

**UNIVERSIDADE FEDERAL DE OURO PRETO – UFOP**

**CIÊNCIA DA COMPUTAÇÃO**



**REDES DE COMPUTADORES**

**TRABALHO PRÁTICO 8**

Marcus Vinícius Souza Fernandes

19.1.4046

**Ouro Preto**

**2021**

## REST

REST não é um protocolo ou padrão, mas sim um conjunto de restrições de arquitetura. Os desenvolvedores de API podem implementar a arquitetura REST de maneiras variadas. Quando um cliente faz uma solicitação usando uma API RESTful, essa API transfere uma representação do estado do recurso ao solicitante ou endpoint.

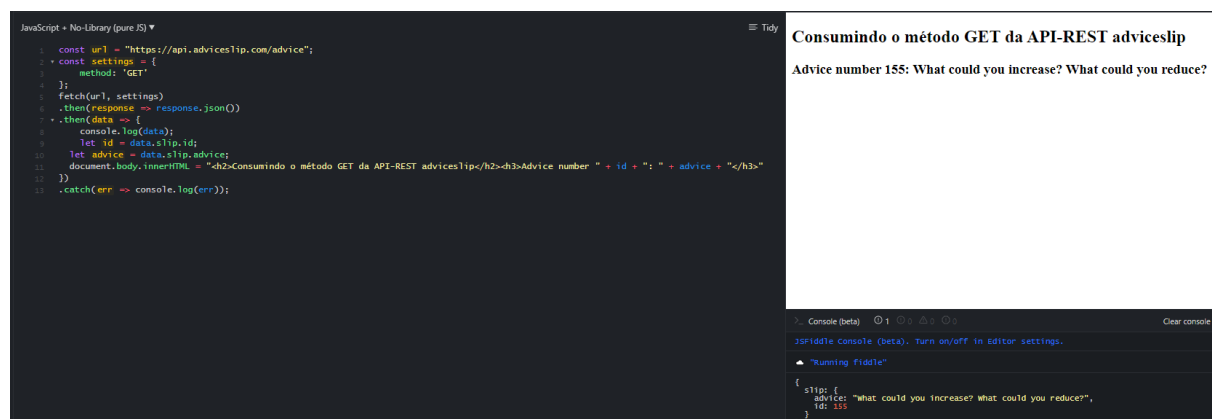
Essa informação (ou representação) é entregue via HTTP utilizando um dos vários formatos possíveis: Javascript Object Notation (JSON), HTML, XLT, Python, PHP ou texto sem formatação. O formato JSON é a linguagem de programação mais usada porque, apesar de seu nome, é independente de qualquer outra linguagem e pode ser lido por máquinas e humanos.

### Consumindo uma API pública (Método GET)

Como solicitado, a api-rest escolhida foi a “Advice Slip” (<https://api.adviceslip.com>), se trata de um serviço simples que basicamente em um de seus métodos GET, nos retorna um conselho aleatório. A mesma possui uma documentação muito interessante, organizada e acessível. Como a demanda foi relativamente simples, não foi necessário a criação de uma estrutura/projeto complexo para consumir o método desta mesma api.

Foi utilizado uma funcionalidade presente na linguagem javascript denominada “fetch”, ela fornece uma interface para acessar e manipular partes do pipeline HTTP, tais como requests e responses. Hoje, esta funcionalidade substituí o “XMLHttpRequest”.

O script criado e evidenciado abaixo se resume a atribuição do endereço equivalente ao método get random da api escolhida, construímos as configurações para a requisição (GET) e em seguida utilizamos a funcionalidade oferecida pelo fetch, no qual modelamos a resposta para o formato JSON e filtramos a data obtida para que possamos mostra-la em tela via construção HTML, comprovando assim, a eficácia do código. Também tratamos a ocorrência de possíveis erros com a funcionalidade catch.



```
JavaScript - No-Library (pure JS)
1 const url = "https://api.adviceslip.com/advice";
2 const settings = {
3   method: 'GET'
4 };
5 fetch(url, settings)
6   .then(response => response.json())
7   .then(data => {
8     console.log(data);
9     let id = data.slip.id;
10    let advice = data.slip.advice;
11    document.body.innerHTML = `<h2>Consumindo o método GET da API-REST adviceslip/<h2><h3>Advice number ` + id + `: ` + advice + `</h3>`;
12  })
13  .catch(err => console.log(err));
```

Consumindo o método GET da API-REST adviceslip

Advice number 155: What could you increase? What could you reduce?

```
{
  slip: {
    advice: "what could you increase? What could you reduce?",
    id: 155
  }
}
```

Contextualizando a imagem, à esquerda se encontra o código criado na linguagem javascript, à direita superior o nosso HTML gerado com as informações obtidas e no canto inferior direito temos o JSON obtido na requisição, sem nenhum tratamento.

## Referências

<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods>

<https://github.com/OAI/OpenAPI-Specification>

[https://developer.mozilla.org/pt-BR/docs/Web/API/Fetch\\_API/Using\\_Fetch](https://developer.mozilla.org/pt-BR/docs/Web/API/Fetch_API/Using_Fetch)

<https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api>