24 - Requisições POST com Javascript

Nessa aula você irá aprender como fazer requisições POST com a função **fetch()** do javascript e quais são as diferenças para uma requisição GET. Para ajudar na aula iremos usar um pacote do npm muito útil quando queremos testar interações com APIs, o **json-server**. Ele permite simular um backend de forma simples e rápida, subindo um pequeno servidor para receber nossas requisições e salvando dados em arquivo. Ele é capaz de receber requisições GET, POST, PUT e DELETE seguindo a arquitetura Rest para trabalhar com quaisquer recursos que quisermos definir.

1. Vamos começar criando uma pasta para a aula, inicializando o npm e instalando também o json-server, a única dependência que precisaremos:

```
npm init -y
```

```
npm install json-server
```

2. Com o json-server instalado, vamos apenas editar o arquivo **package.json** para incluir um script que roda o servidor e salva os dados em um arquivo chamado "db.json" na raiz do projeto:

```
// ... "scripts": { "json-server": "json-server --watch db.json" }, //
...
```

3. O último passo de preparação do json-server será criar um arquivo chamado "db.json" na raiz do projeto (junto com o package.json) e colocar alguns dados iniciais que usaremos no exemplo da aula:

```
{ "articles": [ { "id": 1, "title": "Olá, mundo!", "content": "Lorem ipsum dolor sit amet consectetur adipisicing elit. Quibusdam minus inventore voluptatem, eum repellat sed officiis aliquam", "author": "Isaac" }, { "id": 2, "title": "Requisições POST com Javascript", "content": "Lorem ipsum dolor sit amet consectetur adipisicing elit. Quibusdam minus inventore voluptatem, eum repellat sed officiis aliquam", "author": "Isaac" } ] }
```

4. Agora já podemos deixar o json-server rodando. Abra um terminal e execute-o com o comando:

```
npm run json-server
```

5. Avançando, nós iremos agora criar uma página HTML com um formulário para demonstrar o funcionamento de uma requisição POST. Esse formulário nos permitirá criar novos artigos no banco de dados fictício do json-server:

```
<!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta http-equiv="X-UA-Compatible" content="IE=edge"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Requisições POST com Javascript</title> </head> <body> <main style="display: flex; gap: 4rem; margin: 1rem 2rem;"> <form> <h2>Escrever um artigo</h2> <label for="title">Título:</label> <br> <input type="text" id="title" name="title"> <br> <label for="author">Autor:</label> <br> <input type="text" id="author" name="author"> <br> <label for="content">Conteúdo:</label> <br> <textarea id="content" name="content" cols="30" rows="10"></textarea> <br> <button>Salvar</button> </form> <div> <h2>Artigos</h2> <section id="articles"></section> </div> </main> <script src="index.js"> </script> </body> </html>
```

6. Agora vamos treinar um pouco mais o uso do **fetch()** para requisições GET como vimos na aula anterior. No arquivo **index.js** iremos criar uma função para obter os artigos salvos no db.json e outra para renderizá-los na tela, além de chamar a função fetchArticles() assim que o documento estiver carregado;

```
function renderArticle(articleData) { const article =
document.createElement('article') article.classList.add('article')
article.id = `article-${articleData.id}` const title =
document.createElement('h3') title.classList.add('article-title')
title.textContent = articleData.title const content =
document.createElement('div') content.classList.add('article-content')
content.innerHTML = articleData.content const author =
document.createElement('div') author.classList.add('article-author')
author.textContent = articleData.author article.append(title, author,
content) document.querySelector('#articles').appendChild(article) }
async function fetchArticles() { const articles = await
fetch("http://localhost:3000/articles").then(res => res.json())
articles.forEach(renderArticle) }
document.addEventListener('DOMContentLoaded', () => { fetchArticles()
})
```

7. Repare que se adicionarmos uma action e um method ao nosso form ele já consegue fazer requisições para o json-server, criando um novo artigo, e como a página é atualizada o novo artigo já aparece na tela. No entando, o que queremos é fazer as requisições POST através do javascript sem atualizar a página. Para isso podemos usar o fetch() novamente, dessa vez com algumas opções a mais:

```
const form = document.querySelector('form')
form.addEventListener('submit', async (ev) => { ev.preventDefault()
const articleData = { title: document.querySelector('#title').value,
author: document.querySelector('#author').value, content:
document.querySelector('#content').value } const response = await
fetch('http://localhost:3000/articles', { method: 'POST', headers: {
'Content-Type': 'application/json' }, body:
JSON.stringify(articleData) }) const savedArticle = await
response.json() form.reset() renderArticle(savedArticle)
console.log(savedArticle) })
```