

Missão Prática - Nível 2

//Disciplina: Descobrimdo o JavaScript – 1º Procedimento

//Professor: Maria Manso

//Aluno: Rafael Lima de Medeiros

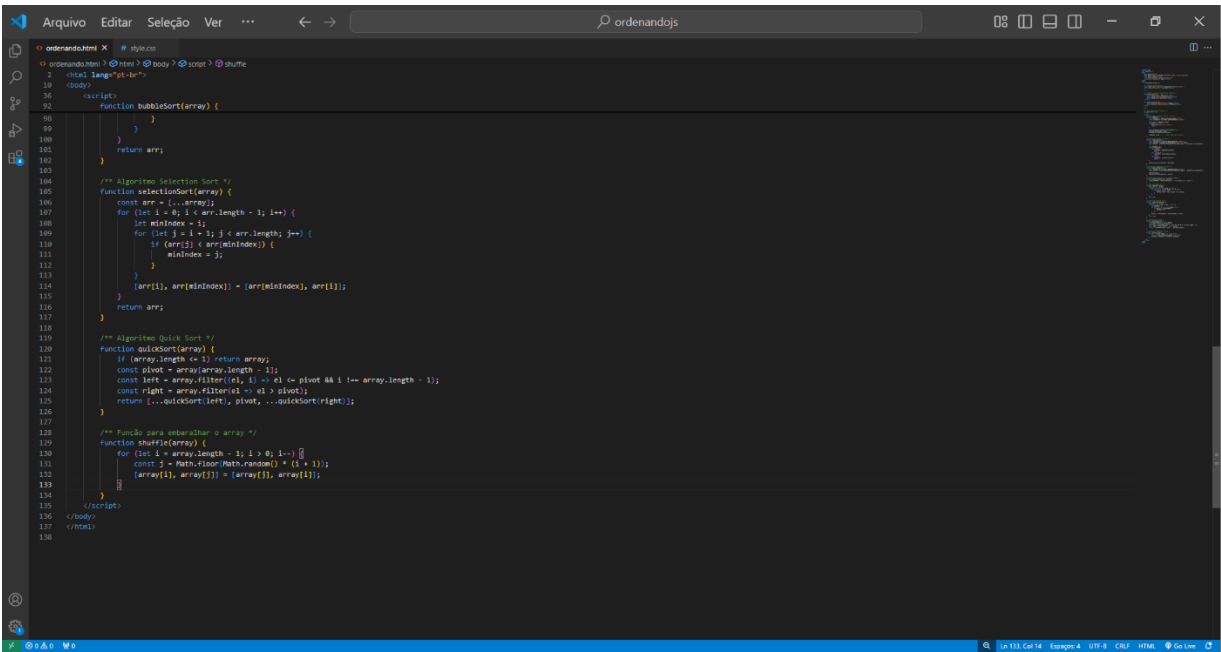
//Turma: 2024.2

//Data atual: 04/01/2025

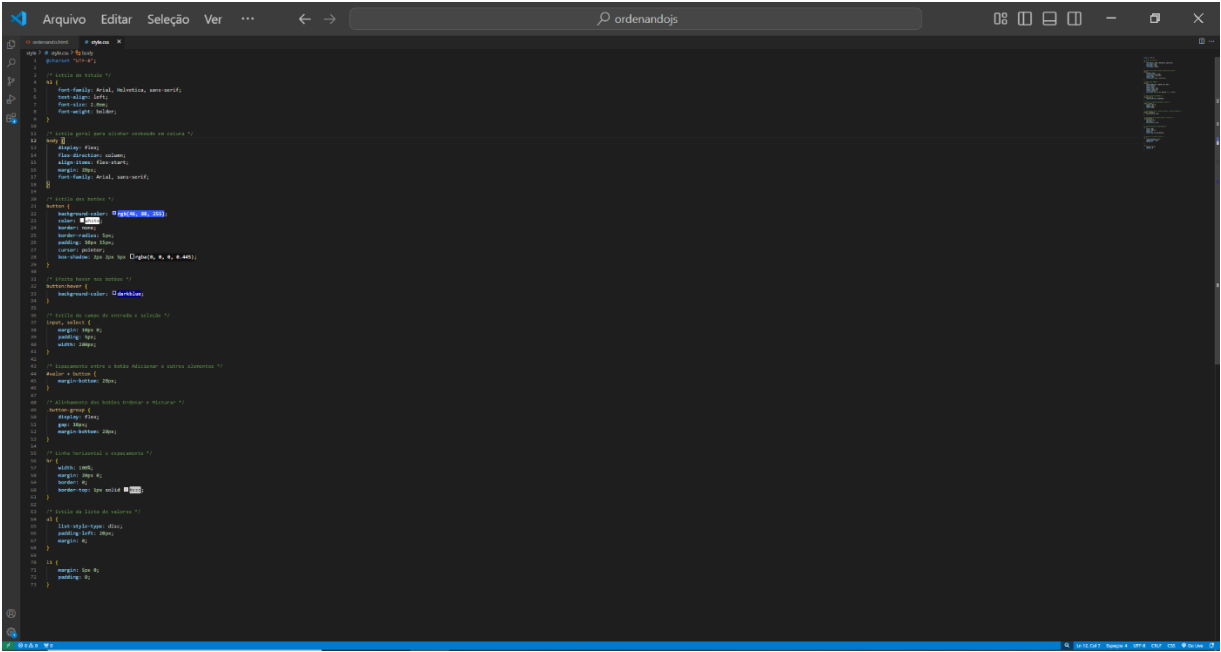
Código HTML + Script:

```
Arquivo Editar Seleção Ver ... ordenandojs
ordenando.html X # styles
ordenando.html > index > body > script > shuffle
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 <title>Lista Ordenação</title>
7 <link rel="stylesheet" href="style.css">
8 <script src="ordenando.js" defer</script>
9 </head>
10 <body>
11 <div Ordenando Valores</div>
12
13 <!-- Campo de entrada numérica -->
14 <input type="number" id="valor" placeholder="Digite um valor" />
15 <button onclick="adicionar()">Adicionar</button>
16
17 <hr />
18
19 <!-- Campo de seleção de algoritmos de ordenação -->
20 <select id="algoritmo">
21 <option value="bubble">Bubble Sort</option>
22 <option value="selection">Selection Sort</option>
23 <option value="quick">Quick Sort</option>
24 </select>
25
26 <div class="button-group">
27 <button onclick="ordenarValores()">Ordenar</button>
28 <button onclick="embaralharValores()">Embaralhar</button>
29 </div>
30
31 <hr />
32
33 <!-- Lista para exibir os valores -->
34 <ul id="valores"><li></li>
35 </ul>
36
37 <script>
38 /** Função para adicionar um valor à lista */
39 function adicionar() {
40   const campoValor = document.getElementById('valor');
41   const listaValores = document.getElementById('valores');
42
43   const valor = campoValor.value;
44   if (valor === '') {
45     alert('Digite um valor válido!');
46     return;
47   }
48
49   const novoItem = document.createElement('li');
50   novoItem.textContent = valor;
51   listaValores.appendChild(novoItem);
52   campoValor.value = ''; // limpa o campo após adicionar
53 }
54
```

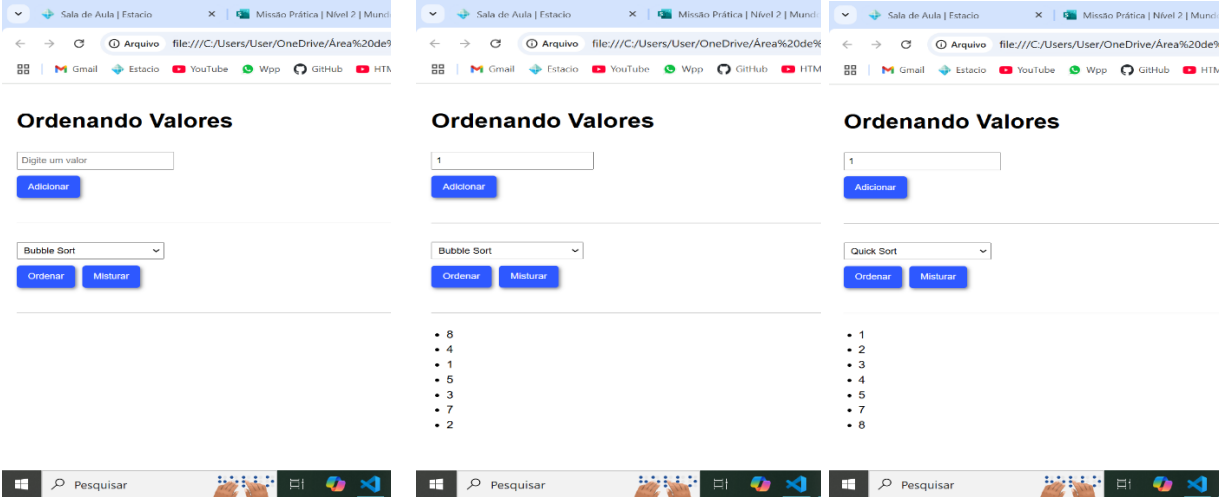
```
Arquivo Editar Seleção Ver ... ordenandojs
ordenando.html X # styles
ordenando.html > index > body > script > shuffle
1 <!DOCTYPE html>
2 <html lang="pt-br">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Lista Ordenação</title>
7 <link rel="stylesheet" href="style.css">
8 <script src="ordenando.js" defer</script>
9 </head>
10 <body>
11 <div Ordenando Valores</div>
12
13 <!-- Campo de entrada numérica -->
14 <input type="number" id="valor" placeholder="Digite um valor" />
15 <button onclick="adicionar()">Adicionar</button>
16
17 <hr />
18
19 <!-- Campo de seleção de algoritmos de ordenação -->
20 <select id="algoritmo">
21 <option value="bubble">Bubble Sort</option>
22 <option value="selection">Selection Sort</option>
23 <option value="quick">Quick Sort</option>
24 </select>
25
26 <div class="button-group">
27 <button onclick="ordenarValores()">Ordenar</button>
28 <button onclick="embaralharValores()">Embaralhar</button>
29 </div>
30
31 <hr />
32
33 <!-- Lista para exibir os valores -->
34 <ul id="valores"><li></li>
35 </ul>
36
37 <script>
38 /** Função para adicionar um valor à lista */
39 function adicionar() {
40   const campoValor = document.getElementById('valor');
41   const listaValores = document.getElementById('valores');
42
43   const valor = campoValor.value;
44   if (valor === '') {
45     alert('Digite um valor válido!');
46     return;
47   }
48
49   const novoItem = document.createElement('li');
50   novoItem.textContent = valor;
51   listaValores.appendChild(novoItem);
52   campoValor.value = ''; // limpa o campo após adicionar
53 }
54
55 /** Função para ordenar os valores */
56 function ordenarValores() {
57   const listaValores = document.getElementById('valores');
58   const algoritmo = document.getElementById('algoritmo').value;
59   const valores = Array.from(listaValores.children).map(li => parseInt(li.textContent));
60
61   let resultado = [];
62   switch (algoritmo) {
63     case 'bubble':
64       resultado = bubbleSort(valores);
65       break;
66     case 'selection':
67       resultado = selectionSort(valores);
68       break;
69     case 'quick':
70       resultado = quickSort(valores);
71       break;
72   }
73
74   atualizarLista(listaValores, resultado);
75 }
76
77 /** Função para embaralhar os valores */
78 function embaralharValores() {
79   const listaValores = document.getElementById('valores');
80   const valores = Array.from(listaValores.children).map(li => parseInt(li.textContent));
81
82   shuffle(valores);
83   atualizarLista(listaValores, valores);
84 }
85
86 /** Função para atualizar a lista exibida no HTML */
87 function atualizarLista(lista, valores) {
88   lista.innerHTML = valores.map(valor => `<li>${valor}</li>`).join('');
89 }
90
91 /** Algoritmo Bubble Sort */
92 function bubbleSort(array) {
93   const arr = [...array];
94   for (let i = 0; i < arr.length - 1; i++) {
95     for (let j = 0; j < arr.length - i - 1; j++) {
96       if (arr[j] > arr[j + 1]) {
97         [arr[j], arr[j + 1]] = [arr[j + 1], arr[j]];
98       }
99     }
100   }
101   return arr;
102 }
103
104 /** Algoritmo Selection Sort */
105 function selectionSort(array) {
106   const arr = [...array];
107   for (let i = 0; i < arr.length - 1; i++) {
108     let minIndex = i;
109     for (let j = i + 1; j < arr.length; j++) {
110       if (arr[j] < arr[minIndex]) {
111         minIndex = j;
112       }
113     }
114     [arr[i], arr[minIndex]] = [arr[minIndex], arr[i]];
115   }
116   return arr;
117 }
118
119 /** Algoritmo Quick Sort */
120 function quickSort(array) {
121   const arr = [...array];
122   if (arr.length < 2) {
123     return arr;
124   }
125   const pivot = arr[arr.length - 1];
126   const [less, equal, greater] = arr.reduce((acc, val, i) => {
127     if (val < pivot) {
128       acc[0].push(val);
129     } else if (val === pivot) {
130       acc[1].push(val);
131     } else {
132       acc[2].push(val);
133     }
134     return acc;
135   }, [[], [], []]);
136   return [...quickSort(less), ...equal, ...quickSort(greater)];
137 }
138
139 /** Função para embaralhar os valores */
140 function shuffle(array) {
141   const arr = [...array];
142   for (let i = arr.length - 1; i > 0; i--) {
143     const j = Math.floor(Math.random() * (i + 1));
144     [arr[i], arr[j]] = [arr[j], arr[i]];
145   }
146   return arr;
147 }
148
149 /** Função para atualizar a lista exibida no HTML */
150 function atualizarLista(lista, valores) {
151   lista.innerHTML = valores.map(valor => `<li>${valor}</li>`).join('');
152 }
153
```



Estilo:



Execução do código:



Código digitado:

Ordenando.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Lista Ordenação</title>
  <link rel="stylesheet" href="style/style.css">
  <script src="ordenando.js" defer></script>
</head>
<body>
  <h1>Ordenando Valores</h1>

  <!-- Campo de entrada numérico -->
  <input type="number" id="valor" placeholder="Digite um valor" />
  <button onclick="addValor()">Adicionar</button>

  <hr />

  <!-- Campo de seleção de algoritmos de ordenação -->
  <select id="algoritmo">
    <option value="bubble">Bubble Sort</option>
    <option value="selection">Selection Sort</option>
    <option value="quick">Quick Sort</option>
  </select>

  <div class="button-group">
    <button onclick="ordenarValores()">Ordenar</button>
    <button onclick="embaralharValores()">Misturar</button>
  </div>

  <hr />

  <!-- Lista para exibir os valores -->
  <ul id="valores"></ul>

<script>
  /** Função para adicionar um valor à lista */
  function addValor() {
    const campoValor = document.getElementById('valor');
    const listaValores = document.getElementById('valores');

    const valor = campoValor.value;
    if (valor === '') {
      alert('Digite um valor válido!');
      return;
    }

    const novoltem = document.createElement('li');
    novoltem.textContent = valor;
    listaValores.appendChild(novoltem);

    campoValor.value = ''; // Limpa o campo após adicionar
  }

  /** Função para ordenar os valores */
  function ordenarValores() {
    const listaValores = document.getElementById('valores');
    const algoritmo = document.getElementById('algoritmo').value;
    const valores = Array.from(listaValores.children).map(li => parseInt(li.textContent));

    let resultado = [];
    switch (algoritmo) {
      case 'bubble':
        resultado = bubbleSort(valores);
        break;
      case 'selection':
        resultado = selectionSort(valores);
        break;
      case 'quick':
        resultado = quickSort(valores);
        break;
    }

    atualizarLista(listaValores, resultado);
  }

  /** Função para embaralhar os valores */
```

```

function embaralharValores() {
  const listaValores = document.getElementById('valores');
  const valores = Array.from(listaValores.children).map(li => parseInt(li.textContent));

  shuffle(valores);
  atualizarLista(listaValores, valores);
}

/** Função para atualizar a lista exibida no HTML */
function atualizarLista(lista, valores) {
  lista.innerHTML = valores.map(valor => `<li>${valor}</li>`).join("");
}

/** Algoritmo Bubble Sort */
function bubbleSort(array) {
  const arr = [...array];
  for (let i = 0; i < arr.length - 1; i++) {
    for (let j = 0; j < arr.length - i - 1; j++) {
      if (arr[j] > arr[j + 1]) {
        [arr[j], arr[j + 1]] = [arr[j + 1], arr[j]];
      }
    }
  }
  return arr;
}

/** Algoritmo Selection Sort */
function selectionSort(array) {
  const arr = [...array];
  for (let i = 0; i < arr.length - 1; i++) {
    let minIndex = i;
    for (let j = i + 1; j < arr.length; j++) {
      if (arr[j] < arr[minIndex]) {
        minIndex = j;
      }
    }
    [arr[i], arr[minIndex]] = [arr[minIndex], arr[i]];
  }
  return arr;
}

/** Algoritmo Quick Sort */
function quickSort(array) {
  if (array.length <= 1) return array;
  const pivot = array[array.length - 1];
  const left = array.filter((el, i) => el <= pivot && i !== array.length - 1);
  const right = array.filter(el => el > pivot);
  return [...quickSort(left), pivot, ...quickSort(right)];
}

/** Função para embaralhar o array */
function shuffle(array) {
  for (let i = array.length - 1; i > 0; i--) {
    const j = Math.floor(Math.random() * (i + 1));
    [array[i], array[j]] = [array[j], array[i]];
  }
}
</script>
</body>
</html>

```

Style.css

@charset "UTF-8";

/* Estilo do título */

```
h1 {
  font-family: Arial, Helvetica, sans-serif;
  text-align: left;
  font-size: 2.0em;
  font-weight: bolder;
}
```

/* Estilo geral para alinhar conteúdo em coluna */

```
body {
  display: flex;
  flex-direction: column;
  align-items: flex-start;
  margin: 20px;
  font-family: Arial, sans-serif;
}
```

/* Estilo dos botões */

```
button {
  background-color: rgb(46, 88, 255);
  color: white;
  border: none;
  border-radius: 5px;
  padding: 10px 15px;
  cursor: pointer;
  box-shadow: 2px 2px 5px rgba(0, 0, 0, 0.445);
}
```

/* Efeito hover nos botões */

```
button:hover {
  background-color: darkblue;
}
```

/* Estilo do campo de entrada e seleção */

```
input, select {
  margin: 10px 0;
  padding: 5px;
  width: 200px;
}
```

/* Espaçamento entre o botão Adicionar e outros elementos */

```
#valor + button {
  margin-bottom: 20px;
}
```

/* Alinhamento dos botões Ordenar e Misturar */

```
.button-group {
  display: flex;
  gap: 10px;
  margin-bottom: 20px;
}
```

/* Linha horizontal e espaçamento */

```
hr {
  width: 100%;
  margin: 20px 0;
  border: 0;
  border-top: 1px solid #ccc;
}
```

/* Estilo da lista de valores */

```
ul {
  list-style-type: disc;
  padding-left: 20px;
  margin: 0;
}
```

```
li {
  margin: 5px 0;
  padding: 0;
}
```

Missão Prática - Nível 2

//Disciplina: Descobrimdo o JavaScript – 2º Procedimento

//Professor: Maria Manso

//Aluno: Rafael Lima de Medeiros

//Turma: 2024.2

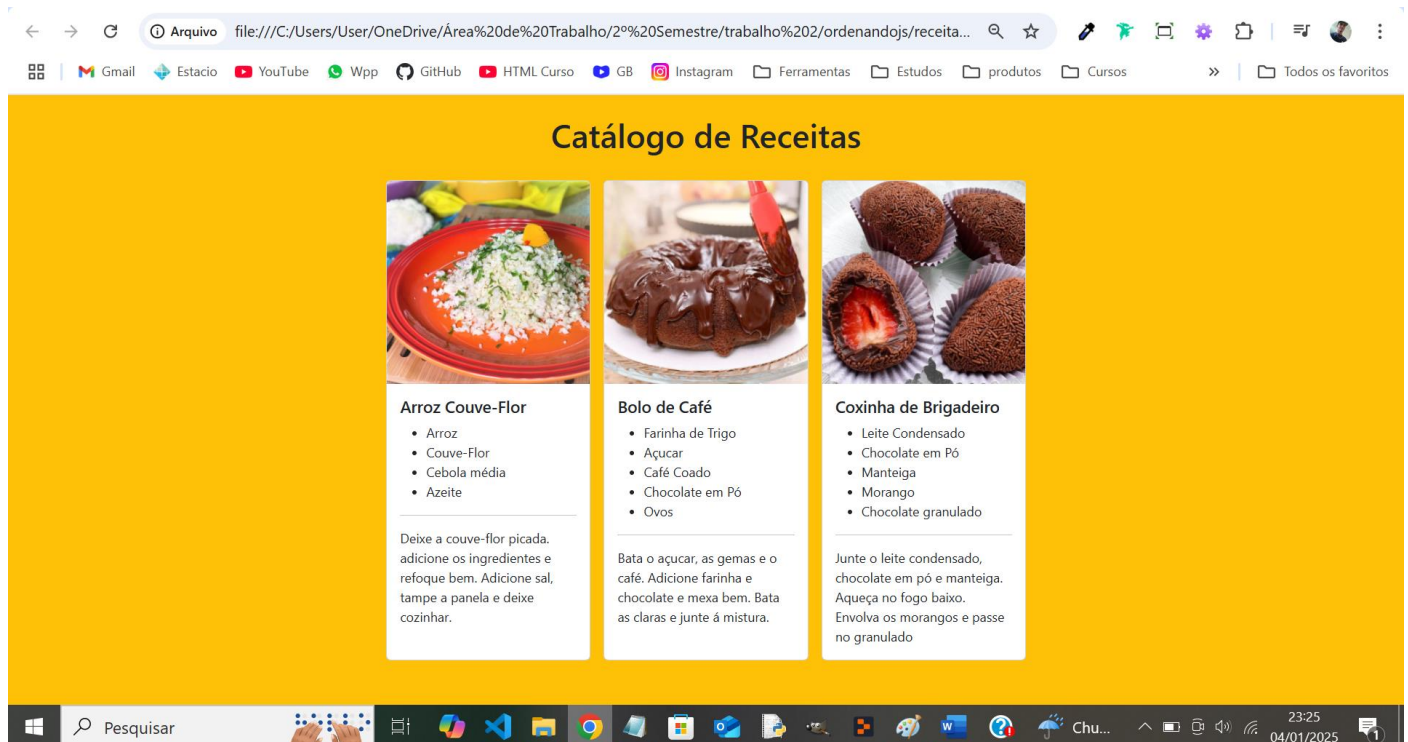
//Data atual: 04/01/2025

Código HTML + Script:

```
Arquivo Editar Seleção Ver ... ordenandojs
recetas.html > # style.css > recetas.html X
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 <title>Catálogo de Receitas</title>
7 <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
8 </head>
9 <body class="bg-warning" onload="preencheCatalogo()">
10 <div class="container-fluid">
11 <div class="text-center my-4">Catálogo de Receitas</div>
12 <div id="pnlCatalogo" class="d-flex flex-wrap justify-content-center"></div>
13 </div>
14 <script>
15 // Dados das receitas
16 const receitas = [
17   {
18     titulo: "Arroz Couve-Flor",
19     imagem: "arroz-couveflor.png",
20     preparo: "Deixe a couve-flor picada, adicione os ingredientes e refogue bem. Adicione sal, tampe a panela e deixe cozinhar.",
21     ingredientes: ["Arroz", "Couve-Flor", "Cebola média", "Azeite"]
22   },
23   {
24     titulo: "Bolo de Café",
25     imagem: "bolo-cafe.png",
26     preparo: "Bata o açúcar, as gemas e o café. Adicione farinha e chocolate e mexa bem. Bata as claras e junte a mistura.",
27     ingredientes: ["Farinha de Trigo", "Açúcar", "Café Coado", "Chocolate em Pó", "Ovos"]
28   },
29   {
30     titulo: "Coxinha de Brigadeiro",
31     imagem: "coxinha-brigadeiro.png",
32     preparo: "Junte o leite condensado, chocolate em pó e manteiga. Aqueça no fogo baixo. Envolve os morangos e passe no granulado.",
33     ingredientes: ["Leite Condensado", "Chocolate em Pó", "Manteiga", "Morango", "Chocolate granulado"]
34   },
35 ];
36
37 // Função para gerar lista de ingredientes
38 function getListIngredients(receita) {
39   return receita.ingredientes
40     .map(ingrediente => `<li>${ingrediente}</li>`)
41     .reduce((lista, item) => lista + item, 'ul') + `</ul>`;
42 }
43
44 </script>
45
```

```
Arquivo Editar Seleção Ver ... ordenandojs
recetas.html > # style.css > recetas.html X
1 <html lang="pt-br">
2 <body class="bg-warning" onload="preencheCatalogo()">
3 <script>
4 const receitas = [
5   {
6     titulo: "Arroz Couve-Flor",
7     imagem: "arroz-couveflor.png",
8     preparo: "Deixe a couve-flor picada, adicione os ingredientes e refogue bem. Adicione sal, tampe a panela e deixe cozinhar.",
9     ingredientes: ["Arroz", "Couve-Flor", "Cebola média", "Azeite"]
10   },
11   {
12     titulo: "Bolo de Café",
13     imagem: "bolo-cafe.png",
14     preparo: "Bata o açúcar, as gemas e o café. Adicione farinha e chocolate e mexa bem. Bata as claras e junte a mistura.",
15     ingredientes: ["Farinha de Trigo", "Açúcar", "Café Coado", "Chocolate em Pó", "Ovos"]
16   },
17   {
18     titulo: "Coxinha de Brigadeiro",
19     imagem: "coxinha-brigadeiro.png",
20     preparo: "Junte o leite condensado, chocolate em pó e manteiga. Aqueça no fogo baixo. Envolve os morangos e passe no granulado.",
21     ingredientes: ["Leite Condensado", "Chocolate em Pó", "Manteiga", "Morango", "Chocolate granulado"]
22   },
23 ];
24
25 // Função para gerar lista de ingredientes
26 function getListIngredients(receita) {
27   return receita.ingredientes
28     .map(ingrediente => `<li>${ingrediente}</li>`)
29     .reduce((lista, item) => lista + item, 'ul') + `</ul>`;
30 }
31
32 // Função para gerar o card da receita
33 function getCard(receita) {
34   return `
35     <div class="card m-2" style="width: 250px;">
36       
37       <div class="card-body">
38         <h3 class="card-title">${receita.titulo}</h3>
39         <div class="card-text">
40           ${getListIngredients(receita)}
41           <hr>
42           ${receita.preparo}
43         </div>
44       </div>
45     </div>
46   `;
47 }
48
49 // Função para preencher o catálogo de receitas
50 function preencheCatalogo() {
51   const pnlCatalogo = document.getElementById('pnlCatalogo');
52   pnlCatalogo.innerHTML = receitas
53     .map(receita => getCard(receita))
54     .reduce((html, card) => html + card, '');
55 }
56
57 </script>
58 </body>
59 </html>
60
```

Execução do código:



Código digitado:

receitas.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Catálogo de Receitas</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
</head>
<body class="bg-warning" onload="preencheCatalogo()">
  <div class="container-fluid">
    <h1 class="text-center my-4">Catálogo de Receitas</h1>
    <div id="pnlCatalogo" class="d-flex flex-wrap justify-content-center"></div>
  </div>

  <script>
    // Dados das receitas
    const receitas = [
      {
        titulo: "Arroz Couve-Flor",
        imagem: "arroz-couveflor.png",
        preparo: "Deixe a couve-flor picada. adicione os ingredientes e refoque bem. Adicione sal, tampe a panela e deixe cozinhar.",
        ingredientes: ["Arroz", "Couve-Flor", "Cebola média", "Azeite"]
      },
    ],
```

```

{
  titulo: "Bolo de Café",
  imagem: "bolo-cafe.png",
  preparo: "Bata o açúcar, as gemas e o café. Adicione farinha e chocolate e mexa bem. Bata as claras e junte á mistura.",
  ingredientes: ["Farinha de Trigo", "Açúcar", "Café Coadado", "Chocolate em Pó", "Ovos"]
},
{
  titulo: "Coxinha de Brigadeiro",
  imagem: "cozinha-brigadeiro.png",
  preparo: "Junte o leite condensado, chocolate em pó e manteiga. Aqueça no fogo baixo. Envolve os morangos e passe no granulado",
  ingredientes: ["Leite Condensado", "Chocolate em Pó", "Manteiga", "Morango", "Chocolate granulado"]
},
];

// Função para gerar lista de ingredientes
function getListIngredients(receita) {
  return receita.ingredientes
    .map(ingrediente => `<li>${ingrediente}</li>`)
    .reduce((lista, item) => lista + item, '<ul>') + '</ul>';
}

// Função para gerar o card da receita
function getCard(receita) {
  return `
    <div class="card m-2" style="width: 250px;">
      
      <div class="card-body">
        <h5 class="card-title">${receita.titulo}</h5>
        <div class="card-text">
          ${getListIngredients(receita)}
          <hr>
          ${receita.preparo}
        </div>
      </div>
    </div>
  `;
}

// Função para preencher o catálogo de receitas
function preencheCatalogo() {
  const pnlCatalogo = document.getElementById('pnlCatalogo');
  pnlCatalogo.innerHTML = receitas
    .map(receita => getCard(receita))
    .reduce((html, card) => html + card, "");
}
</script>
</body>
</html>

```


Missão Prática - Nível 2

//Disciplina: Descobrendo o JavaScript – 3º Procedimento

//Professor: Maria Manso

//Aluno: Rafael Lima de Medeiros

//Turma: 2024.2

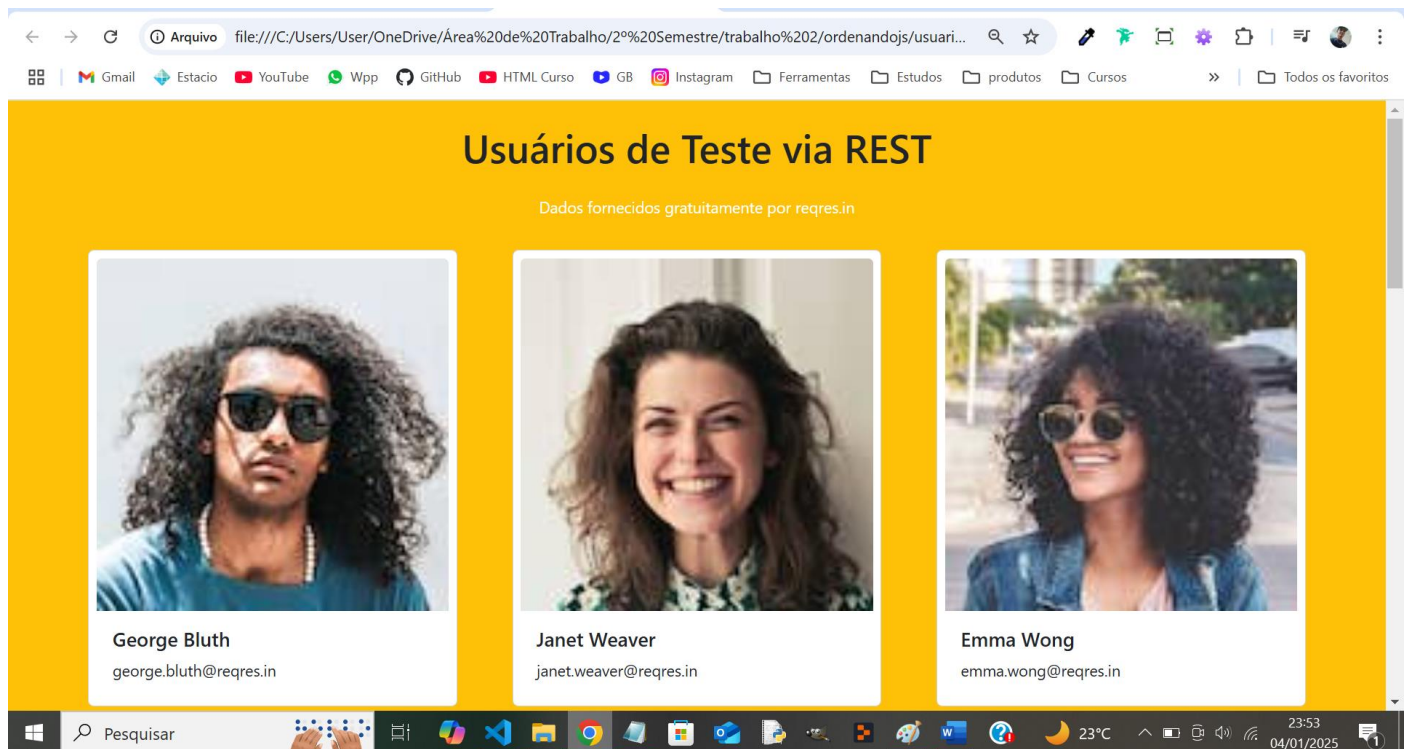
//Data atual: 04/01/2025

Código HTML + Script:

```
Arquivo Editar Seleção Ver ... ordenandojs
usuarios.html > html > head > style > .custom-paragraph a: hover
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 <title>Usuários de Teste via REST</title>
7 <!-- Bootstrap CSS -->
8 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
9 <!-- Vue.js -->
10 <script src="https://cdn.jsdelivr.net/npm/vue@2.7.14/dist/vue.js"></script>
11 <!-- Optional: jQuery -->
12 <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
13 <style>
14 .custom-paragraph {
15   color: white;
16 }
17 .custom-paragraph a {
18   text-decoration: none;
19   color: white;
20   font: bold;
21 }
22 .custom-paragraph a: hover {
23   color: yellow;
24   text-decoration: underline;
25 }
26 </style>
27 </head>
28 <body class="container bg-warning">
29 <h1 class="text-center my-4">Usuários de Teste via REST</h1>
30 <p class="text-center custom-paragraph">Dados fornecidos gratuitamente por <a href="https://reqres.in" target="_blank">reqres.in</a></p>
31
32 <div id="usuarios" class="container-fluid">
33   <div class="row">
34     <div v-for="(user, index) in users" :key="index" class="col-12 col-md-4">
35       <div class="card p-2 m-3">
36         
37       </div>
38     </div>
39   </div>
40 </div>
41
42 </body>
43 </html>
```

```
Arquivo Editar Seleção Ver ... ordenandojs
usuarios.html > html > head > style > .custom-paragraph a: hover
2 <html lang="pt-br">
28 <body class="container bg-warning">
31
32 <div id="usuarios" class="container-fluid">
33   <div class="row">
34     <div v-for="(user, index) in users" :key="index" class="col-12 col-md-4">
35       <div class="card p-2 m-3">
36         
37         <div class="card-body">
38           <div class="card-text">{{user.first_name}} {{user.last_name}}</div>
39           <p class="card-text">{{user.email}}</p>
40         </div>
41       </div>
42     </div>
43   </div>
44 </div>
45
46 <script>
47 new Vue({
48   el: "#usuarios",
49   data: function() {
50     return {
51       users: []
52     };
53   },
54   mounted() {
55     this.loadUsers();
56   },
57   methods: {
58     loadUsers() {
59       fetch("https://reqres.in/api/users?page=10")
60       .then(response => response.json())
61       .then(data => {
62         this.users = data.data;
63       })
64       .catch(error => console.error("Erro ao carregar usuários", error));
65     }
66   }
67 });
68 </script>
69
70 <!-- Bootstrap JS -->
71 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
72 </body>
73 </html>
```

Execução do código:



Código digitado:

receitas.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Usuários de Teste via REST</title>
  <!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
  <!-- Vue.js -->
  <script src="https://cdn.jsdelivr.net/npm/vue@2.7.14/dist/vue.js"></script>
  <!-- Optional: jQuery -->
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <style>
    .custom-paragraph {
      color: white;
    }
    .custom-paragraph a {
      text-decoration: none;
      color: white;
      font: bolder;
    }
    .custom-paragraph a:hover {
      color: yellow;
      text-decoration: underline;
    }
  </style>
</head>
<body class="container bg-warning">
  <h1 class="text-center my-4">Usuários de Teste via REST</h1>
```

```
<p class="text-center custom-paragraph">Dados fornecidos gratuitamente por <a href="https://reqres.in" target="_blank">reqres.in</a></p>

<div id="usuarios" class="container-fluid">
  <div class="row">
    <div v-for="(user, index) in users" :key="index" class="col-12 col-md-4">
      <div class="card p-2 m-3">
        
        <div class="card-body">
          <h5 class="card-title">{{user.first_name}} {{user.last_name}}</h5>
          <p class="card-text">{{user.email}}</p>
        </div>
      </div>
    </div>
  </div>
</div>

<script>
  new Vue({
    el: '#usuarios',
    data: function() {
      return {
        users: []
      };
    },
    mounted() {
      this.loadUsers();
    },
    methods: {
      loadUsers() {
        fetch('https://reqres.in/api/users?per_page=10')
          .then(response => response.json())
          .then(data => {
            this.users = data.data;
          })
          .catch(error => console.error('Erro ao carregar usuários:', error));
      }
    }
  });
</script>

<!-- Bootstrap JS -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```