**Resumo de API REST**

* O que é uma API REST ?

que é uma interface que vai fornecer dados e recursos baseados em requisições e respostas HTTP;

Vamos começar falando um pouco sobre o back-end e o front-end, o que são esses conceitos que são tão falados. Vamos supor que quando você está no seu navegador, você está digitando um endereço, por exemplo, o endereço da Alura, você está trabalhando como se você fosse um cliente, você está digitando o endereço e em algum outro computador do outro lado do mundo da nuvem aonde quer que esteja, um servidor vai processar essas informações, esses dados e vamos devolver para que o seu navegador consiga exibir a página que você precisa, um conteúdo que está proposto.

[01:23] Esse processo acontece no servidor de buscar as informações e os clientes, a comunicação entre eles ocorrem através do protocolo HTTP, que também faz parte da formação, o curso de HTTP para ficar bem claro como acontece essa comunicação e quais são as características do protocolo.

[01:48] Toda a parte dos clientes, de você no seu navegador buscando as informações, o que está acontecendo no seu navegador você está fazendo uma requisição e o servidor está devolvendo uma resposta. Basicamente é assim, vamos trabalhar com requisições e respostas.

[02:10] Nós consideramos que tudo o que está acontecendo no servidor é responsabilidade do back-end, filtrar dados, processar as informações, ver as regras de negócio e o que é exibido ali na sua interface gráfica, o que você interage, os menus, os botões é o que consideramos aplicação front-end.

[02:33] É um resumo bem genérico, só para ficar mais claro o que vamos fazer e vamos falar de API. API é a sigla que significa Application Programming Interface, que seria Interface de Programação de Aplicação. Essas siglas sempre deixam a pessoa um pouco confusa.

[02:53] O que basicamente significa isso, vamos pensar nisso de uma forma bem simples. Você tem uma cafeteria e você senta na mesa, você e outras pessoas, você é o cliente e aí você vai chamar a pessoa, a garçonete e vai pedir o que você quer. "Eu quero um Frapuccino". A garçonete vai à cozinha, vai falar para o barista que a mesa 1 quer um Frapuccino e o barista vai preparar o pedido e quando estiver pronto ele vai avisar que está pronto, que conseguiu atender o que foi pedido.

[03:35] E vai passar de novo para a garçonete que vai devolver para a mesa 1, e aí o cliente vai gostar, vai agradecer e elogiar o pedido. Assim, de forma bem básica, temos três entidades, alguém está fazendo uma requisição, pediu de um jeito que a garçonete entendeu, não importava se o cliente era americano, japonês, mulher, homem, idoso, criança, falando o pedido da forma certa a garçonete entendeu, solicitou o pedido, o pedido foi preparado e ela devolveu isso como uma resposta.

[04:21] Quando estamos falando de API é basicamente a mesma coisa, você tem um sistema que pode ser Web ou mobile que vai fazer uma requisição, à API vai entender aquela requisição, vai processar os dados , coletar as informações, os dados, as funcionalidades, seja o que for que ela precise devolver e vai devolver como uma resposta para quem solicitou.

[04:48] O tempo inteiro vamos está trabalhando com requisições e respostas. E essas respostas geralmente vão acontecer em alguns formatos, os formatos mais comuns são JSON e XML. Para as APIs REST o formato mais utilizado hoje em dia é o JSON, nas antigas APIs SOAP que ainda tem, mas são menos utilizadas, trafegava muito XML.

[05:16] Já que falamos em **API REST**, o REST é um modelo de arquitetura que vai definir como que os sistemas distribuídos que tem essa arquitetura cliente/servidor que comentei exemplificando com a garçonete e o cliente sendo a API e o sistema, como que esses sistemas distribuídos vão se comunicar. O REST é um acrônimo para Representational State Transfer, que seria Transferência de Estado Representacional, outro nome complicado.

[05:57] Vamos falar um pouco sobre alguns aspectos, o primeiro que podemos citar é que tudo vai ter um conceito de recurso. Quando temos uma API ou uma aplicação de forma geral, estamos gerenciando recursos. Um e-commerce vai gerenciar produtos, fornecedores, clientes, forma de pagamento. No caso vamos fazer uma livraria, vamos estar trabalhando com recursos onde cada recurso vai ter uma URI, eu vou ter livros, autores, editoras. Esse é o primeiro aspecto do REST.

[06:37] O REST foi criado pelo mesmo criador do protocolo HTTP, por isso que ele é tão utilizado e simplificado, as APIs construídas com esse modelo são mais simples de utilizar. Vamos trabalhar o tempo inteiro com recursos e cada recurso vai ter a sua URI, o seu identificador único, a forma com que vamos acessar os recursos.

[07:02] Como estamos trabalhando com recursos, no caso os livros, os autores e as editores, é importante, uma base do REST, é que essa comunicação entre quem faz a requisição e quem está devolvendo a resposta seja stateless, não guardamos o estado. Quando eu devolvo uma resposta, por exemplo, de um recurso, um livro com aquele valor naquele momento, aquela representação é sem estado, é aquela representação do momento. Se dois segundos depois alguém alterar o valor do livro no servidor a representação que veio para você continua, não guarda estado.

[07:50] A comunicação é stateless para que a internet aguente as milhões de requisições e respostas o tempo inteiro, como acontece, uma das bases do REST é ter comunicação stateless.

[08:04] Outros aspectos que podemos citar é que ela usa do próprio protocolo HTTP para manipular esses recursos. Porque o que vamos precisar fazer é cadastrar um livro, vamos precisar obter uma lista de livros, atualizar, excluir. Eu vou usar o próprio protocolo HTTP e os seus verbos para fazer essa manipulação, vou usar aquela mesma URI, que é o identificador único do recurso, e só vou trocar o verbo, vou usar o: GET, POST, PUT, DELETE, o que quer que eu queira fazer.

[08:41] Essas características do REST também simplificam a comunicação e como os sistemas vão conversar. O formato de representação do recurso também, como eu citei antes, basicamente poderia ser em qualquer formato, o REST é aberto diferente do SOAP que basicamente é XML obrigatoriamente. No REST não, pode ser um HTML, pode ser o próprio XML, pode ser JSON que é o mais comum. Você vai representar aquele recurso e trafegá-lo nas suas requisições e respostas.

[09:22] Essas são algumas das poucas características que vamos trabalhar ao longo do curso e eu vou reforçar. E agora vamos começar, vamos fazer a API para que alguém faça uma requisição para processamos os dados e devolvemos o que precisamos devolver.

O termo API (Application Programming Interface ou, em português, Interface de Programação de Aplicações) é amplamente utilizado quando se trata de desenvolvimento de software. É muito importante que você conheça esses principais termos, entenda-os melhor, para que possa sempre estar antenado às melhorias e boas práticas que são utilizados na comunicação dos sistemas distribuídos.

**REST: Conceito e fundamentos**



**Guilherme Lima**

22/09/2020

**COMPARTILHE**

REST(*Representational State Transfer*, que significa Transferência Representacional de Estado) **é um modelo de arquitetura e não uma linguagem ou tecnologia de programação, que fornece diretrizes para que os sistemas distribuídos se comuniquem diretamente** usando os princípios e protocolos existentes da Web sem a necessidade de [SOAP](https://pt.wikipedia.org/wiki/SOAP) ou outro protocolo sofisticado.

**A origem do termo REST**

Os conceitos do **REST** foram submetidos à tese de doutorado de [**Roy Fielding**](https://en.wikipedia.org/wiki/Roy_Fielding) nos anos 2000, onde o princípio fundamental é usar o [protocolo HTTP](https://www.alura.com.br/artigos/desmistificando-o-protocolo-http-parte-1) para comunicação de dados.

A arquitetura REST é simples e fornece acesso aos recursos para que o cliente REST acesse e renderize os recursos no lado do cliente. No estilo REST, URI ou IDs globais ajudam a identificar cada recurso.

Esta arquitetura usa várias representações de recursos para representar seu tipo, como **XML**, **JSON**, **Texto**, **Imagens** e assim por diante.

Vale ressaltar que o REST não se limita a solicitações e respostas de registros. Também é possível inserir um novo registro ou deletar um já existente.

**Responsabilidades no REST**

Existe no REST um princípio chamado STATELESSNESS (sem estado), onde o servidor não precisa saber em qual estado o cliente está e vice-versa. Mas o que é um servidor e um cliente?

**Cliente:** é o componente solicitante de um serviço e envia solicitações para vários tipos de serviços ao servidor.

**Servidor:** É o componente que é o provedor de serviços e fornece continuamente serviços ao cliente conforme as solicitações

Nesta arquitetura ou modelo, cliente-servidor ajuda na separação de responsabilidades entre a interface do usuário e o armazenamento de dados. Ou seja, quando uma solicitação REST é realizada, o servidor envia uma representação dos estados que foram requeridos.

Não há limite superior no número de clientes que podem ser atendidos por um único servidor. Também não é obrigatório que o cliente e o servidor residam em sistemas separados.

A comunicação entre cliente e servidor ocorre através da troca de mensagens usando um padrão de solicitação-resposta. O cliente basicamente envia uma solicitação de serviço e o servidor retorna uma resposta.

**Requisições e comunicações**

O REST precisa que um cliente faça uma requisição para o servidor para enviar ou modificar dados. Um requisição consiste em:

* **Um verbo ou método HTTP**, que define que tipo de operação o servidor vai realizar;
* **Um header**, com o cabeçalho da requisição que passa informações sobre a requisição;
* **Um path (caminho ou rota)** para o servidor, como por exemplo <https://www.alura.com.br/artigos/golang-trabalhando-com-datas>;
* **Informação no corpo da requisição**, sendo esta informação opcional.

**Métodos HTTP**

Em aplicação REST, os métodos mais utilizados são:

* O **método GET** é o método mais comum, geralmente é usado para solicitar que um servidor envie um recurso;
* O **método POST** foi projetado para enviar dados de entrada para o servidor. Na prática, é frequentemente usado para suportar formulários HTML;
* O **método PUT** edita e atualiza documentos em um servidor;
* O **método DELETE** que como o próprio nome já diz, deleta certo dado ou coleção do servidor.

[Neste link](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods) você encontrará a lista completa de todos os métodos.

**Códigos de Respostas**

Cada resposta que a aplicação REST retorna, é enviado um código definindo o status da requisição. Por exemplo:

* 200 (OK), requisição atendida com sucesso;
* 201 (CREATED), objeto ou recurso criado com sucesso;
* 204 (NO CONTENT), objeto ou recurso deletado com sucesso;
* 400 (BAD REQUEST), ocorreu algum erro na requisição (podem existir inumeras causas);
* 404 (NOT FOUND), rota ou coleção não encontrada;
* 500 (INTERNAL SERVER ERROR), ocorreu algum erro no servidor.

Estes são os principais, porém [neste link](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status) você encontrará a lista completa do código de cada requisição.

**Conclusão**

Neste artigo, apresentamos a origem, conceitos e os principais fundamentos do REST. [**Os cursos de programação da Alura**](https://www.alura.com.br/cursos-online-programacao) podem ajudar você a começar, aprender e se desenvolver em tecnologia e, além disso, depois você pode se aprofundar e utilizar essa arquitetura no gerenciamento de estados do [React](https://www.alura.com.br/artigos/react-js), por exemplo.