



# Anhanguera

UNIVERSIDADE PITÁGORAS UNOPAR  
POLO DE APOIO, JUNDIAÍ-SP  
CURSO SUPERIOR DE **SISTEMAS DE INFORMAÇÃO**

NOME DO AUTOR: ALISSON HENRIQUE CORREIA

DESENVOLVIMENTO COM FRAMEWORK NODE.JS

São Paulo—SP  
2024

# DESENVOLVIMENTO COM FRAMEWORK NODE.JS

Trabalho apresentado à Universidade Pitágoras Unopar  
como requisito parcial à aprovação no  
Quinto Semestre do curso de  
Sistemas de informação.

São Paulo–SP  
2024

## SUMÁRIO

INTRODUÇÃO.....	4
CONSTRUINDO UM SERVIDOR.....	5/6/7/8/9/10/11
TESTANDO E DEPURANDO APLICAÇÕES NODE.JS.....	12/13
INTERFACES DE USUÁRIO COM NODE.JS.....	14/17
ESTRATÉGIAS DE TESTES.....	18/19/20/21
CONSIDERAÇÕES FINAIS.....	22
REFERÊNCIAS.....	23

## INTRODUÇÃO

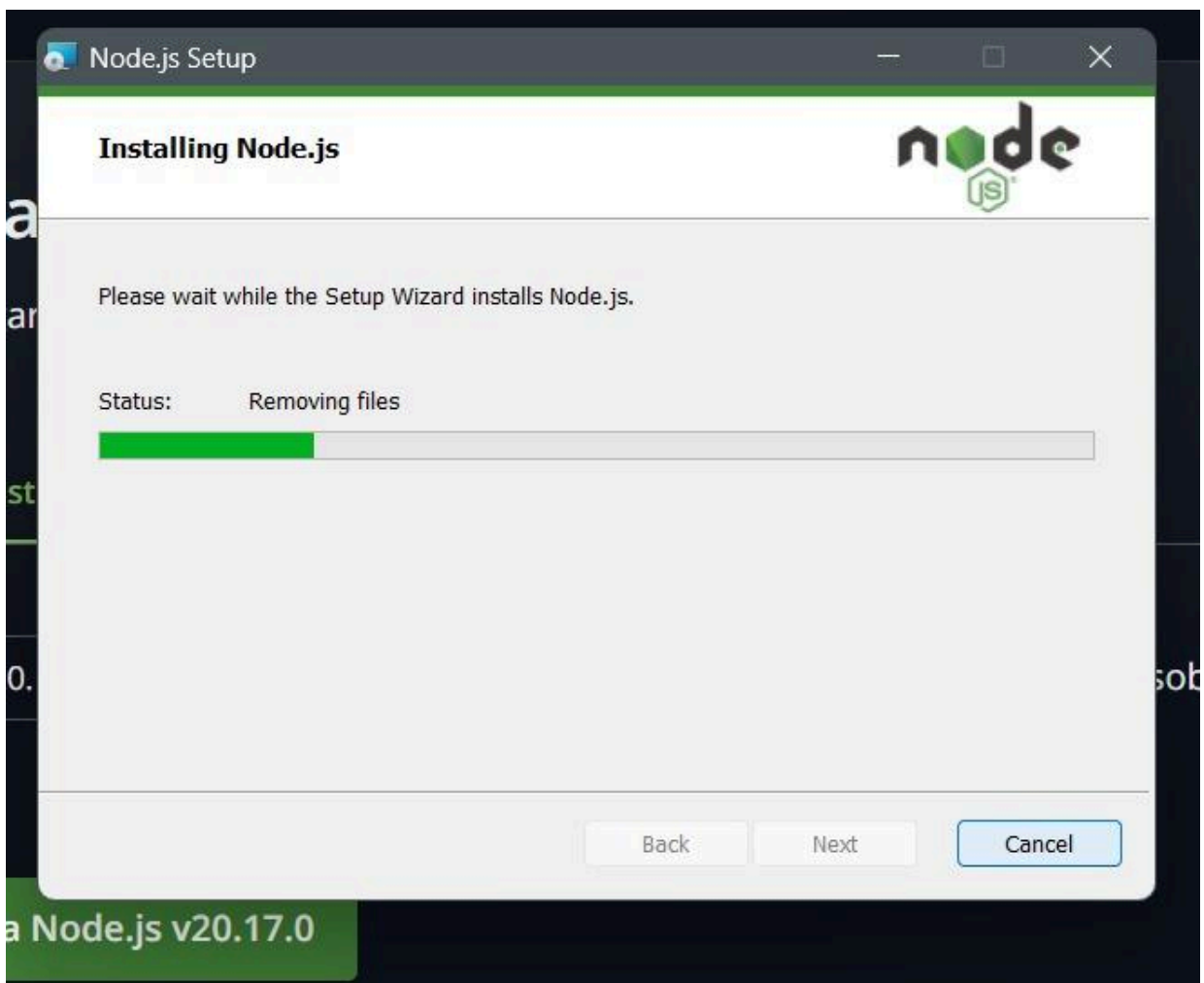
Neste trabalho foi proposto a elaboração de uma série de Projeto de DESENVOLVIMENTO COM FRAMEWORK NODE.JS, seguindo as regras propostas pelo tutor, solucionando assim o exercício solicitado. Com o fim de entender a forma de organizar e desenvolver um projeto com o auxílio de ferramentas dessa ferramenta.

# CONSTRUINDO UM SERVIDOR WEB BÁSICO

## Desenvolvimento

### Inicialização do Projeto Node.js

Para iniciar o projeto, instalei o Node.JS e configurei para que eu pudesse usá-lo dentro da IDE Visual Studio Code(VSCode).



# Criação de pastas, pacotes e arquivo.js

Através de comandos realizados dentro do PowerShell no VSCode, direcionei o projeto para o diretório correto, criei um package.js e também um arquivo servidor.js

Navegando para a pasta usando o comando mkdir.



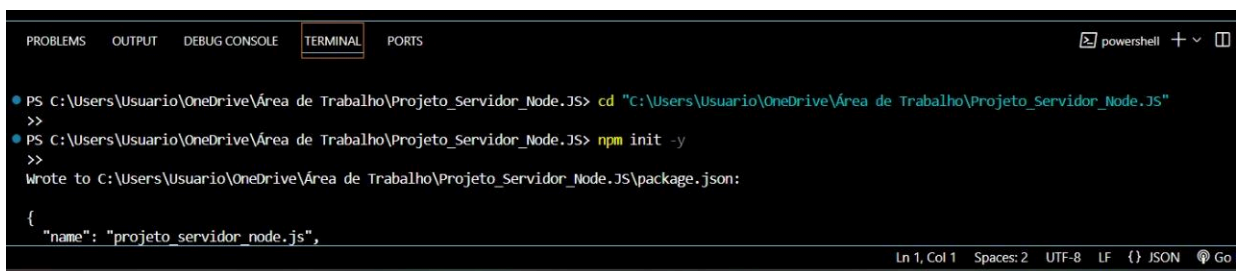
The screenshot shows the VS Code interface with the 'TERMINAL' tab selected. The terminal displays the command `mkdir Projeto_Servidor_Node.JS` and its output, which includes the directory path and a table of file details.

```
PS C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS> mkdir Projeto_Servidor_Node.JS
```

Diretório: C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto\_Servidor\_Node.JS

Mode	LastWriteTime	Length	Name
d----	05/09/2024 16:46		Projeto_Servidor_Node.JS

Criando package.json

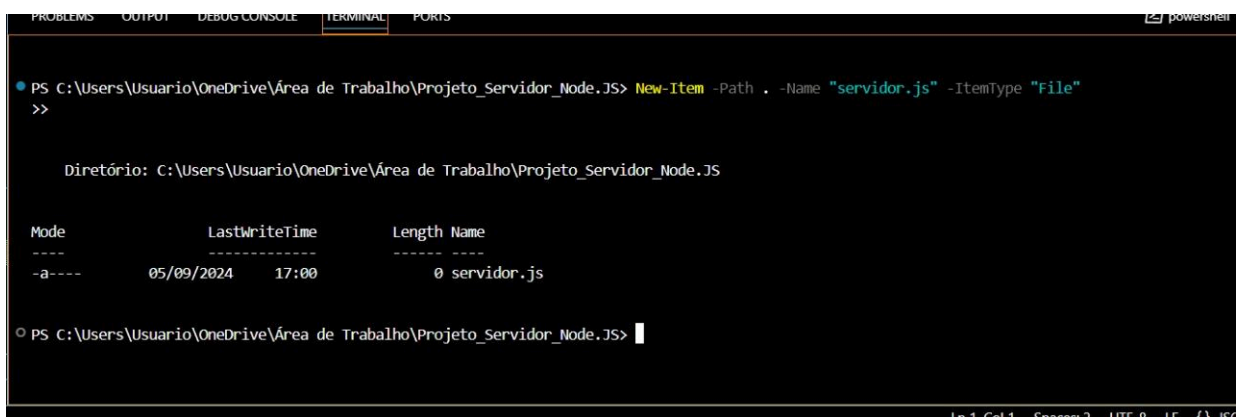


The screenshot shows the VS Code interface with the 'TERMINAL' tab selected. The terminal displays the commands `cd "C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS"` and `npm init -y`, followed by the output of the `npm init` command, which creates a `package.json` file.

```
PS C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS> cd "C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS"
>>
PS C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS> npm init -y
>>
Wrote to C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS\package.json:

{
  "name": "projeto_servidor_node.js",
```

Criando arquivo servidor.js.



The screenshot shows the VS Code interface with the 'TERMINAL' tab selected. The terminal displays the command `New-Item -Path . -Name "servidor.js" -ItemType "File"` and its output, which includes the directory path and a table of file details.

```
PS C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS> New-Item -Path . -Name "servidor.js" -ItemType "File"
>>
```

Diretório: C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto\_Servidor\_Node.JS

Mode	LastWriteTime	Length	Name
-a----	05/09/2024 17:00	0	servidor.js

```
PS C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS>
```

## Configuração das Respostas [/ , /sobre, /contato]

Através de uma estrutura condicional criei um laço para que toda vez que a resposta, ou seja, toda vez que a busca na URL estiver correta o conteúdo deverá ser apresentado, e se não estiver, a mensagem página não encontrada deverá ser apresentada.

```
,  
  
if (url === '/') {  
  res.statusCode = 200;  
  res.end(createResponse('bem-vindo a pagina inicial!'));  
} else if (url === '/sobre') {  
  res.statusCode = 200;  
  res.end(createResponse('esta e a pagina sobre.'));  
} else if (url === '/contato') {  
  res.statusCode = 200;  
  res.end(createResponse('esta e a pagina de contato.'));  
} else {  
  // Resposta padrão para qualquer outra rota (página não encontrada)  
  res.statusCode = 404;  
  res.end(createResponse('pagina nao encontrada.'));  
}  
};
```

## Execução do Servidor

Após todas as configurações serem feitas e da estrutura de código ser montada foi possível executar o servidor com sucesso.

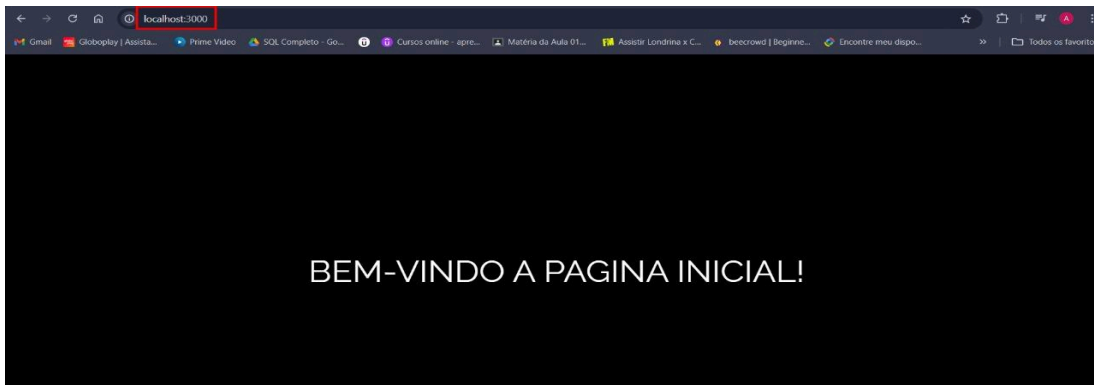
```
52 }  
53 );  
54  
55 // Iniciando o servidor  
56 server.listen(port, () => {  
  
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS  
  
PS C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS> node servidor.js  
Servidor rodando em http://localhost:3000/  
█
```

A mensagem apresentada no terminal indica que está funcionando o servidor dentro dos parâmetros esperados e definidos para ele.

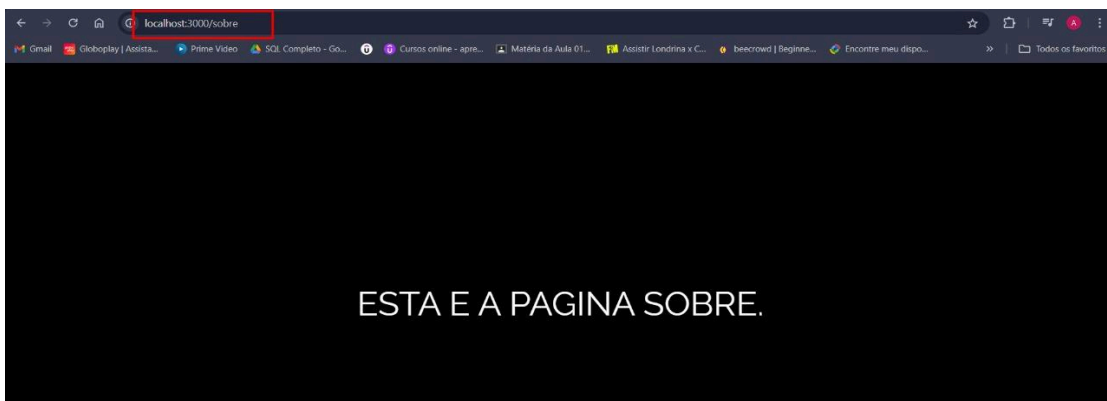
## Testes de Funcionamento

Os testes de funcionamento nos apresenta os resultados positivos do servidor em execução.

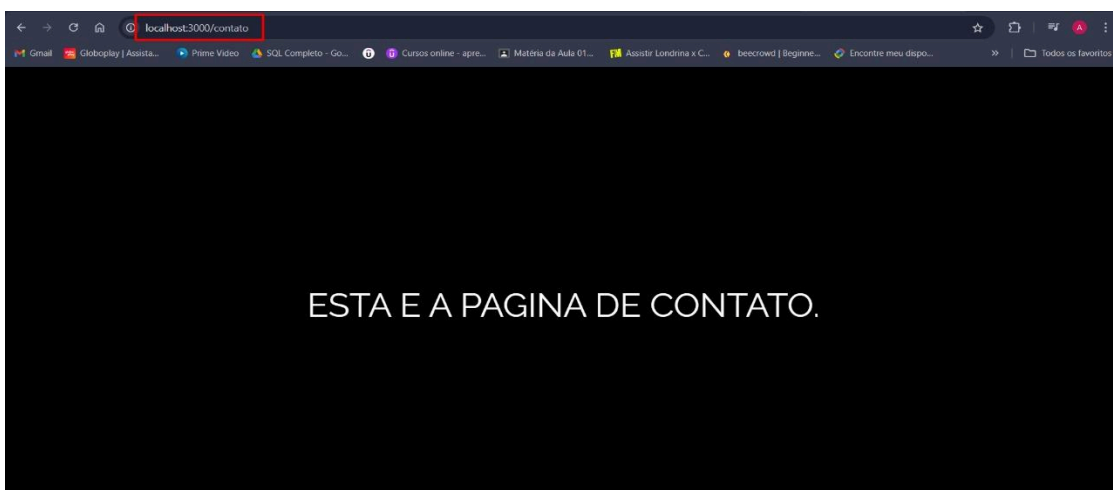
Teste 1 localhost:3000



Teste 2 Localhost:3000/sobre

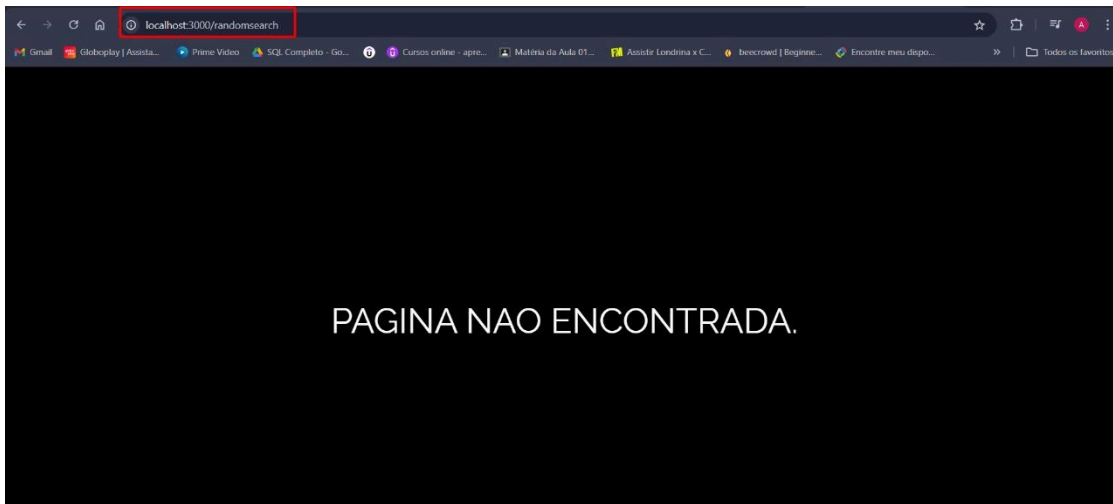


Teste 3 Localhost:3000/contato





## Teste 4 Localhost:3000/informaçãoqualquer



## Estrutura e Organização do Código

O código possui a criação do servidor e define a porta em que ele deve funcionar, também possui comandos HTML para definir cor e fonte das informações apresentadas, tudo isso além da estrutura condicional da apresentação das respostas em resultado da busca da URL.

```
1  const http = require('http');
2
3
4  const port = 3000;
5
6  const server = http.createServer((req, res) => {
7
8    const url = req.url;
9
10
11    res.setHeader('Content-Type', 'text/html');
12
13    const createResponse = (message) => `
14      <html>
15        <head>
16          <title>Servidor Node.js</title>
17          <style>
18            @import url('https://fonts.googleapis.com/css2?family=Raleway:wght@400;700&display=swap');
19            body {
20              font-size: 48px;
21              font-family: 'Raleway', sans-serif;
22              text-transform: uppercase;
23              color: white;
24              background-color: black;
25              display: flex;
26              justify-content: center;
27              align-items: center;
28              height: 100vh;
29              margin: 0;
30            }
31          </style>
```

```

15 servidor.js > [⌘] server > [⌘] http.createServer() callback > [⌘] createResponse
1 6   const server = http.createServer((req, res) => {
13   const createResponse = (message) => `
29       margin: 0;
30   }
31   </style>
32   </head>
33   <body>
34       ${message}
35   </body>
36   </html>
37   `;
38
39   if (url === '/') {
40       res.statusCode = 200;
41       res.end(createResponse('bem-vindo a pagina inicial!'));
42   } else if (url === '/sobre') {
43       res.statusCode = 200;
44       res.end(createResponse('esta e a pagina sobre.'));
45   } else if (url === '/contato') {
46       res.statusCode = 200;
47       res.end(createResponse('esta e a pagina de contato.'));
48   } else {
49       // Resposta padrão para qualquer outra rota (página não encontrada)
50       res.statusCode = 404;
51       res.end(createResponse('pagina nao encontrada.'));
52   }
53   });
54
55   // Iniciando o servidor
56   server.listen(port, () => {
57       console.log(`Servidor rodando em http://localhost:${port}/`);

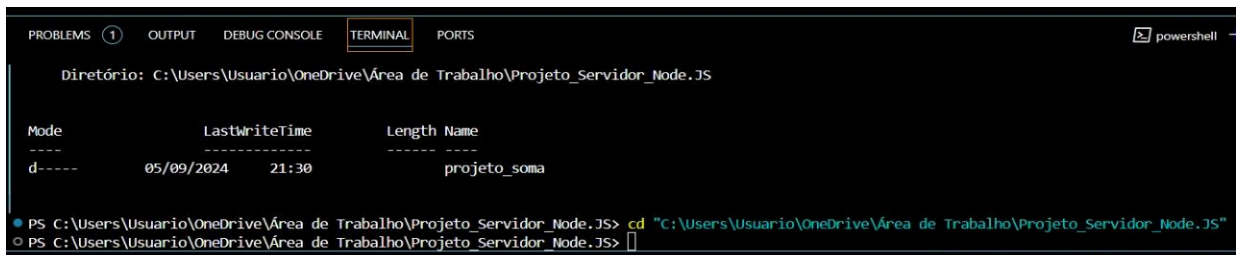
```

# TESTANDO E DEPURANDO APLICAÇÕES NODE.JS

## Desenvolvimento

Inicializar um projeto Node.js e configurar o Mocha como dependência de desenvolvimento.

### Criando o projeto soma



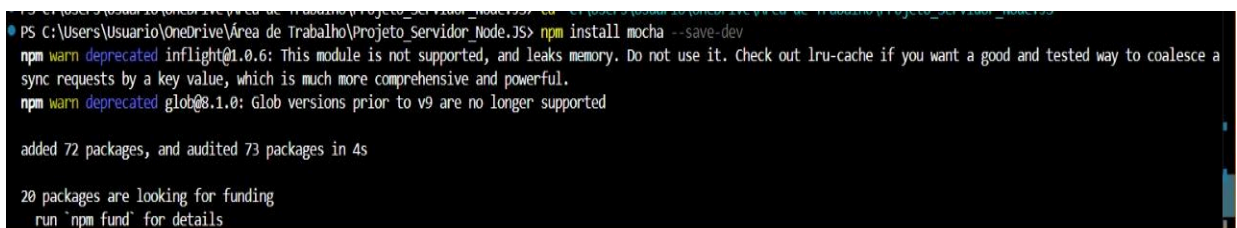
```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell

Diretório: C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS

Mode                LastWriteTime         Length Name
----                -
d-----          05/09/2024   21:30             projeto_soma

PS C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS> cd "C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS"
PS C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS> 
```

### Baixando e configurando o Mocha



```
PS C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS> npm install mocha --save-dev
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache if you want a good and tested way to coalesce a
sync requests by a key value, which is much more comprehensive and powerful.
npm warn deprecated glob@8.1.0: Glob versions prior to v9 are no longer supported

added 72 packages, and audited 73 packages in 4s

20 packages are looking for funding
  run 'npm fund' for details
```

### Criando o arquivo math.js



```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Diretório: C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS

Mode                LastWriteTime         Length Name
----                -
-a-----          05/09/2024   21:37             0 math.js
```

### Criando o arquivo math.test.js

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Diretório: C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS

Mode                               LastWriteTime           Length Name
----                               -
-a----- 05/09/2024    21:37                0 math.js
```

## Função implementada no math.js

```
JS servidor.js 1 JS math.js X JS math.test.js {} package.json

JS math.js > ...
1 // math.js
2 function add(a, b) {
3   |   return a + b;
4 }
5
6 module.exports = { add };
7
```

## Testes escrito no arquivo math.test.js

```
test > JS math.test.js > describe('Funções Matemáticas') callback > describe('Soma') callback > it('Deve retornar 0 ao somar 0 e 0') callback

1 // test/math.test.js
2 const assert = require('assert');
3 const { add } = require('../math');
4
5 describe('Funções Matemáticas', () => {
6   describe('Soma', () => {
7     it('Deve retornar 5 ao somar 2 e 3', () => {
8       |   assert.strictEqual(add(2, 3), 5);
9     });
10
11     it('Deve retornar -1 ao somar -2 e 1', () => {
12       |   assert.strictEqual(add(-2, 1), -1);
13     });
14
15     it('Deve retornar 0 ao somar 0 e 0', () => {
16       |   assert.strictEqual(add(0, 0), 0);
17     });
18   });
19 });
```

## Script de teste no package.json

```
{ package.json > name

1 {
2   "name": "projeto soma",
3   "version": "1.0.0",
4   "main": "index.js",
5   |> Debug
6   "scripts": {
7     | "test": "mocha"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC",
12  "description": "",
13  "devDependencies": {
14    | "mocha": "^10.7.3"
15  }
16 }
```

## Resultado dos testes usando o comando npm test

```
PS C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS> npm test

> projeto_soma@1.0.0 test
> mocha

Funções Matemáticas
  Soma
    ✓ Deve retornar 5 ao somar 2 e 3
    ✓ Deve retornar -1 ao somar -2 e 1
    ✓ Deve retornar 0 ao somar 0 e 0

3 passing (7ms)

PS C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS>
```

# INTERFACES DE USUÁRIO COM NODE.JS

## Desenvolvimento

### Configuração do Ambiente

```
PS C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS> mkdir validacao-cpf-node

Diretório: C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS

Mode                LastWriteTime         Length Name
----                -
d-----           06/09/2024    13:26         validacao-cpf-node

PS C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS>
```

### Criando pastas e arquivos

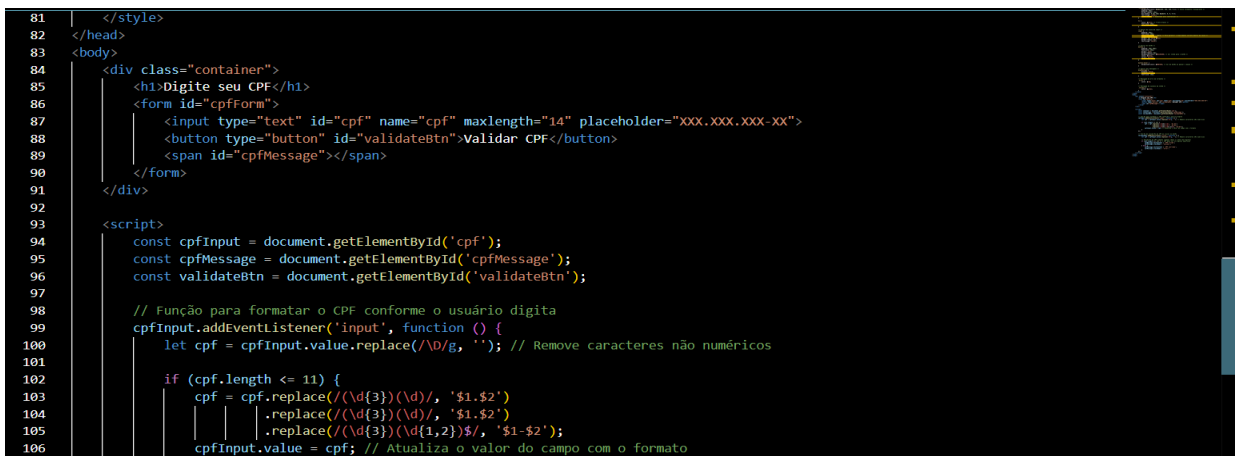
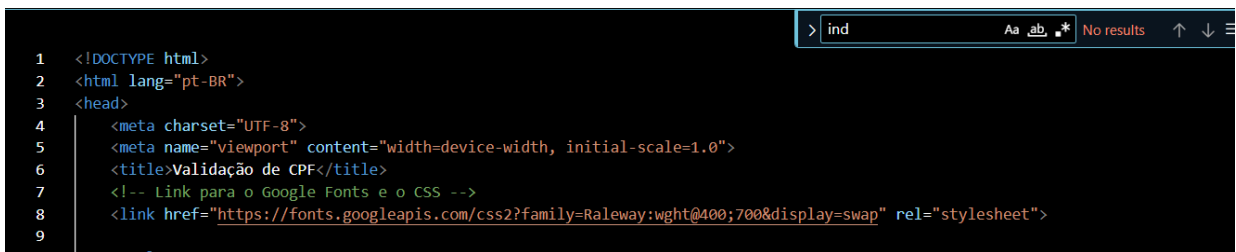
```
PS C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS\validacao-cpf-node> npm init -y
Wrote to C:\Users\Usuario\OneDrive\Área de Trabalho\Projeto_Servidor_Node.JS\validacao-cpf-node\package.json:

{
  "name": "validacao-cpf-node",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

### Estrutura do Projeto



## Criação do HTML



## Estilo CSS

```

1 <style>
2   /* Fundo preto */
3   body {
4     margin: 0;
5     height: 100vh;
6     display: flex;
7     justify-content: center;
8     align-items: center;
9     background-color: #000; /* Fundo preto */
10    font-family: 'Raleway', sans-serif; /* Fonte Raleway */
11  }
12
13  /* Caixa flutuante */
14  .container {
15    background-color: rgba(255, 255, 255, 0.1); /* Caixa levemente transparente */
16    padding: 30px;
17    border-radius: 10px;
18    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.3);
19    text-align: center;
20    width: 350px; /* Aumentado para centralizar */
21  }
22
23  h1 {
24    color: white; /* Título branco */
25    font-size: 24px;
26    margin-bottom: 20px;
27  }

```

```

/* Estilo da caixa de input */
input {
  padding: 10px;
  font-size: 16px;
  width: calc(100% - 22px); /* Para garantir o espaçamento correto dentro da caixa */
  margin-bottom: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
  text-align: center;
}

/* Estilo do botão */
button {
  padding: 10px 20px;
  font-size: 16px;
  border: none;
  border-radius: 5px;
  background-color: #01150100; /* Cor verde para o botão */
  color: white;
  cursor: pointer;
  margin-top: 10px;
}

button:hover {
  background-color: #01150100; /* Cor verde para o botão */
  color: white;
  cursor: pointer;
  margin-top: 10px;
}

button:hover {
  background-color: #90f494; /* Cor do botão ao passar o mouse */
}

/* Estilo das mensagens */
#cpfMessage {
  display: block;
  margin-top: 10px;
  font-size: 14px;
}

/* Mensagem de erro em vermelho */
.error {
  color: red;
}

/* Mensagem de sucesso em verde */
.success {
  color: green;
}
72 </style>

```

## Lógica de Validação em JavaScript

```

1 document.getElementById('cpf').addEventListener('input', function () {
2     const link = document.createElement('link');
3     link.rel = 'stylesheet';
4     link.href = '/public/cpf.css'; // Use um caminho acessível
5     document.head.appendChild(link); // Adiciona a folha de estilo ao DOM
6
7     const cpf = this.value;
8     const cpfMessage = document.getElementById('cpfMessage');
9
10    // Remove caracteres não numéricos
11    const cpfNumeros = cpf.replace(/\D/g, '');
12
13    // Formata o CPF apenas se tiver 11 números
14    if (cpfNumeros.length === 11) {
15        this.value = formatarCPF(cpfNumeros);
16
17        // Validação completa do CPF
18        if (validarCPF(cpfNumeros)) {
19            cpfMessage.textContent = 'CPF válido';
20            cpfMessage.className = 'success';
21        } else {

```

```

22            cpfMessage.textContent = 'CPF inválido';
23            cpfMessage.className = 'error';
24        }
25    } else {
26        cpfMessage.textContent = 'CPF inválido';
27        cpfMessage.className = 'error';
28    }
29 });
30
31 // Função para formatar o CPF
32 function formatarCPF(cpf) {
33     return cpf.replace(/(\d{3})(\d{3})(\d{3})(\d{2})/, '$1.$2.$3-$4');
34 }
35
36 // Função para validar o CPF (simplificada)
37 function validarCPF(cpf) {
38     let soma = 0;
39     let resto;
40
41     if (cpf == "00000000000") return false;
42

```

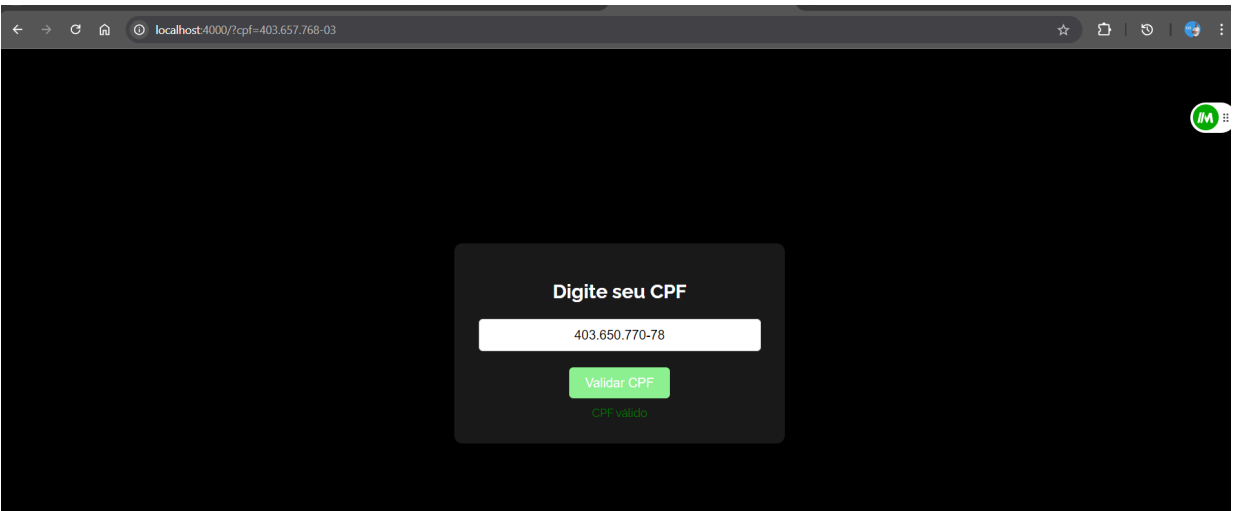
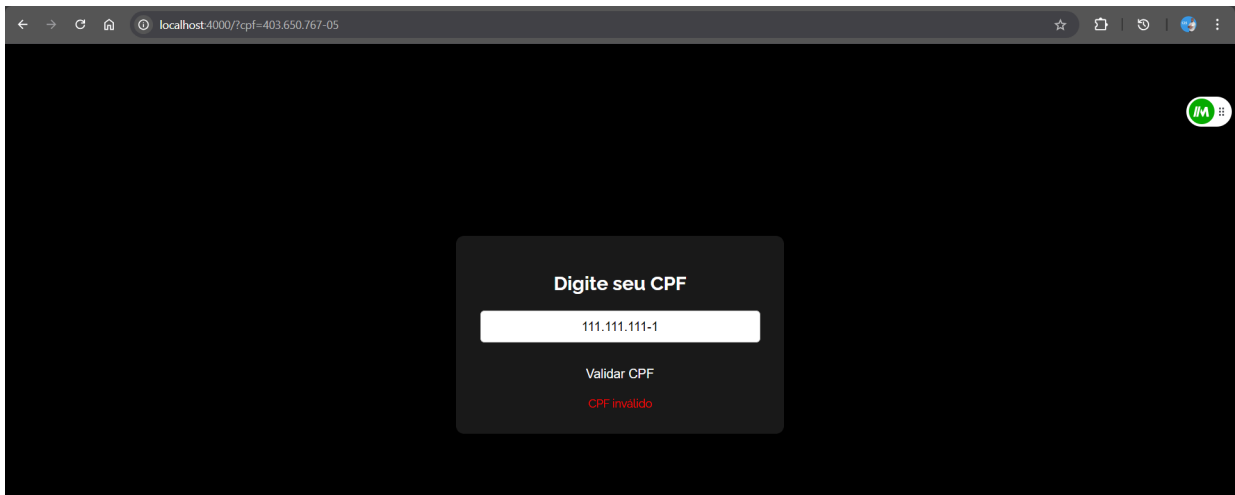
```

43     for (let i = 1; i <= 9; i++) soma = soma + parseInt(cpf.substring(i-1, i)) * (11 - i);
44     resto = (soma * 10) % 11;
45
46     if ((resto == 10) || (resto == 11)) resto = 0;
47     if (resto != parseInt(cpf.substring(9, 10))) return false;
48
49     soma = 0;
50     for (let i = 1; i <= 10; i++) soma = soma + parseInt(cpf.substring(i-1, i)) * (12 - i);
51     resto = (soma * 10) % 11;
52
53     if ((resto == 10) || (resto == 11)) resto = 0;
54     if (resto != parseInt(cpf.substring(10, 11))) return false;
55
56     return true;
57 }
58

```



# Teste de Validação de CPF



# ESTRATÉGIAS DE TESTES

## Desenvolvimento

### Configuração do projeto e instalação de dependências

Criando pastas e arquivos

```
Diretório: C:\Users\Usuario\OneDrive\Área de Trabalho\Testes

Mode                LastWriteTime         Length Name
----                -
d-----          13/09/2024   15:23             servidor-http-simples

PS C:\Users\Usuario\OneDrive\Área de Trabalho\Testes>
```

```
found 2 vulnerabilities
PS C:\Users\Usuario\OneDrive\Área de Trabalho\Testes\servidor-http-simples> mkdir test
>>

Diretório: C:\Users\Usuario\OneDrive\Área de Trabalho\Testes\servidor-http-simples

Mode                LastWriteTime         Length Name
----                -
d-----          13/09/2024   15:29             test

PS C:\Users\Usuario\OneDrive\Área de Trabalho\Testes\servidor-http-simples>
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Wrote to C:\Users\Usuario\OneDrive\Área de Trabalho\Testes\servidor-http-simples\package.json:

{
  "name": "servidor-http-simples",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

## Criação do Servidor HTTP

```
servidor-http-simples > JS server.js > ...
1  const express = require('express');
2  const app = express();
3  const port = 3000;
4
5  app.use(express.json()); // Middleware para parsear JSON
6
7  // Rota GET para "/"
8  app.get('/', (req, res) => {
9    res.send('Hello World');
10 });
11
12 // Rota POST para "/data"
13 app.post('/data', (req, res) => {
14   const data = req.body;
15   res.json({
16     message: 'Success',
17     data: data
18   });
19 });
20
21 // Inicializando o servidor
22 app.listen(port, () => {
23   console.log(`Servidor rodando na porta ${port}`);
24 });
25
26 module.exports = app; // Exporta o app para uso nos testes
27
```

## Estrutura do Projeto

```
JS server.js JS math.js X JS integration.test.mjs {} package.json

servidor-http-simples > JS math.js > ...
1  // Arquivo math.js
2  const add = (a, b) => a + b;
3  const subtract = (a, b) => a - b;
4
5  module.exports = { add, subtract };
6
```

## Testes de Integração

```
servidor-http-simples > test > JS integration.test.mjs > ...
1  import chai from 'chai';
2  import chaiHttp from 'chai-http/server.js';
3  import server from '../server.js';
4
5  const { expect } = chai;
6  chai.use(chaiHttp); // Integrando chai-http ao chai
7
8  describe('Testes de Integração do Servidor HTTP', () => {
9    it('Deve retornar "Hello World" na rota GET /', (done) => {
10      chai.request(server)
11        .get('/')
12        .end((err, res) => {
13          expect(res).to.have.status(200);
14          expect(res.text).to.equal('Hello World');
15          done();
16        });
17    });
18  });
```

```
19  it('Deve retornar um JSON com mensagem de sucesso na rota POST /data', (done) => {
20    const inputData = { name: 'Test', age: 30 };
21    chai.request(server)
22      .post('/data')
23      .send(inputData)
24      .end((err, res) => {
25        expect(res).to.have.status(200);
26        expect(res.body).to.be.an('object');
27        expect(res.body).to.have.property('message').equal('Success');
28        expect(res.body).to.have.property('data').deep.equal(inputData);
29        done();
30      });
31  });
32  });
33  });
```

## Executar os Testes

```
servidor-http-simples > {} package.json > {} devDependencies
1  {
2    "name": "servidor-http-simples",
3    "version": "1.0.0",
4    "main": "server.js",
5    "scripts": {
6      "test": "mocha"
7    },
8    "keywords": [],
9    "author": "",
10   "license": "ISC",
11   "description": "Servidor HTTP simples com Node.js, Express e testes com Mocha/Chai",
12   "dependencies": {
13     "express": "^4.21.0"
14   },
15   "devDependencies": {
16     "chai": "^4.3.7",
17     "chai-http": "^5.0.0",
18     "mocha": "^10.7.3"
19   }
20 }
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
node - servidor-http-simples + - [] ... ^ x

Servidor rodando na porta 3000

Testes de Integração do Servidor HTTP
1) Deve retornar "Hello World" na rota GET /
2) Deve retornar um JSON com mensagem de sucesso na rota POST /data

0 passing (5ms)
2 failing

1) Testes de Integração do Servidor HTTP
  Deve retornar "Hello World" na rota GET /:
  TypeError: chai.request is not a function
1) Testes de Integração do Servidor HTTP
  Deve retornar "Hello World" na rota GET /:
  TypeError: chai.request is not a function
  Deve retornar "Hello World" na rota GET /:
  TypeError: chai.request is not a function
```

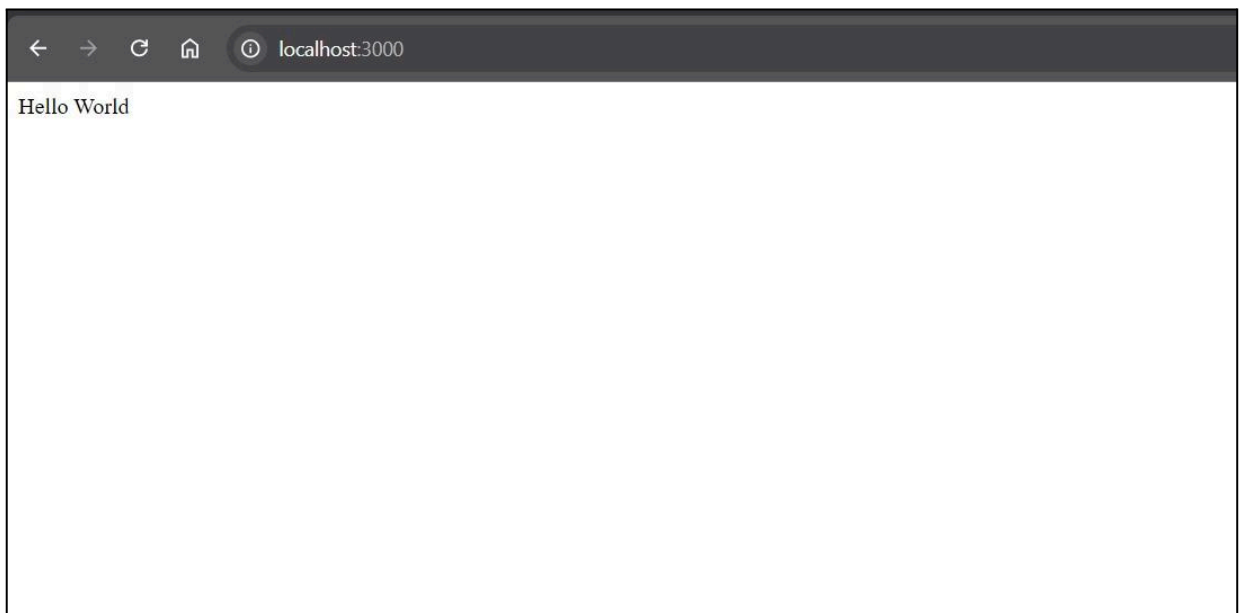
```
at Context.<anonymous> (file:///C:/Users/Usuario/OneDrive/%C3%81rea%20de%20Trabalho/Testes/servidor-http-simples/test/integration.test.mjs:10:10)
at process.processImmediate (node:internal/timers:483:21)

2) Testes de Integração do Servidor HTTP
  at Context.<anonymous> (file:///C:/Users/Usuario/OneDrive/%C3%81rea%20de%20Trabalho/Testes/servidor-http-simples/test/integration.test.mjs:10:10)
  at process.processImmediate (node:internal/timers:483:21)

2) Testes de Integração do Servidor HTTP
  Deve retornar um JSON com mensagem de sucesso na rota POST /data:
  at process.processImmediate (node:internal/timers:483:21)

2) Testes de Integração do Servidor HTTP
  Deve retornar um JSON com mensagem de sucesso na rota POST /data:
  TypeError: chai.request is not a function
```

Infelizmente me deparei com alguns erros que eu não consegui solucionar, mas ainda assim eu alcancei um resultado positivo



## **CONSIDERAÇÕES FINAIS**

Nesse portfólio que foi solicitado pelo Tutor, consegui desenvolver um trabalho que me trouxe mais conhecimento sobre a matéria, durante o desenvolvimento do trabalho tive algumas dificuldades, mas com o conhecimento apresentado nas aulas e em outros cursos que venho estudando, pude desenvolver e finalizar o portfólio.

## REFERÊNCIAS

Aula de: **Desenvolvimento Com Framework Para Node.js**

Tutores: Elisa Antolli Paleari, Gian Carlo Decarli

Conteúdo fornecido por Faculdade Anhanguera

### **Curso de Node.JS**

Tutor: Fessor Bruno

Conteúdo: [https://www.youtube.com/watch?v=XN705pQeoyU&list=PLx4x\\_zx8csUjFC41ev2qX5dnr-0ThpoXE](https://www.youtube.com/watch?v=XN705pQeoyU&list=PLx4x_zx8csUjFC41ev2qX5dnr-0ThpoXE)