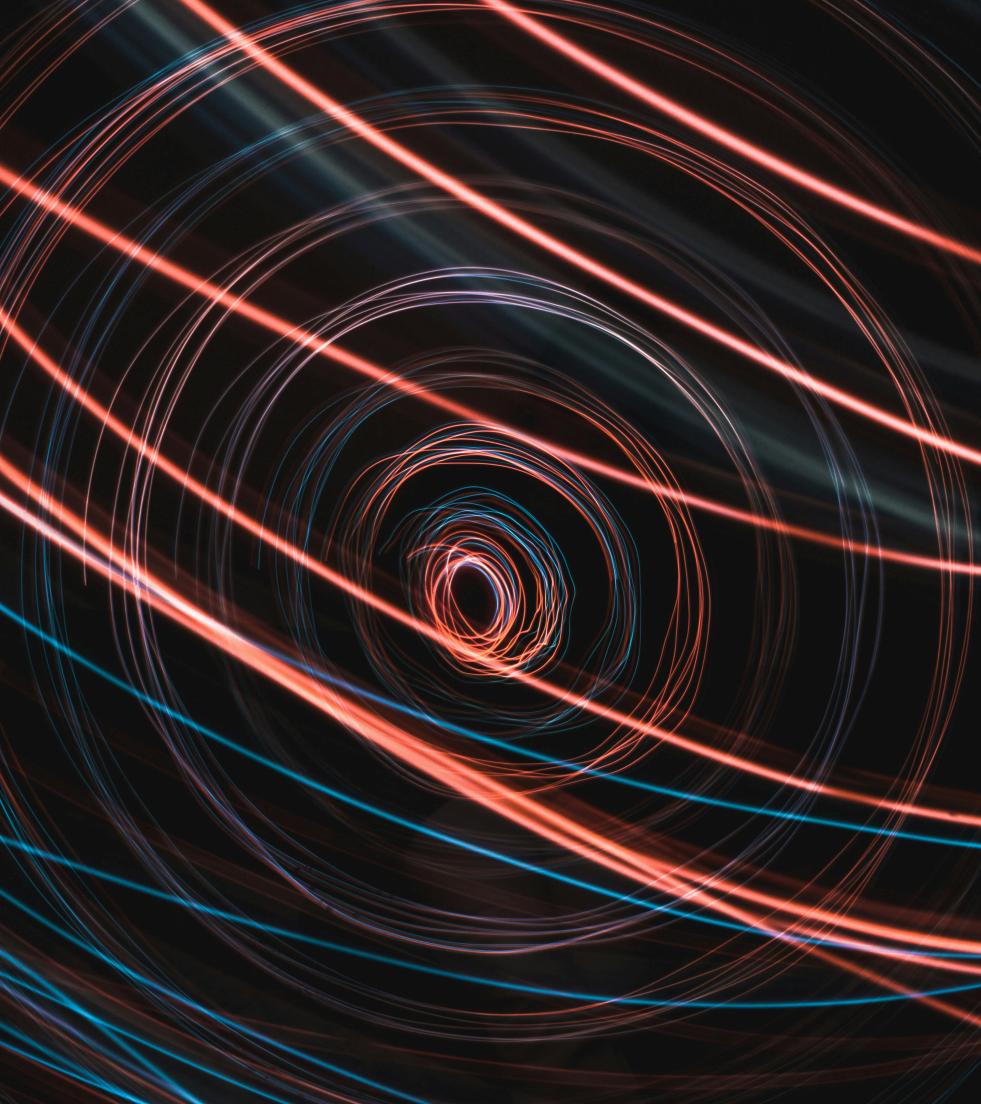


Consumindo APIs de Terceiros

Programação Web 1



Objetivo de Aprendizagem

- Aplicar APIs de terceiros em projetos próprios

Agenda

- Criando requisições com Nodejs
- Consumindo a API Hello

Criando requisições com Nodejs

Uma API define métodos e formatos para os dados das aplicações que podem ser utilizados e reutilizados para solicitação e recebimento de informações através de requisições do protocolo HTTP.

O módulo Express do Nodejs é bastante popular para criação de *endpoints* de APIs para atendimento às requisições dos usuários. Para consumir APIs o Nodejs será para aplicado na criação de requisições.

Como Realizar Requisições HTTP?

Módulos típicos

- `requests` (obsoleto)
- `Got`
 - Quick Start Guide
- `Node Fetch`
 - Common usage
- `SuperAgent`
 - Usage
- `Axios` (mais popular)
 - Example

Axios

- Realiza requisições HTTP a partir do Nodejs
- Suporta Promises
- Transformações de dados JSON automáticas

Promise

Objeto retornado pela chamada de métodos assíncronos e permite acesso às informações de eventual conclusão ou falha de uma operação assíncrona.

Promise

Possíveis estados

- *pending*: estado inicial
- *fulfilled*: operação concluída e com sucesso
- *rejected*: operação falhou

Usando Axios

Instalação

- `npm install axios`

```
1 const axios = require('axios');
```

Axios

GET usando *promise chain*

```
1 // Make a request for a user with a given ID
2 axios.get('/user?ID=12345')
3   .then(function (response) {
4     // handle success
5     console.log(response);
6   })
7   .catch(function (error) {
8     // handle error
9     console.log(error);
10  })
11  .finally(function () {
12    // always executed
13  });

```

Axios

GET com promise e params

```
1 axios.get('/user', {
2     params: {
3         ID: 12345
4     }
5 })
6 .then(function (response) {
7     console.log(response);
8 })
9 .catch(function (error) {
10    console.log(error);
11 })
12 .finally(function () {
```

Axios

GET com async/await

```
1  async function getUser() {  
2    try {  
3      const response = await axios.get('/user?ID=12345');  
4      console.log(response);  
5    } catch (error) {  
6      console.error(error);  
7    }  
8  }
```

Axios

Exemplo de POST

```
1 axios.post('/user', {
2     firstName: 'Fred',
3     lastName: 'Flintstone'
4 })
5     .then(function (response) {
6         console.log(response);
7     })
8     .catch(function (error) {
9         console.log(error);
10    });

```

Axios

```
axios(config)
```

```
1 // Send a POST request
2 axios({
3   method: 'post',
4   url: '/user/12345',
5   data: {
6     firstName: 'Fred',
7     lastName: 'Flintstone'
8   }
9 });
```

Consumindo a API Hello

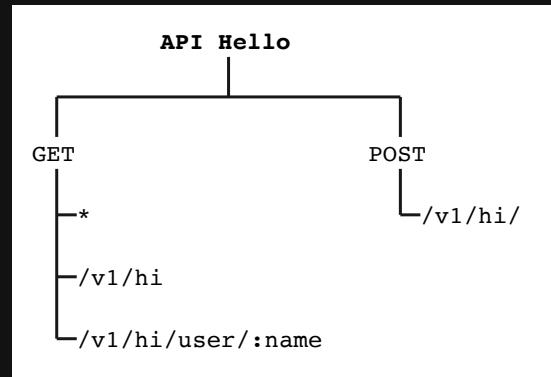
A ideia é criar a aplicacão `HelloApp` que utiliza ou aplica ou consome os recursos da API `Hello` criada na aula anterior.

HelloApp

1. Criar a rota `/` com um *form* de um único campo (Nome do usuário)
2. O *form* envia (via POST) para a rota `/who`
3. Em `/who` o Nome é passado pela API Hello que faz seu trabalho e retorna para HelloApp
4. Ainda em `/who` a informação é mostrada ao usuário

API Hello (Aula anterior)

Endpoints



HelloApp

Iniciando o projeto

1. Criar o diretório helloapp (ou outro nome de sua preferência)
2. `npm init -y`
3. `npm install express axios ejs`
4. Criar o arquivo `app.js`

HelloApp

app.js

```
1 const express = require('express')
2 const app = express()
3 const axios = require('axios')
4 app.set('view engine', 'ejs');
5 const PORT = 3000
6 app.listen(PORT, () => {
7   console.log(`Server running on http://localhost:${PORT}`)
8 })
9 app.use(express.urlencoded({ extended: true }))
10 // URL da API
11 const API_URL =
12 'https://orange-fiesta-9vg5wp94jvc7r67-8000.app.github.dev/'
```

Sugestão: Usar a API Hello através do GitHub Codespaces.

HelloApp

Adicionando as rotas / e /who ao app.js

```
1 app.get('/', async (req, res) => {
2     res.render('index');
3 }
4
5 app.post('/who', async (req, res) => {
6     const username = req.body.username
7     console.log(`Variável username recebida do form: ${username}`)
8     res.render('greeting', { msg: username })
9 })
```

HelloApp

Adicionando os *templates* em `/views`

5. Criar o diretório `views` e dentro dele o diretório `partials`
6. Em `views` adicionar os arquivos `index.ejs` e `message.ejs`
7. Em `partials` adicionar os arquivos `header.ejs` e `footer.ejs`

HelloApp

header.ejs e footer.ejs

```
1  <!doctype html>
2  <html lang="en">
3    <head>
4      <meta charset="utf-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1">
6      <title>HelloApp</title>
7      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-kfA&lt;script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kfA&lt;/body>
3  </html>
```

HelloApp

index.ejs

```
1  <%- include('partials/header'); %>
2  <div class="container d-flex vh-100">
3      <div class="row justify-content-center align-self-center w-100">
4          <div class="col-md-6">
5              <div class="card shadow">
6                  <div class="card-body">
7                      <h4 class="card-title mb-4">Quem é você?</h4>
8                      <form method="POST" action="/who">
9                          <div class="mb-3">
10                             <input type="text" class="form-control" name="username" placeholder="Digite seu nome" required />
11                         </div>
12                         <button type="submit" class="btn btn-primary w-100">Enviar</button>
13                     </form>
14                 </div>
15             </div>
16         </div>
17     </div>
18 </div>
19 <%- include('partials/footer'); %>
```

HelloApp

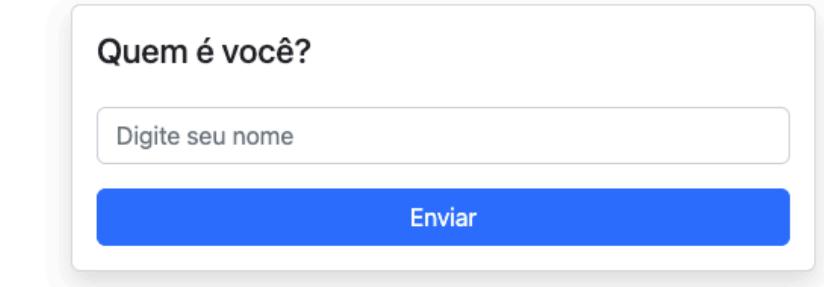
message.ejs

```
1  <%- include('partials/header'); %>
2  <div class="container d-flex vh-100">
3      <div class="row justify-content-center align-self-center w-100">
4          <div class="col-md-6">
5              <div class="card shadow">
6                  <div class="card-body">
7                      <h4 class="card-title mb-3">Resposta da API</h4>
8                      <p class="card-text mb-4"><strong><%= msg %></strong><br></p>
9                      <form method="GET" action="/">
10                         <button type="submit" class="btn btn-secondary w-100">Executar novamente</button>
11                     </form>
12                 </div>
13             </div>
14         </div>
15     </div>
16 </div>
17 <%- include('partials/footer'); %>
```

HelloApp

Teste parcial

8. Rode aplicação `node app` e
veja a tela inicial
9. Pare a aplicação



The image shows a screenshot of a web application interface. At the top, the question "Quem é você?" is displayed in bold black text. Below it is a light gray input field with the placeholder "Digite seu nome". At the bottom is a large blue rectangular button with the word "Enviar" in white.

HelloApp

/who

```
1 app.post('/who', async (req, res) => {
2   const username = req.body.username
3   console.log(`Variável username recebida do form: ${username}`)
4
5   try {
6     const response = await axios.get(`${API_URL}/v1/hi/`)
7     //console.log(response)
8     res.render('message', { msg: response.data.msg });
9   } catch (err) {
10     res.send('Erro ao acessar API.')}
```

E se... a API utilizada aceitar requisições apenas a partir de um método específico? POST, por exemplo.

HelloApp

/whobypost

```
1 app.post('/whobypost', (req, res) => {
2     const username = req.body.username
3     const url_ = `${API_URL}/v1/hi`
4     console.log(`Variável username recebida do form: ${username}`)
5     axios({
6         method: 'post',
7         url: url_,
8         headers: {
9             'Content-Type': 'application/x-www-form-urlencoded'
10        },
11        data: {
12            name: `${username}`
13        }
14    })
15    .then(response => {
16        //console.log(response)
17        res.render('message', {msg: response.data.msg});
18    })
19    .catch(err => {
20        console.error(err.message)
21        res.send('Erro ao acessar API.')})
```

APIs Públcas

- Existem diversos repositórios de APIs públicas que podem ser utilizadas em projetos
- Existem também APIs pagas
- Seguem alguns exemplos de respositórios de APIs públicas:
 - A directory of free and public APIs
 -  POSTMAN Public REST APIs

Exercícios

1

Criar a aplicação Mensagem do Dia que consome a API Ron Swanson Quotes.
Utilize Express, Bootstrap e EJS. Ferramentas alternativas podem ser aplicadas.

Referências

- [mdn web docs Promises](#)

- [API Ron Swanson Quotes](#)

-  [API Hello](#)

-  [HelloAPP](#)

José Roberto Bezerra

✉️ jbroberto@ifce.edu.br

⌚ [jbroberto76](#)

⌚ [pw1-repo](#)



Powered by  Sliddev

Cover image by José Campos from 