

POO2

Encontro 02 – Conceitos de REST API

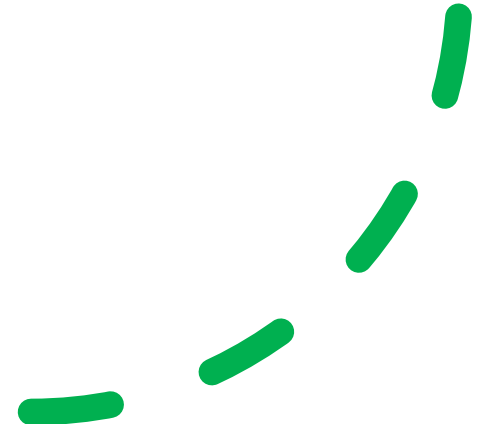
Professor Luiz Augusto Rodrigues

luiz.a.rodrigues@cogna.com.br



Sumário

- HTTP
 - Request / Response
 - Códigos do Servidor HTTP
 - Verbos HTTP
- HTTPS
- REST API
 - Modelo de Maturidade de Richardson
 - Identificação de Recursos
 - Verbos HTTP para REST API
 - Roteamento
 - Padronização de Respostas
 - Versionamento
 - HATEOAS



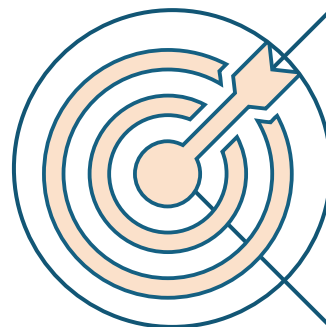
Informações Importantes



Nenhum material será enviado via e-mail. Os materiais serão disponibilizados no **AVA** e no Google Drive da Disciplina.



Dúvidas, questionamentos, entre outros deverão ser realizados pelo **e-mail** e pelo grupo de **Whatsapp** da disciplina.



A Disciplina de Programação Orientada a Objetos, a partir deste slide, será referenciada pela sigla **POO2**.

Informações Importantes

Grupo de Whatsapp da Disciplina:

<https://tinyurl.com/mw2xdkd7>



POO2

**Instruções
Importantíssimas**



Instruções Importantíssimas

Para este semestre, na disciplina de POO, utilizaremos OBRIGATORIAMENTE as seguintes ferramentas:

- Sistema Operacional Windows 10, Linux ou MacOS.
- Linguagem de Programação Java.
- JDK 17 ou superior
 - https://download.oracle.com/java/17/archive/jdk-17.0.6_windows-x64_bin.exe
- VS Code.

Instruções Importantíssimas

- Git 2.39.2 ou superior
 - <https://github.com/git-for-windows/git/releases/download/v2.39.2.windows.1/Git-2.39.2-64-bit.exe>
- Github
 - <https://github.com/>
 - <https://desktop.github.com/>
- Tutorial em Git e Github
 - Tutorial Git e Github 2022 – Introdução prática para iniciantes
 - Canal DevSuperior
 - https://youtu.be/_hZf1teRFNg

Quem for utilizar Linux ou MacOSx, fale comigo após a aula, ou pelo grupo do WhatsApp, para instruções de instalação.

Instruções Importantíssimas

- Os relatórios não serão utilizados nessa disciplina, por se tratar de uma disciplina 90% prática.
- No lugar, teremos Listas de Atividades Formativas (Atividade do Professor), que deverão ser respondidas e justificadas, para fixação do conteúdo discutido nas aulas.
- As atividades de aula estarão disponíveis Google Drive da turma.
- O tutorial do Git e Github é obrigatório.

Pilares de POO



POO2

Conceitos de REST API

HTTP



HTTP – Definições

- O Hypertext Transfer Protocol, sigla HTTP (em português Protocolo de Transferência de Hipertexto) é um protocolo de comunicação (na camada de aplicação segundo o Modelo OSI) utilizado para sistemas de informação de hipermídia, distribuídos e colaborativos.
- Ele é a base para a comunicação de dados da World Wide Web.
- Hipertexto é o texto estruturado que utiliza ligações lógicas (hiperlinks) entre nós contendo texto.
- O HTTP é o protocolo para a troca ou transferência de hipertexto.

HTTP – Características

- Cliente-Servidor.
- Comunicação é feita através de Mensagens HTTP.
- Não é “Keep-Alive”.
- Utilizado de maneira assíncrona.
- Conexões independentes.
- “Stateless”.
- Semântico.

POO2

Conceitos de REST API

HTTP

- Request/Response



HTTP Request

A Request ou requisição traduzindo diretamente para português, é o pedido que um cliente realiza a nosso servidor. Esse pedido contém uma série de dados que são usados para descrever exatamente o que o cliente precisa.

Vamos pensar que um cliente precisa cadastrar um novo produto, ele deve passar todos os dados necessários para o cadastro acontecer de maneira correta, inclusive os dados que foram digitados pelo usuário em um formulário, no caso de uma aplicação web.

No navegador toda vez que trocamos de página ou apertamos enter na barra de endereço uma nova request é feita. Independente se estamos apenas pedindo a exibição de uma página, cadastrando um novo recurso, atualizando ou excluindo.

HTTP Response

Vimos que o cliente envia uma Request (requisição) ao servidor. Essa requisição possui todas as informações acerca do que o cliente espera receber de volta.

O servidor web ao receber essas informações precisa enviar uma resposta ao cliente, nesse ponto entra a Response.

A Response (resposta) nada mais é do que a resposta que o servidor envia ao cliente. Essa resposta pode conter os dados que realmente o cliente esperava receber ou uma resposta informando que alguma coisa deu errado.

IMPORTANTE

Abaixo segue um exemplo de uma comunicação entre um cliente e um servidor HTTP.

- O servidor possui a URL `www.exemplo.com`, porta 80.
- O pedido do cliente (seguido por uma linha em branco, de maneira que o pedido termina com um newline duplo, cada um composto por um carriage return seguido de um line feed): O cabeçalho Host reconhece vários diferentes nomes DNS que tenham o mesmo IP.

```
GET /index.html HTTP/1.1
```

```
Host: www.exemplo.com
```

IMPORTANTE

A resposta do servidor (seguida por uma linha em branco e o texto da página solicitada):

```
HTTP/1.1 200 OK
```

```
Date: Mon, 23 May 2005 22:38:34 GMT
```

```
Server: Apache/1.3.27 (Unix) (Red-Hat/Linux)
```

```
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
```

```
Etag: "3f80f-1b6-3e1cb03b"
```

```
Accept-Ranges: bytes
```

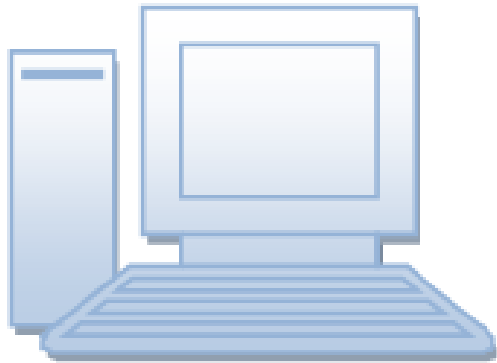
```
Content-Length: 438
```

```
Connection: close
```

```
Content-Type: text/html; charset=UTF-8
```

HTTP Request / Response

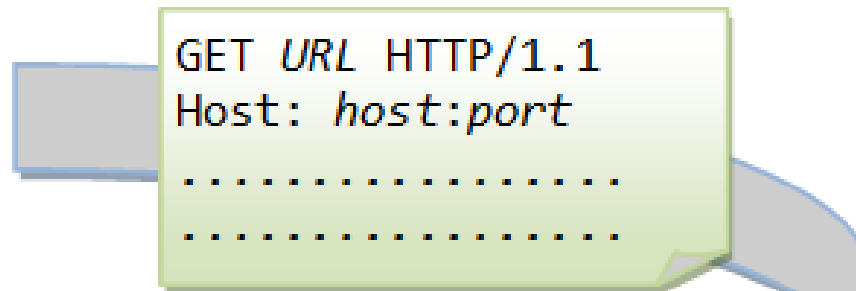
- (1) User issues URL from a browser
`http://host:port/path/file`



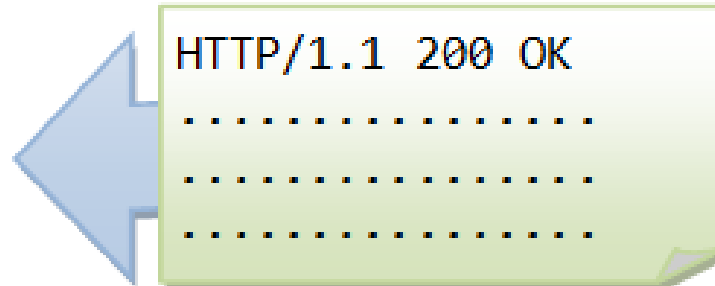
- (5) Browser formats the response
and displays

Client (Browser)

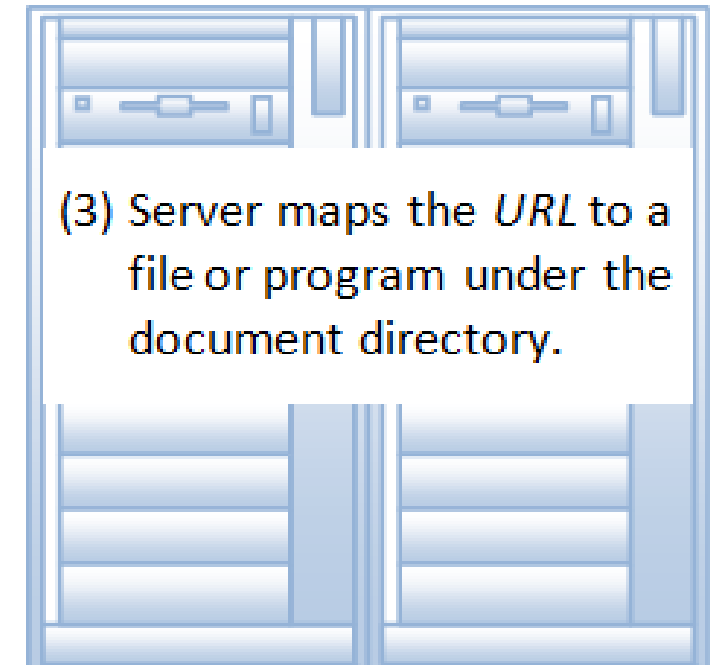
- (2) Browser sends a request message



- (4) Server returns a response message



HTTP (Over TCP/IP)



Server (@ *host:port*)

POO2

Conceitos de REST API

HTTP

- **Códigos do Servidor**



Códigos do Servidor HTTP

Quando o cliente faz uma requisição ele espera uma resposta.

O servidor pode realmente responder o que o cliente esperava ou devolver outra informação, justamente nesse ponto entram os códigos HTTP.

O servidor utiliza um código desse na resposta para indicar o que aconteceu.

Códigos do Servidor HTTP

Os códigos estão entre 100 e 500, sendo que cada centena indica uma categoria:

- 1xx – Informativos
- 2xx – Indicativos de sucesso
- 3xx – Redirecionamentos
- 4xx – Erros do cliente na hora de fazer a solicitação
- 5xx – Erros no lado do servidor

Códigos do Servidor HTTP

Dentro de cada centena temos os códigos específicos, por exemplo:

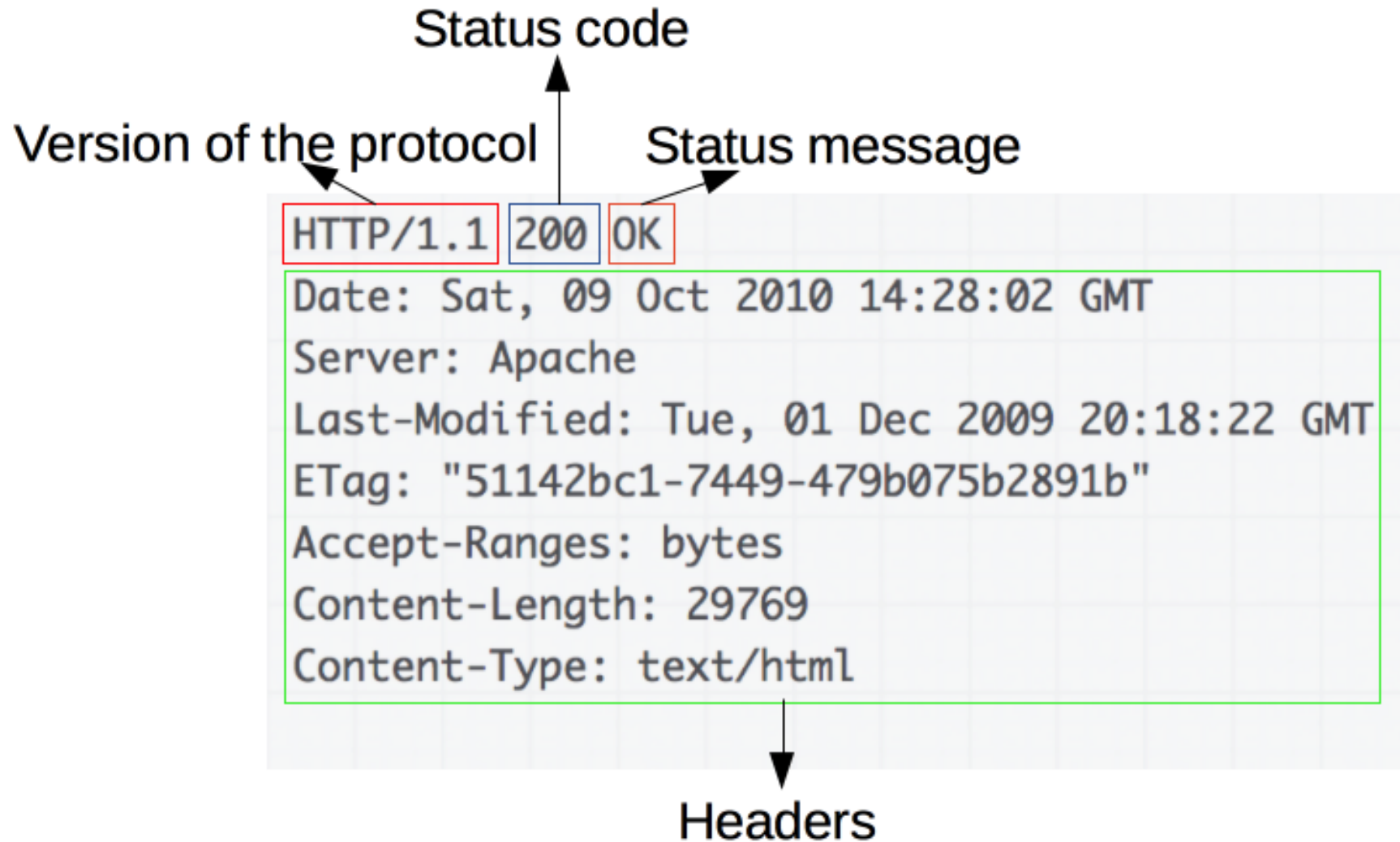
- 200 – Tudo ocorreu corretamente
- 301 – Indica redirecionamento permanente
- 401 – Não autorizado
- 404 – O recurso solicitado não foi encontrado no servidor
- 500 – Erro Interno do Servidor
- 502 – Bad Gateway
- 503 – Serviço Indisponível

Códigos do Servidor HTTP

Para maiores detalhes, verifique estes endereços:

- Wikipedia – Lista de códigos de estado HTTP
https://pt.wikipedia.org/wiki/Lista_de_c%C3%B3digos_de_estado_HTTP
- Mozilla Dev Network - Códigos de status de respostas HTTP
<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status>

IMPORTANTE



POO2

Conceitos de REST API

HTTP

- Verbos HTTP



Verbos HTTP

Uma solicitação HTTP, ou HTTP Request é uma maneira do navegador mostrar uma página da internet utilizando um dos oito métodos de solicitação do protocolo HTTP.

Além de solicitar um determinado arquivo, envia várias informação para o servidor, sendo elas: o seu IP, a versão do navegador que está usando, que página utilizou para pedir a HTTP Request e a idioma que você usa, entre outros.

Isso quer dizer que nós podemos invocar uma mesma URL (ou URI) em uma requisição HTTP, porém, dependendo da atribuição do verbo HTTP, a requisição irá desempenhar uma tarefa diferente no servidor.

Verbos HTTP

O verbo HTTP acaba determinando a semântica – ou significado/intenção – da requisição HTTP.

Observe os verbos mais utilizados em aplicações Web.

GET – Obter os dados de um recurso.

POST – Criar um novo recurso.

PUT – Substituir os dados de um determinado recurso.

PATCH – Atualizar parcialmente um determinado recurso.

DELETE – Excluir um determinado recurso.

HEAD – Similar ao GET, mas utilizado apenas para se obter os cabeçalhos de resposta, sem os dados em si.

OPTIONS – Obter quais manipulações podem ser realizadas em um determinado recurso.

Verbos HTTP

	GET	POST	PUT	DELETE	PATCH
Básico	Consultas	Criação Edição Remoção	X	X	X
Quase lá	Consultas	Criação Edição	X	Remoção	X
RESTful-ish	Consultas	Criação	Edição	Remoção	X
RESTful-ish bônus	Consultas	Criação	Edição	Remoção	Edição parcial

POO2

Conceitos do REST API

HTTPS



HTTPS (Hyper Text Transfer Protocol Secure - protocolo de transferência de hipertexto seguro) é uma implementação do protocolo HTTP sobre uma camada adicional de segurança que utiliza o protocolo SSL/TLS.

Essa camada adicional permite que os dados sejam transmitidos por meio de uma conexão criptografada e que se verifique a autenticidade do servidor e do cliente por meio de certificados digitais.

A porta TCP usada por norma para o protocolo HTTPS é a 443.

O protocolo HTTPS é utilizado, em regra, quando se deseja evitar que a informação transmitida entre o cliente e o servidor seja visualizada por terceiros, como por exemplo no caso de compras online.

A existência na barra de endereços de um cadeado (que pode ficar do lado esquerdo ou direito, dependendo do navegador utilizado) demonstra a certificação de página segura (SSL/TLS).

A existência desse certificado indica o uso do protocolo HTTPS e que a comunicação entre o browser e o servidor se dará de forma segura. Para verificar a identidade do servidor é necessário um duplo clique no cadeado para exibição do certificado.

A ideia principal do HTTPS é criar um canal seguro sobre uma rede insegura. Isso garante uma proteção razoável de pessoas que realizam escutas ilegais (os chamados eavesdroppers) e de ataques de homem-no-meio (man-in-the-middle), dado que a cifragem foi adequadamente utilizada e que o certificado do servidor é verificável e confiável.

A confiança fornecida pelo HTTPS é baseada em autoridades de certificação que vêm pré-instaladas no navegador (isto é equivalente a dizer "Eu confio na autoridade de certificação VeriSign/Microsoft/etc. para me dizer em quem devo confiar").

Portanto, uma conexão HTTPS pode ser confiável se e somente se todos os itens a seguir são verdade:

- O usuário confia que o navegador implementa corretamente HTTPS com autoridades certificadoras pré-instaladas adequadamente;
- O usuário confia que as autoridades verificadoras só irão confiar em páginas legítimas, que não possuem nomes enganosos;
- A página acessada fornece um certificado válido, o que significa que ele foi assinado por uma autoridade de certificação confiável;

- O certificado identifica corretamente a página (por exemplo, quando o navegador acessa "https://exemplo.com", o certificado recebido é realmente de "Exemplo Inc." e não de alguma outra entidade);
- Ou o tráfego na internet é confiável, ou o usuário crê que a camada de encriptação do protocolo SSL/TLS é suficientemente segura contra escutas ilegais (eavesdropping).

HTTP vs HTTPS



HTTPS



POO2

Conceitos de REST API

REST API



REST – Conceitos

- REST é acrônimo de **REpresentational State Transfer**, e tem como objetivo primário a definição de características fundamentais para a construção de aplicações Web seguindo boas práticas.
- Por outra forma de definição, é um estilo de arquitetura de software que define um conjunto de restrições a serem usadas para a criação de web services (serviços Web).

- A sigla API corresponde às palavras em inglês “Application Programming Interface”.
- No português “Interface de Programação de Aplicações”.
- Elas são uma forma de integrar sistemas, possibilitando benefícios como a segurança dos dados, facilidade no intercâmbio entre informações com diferentes linguagens de programação e a monetização de acessos.

REST API – Conceitos

- REST API representa uma fusão de conceitos, e trata-se um conjunto de regras e convenções para criar e utilizar interfaces de programação de aplicativos.
- O objetivo de uma API RESTFul é o mesmo de um Design Pattern, o qual fornece uma solução reutilizável para problemas comuns.
- A ideia é acelerar o processo de desenvolvimento através de um paradigma de desenvolvimento bem testado e adotado.
- Neste contexto, podemos afirmar então que RESTful é o termo atribuído à uma API que possui a inteligência de aplicar o REST.

REST API – Conceitos

- Pode ser encontrada sob outros nomes, como API RESTful.
- Pode ser definida como uma interface que fornece dados em um formato padronizado baseado em requisições HTTP.
- As API's RESTful aumentam a performance para situações de concorrência, ou seja, quando muitas pessoas estão pedindo a mesma coisa ao mesmo tempo. Elas utilizam verbos para definir qual é a finalidade da requisição que está sendo enviada.

Termos utilizados

- **Recurso:** É tudo aquilo que uma API pode gerenciar e manipular dados realizando operações como cadastrar, ler, alterar, remover. Documentos, funcionários e empresas são exemplos de recursos de uma API.
- **Cliente:** É o responsável por realizar as requisições ao servidor utilizando a API.
- **Servidor:** Recebe as requisições e as processa retornando uma resposta de sucesso ou falha.

- Não se pode falar em boas práticas RESTful, sem citar o Modelo de Maturidade de Richardson.
- De acordo com Roy Fielding, um dos idealizadores do modelo arquitetural REST, para que uma API seja considerada RESTful esta deve obrigatoriamente seguir um conjunto de constraints (regras) pré-definidas pelo próprio Roy.
- Tendo em vista a complexidade desse modelo, acabamos nos vendo presos em questões como “Isso não é possível!” ou até mesmo “Esse modelo é inalcançável!”, e seguir com a implementação dessas regras acaba sendo uma tarefa difícil e dolorosa.

- Baseado nessas dores que Leonard Richardson propôs um novo modelo de maturidade dividido entre quatro níveis, visando alcançar o nível máximo REST, ou até melhor... a “glória” REST, denominado de **Modelo de Maturidade de Richardson**.

POO2

Conceitos de REST API

REST API

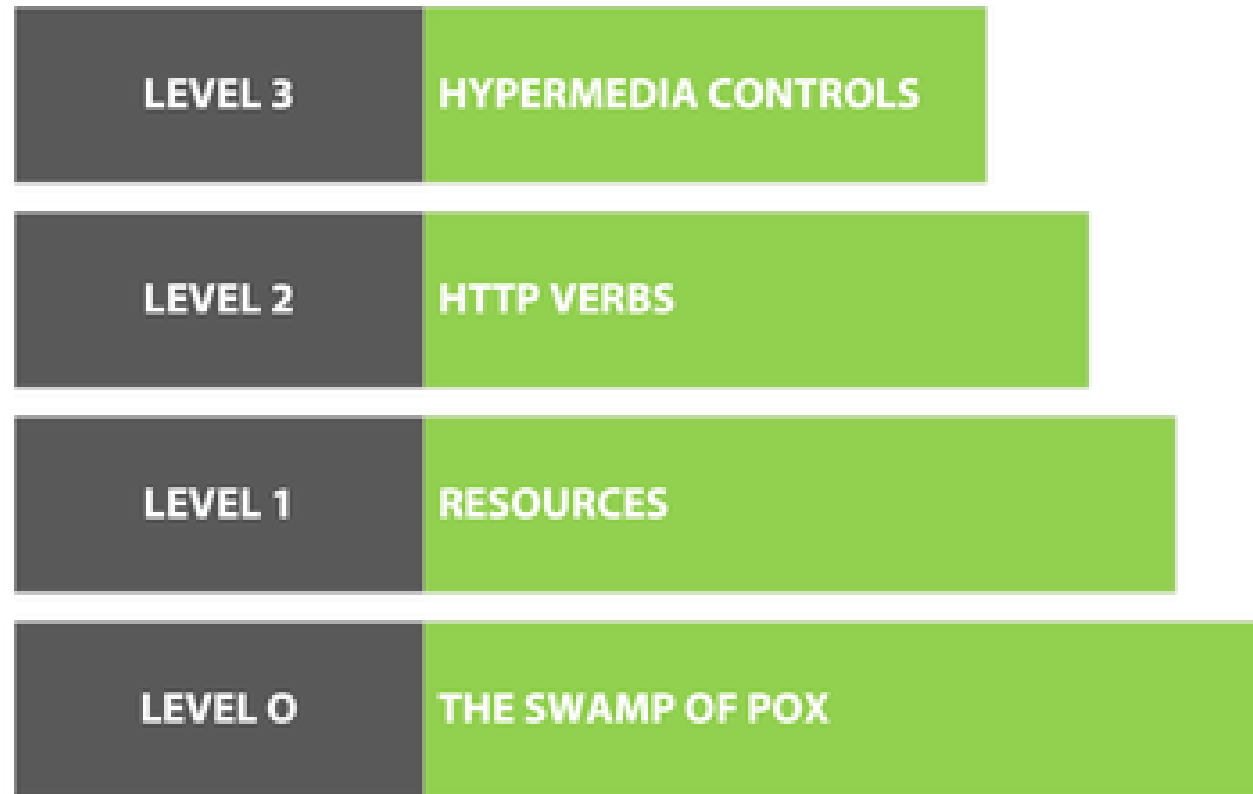
- Modelo de Maturidade de Richardson



Modelo de Maturidade de Richardson



GLORY OF REST



Modelo de Maturidade de Richardson

Organização

- Nível 0
 - Swamp of POX (também chamado de estado da bagunça).
- Nível 1
 - Identificação de Recursos
- Nível 2
 - Verbos HTTP para REST API
 - Roteamento
 - Padronização de Respostas
 - Versionamento
- Nível 3
 - HATEOAS

Modelo de Maturidade de Richardson

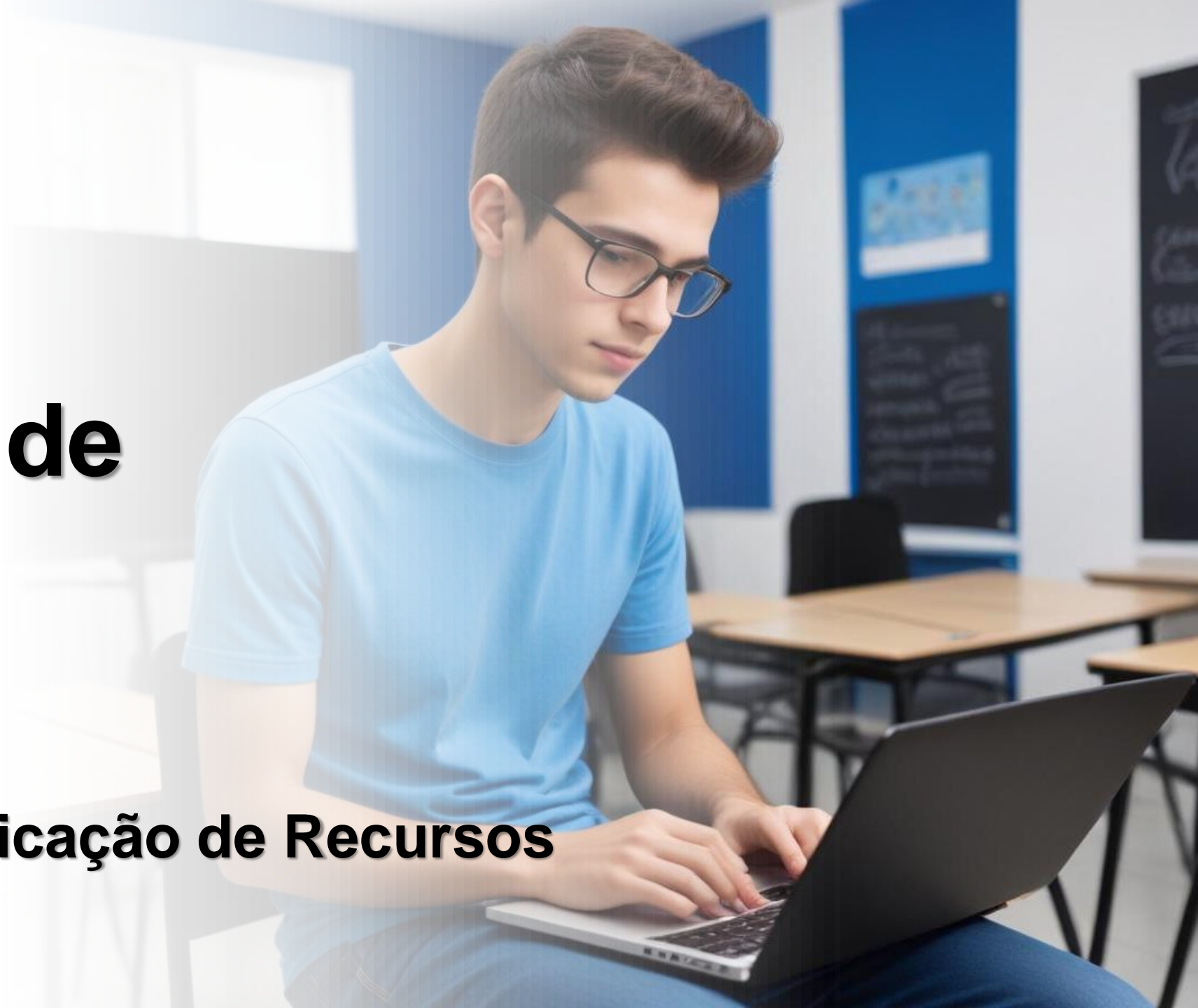
- Aqui começamos a discutir o primeiro nível de Maturidade de Richardson, o Nível 0 – POX (também chamado de estado da bagunça).
- Considerado o nível mais distante de uma API RESTful, aqui não se usa padrão para praticamente nada. Repare que o nome dos recursos (resources) não estão bem definidos, e estão sendo utilizados unicamente para invocar mecanismos remotos baseados em *Remote Procedure Invocation*.
- A troca mensagens neste nível pode até estar serializadas em formatos como XML, JSON ou texto, porém, só isso não faz uma API ser RESTful.

POO2

Conceitos de REST API

REST API

- **Nível 1 – Identificação de Recursos**



Nível 1

Neste nível, usa-se recursos (resources) como uma forma de modelar e organizar APIs. Necessariamente, não precisaremos conhecer as funcionalidade encapsulados em cada método, porém, devemos saber identificar visualmente o que cada recurso representa fazendo menção direta a um substantivo, preferencialmente no plural.

Toda aplicação gerencia algumas informações. Uma aplicação de um E-commerce, por exemplo, gerencia seus produtos, clientes, vendas, etc. Essas coisas que uma aplicação gerencia são chamadas de Recursos no modelo REST.

Um recurso nada mais é do que uma abstração sobre um determinado tipo de informação que uma aplicação gerencia, sendo que um dos princípios do REST diz que todo recurso deve possuir uma identificação única. Essa identificação serve para que a aplicação consiga diferenciar qual dos recursos deve ser manipulado em uma determinada solicitação.

Imagine a seguinte situação: Você desenvolveu um Web Service REST que gerencia seis tipos de recursos. Os clientes desse Web Service manipulam esses recursos via requisições HTTP.

Ao chegar uma requisição para o Web Service, como ele saberá qual dos recursos deve ser manipulado?

É justamente por isso que os recursos devem possuir uma identificação única, que deve ser informada nas requisições.

A identificação do recurso deve ser feita utilizando-se o conceito de URI (Uniform Resource Identifier), que é um dos padrões utilizados pela Web.

As URIs são a interface de utilização dos seus serviços e funcionam como um contrato que será utilizado pelos clientes para acessá-los.

Nível 1

São exemplos de URIs:

- <http://servicorest.com.br/produtos>
- <http://servicorest.com.br/clientes>
- <http://servicorest.com.br/clientes/57>
- <http://servicorest.com.br/vendas>

IMPORTANTE

No Nível 1 do Modelo de Maturidade, realizamos o ajuste de:

- URL (Universal Resource Locator) para
- URI (Uniform Resource Identifier).

POO2

Conceitos de REST API

Nível 2 – Verbos HTTP para REST API



Nível 2 – Verbos

No nível 2, os verbos HTTP passam a ser usados com o verdadeiro propósito pelos quais foram criados.

Os verbos mais utilizados são GET, POST, PUT e DELETE.

O verbo GET, por padrão é utilizado para requisitar dados tem um papel essencial em APIs.

Já POST, é comumente utilizado para gravar registros.

Nível 2 – Verbos

Verbos Protocolo HTTP	
Verbo	Objetivo
CONNECT	Utilizado para abrir uma comunicação bidirecional com determinado recurso
DELETE	Utilizado para deletar determinado domínio
GET	Utilizado para recuperar os dados de um determinado domínio
HEAD	Basicamente faz o mesmo que o verbo GET, porem na sua resposta só é retornado o cabeçalho da requisição
OPTIONS	Utilizado para recuperar as possíveis opções de requisições sobre um determinado domínio
PATCH / MERGE	Utilizado para atualizar parcialmente os dados de um determinado domínio
POST	Utilizado para criar dados de um determinado dominio ou realizar operações lógicas (Por exemplo um calculo)
PUT	Utilizado para atualizar dados de um determinado dominio
TRACE	Utilizado para enviar mensagens por todo o caminho entre a origem e o destino de uma requisição, provendo um grande mecanismo de debug

RECOMENDAÇÕES

- Os princípios do REST envolvem resources e cada um deve possuir uma URI única. Esses resources são manipulados usando solicitações HTTP. E cada método (GET, POST, PUT, PATCH, DELETE) tem um significado específico.
- A maioria das API's utilizam apenas os verbos GET, POST, PUT e DELETE, porém isso não é uma regra, se fizer sentido para sua aplicação, utilize-as.

POO2

Conceitos de REST API

Nível 2 – Roteamento



Nível 2 – Roteamento

O objetivo neste nível é reduzir a necessidade de documentações extensas, utilizando paths legíveis ao olho humano.

Existem diversos tipos de padrões para as rotas, os mais utilizados estão exemplificados a seguir.

Roteamento	
Path	Tipo
Plural	/api/products/
Singular	/api/product/
Camel Case	/api/institutionalProducts
Snack Case	/api/institutional_products
Spinal Case	/api/institutional-products

Nível 2 – Roteamento

O ideal é que seja definido um padrão único para ser aplicado em todas as API's da sua empresa. A recomendação é utilizar o padrão no plural.

Roteamento		
Requisição	Path	Objetivo
GET	/api/products	Consultar todos produtos
GET	/api/products/4	Consultar produto específico
POST	/api/products	Criar um produto
PUT	/api/products/7	Atualizar um produto específico
DELETE	/api/products/2012	Deletar um produto específico

Nível 2 – Roteamento

Procure utilizar verbos do Protocolo HTTP ao invés de operações em seu nome.

Roteamento		
Requisição	Path	Objetivo
GET	/api/products/consultar	Consultar todos produtos
POST	/api/products/cadastrar	Cadastra um novo produto
PUT	/api/products/4/alterar	Altera um produto


O correto é que as rotas sejam apenas substantivos que representem o domínio em si e utilizem os verbos do protocolo HTTP.

Roteamento		
Requisição	Path	Objetivo
GET	/api/products/	Consultar todos produtos
GET	/api/people/	Consultar todas as pessoas
GET	/api/campaigns/	Consultar todas as campanhas

Nível 2 – Roteamento

Evite Paths enormes, limite até no máximo 2 níveis de Paths aninhados ao principal.

Roteamento		
Requisição	Path	Objetivo
GET	/api/brands/	Consulta todas as marcas
POST	/api/brands/	Cria uma Marca
GET	/api/brands/4/models	Consulta todos os modelos de uma marca
GET	/api/brands/4/models/30	Consulta um modelo específico de uma marca
POST	/api/brands/4/models/30/versions	Cria uma versão para um modelo
PUT	/api/brands/4/models/30/versions/1	Atualiza uma versão



POO2

Conceitos de REST API

Nível 2 - Padronização de Respostas

Nível 2 - Padronização de Respostas

Quando o servidor recebe uma solicitação HTTP, o cliente deve saber o resultado da ação. Se houve sucesso ou falhou. Os códigos de status HTTP são vários códigos padronizados, com várias explicações em vários cenários. O servidor sempre deve retornar o código de status correto.

As categorias dos códigos HTTP são as seguintes:

- 2xx - Status de sucesso
- 3xx - Categoria de redirecionamento
- 4xx - Erro no Cliente
- 5xx - Erro no server

Nível 2 - Padronização de Respostas

Categoria 2xx - Sucesso

- 200 Ok - Padrão que representa sucesso. Resposta padrão para GET, quando há resultados.
- 201 Created - Indica que um recurso foi criado. Utilize para a resposta do POST. E também retorne o Header Location indicando o GET dessa informação.
- 204 No Content - Representa que a request foi processada com sucesso, mas não há conteúdo para ser retornado. Utilize para PUT, PATCH e DELETE. E também para situações em que o GET não retornou resultados.
- 202 Accepted - Indica que o servidor aceitou a request. Esse é um ótimo Status para processos assíncronos. Por exemplo um cenário onde o usuário faz upload de uma planilha que será processada em background.

Nível 2 - Padronização de Respostas

Categoria 3xx - Redirect

- Os desenvolvedores não conseguem visualizar cenários em que será utilizado o redirect a partir de uma API RESTFul.
- Em geral o 302 é bastante utilizado em desenvolvimento Frontend, como Razor.
- E o 301 Moved Permanently é utilizado por sysadmins, para indicar que um site foi permanentemente redirecionado para um novo endereço.

Nível 2 - Padronização de Respostas

Categoria 4xx - Client error

- Indica que o client cometeu uma falha na requisição.
- 400 Bad Request - A solicitação não foi processada, pois o servidor não entendeu o que o cliente está solicitando.
- 401 Unauthorized - Indica que o client não está autenticado e não tem autorização para acessar o recurso.
- 403 Forbidden - Indica que o Client está autenticado e a requisição é válida. Porém o client não tem permissão de acesso naquele recurso.
- 404 Not Found - indica que o recurso não foi localizado.

Nível 2 - Padronização de Respostas

Categoria 5xx - Erros no servidor

- Os status 5xx há dois que merecem destaque.
- É o erro 500, houve uma falha no servidor e não conseguiu processar a requisição.
- E o erro 503 que vai botar o pessoal de infra para correr. Indica que o serviço está indisponível.
- Status comum quando o Webserver sobrecarrega de requisições e não consegue mais processar nenhuma requisição.
- O 503 é comum em cenários de ataques DDoS.

POO2

Conceitos de REST API

Nível 2 - Versionamento



Nível 2 - Versionamento

- Quando pensar na evolução da API, o versionamento é uma das considerações mais importantes para o design.
- A partir do momento que a API entra no ambiente de produção, qualquer alteração no payload ou diretamente na assinatura pode implicar em uma quebra direta nos clientes que consomem os recursos da API.
- Versionar a API deve ser uma prática mandatória para todo release.

Nível 2 - Versionamento

API	Versionamento	Exemplo
Twilio	Data na URL	/2010-04-01/Accounts/{AccountSid}/Calls
Atlassian	URL	context/rest/upm/latest ou /context/rest/upm/2
Facebook	URL / Versão opcional	graph.facebook.com/v1.0/me
Youtube data API versioning	URL	googleapis.com/youtube/v3
Stripe	Custom Header	2017-04-06

POO2

Conceitos do REST API

Nível 3 - HATEOAS



Nível 3 - HATEOAS

Aqui começamos a discutir o Nível 3 – HATEOAS, do Modelo de Maturidade de Richardson.

Também conhecido como **Hypermedia as the Engine of Application State**, Roy Fielding descreve HATEOAS como uma das premissas necessárias para considerar um API como RESTful.

Tem como elemento principal a representação *hypermedia*, permitindo que um documento descreva seu estado atual, e o seu relacionamento com outros elementos ou futuros estados (delete, por exemplo). Entenda hypermedia como a capacidade de um recurso se relacionar com os demais em uma coleção.

Nível 3 - HATEOAS

O acrônimo HATEOAS é uma abreviação para - Hypermedia as the Engine of Application State - é uma restrição que faz parte da arquitetura REST. Para o ecossistema da plataforma .NET esse conceito está vinculado às Web APIs.

HATEOAS é um modelo desenvolvido por Leonard Richardson que traz o modelo REST para um novo nível, e cujo objetivo é ajudar os clientes que consomem o serviço REST, ou seja, nossa Web API, a navegar pelos recursos e saber o que podem fazer tornando as ações da WEB API mais fáceis de entender.

Nível 3 - HATEOAS

Observe o seguinte caso de uso a seguir.

- Imagine que você quer comprar um produto pela Web, em algum site de E-commerce.
- Você entra no site, navega pelas categorias de produtos via menu do site, encontra o seu produto e chega até a tela de detalhes dele, que possui as informações de preço, frete e o botão para realizar a compra.
- Mas nem sempre o produto está disponível em estoque, por isso o site costuma fazer um tratamento nessa situação, escondendo o botão de realizar a compra e exibindo um botão para notificação quando o produto voltar a estar disponível.

Nível 3 - HATEOAS

No exemplo anterior, o conceito de HATEOAS foi aplicado duas vezes.

- Primeiro, quando o cliente entrou no site de E-commerce, como ele fez para chegar até a página de detalhes do produto? Navegando via links.
 - Normalmente todo site ou aplicação Web possui diversas funcionalidades, sendo que a navegação entre elas costuma ser feita via links. Um cliente até pode saber a URI de determinada página ou recurso, mas se ele não souber precisamos guiá-lo de alguma maneira para que ele encontre as informações que procura.
- O segundo uso do HATEOAS ocorreu quando o cliente acessou a página de detalhes do produto.
 - Se o produto estivesse em estoque, o site apresentaria o botão de comprar, e o cliente poderia finalizar seu pedido.
 - Caso contrário, ele apenas poderia registrar-se para ser notificado.
 - Tudo isso é feito, principalmente, para garantir a consistência das informações.

Nível 3 - HATEOAS

Perceba que os links foram utilizados como mecanismo para conduzir o cliente quanto à navegação e ao estado dos recursos.

Esse é o conceito que foi chamado de HATEOAS, que nada mais é do que a utilização de Hypermedia, com o uso de links, como o motor para guiar os clientes quanto ao estado atual dos recursos, e também quanto as transições de estado que são possíveis no momento.

Nível 3 - HATEOAS

Aplicando HATEOAS

Temos uma solicitação GET, cuja resposta segue de forma comum, em formato JSON.

```
GET /accounts/12345 HTTP/1.1
Host: bank.example.com
Accept: application/vnd.acme.account+json
...
```

```
HTTP/1.1 200 OK
Content-Type: application/vnd.acme.account+json
Content-Length: ...

{
  "account": {
    "account_number": 12345,
    "balance": {
      "currency": "usd",
      "value": 100.00
    },
    "links": {
      "deposit": "/accounts/12345/deposit",
      "withdraw": "/accounts/12345/withdraw",
      "transfer": "/accounts/12345/transfer",
      "close": "/accounts/12345/close"
    }
  }
}
```

Nível 3 - HATEOAS

Aplicando HATEOAS

Temos uma solicitação GET, cuja resposta segue formatada utilizando HATEOAS. Perceba que agora, ao invés do registro, temos o caminho para o recurso, ou URI.

```
GET /accounts/12345 HTTP/1.1
Host: bank.example.com
Accept: application/vnd.acme.account+json
...
```

```
HTTP/1.1 200 OK
Content-Type: application/vnd.acme.account+json
Content-Length: ...

{
  "account": {
    "account_number": 12345,
    "balance": {
      "currency": "usd",
      "value": -25.00
    },
    "links": {
      "deposit": "/accounts/12345/deposit"
    }
  }
}
```

Nível 3 - HATEOAS

Aplicando HATEOAS

Neste outro exemplo, temos uma resposta para a solicitação `http://meuservicorest.com/produto/1`, da forma comum, e da forma HATEOAS.

```
{  
  "nome": "Caderno",  
  "descricao": "Caderno Espiral 100 folhas",  
  "preco": 5.45  
}
```

```
{  
  "nome": "Caderno",  
  "descricao": "Caderno Espiral 100 folhas",  
  "preco": 5.45,  
  "links" : [{  
    "rel" : "self"  
    "href" : "http://localhost:8080/produto/1"  
    "method": "GET"  
  }]  
}
```

Nível 3 - HATEOAS

Aplicando HATEOAS

Se estendermos o exemplo utilizando um conceito mais avançado de HATEOAS, teremos uma resposta ainda mais intuitiva.

```
{
  "nome": "Caderno",
  "descricao": "Caderno Espiral 100 folhas",
  "preco": 5.45,
  "links" : [{
    "rel" : "self"
    "href" : "http://localhost:8080/produto/1"
    "method": "GET"
  },
  "links" : [{
    "rel" : "update_produto"
    "href" : "http://localhost:8080/produto/1"
    "method": "PUT"
  },
  "links" : [{
    "rel" : "delete_produto"
    "href" : "http://localhost:8080/produto/1"
    "method": "DELETE"
  }]
}
```

Dúvidas?





Dúvidas, críticas e sugestões, entre em contato
através do e-mail do professor:

luiz.a.rodriques@cogna.com.br