Acelerando el Desarrollo de APIs con gRPC

Andres Pineda
Senior Cloud Developer
@pinedax_dev

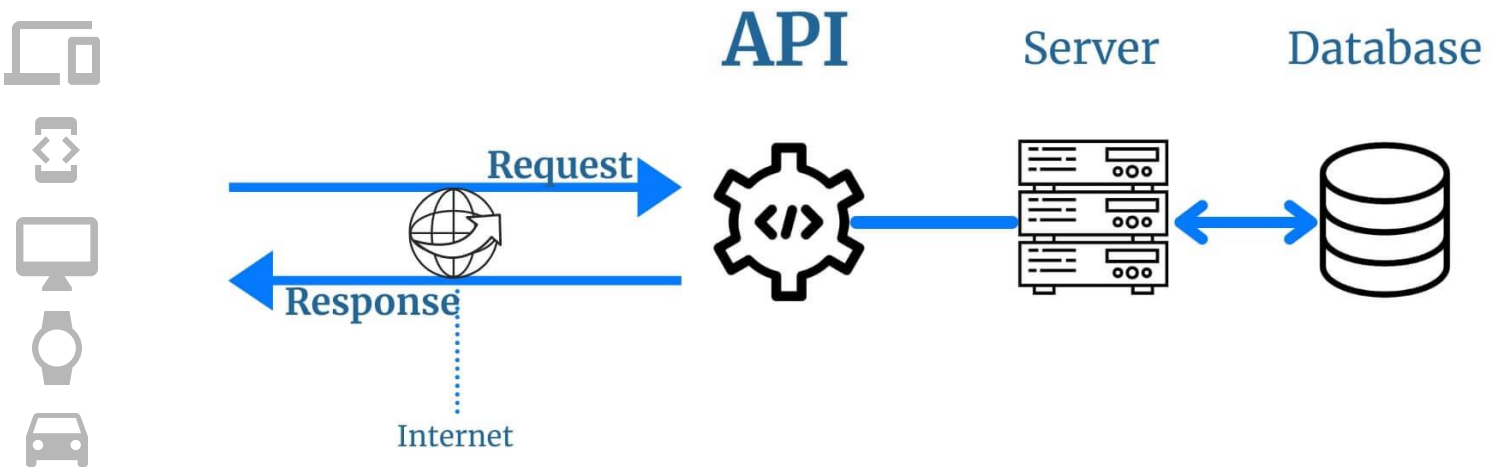
Que es un API?

Un conjunto de reglas y protocolos que permiten a diferentes aplicaciones comunicarse entre sí y compartir información de manera estructurada.

@pinedax_dev

@ajpinedam

Cómo funciona un API?



Soap and REST

@pinedax_dev

@ajpinedam

SOAP

XML

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <soapenv:Header/>
    <soapenv:Body>
        <m:UserDataResponse>
            <m:title>Warden of the North</ns2:title>
            <m:name>Jon Snow</ns2:name>
            <m:allegiance>House Stark</ns2:allegiance>
            <m:age>27</ns2:age>
            <m:religion>Old Gods of the Forest</ns2:religion>
            <m:email>jon.snow@gmail.com</ns2:email>
        </m:UserDataResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

525 bytes

REST

JSON / XML

```
{
    "title": "Warden of the North",
    "name": "Jon Snow",
    "allegiance" : "House Stark",
    "age": 27,
    "religion" : "Old Gods of the Forest",
    "email": "jon.snow@gmail.com"
}
```

192 bytes

Python Rest Frameworks

django REST framework

WEB2PY

FastAPI

Requests
http for humans

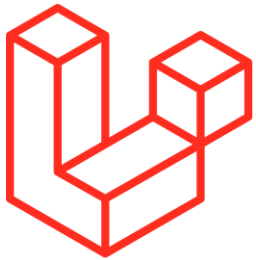
Flask

Bottle

@pinedax_dev

@ajpinedam

Otros REST Frameworks



@pinedax_dev

@ajpinedam

gRPC

@pinedax_dev

@ajpinedam

gRPC

@pinedax_dev

@ajpinedam

Pero antes de…

RPC Remote Procedure Call

Es un paradigma de programación que permite que un programa o proceso solicite la ejecución de un procedimiento o función en un sistema remoto, como si fuera una llamada local.

@pinedax_dev

@ajpinedam

gRPC

@pinedax_dev

@ajpinedam

gRPC


@pinedax_dev


@ajpinedam

Google Remote Procedure Call

Google Remote Procedure Call

1.0 gRPC
1.1 good
…
1.62 guardian
1.53 giggle

https://github.com/grpc/grpc/blob/master/doc/g_stands_for.md

Por que gRPC?

Clientes gRCP

- C++
- C#
- Dart
- Go
- Java
- Kotlin
- Node
- Objective-C
- PHP
- Python
- Ruby
- WebJS

@pinedax_dev @ajpinedam

Protobuf

Protocol Buffer

Protobuf

Protocol Buffer

Es un formato de **serialización** de datos que se utiliza para estructurar, almacenar y transmitir de manera **eficiente** y **compacta**.


@pinedax_dev


@ajpinedam

character.proto

```
syntax = "proto3";

enum Allegiance {
    STARK = 0;
    ARRYN = 1;
    BARATHEON = 2;
    TULLY = 3;
    GREYJOY = 4;
    LANNISTER = 5;
    TYREL = 6;
    MARTELL = 7;
    TARGARYEN = 8;
}

message CharacterRequest {
    int32 user_id = 1;
    Allegiance house = 2;
    int32 max_results = 3;
}

message Character {
    int32 id = 1;
    string title = 2;
    string name = 3;
    Allegiance house = 4;
    int32 age = 5;
    string religion = 6;
    string email = 7;
}

message CharacterResponse {
    repeated Character characters = 1;
}

service GOTCharacter {
    rpc GetCharacter (CharacterRequest) returns (CharacterResponse);
}
```

character.proto

```
syntax = "proto3";

enum Allegiance {
    STARK = 0;
    ARRYN = 1;
    BARATHEON = 2;
    TULLY = 3;
    GREYJOY = 4;
    LANNISTER = 5;
    TYREL = 6;
    MARTELL = 7;
    TARGARYEN = 8;
}

message CharacterRequest {
    int32 user_id = 1;
    Allegiance house = 2;
    int32 max_results = 3;
}

message Character {
    int32 id = 1;
    string title = 2;
    string name = 3;
    Allegiance house = 4;
    int32 age = 5;
    string religion = 6;
    string email = 7;
}

message CharacterResponse {
    repeated Character characters = 1;
}

service GOTCharacter {
    rpc GetCharacter (CharacterRequest) returns (CharacterResponse);
}
```

Tipos

- double
- float
- int32
- int64
- uint32
- uint64
- sint32 / sint64
- fixed32/ fixed64
- sfixed32 / sfixed64
- bool
- string
- bytes

Mapping de tipos

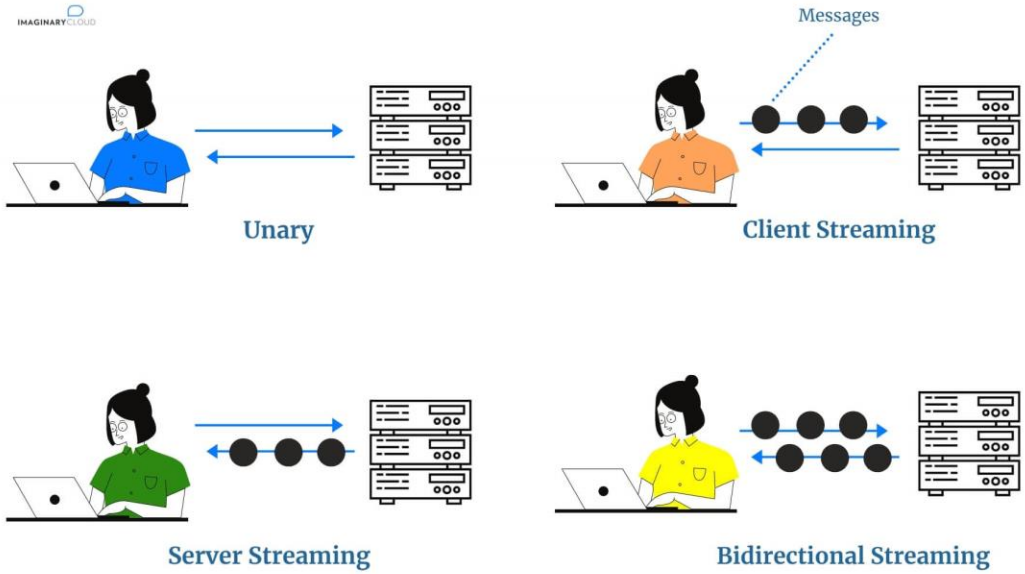
.proto Type	Notes	C++ Type	Java/ Kotlin Type[1]	Python Type[3]	Go Type	Ruby Type	C# Type	PHP Type
double		double	double	float	float64	Float	double	float
float		float	float	float	float32	Float	float	float
int32	Uses variable-length encoding. Inefficient for encoding negative numbers – if your field is likely to have negative values, use sint32 instead.	int32	int	int	int32	Fixnum or Bignum (as required)	int	integer
int64	Uses variable-length encoding. Inefficient for encoding negative numbers – if your field is likely to have negative values, use sint64 instead.	int64	long	int/long[4]	int64	Bignum	long	integer/ string[6]
uint32	Uses variable-length encoding.	uint32	int[2]	int/long[4]	uint32	Fixnum or Bignum (as required)	uint	integer
uint64	Uses variable-length encoding.	uint64	long[2]	int/long[4]	uint64	Bignum	ulong	integer/ string[6]
fixed32	Always four bytes. More efficient than uint32 if values are often greater than 2^{28}.	uint32	int[2]	int/long[4]	uint32	Fixnum or Bignum (as required)	uint	integer
fixed64	Always eight bytes. More efficient than uint64 if values are often greater than 2^{56}.	uint64	long[2]	int/long[4]	uint64	Bignum	ulong	integer/ string[6]
sfixed32	Always four bytes.	int32	int	int	int32	Fixnum or Bignum (as required)	int	integer
sfixed64	Always eight bytes.	int64	long	int/long[4]	int64	Bignum	long	integer/ string[6]
bool		bool	boolean	bool	bool	TrueClass/ FalseClass	bool	boolean
string	A string must always contain UTF-8 encoded or 7-bit ASCII text, and cannot be longer than 2^{32}.	string	String	str/ unicode[5]	string	String (UTF-8)	string	string
bytes	May contain any arbitrary sequence of bytes no longer than 2^{32}.	string	ByteString	str (Python 2) bytes (Python 3)	[]byte	String (ASCII-8BIT)	ByteString	string

https://protobuf.dev/programming-guides/proto3/#scalar

Tipo de Iteraciones

Un servicio RPC puede implementarse de distintas formas:

- Urinary
- Server Streaming
- Client Streaming
- Bidirectional Streaming


IMAGINARYCLOUD
Unary
Messages
Client Streaming
Server Streaming
Bidirectional Streaming

Interceptors

Son componentes de middleware que permiten agregar funcionalidades comunes a los servicios gRPC de manera modular y reutilizable.

Algunos ejemplos de Interceptors

Metadata handling
Logging
Fault injection
Caching
Metrics
Policy enforcement
Server-side Authentication
Server-side Authorization

@pinedax_dev
@ajpinedam

Conclusión

Gracias!!!

 @pinedax_dev

 @ajpinedam