# REST

Representational State Transfer

## Baseado em Recursos

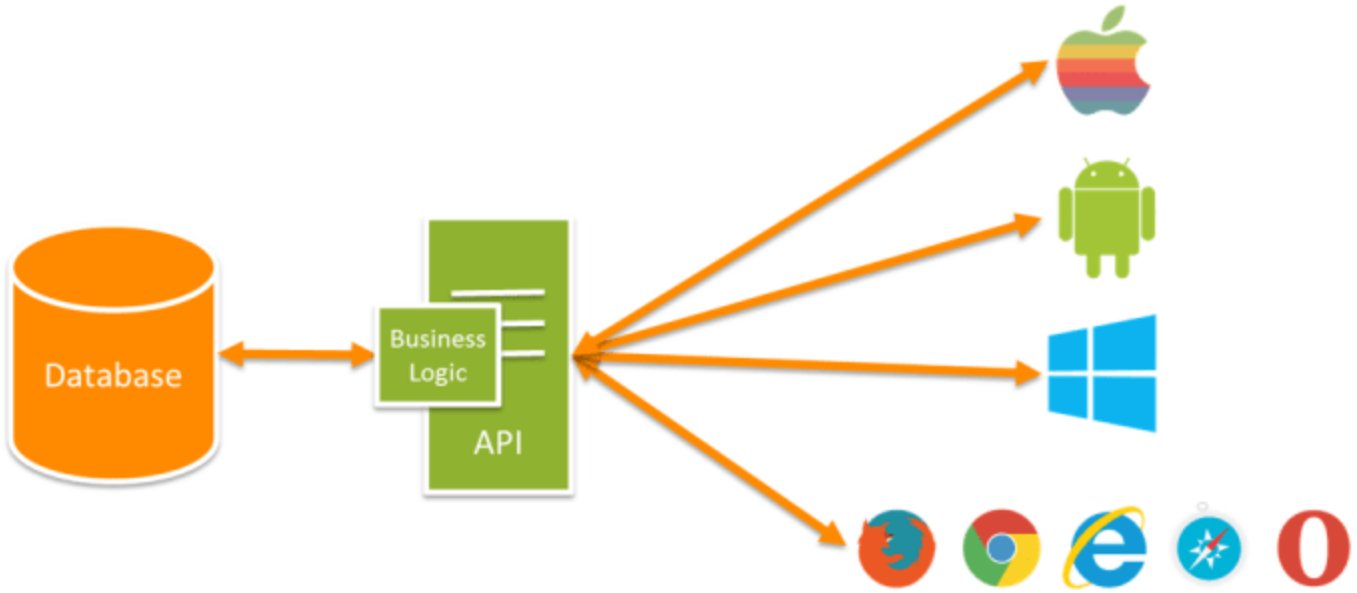
"Um recurso é um elemento abstrato e que nos permite mapear qualquer coisa do mundo real como um elemento para acesso via Web."

Esse mapeamento pode resultar em uma URI
(Identificador Uniforme de Recursos):

http://targettrust.com.br/cursos/imersaojava

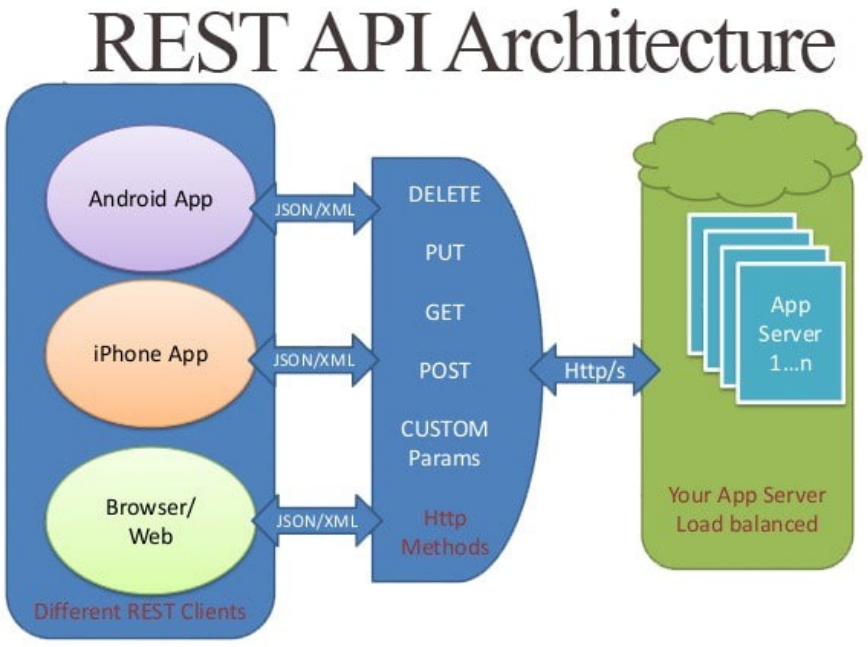
## Utilização

Utilizamos REST para disponibilizar APIs (Application Programming Interface) a partir das nossas aplicações permitindo a interoperabilidade estas.



# Interfaces Uniforme

O HTTP nos fornece uma interface de operações padronizadas (verbos) e respostas adequadas (status).




## Reprentações

São formas de encapsular as informações de determinado recurso. Na Web é muito comum o HTML, mas para troca de dados entre aplicações, existem formatos mais adequados como XML e JSON.

XML:

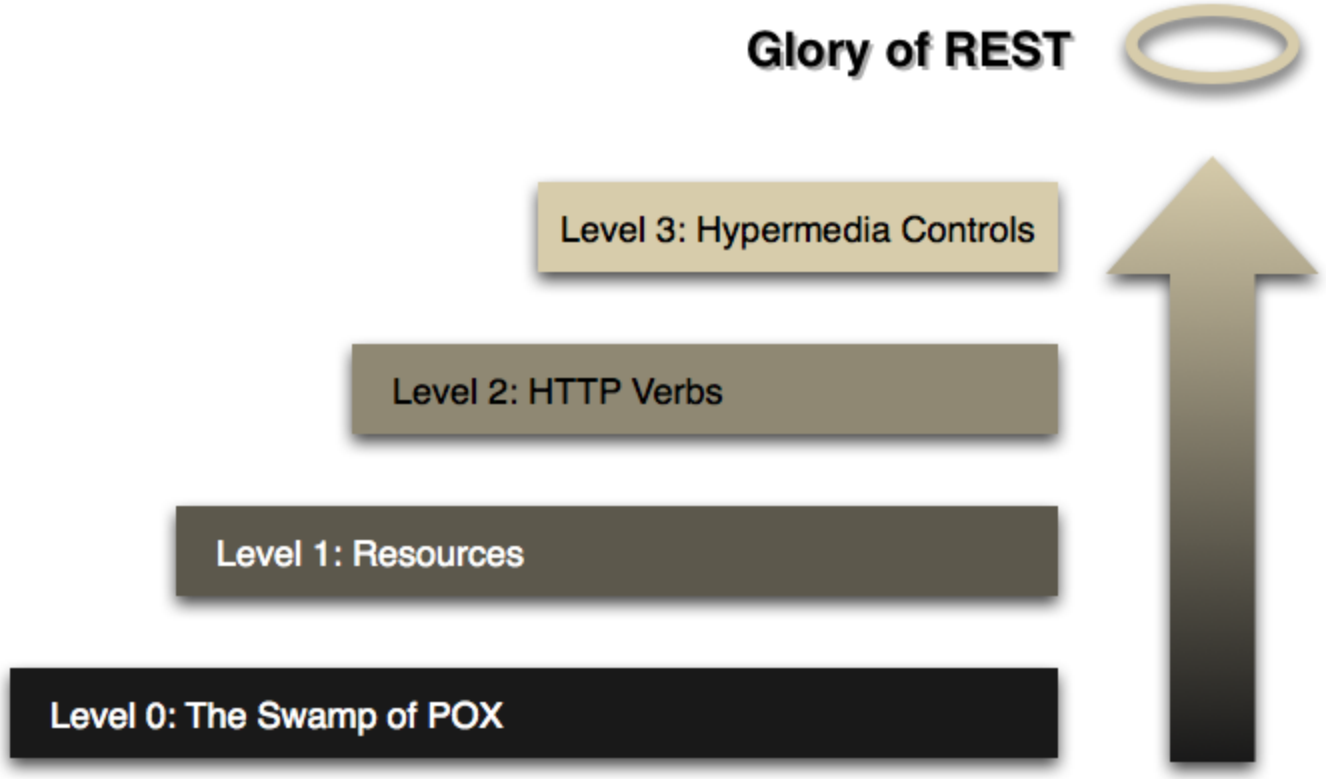
```
<endereco><rua>Rua Borges de Medeiros</rua></endereco>
```

JSON:

```
{ "endereco": { "rua": "Rua Borges de Medeiros"  }}
```

## Hypermedia

Este é o estado mais alto de uma API REST, onde é possível navegar pelos recursos utilizando link's, como fazemos na WEB.




Richardson Maturity Model

## Alguns princípios (extras)

- Separação entre cliente e servidor
- Stateless
- Cacheable
- Divisão em camadas (que o cliente não sabe)

## Formas de enviar os dados

- Path Parameters: /users/{id}
- Query Parameters: /users?role=admin
- Header Parameters: X-MyHeader: Value
    - *Cookie:* Cookie: debug=0; csrftoken=BUSe35dohU
- Body: corpo da requisição.

## Path Parameters

- São partes da URI
- Usados normalmente para obter dados de um item de uma coleção
- São sempre obrigatórios
- Podem possuir subrecursos
- Exemplos:

```
GET /users/{id}
GET /cars/{carId}/drivers/{driverId}
GET /report.{format}
```

## Query Parameters

- Usados normalmente como filtros, ordenação, paginação
- Podem ser opcionais
- Exemplos:

```
GET /pets/findByStatus?status=available
GET /notes?offset=100&limit=50
```

## Header Parameters

- São utilizados no cabeçalho da requisição, ou seja, não fazem parte do recurso.
- Usados para especificar: Content-Type, Accept, Authorization
- Podem ser criados Headers personalizados.

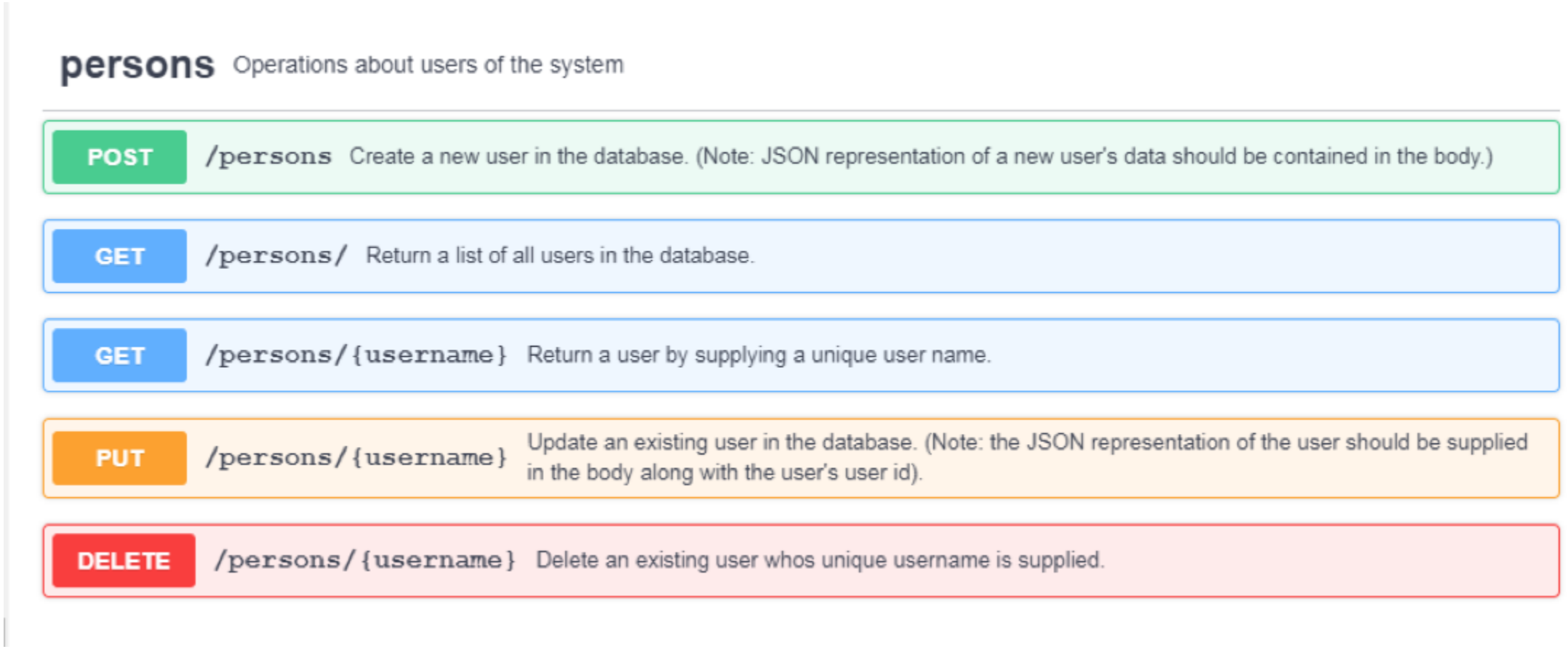
# REST

## Verbos HTTP (operações padronizadas)

- GET - Obter recursos.
- POST - Criar recursos.
- PUT - Atualizar recursos.
- DELETE - Remover recursos.
- PATCH - Atualizar parcialmente recursos.
- HEAD: Obtém apenas cabeçalhos da requisição.

# REST

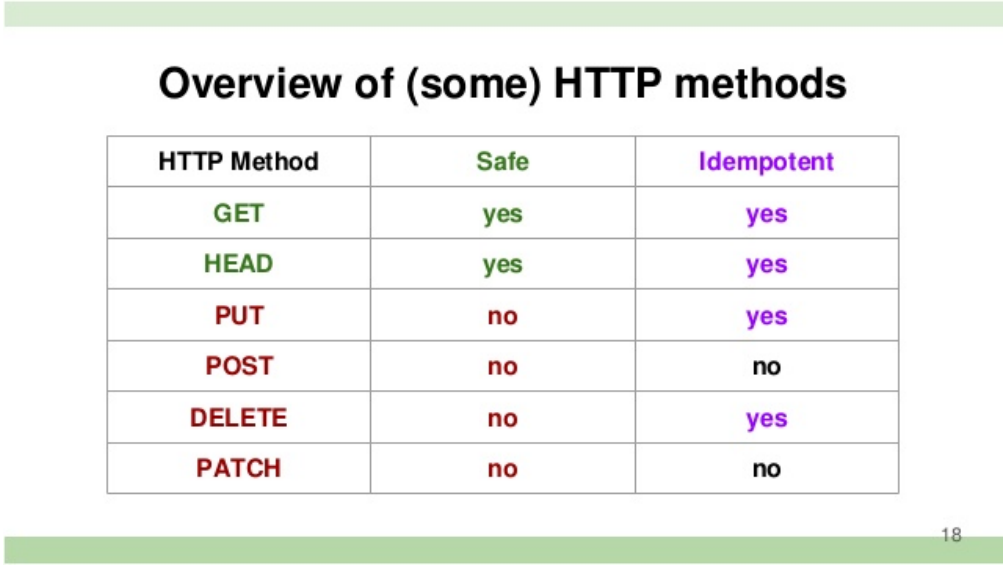
Exemplo de operações CRUD com Rest:




Vamos fazer exercício juntos!

# REST

## Em casos mais complexos, considerar:



**Overview of (some) HTTP methods**

| HTTP Method | Safe | Idempotent |
|---|---|---|
| GET | yes | yes |
| HEAD | yes | yes |
| PUT | no | yes |
| POST | no | no |
| DELETE | no | yes |
| PATCH | no | no |

18

Idempotente: o retorno/ação são os mesmos, mesmo chamando N vezes!

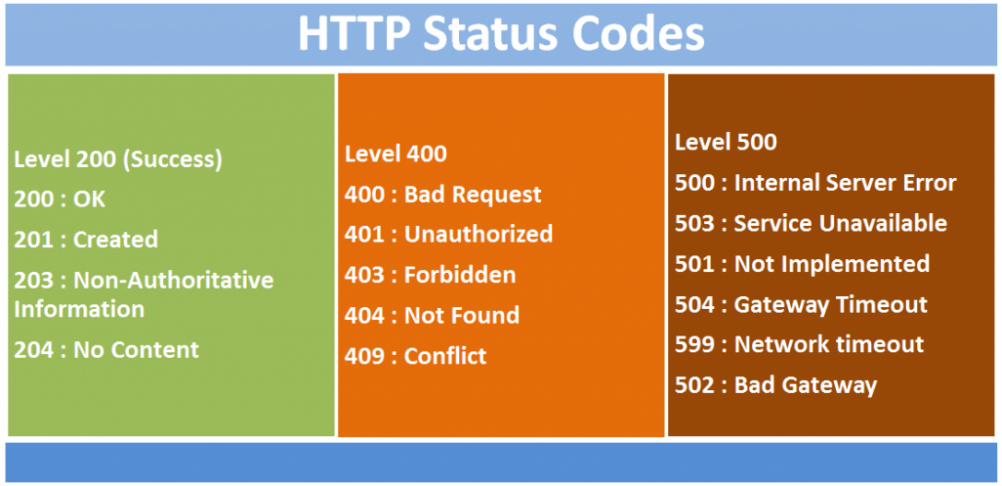
Seguro: Não realiza alterações no recurso!

# REST

## Status HTTP - famílias de erros:

# REST

## Status HTTP mais comuns:




Mais em: developer.mozilla.org e httpstatuses.com.

# REST - HTTP Request e Response

| | | |
|---|---|---|
| GET /index.html HTTP/1.1 | Request Line | HTTP Request |
| Date: Thu, 20 May 2004 21:12:55 GMT<br>Connection: close | General Headers | |
| Host: www.myfavoriteamazingsite.com<br>From: joebloe@somewebsitesomewhere.com<br>Accept: text/html, text/plain<br>User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1) | Request Headers | |
| | Entity Headers | |
| | | |
| | Message Body | |

| | | |
|---|---|---|
| HTTP/1.1 200 OK | Status Line | HTTP Response |
| Date: Thu, 20 May 2004 21:12:58 GMT<br>Connection: close | General Headers | |
| Server: Apache/1.3.27<br>Accept-Ranges: bytes | Response Headers | |
| Content-Type: text/html<br>Content-Length: 170<br>Last-Modified: Tue, 18 May 2004 10:14:49 GMT | Entity Headers | |
| | | |
| <html><br><head><br><title>Welcome to the Amazing Site!</title><br></head><br><body><br> This site is under construction. Please come back later. Sorry! <br></body><br></html> | Message Body | |

# Spring Boot - REST

## Anotações já utilizadas no Spring MVC

- @Controller:
  Stereotype que indica a camada de Controller.
- @RequestMapping:
  Indica que a classe recebe requisições web e também a rota que ativa cada método.
  Continuamos utilizando, pois REST usa URIs.
  Usaremos o atributo:
  `produces = MediaType.APPLICATION_JSON_VALUE`

# Spring Boot - Controller

## Anotações para o Request

- @RequestBody
  Mapeia o corpo da requisição como parâmetro.
- @PathVariable
  Mapeia os Path Parameters /aluno/{**id**}.
- @RequestParam:
  Mapeia os Query Parameters /aluno/?**nome**=ABC.

# Spring Boot - Controller

## Anotações para o Response

- @ResponseBody
  Mapeia o objeto de retorno do método para JSON.
- @RestController
  É uma união do @Controller and @ResponseBody, ou seja, se utilizada substitui ambas.

Mas usaremos também o `ResponseEntity` que nos permite um melhor controle.

## Spring Boot - REST - Anotações do Jackson

- @JsonIgnore:
  Ignora o campo na (des/)serialização.
- @JsonProperty:
  Permite um nome diferente do JSON e do Campo.
- @JsonIgnoreProperties({ "bookName", "bookCategory" })
  Ignora os campos a nível de classe.
- @JsonIgnoreProperties(ignoreUnknown = true)
  Ignora os campos desconhecidos, sem dar exceção.
- @JsonIgnoreType
  Ignora toda a classe anotada.

# Spring Boot - REST

## Anotações para tratamento de exceções

- @ExceptionHandler
- @ResponseStatus