

REST API's RESTFul from 0 to AWS with Spring Boot and Docker Theory

0202 O Que São Webservices



Leandro Costa



<http://www.erudio.com.br/>

YouTube <https://bit.ly/2lkHwoa>



<http://www.semeru.com.br/>



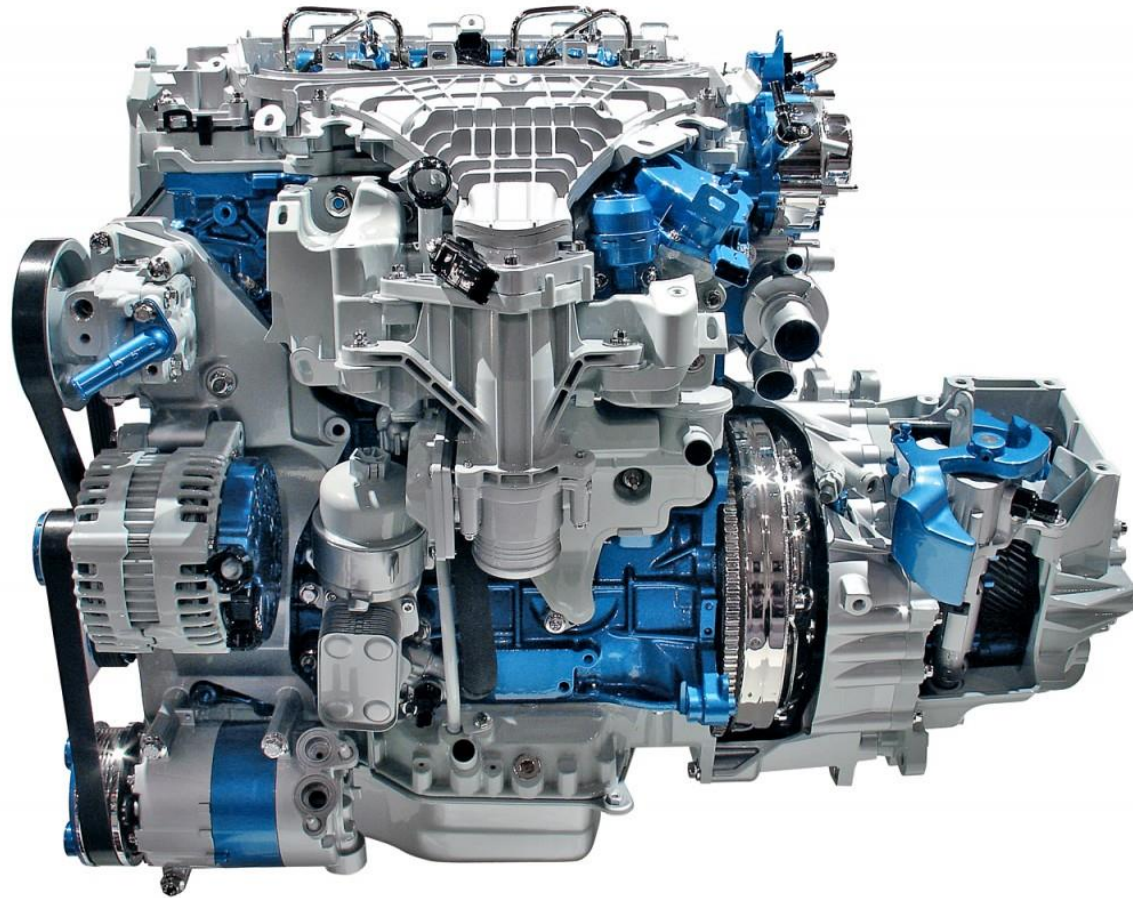
<https://hub.docker.com/u/leandrocgisi/>



<https://github.com/leandrocgisi/RestWithSpringBootUdemy>



O que são Webservices?



A definição do W3C

Para o W3C os webservices são aplicações cliente servidor que se comunicam pela World Wide Web's (WWW) através do protocolo HTTP (HyperText Transfer Protocol) possibilitando a interoperabilidade entre softwares e aplicações executando em uma grande variedade de plataformas e frameworks.

Caracterizam-se por sua grande interoperabilidade e extensibilidade podendo ser combinados de forma baixamente acoplada para executarem operações complexas. Programas proveem simples serviços que podem interagir uns com os outros gerando soluções sofisticadas.

No início...

~ tudo era muito obscuro ~



O mundo antes do REST!



Muitos ‘padrões’ diferentes:

RMI, SOAP, Corba, DCE, DCOM etc.

De muitas empresas diferentes:

Sun, Microsoft, IBM, OASIS, OMG etc.

Causando muitos problemas:

- ✓ Má interoperabilidade;
- ✓ Reinventar a roda;
- ✓ Atrelado a um fornecedor.

0203 SOAP x REST



Leandro Costa



<http://www.erudio.com.br/>

YouTube <https://bit.ly/2lkHwoa>



<http://www.semeru.com.br/>



<https://hub.docker.com/u/leandrocgisi/>



<https://github.com/leandrocgisi/RestWithSpringBootUdemy>



REST x SOAP!



	SOAP	REST
1	Protocolo de troca de mensagens em XML	Um estilo arquitetural
2	Usa WSDL na comunicação entre cliente e servidor	Usa XML, JSON etc. para enviar e receber dados
3	Invoca serviços através de chamadas de método RPC	Simplesmente chama serviços via URL PATH
4	Não retorna um resultado facilmente legível para humanos	Resultado legível por humanos já que é simplesmente JSON ou XML por exemplo
5	Comunicação feita por HTTP mas pode usar outros protocolos como SMTP, FTP etc	Comunicação feita unicamente por HTTP
6	JavaScript pode invocar um serviço SOAP mas essa implementação é bastante complexa de se fazer	Fácil de invocar via JavaScript
7	Comparado com REST sua performance não é das melhores	Comparado com SOAP a performance é melhor consome menos recursos de processamento, código mais enxuto etc



MARTIN LAWRENCE

BIG MOMMA'S

SOAP

Client



Data

Server

Client



Data



SOAP



SOAP Standards
(Envelope)

Server

SOAP

Client



Data + SOAP
Standards (Envelope)

Server

SOAP

Client



Data + SOAP
Standards (Envelope)

Server

SOAP

Client



Data + SOAP
Standards (Envelope)

Server

Os dados trafegados entre cliente e servidor se tornam consideravelmente maiores. Assim como a personagem Vovozona de Martin Lawrence.

REST

Client



Data

Server

No REST os dados são enviados da forma que estão

0204 REST



Leandro Costa



<http://www.erudio.com.br/>

YouTube <https://bit.ly/2lkHwoa>



<http://www.semeru.com.br/>



<https://hub.docker.com/u/leandrocgisi/>



<https://github.com/leandrocgisi/RestWithSpringBootUdemy>



Definindo o REST!

“Representational State Transfer (REST) é um estilo de arquitetura de software para sistemas distribuídos de hipermídia, como a World Wide Web”

REST é baseado em um conjunto de constraints

1. Cliente-servidor

Clientes e servidores separados.

2. Stateless server

O servidor não deve guardar o estado do cliente. Cada request de um cliente contém todas as informações necessárias para atendê-la.

3. Cacheable

O cliente deve ser informado sobre as propriedades de cache de um recurso para que possa decidir quando deve ou não utilizar cache.

4. Interface uniforme

Existe uma interface uniforme entre cliente e servidor.

- ▶ Identificação de recursos (URI).
- ▶ Manipulação de recursos a partir de suas representações.
- ▶ Mensagens auto descritivas.
- ▶ Hypermedia as the engine of application state - HATEOAS

5. Sistema em camadas

Deve suportar conceitos como balanceamento de carga, proxies e firewalls.

6. Código sob Demanda (opcional)

O cliente pode solicitar o código do servidor e executá-lo.



XML



JSON



CSV



Texto





HTML



PDF



Binário



etc

Vantagens dos Web Services RESTful

- ▶ REST é um padrão arquitetural basicamente leve por natureza. Então quando você tiver limitações de banda prefira web services REST;
- ▶ Desenvolvimento fácil e rápido;
- ▶ Aplicativos Mobile tem ganhado cada vez mais espaço e precisam interagir rapidamente com os servidores e o padrão REST é mais rápido no processamento de dados das requests e responses.

Clube dos Bilionários

1+ Bilhão de Requests / Dia



5+ Bilhão de Requests / Dia



(Latin America)
7,2 Bilhão de Requests / Mês



0205 Request e Response



Leandro Costa



<http://www.erudio.com.br/>

YouTube <https://bit.ly/2lkHwoa>



<http://www.semeru.com.br/>



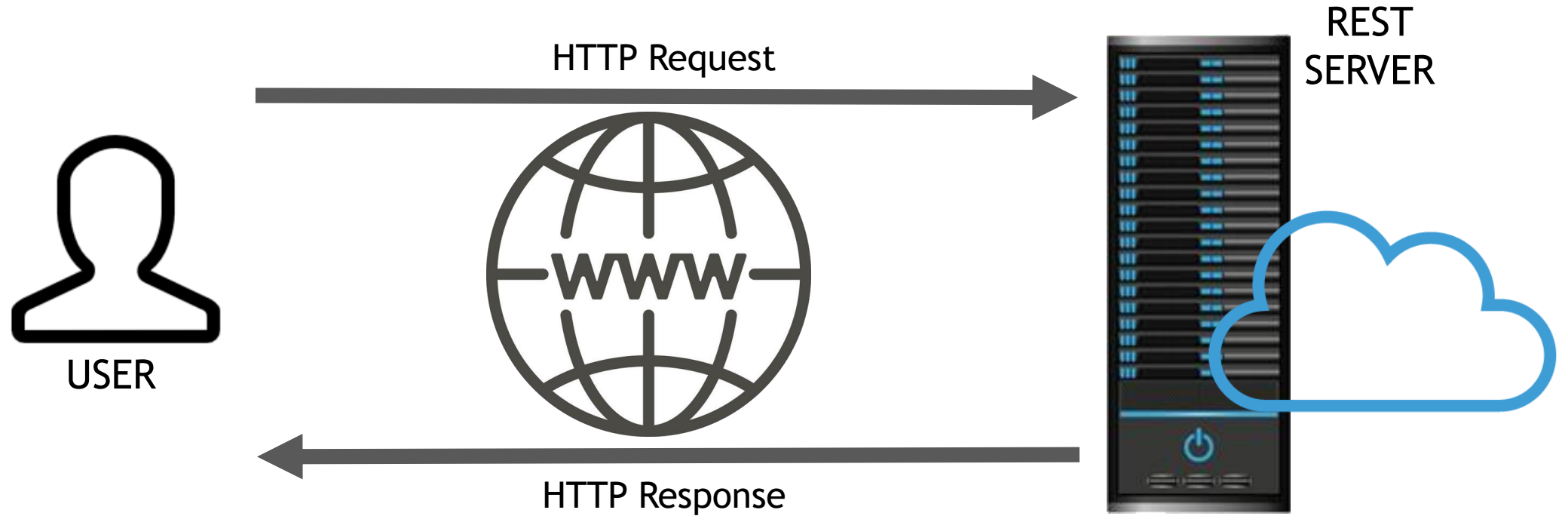
<https://hub.docker.com/u/leandrocgisi/>



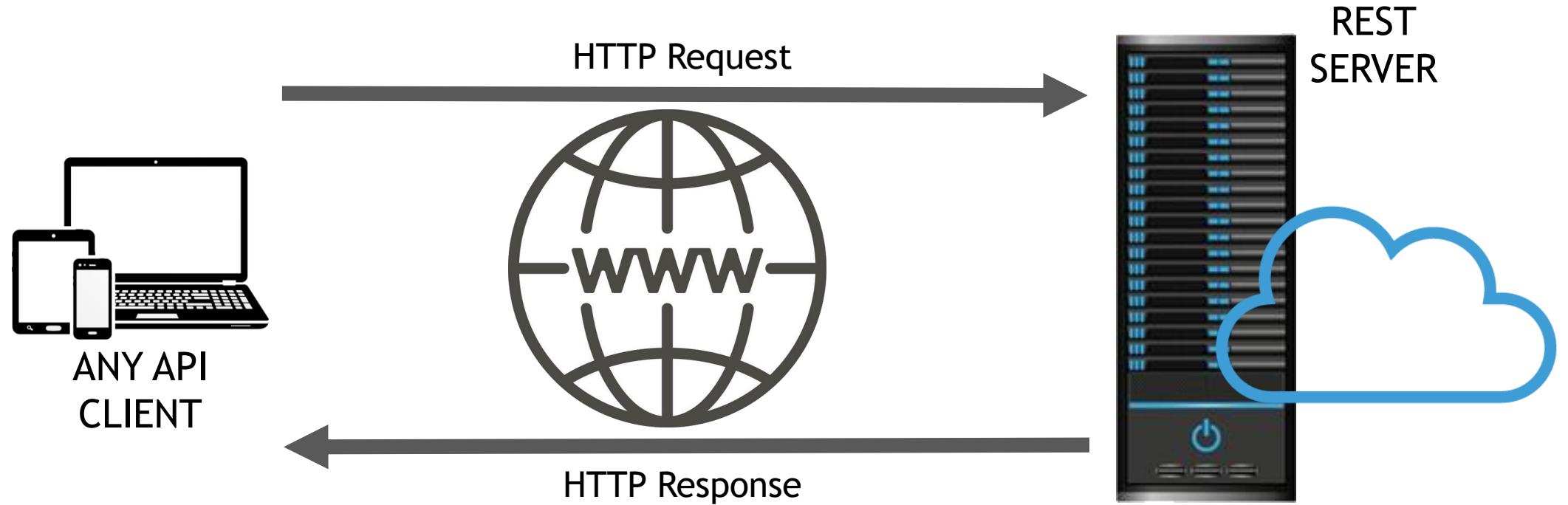
<https://github.com/leandrocgisi/RestWithSpringBootUdemy>



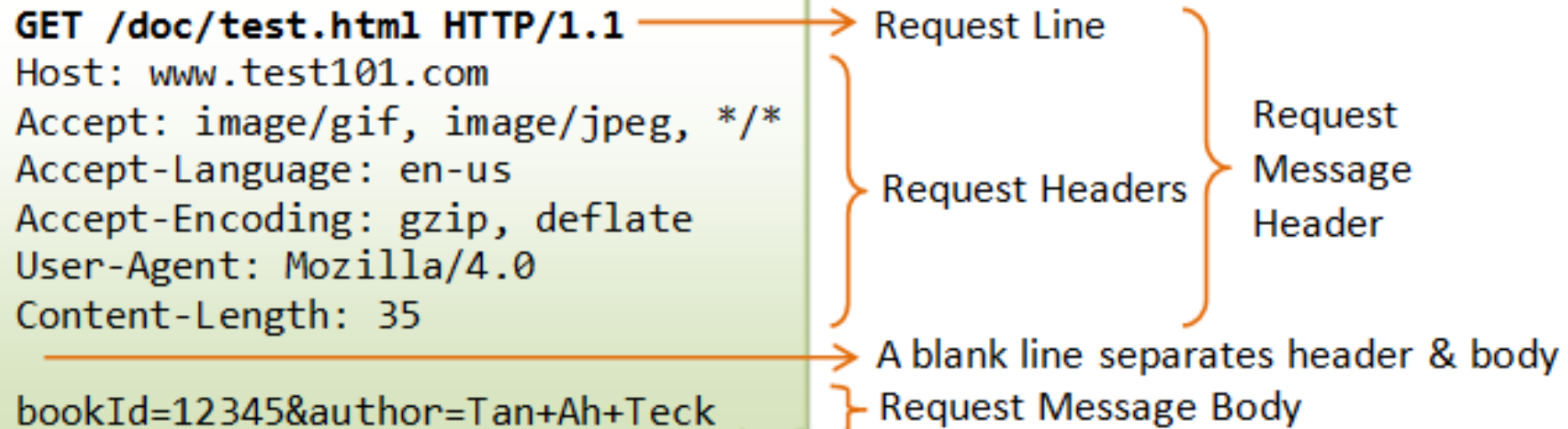
Request e Response



Request e Response



Request



The diagram illustrates the structure of an HTTP request. It features a light green box containing the raw request text. To the right of the box, orange arrows and curly braces point to specific parts of the request, labeling them. The request text is as follows:

```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

bookId=12345&author=Tan+Ah+Teck
```

The annotations on the right are:

- Request Line**: Points to the first line of the request, `GET /doc/test.html HTTP/1.1`.
- Request Headers**: A bracket groups the lines from `Host: www.test101.com` to `Content-Length: 35`.
- Request Message Header**: A bracket groups the **Request Line** and the **Request Headers**.
- A blank line separates header & body**: Points to the empty line between the headers and the body.
- Request Message Body**: A bracket groups the body text, `bookId=12345&author=Tan+Ah+Teck`.

Response

HTTP/1.1 200 OK

Date: Sun, 08 Feb xxxx 01:11:12 GMT

Server: Apache/1.3.29 (Win32)

Last-Modified: Sat, 07 Feb xxxx

ETag: "0-23-4024c3a5"

Accept-Ranges: bytes

Content-Length: 35

Connection: close

Content-Type: text/html

<h1>My Home page</h1>

Status Line

Response Headers

Response
Message
Header

A blank line separates header & body

Response Message Body

0206 Tipos de Parâmetros



Leandro Costa



<http://www.erudio.com.br/>

YouTube <https://bit.ly/2lkHwoa>



<http://www.semeru.com.br/>



<https://hub.docker.com/u/leandrocgisi/>



<https://github.com/leandrocgisi/RestWithSpringBootUdemy>





https://your_host/api/books/v1/find-with-paged-search/asc/10/1



https://your_host/api/books/v1/find-with-paged-search/asc/10/1



https://your_host/api/books/v1/find-by-title?firstName=Clean&lastName=Coder



https://your_host/api/books/v1/find-by-title?firstName=Clean&lastName=Coder



https://your_host/api/books/v1/find-by-title?firstName=Clean&lastName=Coder&otherParam=value



https://your_host/api/books/v1/find-by-title?firstName=Clean&lastName=Coder&otherParam=value&otherParam=value



https://your_host/api/books/v1/find-by-title?lastName=Coder



https://your_host/api/books/v1/find-by-title?firstName=Clean

Tipos de Parâmetros - Header Params

GET

▼

{{host}}:{{port}}/api/file/v1

Params

Authorization

Headers (3)

Body

Pre-request Script

Tests ●

Settings

▼ Headers (3)

	KEY	VALUE
<input checked="" type="checkbox"/>	Accept	text/plain, application/json, text/json, application/xml, text/xml
<input checked="" type="checkbox"/>	Content-Type	application/json
<input checked="" type="checkbox"/>	Authorization	Bearer {{bearer_token}}
	Key	Value

Tipos de Parâmetros - Body Params

POST ▼

{{host}}:{{port}}/api/login/v1

Params

Authorization

Headers (2)

Body ●

Pre-request Script

Tests ●

Settings



none



form-data



x-www-form-urlencoded



raw



binary



GraphQL ^{BETA}

JSON ▼

1 ▼

```
[  
  "login": "{{user}}",  
  "accessKey": "{{password}}"  
]
```

0207 Status Codes



Leandro Costa



<http://www.erudio.com.br/>

YouTube <https://bit.ly/2lkHwoa>



<http://www.semeru.com.br/>



<https://hub.docker.com/u/leandrocgisi/>



<https://github.com/leandrocgisi/RestWithSpringBootUdemy>



HTTP Status Codes

1xx Informacionais

2xx Sucesso

3xx Redirecionamento

4xx Erro de Client

5xx Erro de Server

1xx Informativos

100 Continue

101 Switching Protocols

102 Processing

2xx Sucesso

200 OK

201 Created

202 Accepted

203 Non-authoritative
Information

204 No Content

205 Reset Content

206 Partial Content

207 Multi-Status

208 Already Reported

226 IM Used

3xx Redirecionamiento

300 Multiple Choices

301 Moved Permanently

302 Found

303 See Other

304 Not Modified

305 Use Proxy

307 Temporary Redirect

308 Permanent Redirect

4xx Erro de Client

- 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 Payload Too Large
- 414 Request-URI Too Long
- 415 Unsupported Media Type
- 416 Requested Range Not Satisfiable
- 417 Expectation Failed
- 418 I'm a teapot
- 421 Misdirected Request
- 422 Unprocessable Entity
- 423 Locked
- 424 Failed Dependency
- 426 Upgrade Required
- 428 Precondition Required
- 429 Too Many Requests
- 431 Request Header Fields Too Large
- 444 Connection Closed Without Response
- 451 Unavailable For Legal Reasons
- 499 Client Closed Request

5xx Erro de Server

500 Internal Server Error

501 Not Implemented

502 Bad Gateway

503 Service Unavailable

504 Gateway Timeout

505 HTTP Version Not Supported

506 Variant Also Negotiates

507 Insufficient Storage

508 Loop Detected

510 Not Extended

511 Network Authentication Required

599 Network Connect Timeout Error

HTTP Status Codes em Serviços REST

- ▶ **200 OK** - Request de criação ou deleção executada com sucesso.
- ▶ **201 Created** - Criação de uma fila, tópico, fila temporária, tópico temporária, session, producer, consumer, listener, queue browser ou mensagem realizada com sucesso.
- ▶ **204 No Content** - deleção de uma fila, tópico, sessão, producer ou listener bem sucedida mas sem retorno de conteúdo.

HTTP Status Codes em Serviços REST

- ▶ **400 Bad Request** - O path informado está em um formato incorreto, um parâmetro ou valor do corpo da requisição não está formatado corretamente ou um parâmetro obrigatório não foi informado, ou está formatado corretamente mas pode estar eventualmente inválido (por exemplo, o ID informado não existe - NullPointerException, o conteúdo retornado é muito grande ou o ID informado já está em uso).
- ▶ **401 Unauthorized** - O cliente não tem autorização para executar requisições na operação em questão.
- ▶ **403 Forbidden** - O cliente não tem permissão para executar requisições na operação em questão.

HTTP Status Codes em Serviços REST

- ▶ **404 Not Found** - o objeto requisitado pelo path não existe (NullPointerException ou NullPointerException).
- ▶ **405 Method Not Allowed** - O usuário não tem permissão de acesso ao path.
- ▶ **409 Conflict** - An attempt was made to create an object that already exists.
- ▶ **500 Internal Erro de Server** - Ocorreu uma falha no servidor, podendo ser desde uma falha no SQL por exemplo.

0208 Os Verbos HTTP e o REST



Leandro Costa



<http://www.erudio.com.br/>

YouTube <https://bit.ly/2lkHwoa>



<http://www.semeru.com.br/>



<https://hub.docker.com/u/leandrocgisi/>

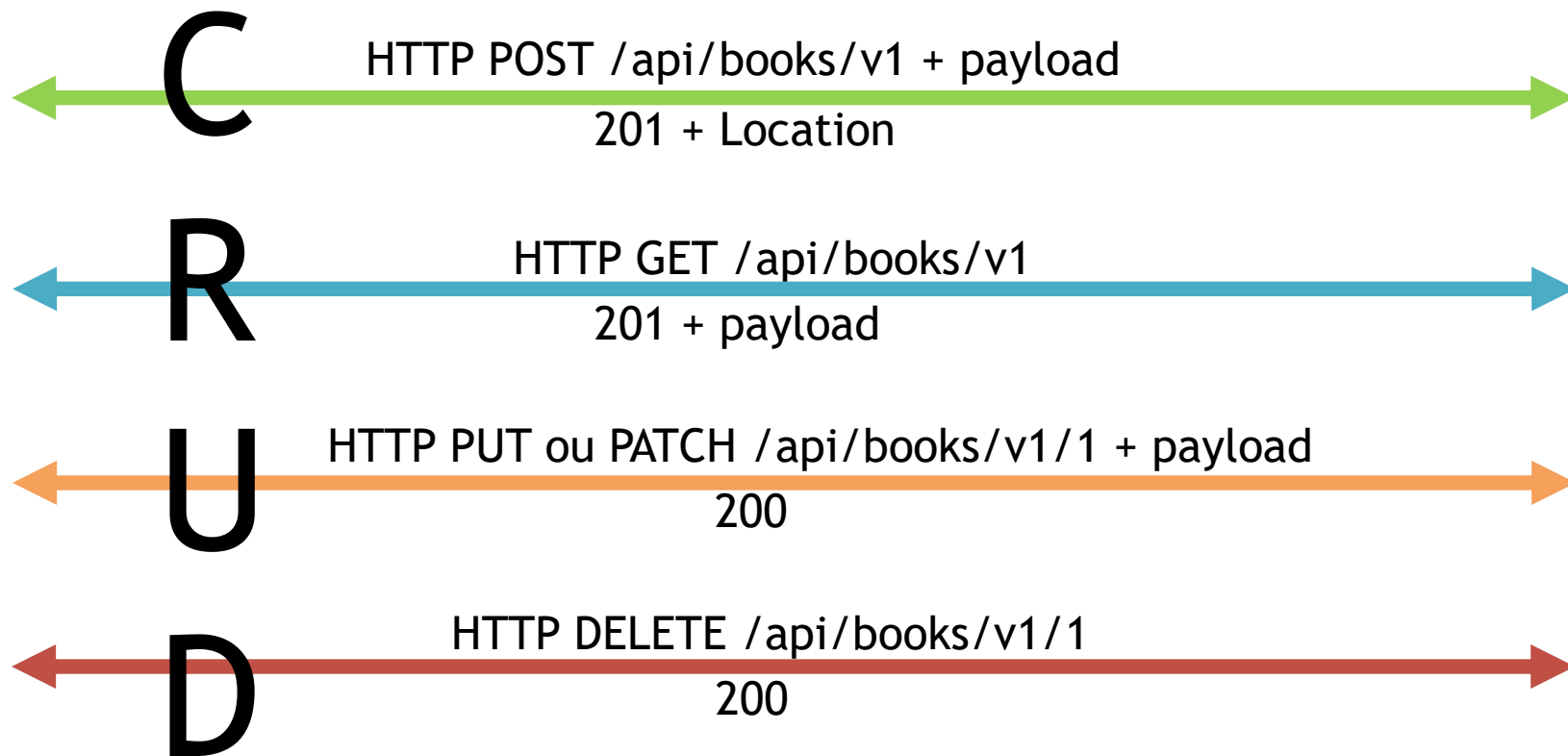


<https://github.com/leandrocgisi/RestWithSpringBootUdemy>





REST
CLIENT



REST
SERVER



GET - READ / para selecionar/recuperar um recurso

- ▶ O verbo HTTP GET é usado para ler ou recuperar uma representação de um recurso. Em um “cenário feliz”, uma requisição GET retorna uma representação em XML ou JSON e um HTTP status code 200 (OK). Em um cenário de erro o retorno mais comum é 404 (NOT FOUND) ou 400 (BAD REQUEST).

Parâmetros suportados

- ▶ Via URL (PATH ou QUERY PARAMS)
- ▶ Via HEADER

POST - CREATE / para inserir recurso

- ▶ O verbo **HTTP POST** é mais frequentemente usado para criar novos recursos — inserir um novo item na base.
- ▶ Em uma aplicação **REST** perfeita quando uma operação é executada com sucesso, retorna-se o status code 200 ou 201.

Parâmetros suportados

- ▶ Via URL (PATH ou QUERY PARAMS)
- ▶ Via HEADER
- ▶ Via Body

PUT - UPDATE / para modificar um recurso

- ▶ O verbo **PUT** é comumente usado para atualizar informações, colocando um recurso conhecido no (body) corpo da requisição contendo novas informações que representam o recurso original;
- ▶ Um update bem sucedido, retorna um **status code** 200 (ou 204 quando não retorna nenhum conteúdo no body).

Parâmetros suportados

- ▶ Via URL (PATH ou QUERY PARAMS)
- ▶ Via HEADER
- ▶ Via Body

DELETE - DELETE / para remover um recurso

- ▶ O verbo DELETE é fácil de entender, ele é usado para deletar um recurso identificado por uma URI;
- ▶ Em uma deleção bem sucedida retorna-se um **status code 200 (OK)** juntamente com um response body, possivelmente uma representação do item deletado (o que acaba por demandar muita banda), ou uma response customizada;
- ▶ Ou retornar o **status code 204 (NO CONTENT)** sem response body ou um **status code 204** sem corpo, ou **JSEND-style response** com um **status code 200** são as responses mais recomendadas.

Parâmetros suportados

- ▶ Via URL (PATH ou QUERY PARAMS)
- ▶ Via HEADER
- ▶ Via Body

0209 Os Verbos HTTP Menos Conhecidos



Leandro Costa



<http://www.erudio.com.br/>

YouTube <https://bit.ly/2lkHwoa>



<http://www.semeru.com.br/>



<https://hub.docker.com/u/leandrocgisi/>



<https://github.com/leandrocgisi/RestWithSpringBootUdemy>



Outros verbos menos conhecidos

- ▶ **PATCH** - O verbo PATCH pode ser usado para realizar updates parciais de um recurso. Por exemplo, quando você precisar alterar apenas um campo em um recurso, executar um POST com todo o objeto é pesado e acarreta em um maior consumo de banda.
- ▶ Use-o com moderação pois colisões entre múltiplas PATCH requests são mais perigosas que colisões entre PUT requests por que exige que o cliente tenha informações básicas do recurso ou irão corrompê-lo.

Outros verbos menos conhecidos

- ▶ **HEAD** - O verbo **HEAD** possui uma funcionalidade similar ao verbo **GET**, exceto pelo fato do servidor retornar uma response line e headers, mas sem um entity-body.

Outros verbos menos conhecidos

- ▶ **TRACE** - O verbo **TRACE** é usado para recuperar o conteúdo de uma requisição **HTTP** de volta podendo ser usado com o propósito de debug durante o processo de desenvolvimento.

Outros verbos menos conhecidos

- ▶ **OPTIONS** - O verbo **OPTIONS** é usado pelo cliente para encontrar operações **HTTP** e outras opções suportadas pelo servidor. O cliente pode especificar uma **URL** para o verbo **OPTIONS** ou um asterisco (*) para se referir a todo o servidor.

Outros verbos menos conhecidos

- ▶ **CONNECT** - O verbo **CONNECT** é usado pelo cliente para estabelecer uma conexão de rede com um servidor via **HTTP**.

0210 Níveis de Maturidade do REST



Leandro Costa



<http://www.erudio.com.br/>

YouTube <https://bit.ly/2lkHwoa>



<http://www.semeru.com.br/>



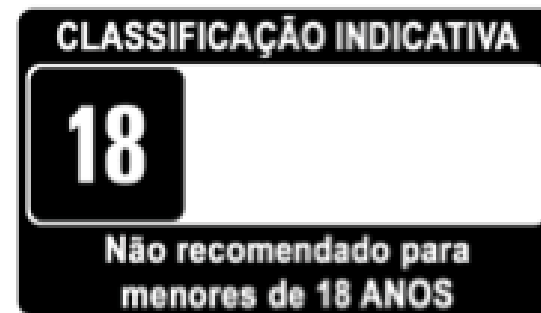
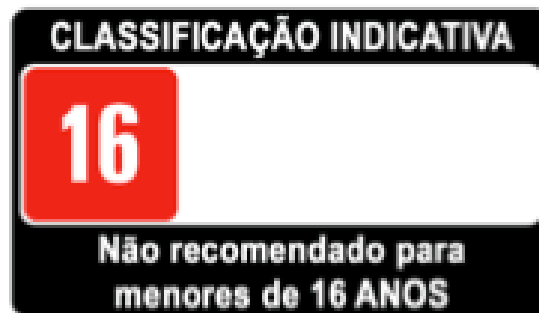
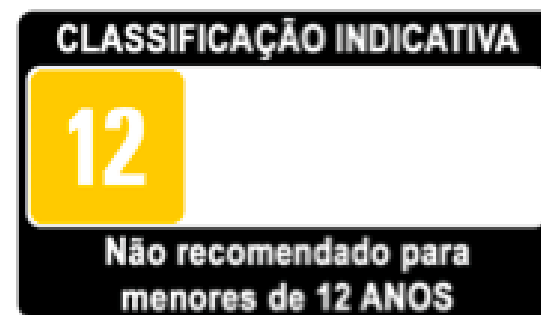
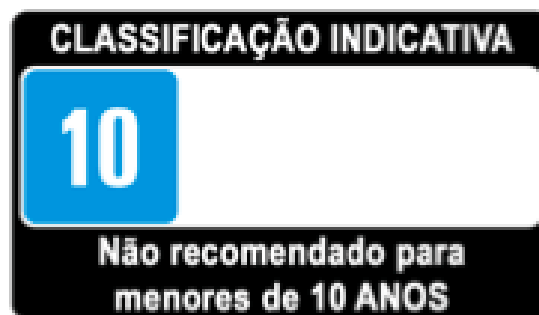
<https://hub.docker.com/u/leandrocgisi/>



<https://github.com/leandrocgisi/RestWithSpringBootUdemy>

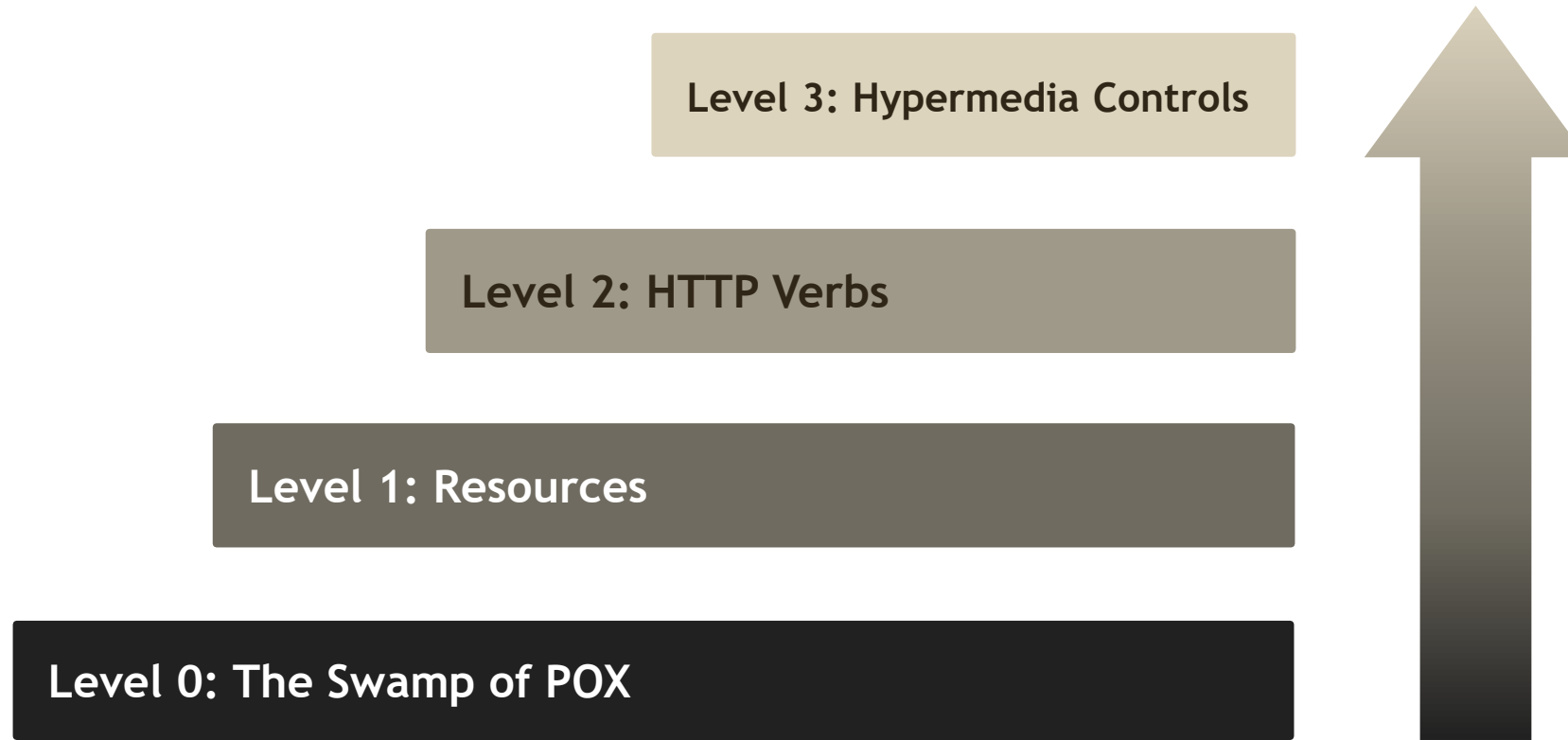


Richardson Maturity Model



Richardson Maturity Model

Glory of REST



* POX = Plain Old XML, Richardson Maturity Model

Então, são nível 0, 1, 2 e RESTful?

“O que precisa ser feito para tornar o estilo de arquitetura **REST** claro sobre a noção de que o **hipertexto é uma restrição**? Em outras palavras, se o mecanismo do estado do aplicativo (e, portanto, a API) **não estiver sendo orientado ao hipertexto**, então **não poderá ser RESTful** e não poderá ser uma API RESTful por completo. Há algo de errado em algum lugar que precisa ser consertado?

0211 HATEOAS



Leandro Costa



<http://www.erudio.com.br/>

YouTube <https://bit.ly/2lkHwoa>



<http://www.semeru.com.br/>



<https://hub.docker.com/u/leandrocgisi/>



<https://github.com/leandrocgisi/RestWithSpringBootUdemy>





Hypermedia As The Engine Of Application State (HATEOAS)



HATEOAS

```
[
  {
    "id": 1,
    "firstName": "Leandro",
    "lastName": "Costa",
    "address": "Uberlândia - Minas Gerais - Brasil",
    "gender": "Male",
    "links": [
      {
        "rel": "self",
        "href": "http://localhost:50904/api/persons/v1/1",
        "type": "application/json",
        "action": "GET"
      },
      {
        "rel": "self",
        "href": "http://localhost:50904/api/persons/v1/1",
        "type": "application/x-www-form-urlencoded",
        "action": "POST"
      },
      {
        "rel": "self",
        "href": "http://localhost:50904/api/persons/v1/1",
        "type": "application/x-www-form-urlencoded",
        "action": "PUT"
      },
      {
        "rel": "self",
        "href": "http://localhost:50904/api/persons/v1/1",
        "type": "int",
        "action": "DELETE"
      }
    ]
  }
]
```



That's all Folks!