**Artigo web REST**

A quantidade de dispositivos que surgem e são utilizados por consumidores tem aumentado de forma significativa, e criar sistemas que possuem uma única interface gráfica (GUI) pode não fazer tanto sentido como um tempo atrás.

As APIs RESTful nos permite desenvolver funcionalidades de forma desacoplada das nossas GUIs, o que nos deixa capazes de criar um mesmo conjunto de regras de negócios para vários dispositivos.

Além disso, a quantidade de usuários que possuem acesso à Internet é cada vez maior, e por isso precisamos criar sistemas cada vez mais escaláveis.

Pensando em algumas dessas motivações, neste post iremos discutir alguns conceitos primordiais e entender um pouco melhor sobre o que de fato é REST.

O que é REST?

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.

A Web como a conhecemos hoje, funciona seguindo práticas REST, e para entendermos melhor, vamos considerar o exemplo abaixo:

1. Você entra com um endereço em seu navegador (Chrome, Firefox, Edge, etc).

2. Seu navegador estabelece uma conexão TCP/IP com o servidor de destino e envia uma requisição GET HTTP com o endereço que você digitou.

3. O servidor interpreta sua requisição e de acordo com o que foi solicitado, uma resposta HTTP é retornada ao seu navegador.

4. A resposta retornada pode ser de sucesso, contendo alguma representação em formato HTML, ou pode ser algum erro, como por exemplo o famoso 404 Not Found, que indica que o endereço/recurso que você solicitou não pôde ser encontrado.

5. Em caso de sucesso, o seu navegador interpreta o HTML e você consegue navegar pela página renderizada.

Todo esse processo é repetido enquanto você está navegando em alguma página Web. Cada link que você clica ou formulário que submete, efetua os passos que discutimos acima.

Se analisarmos detalhadamente o que discutimos, podemos extrair alguns elementos principais; esses elementos são responsáveis por permitirmos criar aplicações Web da forma que conhecemos hoje, e são esses elementos que vamos detalhar melhor abaixo.

Recursos

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

Se analisarmos o primeiro passo da lista anterior, podemos relembrar que falamos de um endereço. Esse endereço possui basicamente uma parte que nos permite endereçar algo, e é o que chamamos de URL (*Universal Resource Locator*), um exemplo seria http://www.algaworks.com.

A partir desse endereço, estamos aptos a acessar algum recurso, que poderia ser por exemplo, “cursos” ou “alunos”.

Dito isso, podemos pensar que ao acessarmos por exemplo, o endereço http://www.algaworks.com/cursos, nos é retornado uma representação HTML do recurso “cursos”.

Da mesma forma, se acessarmos por exemplo, http://www.algaworks.com/alunos, podemos pensar que esse recurso não existe e que uma página indicando a resposta 404 nos será mostrada.

Interfaces Uniforme

Para interagirmos com os recursos que aprendemos no tópico anterior, o HTTP nos fornece uma interface de operações padronizadas, permitindo que possamos criar, atualizar, pesquisar, remover e executar operações sob um determinado recurso.

Além de operações padronizadas, o HTTP fornece um conjunto de respostas para que os clientes (navegadores, APIs, etc) possam saber, de forma adequada, como agir perante uma determinada resposta.

Abaixo, veja uma breve descrição dos principais métodos e respostas.

GET

O método GET é utilizado quando existe a necessidade de se obter um recurso. Ao executar o método GET sob um recurso, uma representação será devolvida pelo servidor.

Em aplicações Web, normalmente é retornado uma representação HTML.

POST

Utilizamos o método POST quando desejamos criar algum recurso no servidor a partir de uma determinada representação. Exemplo disso é quando fazemos a submissão de algum formulário em uma página Web.

PUT

Semelhante ao método POST, a ideia básica do método PUT é permitir a atualização de um recuso no servidor.

DELETE

Como você já deve estar imaginando, o método DELETE é utilizado com o intuito de remover um recurso em um determinado servidor.

Respostas

Baseado nos métodos que discutimos, o servidor deve processar cada uma das requisições e retornar uma resposta adequada. Veja um resumo de cada uma dessas respostas.

1XX – Informações Gerais  
2XX – Sucesso  
3XX – Redirecionamento  
4XX – Erro no cliente  
5XX – Erro no servidor

Para cada tipo que você pode ver, existe uma série de respostas relacionadas. Por exemplo, se o servidor retornar um “200 OK”, significa que o recurso pedido foi retornado com sucesso.

Por outro lado, se o servidor retornar um “404 Not Found”, significa que o recurso que estamos pedindo não foi encontrado.

Apesar de muitas aplicações Web não seguirem o uso adequado de métodos e respostas, devemos sempre que possível, construir nossas aplicações utilizando-as da maneira mais adequada possível.

Representações

Em aplicações Web, a representação mais utilizada é o HTML. Essa representação é utilizada como forma de encapsular as informações relacionadas a um determinado recurso.

Além do HTML, podemos utilizar XML, JSON, ou algum outro formato que melhor atenda o cenário que estamos desenvolvendo.

É importante citar que um recurso pode ter mais de uma representação, ou seja, podemos construir aplicações que para determinados cenários, retornem representações diferentes, baseado nas necessidades de cada cliente.

Hypermedia

*Hypermedia* é um conceitos importante em aplicações que seguem o modelo REST à risca. É essa característica que permite criarmos aplicações que possam evoluir de formas tão surpreendentes.

Uma representação *hypermedia* trabalha basicamente como um motor de estado, permitindo que clientes naveguem nos mesmos. Cada estado é um documento (uma página HTML) e possui referências para futuros estados (a partir de links).

|  |  |
| --- | --- |
| 1  2  3 | <a href="produtos.html">Produtos</a>  <a href="clientes.html">Clientes</a>  <a href="contato.html">Contato</a> |

Se repararmos no código acima, podemos verificar que temos uma representação HTML que nos permite ir para as próximas páginas a partir do uso de links.

O seu navegar sabe interpretar o significado da tag <a> e criar uma requisição correta para o servidor.

Conclusão

Neste artigo discutimos uma série de características de uma aplicação REST. Não foi nosso objetivo discutir como podemos criar **RESTful** APIs.

Autor:

[](https://blog.algaworks.com/author/emilio-dias/)

[Emílio Dias](https://blog.algaworks.com/author/emilio-dias/)

Formado em Ciência da Computação, Especialista em Segurança da Informação e detentor das certificações SCJP e LPIC-1. Palestrante de fóruns internacionais de Software Livre e congressos de Engenharia de Software. Atua também como Professor nos cursos de Ciência da Computação e Sistemas de Informação da Unitri em Uberlândia.

**Fonte:**

Site da interner: <https://blog.algaworks.com/4-conceitos-sobre-rest-que-qualquer-desenvolvedor-precisa-conhecer/#:~:text=REST%20%C3%A9%20acr%C3%B4nimo%20de%20Representational,1.> Acessado em 06/08/2020.