Artigo

HTTP: Verbos

A ideia geral é a seguinte: seu serviço vai prover uma url base e os verbos HTTP vão indicar qual ação está sendo requisitada pelo consumidor do serviço.

A ideia geral é a seguinte: seu serviço vai prover uma url base e os **verbos HTTP** vão indicar qual ação está sendo requisitada pelo consumidor do serviço.

Guia do artigo:

Por exemplo, considerando a URL *www.dominio.com/rest/notas/*, se enviarmos para ela uma requisição HTTP utilizando o verbo GET, provavelmente obteremos como resultado uma listagem de registros (notas, nesse caso). Por outro lado, se utilizarmos o verbo POST, provalmente estaremos tentando adicionar um novo registro, cujos dados serão enviados no corpo da requisição.

Da mesma forma, a URL *www.dominio.com/rest/notas/1*, por exemplo, poderia ser usada para diferentes finalidades, dependendo do verbo enviado na requisição. No caso do GET, essa URL provavelmente deveria nos retornar o registro de ID 1 (nesse caso, a nota de ID = 1). Já o verbo DELETE indicaria que desejamos remover esse registro.

Repare que a URL se mantém – o verbo indica o que estamos fazendo de fato. Por exemplo, não precisamos disponibilizar no serviço uma URL como `/notas/listar` ou `/notas/remover/1`.

Exemplo prático

Os exemplos a seguir foram feitos utilizando a extensão Postman, do Google Chrome, para comunicar com a API http://devmedianotesapi.azurewebsites.net, criada no curso Criando serviços RESTful em .NET. Contudo, os conceitos apresentados aqui podem ser aplicados a outras tecnologias, uma vez que REST é um padrão e independe de linguagem.

Verbo GET

Sem passagem de ID: vai retornar todas as notas (ou as notas mais recentes, isso cabe a regra de negócio da aplicação), como mostra a **Figura 1**.

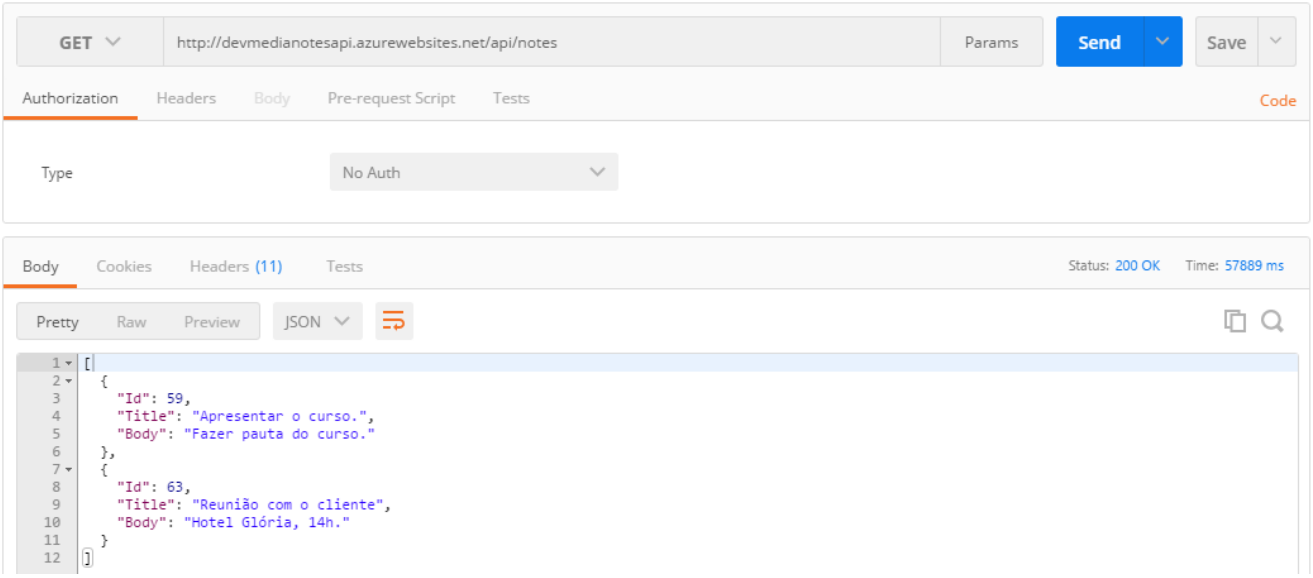

Figura 1. Requisição GET sem parâmetros

Com passagem de ID: vai retornar a nota com ID especificado, como mostra a **Figura 2**.

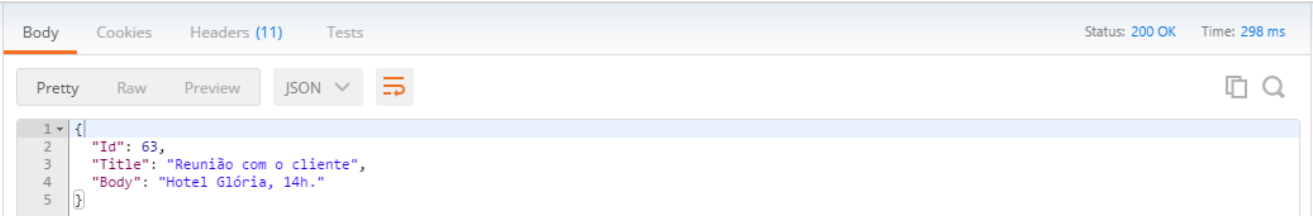

Figura 2, Requisição GET com parâmetros

Verbo POST

Normalmente usado sem passagem de parâmetro – usado para criar uma nova nota, como mostra a **Figura 3**.

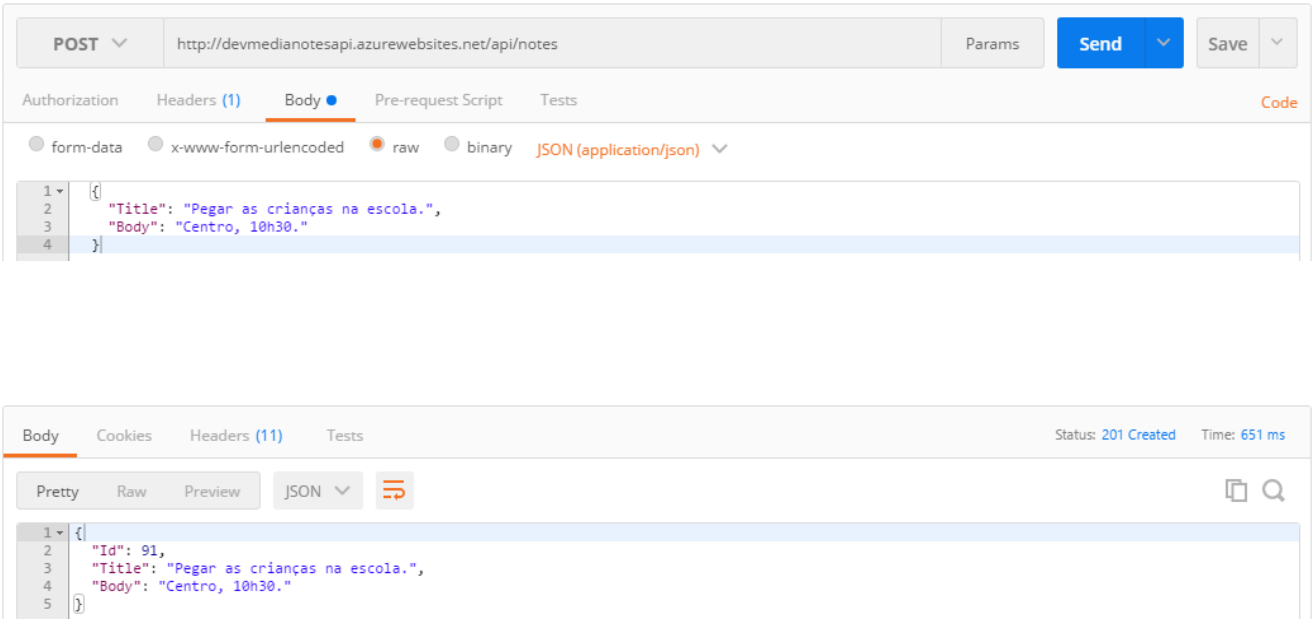

Figura 3, Requisição POST

Verbo DELETE

Usado para remover o recurso (por exemplo uma nota): utilize com passagem de ID, como mostra a **Figura 4**.

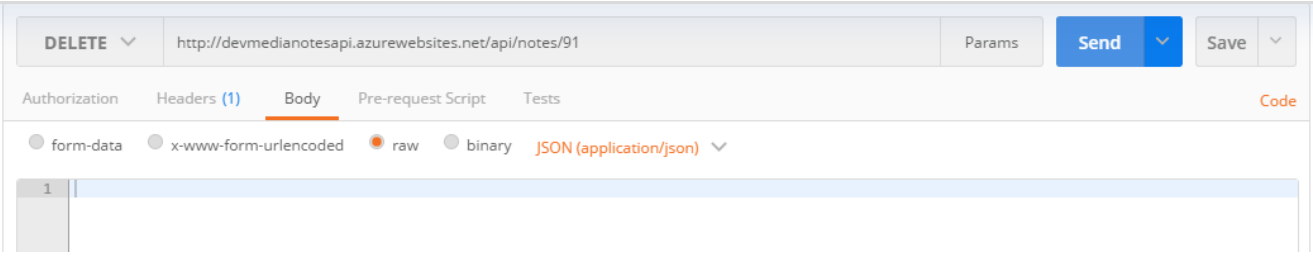

Figura 4. Requisição DELETE

Verbo PUT

Normalmente usado com parâmetro: Use para editar o recurso – neste exemplo, uma nota, como mostra a **Figura 5**.

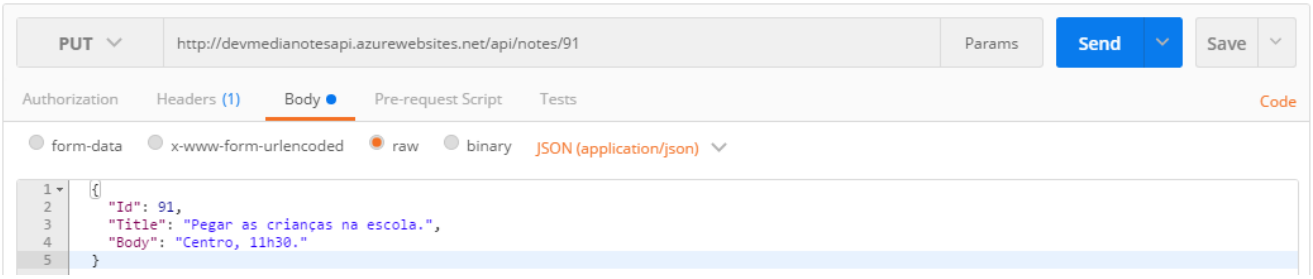

Figura 5, Requisição PUT

> **Nota**: A literatura indica que o verbo PUT deve passar todos os dados do recurso preenchidos, independente de quais dados você de fato editou. Por exemplo, digamos que sua classe nota possui os atributos titulo e descrição – e você editou apenas o título. A documentação indica que você deve passar ambos os atributos preenchidos para o serviço (mesmo só tendo editado o título).

Link de referência

Para resolver essa questão de forma elegante a comunidade adotou, por

Verbo PATCH

Usado para editar o recurso sem a necessidade de enviar todos os atributos – o consumidor envia apenas aquilo que de fato foi alterado (mais o ID como parâmetro, para que o serviço saiba o que vai ser alterado).

Padrões de resposta do serviço

A documentação indica que o serviço pode retornar o resultado em diversos formatos – JSON, XML, texto plano, etc. Contudo, atualmente o formato mais adotado tem sido o JSON, por seu formato leve, legível e sua fácil interpretação por diversas tecnologias.

Além disso, o protocolo HTTP dispõe de diveros códigos (ou status) que devem ser incluídos na resposta, indicando o resultado do processamento.

Os códigos iniciados em "2" indicam que a operação foi bem sucedida. Nessa categoria temos, por exemplo, código 200 (OK), iniciando que o método foi executado com sucesso; 201 (Created) quando um novo recurso foi criado no servidor; e 204 (No Content) quando a requisição foi bem sucedida, mas o servidor não precisa retornar nenhum conteúdo para o cliente

Já os códigos iniciados em "4" indicam algum erro que provavelmente partiu do cliente. Por exemplo, o código 400 (Bad Request) indica que a requisição não pôde ser compreendida pelo servidor, enquanto o 404 (Not Found) indica que o recurso não foi localizado.

Há, ainda, os códigos que indicam erro do lado do servidor. Nesse caso, eles iniciam com "5", como o 500 (Internal Server Error), que indica que ocorreu um erro internamente no servidor que o impediu de processar e responder

Pra conhecer todos os status do HTTP, você pode consultar a especificação do protocolo.

Referência rápida para APIs

Na **Tabela 1** a seguir temos cada ação em CRUD que pode ser implementada por um servidor e os códigos de retorno associados a elas. Use como fonte de consulta em suas próprias APIs.

Verbo HTTP	Ação	Geral (p.ex. /notas)	Específico (p.ex /notas/{id})
POST	Criar	201 Created	404 Not Found 409 Conflict
GET	Ler	200 OK	200 OK 404 Not Found
PUT	Atualizar	405 Method Not Allowed	200 OK 204 No Content 404 Not Found,
PATCH	Modificar parcialmente	405 Method Not Allowed	200 OK 204 No Content. 404 Not Found
DELETE	Excluir	405 Method Not Allowed	200 OK 404 Not Found

Tabela 1. Ação em CRUD

Diferenças entre PUT e POST

Encontramos na literatura indicações de que apenas três verbos são suficientes para um CRUD completo: GET, DELETE e PUT – sendo o PUT utilizado para criar ou editar um recurso.

Interpretando a documentação, temos o seguinte: PUT é usado para criar ou editar um recurso, enquanto POST pode ser utilizado para qualquer coisa, cabendo ao back-end a definição dessa semântica.

O mundo não-acadêmico, no entanto, adotou por convenção o uso do POST para incluir e do PUT para alterar – e em situações de programação mais elegantes, também o uso de PATCH para editar parcialmente.

Confira também

Curso de Front-End

Curso

Anotar Marcar como concluído

Por **Devmedia**

Em 2020