## Node.js

Node.js é uma plataforma de código aberto construída sobre o motor JavaScript V8 do Google Chrome. Ele permite que os desenvolvedores usem JavaScript para escrever código do lado do servidor, o que antes era possível apenas do lado do cliente, no navegador.

Com Node.js, os desenvolvedores podem criar aplicativos web escaláveis e de alto desempenho, lidar com milhares de conexões simultâneas e criar APIs RESTful eficientes. Sua natureza assíncrona e baseada em eventos o torna ideal para lidar com operações de entrada e saída intensivas, como acesso a banco de dados e comunicação de rede.

Esses conceitos são fundamentais para entender o funcionamento do desenvolvimento back-end e são a base sobre a qual os desenvolvedores constroem aplicativos web robustos e escaláveis. Ao dominar esses fundamentos, os desenvolvedores estão bem equipados para enfrentar os desafios do desenvolvimento back-end moderno.

**Criando Primeiro Projeto Node**

O módulo HTTP é uma parte essencial do Node.js, permitindo criar servidores web e lidar com solicitações HTTP. Vamos criar nosso primeiro projeto node usando http:

1. Certifique-se de ter o Node.js instalado em sua máquina. Você pode baixá-lo e instalá-lo a partir do site oficial: https://nodejs.org/.
2. Inicialização do Projeto. Crie um novo diretório para o seu projeto e navegue até ele no terminal. Em seguida, inicialize um novo projeto Node.js executando o seguinte comando:

   ```
   npm init -y
   ```

   Isso criará um arquivo *package.json* com as configurações padrão para o seu projeto.

3. Crie um novo arquivo JavaScript para o seu servidor Node. Você pode chamá-lo de *server.js*, por exemplo.
4. No arquivo *server.js*, importe o módulo HTTP do Node.js adicionando a seguinte linha de código:

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

5. Utilize o método *http.createServer()* para criar um servidor HTTP. Você precisa passar uma função de callback que será chamada sempre que o servidor receber uma solicitação. Dentro desta função, você pode adicionar código para lidar com as solicitações HTTP. Por exemplo, para enviar uma resposta simples para todas as solicitações:

```
const server = http.createServer((req, res) => {
   res.writeHead(200, { 'Content-Type': 'text/plain' });
   res.end('Hello, world!');
});
```

6. Utilize o método *server.listen()* para fazer o servidor escutar as requisições HTTP em uma porta específica. Você pode escolher qualquer número de porta válido. Por exemplo, para ouvir na porta 3000:

```
const PORT = 3000;
server.listen(PORT, () => {
  console.log(`Server is listening on port ${PORT}`);
});
```

7. Por fim, execute o arquivo server.js para iniciar o servidor Node.js. No terminal, navegue até o diretório do seu projeto e execute o seguinte comando:

```
node server.js
```

8. Agora você pode abrir o seu navegador e acessar http://localhost:3000. Assim conseguirá ver a mensagem 'Hello, world!'. Parabéns, você acaba de criar seu primeiro servidor.

Acabamos de criar um servidor usando http no entanto, devido à sua natureza de baixo nível e complexidade para tarefas comuns, como roteamento e manipulação de solicitações, muitos desenvolvedores preferem usar frameworks como *Express.js*, que simplificam essas tarefas e oferecem uma experiência de programação mais produtiva. Vamos ver também um exemplo usando o Express:

1. Certifique-se de ter o Node.js instalado em sua máquina. Você pode baixá-lo e instalá-lo a partir do site oficial: https://nodejs.org/.

2. Inicie um novo projeto Node.js. Abra um terminal e navegue até o diretório onde deseja criar o projeto. Em seguida, execute o seguinte comando para criar um novo arquivo package.json:

```
npm init -y
```

3. Instale o Express como uma dependência do seu projeto. No terminal, execute o seguinte comando:

```
npm install express
```

4. Crie um arquivo JavaScript para o seu servidor. Por exemplo, você pode nomeá-lo server.js.
5. Abra o arquivo server.js no seu editor de código favorito e comece configurando o servidor Express. Aqui está um exemplo básico para iniciar um servidor e definir uma rota simples:

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

app.get('/', (req, res) => {
  res.send('Bem-vindo ao meu servidor com Express!');
});

app.listen(PORT, () => {
  console.log(`Servidor Express rodando na porta ${PORT}`);
});
```

6. Salve o arquivo server.js, vá ao terminal, navegue até o diretório do seu projeto e execute o seguinte comando para iniciar o servidor:

```
node server.js
```

7. Abra um navegador da web e vá para http://localhost:3000. Você deverá ver a mensagem "Bem-vindo ao meu servidor com Express!".

Isso é tudo! Você criou com sucesso um servidor simples usando o Express. A partir daqui, você pode expandir seu servidor adicionando mais rotas, middleware, manipulação de erros e muito mais, conforme necessário para o seu aplicativo.

## Conceito e Implementação do CRUD

CRUD é um acrônimo para Create (Criar), Read (Ler), Update (Atualizar) e Delete (Excluir), e refere-se às quatro operações básicas que podem ser realizadas em dados persistentes. No contexto de desenvolvimento de software, o CRUD é comumente utilizado para descrever as operações básicas de manipulação de dados em um banco de dados ou sistema de armazenamento.

- **Create (Criar):** Esta operação envolve a criação de novos registros ou objetos no sistema. Por exemplo, criar um novo usuário em um sistema de gerenciamento de usuários.
- **Read (Ler):** Esta operação envolve a recuperação de dados existentes do sistema. Por exemplo, ler as informações de um usuário específico a partir do banco de dados.
- **Update (Atualizar):** Esta operação envolve a modificação dos dados existentes no sistema. Por exemplo, atualizar o endereço de e-mail de um usuário.
- **Delete (Excluir):** Esta operação envolve a remoção de registros ou objetos do sistema. Por exemplo, excluir um usuário do banco de dados.

O CRUD é uma parte fundamental do desenvolvimento de aplicações web e é utilizado em conjunto com várias tecnologias e frameworks para criar, ler, atualizar e excluir dados de forma eficiente e segura.