

## Material de Apoio

Site: [Geração Tech](#)  
Curso: Formação em Desenvolvedor Web - Online  
Livro: Material de Apoio

Impresso por: JOÃO VITOR DE MELO FREITAS  
Data: quinta-feira, 18 jul. 2024, 23:08

# Índice

## **1. Back-End**

1.1. Tecnologias de Back-End

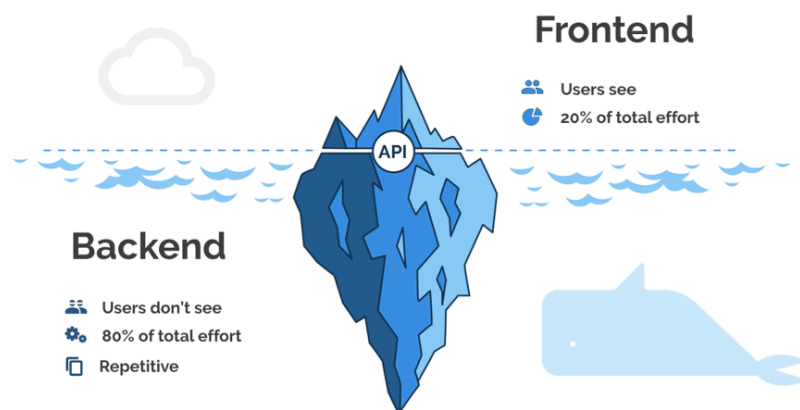
1.2. Video da Aula

1.3. Executando JavaScript com Node.js

1.4. Video da Aula

# 1. Back-End

Olá pessoal, tudo bem? Sou o Márcio Ferreira, especialista em desenvolvimento web. Estarei com vocês nesse curso de back-end. Vamos começar entendendo o que é o back-end e como ele funciona.



## 1.1. Tecnologias de Back-End

### Tecnologias de Back-End

Existem várias tecnologias que podem ser utilizadas para desenvolver o back-end, como:

- **Node.js**
- **PHP**
- **Java**
- **Python**
- **Ruby**

Cada uma dessas tecnologias tem suas características e é utilizada de acordo com as necessidades do projeto.

### Fluxograma de Funcionamento

Vamos desenhar um fluxograma básico para entender melhor como o back-end interage com o front-end e o banco de dados.

#### 1. Front-End:

- **HTML e CSS:** Estrutura e estilo da página.
- **JavaScript:** Interatividade e lógica no cliente.

#### 2. Back-End:

- **Node.js, PHP, Java, Python, Ruby:** Processamento e lógica no servidor.
- **Banco de Dados:** Armazenamento e recuperação de dados.

### Comunicação Cliente-Servidor

#### 1. Cliente:

- O cliente (usuário) interage com o front-end.
- Exemplo: Um usuário pesquisa "guitarra" no Google.

#### 2. Servidor:

- O front-end envia uma solicitação para o back-end.
- O back-end processa a solicitação, consulta o banco de dados e retorna as informações necessárias.
- Exemplo: O servidor do Google retorna os resultados da pesquisa.

### Tecnologias Server-Side e Client-Side

- **Client-Side:** Executado no navegador do cliente (HTML, CSS, JavaScript).
- **Server-Side:** Executado no servidor (Node.js, PHP, Java, Python, Ruby).

### Introdução ao Node.js

Node.js é uma tecnologia que permite a execução de JavaScript no lado do servidor. Com Node.js, podemos:

- Fazer conexão com o banco de dados.
- Processar requisições paralelas.
- Trabalhar com sockets.
- Interagir com outras APIs.

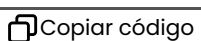
### Instalação do Node.js

Para instalar o Node.js:

1. Acesse o site oficial do Node.js.
2. Faça o download da versão recomendada.
3. Siga as instruções de instalação.

Após a instalação, verifique se está tudo correto:

```
bash
```



Copiar código


```
node -v npm -v
```

O Node.js utiliza o motor V8 para executar JavaScript diretamente no servidor, convertendo-o para uma linguagem que o computador possa entender.

## Primeiros Passos com Node.js

Vamos criar um exemplo simples para entender melhor como o Node.js funciona. Crie um arquivo chamado `app.js` e adicione o seguinte código:


```
javascript
```

 Copiar código

```
console.log("Olá, mundo!");
```

Para executar, use o comando:

```
bash
```

 Copiar código

```
node app.js
```

Esse comando irá rodar o código JavaScript diretamente no servidor, exibindo "Olá, mundo!" no console.

## Conclusão

Neste curso, vamos explorar profundamente o Node.js, aprendendo como ele pode ser utilizado para construir aplicações robustas e escaláveis. Vamos entender como criar APIs, interagir com bancos de dados e muito mais.

Estou animado para embarcar nessa jornada com vocês e espero que aproveitem ao máximo este curso!

Vamos juntos!

## 1.2. Video da Aula

dia 01 nodejs 1 1



## 1.3. Executando JavaScript com Node.js

Olá pessoal, tudo bem? Sou o Márcio Ferreira, especialista em desenvolvimento web. Hoje vamos continuar a nossa aula de back-end. Na aula passada, vimos como instalar o Node.js e como funciona um pouco do back-end. Hoje, iremos executar JavaScript dentro da nossa máquina com Node.js.

### Passos para Configurar o Ambiente

#### 1. Verificação do Node.js e npm

- Verifique se o Node.js está instalado:

```
bash
```

```
Copiar código
```

```
node -v
```

- Verifique a versão do npm:

```
bash
```

```
Copiar código
```

```
npm -v
```

#### 2. Abrir o Visual Studio Code

- Abra o Visual Studio Code e crie uma nova pasta para o projeto:

```
plaintext
```

```
Copiar código
```

```
File -> Open Folder -> Selecione a pasta -> New Folder -> backend-aula-1
```

#### 3. Criar um Arquivo JavaScript

- Crie um arquivo chamado `primeiro-programa.js`:

```
javascript
```

```
Copiar código
```

```
console.log("Olá, sou JS");
```

#### 4. Executar o Código JavaScript no Terminal

- Abra o terminal no VS Code:

```
plaintext
```

```
Copiar código
```

```
Terminal -> New Terminal
```

- Execute o arquivo JavaScript com o Node.js:

```
bash
```

```
Copiar código
```

```
node primeiro-programa.js
```

- O output deve ser:

```
plaintext
```


```
Copiar código
```

Olá, sou JS

## Trabalhando com Variáveis e Lógica de Programação

Vamos expandir nosso exemplo com variáveis e operações:


javascript

 Copiar código

```
const numero1 = 10; const numero2 = 20; const soma = numero1 + numero2; console.log("A soma é: " + soma);
```

Execute novamente com o comando:

bash

 Copiar código

```
node primeiro-programa.js
```


## Introdução ao npm

O npm (Node Package Manager) é uma ferramenta que acompanha o Node.js e é usada para gerenciar pacotes e bibliotecas de terceiros. Vamos entender como ele funciona.

### 1. Inicializando um Projeto com npm

- No terminal, dentro da pasta do projeto, execute:

bash

 Copiar código

```
npm init
```

- O comando `npm init` interativo irá perguntar várias informações para criar um arquivo `package.json`.

### 2. Configuração Rápida com `npm init -y`

- Para criar o arquivo `package.json` rapidamente, execute:

bash

 Copiar código

```
npm init -y
```

## Configurando o `package.json`

O arquivo `package.json` contém informações sobre o projeto, incluindo dependências que o projeto precisa para funcionar. Aqui está um exemplo básico:

json

 Copiar código

```
{ "name": "primeiro-programa", "version": "1.0.0", "description": "Aula de back-end", "main": "primeiro-programa.js",  
  "scripts": { "test": "echo \"Error: no test specified\" && exit 1" }, "author": "Márcio Ferreira", "license": "ISC" }
```

## Instalação de Pacotes com npm

Vamos instalar um pacote chamado `nodemon` que nos ajuda a reiniciar automaticamente o servidor sempre que houver alterações no código:

bash


 Copiar código



```
npm install --save-dev nodemon
```

Atualize o `scripts` no `package.json` para usar o `nodemon`:

```
json
```

 Copiar código

```
"scripts": { "start": "nodemon primeiro-programa.js" }
```

Agora, inicie o projeto com:

```
bash
```

 Copiar código

```
npm start
```

O `nodemon` reiniciará automaticamente sempre que houver mudanças no arquivo.

## Conclusão

Nesta aula, vimos como configurar um ambiente de desenvolvimento Node.js, criar e executar scripts JavaScript e utilizar o npm para gerenciar pacotes. Continuaremos explorando mais funcionalidades do Node.js e do npm nas próximas aulas.

Espero que tenham gostado da aula e nos vemos na próxima sessão!

## 1.4. Video da Aula

dia 01 nodejs 2 1

