

CENTRO UNIVERSITÁRIO ANHANGUERA PITÁGORAS AMPLI  
THIAGO PEREIRA DOS SANTOS

PROGRAMAÇÃO WEB

GUARULHOS  
2024

THIAGO PEREIRA DOS SANTOS

PROGRAMAÇÃO WEB

TRABALHO DE CONCLUSÃO DA DISCIPLINA PROGRAMAÇÃO WEB

## SUMÁRIO

<b>1 INTRODUÇÃO.....</b>	<b>5</b>
<b>2 AULA PRÁTICA 1: HTML E CSS.....</b>	<b>5</b>
<b>2.1 ESTRUTURA HTML.....</b>	<b>5</b>
<b>2.2 APLICANDO ESTILO COM CSS.....</b>	<b>6</b>
<b>2.3 CÓDIGOS.....</b>	<b>7</b>
2.3.1 Arquivo style.css.....	7
2.3.2 Arquivo index.html.....	8
<b>3 AULA PRÁTICA 2: HTML, CSS E JAVASCRIPT.....</b>	<b>9</b>
<b>3.1 ESTRUTURA HTML.....</b>	<b>9</b>
<b>3.2 JAVASCRIPT.....</b>	<b>10</b>
<b>3.3 CÓDIGOS.....</b>	<b>10</b>
3.3.1 Index.html.....	10
3.3.2 Calcular.js.....	11
<b>4 MODULARIZANDO O JAVASCRIPT DA CALCULADORA.....</b>	<b>13</b>
<b>4.1 CÓDIGOS.....</b>	<b>13</b>
4.1.1 Calculadora.html.....	13
4.1.2 Calcular.js.....	14
4.1.3 Operacoes.js.....	15
<b>5 CALCULADORA COM NODE.JS.....</b>	<b>16</b>
<b>5.1 CÓDIGO.....</b>	<b>17</b>
5.1.1 somar.js.....	17
5.1.2 subtrair.js.....	17
5.1.3 dividir.js.....	17

5.1.4 multiplicar.js.....	18
5.1.5 app.js.....	18
<b>6 CALCULADORA PHP.....</b>	<b>19</b>
<b>6.1 RESULTADO.....</b>	<b>19</b>
<b>6.2 CÓDIGOS.....</b>	<b>19</b>
6.2.1 index.html.....	19
6.2.2 calcular.js.....	20
6.3.3 operações.js.....	21
6.3.4 calcula.php.....	23
<b>7 SALVANDO TEXTO EM ARQUIVO COM PHP.....</b>	<b>26</b>
<b>7.1 RESULTADO.....</b>	<b>26</b>
<b>7.2 CÓDIGOS.....</b>	<b>27</b>
7.2.1 index.html.....	27
7.2.2 write.php.....	28
<b>8 CADASTRO DE USUÁRIOS COM PHP.....</b>	<b>28</b>
<b>8.1 CRIAÇÃO DO BANCO DE DADOS.....</b>	<b>29</b>
<b>8.2 RESULTADO.....</b>	<b>29</b>
<b>8.3 CÓDIGO.....</b>	<b>30</b>
8.3.1 index.php.....	30
8.3.2 style.css.....	32
8.3.3 database.sql.....	34
<b>9 CONCLUSÃO.....</b>	<b>34</b>

# 1 INTRODUÇÃO

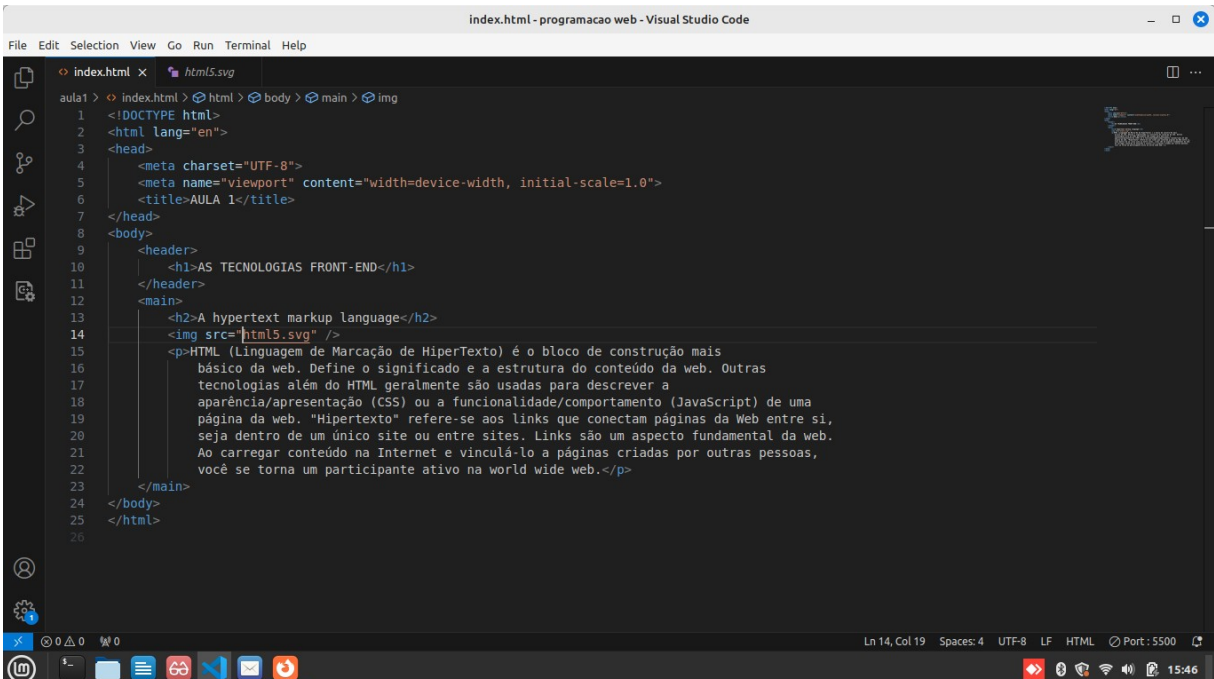
Roteiro de aula prática, com o objetivo de fixar o aprendizado na área de programação Web.

## 2 AULA PRÁTICA 1: HTML E CSS

O projeto proposto foi criar uma interface Web e estilizá-la.

### 2.1 ESTRUTURA HTML

Com o editor Visual Studio Code, criamos o seguinte código HTML.



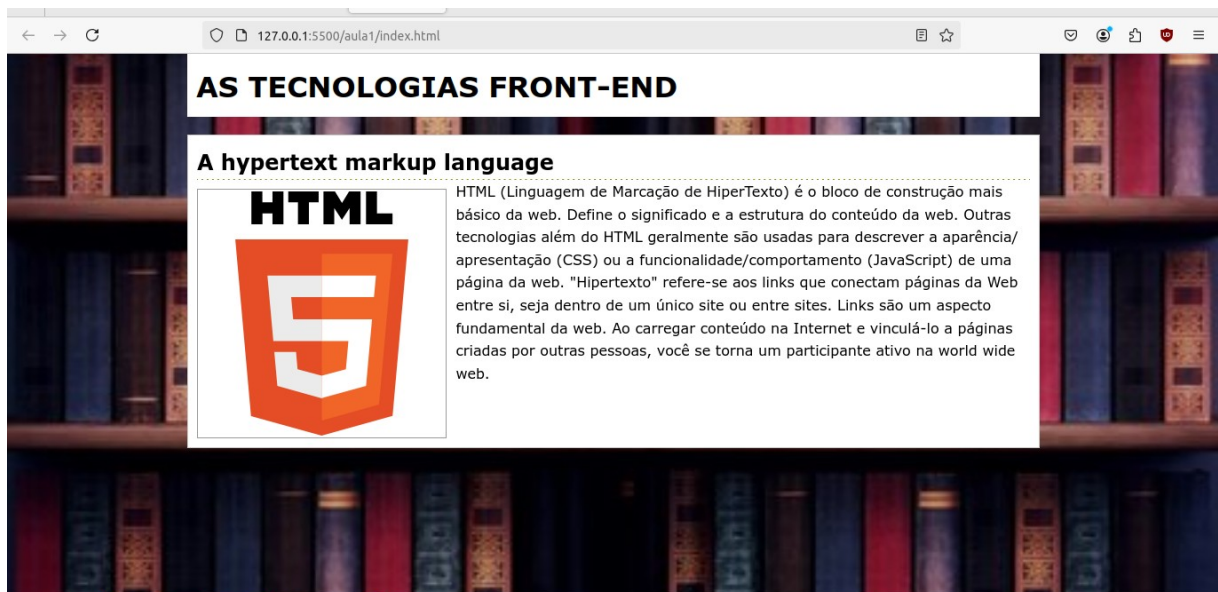
```
index.html - programacao web - Visual Studio Code
File Edit Selection View Go Run Terminal Help
index.html x html5.svg
aula1 > index.html > html > body > main > img
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>AULA 1</title>
7 </head>
8 <body>
9 <header>
10 <h1>AS TECNOLOGIAS FRONT-END</h1>
11 </header>
12 <main>
13 <h2>A hypertext markup language</h2>
14 
15 <p>HTML (Linguagem de Marcação de HiperTexto) é o bloco de construção mais
16 básico da web. Define o significado e a estrutura do conteúdo da web. Outras
17 tecnologias além do HTML geralmente são usadas para descrever a
18 aparência/apresentação (CSS) ou a funcionalidade/comportamento (JavaScript) de uma
19 página da web. "HiperTexto" refere-se aos links que conectam páginas da Web entre si,
20 seja dentro de um único site ou entre sites. Links são um aspecto fundamental da web.
21 Ao carregar conteúdo na Internet e vinculá-lo a páginas criadas por outras pessoas,
22 você se torna um participante ativo na world wide web.</p>
23 </main>
24 </body>
25 </html>
26
Ln 14, Col 19 Spaces: 4 UTF-8 LF HTML Port: 5500
15:46
```

E com este código obtivemos o seguinte resultado:



## 2.2 APLICANDO ESTILO COM CSS.

Após inserimos uma folha de estilos CSS, obtivemos o seguinte resultado



Para a tela ficar conforme o problema proposto, tivemos que executar algumas ações a mais. Como esta imagem que eu utilizei é um svg, ele não tem limite de largura, e ocupou a tela inteira. Então adicionamos a propriedade “*width: 30%;*” na imagem, assim ela ocupa apenas 30% do container. E devido estarmos utilizando a propriedade “float:left”, o navegador terminou de renderizar o container antes do texto do parágrafo, deixando o texto pra fora. Para resolver isso, foi criada uma div vazia com a classe “clear” e adicionamos a seguinte propriedade à classe: “clear:both”.

## 2.3 CÓDIGOS

Segue abaixo os códigos do projeto em questão.

### 2.3.1 ARQUIVO STYLE.CSS

```
*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

.container{
  margin: 0 auto;
  width:960px;
}

body{
  font-family: Verdana, Geneva, Tahoma, sans-serif;
  line-height: 1.6;
  background: url(background.jpg) no-repeat center center fixed;
  background-size: cover;
}

h1{
  background-color: #fff;
  padding: 10px;
}
```

```
h2{
  border-bottom: 1px dashed #7e9719;
}

.bloco{
  background-color: #fff;
  border-color: #ccc;
  border: 1px solid #ccc;
  margin-top: 20px;
  padding:10px
}

img{
  border: 1px solid #999;
  float: left;
  margin: 10px 10px 0 0;
  padding: 2px;
  width: 30%;
}

.clear{
  clear: both;
}
```

### 2.3.2 ARQUIVO INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="style.css">
  <title>AULA 1</title>
</head>
<body>
```



```
<div class="container">
  <h1>AS TECNOLOGIAS FRONT-END</h1>

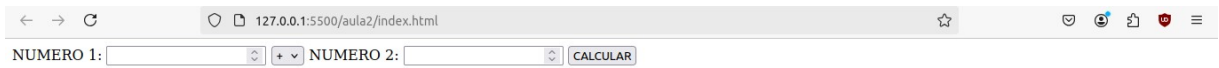
  <div class="bloco">
    <h2>A hypertext markup language</h2>
    
    <p>HTML (Linguagem de Marcação de HiperTexto) é o bloco de construção mais
      básico da web. Define o significado e a estrutura do conteúdo da web. Outras
      tecnologias além do HTML geralmente são usadas para descrever a
      aparência/apresentação (CSS) ou a funcionalidade/comportamento (JavaScript) de
      uma
      página da web. "Hipertexto" refere-se aos links que conectam páginas da Web
      entre si,
      seja dentro de um único site ou entre sites. Links são um aspecto fundamental
      da web.
      Ao carregar conteúdo na Internet e vinculá-lo a páginas criadas por outras
      pessoas,
      você se torna um participante ativo na world wide web.</p>
    <div class="clear"></div>
  </div>
</body>
</html>
```

### 3 AULA PRÁTICA 2: HTML, CSS E JAVASCRIPT

O projeto proposto foi criar uma interface HTML, no qual a partir do Javascript, se pudesse realizar as funções de uma calculadora.

#### 3.1 ESTRUTURA HTML

A partir do código HTML, obtivemos o seguinte resultado.



Por questões de sintaxe, a *div* que vai o resultado não atribuímos a propriedade *name*, pois a mesma é inválida. A substituímos pelo atributo *class*.

## 3.2 JAVASCRIPT

Foi criado um código Javascript, o qual verifica se há números válidos, e se sim realiza a operação. Se a operação selecionada for divisão, o código verificará se o divisor não é zero, e se não, realizará a divisão.

## 3.3 CÓDIGOS

Segue abaixo os códigos do problema proposto.

### 3.3.1 INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```

<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<script src="calcular.js"></script>
<title>aula 2</title>
</head>
<body>
  <form id="calculadora">
    <label for="numero1">NUMERO 1: <input type="number" id="numero1" /></label>
    <select id="operacao">
      <option value="+">+</option>
      <option value="-">-</option>
      <option value="/">/</option>
      <option value="*">*</option>
    </select>
    <label for="numero2">NUMERO 2: <input type="number" id="numero2" /></label>
    <button type="submit">CALCULAR</button>
  </form>
  <div class="resultado"></div>
</body>
</html>

```

### 3.3.2 CALCULAR.JS

```

window.onload = () =>{
  let campo_numero1 = document.getElementById("numero1")
  let campo_numero2 = document.getElementById("numero2")
  let select_operacao = document.getElementById("operacao")
  let calculadora = document.getElementById('calculadora')

  const calcular = () =>{
    if(campo_numero1.value == null || campo_numero1.value == '' || campo_numero2.value
    == null || campo_numero2.value == '')
    {
      alert("Digite um valor válido")
      return
    }
  }
}

```

```
}  
let numero1 = parseFloat(campo_numero1.value)  
let numero2 = parseFloat(campo_numero2.value)  
let operacao = select_operacao.value  
let resultado_container = document.querySelector(".resultado")  
let resultado = 0  
  
switch(operacao){  
  case '*':  
    resultado = numero1 * numero2  
    break  
  case '+':  
    resultado = numero1 + numero2  
    break  
  case '-':  
    resultado = numero1 - numero2  
    break  
  case '/':  
    if(numero2 === 0){  
      alert("Não é possível divisão por zero")  
      return  
    }  
    resultado = numero1 / numero2  
    break  
  default:  
    alert("Selecione uma operação válida")  
    return  
}  
resultado_container.innerHTML = `0 resultado é ${resultado.toFixed(2)}`  
  
}  
  
calculadora.addEventListener("submit", (e) => {  
  e.preventDefault()  
  calcular()  
})  
}
```

## 4 MODULARIZANDO O JAVASCRIPT DA CALCULADORA

Foi proposto que se repetisse a atividade anterior, entretanto, alterando o projeto, para que as funções de operações matemáticas ficassem separadas em um arquivo. Foi criado o arquivo Javascript e importado no documento HTML, e foi inserido um bloco *try...catch* para processar as divisões por zero.

### 4.1 CÓDIGOS

Segue os códigos da aula prática

#### 4.1.1 CALCULADORA.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <script src="operacoes.js"></script>
  <script src="calcular.js"></script>
  <title>aula 3</title>
</head>
<body>
  <form id="calculadora">
    <label for="numero1">NUMERO 1: <input type="number" id="numero1" /></label>
    <select id="operacao">
      <option value="+">+</option>
      <option value="-">-</option>
      <option value="/">/</option>
      <option value="*">*</option>
    </select>
    <label for="numero2">NUMERO 2: <input type="number" id="numero2" /></label>
    <button type="submit">CALCULAR</button>
  </form>
  <div class="resultado"></div>
</body>
```

```
</html>
```

#### 4.1.2 CALCULAR.JS

```
window.onload = () =>{
  let campo_numero1 = document.getElementById("numero1")
  let campo_numero2 = document.getElementById("numero2")
  let select_operacao = document.getElementById("operacao")
  let calculadora = document.getElementById('calculadora')

  const calcular = () =>{
    if(campo_numero1.value == null || campo_numero1.value == '' || campo_numero2.value
    == null || campo_numero2.value == '')
    {
      alert("Digite um valor válido")
      return
    }
    let numero1 = parseFloat(campo_numero1.value)
    let numero2 = parseFloat(campo_numero2.value)
    let operacao = select_operacao.value
    let resultado_container = document.querySelector(".resultado")
    let resultado = 0

    try{
      switch(operacao){
        case '*':
          resultado = multiplicacao(numero1, numero2)
          break
        case '+':
          resultado = soma(numero1, numero2)
          break
        case '-':
          resultado = subtracao(numero1, numero2)
          break
        case '/':
          resultado = divisao(numero1, numero2)
          break
      }
    }
  }
}
```

```

        default:
            alert("Selecione uma operação válida")
            return
        }
    }catch(error){
        alert(error.message)
        return
    }
    resultado_container.innerText = `O resultado é ${resultado.toFixed(2)}`
}

calculadora.addEventListener("submit", (e) => {
    e.preventDefault()
    calcular()
})
}

```

#### 4.1.3 OPERACOES.JS

```

/**
 *
 * @param {number} numero_1
 * @param {number} numero_2
 * @returns {number}
 */
const soma = (numero_1, numero_2) =>{
    return numero_1 + numero_2
}

/**
 *
 * @param {number} minuendo
 * @param {number} subtraendo
 * @returns {number} diferenca
 */
const subtracao = (minuendo, subtraendo) =>{
    return minuendo - subtraendo
}

```

```

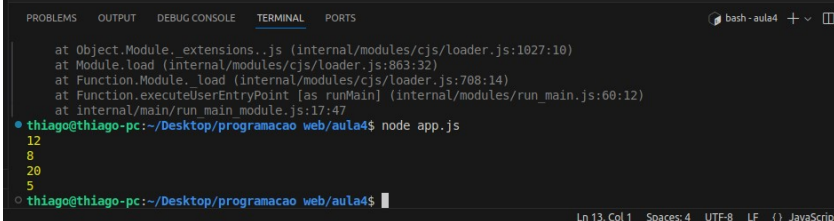
/**
 *
 * @param {number} multiplicando
 * @param {number} multiplicador
 * @returns {number} produto
 */
const multiplicacao = (multiplicando, multiplicador) => {
    return multiplicando * multiplicador
}

/**
 *
 * @param {number} dividendo
 * @param {number} divisor
 * @returns {number} quociente
 * @throws {Error}
 */
const divisao = (dividendo, divisor) => {
    if(divisor == 0){
        throw new Error("Não é possível dividir por zero")
    }
    return dividendo / divisor
}

```

## 5 CALCULADORA COM NODE.JS

A atividade proposta foi criar a mesma aplicação(calculadora) porém utilizando Node JS. Foi criado a aplicação e este foi o resultado impresso no terminal



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash - aula4 + v

at Object.Module._extensions..js (internal/modules/cjs/loader.js:1027:10)
at Module.load (internal/modules/cjs/loader.js:863:32)
at Function.Module._load (internal/modules/cjs/loader.js:708:14)
at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:60:12)
at internal/main/run_main_module.js:17:47
thiago@thiago-pc:~/Desktop/programacao web/aula4$ node app.js
12
8
20
5
thiago@thiago-pc:~/Desktop/programacao web/aula4$
Ln 13, Col 1 Spaces: 4 UTF-8 LF (i) JavaScript

```



## 5.1 CÓDIGO

### 5.1.1 SOMAR.JS

```
/**
 *
 * @param {number} numero_1
 * @param {number} numero_2
 * @returns {number}
 */
const soma = (numero_1, numero_2) =>{
  return numero_1 + numero_2
}

module.exports = soma
```

### 5.1.2 SUBTRAIR.JS

```
/**
 *
 * @param {number} minuendo
 * @param {number} subtraendo
 * @returns {number} diferenca
 */
const subtracao = (minuendo, subtraendo) =>{
  return minuendo - subtraendo
}

module.exports = subtracao
```

### 5.1.3 DIVIDIR.JS

```
const divisao = (dividendo, divisor) => {
  if(divisor == 0){
```

```
        throw new Error("Não é possível dividir por zero")
    }
    return dividendo / divisor
}

module.exports = divisao
```

#### 5.1.4 MULTIPLICAR.JS

```
/**
 *
 * @param {number} multiplicando
 * @param {number} multiplicador
 * @returns {number} produto
 */
const multiplicacao = (multiplicando, multiplicador) => {
    return multiplicando * multiplicador
}

module.exports = multiplicacao
```

#### 5.1.5 APP.JS

```
const somar = require('./somar')
const subtrair = require('./subtrair')
const multiplicar = require('./multiplicar')
const dividir = require('./dividir')

var numero1 = 10
var numero2 = 2

console.log(somar(numero1, numero2))
console.log(subtrair(numero1, numero2))
console.log(multiplicar(numero1, numero2))
```

```
console.log(dividir(numero1,numero2))
```

## 6 CALCULADORA PHP

Foi proposto que fosse criado uma calculadora em PHP

### 6.1 RESULTADO

Foi criado um arquivo calcula.php, que realiza as operações. O Javascript realiza algumas validações, converte os dados do formulário em JSON e envia ao PHP. O PHP refaz as validações, calcula o resultado e devolve ao Front-End.

### 6.2 CÓDIGOS

#### 6.2.1 INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <script src="operacoes.js"></script>
  <script src="calcular.js"></script>
  <title>aula 3</title>
</head>
<body>
  <form id="calculadora">
    <label for="numero1">NUMERO 1: <input type="number" id="numero1" /></label>
```

```

<select id="operacao">
  <option value="+">+</option>
  <option value="-">-</option>
  <option value="/">/</option>
  <option value="*">*</option>
</select>
<label for="numero2">NUMERO 2: <input type="number" id="numero2" /></label>
<button type="submit">CALCULAR</button>
</form>
<div class="resultado"></div>
</body>
</html>

```

## 6.2.2 CALCULAR.JS

```

window.onload = () =>{
  let campo_numero1 = document.getElementById("numero1")
  let campo_numero2 = document.getElementById("numero2")
  let select_operacao = document.getElementById("operacao")
  let calculadora = document.getElementById('calculadora')

  const calcular = async() =>{
    if(campo_numero1.value == null || campo_numero1.value == '' || campo_numero2.value
    == null || campo_numero2.value == '')
    {
      alert("Digite um valor válido")
      return
    }
    let numero1 = parseFloat(campo_numero1.value)
    let numero2 = parseFloat(campo_numero2.value)
    let operacao = select_operacao.value
    let resultado_container = document.querySelector(".resultado")
    let resultado = 0

    try{
      switch(operacao){
        case '*':
          resultado = await multiplicacao(numero1, numero2)

```

```

        break
      case '+':
        resultado = await soma(numero1, numero2)
        break
      case '-':
        resultado = await subtracao(numero1, numero2)
        break
      case '/':
        resultado = await divisao(numero1, numero2)
        break
      default:
        alert("Selecione uma operação válida")
        return
    }
  }catch(error){
    alert(error.message)
    return
  }
  resultado_container.innerText = `O resultado é ${resultado.toFixed(2)}`
}

calculadora.addEventListener("submit", (e) => {
  e.preventDefault()
  calcular()
})
}

```

### 6.3.3 OPERAÇÕES.JS

```

/**
 *
 * @param {number} numero_1
 * @param {number} numero_2
 * @param {string} operacao
 * @returns
 */
const api_calcula = async (numero_1, numero_2, operacao) => {

```

```
    let data = {
      "numero_1": parseFloat(numero_1),
      "numero_2": parseFloat(numero_2),
      "operacao": operacao
    }

    let resposta = await fetch("calcula.php",{
      method: "POST",
      body: JSON.stringify(data)
    }).then(response=> response.json())

    if(!resposta.success) throw new Error(resposta.error);

    return resposta.value
  }

/**
 *
 * @param {number} numero_1
 * @param {number} numero_2
 * @returns {number}
 */
const soma = async(numero_1, numero_2) =>{
  return await api_calcula(numero_1, numero_2, "+");
}

/**
 *
 * @param {number} minuendo
 * @param {number} subtraendo
 * @returns {number} diferenca
 */
const subtracao = async(minuendo, subtraendo) =>{
  return await api_calcula(minuendo, subtraendo, "-")
}

/**
 *
 * @param {number} multiplicando
 * @param {number} multiplicador
```

```

    * @returns {number} produto
    */
    const multiplicacao = async(multiplicando, multiplicador) => {
        return await api_calcula(multiplicacao, multiplicador, "*");
    }

    /**
     *
     * @param {number} dividendo
     * @param {number} divisor
     * @returns {number} quociente
     * @throws {Error}
     */
    const divisao = async(dividendo, divisor) => {
        if(divisor == 0){
            throw new Error("Não é possível dividir por zero")
        }
        return await api_calcula(dividendo,divisor,"/");
    }
}

```

#### 6.3.4 CALCULA.PHP

```

<?php

header('Content-Type: application/json');

/**
 *
 * @param string $error
 * @return never
 */
function error($error){
    echo json_encode(['success' => false, 'error' => $error]);
    die();
}

```

```
/**
 * Summary of success
 * @param numeric $value
 * @return never
 */
function success($value){
    echo json_encode(['success' => true, 'value' => $value]);
    die();
}

/**
 * @param numeric $valor1
 * @param numeric $valor2
 * @return never
 */
function soma($valor1, $valor2){
    success($valor1 + $valor2);
}

/**
 * Summary of subtracao
 * @param numeric $minuendo
 * @param numeric $subtraendo
 * @return never
 */
function subtracao($minuendo, $subtraendo){
    success($minuendo - $subtraendo);
}

/**
 * Summary of multiplicacao
 * @param numeric $multiplicando
 * @param numeric $multiplicador
 * @return never
 */
function multiplicacao($multiplicando, $multiplicador){
    success($multiplicando * $multiplicador);
}

/**
```



```

* Summary of divisao
* @param numeric $dividendo
* @param numeric $divisor
* @return never
*/
function divisao($dividendo, $divisor){
    if($divisor == 0) error("não é possível divisão por zero");
    success($dividendo / $divisor);
}

if($_SERVER['REQUEST_METHOD'] !== "POST") error("método não permitido");

$data = file_get_contents("php://input");

$data_json = json_decode($data);

if(json_last_error() !== JSON_ERROR_NONE) error("dados inválidos enviados");

if(!isset($data_json->numero_1) or !isset($data_json->numero_2) or !isset($data_json->operacao)) error("dados inválidos enviados");

$numero_1 = $data_json->numero_1;

$numero_2 = $data_json->numero_2;

$operacao = $data_json->operacao;

if(!is_numeric($numero_1) or !is_numeric($numero_2)) error("apenas valores numéricos são permitidos");

switch($operacao){
    case "+":
        soma($numero_1,$numero_2);
        break;
    case "-":
        subtracao($numero_1,$numero_2);
        break;
    case "/":
        divisao($numero_1,$numero_2);
        break;
}

```

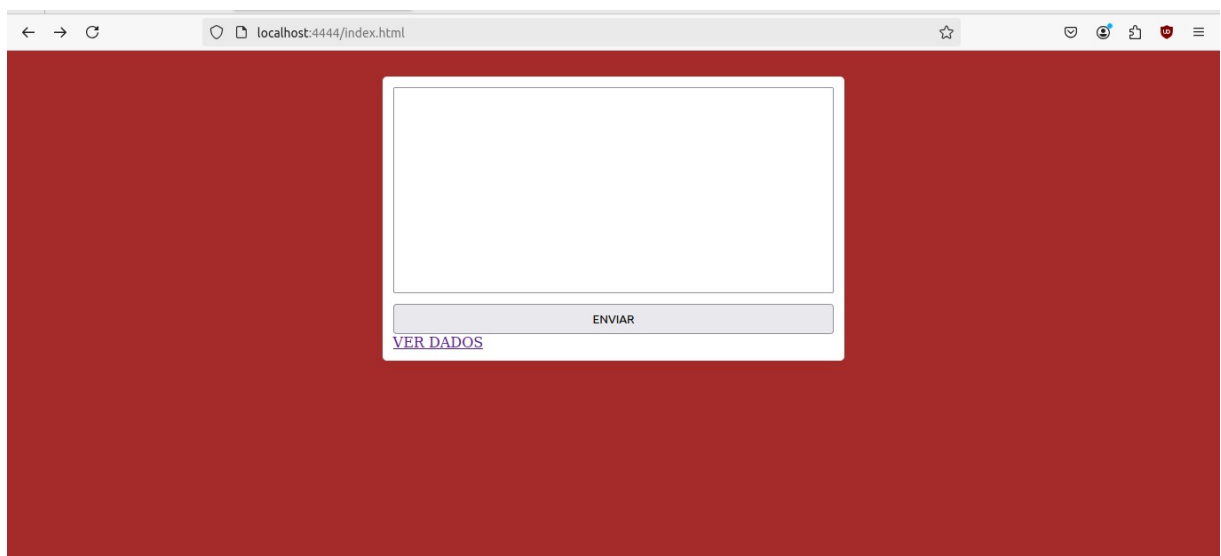
```
case "*":
    multiplicacao($numero_1, $numero_2);
    break;
default:
    error("operação inválida");
}
```

## 7 SALVANDO TEXTO EM ARQUIVO COM PHP

A atividade proposta foi criar um sistema que capturasse os dados digitados pelo usuário e salvasse no arquivo.

### 7.1 RESULTADO

Foi criada uma interface HTML com um formulário que envia via POST os dados enviados pelo usuário. No script PHP, ele abre o arquivo “dados.txt” no modo append, e escreve os dados, e adiciona duas linhas para separar as informações.



## 7.2 CÓDIGOS

### 7.2.1 INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    *{
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body{
      width: 100%;
      height: 100%;
      background-color: brown;
    }

    .container{
      margin: 5vh auto;
      width: min(520px,95%);
      background-color: white;
      padding: 2vh;
      border-radius: 5px;
    }

    textarea{
      width: 100%;
      height: 40vh;
      resize: none;
      padding: 1.2em;
      margin-bottom: 2vh;
    }

    button{
      width: 100%;
```

```
        padding: 1.2vh;
    }
</style>
</head>
<body>
    <form class="container" method="POST" action="write.php">
        <textarea name="text"></textarea>
        <button type="submit">ENVIAR</button>
        <a href="dados.txt">VER DADOS</a>
    </form>
</body>
</html>
```

### 7.2.2 WRITE.PHP

```
<?php

$text = $_POST["text"];

$file = fopen("dados.txt","a");

fwrite($file,$text);
fwrite($file,"\r\n\r\n");

fclose($file);

header("location:index.html");
```

## 8 CADASTRO DE USUÁRIOS COM PHP

Foi proposto criar um script php que realiza a inserção de alunos em um banco de dados.

## 8.1 CRIAÇÃO DO BANCO DE DADOS.

O banco foi criado executando os seguintes comandos.

```
mysql>
mysql>
mysql> CREATE USER 'ads'@'localhost' IDENTIFIED BY "123456"
-> ;
Query OK, 0 rows affected (0,12 sec)

mysql> CREATE DATABASE 'alunocadastro';
Query OK, 1 row affected (0,03 sec)

mysql> GRANT ALL PRIVILEGES ON alunocadastro.* to 'ads'@'localhost' WITH GRANT OPTION;
Query OK, 0 rows affected (0,03 sec)

mysql> USE alunocadastro;
Database changed
mysql> CREATE TABLE `tbl_aluno` (
-> `id` INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
-> `nome` VARCHAR(255) NOT NULL,
-> `email` VARCHAR(100) NOT NULL,
-> `telefone` VARCHAR(50) NOT NULL,
-> `idade` INT NOT NULL);
Query OK, 0 rows affected (0,21 sec)

mysql>
```

Criamos um usuário para manipular o banco, e usaremos sua credencial para manipular o banco no PHP.

## 8.2 RESULTADO

Foi criado a seguinte interface gráfica.

**CADASTRO DE ALUNOS**

nome:

telefone:

email:

idade:

id	nome	email	telefone	idade
1	Thiago	contato@thiagosantos.blog	3234234	26

No formulário é possível cadastrar os alunos, e abaixo, se faz uma seleção da tabela e exibe os alunos já cadastrados.

## 8.3 CÓDIGO

### 8.3.1 INDEX.PHP

```
<?php
$host = "localhost";
$user = "ads";
$db = "alunocadastro";
$senha = "123456";

$erro = '';

$pdo = new PDO("mysql:host=".$host.";dbname=".$db,$user,$senha);

if(!$pdo) die("Erro ao conectar no banco de dados");

if($_SERVER['REQUEST_METHOD'] === 'POST'){
    try
    {
        $nome = $_POST['nome'];
        $email = $_POST['email'];
        $idade = $_POST['idade'];
        $telefone = $_POST['telefone'];

        $st = $pdo->prepare("INSERT INTO tbl_aluno(nome,email,idade,telefone)
VALUES(:nome,:email,:idade,:telefone)");
        $st->bindValue(":nome",$nome);
        $st->bindValue(":email",$email);
        $st->bindValue(":idade",$idade);
        $st->bindValue(":telefone",$telefone);

        $st->execute();

        $st->closeCursor();
    }catch(Exception $e){
        $erro = $e->getMessage();
    }
}
?>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="style.css">
  <title>CADASTRO DE ALUNOS</title>
</head>
<body>
  <header >
    <h1>CADASTRO DE ALUNOS</h1>
  </header>
  <?php

  if($erro != '')
  {
    echo '<div class="erro"></div>';
    echo '<h2>ERRO:'. $erro. '</h2>';
    echo '</div>';
  }

  ?>

  <main>
    <form class="container" method="POST">
      <label for="nome">nome: <input type="text" name="nome" id="nome" /></label>
      <label for="telefone">telefone: <input type="text" name="telefone"
id="telefone" /></label>
      <label for="email">email: <input type="text" name="email" id="email" /></label>
      <label for="idade">idade: <input type="text" name="idade" id="idade" /></label>
      <button type="submit">CADASTRAR</button>
    </form>

    <table class="container">
      <thead>
        <th>id</th>
        <th>nome</th>
        <th>email</th>
        <th>telefone</th>
        <th>idade</th>
      </thead>
    </table>
  </main>
</body>
</html>
```

```

        </thead>
        <tbody>
            <?php
                $stmt = $pdo->prepare("SELECT * FROM tbl_aluno");
                $stmt->execute();
                $alunos = $stmt->fetchAll(PDO::FETCH_ASSOC);
                if(count($alunos) === 0){
                    echo "<tr><td colspan='5' align='center'>Não há alunos
cadastrados</td></tr>";
                }else{
                    foreach($alunos as $aluno){
                        echo '<tr>';
                        echo '<td>'.$aluno['id'].'</td>';
                        echo '<td>'.$aluno['nome'].'</td>';
                        echo '<td>'.$aluno['email'].'</td>';
                        echo '<td>'.$aluno['telefone'].'</td>';
                        echo '<td>'.$aluno['idade'].'</td>';

                        echo '</tr>';
                    }
                }
            ?>
        </tbody>
    </table>
</main>
</body>
</html>

```

### 8.3.2 STYLE.CSS

```

*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

.container{
    margin: 5vh auto;
}

```



```
width: min(520px,95%);
background-color: white;
padding: 2vh;
border-radius: 5px;
}
body{
width: 100%;
height: 100%;
background-color: brown;
}

header{
background-color: aliceblue;
text-align: center;
padding: 3vh;
}

form label{
width:100%;
padding: 1vh;
}

form input{
width: 100%;
padding:1.5vh;
margin-bottom: 1vh;
}

form button{
width: 100%;
padding:1.5vh;
margin-bottom: 1vh;
}

.erro{
margin: 5vh auto;
width: min(520px,95%);
background-color: rgb(243, 153, 153);
padding: 2vh;
```

```

border-radius: 5px;
border: 1px solid red;
text-align: center;
}

table tr{
padding: 1vh;
text-align: center;
border: 1px solid black;
border-collapse: collapse;
}
tr,td,th{
border: 1px solid black;
border-collapse: collapse;
}

```

### 8.3.3 DATABASE.SQL

```

CREATE USER `ads`@`localhost` IDENTIFIED BY "123456";
CREATE DATABASE `alunocadastro`;
GRANT ALL PRIVILEGES ON alunocadastro.* to `ads`@`localhost` WITH GRANT OPTION;

CREATE TABLE `tbl_aluno`(`id` INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
`nome` VARCHAR(255) NOT NULL,
`email` VARCHAR(100) NOT NULL,
`telefone` VARCHAR(50) NOT NULL,
`idade` INT NOT NULL);

```

## 9 CONCLUSÃO

À partir das atividades propostas, foi possível obter conhecimento prático sobre como criar uma aplicação Web, estruturar o Front End e o Back End. Também como armazenar dados advindo dos clientes, ou em arquivos ou em banco de dados, e o básico sobre o Protocolo HTTP.

