Trabalho Prático - 05

🎓 **Aluno**: Lucas de Araújo
Matrícula: 18.2.4049

📌 Vídeo: https://drive.google.com/file/d/1Z_fBELBXoYnisomwdSD-o23q8eAEC2zk/view?usp=sharing

---

O que é REST?
Verbos HTTP
    Verbo GET
        Exemplo
    Verbo POST
        Exemplo
    Verbo Delete
        Exemplo
    Verbo PUT
        Exemplo
    Verbo HEAD
        Exemplo
    Verbo CONNECT
        Exemplos
OAS3.0
    OpenAPI
    OpenAPI 3.0
Bibliografia

O que é REST?

Antes explicar o que são GET, HEAD, POST, PUT, DELETE e CONNECT, é necessário entender o que é o REST.

O REST é um conjunto de regras, ou melhor, restrições, de arquitetura para a criação de uma API. Durante o desenvolvimento da mesma, os desenvolvedores podem implementar esta arquitetura de diferentes maneiras, porém, existem critérios importantes para que a API seja considerada RESTful (Complementa REST)

- A arquitetura de cliente e servidor precisa ser formada por clientes, recursos e servidores, com solicitações gerenciados por meio de HTTP (Aqui vai entrar as solicitações anteriormente mencionadas)
- Estabelecer uma comunicação *stateless* (nenhuma referência ou informação sobre transações antigas são armazenadas e cada nova transação é feita do zero) entre o cliente e servidor. Isso significa que nenhuma informação será armazenada entre solicitações GET e todas as solicitações serão separadas e desconectadas
- Armazenar dados em cache para otimizar interações
- Ter uma interface uniforme entre os componentes para que as informações sejam transferidas em um formato padronizado
- Ter um sistemas em camadas que organiza os tipos de servidores envolvidos na recuperação de informações solicitadas em hierarquias que o cliente não pode ver.

Vale dizer também que, durante a entrega de informações do servidor ao cliente, é utilizado comumente o JSON (anteriormente o XML) pelo seu fácil entendimento para nós humanos na visualização e pela fácil manipulação através da máquina

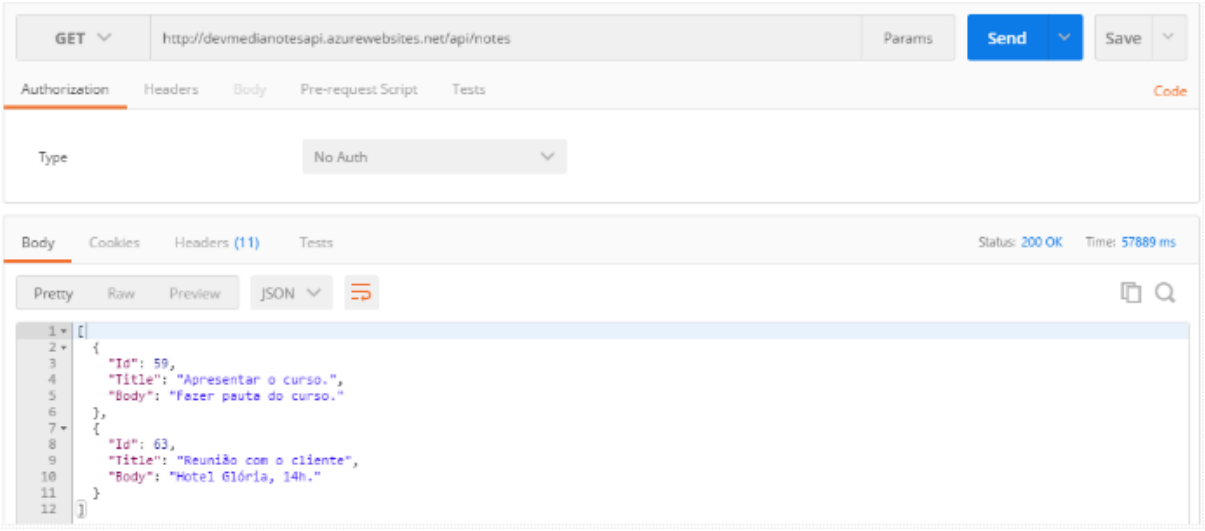
Verbos HTTP

Introduzido este conceito de REST, podemos agora explicar os verbos HTTP mencionados no começo deste PDF

Verbo GET

O verbo GET tem como objetivo retornar algo ao cliente, ou seja "pegar" (get) algo através da requisição. Essa requisição pode ser feita utilizando zero ou mais parâmetros na URL.

Exemplo

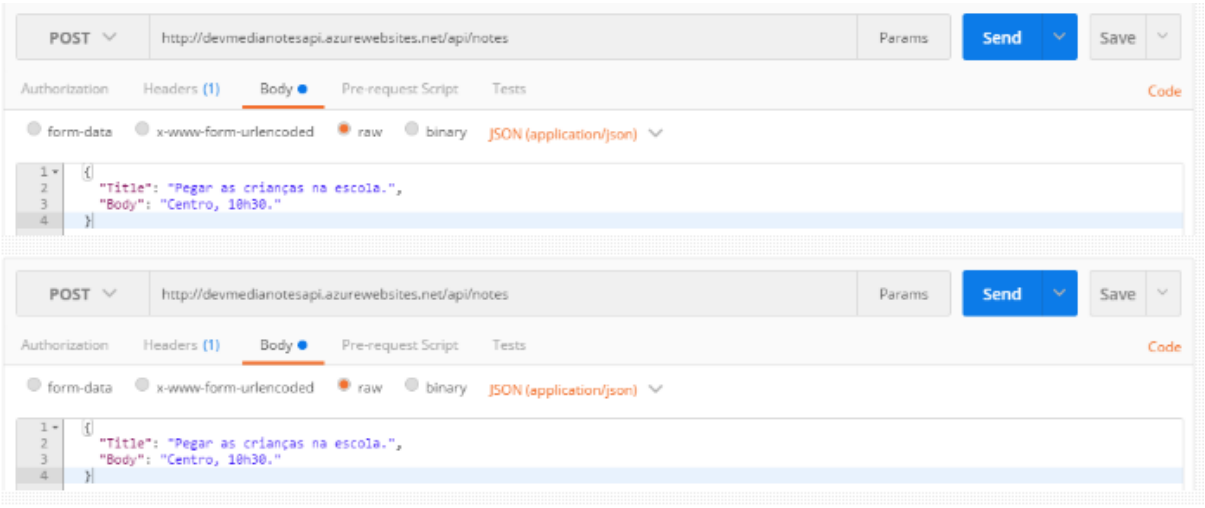


Realizadoo GET para a url `https://deviamediasnotesapi.azurewebsite.net` na rota `/api/notes` foi retornado a lista de tarefas em formato de JSON.

Verbo POST

O verbo POST tem como objetivo entregar algo ao ser servidor por parte do cliente, ou seja, "publicar" (post) algo novo no servidor. Nessa requisição é enviado algo no corpo da mesma em formato JSON e o servidor a recebe e realiza uma ação com a mesma

Exemplo



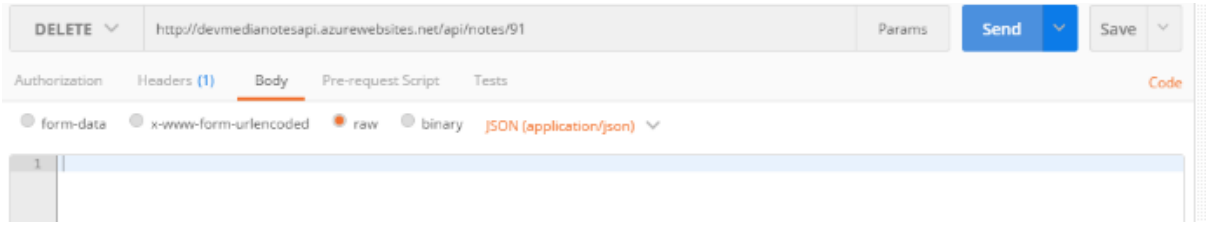
Realizadoo POST para a url `https://deviamediasnotesapi.azurewebsite.net` na rota `/api/notes` e enviando um JSON com um título e corpo, foi criado uma nova tarefa.

Atenção que as rotas são a mesma, porém mudando o verbo (GET → POST) a função exercida pelo servidor é totalmente diferente.

Verbo Delete

Como o nome sugere, ao contrario do POST, será usada para deletar algo no servidor. Para identificar o que será deletado, geralmente passamos um parâmetro de identificação único (ID)

Exemplo

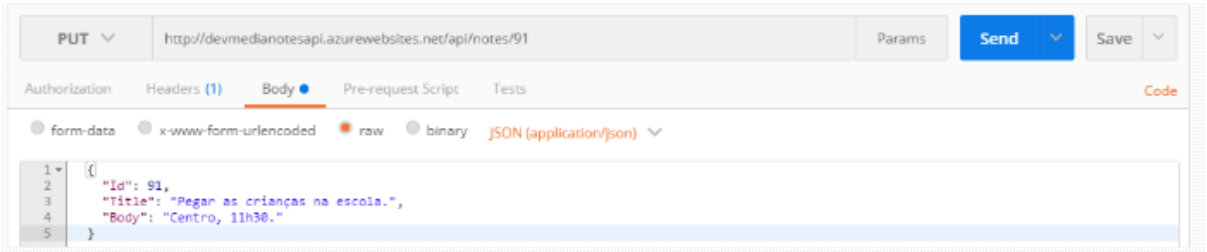


Realizado DELETE para a url `https://deviamediasnotesapi.azurewebsite.net` na rota `/api/notes/91` foi deletada a tarefa de número 91.

Verbo PUT

Tem como principal uso, editar um recurso já existente no servidor. Porém, diferente de outro verbo, o PATCH, no PUT é necessário sempre enviar no corpo da requisição todos os dados da mesma que você deseja editar, mesmo que você esteja editando somente um campo do seu determinado objeto

Exemplo



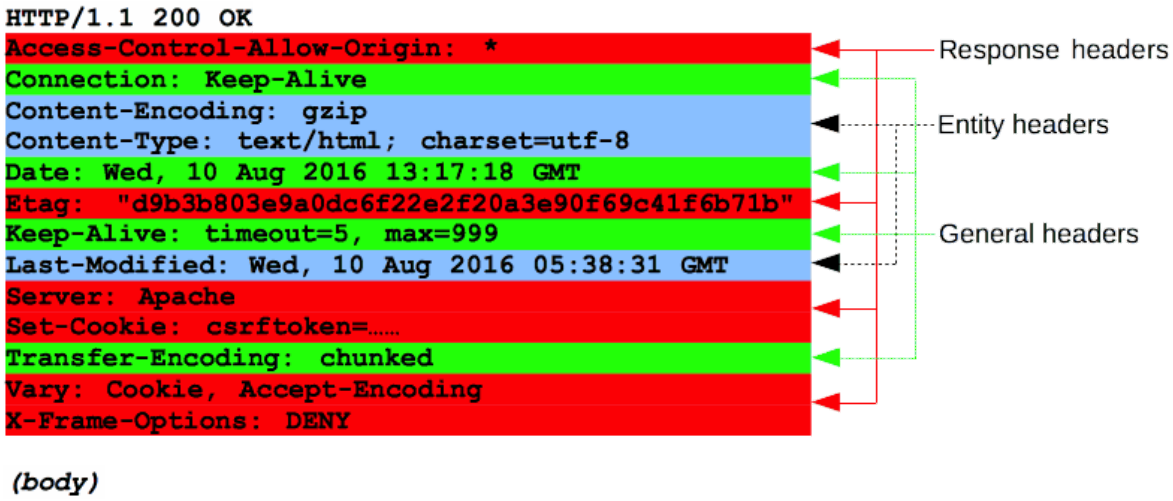
Realizado PUT para a url `https://deviamediasnotesapi.azurewebsite.net` na rota `/api/notes/91` e enviando no corpo um JSON com a tarefa modificada, o servidor irá modificar a mesma salva nele.

Verbo HEAD

O verbo HEAD é muito parecido com o verbo GET previamente mencionado aqui, porém com a diferença de que: No HEAD, não é retornado o corpo da mensagem, apenas as

informações contidas no header do HTTP.
Esse método tem como objetivo obter metainformações sobre a entidade entre os requests porém sem todo esforço de mandar o corpo da requisição, ou seja, agilizando o processo. Além disso, por se tratarem de informações relativamente estáticas (Versão do navegador, sistema operacional etc) é possível realizar o cache dessas informações no servidor a fim de evitar o gasto de tempo entre as requisições

Exemplo



Verbo CONNECT

Este verbo tem como utilidade iniciar uma comunicação bidirecional (abrir um túnel) com o recurso solicitado.

Por exemplo, este método pode ser utilizado para acessar websites que utilizam o protocolo SSL. O cliente solicita a um servidor proxy HTTP que tunelize a conexão TCP para o destino desejado. O servidor então procede para fazer a conexão em nome do cliente. Uma vez que a conexão foi estabelecida pelo servidor, o servidor Proxy continua a proxy do fluxo

Exemplos

> CONNECT www.exemplo.com:443 HTTP/1.1

> CONNECT server.exemplo.com:80 HTTP/1.1
> Host: server.exemplo.com:80
> Proxy-Authorization: basic aGVsbG86d29ybGQ=

OAS3.0

OpenAPI

Conhecida como especificação OpenAPI (OAS), é um definição para a descrição de APIs que permite tanto nós humanos como os computadores a descobrir e entender as funções de um serviço sem requerer acesso ao código fonte, documentação, entre outros.

OpenAPI 3.0

Conhecida como a OpenAPI3.0 (OAS3.0), é a versão mais recente do serviço da OpenAPI que trouxe algumas novidades, sendo elas:

- Maior Reusabilidade
- Mudança de parâmetros
- Suporte para descrição de callbacks
- Links para expressar o relacionamento entre operações da API
- Exemplos melhorados
- Aumento de segurança

Bibliografia

📖 https://www.devmedia.com.br/servicos-restful-verbos-http/37103
https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api
https://repositorio-aberto.up.pt/bitstream/10216/281/2/67564.pdf
https://github.com/OAI/OpenAPI-Specification
https://en.wikipedia.org/wiki/OpenAPI_Specification
https://support.smartbear.com/swaggerhub/docs/tutorials/openapi-3-tutorial.html