestamp da Google ou criar um tipo customizado para representar essa informação de forma estruturada;

## Resposta do Especialista:

* Dentro do .proto, começo declarando a API do serviço com um único método remoto que recebe a mensagem de request e tem uma mensagem de retorno. Seri algo como:
* service CarrosGrpcService {  
   rpc adicionar(NovoCarroRequest) returns (NovoCarroResponse) {}  
  }
* Em seguida implemento a mensagem NovoCarroRequest. Eu declaro os campos placa e modelo como string; o campo ano como int32, o tipo de combustível como enum contendo os 3 tipos sugeridos e +1 tipo "unknown" para representar um valor desconhecido ou não preenchido pelo usuário, e por último os dados do proprietário como uma nova mensagem: Proprietario. No fim teria algo como:
* message NovoCarroRequest {  
    
   enum Combustivel {  
   UNKNOWN\_COMBUSTIVEL = 0;  
   GASOLINA = 1,  
   ALCOOL = 2,  
   FLEX = 3  
   }  
    
   message Proprietario {  
   string nome = 1;  
   string cpf = 2;  
   }  
    
   string modelo = 1;  
   string placa = 2;  
   int32 ano = 3;  
   Combustivel tipo = 4;  
   Proprietario proprietario = 5;  
  }
* Agora crio a mensagem NovoCarroResponse com os 2 campos sugeridos: um campo ID do tipo string e um campo criadoEm do tipo Timestamp da Google pois ele já me fornece uma API pronta para representar data e hora entre linguagens e plataformas diferentes. Lembrando que para usar o tipo da Google eu preciso importá-lo no .proto. O código ficaria parecido com esse:
* message NovoCarroResponse {  
   string id = 1;  
   Timestamp criadoEm = 2; // tipo da google  
  }
* Por fim, eu tento gerar os stubs compilando o arquivo .proto. Se algo der errado eu analiso o erro reportado para tentar corrigi-lo e/ou consulto a documentação oficial do Protobuf;