A Evolução do REST
#Caipyra2019

Be Free. KDE

About me

@marcosptf

pytero -> Grupy-SP
javero -> NetCat

A Evolucao do REST

+Topicos da palestra:

-mostrar um pouco do historico dos Web Services;
-quais sao?
-quando foram criados?
-quais eram seus pontos positivos e negativos?
-o que eh?
-pra que serve?
-quais os problemas que ele veio resolver do REST?
-suas vantagens;
-como funciona?

Pq precisamos de API ?

A necessidade de transferir dados entre empresas como:
Bancos
Governos
Empresas de telefonia entre outras;

A necessidade de compartilhar informações entre elas de maneira, rapida, segura e eficaz;

Historia do FTP

Desde 16 de abril de 1971, quando Abhay Bhushan criou a RFC 114 para a criacao do File Trasfer Protocol, o nosso velho conhecido como Transferencia de arquivos via o servico FTP;

Trasferencia via FTP

Transferencias de arquivos entre empresas é usado desde a decada de 1970 ate os dias de hoje;

Principalmente por bancos, ainda é usado o sistema de transferencias de arquivos CNAB, veja este exemplo do banco da Caixa;

Trasferencia via FTP

CAIXA

MO **67118** 010

3.5. Composição do Arquivo Remessa

3.5.1. Registro Tipo 0 (Obrigatório) - Header de Arquivo Remessa

Campo	Nome do campo		Posição		"Picture"	Conteúdo	Descrição
			De	Até			
01.0	Controle	Código do Banco	1	3	9(003)	Preencher '104'	G001
02.0		Código do Lote	4	7	9(004)	Preencher '0000'	G002
03.0		Tipo de Registro	8	8	9(001)	Preencher '0' (equivale a Header de Arquivo)	G003
04.0	CNAB	Filler	9	17	X(009)	Preencher com espaços	G004
05.0	Empresa Beneficiária	Tipo de Inscrição do Beneficiário	18	18	9(001)	Preencher com o tipo de inscrição do Beneficiário: '1', se CPF (pessoa física); ou '2' se CNPJ (pessoa jurídica)	G005
06.0		Número de Inscrição do Beneficiário	19	32	9(014)	Ver Nota Explicativa G006	G006
07.0		Uso Exclusivo CAIXA	33	52	9(020)	Preencher com zeros	-
08.0		Agência Mantenedora da Conta	53	57	9(005)	Preencher com o código da agência detentora da conta, com um zero à esquerda	G008
09.0		Dígito Verificador da Agência	58	58	X(001)	Preencher com o dígito verificador da agência, informado pela CAIXA	G009
10.0		Código do Beneficiário	59	64	9(006)	Ver Nota Explicativa G007	G007
11.0		Uso Exclusivo CAIXA	65	71	9(007)	Preencher com zeros	-
12.0			72	72	9(001)		-
13.0		Nome da Empresa	73	102	X(030)	Ver Nota Explicativa G013	G013
14.0	Banco Beneficiário	Nome do Banco	103	132	X(030)	Ver Nota Explicativa G014	G014
15.0	CNAB	Filler	133	142	X(010)	Preencher com espaços	G004
16.0		Código Remessa / Retorno	143	143	9(001)	Preencher '1'	G015

Trasferencia via FTP

CAIXA

MO **67118** 010

3. ESTRUTURA DO ARQUIVO CNAB 240

3.1. Composição do Arquivo

O padrão dos arquivos de remessa e retorno segue o estabelecido pelo CNAB (Centro Nacional de Automação Bancária), e deve ser gravado contendo um registro header de arquivo, lotes do Serviço/Produto e um registro trailer de arquivo, conforme ilustra a figura abaixo:

ARQUIVOS	**Registro Header de Arquivo**		(Tipo = 0)
	LOTES	Registro Header de Lote	(Tipo = 1)
		Registros de detalhe Segmentos ...	(Tipo = 3)

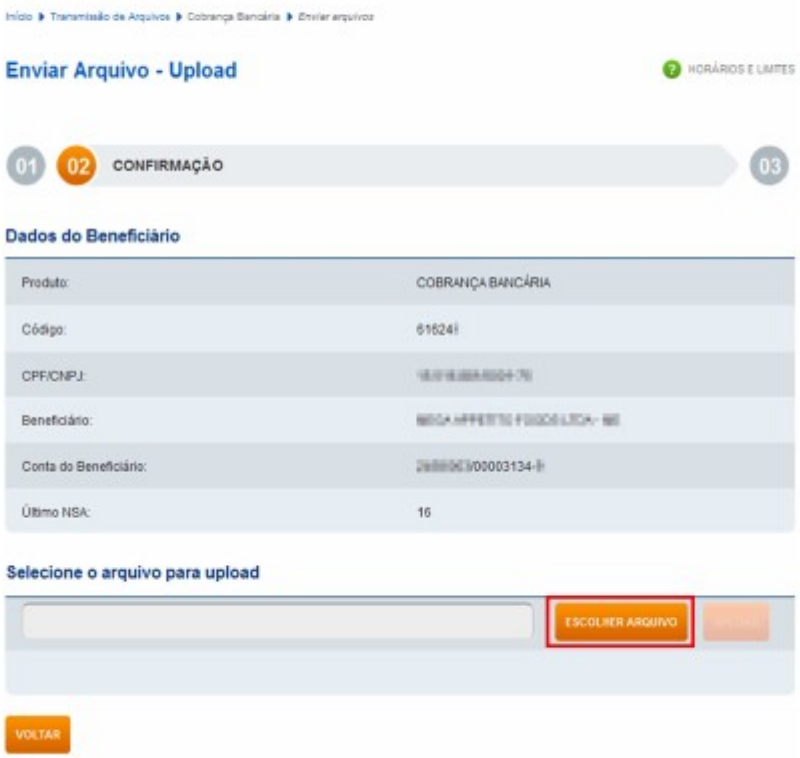
Trasferencia via FTP

TRANSMISSÃO DE ARQUIVOS DE
COBRANÇA BANCÁRIA NO INTERNET BANKING CAIXA

CAIXA

3.3. Enviar Arquivos

Após a seleção do beneficiário, é apresentada a tela da opção de envio (upload) de arquivo remessa. É permitido o envio de 1 (um) arquivo por vez, respeitando os horários de envio (das 7h30 às 17h59, de segunda a sexta-feira, exceto feriados) e no formato esperado pelo Internet Banking CAIXA (leiaute da cobrança bancária CNAB 240 ou 400, formato texto sem formatação, extensão *.rem* ou *.txt*).



15

Be Free. KDE

Exploits de segurança FTP

Mas como o FTP nao conversa diretamente com sistemas e/ou aplicacoes, nao podemos chama-los de web-services, fora que ele possui inumeras vulnerabilidades:

Brute force attack

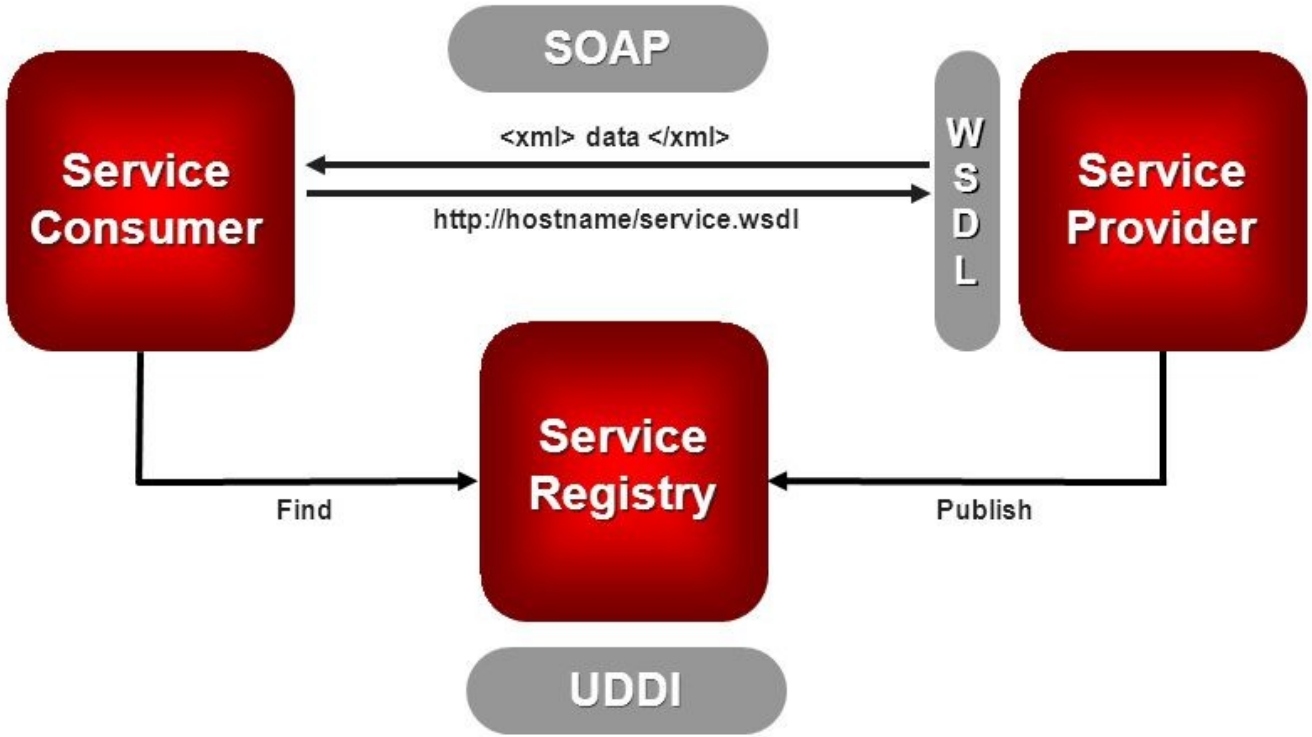
FTP bounce attack

Packet capture

Exploits de segurança FTP

- Port stealing (guessing the next open port and usurping a legitimate connection)
- Spoofing attack
- Username enumeration

SOA – Web Services



SOA – Web Services

O termo "Service-Oriented Architecture" (SOA) ou Arquitetura Orientada a Serviços expressa um conceito no qual aplicativos ou rotinas são disponibilizadas como serviços em uma rede de computadores (Internet ou Intranets) de forma independente e se comunicando através de padrões abertos.

SOA – Web Services

O SOA foi criado como um conceito no qual os apps ou jobs pudesse ser disponibilizados como servicos em uma rede decomputadores de forma independente e se comunicando atraves de padroes abertos; Existem diversas maneiras de se implementar SOA:

-SOAP

-WSDL

-WADL

-REST

-GraphQL

SOA – Web Services

Agora vamos olhar cada um, analisar seu uso e também qual a sua vantagem e desvantagem de uso de cada Web Service SOA

WSDL - Web Services

WSDL – Web Services

A Web Services Description Language (WSDL) é uma linguagem baseada em XML utilizada para descrever Web Services funcionando como um contrato do serviço, usando o protocolo SOAP;

Foi submetida ao W3C por Ariba, IBM e Microsoft em março de 2001 sendo que seu primeiro rascunho foi disponibilizado em julho de 2002.

WSDL – Web Services

Como ele basicamente funciona?

Primeiro precisa ser criado as funcoes para que as mesmas sejam expostas pela API;

Depois criar as assinaturas de cada funcao precisa receber em seus parametros;

Para expor, precisa de um endpoint.wsdl onde neste existe em xml a descricao do que esta sendo exposto pelo Web Service SOAP;

WSDL – Web Services

Anatomia de um WSDL

SOAP-ENV: Envelope

O envelope, que define o conteúdo da mensagem e informa como processá-la;

SOAP-ENV: Header

um conjunto de regras de codificação para os tipos de dados;

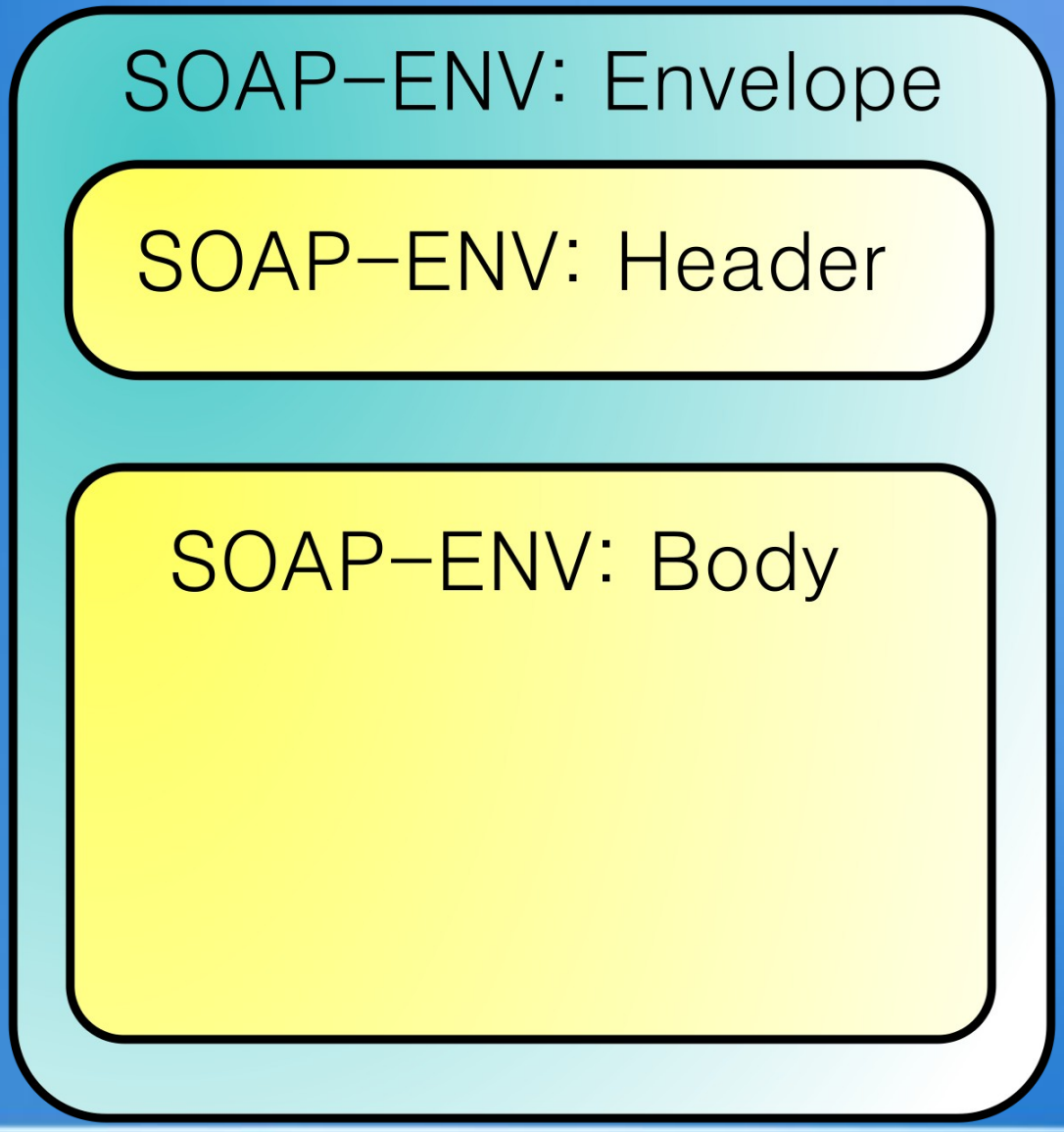
WSDL – Web Services

Anatomia de um WSDL

SOAP-ENV: Body

o layout para os procedimentos de chamadas e respostas.

WSDL – Web Services



WSDL – Web Services

```
<!--
   Published by JAX-WS RI (http://jax-ws.java.net). RI's version is Metro/2.3.2-b608 (trunk-7979; 2015-01-21T12:50:19+0000) JAXWS-RI/2.2.11-b150120.1832 JAXWS-API/2.2.12 JAXB-RI/2.2.1
-->
<!--
   Generated by JAX-WS RI (http://jax-ws.java.net). RI's version is Metro/2.3.2-b608 (trunk-7979; 2015-01-21T12:50:19+0000) JAXWS-RI/2.2.11-b150120.1832 JAXWS-API/2.2.12 JAXB-RI/2.2.1
-->
<definitions targetNamespace="http://ws.loja.fsvas.com/" name="ProductWebService">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://ws.loja.fsvas.com/" schemaLocation="http://timseguros.whitelabel.com.br:80/Loja-war/ProductWebService?xsd=1"/
    </xsd:schema>
  </types>
  <message name="createUser">
    <part name="parameters" element="tns:createUser"/>
  </message>
  <message name="createUserResponse">
    <part name="parameters" element="tns:createUserResponse"/>
  </message>
  <message name="Exception">
    <part name="fault" element="tns:Exception"/>
  </message>
  <message name="createNewSessionSecret">
    <part name="parameters" element="tns:createNewSessionSecret"/>
  </message>
  <message name="createNewSessionSecretResponse">
    <part name="parameters" element="tns:createNewSessionSecretResponse"/>
  </message>
```

WSDL – Web Services

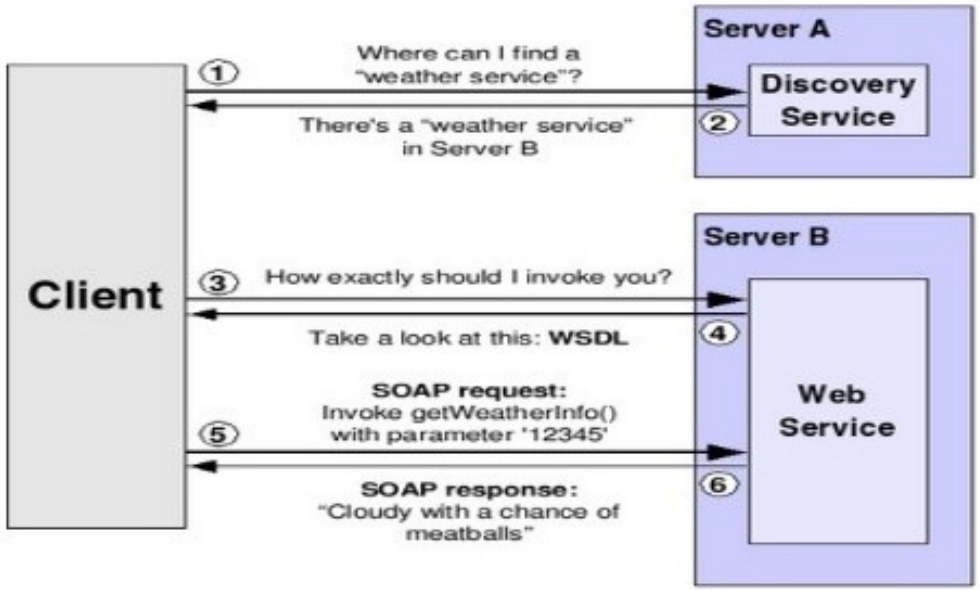
Esse "envelope" é enviado por meio de (por exemplo) HTTP/HTTPS.

E uma RPC (Remote Procedure Call) é executada, e o envelope retorna com as informações do documento XML formatado.

Usando um endpoint, todos os requests sao enviados usando o protocolo SOAP via HTTP GET;

WSDL – Web Services

WSDL example



WSDL – Web Services

+Vantagens

-Ao criar funcionalidades para chamar determinadas funcionalidades de um Web Service WSDL, o xml

descreve todas as funcoes publicas que ele deixa disponivel para chamadas;

-funciona com a maioria das linguagens de programacao atuais;

-Uma das vantagens do SOAP é o uso de um método de transporte "genérico".

WSDL – Web Services

+Vantagens

o SOAP pode usar qualquer meio de transporte existente para enviar sua requisição, desde SMTP até mesmo JMS (Java Messaging Service).

-SOAP é bastante maduro e bem definido e vem com uma especificação completa.

WSDL – Web Services

+Vantagens

-WSRM - WebService Reliable Messaging - IBM

Processamento e chamada assíncronos: se o aplicativo precisa de um nível garantido de confiabilidade e segurança para a troca de mensagens, então o SOAP 1.2 oferece padrões adicionais para esse tipo de operação como por exemplo o WSRM (WS-Reliable Messaging).

WSDL – Web Services

+Vantagens

-WS-BPEL - WebService Business Process Execution Language - IBM

implementacao e especifico implementar WSDL com algoritimos de criptografia assinatura nas mensagens;

https://www.ibm.com/developerworks/br/webservices/tutorials/ws-understand-web-services4/

WSDL – Web Services

+Vantagens

-Contratos formais: se ambos os lados (fornecedor e consumidor) têm que concordar com o formato de intercâmbio de dados, então o SOAP 1.2 fornece especificações rígidas para esse tipo de interação.

WSDL – Web Services

+Vantagens

-Operações stateful: para o caso de o aplicativo precisar de informação contextual e gerenciamento de estado com coordenação e segurança, o SOAP 1.2 possui uma especificação adicional em sua estrutura que apoia essa necessidade (segurança, transações, coordenação etc.). Comparativamente, usar o REST exigiria que os desenvolvedores construíssem uma solução personalizada.

WSDL – Web Services

+Desvantagens

-descrever as coisas via arquivos xml sao bem grandes e verbosos;

-a implementacao de um web service WSDL eh custoso em tempo;

No entanto, uma desvantagem percebida no uso de XML é a sua natureza prolixa e o tempo necessário para analisar o resultado apresentado.

WADL – Web Services

+WADL - Web Application Description Language

É uma evolucao do WSDL, ele é um padrao de web services que nao implementa SOAP como protocolo, mais implementa HTTP como protocolo padrao, eh considerado a primeira versao do REST, pois ele trabalha com todos os verbos HTTP nos requests, diferente do WSDL que usa o protocolo SOAP;

WADL – Web Services

```
−<application>
   <doc jersey:generatedBy="Jersey: 1.11.1 03/31/2012 06:49 PM"/>
 −<grammars>
  −<include href="application.wadl/xsd0.xsd">
     <doc title="Generated" xml:lang="en"/>
   </include>
 </grammars>
 −<resources base="http://localhost:8080/MySQLDemoService/webresources/">
  −<resource path="com.mysql.entities.language">
   −<method id="findAll" name="GET">
    −<response>
       <ns2:representation element="language" mediaType="application/xml"/>
       <ns2:representation element="language" mediaType="application/json"/>
     </response>
    </method>
   −<method id="create" name="POST">
    −<request>
       <ns2:representation element="language" mediaType="application/xml"/>
       <ns2:representation element="language" mediaType="application/json"/>
     </request>
    </method>
   −<method id="edit" name="PUT">
    −<request>
       <ns2:representation element="language" mediaType="application/xml"/>
       <ns2:representation element="language" mediaType="application/json"/>
     </request>
    </method>
   −<resource path="{id}">
     <param name="id" style="template" type="xs:short"/>
     <method id="remove" name="DELETE"/>
    −<method id="find" name="GET">
```


WADL – Web Services

Vantagens:

-os requests realizados com o WADL, pode ser realizados usando os verbos HTTP;

-funciona com a maioria das linguagens de programacao atuais;

WADL – Web Services

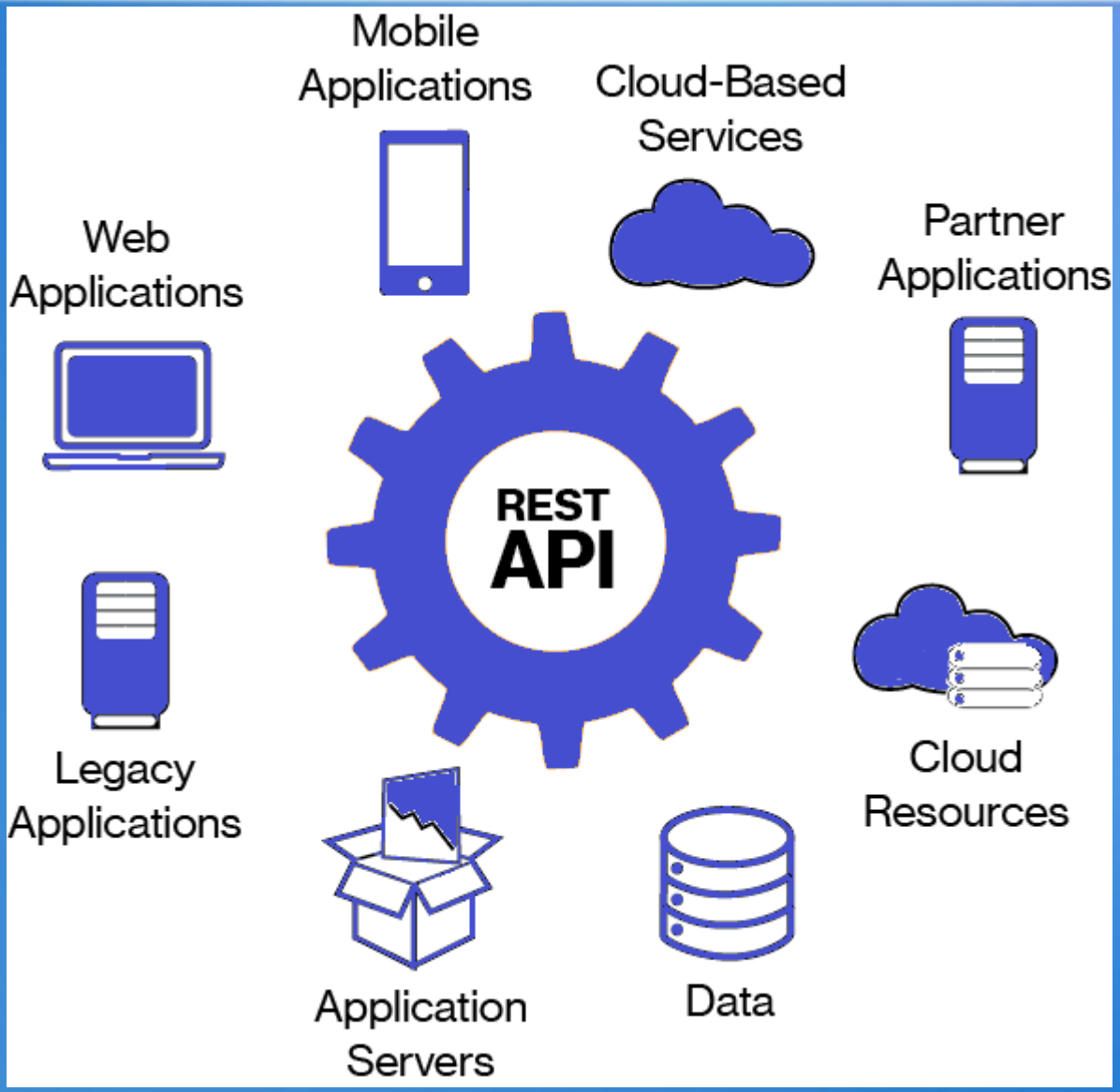
Desvantagem:

-Ainda precisa de usar xml para realizar os requests, o que torna muito verboso;

-Nao suporta toda a funcionalidade do REST;

-Nao implementa RESTFul;

REST - Web Services



REST - Web Services

REST é uma evolucao do WADL, um padrao de web service que veio para facilitar o trabalho do desenvolvedor, para que nao precise se preocupar com padroes complicados de xml, onde onse estiver um web service com um

HTTP ou HTTPS ja é suficiente para implementa-lo, alem de poder ser reaproveitado toda a infra estrutura ja existente e uma curva de aprendizado muito baixa;

REST - Web Services

REST é estilo de arquitetura SOA onde se usa o HTTP como protocolo tais quais como seus verbos em suas requisicoes;

Para que este estilo de arquitetura funcione, precisa que exista uma API que implemente as especificações REST, assim ela pode ser chamada de API RESTFul;

REST - Web Services

Uma API RESTFul, deve ter minimamente:

uma URL que represente o protocolo, dominio, versao e o recurso que ele deseja acessar, exemplo de sintaxe:

[GET] <protocolo>://<dominio>/<versao>/<recurso>/

[GET] http://minha-loja-de-doces.com.br/v1/bolacha/

REST - Web Services

Como ele precisa implementar os metodos HTTP, entao para cada request, ele precisa ser enviado usando os verbos:

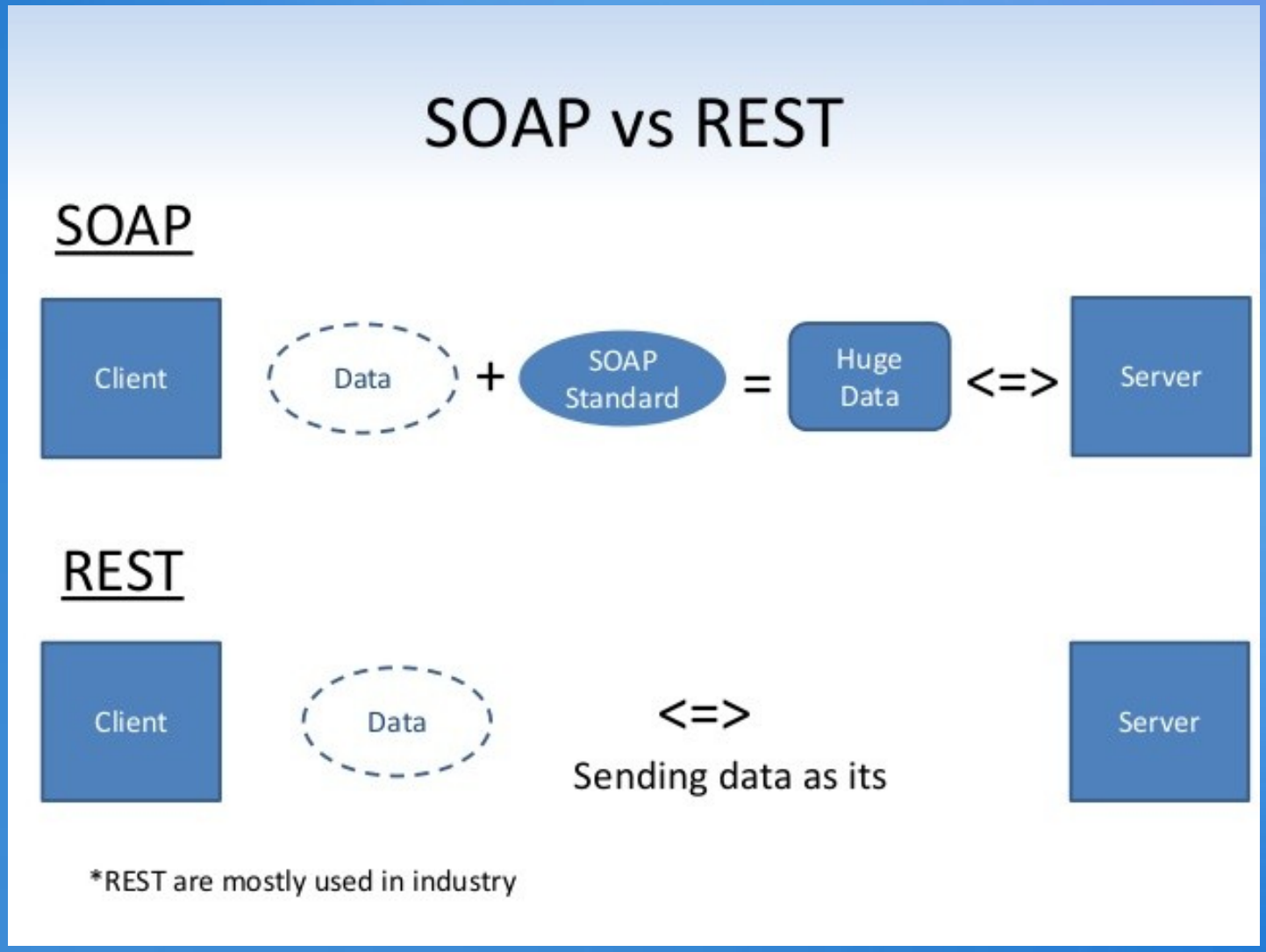
[PUT] http://minha-loja-de-doces.com.br/v1/bolacha/

[POST] http://minha-loja-de-doces.com.br/v1/bolacha/

[DELETE] http://minha-loja-de-doces.com.br/v1/bolacha/

[HEAD] http://minha-loja-de-doces.com.br/v1/bolacha/

REST - Web Services



REST - Web Services

Vantagens em cima do SOAP

-Simplicidade de uma interface uniforme;

-Enquanto que o REST faz uso de HTTP/HTTPS;

-Situações em que há limitação de recursos e de largura de banda: A estrutura de retorno é em qualquer formato definido pelo desenvolvedor e qualquer navegador pode ser usado. Isso porque a abordagem REST usa o padrão de chamadas GET, PUT, POST e DELETE. O REST também pode usar objetos XMLHttpRequest;

REST - Web Services

Vantagens em cima do SOAP

-Operações totalmente sem-estado: se uma operação precisa ser continuada, o REST não será a melhor opção. No entanto, se forem necessárias operações de CRUD stateless (Criar, Ler, Atualizar e Excluir), o REST seria a melhor alternativa.

-Situações que exigem cache: se a informação pode ser armazenada em cache, devido à natureza da operação stateless do REST, esse seria um cenário adequado para a tecnologia.

REST - Web Services

Vantagens em cima do SOAP

-Usa Json para transferencia de dados, eh mais simples, legivel e não eh verboso como os arquivos xml usados no WSDL;

REST - Web Services

Desvantagens do REST - versionamento

Somos uma loja de doces em Sao Paulo e vendemos no atacado para outras lojas de doces para outros estados no brasil, entao

precisamos expor um web service para que nossos clientes consultem os precos, a api esta pronta e vamos expor esta api para que nossos clientes consultem nossos precos:

REST - Web Services

Desvantagens do REST - versionamento

[GET]
http://minha-loja-de-doces.com.br/v1/bolacha/

```
{
    "bolacha" : 1.75,
    "qtde" : 50
}
```

REST - Web Services

Desvantagens do REST - versionamento

Vamos imaginar aqui que estamos tendo um grande problema nesta url, pois os clientes

dos estados de Minas Gerais, Rio de Janeiro, Rio Grande do Sul e Bahia tem problemas de encontrar o mesmo

produto mais no seu dialeto local:

REST - Web Services

Desvantagens do REST - versionamento

bolacha => São Paulo

biscoito => Rio de Janeiro

trem doce => Minas Gerais

cacetinho docinho tche => Rio Grande do Sul

redondin gostoso du painhu => Bahia

REST - Web Services

Desvantagens do REST - versionamento

http://minha-loja-de-doces.com.br/v1/bolacha/sao-paulo/

http://minha-loja-de-doces.com.br/v2/biscoito/rio-de-janeiro/

http://minha-loja-de-doces.com.br/v3/trem-doce/minas-gerais/

http://minha-loja-de-doces.com.br/v4/cacetinho-docinho-tche/rio-grande-do-sul/

http://minha-loja-de-doces.com.br/v5/redondin-gostoso-du-painhu/bahia/

REST - Web Services

Desvantagens do REST – Verbosidade

Se o problema da escalabilidade acima ainda nao te convenceu?

imagina o seguinte, para cada estado do brasil que eu vendo meus doces, imagina que eu tenho que expor

uma API REST para que estes estados informando alguns custos de envio como por exemplo:

REST - Web Services

Desvantagens do REST – Verbosidade

[GET]
http://minha-loja-de-doces.com.br/v1/bolacha/

```
{
    "bolacha" : 1.75,     "qtde" : 50,
    "ICMS" : 12,   "COFINS" : 7,
    "ISS" : 3,  "IPI" : 8,
    "IR" : 5,  "IOF" : 18,
    "FRETE" : 25
}
```

REST - Web Services

Desvantagens do REST – Verbosidade

Minas Gerais => ICMS e o Frete;

Acre => produto + Cofins + Frete;

São Paulo => ISS, porque eu nao cobro frete para o mesmo estado;

Caso eu tenha milhares de requests, tenho muitas informacoes que não estao sendo consumidas pelos meus clientes;

REST - Web Services

Desvantagens do REST – Verbosidade

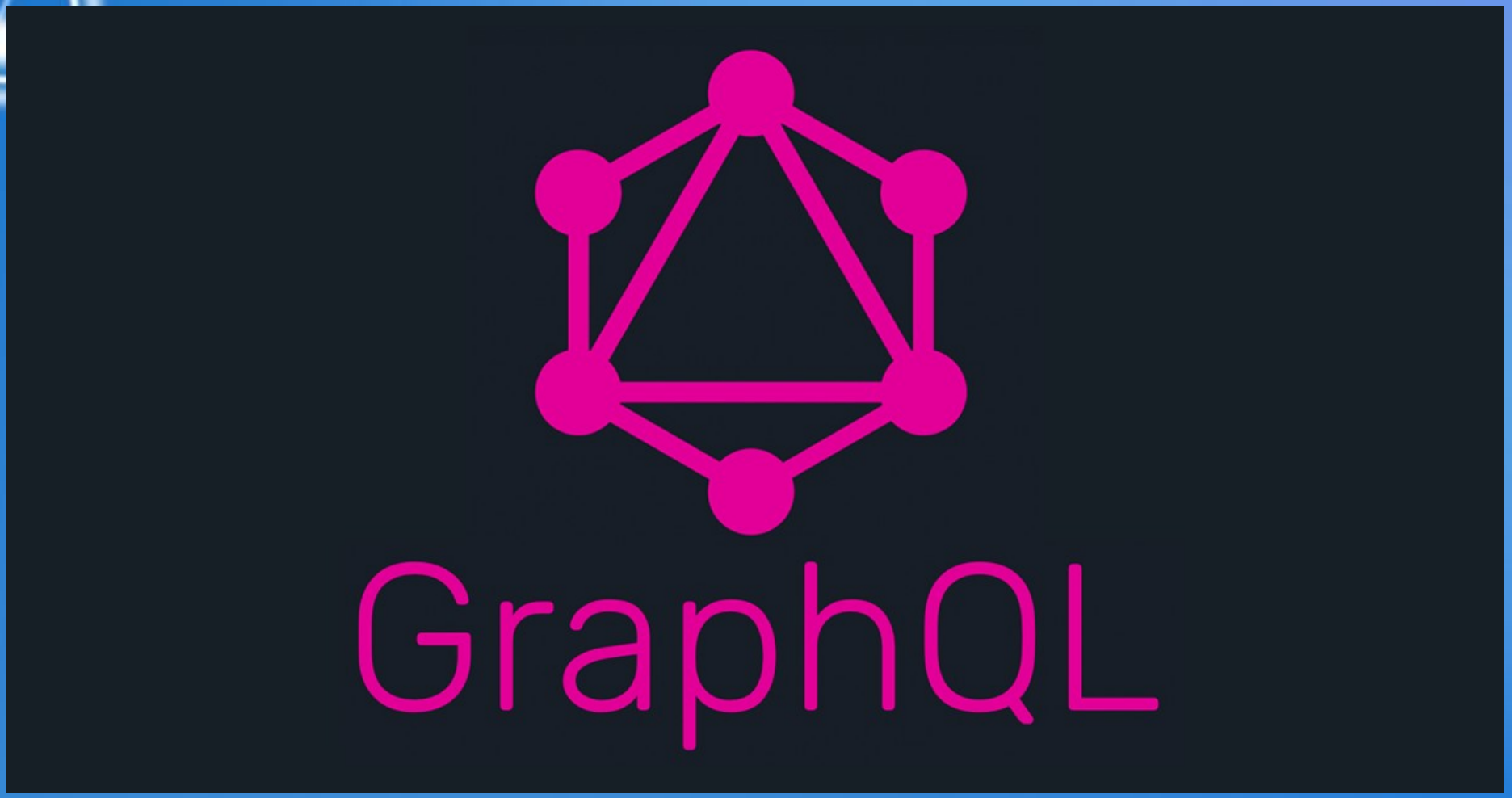
Como resolvemos este problema?

deixa a API assim mesmo? mostrando informações extras que ninguem vai aproveitar 100% ?

Coloco "ifs" dentro do meu codigo para validar isto?

faço igual ao exemplo acima? crio varias versoes da mesma API?

GraphQL - Web Services



GraphQL - Web Services

Assim como o REST, é um SOA, um estilo de arquitetura de WebService, Criado pelo Facebook para resolver alguns problemas que os engenheiros de software da empresa encontraram no REST, para deixar a performance das requests e mais performaticas;

Ele comecou a ser desenvolvido e usado internamente no ano de 2012 e foi publicado no ano de 2015;

GraphQL - Web Services

O GraphQL diferente do REST, nao usa o protocolo nem os verbos HTTP, ele necesita apenas de uma URL para ser disponivel para o cliente,

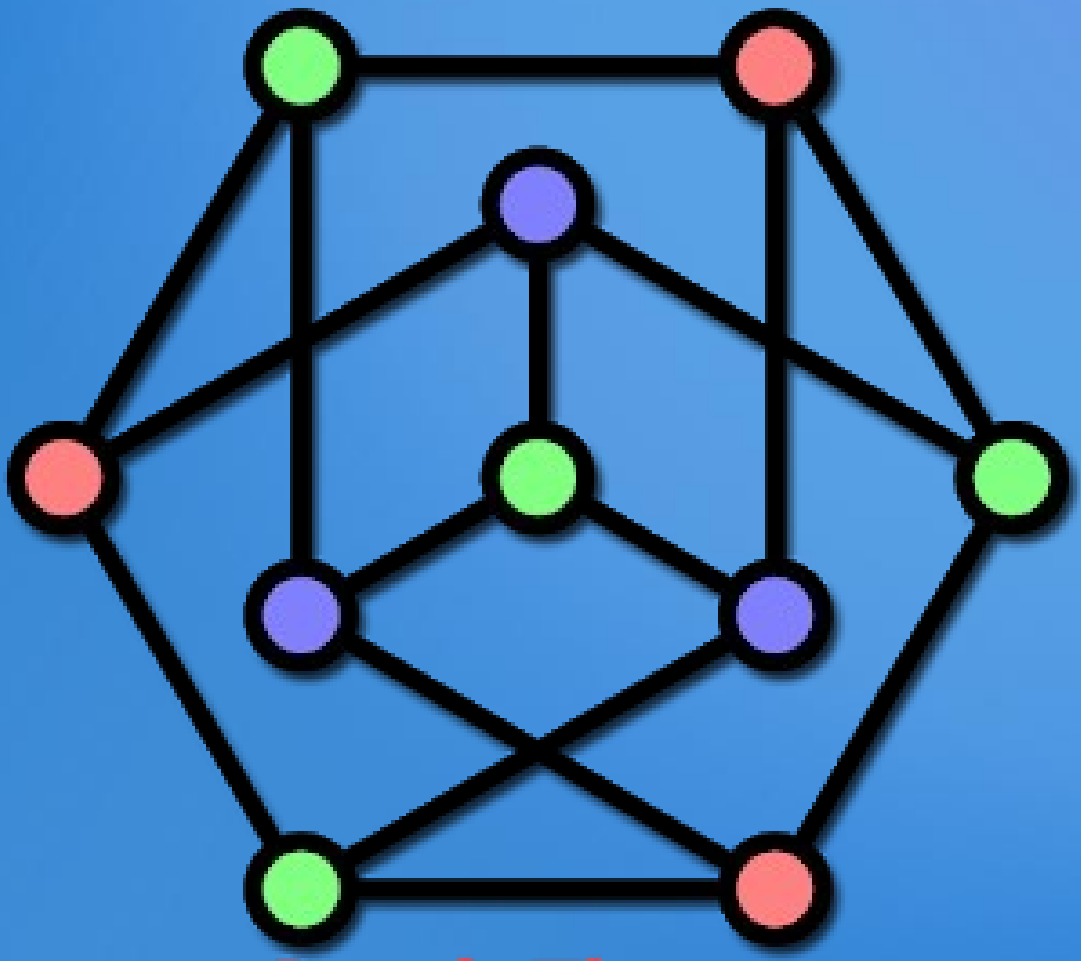
os dados para que ele os acesse, basta que ele realiza uma busca via query language, e ele obtera apenas as informações que ele realmente precisa;

GraphQL - Web Services

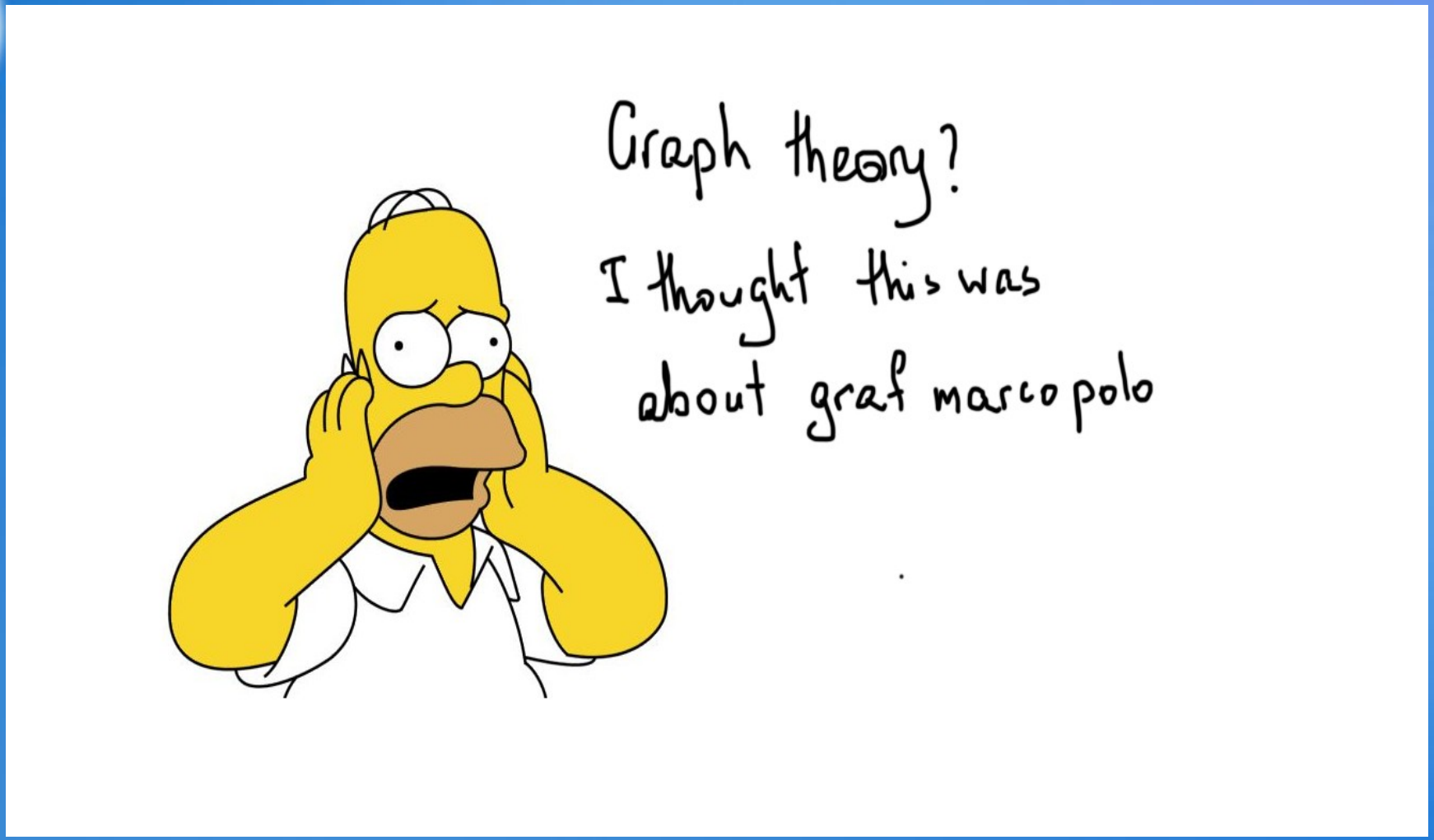
O GraphQL diferente do REST, nao usa o protocolo nem os verbos HTTP, ele necesita apenas de uma URL para ser disponivel para o cliente,

os dados para que ele os acesse, basta que ele realiza uma busca via query language, e ele obtera apenas as informações que ele realmente precisa;

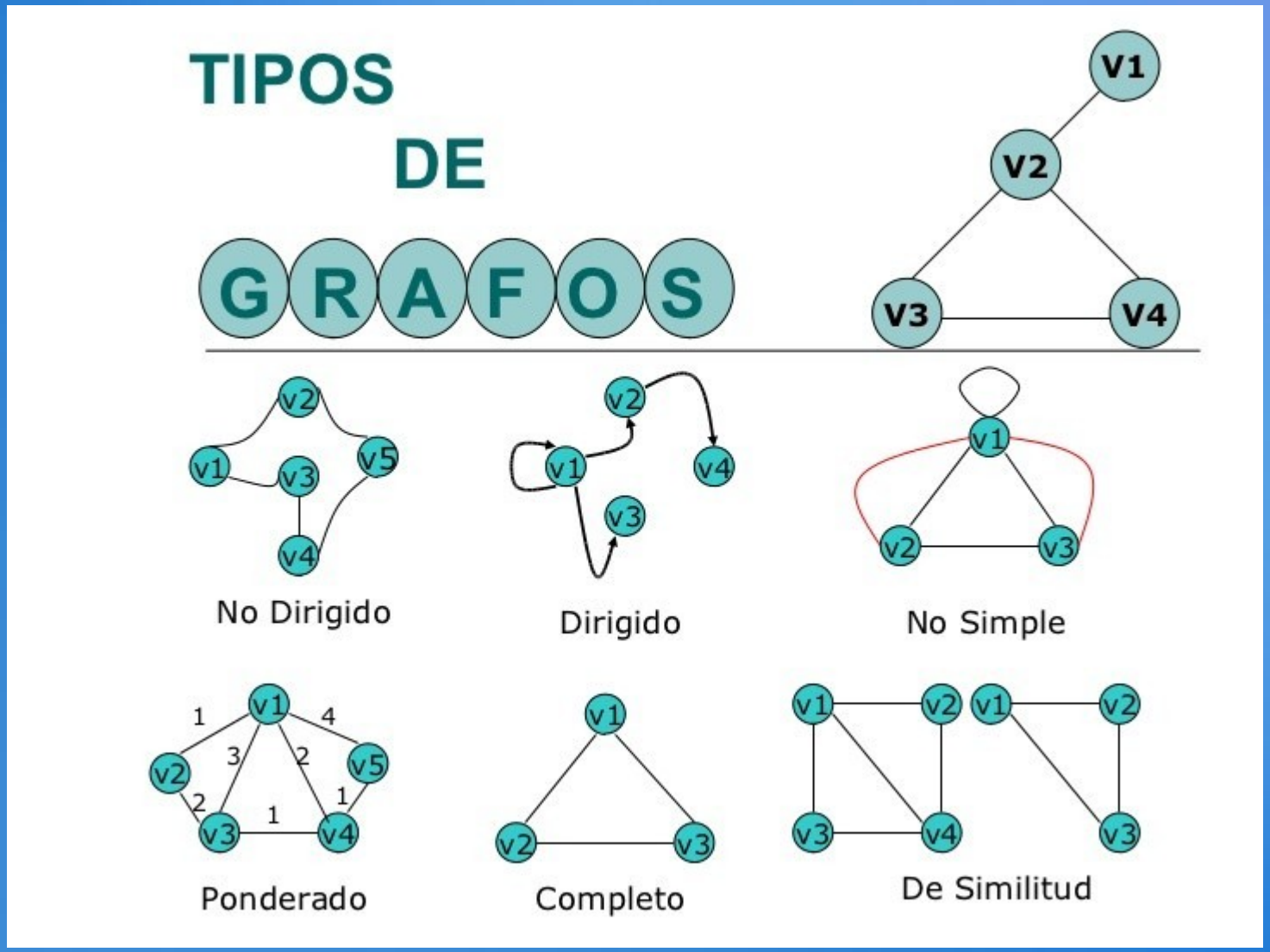
GraphQL - Web Services



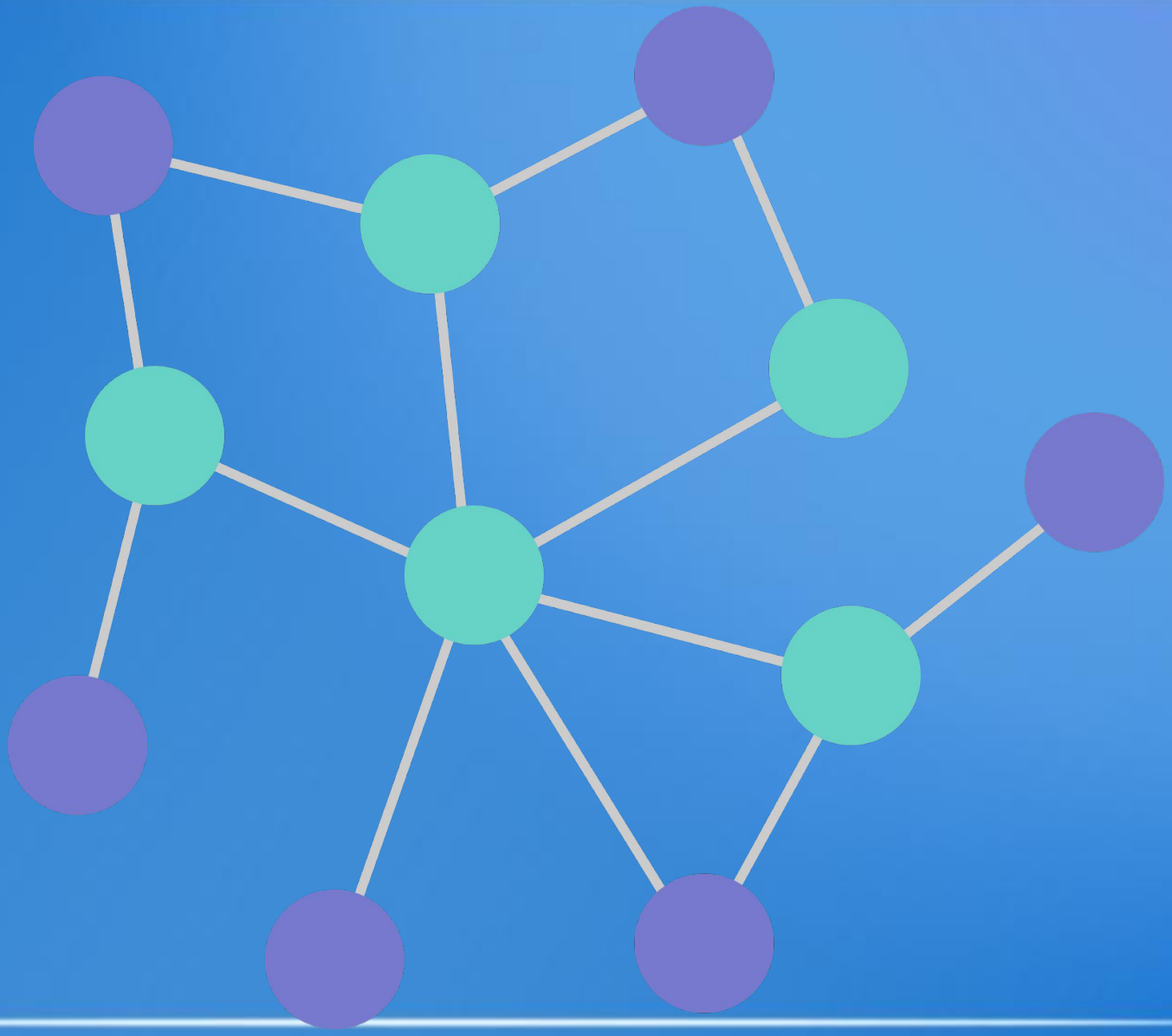
GraphQL - Web Services



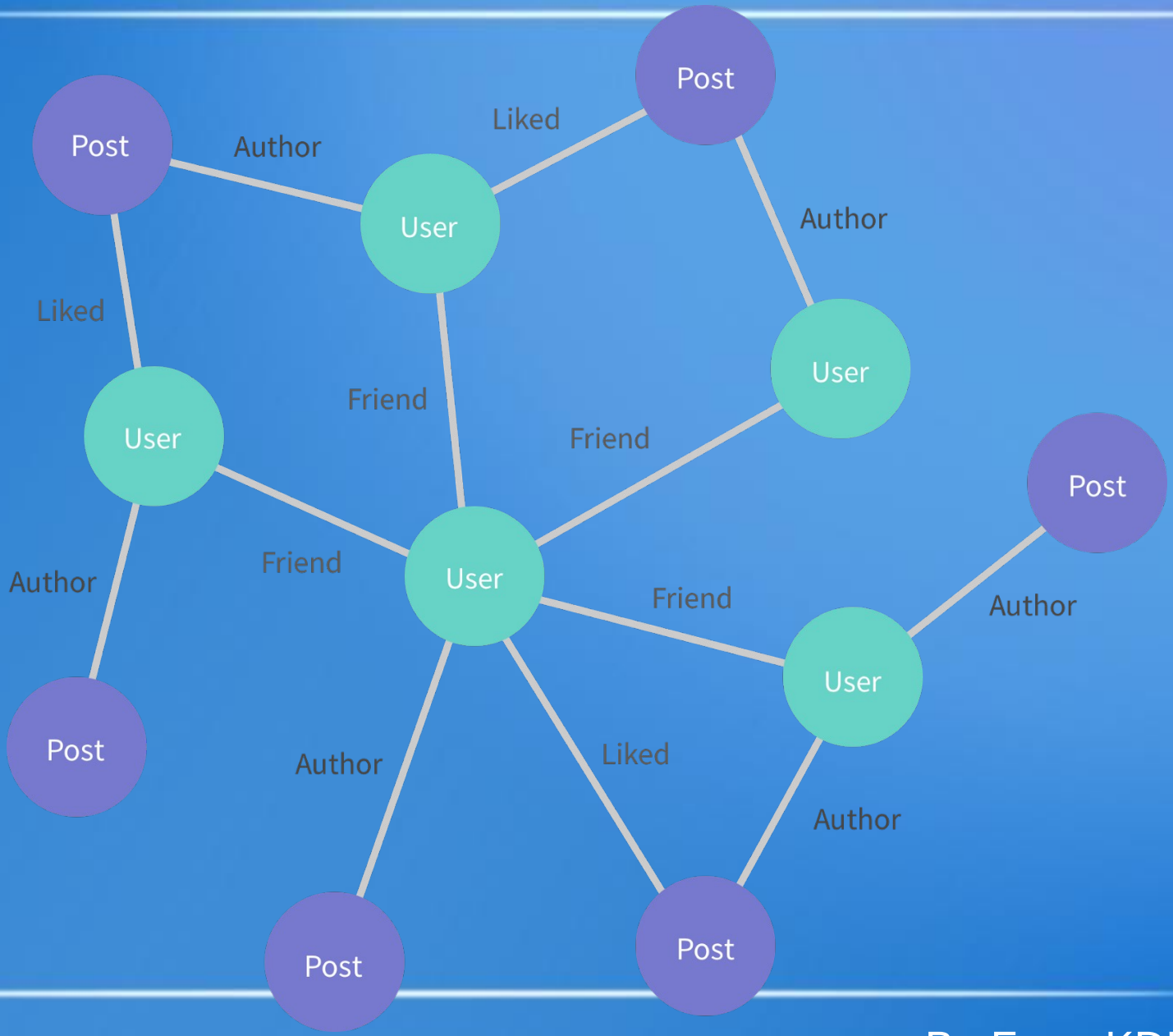
GraphQL - Web Services



GraphQL - Web Services



GraphQL - Web Services



Be Free. KDE

GraphQL - Web Services

O GraphQL nos traz novos conceitos e novos termos, vamos conhecer alguns deles:

Querys => na QL eh usado para trazer dados de um servidor GraphQL, como se fosse os requests GET do REST;

Mutations => Quando eh necessario realizar mudanca de estado em um servico GraphQL, dizemos que precisamos fazer uma Mutacao deste estado; no mundo REST eh parecido com o PUT/DELETE/POST;

GraphQL - Web Services

Edges => eh o elo de ligacao entre os nodes;

Cursor => eh um pagination model, a partir dele eh usado o contexto de fetch data;

Node => eh o noh em que existe a informacao ;

GraphQL - Web Services

Quem usa GraphQL ???

GraphQL - Web Services

GraphQL - Web Services

GraphQL - Web Services

GraphQL - Web Services

GraphQL - Web Services

GraphQL - Web Services

GraphQL - Web Services

Linguagens de Programação suportadas

C# / .NET
Python
Ruby
Elixir
Go
Java
JavaScript
PHP
Scala
Clojure
Erlang
Groovy

GraphQL - Web Services

Tipos suportados pelo GraphQL

Enums

Scalars

Lists and Non-Null

ObjectTypes

Interfaces

Unions

Schema

Mutations

AbstractTypes

GraphQL - Web Services

GraphQL - Web Services

https://developer.github.com/v4/explorer/

#teste de exemplo-1:

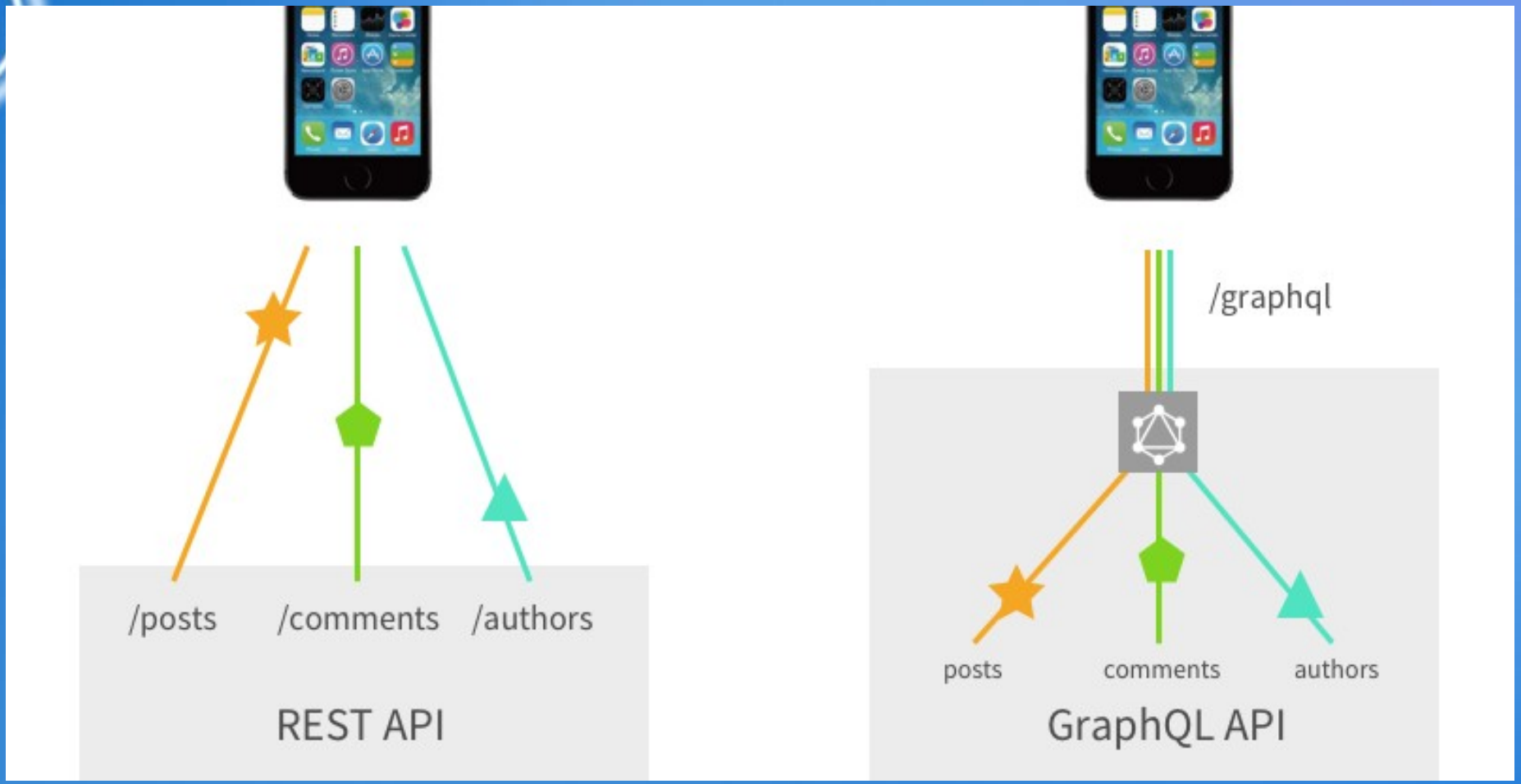
```
query {
  repository(owner: "graphql", name: "graphql-js")
{
    name
    description
  }
}
```

GraphQL - Web Services

GraphQL – Mutations

```
mutation AddReactionToIssue {
  addReaction(input:
{subjectId:"MDU6SXNzdWUyMzEzOTE1NTE=",
content:HOORAY}) {
    reaction {
      content
    }
    subject {
      id
    }
  } }
```


GraphQL - Web Services



GraphQL - Web Services

+ Pontos positivos do GraphQL

Melhor aproveitado quando existe um grande volume de dados em uma API com um numero consideravel de requests;

Nao Precisa versionar os endpoints

Nao preciso guardar uma collections de endpoints

O cliente da API seleciona o que ele quer obter, não tenho que responder um json cheio de informacoes irrelevantes;

GraphQL - Web Services

+ Pontos negativos do GraphQL

Melhor aproveitado quando existe um grande volume de dados em uma API com um numero consideravel de requests, ou seja, inviavel em projetos pequenos;

Quando vc precisa de um Web Service Stateless;

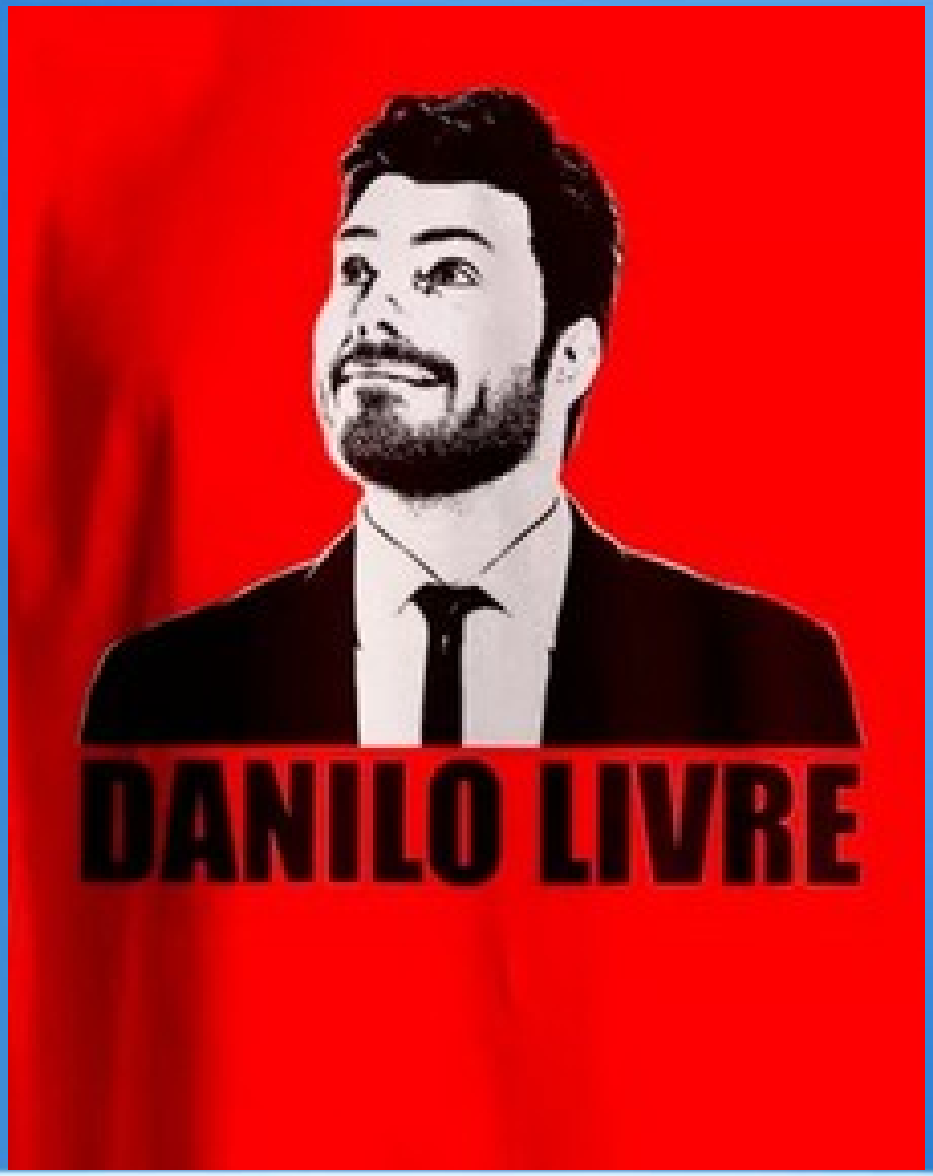
Curva de aprendizado de media para alta;

Qual Melhor ? Web Services

Qual o melhor meio de transferir dados?

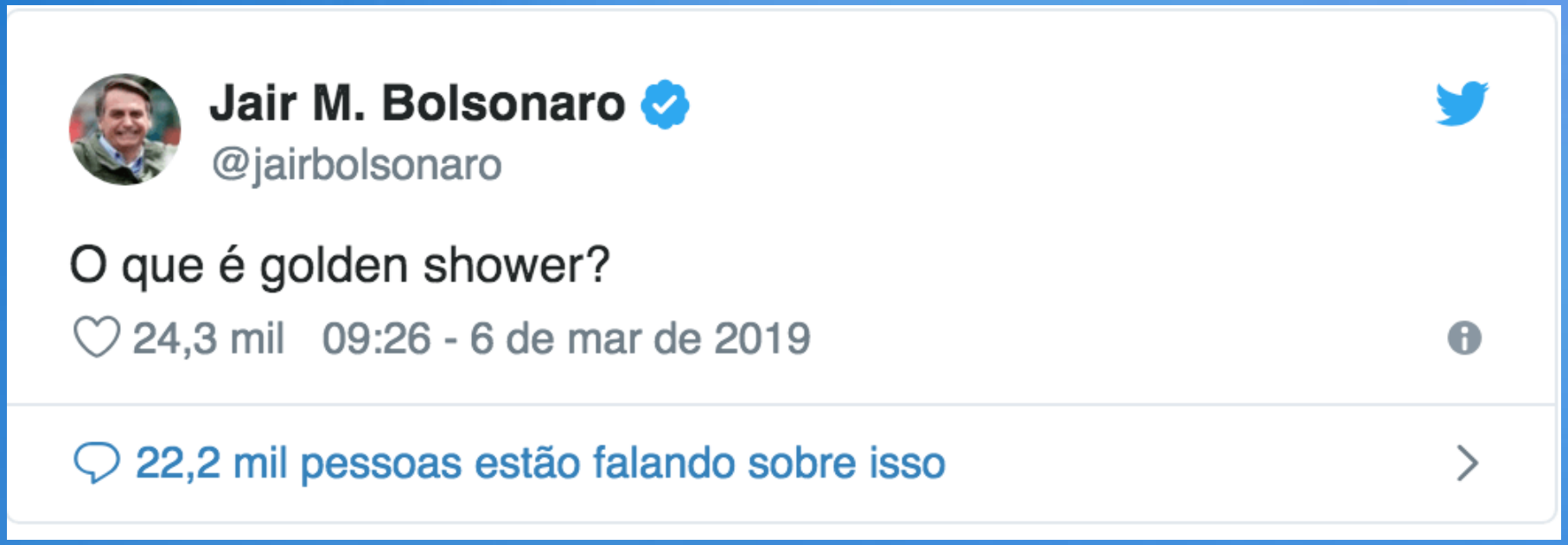
FTP?
SOA?
SOAP?
WSDL?
WADL?
REST?
GraphQL?

Danilo Livre!!!



Be Free. KDE

Alguma Pergunta?



Alguma Pergunta?

About me

@marcosptf

pytero
phpzero
javero
open source evangelist