

PORTFÓLIO DE DESENVOLVIMENTO COM FRAMEWORK PARA NODE.JS

5ºSEMESTRE

DATA: 08/10/2024

PROFESSORES: Elisa Antolli Paleari e Gian Carlo Decarli

NOME DO ALUNO: Fernando Henrique Panini

ATIVIDADE - I

Criação de um servidor básico, HTTP, para 4 situações:

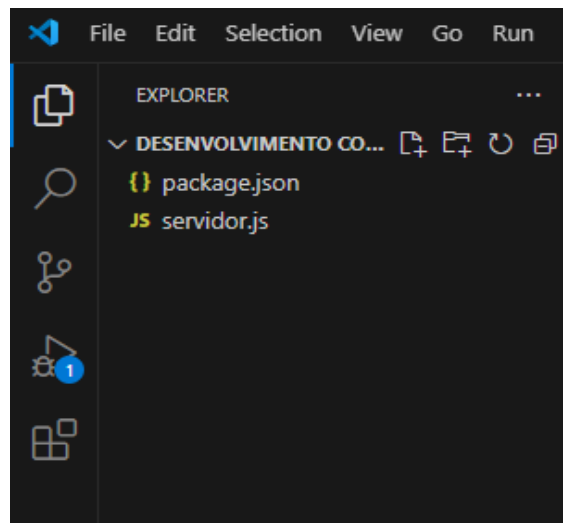
localhost:3000

localhost:3000/sobre

localhost:3000/contato

E uma resposta de 'Página não encontrada' para demais casos.

Arquivos Criados:



Códigos do Arquivo Servidor.js:

```
JS servidor.js
JS servidor.js > ...
1  const http = require('http');
2
3  const hostname = 'localhost';
4  const port = 3000;
5
6  const server = http.createServer((req, res) => {
7    if (req.url === '/') {
8      res.statusCode = 200;
9      res.setHeader('Content-Type', 'text/plain');
10     res.end('Bem vindo a pagina inicial!');
11   } else if (req.url === '/sobre') {
12     res.statusCode = 200;
13     res.setHeader('Content-Type', 'text/plain');
14     res.end('Esta e a pagina sobre nos.');
```

```
15   } else if (req.url === '/contato') {
16     res.statusCode = 200;
17     res.setHeader('Content-Type', 'text/plain');
18     res.end('Esta e a pagina de contato.');
```

```
19   } else {
20     res.statusCode = 404;
21     res.setHeader('Content-Type', 'text/plain');
22     res.end('Pagina nao encontrada');
```

```
23   }
24 });
25
26 server.listen(port, hostname, () => {
27   console.log(`Servidor rodando em http://${hostname}:${port}/`);
28 });
```

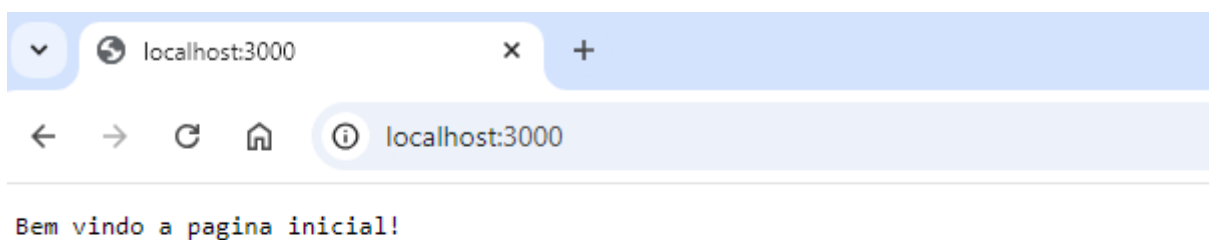
Servidor Funcionando:

```
JS servidor.js X [Icons]
JS servidor.js > [Icon] server > http.createServer() callback
1  const http = require('http');
2
3  const hostname = 'localhost';
4  const port = 3000;
5
6  const server = http.createServer((req, res) => {
7    if (req.url === '/') {
8      res.statusCode = 200;
9      res.setHeader('Content-Type', 'text/plain');
10     res.end('Bem vindo a pagina inicial!');
11   } else if (req.url === '/sobre') {
12     res.statusCode = 200;
13     res.setHeader('Content-Type', 'text/plain');
14     res.end('Esta e a pagina sobre nos.');
15   } else if (req.url === '/contato') {
16     res.statusCode = 200;
17     res.setHeader('Content-Type', 'text/plain');
18     res.end('Esta e a pagina de contato.');
19   } else {
20     res.statusCode = 404;
  
```

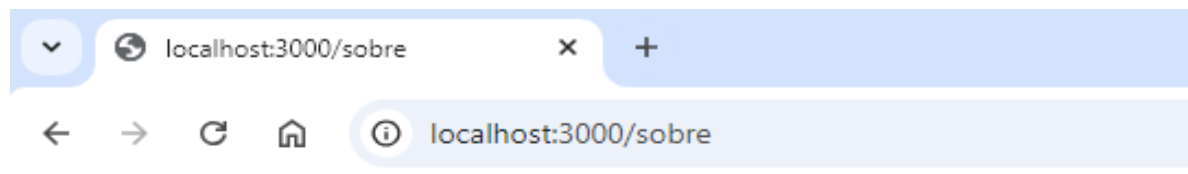
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Filter (e.g. t

```
C:\Program Files\nodejs\node.exe .\servidor.js
Servidor rodando em http://localhost:3000/
```

http://localhost:3000

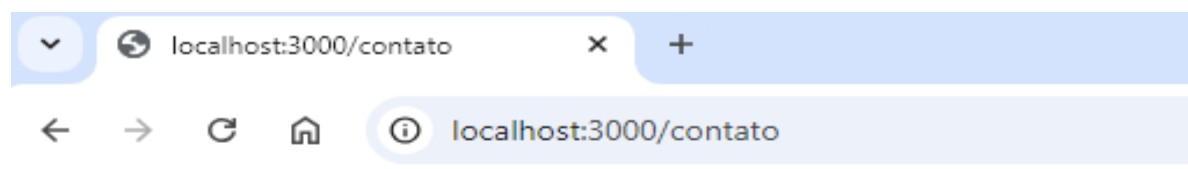


http://localhost:3000/sobre



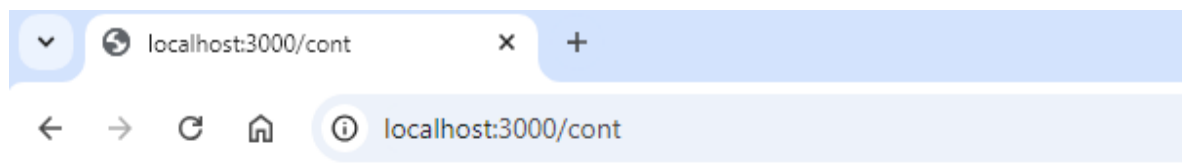
Esta e a pagina sobre nos.

http://localhost:3000/contato



Esta e a pagina de contato.

http://localhost:3000/cont (Outra Qualquer)

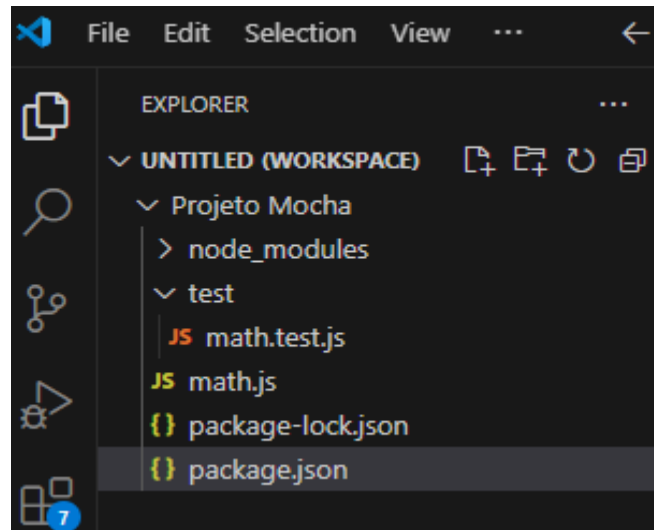


Pagina nao encontrada

ATIVIDADE – II

Criação de projeto de teste simples usando mocha para uma aplicação Node.js, utilizando uma função básica de soma.

Arquivos Criados:



Códigos do Arquivo math.js:

```
JS math.js  x  JS math.test.js  {} package.json
Projeto Mocha > JS math.js > ...
1  // math.js
2  function soma(a, b) {
3    |   return a + b;
4  }
5
6  module.exports = { soma };
7  |
```

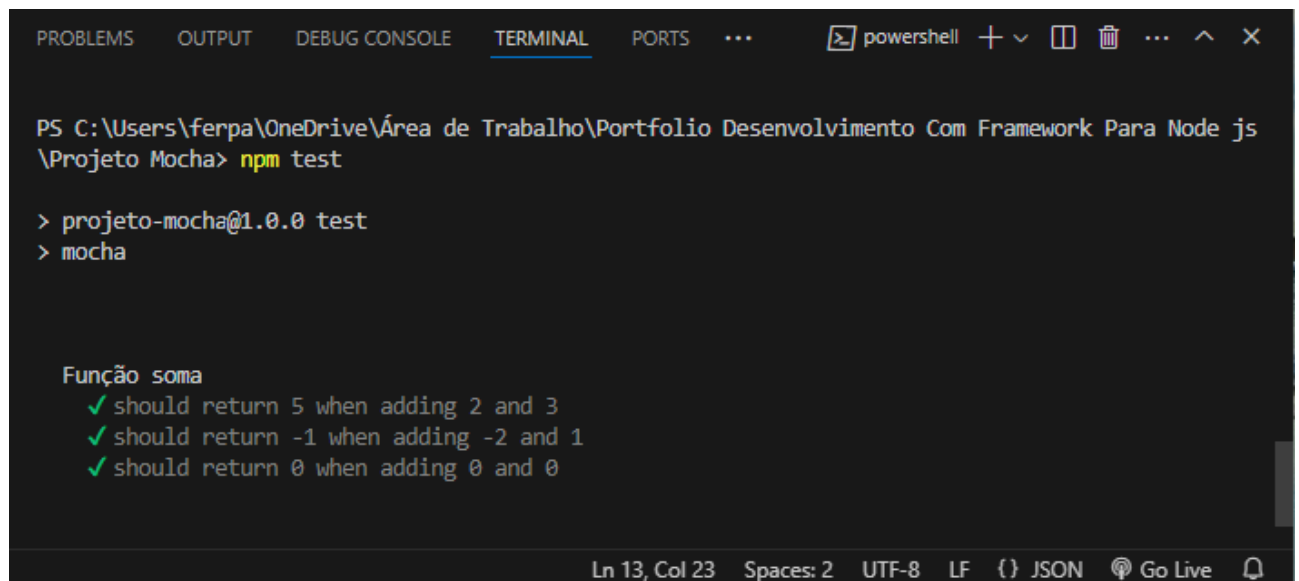
Códigos do Arquivo math.test.js:

```
JS math.js    JS math.test.js X    {} package.json
Projeto Mocha > test > JS math.test.js > ...
1  // test/math.test.js
2  const { soma } = require('../math');
3
4  describe('Função soma', () => {
5    it('should return 5 when adding 2 and 3', () => {
6      const resultado = soma(2, 3);
7      if (resultado !== 5) throw new Error(`esperado 5, mas obtido ${resultado}`);
8    });
9
10   it('should return -1 when adding -2 and 1', () => {
11     const resultado = soma(-2, 1);
12     if (resultado !== -1) throw new Error(`esperado -1, mas obtido ${resultado}`);
13   });
14
15   it('should return 0 when adding 0 and 0', () => {
16     const resultado = soma(0, 0);
17     if (resultado !== 0) throw new Error(`esperado 0, mas obtido ${resultado}`);
18   });
19 });
20
```

Script no Arquivo package.json:

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

Resultado dos Testes:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ... powershell + - ... ^ X

PS C:\Users\ferpa\OneDrive\Área de Trabalho\Portfolio Desenvolvimento Com Framework Para Node js
\Projeto Mocha> npm test

> projeto-mocha@1.0.0 test
> mocha

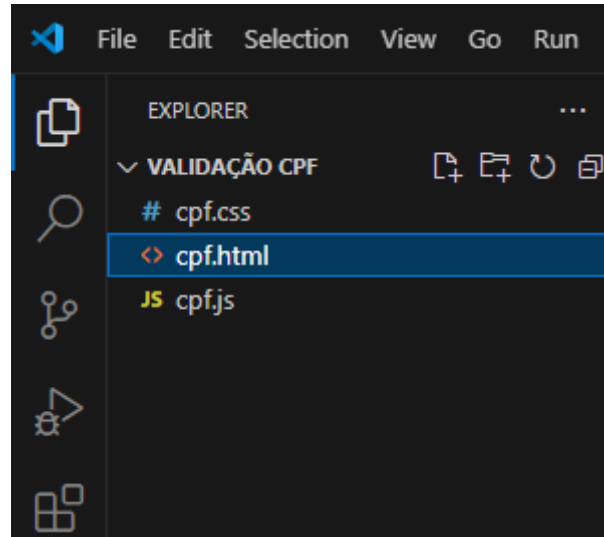
Função soma
  ✓ should return 5 when adding 2 and 3
  ✓ should return -1 when adding -2 and 1
  ✓ should return 0 when adding 0 and 0
```

Ln 13, Col 23 Spaces: 2 UTF-8 LF {} JSON Go Live

ATIVIDADE - III

Criação de projeto de aplicação de validação de campo CPF (HTML, CSS e Javascript), com a mensagem se o CPF digitado é válido ou inválido.

Arquivos Criados:



Códigos em HTML:

```
< CPF.html  X  # cpf.css  JS cpf.js

< CPF.html > html > body > div.container
1  <!DOCTYPE html>
2  <html lang="pt-BR">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <link rel="stylesheet" href="cpf.css">
7      <title>Validação de CPF</title>
8  </head>
9  <body>
10     <div class="container">
11         <h1>Validação de CPF</h1>
12         <form id="cpfForm">
13             <label for="cpf">Digite seu CPF:</label>
14             <input type="text" id="cpf" placeholder="000.000.000-00" required>
15             <button type="submit">Validar</button>
16         </form>
17         <p id="message" class="message"></p>
18     </div>
19     <script src="cpf.js"></script>
20 </body>
21 </html>
22
```

Códigos em CSS:

```
<> cpf.html # cpf.css X JS cpf.js
# cpf.css > ...
1  body {
2    font-family: Arial, sans-serif;
3    background-color: #f4f4f4;
4    padding: 20px;
5  }
6
7  .container {
8    max-width: 400px;
9    margin: 0 auto;
10   background: #fff;
11   padding: 20px;
12   border-radius: 5px;
13   box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
14 }
15
16 h1 {
17   text-align: center;
18 }
19
20 .message {
21   margin-top: 15px;
22   font-weight: bold;
23 }
24
25 .success {
26   color: green;
27 }
28
29 .error {
30   color: red;
31 }
32
```

Códigos em Javascript:

```
JS cpfjs > ...
1 document.getElementById('cpfForm').addEventListener('submit', function(event) {
2     event.preventDefault();
3
4     const cpfInput = document.getElementById('cpf').value;
5     const messageElement = document.getElementById('message');
6
7     if (validarCPF(cpfInput)) {
8         messageElement.textContent = 'CPF Válido!';
9         messageElement.className = 'message success';
10    } else {
11        messageElement.textContent = 'CPF Inválido!';
12        messageElement.className = 'message error';
13    }
14 });
15
16 function validarCPF(cpf) {
17
18     cpf = cpf.replace(/[^\d]+/g, '');
19
20     // CPF deve ter 11 dígitos
21     if (cpf.length !== 11 || /^(\\d)\\1{10}$/.test(cpf)) {
22         return false;
23     }
24
25     let soma = 0;
26     let resto;
27
28     // Validação do primeiro dígito
29     for (let i = 1; i <= 9; i++) {
30         soma += parseInt(cpf.charAt(i - 1)) * (11 - i);
31     }
32     resto = (soma * 10) % 11;
33     if (resto === 10 || resto === 11) resto = 0;
34     if (resto !== parseInt(cpf.charAt(9))) return false;
35
36     soma = 0;
37
38     // Validação do segundo dígito
39     for (let i = 1; i <= 10; i++) {
40         soma += parseInt(cpf.charAt(i - 1)) * (12 - i);
41     }
42     resto = (soma * 10) % 11;
43     if (resto === 10 || resto === 11) resto = 0;
44     if (resto !== parseInt(cpf.charAt(10))) return false;
45
46     return true;
47 }
48
```

Resultados:

CPF Válido:



← → ↻ 🏠 ⓘ Arquivo C:/Users/fernando.panini.HAPVIDA/Downloads/Desenvolvimento%20Com%20Framework%20Para%20Node%20js/Validação%20CPF/cpf.html ☆ ⬇️ B ⋮

Validação de CPF

Digite seu CPF:

CPF Válido!

CPF Inválido:



← → ↻ 🏠 ⓘ Arquivo C:/Users/fernando.panini.HAPVIDA/Downloads/Desenvolvimento%20Com%20Framework%20Para%20Node%20js/Validação%20CPF/cpf.html ☆ ⬇️ B ⋮

Validação de CPF

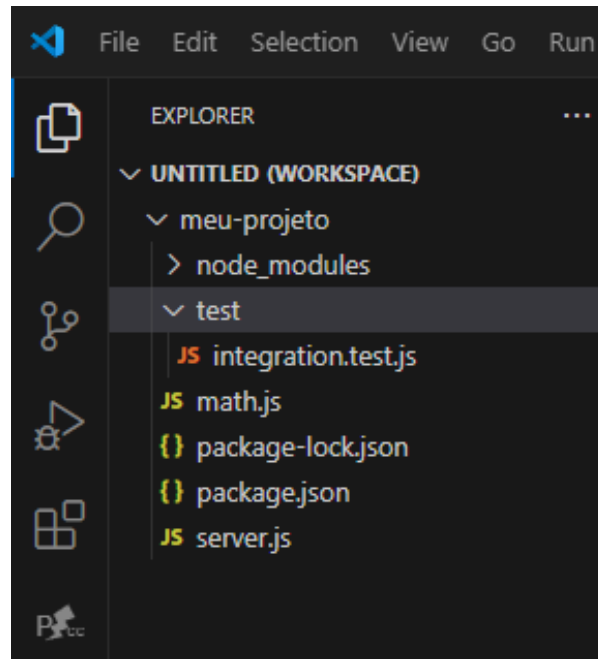
Digite seu CPF:

CPF Inválido!

ATIVIDADE - IV

Criação de um servidor HTTP simples usando Node.js e escrever testes de integração para validar o funcionamento das rotas desse servidor, usando a biblioteca Mocha para escrever os testes e a biblioteca Chai para asserções.

Arquivos Criados:



Códigos Arquivo server.js:

```
JS server.js x
meu-projeto > JS server.js > ...
1 // server.js
2 import express from 'express';
3
4 const app = express();
5 const PORT = 3000;
6
7 app.use(express.json());
8
9 app.get('/', (req, res) => {
10   res.send('Hello World');
11 });
12
13 app.post('/data', (req, res) => {
14   const data = req.body;
15   res.json({ message: 'Success', data: data });
16 });
17
18 app.listen(PORT, () => {
19   console.log(`Server running on port ${PORT}`);
20 });
21
22 export default app;
```

Códigos Arquivo math.js:

```
JS math.js X
meu-projeto > JS math.js > ...
1 // math.js
2 export function add(a, b) {
3   return a + b;
4 }
5
6 export function subtract(a, b) {
7   return a - b;
8 }
9
```

Códigos Arquivo integration.test.js:

```
JS integration.test.js X
meu-projeto > test > JS integration.test.js > ...
1 // test/integration.test.js
2 import { request } from 'express';
3 const chai = import('chai');
4 const chaiHttp = import('chai-http');
5 const app = import ('../server.js'); // Usando import para trazer o app
6
7 const { expect } = chai; // Desestruturando para usar 'expect'
8
9 describe('Server Routes', () => {
10   it('should GET / and return Hello World', (done) => {
11     chai.request(app)
12       .get('/')
13       .end((err, res) => {
14         expect(res).to.have.status(200);
15         expect(res.text).to.equal('Hello World');
16         done();
17       });
18   });
19
20   it('should POST /data and return success message with data', (done) => {
21     const data = { name: 'Test' };
22     chai.request(app)
23       .post('/data')
24       .send(data)
25       .end((err, res) => {
26         expect(res).to.have.status(200);
27         expect(res.body).to.be.a('object');
28         expect(res.body).to.have.property('message').eql('Success');
29         expect(res.body).to.have.property('data').eql(data);
30         done();
31       });
32   });
33 });
34
```

Script no Arquivo package.json:

```
{ package.json X
meu-projeto > {} package.json > ...
1  {
2    "name": "meu-projeto",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "type": "module",
7    "scripts": {
8      "test": "mocha"
9    },
10   "keywords": [],
11   "author": "",
12   "license": "ISC",
13   "dependencies": {
14     "chai": "^5.1.1",
15     "chai-http": "^5.0.0",
16     "express": "^4.21.0",
17     "mocha": "^10.7.3"
18   }
19 }
20
```

Resultado dos Testes:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POLYGLOT NOTEBOOK

Server Routes
Server running on port 3000
  1) should GET / and return Hello World
  2) should POST /data and return success message with data

0 passing (24ms)
2 failing

1) Server Routes
   should GET / and return Hello World:
     TypeError: chai.request is not a function
       at Context.<anonymous> (file:///C:/Users/ferpa/OneDrive/%C3%81rea%
ojeto/test/integration.test.js:11:14)
       at process.processImmediate (node:internal/timers:478:21)

2) Server Routes
   should POST /data and return success message with data:
     TypeError: chai.request is not a function
       at Context.<anonymous> (file:///C:/Users/ferpa/OneDrive/%C3%81rea%
ojeto/test/integration.test.js:22:17)
       at process.processImmediate (node:internal/timers:478:21)
```