# gRPC e ProtoBuf como solução de mensageria

Eduardo Afonso Dutra Silva - 19/0012307
Rafael Cleydson da Silva Ramos - 19/0019085

# gRPC

Visão Geral

Características

Funcionalidades

Exemplos

# Protocol Buffers (Protobuf)

Os buffers de protocolo são o mecanismo extensível, neutro em termos de linguagem e plataforma do Google para serializar dados estruturados

# PROTOBUF

- Independe de linguagem ou plataforma, compatível com JAVA, C, C++, Python, GO, ruby, Kotlin e Dart.
- Mensagens estão em formato binário e incorpora um conjunto de regras para definição e troca de mensagem
- Precisa ter o conhecimento do esquema para decodigicar a mensagem
- Consegue (de)serializar uma maior varidade de tipo de dados

# JSON

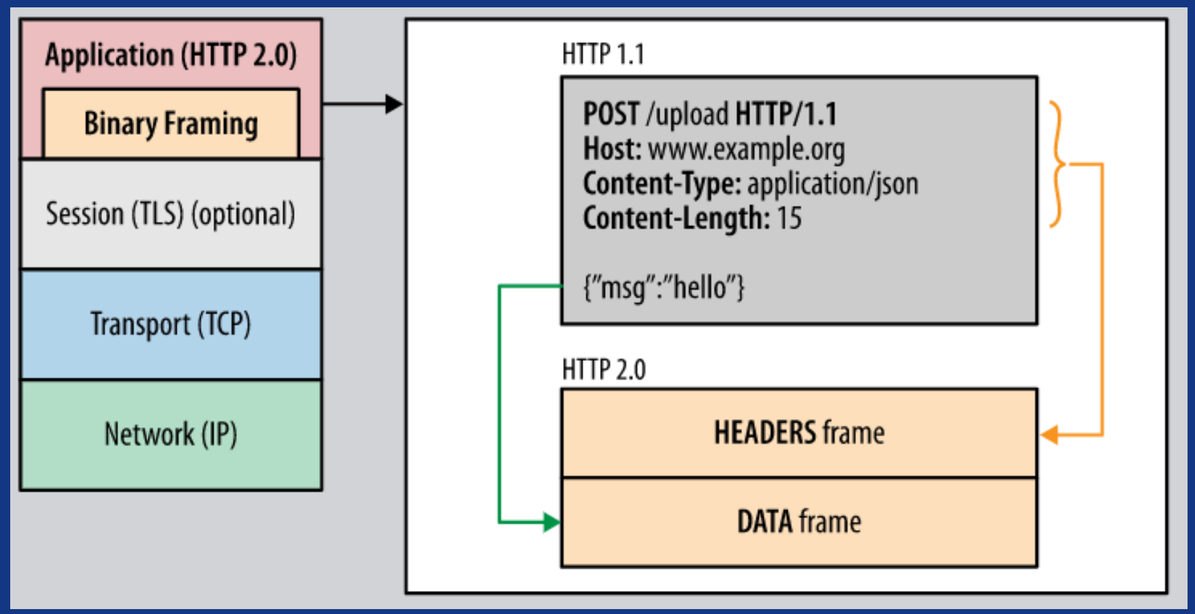

- Derivado do Javascript e suportado pela maior parte das linguagens de programação.
- Mensagens são trocdas em um formato de texto compreesível para humanos em formato simples sem esquema adicional
- Mais utilizado em aplicações web.
- Mensagem é facilmente decodificada sem o conhecimento do esquema.
- É mais restrito quanto aos tipos de dados que podem ser (de)serializados.

# Arquivo .proto

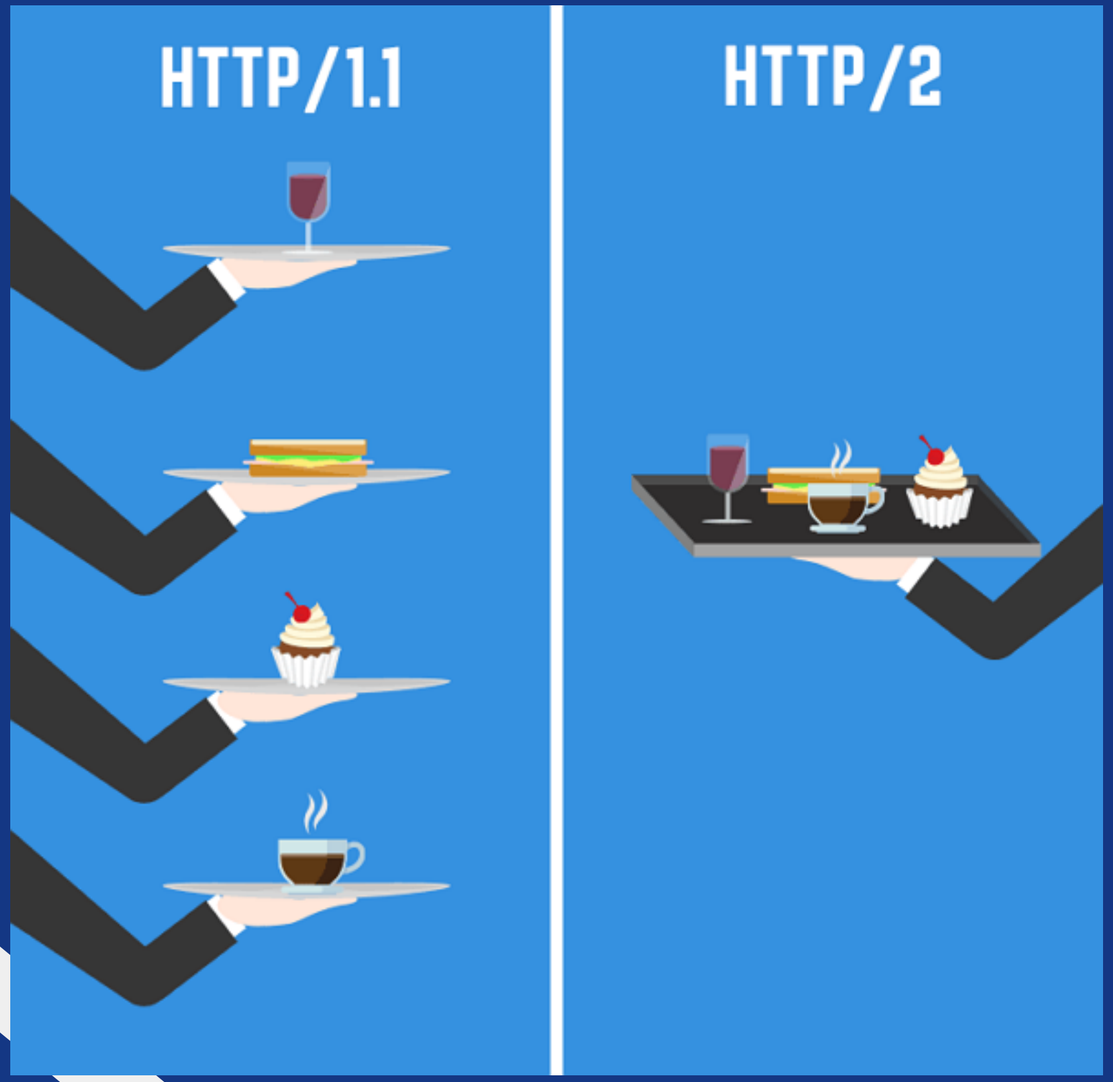
```
15  syntax = "proto3";
16
17  option java_multiple_files = true;
18  option java_package = "io.grpc.examples.finder";
19  option java_outer_classname = "FinderProto";
20  option objc_class_prefix = "FND";
21
22  package finder;
23
24  // The finding service definition.
25  service Finder {
26    // Sends calculated numbers
27    rpc CalculateMinMax (NumbersRequest) returns (NumbersReply) {}
28  }
29
30  // The request message containing the float numbers.
31  message NumbersRequest {
32    repeated float numbers = 1;
33  }
34
35  // The response message containing the min and max numbers from the float vector.
36  message NumbersReply {
37    float min = 1;
38    float max = 2;
39  }
40
```

# HTTP/2

- Redução de latência
- Server Push
- Multiplexação
- Compressão do cabeçalho
- Segurança



# HTTP/2



# Server

```
56  function main() {
57    var argv = parseArgs(process.argv.slice(2));
58    var ipPort;
59    if (argv._[0]) ipPort = argv._[0];
60    else ipPort = "127.0.0.1:50051";
61
62    var server = new grpc.Server();
63    server.addService(hello_proto.Finder.service, {
64      calculateMinMax: calculateMinMax,
65    });
66    server.bindAsync(ipPort, grpc.ServerCredentials.createInsecure(), () => {
67      server.start();
68    });
69  }
70
```

```
32
33  const findMinMax = (numberList) => {
34    var min = Infinity;
35    var max = -Infinity;
36    numberList.forEach((element) => {
37      if (element < min) min = element;
38      if (element > max) max = element;
39    });
40    return [min, max];
41  };
42
43  /**
44   * Implements the SayHello RPC method.
45   */
46  function calculateMinMax(call, callback) {
47    console.log("Números recebidos com sucesso!");
48    let minMax = findMinMax(call.request.numbers);
49    callback(null, { min: minMax[0], max: minMax[1] });
50  }
51
```

# Client Thread Principal

```
78  function main() {
79    if (isMainThread) {
80      var argv = parseArgs(process.argv.slice(2));
81
82      var target = [];
83      if (argv._.length > 0) {
84        target = argv._;
85      } else {
86        target.push("localhost:50051");
87      }
88
89      const randomNumbers = generateRandomNumbers(NUMBERS_LENGTH);
90
91      var result = distributedService(target, randomNumbers);
```

```
49      indexBase = Math.floor(NUMBERS_LENGTH / clientSize);
50      var endNumberIndex = indexBase;
51
52      client.forEach((c, index) => {
53        if (index == clientSize - 1) endNumberIndex = NUMBERS_LENGTH;
54
55        const worker = new Worker(__filename);
56        worker.once("message", (message) => {
57          if (message.min < min) min = message.min;
58          if (message.max > max) max = message.max;
59          finishedWorkers++;
60          if (finishedWorkers === clientSize)
61            console.log("Menor: ", min.toFixed(3), "\nMaior: ", max.toFixed(3));
62        });
63        worker.on("error", console.error);
64        console.log(
65          `worker "${c}" inicio "${initialNumberIndex}" fim "${endNumberIndex}"`
66        );
67
68        worker.postMessage({
69          ipPort: c,
70          numbers: numberList.slice(initialNumberIndex, endNumberIndex),
71        });
72
73        initialNumberIndex = endNumberIndex + 1;
74        endNumberIndex += indexBase;
75      });
76    }
```

# Client Threads Auxiliares

```
99    } else {
100     parentPort.once("message", (message) => {
101       const numbers = message.numbers;
102
103       const client = new finder_proto.Finder(
104         message.ipPort,
105         grpc.credentials.createInsecure()
106       );
107
108       client.calculateMinMax(
109         {
110           numbers: numbers,
111         },
112         function (err, response) {
113           parentPort.postMessage({ min: response.min, max: response.max });
114         }
115       );
116     });
117   }
118 }
```

# Agradecemos!


Até a próxima!!

# Referências


https://grpc.io/

https://developers.google.com/protocol-buffers

https://web.dev/performance-http2/

https://www.educba.com/protobuf-vs-json/

https://king.host/blog/2017/04/o-que-e-o-http2-e-quais-os-seus-beneficios/