# REST API's (12h)

Márcio Pereira – marcio.pereira00@gmail.com

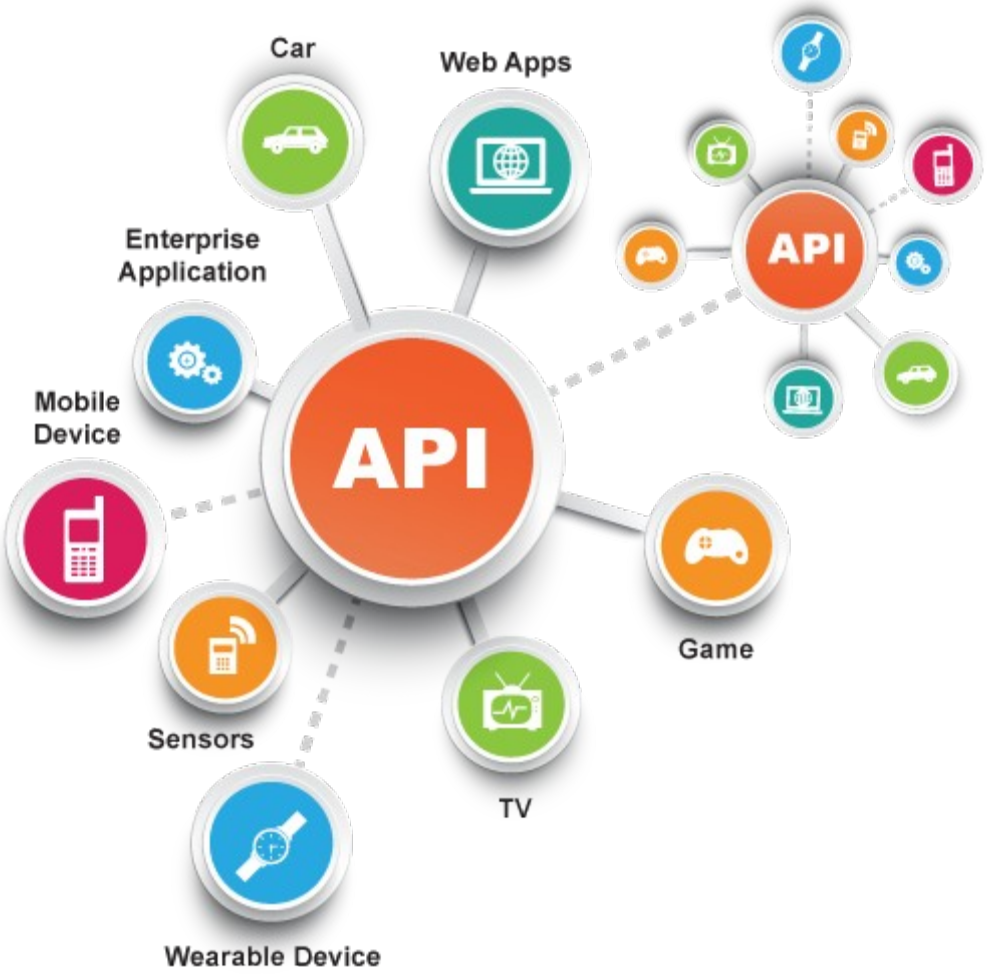
FLAG

- O que é uma API REST
- Protocolo HTTP
- Tipos de pedidos e códigos de resposta
- Respostas JSON
- Respostas XML
- Autenticação com base em tokens
- Criação da documentação para a API

## O que é uma API (*Application Programming Interface*) ?

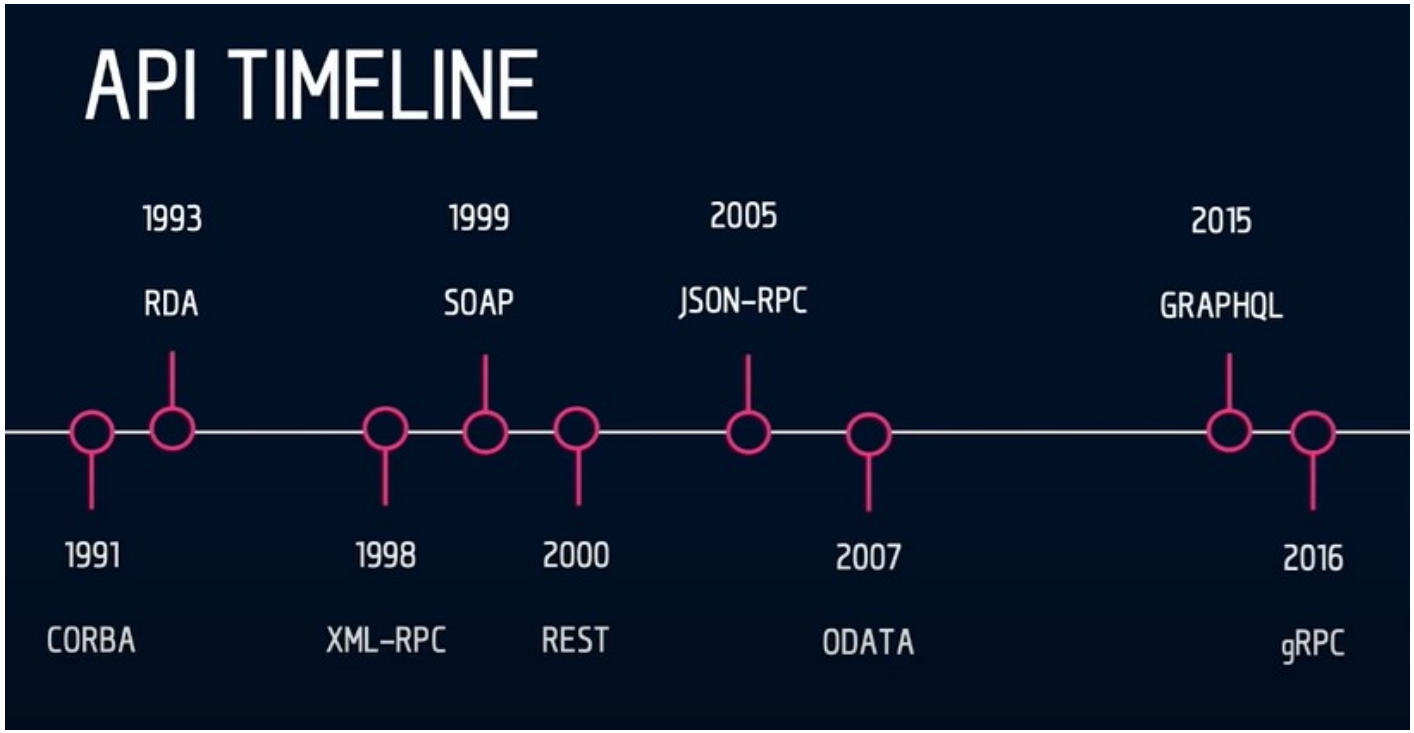
Um conjunto de funções e procedimentos que permitem que aplicações acedam a recursos ou dados de uma outra aplicação.

**É uma forma das aplicações comunicarem entre si...**

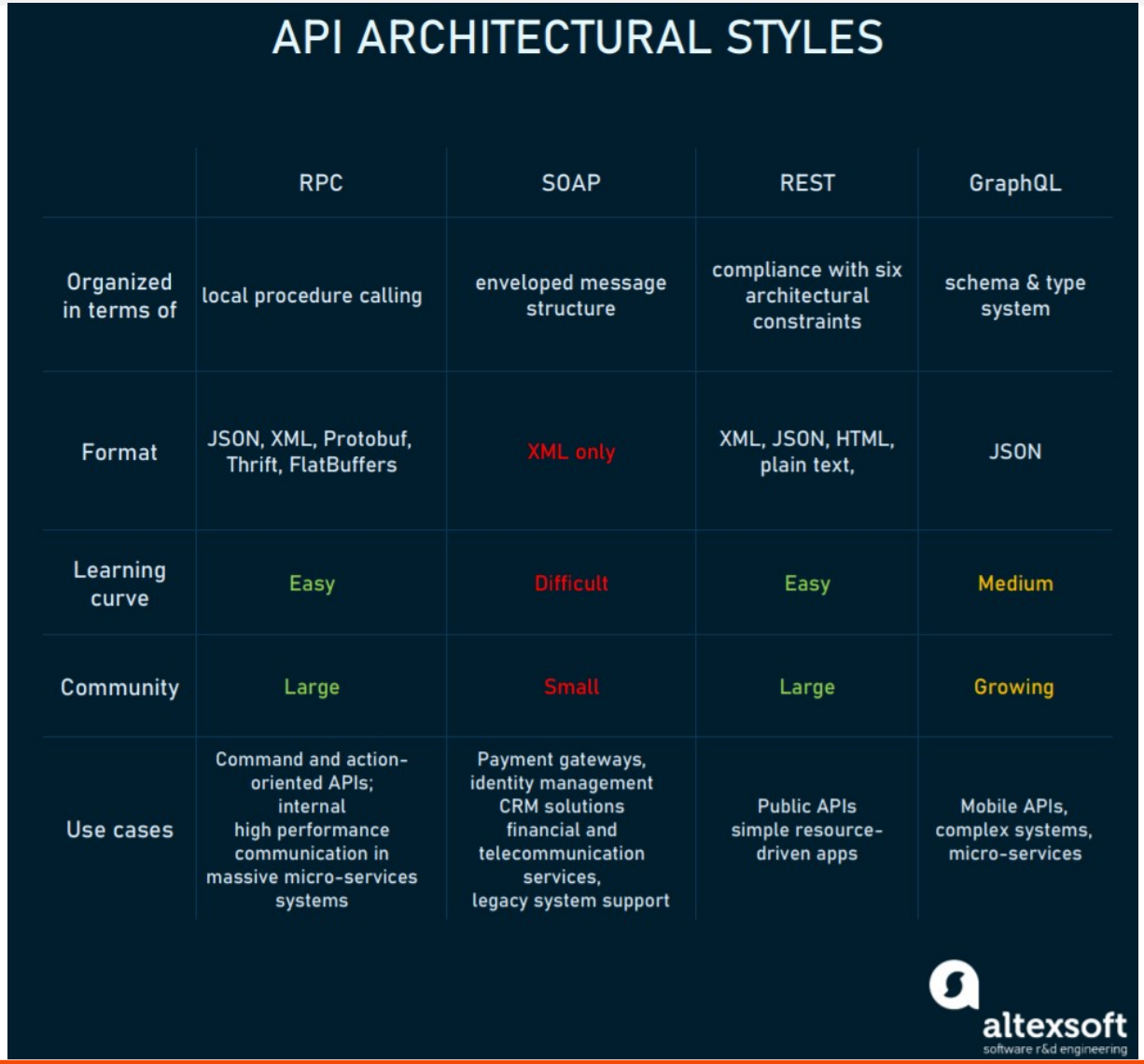


## O que é uma WEB API?

Uma web API server-side é uma interface programática consistente de **um ou mais endpoints** publicamente expostos para um sistema definido de mensagens **pedido-resposta**, tipicamente expressado em JSON ou XML, que é exposto via a internet - mais comumente por meio de um servidor web baseado em **HTTP**.



# REST API


API ARCHITECTURAL STYLES
RPC
SOAP
REST
GraphQL
Organized in terms of
local procedure calling
enveloped message structure
compliance with six architectural constraints
schema & type system
Format
JSON, XML, Protobuf, Thrift, FlatBuffers
XML only
XML, JSON, HTML, plain text,
JSON
Learning curve
Easy
Difficult
Easy
Medium
Community
Large
Small
Large
Growing
Use cases
Command and action-oriented APIs; internal high performance communication in massive micro-services systems
Payment gateways, identity management CRM solutions financial and telecommunication services, legacy system support
Public APIs simple resource-driven apps
Mobile APIs, complex systems, micro-services
altexsoft
software r&d engineering

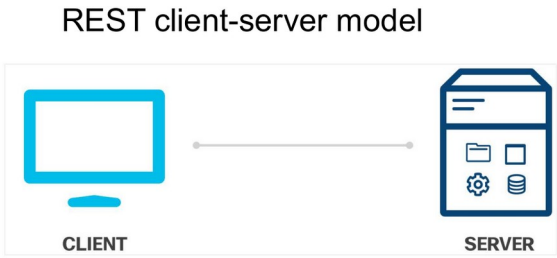
## *REST* - REPRESENTATIONAL STATE TRANSFER

REST é uma arquitetura de software da autoria de Roy Thomas Fielding projetado em meados do ano 2000 para sistemas distribuídos e, particularmente, a World Wide Web, que funciona "*em cima*" do protocolo HTTP.
A arquitetura REST é simplesmente o seguir de algumas diretrizes sobre como uma aplicação Web bem projetada se comporta, usando uma organização lógica que envolve uma série de links (endpoints) e transições de estados

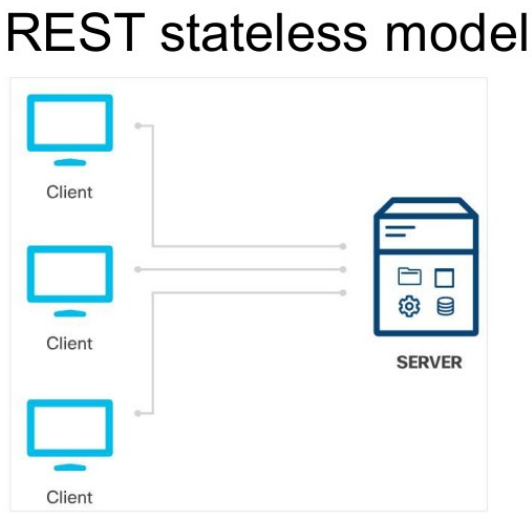
- Rest não é um Protocolo
- Rest não é um padrão de desenvolvimento de software.

## Princípios Orientadores do REST

- **Cliente-servidor**
  - O cliente e o servidor devem ser independentes entre si.
  - Esta independência contribuí para uma melhor separação de responsabilidades, e melhora a escalabilidade da aplicação.

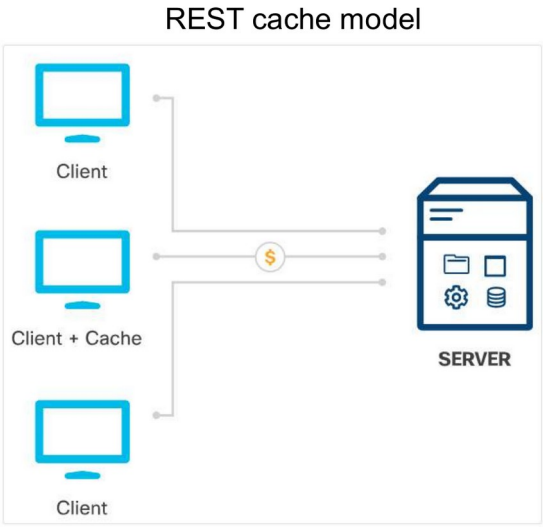



- **Sem estado (stateless)**
  - As solicitações do cliente para o servidor devem contém o modelo cliente-servidor REST e todos os informações que o servidor precisa fazer o pedido.
  - O servidor não pode conter estados de sessão.



## Princípios Orientadores do REST

- **Cache model**
  - As respostas do servidor devem indicar se a informação da resposta pode ser armazenada em cache ou não.
  - Se for armazenável em cache, o cliente pode usar os dados da resposta para solicitações posteriores.

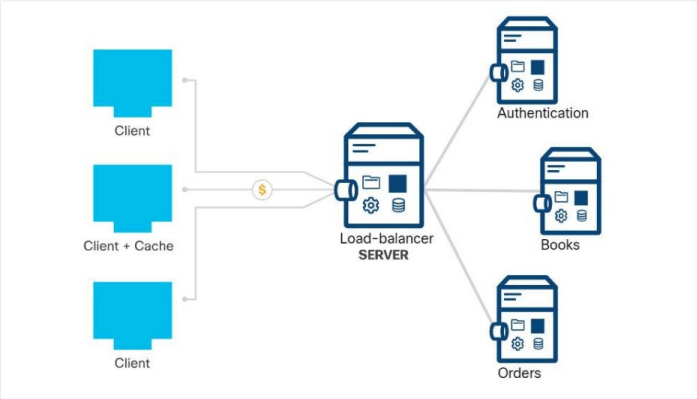



- **Interface uniforme**
  - O interface entre o cliente e o servidor têm 4 restrições de interface:
    - identificação de recursos;
    - manipulação de recursos por meio de representações;
    - mensagens auto-descritivas;
    - Usa hipermídia como o motor do estado do aplicativo.

## Princípios Orientadores do REST

- **Sistema de camadas**
  - O estilo de sistema em camadas permite que uma arquitetura seja composta de camadas hierárquicas, restringindo o comportamento do componente de forma que cada componente não possa "ver" além da camada imediata com a qual está interagindo.

REST layered system model

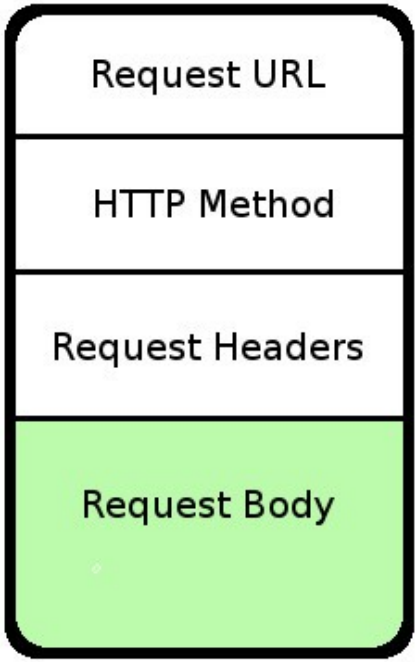



- **Código sob demanda (opcional)**
  - O rest premite que s informações retornadas por um serviço REST podem incluir código executável (por exemplo, javascript) ou links destinados a estender a funcionalidade do cliente de forma útil.
  - A restrição é opcional porque a execução de códigos de terceiros apresenta potencial riscos de segurança.

**Principais conceitos de uma rest api**

- Resources
- URI
- Verb
- Resource representation
- Response codes

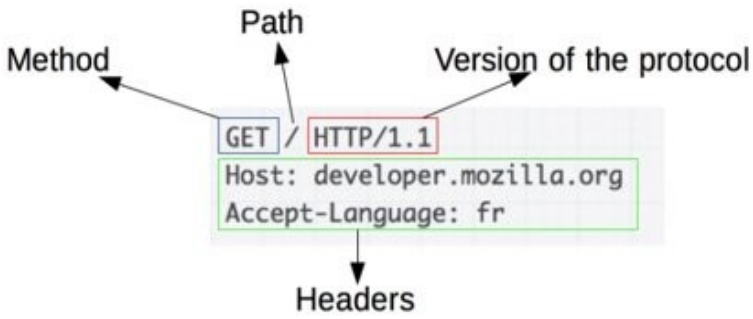
**Principais conceitos HTTP (REQUEST)**



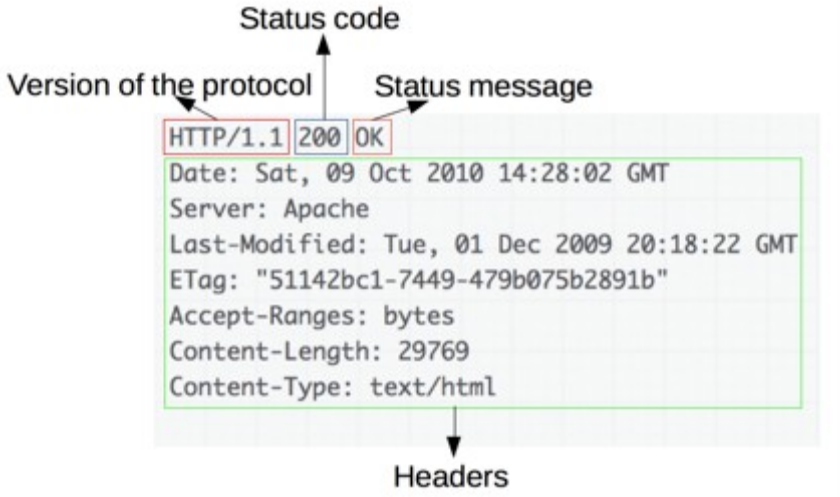
## HTTP Request / Response

URL


REQUEST




RESPONSE

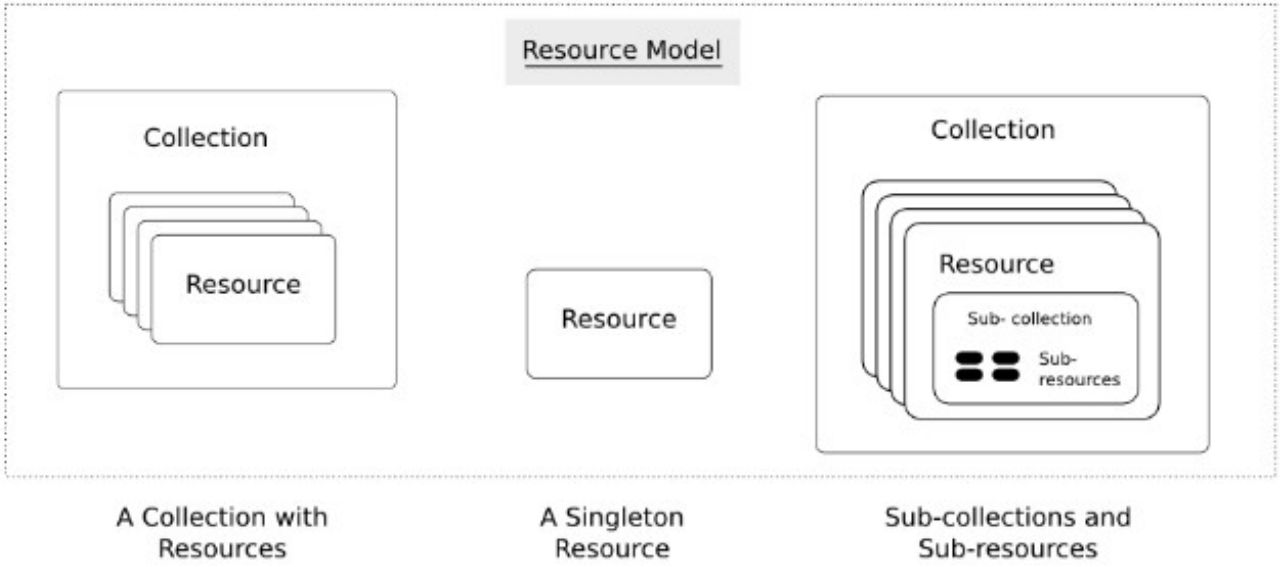



**+ Body na request e reponse**

## Resources

Tudo na arquitetura RESTful é sobre recursos.

- Um recurso é um objeto com seus próprios dados associados.
- Recursos têm relacionamentos com outros recursos.
- Um recurso pode conter outros recursos.
- É semelhante a uma instância de objeto em uma linguagem de programação orientada a objeto.



## Resources

- Os dados devem de ser "explicados" na sua documentação
- Os dados de um recurso devem refelectir a sua estrutura, tendo por isso um tipo associado (ex: string, int, array)
- Esses dados são chamados de atributos

| Attribute | Type | Meaning |
|---|---|---|
| id | String | Identifies the unique ID of a resource. |
| href | String | Identifies the URL of the current resource. |
| link | Object | Identifies a relationship for a resource. This attribute is itself an object and has “rel” “href” attributes. |

## Resources

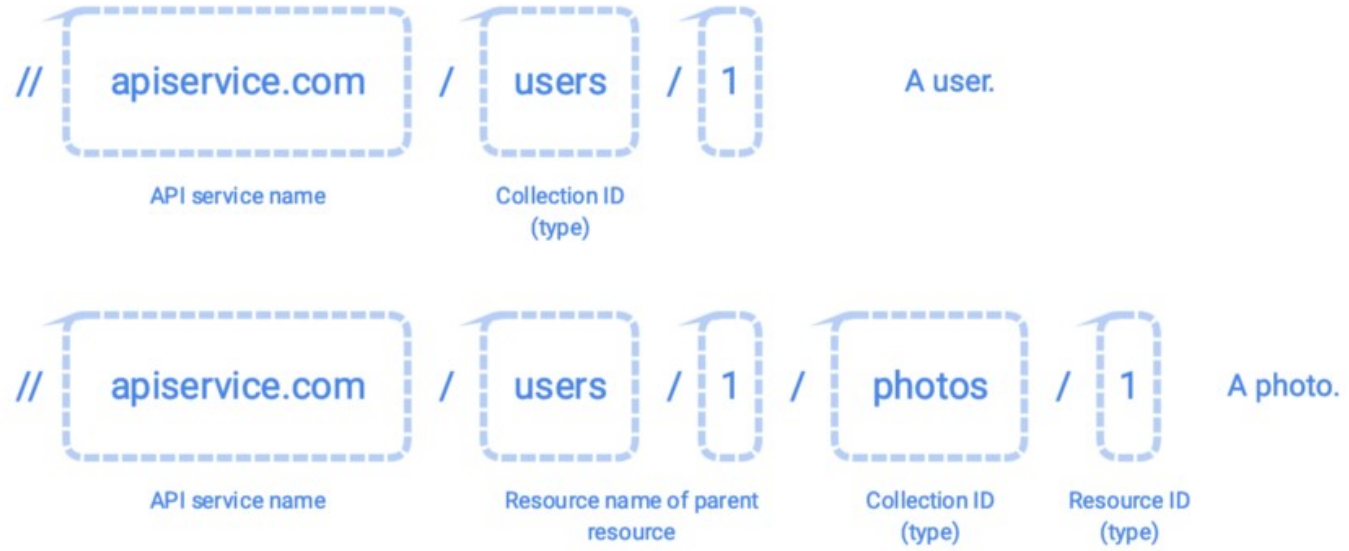
- Os recursos podem ser representados em diferentes formatos de dados como por exemplo JSON e XML-
- Esta representação é enviada no HTTP body

```json
{
  "_type": "vm",
  "name": "A virtual machine",
  "memory": 1024,
  "cpu": {
    "cores": 4,
    "speed": 3600
  },
  "boot": {
    "devices": ["cdrom", "harddisk"]
  }
}
```

```xml
<vm xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <name type="xs:string">My VM</name>
  <memory type="xs:int">1024</memory>
  <cpu>
    <cores type="xs:int">4</cores>
    <speed type="xs:int">3600</speed>
  </cpu>
  <boot>
    <devices type="xs:list">
      <device type="xs:string">cdrom</device>
      <device type="xs:string">harddisk</device>
    </devices>
  </boot>
</vm>
```

## Resources

- Os recursos são identificados pelo seu **URI** - Uniform Resource Identifier



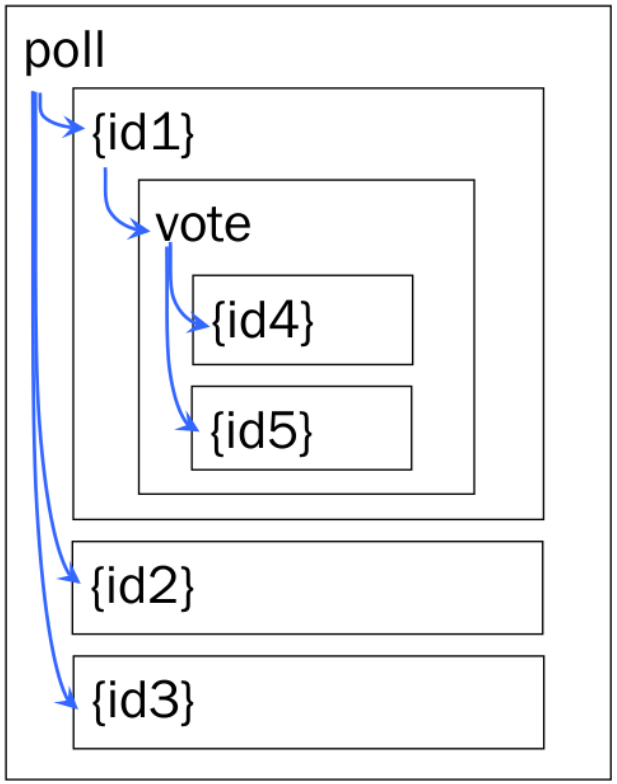
## Resources

- As interações feitas com um recurso são definidas pelos métodos HTTP

| HTTP Method | Action | Description |
|---|---|---|
| POST | Create | Create a new object or resource. |
| GET | Read | Retrieve resource details from the system. |
| PUT | Update | Replace or update an existing resource. |
| PATCH | Partial Update | Update some details from an existing resource. |
| DELETE | Delete | Remove a resource from the system. |

## Resources - Resume

1. Resources:
   **polls and votes**
2. Containment Relationship:




| | GET | PUT | POST | DELETE |
|---|---|---|---|---|
| /poll | ✓ | ✗ | ✓ | ✗ |
| /poll/*{id}* | ✓ | ✓ | ✗ | ✓ |
| /poll/*{id}*/vote | ✓ | ✗ | ✓ | ✗ |
| /poll/*{id}*/vote/*{id}* | ✓ | ✓ | ✗ | ? |

3. URIs embed IDs of “child” instance resources
4. POST on the container is used to create child resources
5. PUT/DELETE for updating and removing child resources

## HTTP STATUS CODE

A arquitetura REST faz uso do http status code para passar informação adicional ao cliente na resposta .

Existem cinco "categorias" diferentes de HTTP status code:

- **1xx - Informativo -** para fins informativos, as respostas não contêm um corpo;
- **2xx - Sucesso -** o servidor recebeu e aceitou o pedido;
- **3xx - Redireccionamento -** o cliente tem uma ação adicional a realizar para que a solicitação seja concluída;
- **4xx - Erro do cliente –** o pedido contém erros, como sintaxe incorreta ou entrada inválida
- **5xx - Erro do servidor -** não foi possível atender às solicitações válidas.

## HTTP STATUS CODE

100 Continue
200 OK
201 Created
202 Accepted
203 Non-Authoritative
204 No Content
205 Reset Content
206 Partial Content
300 Multiple Choices
301 Moved Permanently
302 Found
303 See Other
304 Not Modified
305 Use Proxy
307 Temporary Redirect

400 Bad Request
401 Unauthorized
402 Payment Required
403 Forbidden
404 Not Found
405 Method Not Allowed
406 Not Acceptable
407 Proxy Authentication Required
408 Request Timeout
409 Conflict
410 Gone
411 Length Required
412 Precondition Failed
413 Request Entity Too Large
414 Request-URI Too Long
415 Unsupported Media Type
416 Requested Range Not Satisfiable
417 Expectation Failed

4xx Client's fault

500 Internal Server Error
501 Not Implemented
502 Bad Gateway
503 Service Unavailable
504 Gateway Timeout
505 HTTP Version Not Supported

5xx Server's fault

## HTTP HEADER

A arquitetura REST faz uso do http Headers para passar informação adicional ao servidor, como por exemplo as credencias de acesso como o token ou Basic Auth ou informação do formato do corpo da mensagem enviada.

Os cabeçalhos HTTP são formatados como pares nome-valor separados por dois pontos (:), nome:valor.

- **Request headers -** incluem informações adicionais que não estão relacionadas ao conteúdo da mensagem.

| Key | Example Value | Description |
|---|---|---|
| Authorization | Basic dmFncmFudDp2YWdyYW | Provide credentials to authorize the request |

- **Entity headers -** incluem informações adicionais que descrevem o conteúdo do corpo da mensagem.

| Key | Example Value | Description |
|---|---|---|
| Content-Type | application/ json | Specify the format of the data in the body |

# Autenticação REST

## REST - Autenticação vs. Autorização

Tal como nas aplicações web, as APIs REST requerem autenticação para que os utilizadores não possam aceder, criar, atualizar ou apagar informações incorretam ente ou maliciosamente.
Algumas APIs que não requerem autenticação são somente leitura e não contêm informações críticas ou confidenciais.

**Autenticação**:

- A autenticação prova a identidade do utilizador.
- Normalmente nas applicações web é definida por username e password.

**Autorização**:

- Autorização define o acesso do utilizador.
- É o ato em que o utilizador (depois autenticado) prova ter permissões para executar a ação solicitada naquele recurso.

## REST – Mecanismos de autenticação

Os tipos comuns de mecanismos de autenticação incluem:

**Basic authentication:** Transmite credenciais como pares de nome de username / password separados por dois pontos (:) e codificado usando Base64.

**Token authentication:** usa um token do portador, que é uma string gerada por um servidor de autenticação, como um serviço de identidade (IdS). Os mais utilizados são o Bearer token e o JWT.

**API Key:** É uma string alfanumérica exclusiva gerada pelo servidor e atribuída a um do utilizador.

**OAuth:** O OAuth não é apenas um método de autenticação, e sim um protocolo completo com diversas especificações de segurança. Ele é extremamente útil para o processo de autenticação e autorização, e por isso, atualmente é o método mais recomendado para o cenário de APIs.

# Explore REST API

## Explore REST API's

Para perceber melhor como funcionam na prática as api's Rest, vamos explorar algumas.

**The-one-api (READ ONLY)**

- Doc: https://the-one-api.dev/documentation
- Access token: KGUGxeatxZWVd_x8eO6b

**football-data.org (READ ONLY)**

- Doc: https://www.football-data.org/documentation/quickstart
- Chave api: 14365804cf744b81846af954f0d29734

**OpenBank-project**

- Doc: https://apiexplorer.openbankproject.com/
- OAuth Doc: https://apiexplorer.openbankproject.com/glossary#Direct-Login
- Consumer ID: 30f87311-a793-498e-9128-280b205596c9
- Consumer Key: kvtwmqrtpef4ymwifgd5inxxhwhbajzo0mkfyzr1
- Consumer Secret: chfjjmcxxkpd13xikfhxjo1ekprukulgt0uxcimg

# Laravel - REST API