PRÁTICAS AVANÇADAS EM DESENVOLVIMENTO WEB

Carga horária: 90 horas

Davi Schneid

08/07/2024

Agenda

- Criar o servidor back-end NODE.js
- APIs RESTful
  - Projeto front-end + back-end
- Banco dados
  - Relacionais e não relacionais.
- Segurança e Autenticação
  - Métodos de autenticação
  - Proteger dados sensíveis
- Frameworks React.js ou Angular.js
  - Integrar back-end
  - Gerenciar dependências Node.js
- Git code version

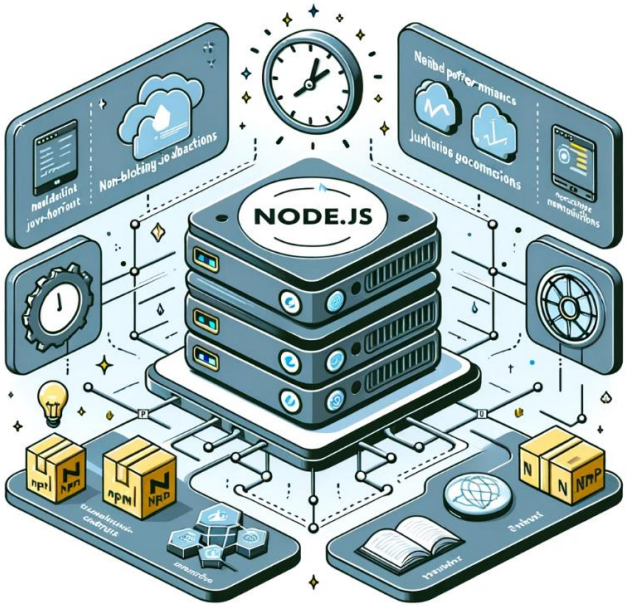
Cronograma

08/07-12/07	15/07-19/07	22/07-26/07	29/07-02/08	05/09-09/08	12/08-16/08
Criação servidor Nodejs. • Configuração inicial do ambiente • Implementação de um servidor básico **API RESTFul.** • Desenvolvimento de endpoints • Testes iniciais da API **Projeto Implementação.** • Definição do escopo do projeto • Início da implementação do projeto	**Projeto Implementação** • Desenvolvimento das funcionalidades principais. • Integração inicial dos componentes **Banco de dados** • Configuração do banco de dados. • Conexão com o servidor Node.js • Criação de tabelas e schemas	**Segurança e autenticação** • Introdução aos conceitos de segurança • Implementação de autenticação **Métodos de autenticação** • Autenticação por token • Autenticação baseada em sessões **Proteção dos dados sensíveis** • Criptografia de dados • Gerenciamento de permissões e acessos	**Frameworks React.js ou Angular.js** • Introdução aos frameworks • Configuração inicial do projeto frontend **Integrar back-end** • Comunicação entre frontend e backend • Testes de integração **Gerenciar dependências Node.js** • Uso de npm/yarn para gerenciar pacotes • Atualização e manutenção de dependências	**Frameworks React.js ou Angular.js** • Introdução aos frameworks • Configuração inicial do projeto frontend **Integrar back-end** • Comunicação entre frontend e backend • Testes de integração **Gerenciar dependências Node.js** • Uso de npm/yarn para gerenciar pacotes • Atualização e manutenção de dependências	**Apresentação dos projetos** • Preparação e refinamento final do projeto • Ensaios de apresentação • Apresentação final dos projetos

Introdução

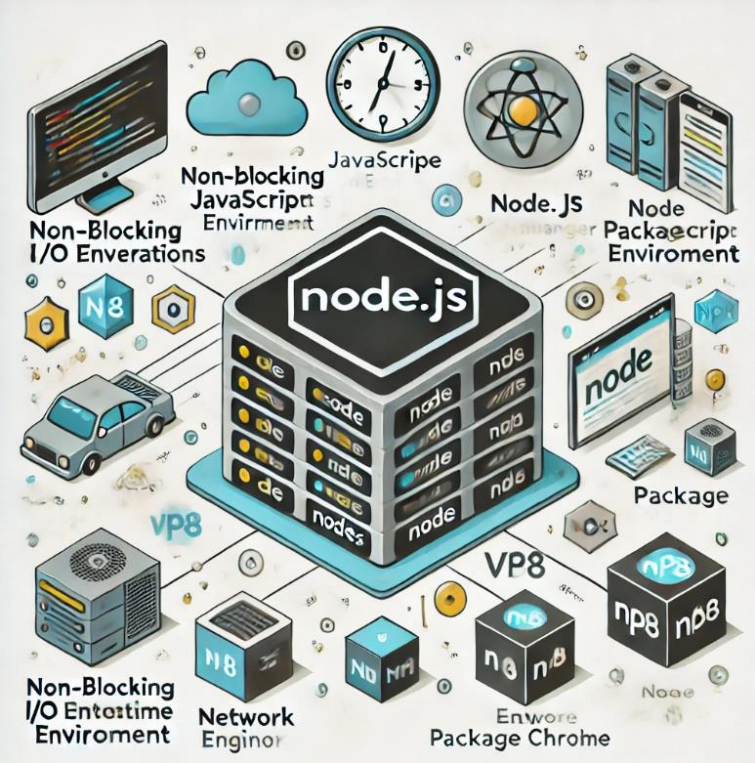
- O que é Node.js?
- V8 Engine
- Event-driven
- Non-blocking I/O
- Single-Threaded
- Node Package Manager - NPM

Node.js



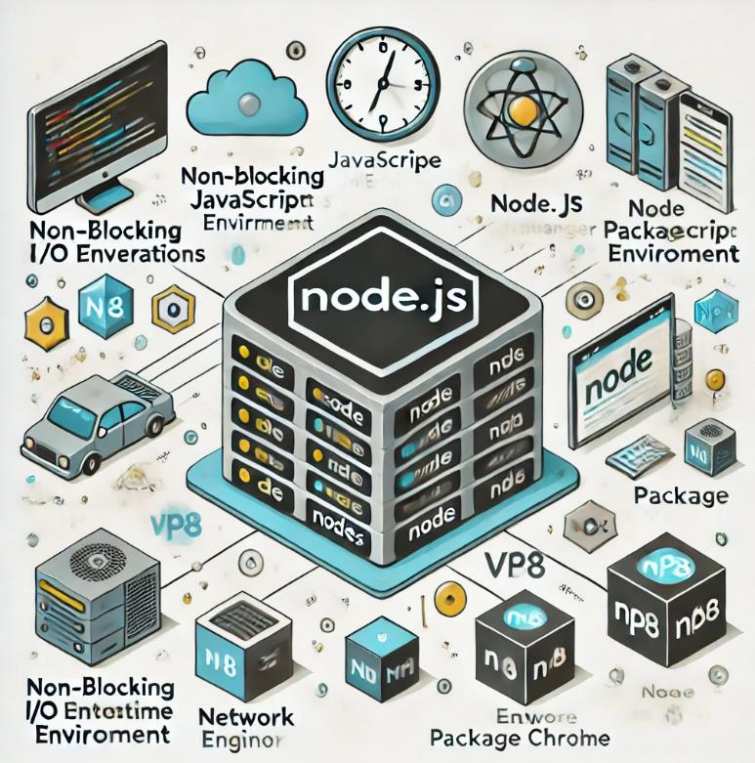
- JavaScript no Servidor
- Módulos Nativos

Engine V8



- Compilação Just-In-Time (JIT)
- Garbage Collection (Coleta de Lixo)
- Suporte a ES6
- Otimizações de Desempenho
- Multi-threading

Event Driven



- Eventos e Listeners
- Event Loop
- Callbacks e Assíncrono
- Event Emitters
- Multi-threading

Event Driven

- Eventos e Listeners
- Event Loop
- Callbacks e Assíncrono
- Event Emitters
- Multi-threading

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

// Cria um servidor HTTP
const server = http.createServer((req, res) => {
  // Evento 'request'
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello, world!\n');
});

// Ouvinte para o evento 'listening'
server.on('listening', () => {
  console.log('Server is listening on port 3000');
});

// Inicia o servidor e escuta na porta 3000
server.listen(3000);
```

Non-blocking I/O

- Operações de entrada e saída
- Não bloqueia a thread
- Leitura e escrita de arquivos
- Acesso ao BD
- I/O é iniciada event loop
- I/O é concluída
- Callback

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

// Leitura assíncrona de um arquivo
fs.readFile('example.txt', 'utf8', (err, data) => {
  if (err) {
    console.error('Error reading file:', err);
    return;
  }
  console.log('File content:', data);
});

console.log('This will run before the file reading is completed');
```

Single-Threaded (Monothread)

- Single-Threaded
- Simplicidade de Programação
- Uso Eficiente de Recursos
- Escalabilidade

```
console.log('Start');

setTimeout(() => {
  console.log('Timeout callback');
}, 1000);

console.log('End');
```

Node Package Manager (NPM)

- Instalação de Pacotes

```
npm install express
```

- Gerenciamento de Dependências

```
npm install
```

- Publicação de Módulos

```
npm publish
```

- Scripts do NPM

```
{
  "scripts": {
    "start": "node app.js",
    "test": "mocha"
  }
}
```

```
npm start
npm test
```

- **Repositório Central**
  https://www.npmjs.com/

Node Package Manager (NPM)

- Inicialização do Projeto

```
npm init -y
```

- Instalação de Pacotes

```
npm install express
```

- Uso dos Pacotes Instalados

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

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

app.listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

Exercicios

- Criar projeto