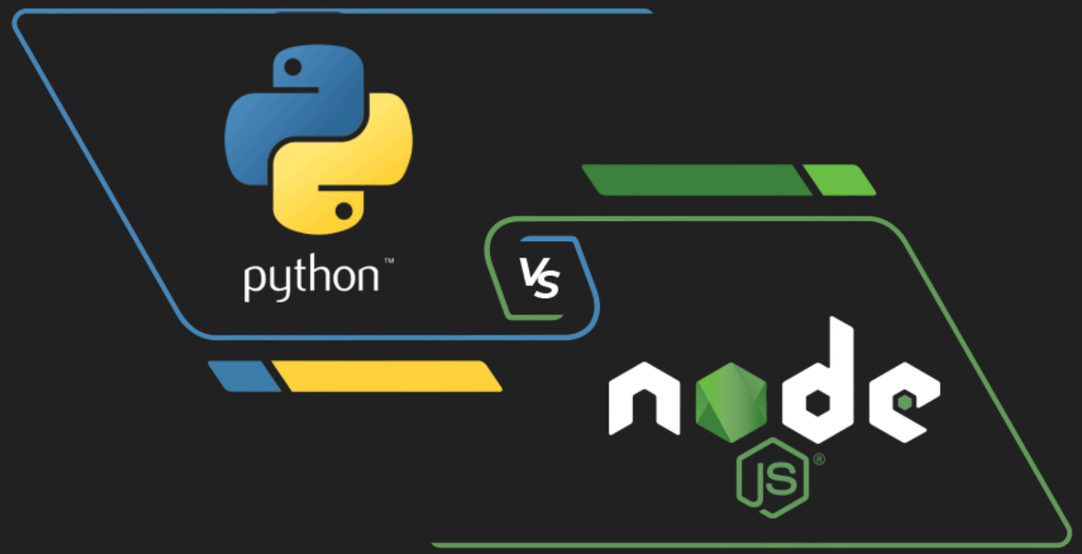
SISTEMAS DISTRIBUÍDOS

QUIZ INTERATIVO USANDO GRPC

COMPONENTES:

LUIZ ROBERTO E LUIZ FERNANDO

DIVISÃO DE RESPONSABILIDADES



JAVASCRIPT

Servidor: Node.js (JavaScript) - O Back-end

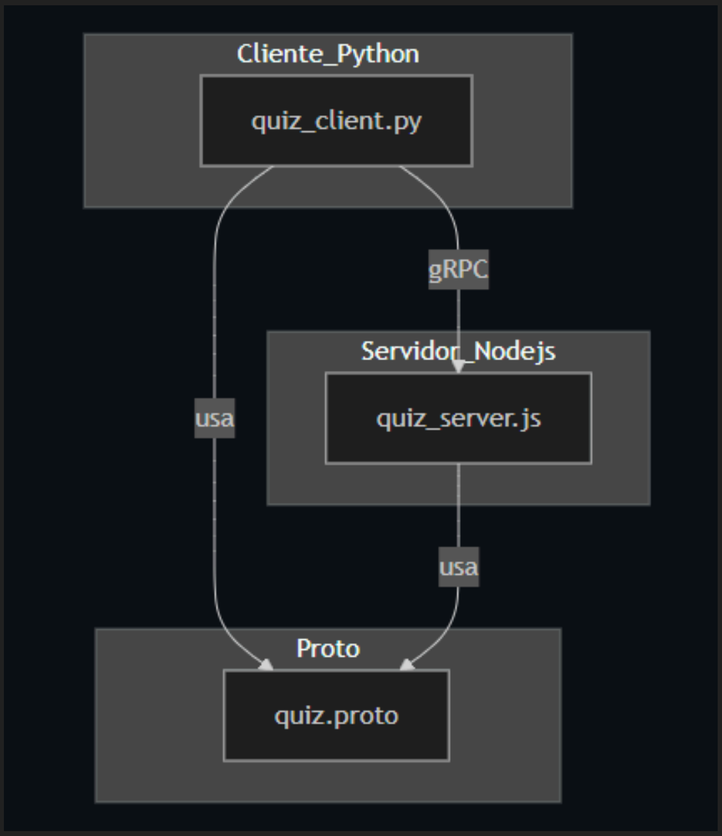
- Função: Gerenciador da Lógica de Negócio.
- Responsabilidades:
  - Implementar os serviços gRPC (as "funções" que podem ser chamadas remotamente).
  - Manter o estado do quiz em memória (perguntas, respostas, sessões).
  - Processar as respostas enviadas pelo cliente e calcular a pontuação.
  - Servir como a fonte autoritativa da verdade para o jogo.

PYTHON

Cliente: Python - A Interface com o Usuário

- Função: Camada de Apresentação e Interação.
- Responsabilidades:
  - Conectar-se ao servidor gRPC.
  - Exibir as perguntas e as opções para o usuário no terminal.
  - Capturar a entrada do usuário (a resposta).
  - Invocar os métodos remotos no servidor para enviar a resposta e pedir a próxima pergunta.
  - Apresentar o resultado final.

EXPLICANDO A ARQUITETURA



Componente Central: O Arquivo .proto

- Funciona como um contrato universal entre o cliente e o servidor.
- Define os serviços (QuizService) e os métodos disponíveis (StartQuiz, GetQuestion, etc.).
- Define as mensagens (os dados) que serão trocadas, garantindo que Python e Node.js "falem a mesma língua".

Comunicação via gRPC:

- Baseado em HTTP/2, o que o torna muito mais rápido e eficiente que alternativas como REST com JSON.
- A comunicação é binária e fortemente tipada, reduzindo o tamanho dos dados trafegados e evitando erros de serialização.
- Permite um modelo de programação natural (RPC), onde o cliente chama uma função no servidor como se fosse local.

Fluxo de Geração de Código:

1. O arquivo quiz.proto é usado como entrada.
2. As ferramentas do gRPC geram o código-base (stubs) em Python e em JavaScript.
3. Esse código gerado lida com toda a complexidade da comunicação de rede, permitindo que o desenvolvedor foque apenas na lógica da aplicação.

GRPC

```
def start_quiz(stub):
    # CRIAÇÃO DA MENSAGEM PROTOBUF
    req = quiz_pb2.StartQuizRequest(player_name=player_name)
    resp = stub.StartQuiz(req)  # ← AQUI ACONTECE A COMUNICAÇÃO gRPC
    return resp.session_id, resp.total_questions
```

O que acontece internamente:

1. `StartQuizRequest` é **serializada** em Protocol Buffers
2. Dados são **enviados via HTTP/2** para o servidor
3. Servidor **deserializa** e processa
4. Resposta é **serializada** e enviada de volta
5. Cliente **deserializa** `StartQuizResponse`

STUBS DO GRPC

Definição Simples:

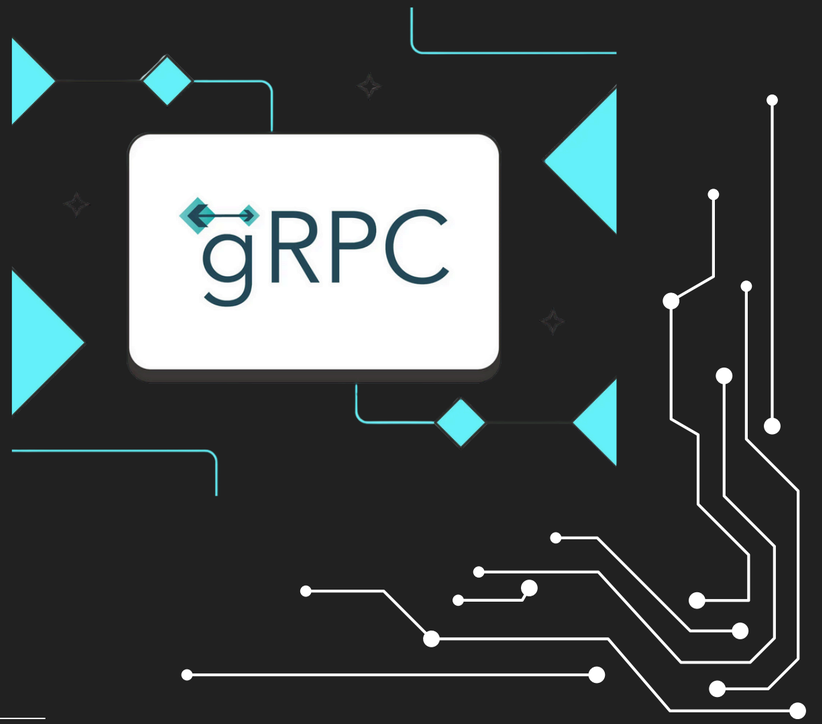
STUB = "Representante Local de um Serviço Remoto"

Imagine que você quer ligar para alguém em outro país. O STUB é como seu telefone - você disca localmente, mas ele se conecta automaticamente com o destino remoto.

EXPLICANDO O PROJETO

Como o Quiz Funciona?

- Fluxo do Usuário:
  - O usuário inicia o cliente Python no seu terminal.
  - O cliente se conecta ao servidor, inicia uma sessão de quiz e solicita a primeira pergunta.
  - O usuário envia as respostas, uma a uma.
  - Ao final, o cliente exibe a pontuação total recebida do servidor.
- Lógica do Sistema:
  - O servidor Node.js é o "cérebro" do sistema: ele controla as perguntas, valida as respostas e gerencia o estado do jogo (pontuação, pergunta atual).
  - O cliente Python é apenas a "interface": ele não contém nenhuma lógica sobre o quiz, apenas exibe o que o servidor manda e envia o que o usuário digita.
- Interação Chave:
  - Toda a comunicação, desde "iniciar o quiz" até "receber o resultado final", acontece por meio de chamadas de procedimento remoto (RPC) via gRPC.



NODE . JS (SERVIDOR)

```javascript
function main() {
  const server = new grpc.Server();
  server.addService(quizProto.QuizService.service, quizService);
  server.bindAsync(`0.0.0.0:${PORT}`, grpc.ServerCredentials.createInsecure(), (err, port) => {
    if (err) {
      console.error('Erro ao iniciar o servidor:', err);
      return;
    }
    server.start();
    console.log(`Servidor gRPC Quiz rodando na porta ${port}`);
  });
}
```

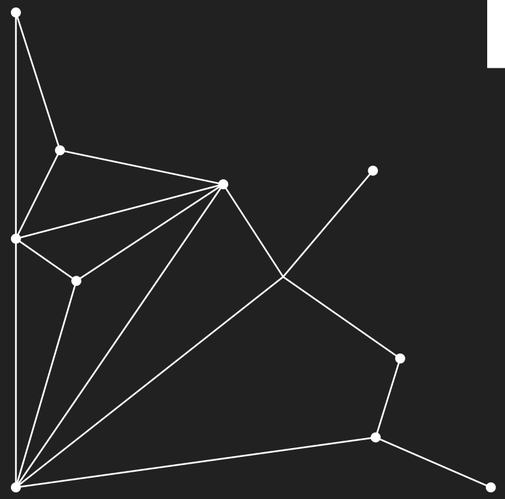
```python
def start_quiz(stub):
    player_name = input('Digite seu nome: ')
    req = quiz_pb2.StartQuizRequest(player_name=player_name)
    resp = stub.StartQuiz(req)
    print(Fore.CYAN + resp.message)
    return resp.session_id, resp.total_questions

def get_question(stub, session_id):
    req = quiz_pb2.GetQuestionRequest(session_id=session_id)
    return stub.GetQuestion(req)

def submit_answer(stub, session_id, answer_index):
    req = quiz_pb2.SubmitAnswerRequest(session_id=session_id, answer_index=answer_index)
    return stub.SubmitAnswer(req)

def finish_quiz(stub, session_id):
    req = quiz_pb2.FinishQuizRequest(session_id=session_id)
    return stub.FinishQuiz(req)

```

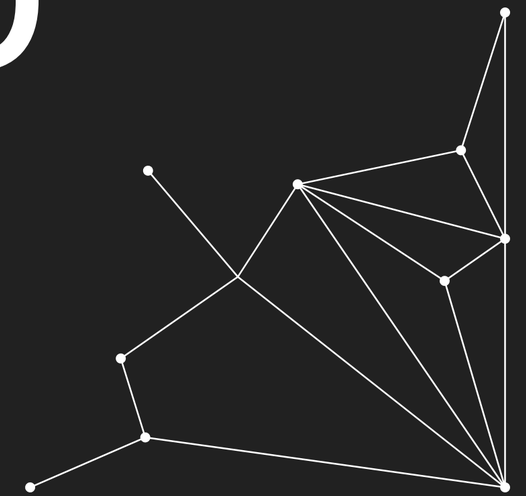
PYTHON (CLIENTE)

CÓDIGO PROTO

```
// Serviço principal do Quiz
service QuizService {
  // Inicia uma nova sessão de quiz
  rpc StartQuiz(StartQuizRequest) returns (StartQuizResponse);

  // Obtém a próxima pergunta
  rpc GetQuestion(GetQuestionRequest) returns (GetQuestionResponse);

  // Submete uma resposta
  rpc SubmitAnswer(SubmitAnswerRequest) returns (SubmitAnswerResponse);

  // Finaliza o quiz e obtém resultado
  rpc FinishQuiz(FinishQuizRequest) returns (FinishQuizResponse);
}
```

OBRIGADO PELA ATENÇÃO





https://github.com/luiizr/gRPC-Quiz