



Desenvolvimento de Software para WEB

Aula 12 - Ajax

Professor: Anderson Almada

Introdução

- AJAX é o acrônimo de Asynchronous Javascript and XML
- Chamada de um recurso no servidor a partir de um código Javascript no navegador web, de forma que o resultado atualize apenas uma parte da página sem precisar fazer uma atualização dela inteira.
- Esta chamada é assíncrona, ou seja, o script que a chamou continua sua execução sem esperar pela resposta.

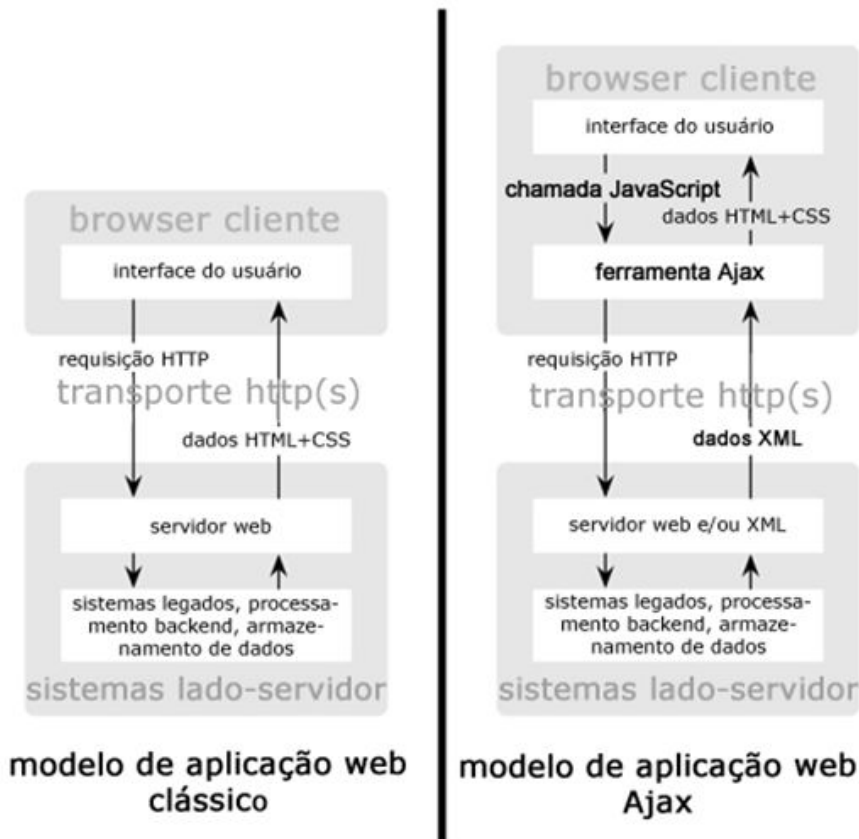
Introdução

- Quando o servidor responde, uma função Javascript especificada trata corretamente os dados retornados, fazendo a atualização de parte da tela apenas.
- conjunto de tecnologias que já existiam há muito tempo, como Javascript, DOM, CSS, XML, etc., para a criação de interfaces mais dinâmicas e responsivas

Introdução

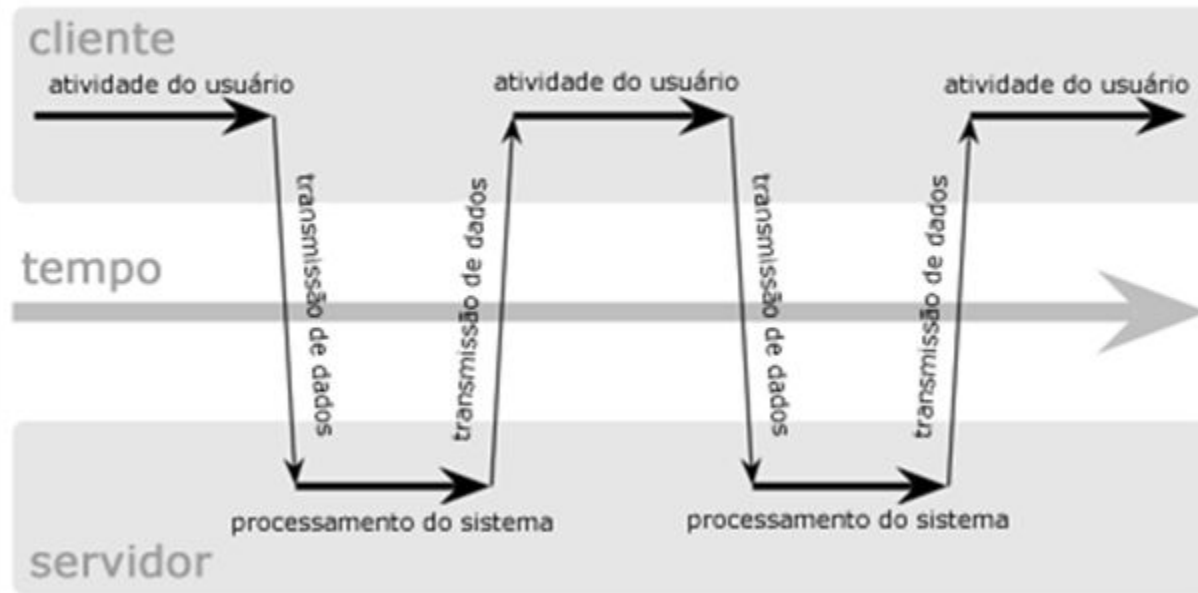
- XMLHttpRequest é o componente técnico que torna possível a comunicação assíncrona com o servidor
- No método open() do objeto XMLHttpRequest, o último parâmetro booleano determina como será a chamada: true para assíncrona, e false para síncrona.

Clássico x Ajax



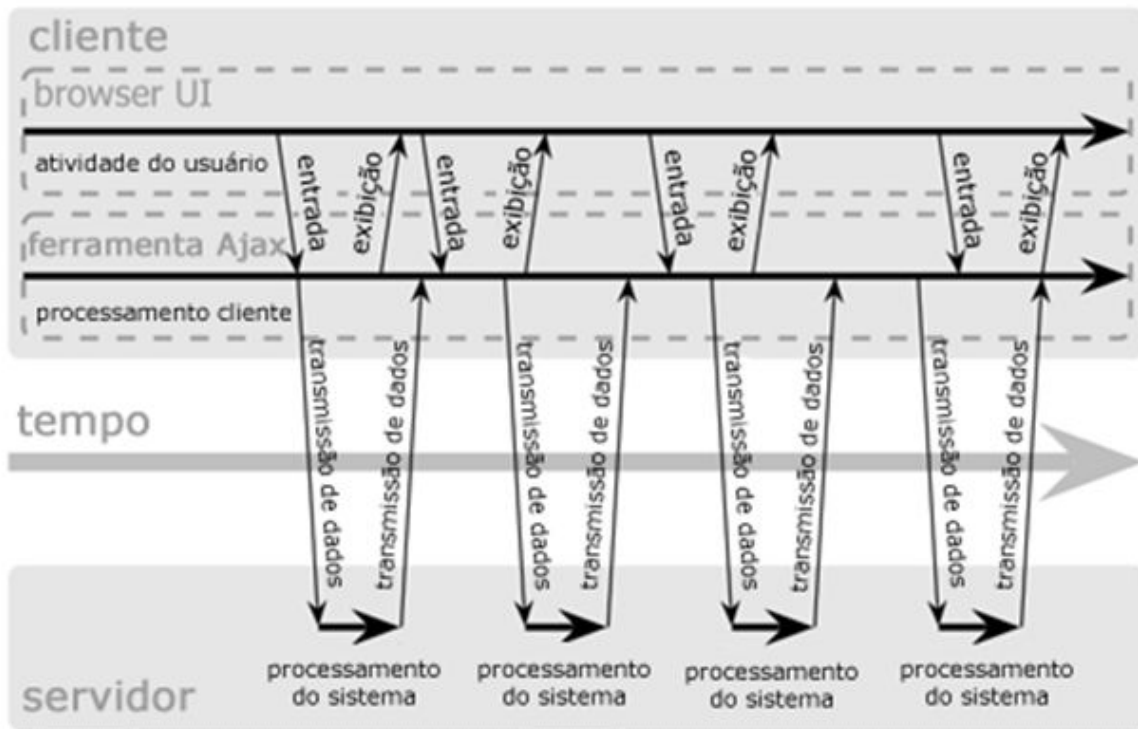
Síncrono

modelo de aplicação web clássico (síncrono)



Assíncrono

modelo de aplicação web Ajax (assíncrono)



Download e instalação

- NPM
 - <https://nodejs.org/en/download/>
- JSON-Server
 - <https://github.com/typicode/json-server>

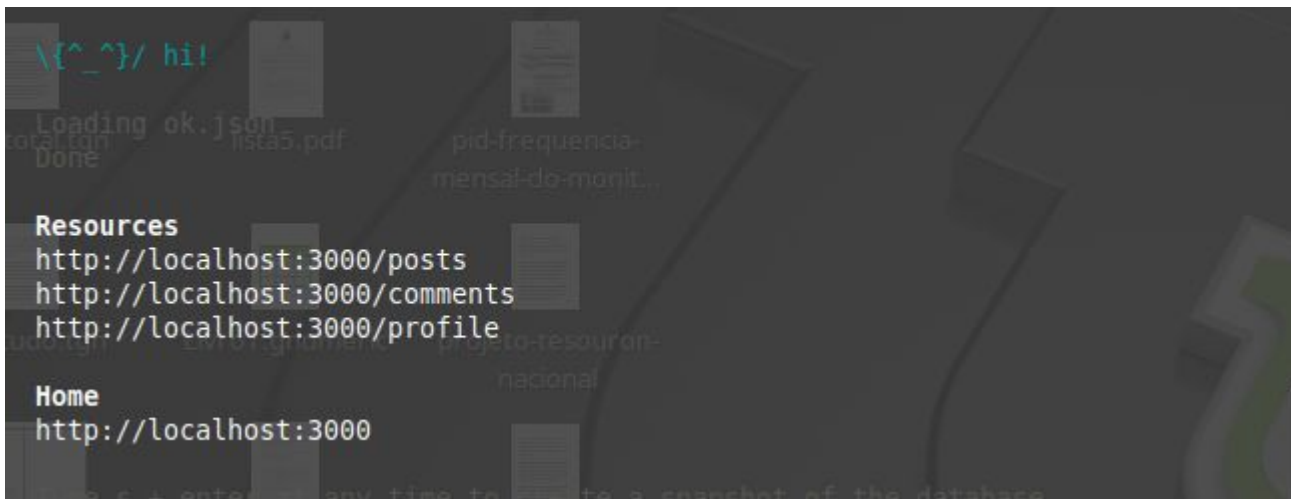
Exemplo de resources

- Crie o arquivo chamado **db.json**

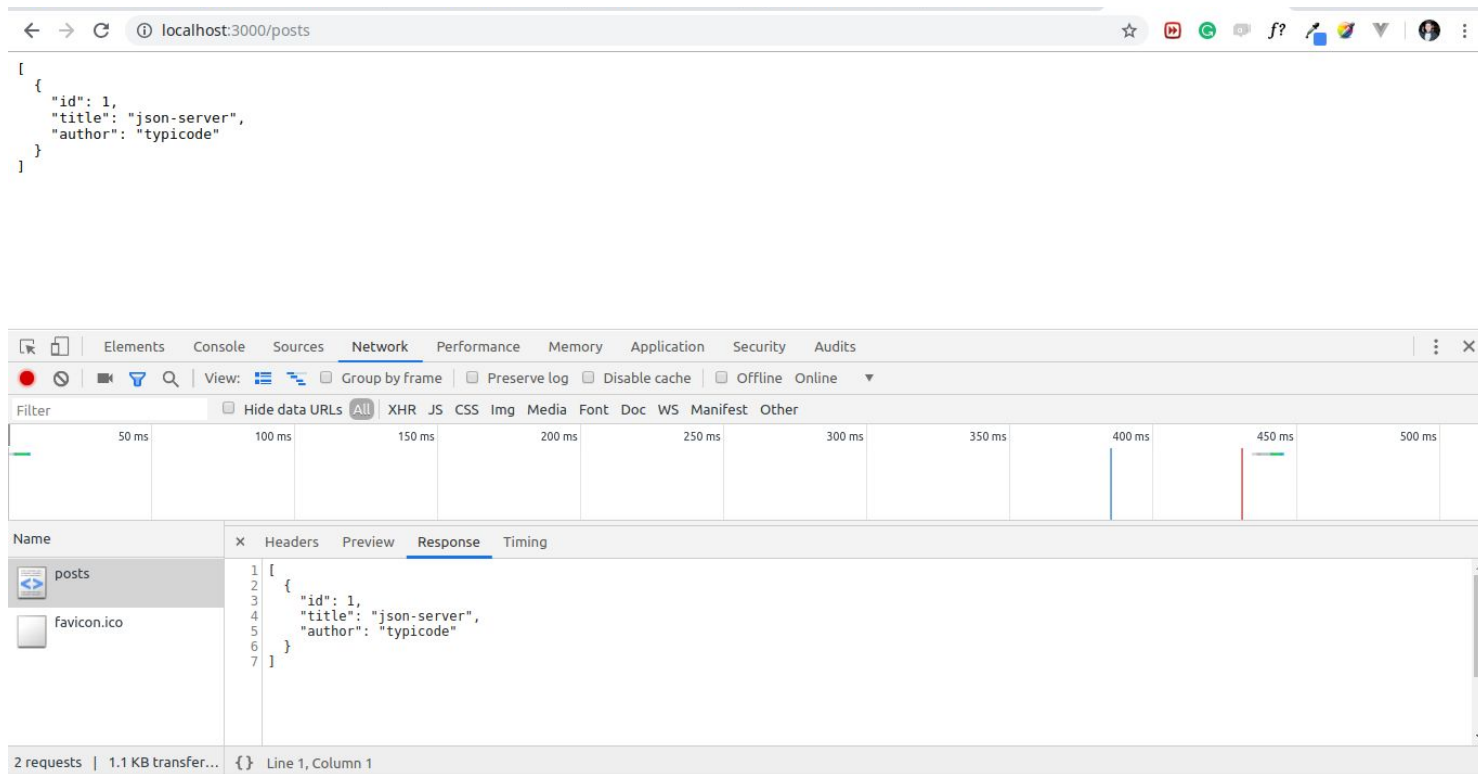
```
{  
  "posts": [  
    { "id": 1, "title": "json-server", "author": "typicode" }  
  ],  
  "comments": [  
    { "id": 1, "body": "some comment", "postId": 1 }  
  ],  
  "profile": { "name": "typicode" }  
}
```

Execute o JSON-Server

- `json-server --watch db.json`



Resultado no navegador



Postman - get

The screenshot displays the Postman application interface. At the top, there's a navigation bar with buttons for 'NEW', 'Runner', 'Import', and 'Builder'. Below this is a status bar with a notification about team collaboration. The main interface is divided into three sections: a left sidebar, a top bar, and a main workspace.

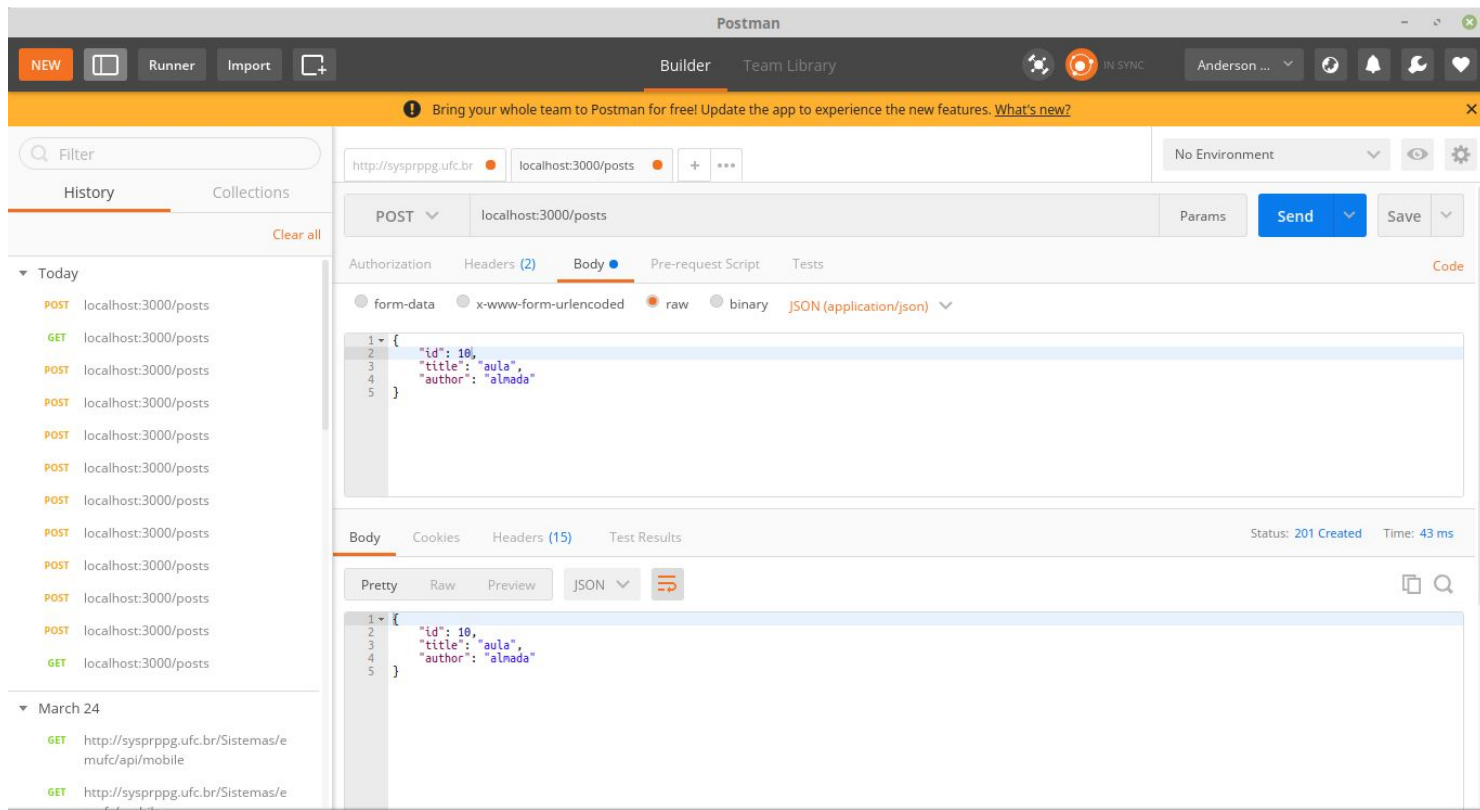
Left Sidebar: Contains a 'Filter' input, 'History' and 'Collections' tabs, and a 'Clear all' button. The 'History' tab is active, showing a list of recent requests. The first entry is a GET request to 'localhost:3000/posts'.

Top Bar: Shows the current request details: 'http://sysprpgg.ufc.br' and 'localhost:3000/posts'. It also includes a 'No Environment' dropdown, a 'Send' button, and a 'Save' button.

Main Workspace: The 'GET' method is selected. The 'Authorization' tab is active, showing 'No Auth'. The 'Body' tab is also active, displaying the response in 'Pretty' format. The response is a JSON array of 6 objects, each representing a post with 'id', 'title', and 'author' fields.

```
[{"id": 1, "title": "json-server", "author": "typicode"}, {"id": 2}, {"{"id": 3, "title": "aula", "author": "almda"}, {"id": 4}, {"{"id": 5, "title": "aula", "author": "almda"}, {"id": 6}, {"{"title": "aula", "author": "almda"}]
```

Postman - post



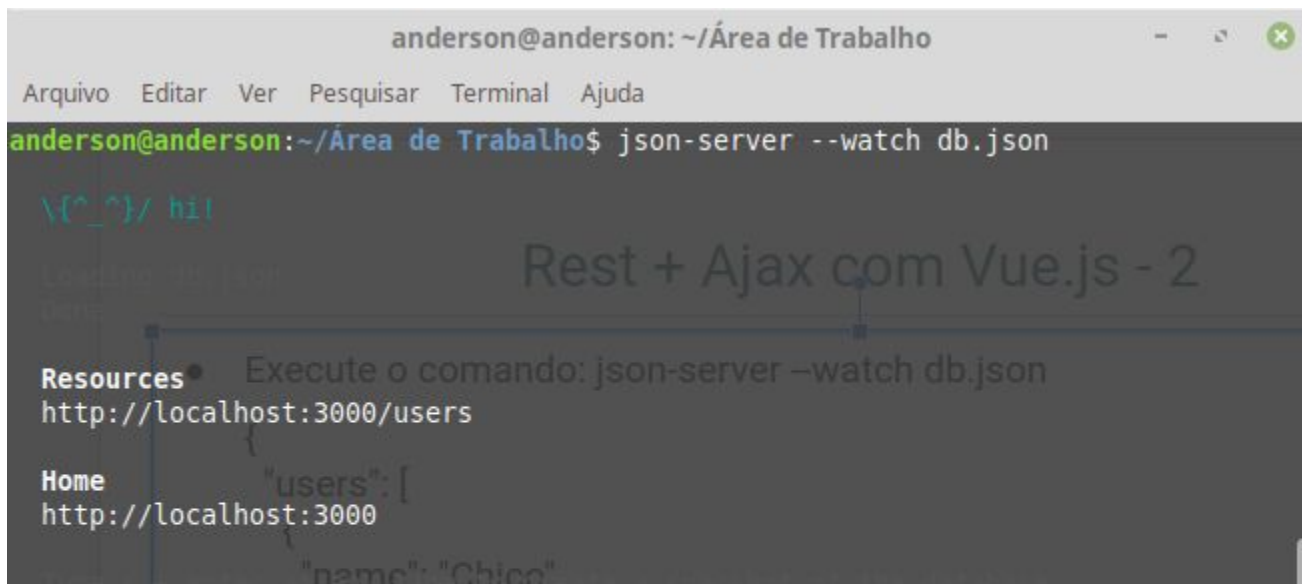
Rest + Ajax com Vue.js - 1

- Primeiro vamos construir nossa api rest (fake). Para isso, basta criar o arquivo **db.json**.

```
{
  "users": [
    {
      "name": "Chico",
      "username": "chiquim",
      "id": 1
    }
  ]
}
```

Rest + Ajax com Vue.js - 2

- Execute o comando: **json-server --watch db.json**



The screenshot shows a terminal window titled "anderson@anderson: ~/Área de Trabalho". The command `json-server --watch db.json` has been executed. Below the terminal, a REST client interface is visible, showing a "Resources" section with the URL `http://localhost:3000/users` and a "Home" section with the URL `http://localhost:3000`. The interface also displays a JSON response for the `users` endpoint, which is an array containing an object with `"name": "Chico"`.

```
anderson@anderson: ~/Área de Trabalho
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
anderson@anderson:~/Área de Trabalho$ json-server --watch db.json

\{^_^)/ hi!

Loading db.json
Done

Resources
http://localhost:3000/users

Home
http://localhost:3000

"users": [
  {
    "name": "Chico"
  }
]
```

Rest + Ajax com Vue.js - 3

- **HTML**

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
```

Ou

```
<script src="js/axios.min.js"></script>
```


Rest + Ajax com Vue.js - 4

- **HTML**

```
<button @click="getUsers">GET ALL</button>
<ul>
  <li v-for="usuario in usuarios">{{ usuario.id}} - {{ usuario.name
}}</li>
</ul>
```

Rest + Ajax com Vue.js - 5

- JS

```
var app = new Vue({
  el: "#app",
  data: {
    url: "http://localhost:3000/users",
    usuarios: [],
  },
  methods: {
    getUsers() {
      vm = this;
      axios.get(this.url).then(function (r) {
        console.log(r.data);
        vm.usuarios = r.data;
      }).catch(function (error) {
        console.log(error);
      });
    }
  }
});
```

Rest + Ajax com Vue.js - 6

- **HTML**

```
ID: <input type="text" v-model="id"><br>
<br><button @click="getUser">GET</button>
<p>{{ usuario.name }}</p>
```

- **JS**

```
getUser() {
  vm = this;
  axios.get(this.url+"/"+this.id).then(function (r) {
    console.log(r.data);
    vm.usuario = r.data;
  }).catch(function (error) {
    console.log(error);
  });
},
```

Rest + Ajax com Vue.js - 7

- **HTML**

```
ID: <input type="text" v-model="id"><br>
<br><button @click="getUser">GET</button>
<p>{{ usuario.name }}</p>
```

- **JS**

```
getUser() {
  vm = this;
  axios.get(this.url+"/"+this.id).then(function (r) {
    console.log(r.data);
    vm.usuario = r.data;
  }).catch(function (error) {
    console.log(error);
  });
},
```

Crie esse campo no **data**

Rest + Ajax com Vue.js - 8

- **HTML**

Nome: `<input type="text" v-model="name">
`

Usuário: `<input type="text" v-model="username">
`

`<button @click="postUser">POST</button>`

Rest + Ajax com Vue.js - 9

- JS

```
postUser() {  
  vm = this;  
  axios.post(this.url, {  
    name: vm.name,  
    username: vm.username  
  }).then(function (r) {  
    console.log(r.data);  
  }).catch(function (error) {  
    console.log(error);  
  });  
},
```



Crie esses campos no **data**

Rest + Ajax com Vue.js - 10

- **HTML**

```
<button type="button" @click="deleteUser">PUT</button>
```

- **JS**

```
deleteUser() {  
  axios.delete(this.url+"/"+this.id).then(function (r) {  
    console.log(r);  
  }).catch(function (error) {  
    console.log(error);  
  });  
}
```

Links importantes

- <https://www.devmedia.com.br/ajax-tutorial/24797https://blog.caelum.com.br/rest-principios-e-boas-praticas/>
- <https://developer.mozilla.org/pt-BR/docs/Web/Guide/AJAX>



Dúvidas??

E-mail: almada@crateus.ufc.br