Backend com Node.js – Aula 2

Por **Wanderson Guimarães**

O que vamos aprender hoje:

- Revisão
- Callbacks e events
- Axios
- Chamadas Crud (Create, Read, Update e Delete) no Axios
- Express

Revisão da aula anterior

- Npm e node
- Import, Export ,Módulos
- Manipulando arquivos com fs
- Criando um servidor com http

Cjs (Common JS) e ESM (Es Modules)

CJS - É o sistema de módulos tradicional do Node.js.

```
// importar
const fs = require('fs');


// exportar
module.exports = minhaFuncao;
```

Cjs (Common JS) e ESM (Es Modules)

-Indicado para versões mais antigas do node.

-Suporte amplo em bibliotecas do ecossistema Node.

ESM (ES Modules)

- **É o padrão moderno de módulos em JavaScript.**
- Usado nativamente no navegador e também no Node.js (desde a versão 12+ com suporte completo na 14+).

ESM (ES Modules)

```
// importar

import fs from 'fs';

// exportar

export default minhaFuncao;
```

Callbacks

- Um Callback em Node.js é um equivalente assíncrono para uma função.
- É um tipo especial de função passada como argumento para outra função. O Node.js faz uso intenso de callbacks.
- Callbacks nos ajudam a fazer chamadas assíncronas.

Callbacks - Estrutura

```
function saudacao (nome, callback)
// (argumento, funcao que será chamada dentro da funcao saudacao)
{
 console.log(`Olá, ${nome}`);
 callback();
}
saudacao("João", () //aqui é onde é chamado o callback => {


console.log("Callback executado após a saudação.");
});
```

Nesse exemplo , temos:

function saudacao(nome, callback):

Cria uma função chamada saudacao.

Ela recebe dois parâmetros:

- nome: um texto qualquer (por exemplo, "João").
- callback: uma função que será executada depois da saudação.

console.log(\Olá, ${nome}`)`:

Exibe "Olá, João" no console.

callback();:

Chama a função que foi passada como argumento.

Chamada síncrona vs assíncrona

- Chamada síncrona : código é executado linha por linha.
- Cada linha **bloqueia a próxima** até terminar.

```
console.log("Início");

console.log("Meio");

console.log("Fim");
```

Chamada Assíncrona

- A execução **não bloqueia** a próxima linha.
- Permite fazer tarefas **em paralelo**, como ler arquivos ou acessar APIs.
- console.log("Início");

```
setTimeout(() => {
  console.log("Assíncrono: depois de 2 segundos");
}, 2000);

console.log("Fim");
```

Events no Node.js

Um **evento** representa algo que acontece em um sistema.

Muito usado para:

Streams
HTTP
Timers
Leitura de arquivos

Estrutura Básica

```
const EventEmitter = require('events');
const meuEmissor = new EventEmitter();

meuEmissor.on('evento', () => {
  console.log('Evento capturado!');
});

meuEmissor.emit('evento');
```

Explicação

.on ('evento', função) → Escuta um evento.

.emit ('evento') → Dispara (emite) um evento.

Pode passar parâmetros no .emit().

AXIOS

- **Axios** é uma **biblioteca baseada em Promises** usada para fazer requisições HTTP (GET, POST, PUT, DELETE, etc).

  Ela é leve, poderosa e funciona **tanto no frontend (navegador)** quanto no **backend (Node.js)**.

Exemplos de uso do AXIOS

```
axios.get('https://jsonplaceholder.typicode.com/posts/1')
 .then(response => {
  console.log(response.data);
 })
 .catch(error => {
  console.error(error);
 });
```

Quando usar o Axios?

- Quando precisa de uma API HTTP mais simples do que fetch.
- Quando deseja tratar erros facilmente.
- Quando precisa de interceptadores (para tokens, logs, etc.).
- Em projetos Node.js que acessam APIs REST.

EXPRESS

- **Express** é um **framework minimalista e flexível** para **Node.js**, usado para criar **servidores web** e **APIs** de forma rápida e organizada.

- Ele simplifica a criação de aplicações web ao lidar com rotas, requisições HTTP, middlewares, e muito mais.

Características principais

- **Fácil de usar e aprender**
- **Sistema de rotas** robusto
- **Middleware**: permite interceptar requisições e respostas
- Integração com bancos de dados, autenticação
- Suporte para views (template engines como EJS, Pug, etc)
- Organização ideal para projetos grandes (MVC, REST, etc)

Express - Exemplo

```
const express = require('express');
const app = express();
const PORT = 3000;

app.get('/', (req, res) => {
  res.send('Olá, mundo!');
});

app.listen(PORT, () => {
  console.log(`Servidor rodando em http://localhost:${PORT}`);
});
```

Express – Exemplo de uso

- express() → Cria a aplicação
- app.get() → Define uma rota GET
- res.send() → Responde ao cliente
- app.listen() → Inicia o servidor

Express – Exemplo rotas

```
app.get('/usuarios', (req, res) => {
  res.send('Listagem de usuários');
});

app.post('/usuarios', (req, res) => {
  res.send('Usuário criado');
});
```

Express – Exemplo Middleware

```
app.use(express.json()); // Permite ler JSON no corpo da requisição

app.post('/dados', (req, res) => {
  console.log(req.body);
  res.send('Dados recebidos');
});
```

ATIVIDADE - Axios com json server

npm install axios json-server

Criar um script json db.json

```
{
  "usuarios": [
    { "id": 1, "nome": "Maria" },
    { "id": 2, "nome": "João" }
  ]
}
```

npx json-server --watch db.json --port 3001

Axios com json server – Get e Post

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

// Fazendo uma requisição GET
axios.get('http://localhost:3001/usuarios')
  .then(response => {
    console.log('Usuários:', response.data);
  })
  .catch(error => {
    console.error('Erro ao buscar usuários:', error);
  });

// Requisição POST
axios.post('http://localhost:3001/usuarios', {
  nome: 'Carla'
}).then(response => {
  console.log('Usuário adicionado:', response.data);
});
```

Axios com json server - Put

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

axios.put('http://localhost:3001/usuarios/2', {
  nome: 'João Silva'
})
.then(response => {
  console.log('Usuário atualizado:', response.data);
})
.catch(error => {
  console.error('Erro ao atualizar usuário:',
error.message);
});
```

Axios com json server - Delete

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

axios.delete('http://localhost:3001/usuarios/3')
  .then(response => {
    console.log('Usuário removido com sucesso!');
  })
  .catch(error => {
    console.error('Erro ao remover usuário:', error.message);
  });
```