Estudio de gRPC como tecnología para implementar llamadas a procedimientos remotos

Pelayo Iglesias Manzano – UO266600

César Llano Rodríguez – UO289969

Pelayo Calleja Villafañe – UO292393

Martín Braña Peralvo – UO276268

Tabla de contenidos

01

Fundamentos del RPC

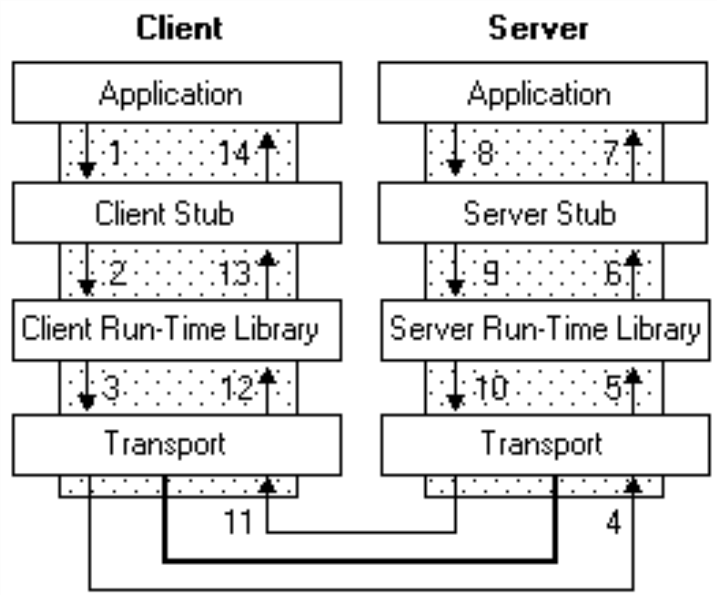
- Qué es y cómo funciona
- Características clave
- Tecnologías RPC

Qué es y cómo funciona

Es un protocolo que permite a un programa informático ejecutar un procedimiento o función en otro ordenador o servidor remoto.

Utiliza un modelo cliente-servidor :

- **El cliente inicia una petición.**
- **El servidor la recibe.**
- **El servidor localiza y ejecuta el procedimiento solicitado.**
- **Devuelve el resultado al cliente.**



Características clave

- Transparencia de localización

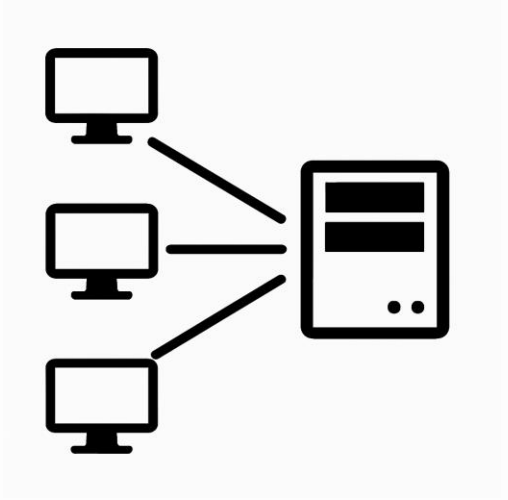
Oculta la ubicación del servidor al cliente, ya que cliente no necesita conocer la dirección física o de red del servidor.

- Transparencia de migración

El cliente no necesita saber en qué máquina está realmente ejecutándose el procedimiento remoto que ha invocado. Siempre llama a la función de la misma forma, como si fuera local.

CORBA

- Componentes software escritos en diferentes lenguajes y corriendo sobre distintos SO podían trabajar en conjunto.



JAVA RMI

- Trabaja con objetos, permitiendo enviar referencias a objetos remotos y aprovechar las características de la programación orientada a objetos.

.NET Remoting

- Es una tecnología desarrollada por Microsoft que permite la comunicación entre objetos en diferentes procesos o máquinas, dentro del entorno .NET.

02

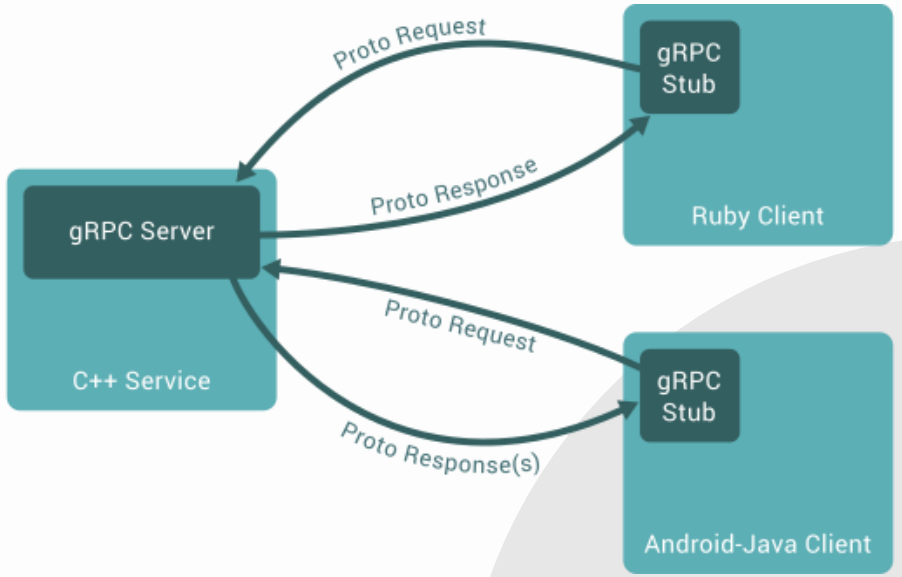
¿Qué es gRPC?

- HTTP/2

¿Qué es gRPC?

Es una implementación de llamadas a procedimiento remoto (RPC) diseñado originalmente por Google.

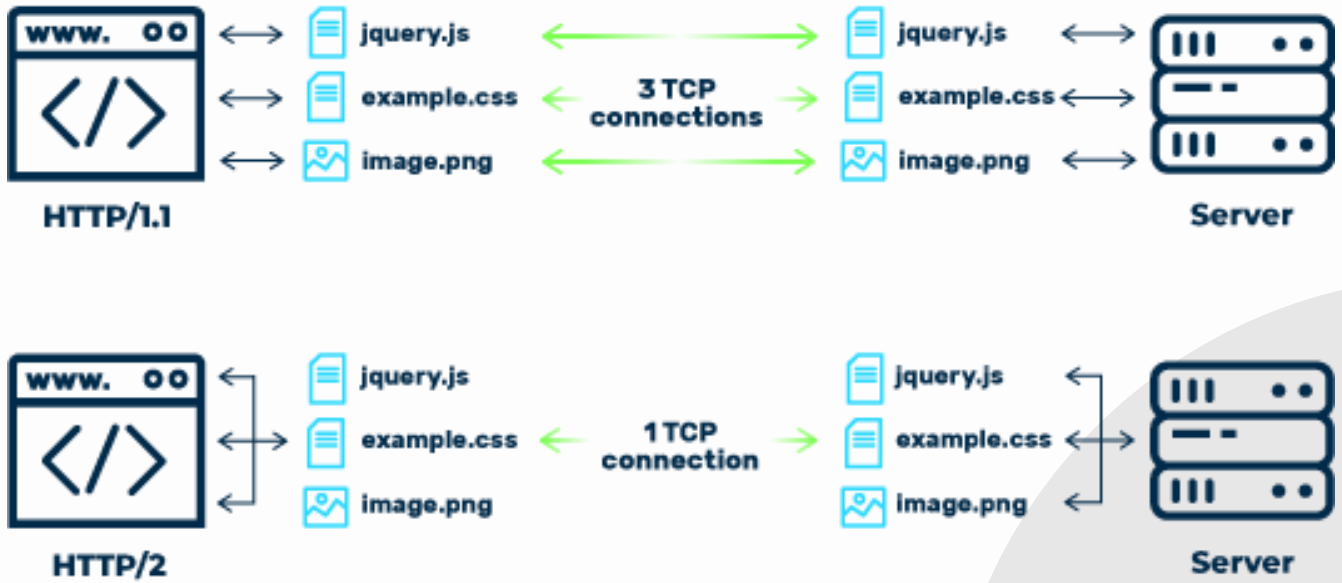
Surgió como una necesidad de modernizar las comunicaciones entre servicios distribuidos.



HTTP/2

Es un protocolo de comunicación que permite la transferencia de información en internet.

- **Multiplexación**
- **Compresión de cabeceras**
- **Formato binario**



03

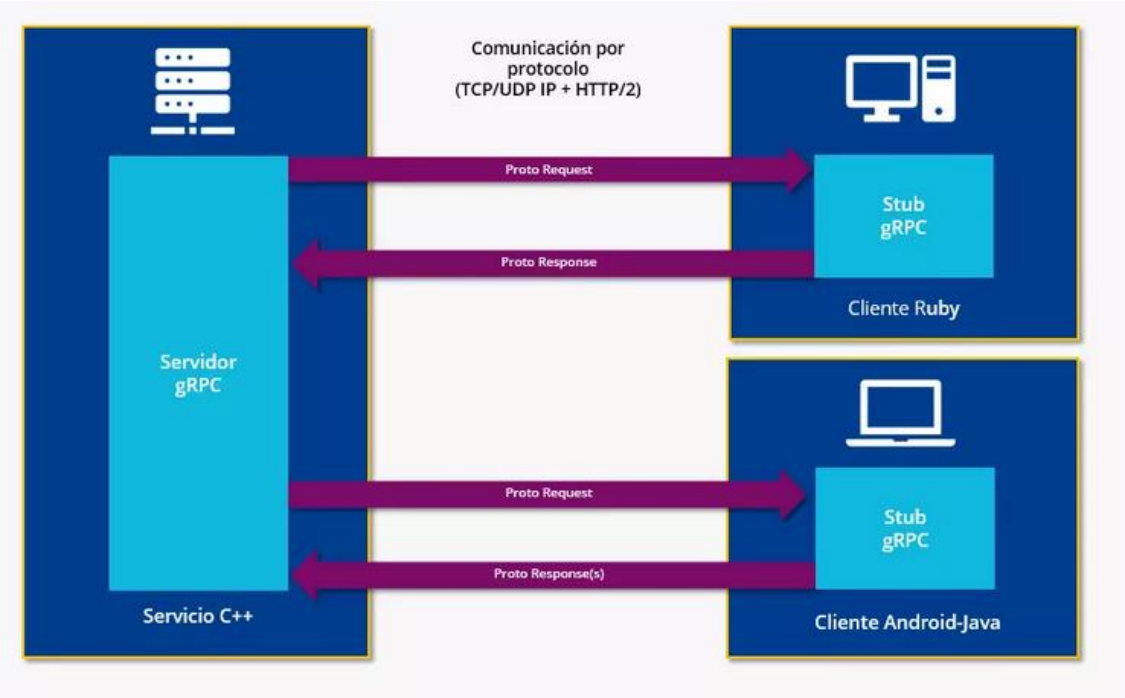
Arquitectura de gRPC

- Componentes principales
- Protocol Buffers
- Flujo de una llamada gRPC
- Tipos de llamadas

Componentes Principales

Los componentes principales de la arquitectura gRPC son:

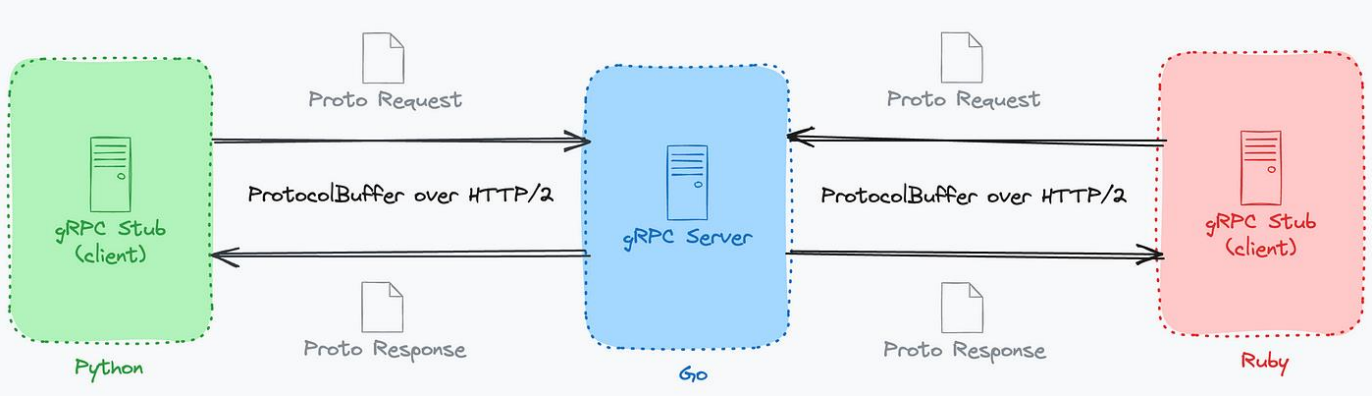
- **Cliente**
- **Servidor**
- **STUB**
- **Service Definition**



Protocol Buffers

- Es un formato de serialización de datos desarrollado por Google para estructurar y serializar datos de manera eficiente. Se utiliza principalmente para la comunicación entre sistemas distribuidos, como en el caso de gRPC.

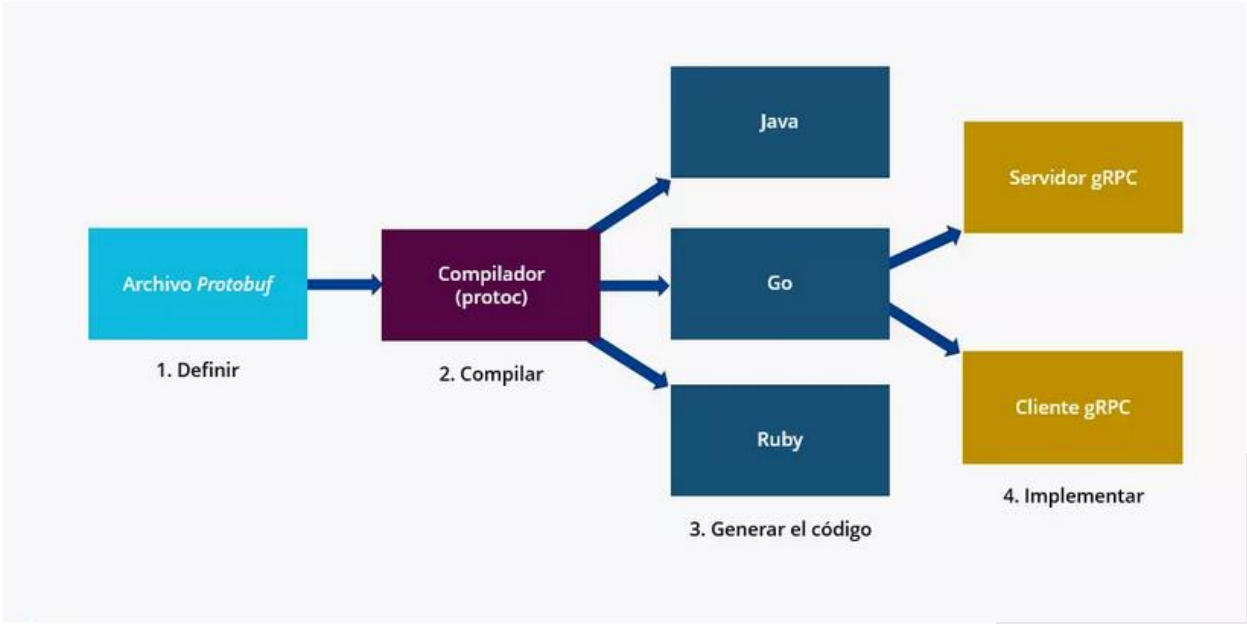
- Los datos se definen utilizando un lenguaje de esquema simple y legible llamado "Protocol Buffers Language" o "proto". En este esquema, se defines las estructuras de datos que se van a serializar, incluyendo los tipos de datos y sus nombres.



Flujo de una llamada

El flujo de una llamada se divide en :

- **Definición del contrato de servicio**
- **Generación del código gRPC**
- **Implementación del servidor**
- **Creación del stub del cliente**



Tipos de llamadas

Hay 4 tipos de llamadas:

- **Unary RPC**
- **Server Streaming RPC**
- **Client Streaming RPC**
- **Bidirectional Streaming RPC**

04

Ventajas y desventajas

- Ventajas
- Desventajas

Ventajas

- Rendimiento
- Mejora en el streaming
- Estrategia API First
- Interoperabilidad
- Seguridad

Desventajas

- Legibilidad por parte del usuario
- Limitación con navegadores

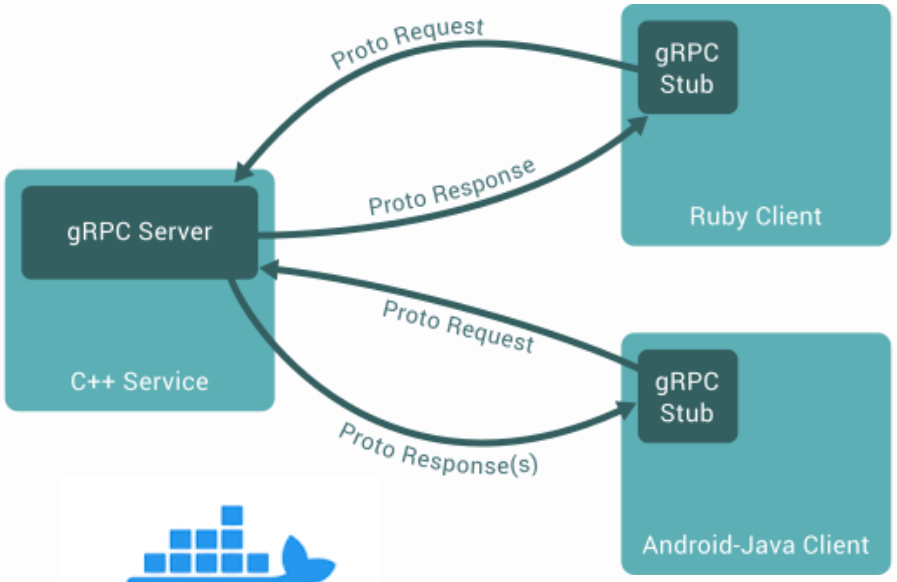
05

Casos de Uso

- Microservicios y Cloud-Native
- Comunicación eficiente entre servicios internos
- Aplicaciones en tiempo real
- Empresas y Tecnologías que usan gRPC
- Interoperabilidad entre lenguajes

Microservicios y Cloud-Native

- Ideal para la comunicación de microservicios desplegados en Docker, Kubernetes, etc.
- Baja latencia gracias a HTTP/2
- Compatible con diferentes lenguajes



kubernetes

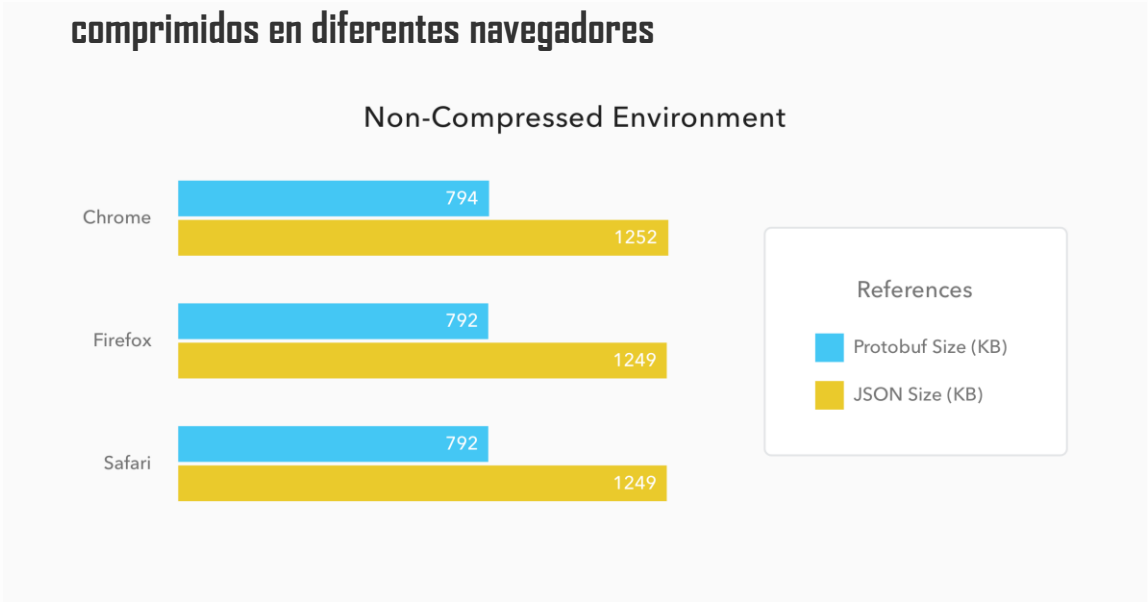
docker

Comunicación eficiente entre servicios internos

Utilizado por diferentes empresas para la comunicación interna entre componentes de sus sistemas.

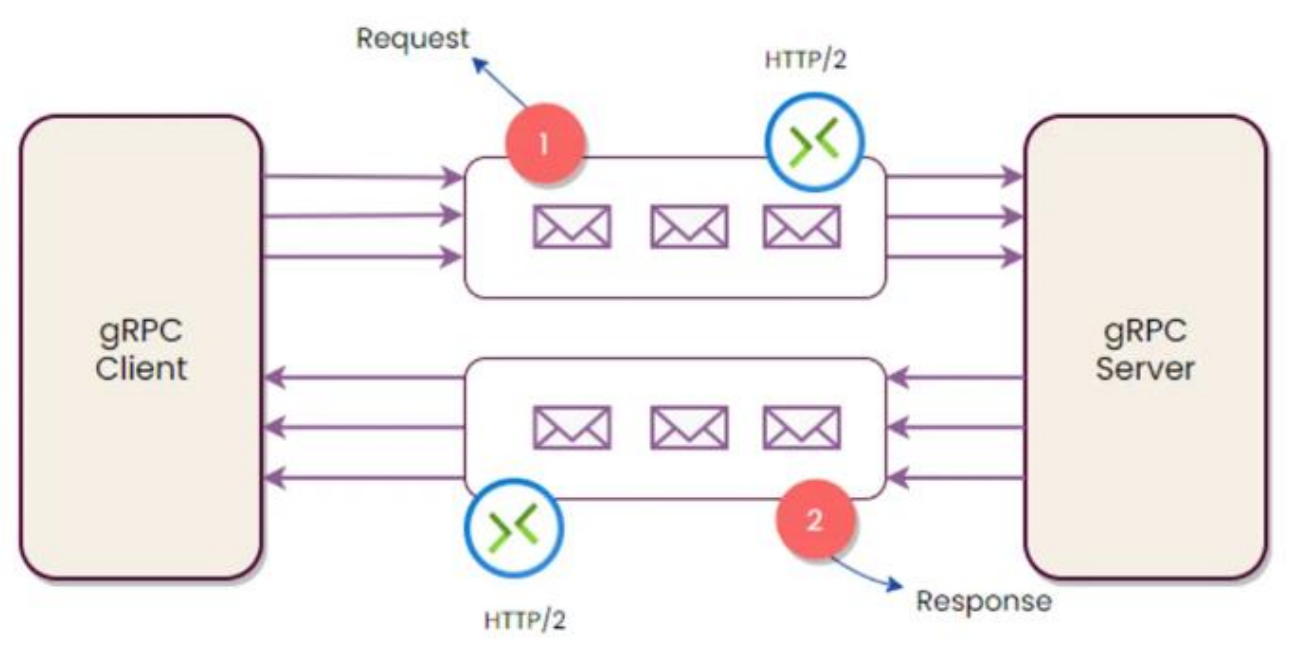
A diferencia de REST:

- Reduce el tamaño de los mensajes
- Reduce la sobrecarga de la red
- Reemplaza el uso de REST + JSON



Aplicaciones en tiempo real

Las aplicaciones en tiempo real que hacen uso de gRPC se benefician de su baja latencia y eficiencia. Además de su soporte para streaming unidireccional y bidireccional



Empresas que usan gRPC

- Desarrollado por Google, es utilizado en Google Cloud Platform (GCP)
- Gran parte de la comunicación interna de Netflix usa gRPC
- Servicios como DropBox, Square o Cisco han migrado de REST a gRPC

Google Cloud

Square

CISCO

Interoperabilidad entre varios lenguajes

Permite la comunicación de servicios escritos en diferentes lenguajes de programación gracias al uso de Protocol Buffers.
Tiene soporte nativo para lenguajes como:

- Java
- Python
- Go
- C#
- C++
- Etc.

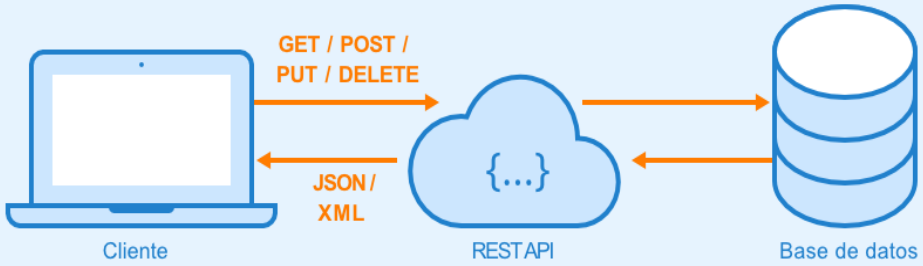
06

Comparaciones con otras tecnologías

- gRPC frente a REST
- gRPC frente a GraphQL
- gRPC frente a SOAP

gRPC frente a REST

- Utiliza formatos como JSON o XML
- Sencillez de implementación
- Trabajar con formatos de texto introduce una sobrecarga



Principales diferencias:

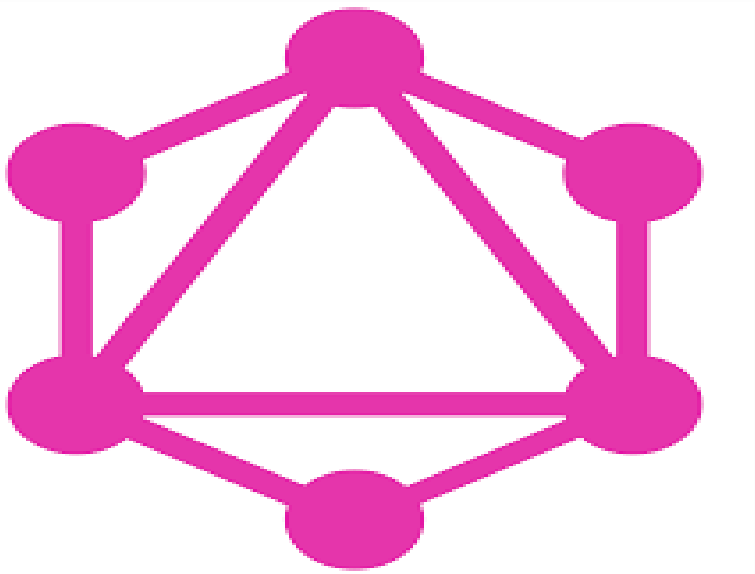
- Protocolo
- Formato de datos
- Rendimiento
- Compatibilidad

gRPC frente a graphQL

- Tecnología que permite definir datos requeridos en una solicitud
- Limitaciones en términos de rendimiento.

Principales diferencias:

- Consultas
- Transporte
- Enfoque



gRPC frente a SOAP

- SOAP se basa en el intercambio de mensajes XML a través de HTTP
- Complejo y dependencia de XML

Principales diferencias:

- Formato
- Protocolos soportados
- Seguridad
- Caso de uso

08

Conclusiones

- Tecnología muy potente para la comunicación entre servicios
- Pese algunas limitaciones, pueden solucionarse con herramientas complementarias
- Práctico y flexible