

UC: Projetos - FrontEnd

DIAS: 17 e 18/10/2024 – Parte 2

Assunto:

Projeto 26 – Dashboard de Vendas

P26 - DASHBOARD DE VENDAS

- > images
- # estilos.css
- ↳ index.html
- 🖼 p26_screenshot.jpg
- JS scripts.js
- Excel Vendedor.xlsx

index.html X

index.html > ...

```
1  <!DOCTYPE html>
2  <!-- Declaração DOCTYPE para HTML5, instruindo o navegador a
3  |   |   usar o modo de padrões mais recentes para renderizar a página. --&gt;
4
5  &lt;html lang="pt-br"&gt;
6  <!-- A tag &lt;html&gt; começa o documento HTML, e o atributo 'lang="pt-br"' 
7  |   |   especifica que o idioma principal do documento é o
8  |   |   Português Brasileiro. --&gt;
9
10 &lt;head&gt;
11  <!-- A tag &lt;head&gt; contém metadados, links para folhas de estilo e scripts
12  |   |   que não são exibidos diretamente na área de conteúdo da página web. --&gt;
13
14  &lt;meta charset="UTF-8"&gt;
15  <!-- A tag &lt;meta&gt; com o atributo 'charset="UTF-8"' define a codificação
16  |   |   de caracteres para o documento como UTF-8, que suporta todos
17  |   |   os caracteres de idiomas modernos. --&gt;
18
19  &lt;meta name="viewport" content="width=device-width, initial-scale=1.0"&gt;
20  <!-- A tag &lt;meta&gt; com 'name="viewport"' configura a área de visualização
21  |   |   para dispositivos móveis. 'width=device-width' ajusta a largura
22  |   |   da página para seguir a largura da tela do dispositivo.
23  |   |   'initial-scale=1.0' define o nível de zoom inicial quando a
24  |   |   página é carregada. --&gt;
25
26  &lt;title&gt;Dashboard de Vendas&lt;/title&gt;
27  <!-- A tag &lt;title&gt; define o título da página, que é exibido na aba
28  |   |   do navegador. Este título é também usado por motores de
29  |   |   busca e bookmarks. --&gt;
30</pre>
```

✓ P26 - DASHBOARD DE VENDAS

```
> images  
# estilos.css  
<> index.html  
[img] p26_screenshot.jpg  
JS scripts.js  
[x] Vendedor.xlsx
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

<> index.html > html > body > div.container > div.barra-lateral > ul

```
61      <ul>
62          <!-- Lista não ordenada usada para agrupar os links de
63              | navegação do dashboard. --&gt;
64
65          &lt;li&gt;&lt;a href="#" id="mostrar-top-3"&gt;01 - Vendedor TOP 3&lt;/a&gt;&lt;/li&gt;
66          <!-- Primeiro item da lista com um link (âncora) que,
67              | exibe os três melhores vendedores.
68              | O 'href="#"' indica que o link é funcional através de JavaScript,
69              | não levando a uma nova página.
70              | 'id="mostrar-top-3"' é um identificador único usado para conectar o
71              | elemento a scripts que manipulam eventos de clique. --&gt;
72
73          &lt;li&gt;&lt;a href="#" id="mostrar-ranking"&gt;02 - Rank Vendas&lt;/a&gt;&lt;/li&gt;
74          <!-- Segundo item da lista com um link para mostrar o ranking de vendas.
75              | Similar ao primeiro item, utiliza-se JavaScript para
76              | controlar a ação do link. --&gt;
77
78          &lt;li&gt;&lt;a href="#" id="mostrar-resumo"&gt;03 - Resumo&lt;/a&gt;&lt;/li&gt;
79          <!-- Terceiro item da lista, um link para mostrar um
80              | resumo das vendas.
81              | Este também é controlado por JavaScript, indicado
82              | pelo 'href="#"' e pelo identificador único 'id'. --&gt;
83
84      &lt;/ul&gt;
85      <!-- Fim da lista de navegação na barra lateral. --&gt;
86
87  &lt;/div&gt;
88  <!-- Fim da div 'barra-lateral'. --&gt;</pre>
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

<> index.html > html > body > div.container > div.barra-lateral > ul

```
90     <div class="principal">
91         <!-- Div 'principal' que contém o conteúdo principal do dashboard,
92             exibindo diferentes seções de dados com base na seleção
93             do usuário na barra lateral. --&gt;
94
95         &lt;h1 id="titulo-principal"&gt;Melhores Vendedores do Mês&lt;/h1&gt;
96         <!-- Cabeçalho principal que exibe o título da seção ativa.
97             'id="titulo-principal"' é usado para acessar e
98                 modificar este elemento via JavaScript. --&gt;
99
100        &lt;div id="top-3-vendedores" class="melhores-vendedores"&gt;
101            <!-- Div para exibir os três melhores vendedores. Inicialmente
102                visível ao carregar a página.
103                'id="top-3-vendedores"' permite manipulação específica via
104                    JavaScript, como atualizar ou ocultar esta seção. --&gt;
105            &lt;!-- Os dados serão inseridos aqui pelo JavaScript --&gt;
106            &lt;!-- Comentário indicando que o conteúdo dinâmico,
107                detalhes dos vendedores, será adicionado por JavaScript. --&gt;
108
109        &lt;/div&gt;
110</pre>
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

<> index.html > html > body > div.container > div.principal > div#ranking-vendedores.ranking-vendedores

```
111     <div id="ranking-vendedores" class="ranking-vendedores" style="display: none;">
112         <!-- Div para exibir o ranking de todos os vendedores. Oculta
113             por padrão (style="display: none;").
114             'id="ranking-vendedores"' identifica esta seção para
115             operações específicas como exibição baseada em
116             ações do usuário. --&gt;
117         &lt!-- Os dados serão inseridos aqui pelo JavaScript --&gt;
118         &lt!-- Indica que dados dinâmicos serão adicionados aqui pelo
119             JavaScript, como uma lista ou tabela de vendedores
120             e suas vendas. --&gt;
121     &lt;/div&gt;
123
124     &lt;div id="resumo-vendas" class="resumo-vendas" style="display: none;"&gt;
125         <!-- Div para exibir um resumo detalhado das vendas.
126             Também oculta por padrão.
127             'id="resumo-vendas"' é usado para mostrar esta seção
128             através de JavaScript conforme interação do usuário. --&gt;
129
130     &lt;div class="filtros"&gt;
131         <!-- Subdiv que contém campos de entrada para filtrar o resumo
132             das vendas por diferentes critérios. --&gt;
133
134         &lt;input type="text" id="filtro-vendedor" placeholder="Filtrar por Vendedor"&gt;
135         &lt!-- Campo para inserir o nome do vendedor como filtro. --&gt;
136
137         &lt;input type="text" id="filtro-produto" placeholder="Filtrar por Produto"&gt;
138         &lt!-- Campo para inserir o nome do produto como filtro. --&gt;
139</pre>
```

P26 - DASHBOARD DE VENDAS

- > images
- # estilos.css
- ↳ index.html
- 🖼️ p26_screenshot.jpg
- JS scripts.js
- /vnd Vendedor.xlsx

```
↳ index.html > html > body > div.container > div.principal > div#ranking-vendedores.ranking-vendedores  
140     <input type="text" id="filtro-total" placeholder="Filtrar por Total">  
141         <!-- Campo para inserir um valor total como filtro,  
142             para filtrar vendas acima ou abaixo de um certo valor. -->  
143  
144         <button id="exportar-resumo">Exportar para Excel</button>  
145         <!-- Botão para exportar os dados filtrados para um arquivo  
146             Excel. 'id="exportar-resumo"' permite vincular a funcionalidade  
147             de exportação via JavaScript. -->  
148     </div>  
149  
150     <table id="tabela-resumo">  
151         <!-- Tabela para exibir os dados filtrados do  
152             resumo das vendas. -->  
153  
154         <thead>  
155             <!-- Cabeçalho da tabela, definindo as colunas. -->  
156  
157             <tr>  
158                 <!-- Linha de cabeçalho da tabela. -->  
159  
160                 <th>Vendedor</th>  
161                     <!-- Coluna para o nome do vendedor. -->  
162  
163                 <th>Produto</th>  
164                     <!-- Coluna para o nome do produto. -->  
165  
166                 <th>Total</th>  
167                     <!-- Coluna para o total de vendas. -->  
168  
169
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

```
170          </tr>
171      </thead>
172
173      <tbody>
174          |     <!-- Corpo da tabela onde as linhas de dados serão
175          |     | inseridas dinamicamente por JavaScript. --&gt;
176          |     |     <!-- Indica que o conteúdo será adicionado dinamicamente. --&gt;
177
178      &lt;/tbody&gt;
179  &lt;/table&gt;
180&lt;/div&gt;
181<!-- Fim da div 'resumo-vendas'. --&gt;
182
183&lt;div id="detalhes-vendedor" class="detalhes-vendedor" style="display: none;"&gt;
184    |     <!-- Esta div é usada para mostrar detalhes específicos de um
185    |     |     vendedor selecionado, inicialmente oculta
186    |     |     (style="display: none;").
187    |     |     A visibilidade da div pode ser alterada dinamicamente via
188    |     |     JavaScript, baseada em interações do usuário com
189    |     |     outras partes da interface. --&gt;
190
191&lt;button id="voltar-ranking"&gt;Voltar&lt;/button&gt;
192    |     <!-- Botão para voltar à visualização anterior, geralmente o
193    |     |     ranking ou lista principal de vendedores.
194    |     |     O JavaScript pode ser usado para adicionar funcionalidade a
195    |     |     este botão, permitindo aos usuários retornar à tela
196    |     |     anterior sem recarregar a página. --&gt;
197</pre>
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

```
<div id="detalhes-vendedor.detalhes-vendedor">
    <button id="exportar-detalhes">Exportar para Excel</button>
    <!-- Botão para exportar os detalhes do vendedor atual
         para um arquivo Excel.
         Essa funcionalidade é tipicamente implementada via JavaScript
         utilizando uma biblioteca como XLSX.js, que foi
         referenciada anteriormente no <head>. -->

    <table id="tabela-detalhes">
        <!-- Tabela para apresentar informações detalhadas sobre
             vendas de um vendedor específico.
             A tabela é preenchida dinamicamente com dados relevantes,
             como descrito nos elementos <th> abaixo. -->

        <thead>
            <!-- Cabeçalho da tabela, definindo as categorias de dados
                 que serão exibidas em cada coluna. -->

        <tr>
            <!-- Linha de cabeçalho contendo os títulos das colunas. -->

            <th>Vendedor</th>
            <!-- Coluna para o nome do vendedor. -->

            <th>Produto</th>
            <!-- Coluna para o nome do produto vendido. -->

            <th>Total</th>
            <!-- Coluna para o total de vendas do produto pelo vendedor. -->

        </tr>
    </table>
</div>
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

<> index.html X

```
<> index.html > html > body > div.container > div.principal > div#detalhes-vendedor.detalhes-vendedor > table#tabela-detalhes >

228          </thead>
229
230          <tbody>
231
232          <!-- Corpo da tabela onde as linhas de dados específicos
233              serão inseridas dinamicamente pelo JavaScript.
234              Os dados aqui podem incluir várias entradas para um único
235              vendedor, cada uma representando uma venda diferente. --&gt;
236          &lt;!-- Os dados serão inseridos aqui pelo JavaScript --&gt;
237
238          &lt;/tbody&gt;
239      &lt;/table&gt;
240  &lt;/div&gt;
241  &lt;!-- Fim da div 'detalhes-vendedor'. --&gt;
242
243      &lt;/div&gt;
244  &lt;!-- Fim da div 'principal'. --&gt;
245
246  &lt;/div&gt;
247  &lt;!-- Fim da div 'container'. --&gt;
248
249  &lt;script src="scripts.js"&gt;&lt;/script&gt;
250  &lt;!-- Script externo 'scripts.js' incluído no final do corpo para
251      garantir que todos os elementos HTML sejam carregados antes
252      de qualquer manipulação pelo JavaScript. --&gt;
253  &lt;!-- Esse script contém a lógica para controlar a interatividade do
254      dashboard, como eventos de clique, filtragem de dados e
255      exportação para Excel. --&gt;
256
257  &lt;/body&gt;</pre>
```

EXPLORER

...

index.html X

...

<> index.html > html > body

257 </body>

258 </html>

259 <!-- Fim do documento HTML. -->

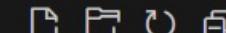
<> index.html

 p26_screenshot.jpg

JS scripts.js

Excel Vendedor.xlsx

P26 - DASHBOARD DE VENDAS



> images

estilos.css

index.html

p26_screenshot.jpg

scripts.js

Vendedor.xlsx

```
1  body {  
2      font-family: Arial, sans-serif;  
3      /* Define a família de fontes do corpo do documento para Arial,  
4          com sans-serif como fallback se Arial não estiver disponível. */  
5  
6      margin: 0;  
7      /* Remove a margem padrão ao redor do <body>, que navegadores  
8          geralmente aplicam. */  
9  
10     padding: 0;  
11     /* Remove o padding padrão interno do <body> para garantir que o  
12        layout utilize todo o espaço da janela do navegador. */  
13  
14     display: flex;  
15     /* Define o modelo de layout do <body> para Flexbox, o que permite  
16        que seus elementos filhos sejam alinhados e distribuídos  
17        flexivelmente. */  
18  
19     height: 100vh;  
20     /* Ajusta a altura do <body> para ser igual à altura da viewport  
21        (100% da altura visível da janela do navegador). */  
22  
23  
24     background-color: #f4f4f9;  
25     /* Define a cor de fundo do <body> para um cinza muito claro (#f4f4f9),  
26        fornecendo uma cor de fundo neutra para o conteúdo. */  
27  
28 }  
29
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

< index.html

p26_screenshot.jpg

scripts.js

Vendedor.xlsx

estilos.css > body

```
30 .container {  
31     display: flex;  
32     /* Usa Flexbox para o contêiner principal, permitindo um layout  
33      | | | responsivo e flexível dos elementos internos. */  
34     width: 100%;  
35     /* Define a largura do contêiner para ocupar 100% da largura de seu  
36      | | | elemento pai (neste caso, o <body>). */  
37 }  
38  
39 .barra-lateral {  
40     width: 20%;  
41     /* Define a largura da barra lateral para 20% da largura de seu  
42      | | | contêiner pai, o que a torna uma barra lateral estreita. */  
43     background-color: #002147;  
44     /* Define a cor de fundo da barra lateral para um azul marinho escuro (#002147),  
45      | | | o que ajuda a destacar a barra lateral do restante do conteúdo. */  
46     color: white;  
47     /* Define a cor do texto dentro da barra lateral para branco,  
48      | | | garantindo uma boa legibilidade contra o fundo escuro. */  
49     padding: 20px;  
50     /* Aplica um padding de 20 pixels em todos os lados dentro da  
51      | | | barra lateral, criando espaço entre o conteúdo da barra  
52      | | | lateral e suas bordas. */  
53 }
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

estilos.css > .barra-lateral

```
60
61 }
62
63
64 .barra-lateral h2 {
65
66     font-size: 1.5em;
67     /* Define o tamanho da fonte do cabeçalho <h2> para 1.5 vezes o
68     | | | tamanho da fonte padrão do elemento pai, tornando-o
69     | | | destacado e visível. */
70
71     text-align: center;
72     /* Centraliza o texto do cabeçalho <h2> horizontalmente, melhorando a
73     | | | estética e a legibilidade. */
74
75     border-bottom: 1px solid ■white;
76     /* Adiciona uma borda sólida branca de 1 pixel na parte inferior do
77     | | | cabeçalho <h2>, funcionando como um divisor visual entre o
78     | | | título e o conteúdo abaixo. */
79
80     padding-bottom: 10px;
81     /* Aplica um espaço de 10 pixels abaixo do texto dentro do cabeçalho <h2>,
82     | | | criando uma separação entre o texto e a borda inferior. */
83
84 }
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

< index.html

p26_screenshot.jpg

scripts.js

Vendedor.xlsx

```
# estilos.css > .barra-lateral ul
86  .barra-lateral ul {
87
88    list-style: none;
89    /* Remove qualquer marcador padrão de lista para <ul>, como discos
90     |   ou círculos, limpa visualmente a área de navegação. */
91
92    padding: 0;
93    /* Remove o padding padrão do <ul>, permitindo que o controle total
94     |   sobre o espaçamento seja definido pelos estilos dos
95     |   elementos <li>. */
96
97 }
98
99 .barra-lateral ul li {
100
101   margin: 20px 0;
102   /* Define a margem vertical para 20 pixels acima e abaixo de cada
103    |   item de lista <li>, ajudando a separar visualmente
104    |   cada item de lista. */
105
106 }
107
108 .barra-lateral ul li a {
109
110   color: white;
111   /* Define a cor do texto dos links dentro dos itens de lista
112    |   para branco, garantindo legibilidade contra o fundo
113    |   escuro da barra lateral. */
114 }
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

estilos.css > .barra-lateral ul

```
115     text-decoration: none;
116     /* Remove a decoração padrão de sublinhado dos links, proporcionando
117      |   |   um visual mais limpo e menos distrativo. */
118
119     font-size: 1.2em;
120     /* Aumenta o tamanho da fonte dos links para 1.2 vezes o tamanho
121      |   |   padrão da fonte do elemento pai, fazendo os links
122      |   |   ficarem mais acessíveis e fáceis de clicar. */
123
124 }
125
126
127 .principal {
128
129     width: 80%;
130     /* Define a largura da div .principal para ocupar 80% da
131      |   |   largura de seu elemento pai. Isso é tipicamente usado
132      |   |   em layouts que incluem uma barra lateral que ocupa o
133      |   |   resto do espaço. */
134
135     padding: 20px;
136     /* Aplica um espaçamento interno de 20 pixels em todos os lados
137      |   |   dentro da div .principal, criando espaço entre o conteúdo
138      |   |   e as bordas da div. */
139
140 }
141
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

< index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

```
# estilos.css > .principal h1
142 .principal h1 {
143
144     text-align: center;
145     /* Centraliza o texto do cabeçalho <h1> horizontalmente, geralmente
146     |   usado para garantir que o título seja o ponto focal
147     |   visual na página. */
148
149     margin-bottom: 20px;
150     /* Aplica uma margem de 20 pixels abaixo do cabeçalho <h1>,
151     |   criando separação entre o título e o conteúdo que segue. */
152
153     font-size: 2em;
154     /* Aumenta o tamanho da fonte para 2 vezes o tamanho da fonte base
155     |   do elemento, tornando o cabeçalho <h1> proeminente e
156     |   mais legível. */
157
158     color: #333;
159     /* Define a cor do texto do cabeçalho para um cinza escuro (#333),
160     |   proporcionando um contraste moderado que é fácil para os olhos. */
161 }
162
163
164 .melhores-vendedores {
165
166     display: flex;
167     /* Utiliza Flexbox para estabelecer um layout flexível, permitindo
168     |   que os elementos filhos (vendedores) sejam facilmente
169     |   organizados. */
170 }
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

< index.html

p26_screenshot.jpg

scripts.js

Vendedor.xlsx

```
# estilos.css > .principal h1
171     justify-content: space-around;
172     /* Distribui espaço igualmente em torno de cada item filho (cada vendedor),
173      garantindo que eles sejam espaçados uniformemente ao
174      longo do eixo horizontal. */
175
176     align-items: center;
177     /* Alinha os itens filhos (vendedores) ao centro no eixo vertical,
178      garantindo que eles estejam alinhados perfeitamente no meio,
179      independentemente de suas alturas individuais. */
180
181 }
182
183 .vendedor {
184
185     text-align: center;
186     /* Centraliza o texto dentro de cada elemento .vendedor, útil
187      para títulos e descrições sob as imagens dos vendedores. */
188
189     margin: 0 20px;
190     /* Aplica uma margem horizontal de 20 pixels em cada lado de cada
191      elemento .vendedor, criando espaço entre vendedores adjacentes. */
192
193 }
194
195 .vendedor img {
196
197     border-radius: 50%;
198     /* Arredonda as bordas da imagem para formar um círculo perfeito,
199      comumente usado para retratos ou avatares de usuário. */
200 }
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

```
# estilos.css > .vendedor img
201     width: 100px;
202     height: 100px;
203     /* Define a largura e altura da imagem para 100 pixels, garantindo
204      que todas as imagens de vendedores tenham o mesmo
205      tamanho e forma. */
206
207 }
208
209 .vendedor .podio {
210
211     display: flex;
212     /* Utiliza Flexbox para o layout do .podio, permitindo um
213      controle fino sobre a disposição dos elementos internos. */
214
215     flex-direction: column;
216     /* Define a direção do Flexbox para coluna, empilhando os
217      elementos filhos verticalmente. */
218
219     align-items: center;
220     /* Centraliza os elementos filhos (como medalhas ou números de
221      posição) ao longo do eixo transversal, garantindo que
222      estejam alinhados ao centro. */
223
224 }
225
226
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

< index.html

p26_screenshot.jpg

scripts.js

Vendedor.xlsx

estilos.css > .vendedor .podio .posicao

```
227 .vendedor .podio .posicao {  
228     width: 100px;  
229     /* Define a largura do elemento .posicao para 100 pixels,  
230      | garantindo uniformidade no tamanho entre  
231      | diferentes vendedores. */  
232  
233     height: 150px;  
234     /* Define a altura inicial para 150 pixels, criando um  
235      | espaço vertical adequado para exibir informações  
236      | ou decorações de ranking. */  
237  
238     display: flex;  
239     /* Usa Flexbox para facilitar o alinhamento dos elementos  
240      | internos (como o parágrafo com a posição do vendedor). */  
241  
242     justify-content: center;  
243     /* Centraliza os elementos filhos horizontalmente dentro  
244      | do elemento .posicao. */  
245  
246     align-items: flex-end;  
247     /* Alinha os elementos filhos ao final do contêiner no eixo  
248      | vertical, útil para posicionar conteúdo na base  
249      | do elemento. */  
250  
251  
252     border-top-left-radius: 10px;  
253     border-top-right-radius: 10px;  
254     /* Arredonda os cantos superiores esquerdo e direito para 10  
255      | pixels, adicionando um toque estético suave. */  
256
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

< index.html

p26_screenshot.jpg

scripts.js

Vendedor.xlsx

```
# estilos.css > .vendedor .podio .posicao
257     margin-top: 20px;
258     /* Adiciona uma margem superior de 20 pixels, separando
259      visualmente este elemento de outros acima dele. */
260
261     position: relative;
262     /* Estabelece um contexto de posicionamento relativo, permitindo
263      que elementos absolutamente posicionados dentro sejam
264      alinhados em relação a este container. */
265
266 }
267
268 .vendedor .podio .posicao p {
269
270     margin: 0;
271     /* Remove qualquer margem padrão do parágrafo, permitindo
272      controle mais preciso sobre o posicionamento. */
273
274     padding: 10px;
275     /* Aplica um preenchimento de 10 pixels em todos os lados do
276      parágrafo, aumentando a legibilidade do texto ao
277      adicionar espaço ao redor. */
278
279     font-size: 1.2em;
280     /* Aumenta o tamanho da fonte para 1.2 vezes o tamanho padrão,
281      destacando visualmente o texto. */
282
283     width: 100%;
284     /* Define a largura do parágrafo para ocupar 100% do seu container
285      pai, garantindo que ele se estenda completamente horizontalmente
286      dentro do elemento .posicao. */
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

```
# estilos.css > .vendedor .podio .posicao p
287
288     text-align: center;
289     /* Centraliza o texto dentro do parágrafo, proporcionando uma
290      |   |   apresentação equilibrada e fácil de ler. */
291
292     position: absolute;
293     /* Posiciona o parágrafo absolutamente dentro de seu contêiner
294      |   |   relativo, permitindo um alinhamento preciso. */
295
296     top: 0;
297     left: 0;
298     /* Alinha o parágrafo no topo e à esquerda do contêiner .posicao,
299      |   |   garantindo que ele se inicie exatamente nos limites
300      |   |   superiores do contêiner. */
301
302 }
303
304 .posicao-1 {
305
306     background-color: #FF6347;
307     /* Define a cor de fundo para um vermelho tomate, destacando
308      |   |   visualmente o vendedor na primeira posição. */
309
310     height: 200px;
311     /* Aumenta a altura para 200 pixels, diferenciando visualmente a
312      |   |   primeira posição como sendo superior. */
313
314 }
315
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

estilos.css > .vendedor .podio .posicao p

```
316  .posicao-2 {  
317    
318      background-color: #ffd700;  
319      /* Define a cor de fundo para um amarelo ouro, indicando  
320      | | | visualmente a segunda posição. */  
321    
322      height: 175px;  
323      /* Define a altura para 175 pixels, um pouco menor que a  
324      | | | primeira posição mas ainda destacada. */  
325    
326 }  
327    
328 .posicao-3 {  
329    
330     background-color: #32CD32;  
331     /* Define a cor de fundo para um verde limão,  
332     | | | associado à terceira posição. */  
333    
334     height: 150px;  
335     /* Mantém a altura inicial de 150 pixels, consistente  
336     | | | com a altura padrão definida para os elementos  
337     | | | de posição. */  
338    
339 }  
340 
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

estilos.css > .ranking-vendedores

```
341 .ranking-vendedores {  
342     display: flex;  
343     /* Utiliza Flexbox para o layout da seção de ranking,  
344      facilitando o alinhamento e a organização  
345      dos itens de ranking. */  
346  
347     flex-direction: column;  
348     /* Configura a direção do Flexbox para 'column', fazendo  
349      com que os itens de ranking sejam organizados  
350      verticalmente. */  
351  
352     align-items: center;  
353     /* Centraliza os itens de ranking horizontalmente dentro da  
354      seção, garantindo que todos estejam alinhados ao meio. */  
355  
356     padding: 20px;  
357     /* Adiciona um espaçamento interno de 20 pixels em todas as  
358      direções dentro da seção de ranking, criando espaço  
359      entre os bordos e os itens. */  
360  
361 }  
362  
363  
364 .ranking-vendedores .item-ranking {  
365  
366     display: flex;  
367     /* Configura Flexbox para cada item de ranking, permitindo um  
368      alinhamento eficiente dos elementos internos,  
369      como imagem e informações. */  
370 }
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

```
# estilos.css > .ranking-vendedores .item-ranking
371     align-items: center;
372     /* Alinha os elementos internos de cada item de ranking
373      ao centro no eixo vertical. */
374
375     background-color: white;
376     /* Define a cor de fundo de cada item de ranking para
377      branco, criando um contraste visual com o
378      conteúdo e o fundo. */
379
380     border-radius: 10px;
381     /* Arredonda os cantos de cada item de ranking para 10
382      pixels, suavizando o design visual. */
383
384     box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
385     /* Aplica uma sombra suave para dar um efeito elevado aos
386      itens de ranking, aumentando a profundidade e o
387      interesse visual. */
388
389     margin-bottom: 10px;
390     /* Adiciona uma margem abaixo de cada item de ranking para
391      separá-los visualmente uns dos outros. */
392
393     width: 60%;
394     /* Define a largura de cada item de ranking para 60% da
395      largura de seu contêiner, garantindo uniformidade. */
396
397     padding: 10px;
398     /* Aplica um espaçamento interno de 10 pixels dentro de cada
399      item de ranking, aumentando o conforto visual ao
400      redor do conteúdo interno. */
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

```
# estilos.css > .ranking-vendedores .item-ranking
401
402     transition: transform 0.3s ease;
403     /* Adiciona uma transição suave para qualquer transformação
404      aplicada (como hover), melhorando a experiência do
405      usuário com feedback visual interativo. */
406
407     cursor: pointer;
408     /* Muda o cursor para um ponteiro quando o usuário passa o
409      mouse sobre um item de ranking, indicando que o
410      item é clicável. */
411
412 }
413
414 .ranking-vendedores .item-ranking:hover {
415
416     transform: translateY(-5px);
417     /* Move o item de ranking 5 pixels para cima quando o mouse
418      está sobre ele, criando um efeito visual de 'levantar',
419      o que pode indicar interatividade. */
420
421 }
422
423 .ranking-vendedores .item-ranking img {
424
425     border-radius: 50%;
426     /* Arredonda a imagem dentro do item de ranking para formar
427      um círculo completo, geralmente usado para avatares
428      ou ícones. */
429 }
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

estilos.css > .ranking-vendedores .item-ranking

```
430     width: 50px;
431     height: 50px;
432     /* Define a largura e a altura da imagem para 50 pixels,
433      |   | garantindo que todas as imagens sejam uniformes em tamanho. */
434
435     margin-right: 15px;
436     /* Adiciona uma margem à direita da imagem, separando-a
437      |   | visualmente das informações textuais ao lado. */
438
439 }
440
441 .ranking-vendedores .item-ranking .info {
442
443     display: flex;
444     /* Utiliza Flexbox dentro do contêiner de informações para
445      |   | alinhar corretamente os elementos de texto. */
446
447     flex-direction: column;
448     /* Configura a direção dos itens de Flexbox para 'column',
449      |   | organizando o texto verticalmente. */
450
451     align-items: flex-start;
452     /* Alinha os elementos de texto ao início do contêiner,
453      |   | garantindo que o texto esteja alinhado à esquerda. */
454
455 }
456
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

< index.html

p26_screenshot.jpg

scripts.js

Vendedor.xlsx

```
# estilos.css > .ranking-vendedores .item-ranking .info p
457 .ranking-vendedores .item-ranking .info p {
458
459     margin: 0;
460     /* Remove todas as margens do parágrafo para permitir um
461      |   |   controle mais preciso do layout e espaçamento. */
462
463     font-size: 1.1em;
464     /* Aumenta ligeiramente o tamanho da fonte para 1.1 vezes o
465      |   |   tamanho normal, destacando o texto principal. */
466
467     color: #333;
468     /* Define a cor do texto para um cinza escuro (#333),
469      |   |   oferecendo boa legibilidade. */
470
471 }
472
473 .ranking-vendedores .item-ranking .info p:nth-child(2) {
474
475     font-size: 1em;
476     /* Define o tamanho da fonte para o tamanho padrão (1em)
477      |   |   para o segundo parágrafo, que é tipicamente
478      |   |   menos proeminente. */
479
480     color: #777;
481     /* Define a cor do texto para um cinza médio (#777),
482      |   |   diferenciando-o do texto principal e indicando
483      |   |   que é uma informação secundária. */
484
485 }
486
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

estilos.css > .resumo-vendas

```
487  .resumo-vendas {  
488  
489      padding: 20px;  
490      /* Adiciona um espaçamento interno de 20 pixels em todas  
491      |      |      as direções dentro da seção .resumo-vendas, criando  
492      |      |      espaço entre o conteúdo e as bordas da div, facilitando a  
493      |      |      leitura e a visualização dos elementos contidos. */  
494  
495  }  
496  
497  .filtros {  
498  
499      display: flex;  
500      /* Utiliza Flexbox para o layout dos elementos de filtro,  
501      |      |      facilitando o alinhamento e a distribuição dos  
502      |      |      controles de filtro (inputs e botão). */  
503  
504      justify-content: space-between;  
505      /* Distribui o espaço entre os elementos filhos de forma  
506      |      |      equitativa, colocando espaço máximo entre eles, o  
507      |      |      que ajuda a separar visualmente cada controle de filtro. */  
508  
509      margin-bottom: 20px;  
510      /* Adiciona uma margem de 20 pixels abaixo do container de  
511      |      |      filtros, separando-o dos outros conteúdos abaixo,  
512      |      |      como tabelas ou outras seções. */  
513  
514  }
```

P26 - DASHBOARD DE VENDAS

- > images
- # estilos.css
- <> index.html
- 🖼 p26_screenshot.jpg
- JS scripts.js
- Excel Vendedor.xlsx

estilos.css > .resumo-vendas

```
516   .filtros input {  
517     padding: 8px;  
518     /* Aplica um preenchimento de 8 pixels em todas as direções  
519      dentro de cada campo de entrada, aumentando a área  
520      clicável e melhorando a legibilidade do texto inserido. */  
521  
522     font-size: 1em;  
523     /* Define o tamanho da fonte para o tamanho padrão (1em), mantendo a  
524      consistência com o estilo de texto geral da interface. */  
525  
526     width: 30%;  
527     /* Define a largura de cada campo de entrada para ocupar 30% da  
528      largura do seu contêiner pai, garantindo que os campos  
529      não se tornem muito largos ou estreitos. */  
530  
531     border: 1px solid #ccc;  
532     /* Adiciona uma borda sólida com 1 pixel de espessura e uma cor  
533      cinza clara (#ccc) em volta dos campos de entrada,  
534      ajudando a definir claramente a área dos campos. */  
535  
536     border-radius: 5px;  
537     /* Arredonda os cantos das bordas dos campos de entrada para 5 pixels,  
538      suavizando o design visual e tornando-o mais  
539      moderno e agradável. */  
540  
541   }  
542  
543
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

estilos.css > .filtros button

```
544  .filtros button {  
545  
546      padding: 10px 20px;  
547      /* Aplica um preenchimento de 10 pixels na vertical e 20 pixels  
548      |     na horizontal no botão, tornando-o fácil de clicar e  
549      |     visualmente proporcional. */  
550  
551      font-size: 1em;  
552      /* Mantém o tamanho da fonte do texto no botão consistente  
553      |     com o dos campos de entrada (1em). */  
554  
555      border: none;  
556      /* Remove qualquer borda padrão do botão, criando um design  
557      |     mais limpo e integrado. */  
558  
559      border-radius: 5px;  
560      /* Arredonda os cantos do botão para 5 pixels, correspondendo ao  
561      |     estilo dos campos de entrada e mantendo a  
562      |     consistência visual. */  
563  
564      background-color: #002147;  
565      /* Define a cor de fundo do botão para um azul marinho  
566      |     profundo (#002147), destacando o botão contra outros  
567      |     elementos e incentivando a interação. */  
568  
569      color: white;  
570      /* Define a cor do texto dentro do botão para branco, garantindo  
571      |     alta legibilidade contra o fundo escuro. */  
572
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

XLS Vendedor.xlsx

```
# estilos.css > .filtros button
573     cursor: pointer;
574     /* Muda o cursor para um ponteiro quando o usuário passa o
575      mouse sobre o botão, indicando que o elemento é clicável. */
576
577 }
578
579 .filtros button:hover {
580
581     background-color: #001F3F;
582     /* Altera a cor de fundo do botão para um azul um pouco
583      mais escuro (#001F3F) quando o mouse está sobre ele,
584      proporcionando feedback visual de que o botão é interativo. */
585
586 }
587
588 .resumo-vendas table {
589
590     width: 100%;
591     /* Estabelece que a tabela ocupe 100% da largura do seu
592      contêiner pai, garantindo que utilize todo o
593      espaço disponível. */
594
595     border-collapse: collapse;
596     /* A propriedade 'border-collapse: collapse' faz com que as
597      bordas das células se fundam em uma única borda, em
598      vez de cada célula ter sua própria borda, resultando
599      em um visual mais limpo e contínuo. */
600 }
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

< index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

estilos.css > .resumo-vendas table

```
601     margin-top: 20px;  
602     /* Adiciona uma margem superior de 20 pixels à tabela, criando  
603      um espaçamento entre quaisquer elementos acima  
604      dela e a tabela em si. */  
605  
606 }  
607  
608 .resumo-vendas table, .resumo-vendas th, .resumo-vendas td {  
609  
610     border: 1px solid #ddd;  
611     /* Aplica uma borda sólida com 1 pixel de espessura e cor  
612      cinza claro (#ddd) em torno da tabela, e cada  
613      célula de cabeçalho (th) e célula de dados (td)  
614      dentro dela, para definir claramente os limites dos dados. */  
615  
616 }  
617  
618 .resumo-vendas th, .resumo-vendas td {  
619  
620     padding: 8px;  
621     /* Adiciona preenchimento de 8 pixels em todas as direções  
622      dentro de cada célula de cabeçalho e célula de dados,  
623      melhorando a legibilidade ao aumentar o espaço ao  
624      redor do texto. */  
625  
626     text-align: left;  
627     /* Alinha o texto dentro das células de cabeçalho e de dados à  
628      esquerda, proporcionando uma apresentação clara e  
629      consistente da informação. */  
630
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

> OUTLINE

> TIMELINE

```
# estilos.css > .resumo-vendas th
631    }
632
633    .resumo-vendas th {
634
635        background-color: #f2f2f2;
636        /* Define a cor de fundo para as células do cabeçalho da
637         |   |   |   tabela para um cinza muito claro (#f2f2f2), distinguindo
638         |   |   |   visualmente o cabeçalho do resto da tabela. */
639
640    }
641
642    .detalhes-vendedor {
643
644        padding: 20px;
645        /* Adiciona preenchimento de 20 pixels em todas as direções
646         |   |   |   dentro da div .detalhes-vendedor, criando espaço
647         |   |   |   entre o conteúdo interno e as bordas da div. */
648
649    }
650
651    .detalhes-vendedor table {
652
653        width: 100%;
654        /* Similar à tabela em .resumo-vendas, faz com que a tabela em
655         |   |   |   .detalhes-vendedor também ocupe 100% da largura
656         |   |   |   do seu contêiner pai. */
657
658        border-collapse: collapse;
659        /* Usa a mesma propriedade de 'border-collapse' para garantir
660         |   |   |   que as bordas das células se fundam em uma única borda. */
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

< index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

```
< index.html # estilos.css X
# estilos.css > .detalhes-vendedor table
661
662     margin-top: 20px;
663     /* Adiciona uma margem superior de 20 pixels, semelhante ao
664      |   |   usado na tabela .resumo-vendas, para manter a
665      |   |   consistência do design. */
666
667 }
668
669 .detalhes-vendedor table, .detalhes-vendedor th, .detalhes-vendedor td {
670
671     border: 1px solid #ddd;
672     /* Aplica bordas cinza claro (#ddd) em torno da tabela e suas
673      |   |   células, assim como na seção .resumo-vendas, mantendo a
674      |   |   uniformidade visual entre diferentes seções. */
675
676 }
677
678 .detalhes-vendedor th, .detalhes-vendedor td {
679
680     padding: 8px;
681     /* Adiciona um preenchimento de 8 pixels em todas as direções
682      |   |   dentro das células de cabeçalho (th) e das
683      |   |   células de dados (td), aumentando a legibilidade ao
684      |   |   separar visualmente o conteúdo das bordas das células. */
685
686     text-align: left;
687     /* Alinha o texto à esquerda dentro das células, proporcionando
688      |   |   uma apresentação clara e consistente da informação,
689      |   |   facilitando a leitura em sequência dos dados tabulares. */
690
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

< index.html

p26_screenshot.jpg

scripts.js

Vendedor.xlsx

```
# estilos.css > .detalhes-vendedor th
691    }
692
693    .detalhes-vendedor th {
694
695        background-color: #f2f2f2;
696        /* Define a cor de fundo das células do cabeçalho para um
697           cinza muito claro (#f2f2f2), destacando visualmente a
698           área do cabeçalho do restante das células da tabela e
699           ajudando a diferenciar categorias de informação. */
700
701    }
702
703    .detalhes-vendedor button {
704
705        background-color: #002147;
706        /* Estabelece a cor de fundo dos botões para um azul
707           marinho profundo (#002147), criando um contraste
708           eficaz com o texto, que é configurado para branco,
709           facilitando a localização e utilização dos botões. */
710
711        color: white;
712        /* Define a cor do texto dentro do botão para branco,
713           garantindo alta legibilidade contra o fundo escuro. */
714
715        padding: 10px 20px;
716        /* Aplica um preenchimento de 10 pixels na vertical e 20
717           pixels na horizontal dentro do botão,
718           tornando-o visualmente proporcional e fácil de interagir. */
719
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

```
# estilos.css > .detalhes-vendedor th
720    border: none;
721    /* Remove qualquer borda padrão do botão, criando um
722       visual mais limpo e integrado ao design geral. */
723
724    border-radius: 5px;
725    /* Arredonda os cantos do botão para 5 pixels, suavizando o
726       design visual e mantendo a consistência com outros
727       elementos de entrada. */
728
729    cursor: pointer;
730    /* Muda o cursor para um ponteiro quando o usuário passa o
731       mouse sobre o botão, indicando interatividade. */
732
733    margin-right: 10px;
734    /* Adiciona uma margem à direita do botão de 10 pixels,
735       separando-o de outros elementos ou botões adjacentes
736       para evitar sobreposição visual. */
737
738 }
739
740 .detalhes-vendedor button:hover {
741
742    background-color: #001F3F;
743    /* Muda a cor de fundo do botão para um azul um pouco
744       mais escuro (#001F3F) quando o mouse está sobre ele,
745       proporcionando feedback visual que o botão é
746       interativo e pode ser clicado. */
747
748 }
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

```
1 // Adiciona um ouvinte de evento que executa a função
2 // quando o documento HTML é completamente carregado.
3 document.addEventListener('DOMContentLoaded', function () {
4     // Essa abordagem garante que o script só será executado após
5     // todo o conteúdo da página estar disponível, prevenindo erros.
6
7     // Inicializa um array para armazenar os
8     // dados dos vendedores.
9     let dadosVendedores = [];
10
11    // Realiza uma requisição para obter o arquivo
12    // Excel 'Vendedor.xlsx'.
13    fetch('Vendedor.xlsx') // O caminho do arquivo Excel
14    // deve ser ajustado conforme necessário.
15
16    .then(response => response.arrayBuffer())
17    // A resposta da requisição é transformada em um ArrayBuffer, que é
18    // uma representação genérica de dados binários.
19
20    .then(data => {
21        // Utiliza a biblioteca XLSX para ler os
22        // dados do ArrayBuffer.
23
24        const workbook = XLSX.read(data, { type: 'array' });
25        // 'workbook' refere-se ao arquivo Excel inteiro.
26
27        // Acessa o nome da primeira aba da planilha.
28        const nomeDaAba = workbook.SheetNames[0];
29        // 'SheetNames' é um array que contém os nomes de todas as
30        // abas na planilha; aqui, seleciona-se a primeira.
```

JS scripts.js > ⚡ document.addEventListener('DOMContentLoaded') callback > ⚡ then() callback

```
31          // Acessa os dados da aba especificada.
32          const aba = workbook.Sheets[nomeDaAba];
33          // 'Sheets' é um objeto que contém todas as abas como
34          // propriedades, acessíveis pelo nome da aba.
35
36
37          // Converte os dados da aba para um formato JSON para
38          // facilitar o manuseio.
39          dadosVendedores = XLSX.utils.sheet_to_json(aba);
40          // 'sheet_to_json' é um método que transforma os dados
41          // da planilha em um array de objetos JSON.
42
43          // Inicializa um objeto para armazenar as vendas
44          // totais por vendedor.
45          const vendas = {};
46
47          // Itera sobre cada item na lista de dados dos
48          // vendedores obtidos do arquivo Excel.
49          dadosVendedores.forEach(item => {
50              // Para cada 'item', que representa um registro de
51              // venda individual, as seguintes operações
52              // são realizadas:
53
54              const vendedor = item.Vendedor;
55              // Extrai o nome do vendedor do objeto 'item'. 'Vendedor' é
56              // uma chave no objeto que identifica o vendedor.
57
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

JS scripts.js > document.addEventListener('DOMContentLoaded') callback > then() callback > dadosVendedores.forEach() callback

```
58     const total = parseFloat(item.Total); // Converter para número
59     // Extrai o valor total de vendas do objeto 'item', que
60     // está armazenado como string, e converte para
61     // um número flutuante.
62     // 'parseFloat' é usado para garantir que o valor seja
63     // tratado como um número, o que é necessário para
64     // realizar operações matemáticas.

65
66     // Verifica se o vendedor já existe no objeto 'vendas'.
67     if (vendas[vendedor]) {
68
69         // Se o vendedor já está registrado no objeto 'vendas':
70         vendas[vendedor] += total;
71         // Adiciona o valor total de vendas atual ao valor acumulado
72         // anteriormente para esse vendedor.
73         // Isso é feito usando '+=' que é um operador de atribuição
74         // que soma o valor à direita ao valor já existente na variável.
75
76     } else {
77
78         // Se o vendedor não está registrado no objeto 'vendas':
79         vendas[vendedor] = total;
80         // Inicializa o total de vendas para esse
81         // vendedor no objeto 'vendas'.
82         // Aqui, o vendedor é adicionado como uma nova
83         // chave no objeto com seu total de vendas como valor.
84
85     }
86
87 );
```

P26 - DASHBOARD DE VENDAS

- > images
- # estilos.css
- <> index.html
- 🖼️ p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

> OUTLINE

> TIMELINE

JS scripts.js > ⚡ document.addEventListener('DOMContentLoaded') callback > ⚡ then() callback > ⚡ dadosVendedores.forEach() callback

```
89         // Converter para um array de objetos e ordenar
90             // pelo total de vendas
91             const vendasOrdenadas = Object.entries(vendas)
92
93                 .map(([vendedor, total]) => ({ vendedor, total }))
94             /* 'Object.entries(vendas)' converte o objeto 'vendas' em
95                 um array de arrays, onde cada sub-array contém [chave, valor],
96                 ou seja, [nome do vendedor, total de vendas].
97             O método '.map()' transforma cada sub-array em um objeto
98                 com as propriedades 'vendedor' e 'total'.
99             Isto é útil para manipulações subsequentes, como ordenação,
100                que são mais intuitivas quando os dados estão em
101                    formato de objeto. */
102
103             .sort((a, b) => b.total - a.total);
104             /* A função '.sort()' é usada para ordenar o array de
105                 objetos baseado no 'total' de vendas.
106             A função de comparação toma dois objetos 'a' e 'b'.
107             Subtraindo 'total' de 'b' de 'total' de 'a',
108                 a ordenação é feita em ordem decrescente, ou seja,
109                     os vendedores com maiores vendas aparecem primeiro no array. */
110
111         // Pegar os top 3 vendedores
112         const topVendedores = vendasOrdenadas.slice(0, 3);
113             /* 'slice(0, 3)' é usado para extrair os três primeiros
114                 elementos do array 'vendasOrdenadas'.
115             Isso retorna um novo array contendo apenas os três
116                 vendedores com as maiores vendas, que são
117                     considerados os top 3 vendedores.
118             Esta operação não modifica o array original 'vendasOrdenadas'. */
```

JS scripts.js > ⚡ document.addEventListener('DOMContentLoaded') callback > ⚡ then() callback

```
119
120
121    // Seleciona o elemento HTML pelo ID 'top-3-vendedores'
122    // para ser o contêiner onde os dados dos
123    // vendedores serão exibidos.
124    const containerTopVendedores = document.querySelector('#top-3-vendedores');
125
126    // Atualiza o conteúdo HTML do contêiner selecionado
127    // com os dados dos top 3 vendedores.
128    containerTopVendedores.innerHTML = topVendedores.map((vendedor, index) =>
129
130        <!-- Cria um div para cada vendedor que inclui uma imagem,
131            informações de posição no pódio, nome e total de vendas --&gt;
132        &lt;div class="vendedor"&gt;
133
134            <!-- Imagem do vendedor: utiliza o nome do vendedor para
135                gerar o caminho da imagem dinamicamente --&gt;
136            &lt;img src="images/${vendedor.vendedor}.jpg" alt="${vendedor.vendedor}"&gt;
137
138            <!-- Div que agrupa elementos relacionados à classificação do vendedor --&gt;
139            &lt;div class="podio"&gt;
140
141                <!-- Div que mostra a posição do vendedor com uma classe que
142                    pode variar dependendo do índice para estilização específica --&gt;
143                &lt;div class="posicao posicao-${index + 1}"&gt;
144
145                    <!-- Parágrafo que exibe o número da posição do vendedor --&gt;
146                    &lt;p&gt;${index + 1}&lt;/p&gt;
147
148            &lt;/div&gt;</pre>
```

✓ P26 - DASHBOARD DE VENDAS

> images

estilos.css

< index.html

🖼 p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

scripts.js > document.addEventListener('DOMContentLoaded') callback > then() callback > topVendedores.map() callback

```
149          <!-- Cabeçalho que mostra o nome do vendedor -->
150          <h3>${vendedor.vendedor}</h3>
151
152          <!-- Parágrafo que mostra o total de vendas do
153              | vendedor formatado como moeda brasileira -->
154          <p>${vendedor.total.toLocaleString('pt-BR', { style: 'currency', currency:
155              | 'BRL' })}</p>
156
157          </div>
158      </div>
159  `).join(''); // Combina todos os elementos do array em uma única
160          | // string HTML sem separador entre eles.
161
162
163  // Seleciona o contêiner no DOM onde o ranking completo dos
164      | // vendedores será exibido.
165  const containerRankingVendedores = document.querySelector('#ranking-vendedores');
166
167  // Atualiza o HTML interno do contêiner com os dados do
168      | // ranking completo de vendedores.
169  containerRankingVendedores.innerHTML = vendasOrdenadas.map((vendedor, index) =>
170
171          <!-- Cria um div para cada vendedor com uma classe específica e um
172              | atributo data que armazena o nome do vendedor para
173              | referência futura. -->
174          <div class="item-ranking" data-vendedor="${vendedor.vendedor}">
```

JS scripts.js > ⚡ document.addEventListener('DOMContentLoaded') callback > ⚡ then() callback > ⚡ vendasOrdenadas.map() callback

```
176     <!-- Imagem do vendedor: utiliza o nome do vendedor para gerar
177         | dinamicamente o caminho da imagem -->
178     
179
180     <!-- Div que agrupa elementos de texto com
181         | informações do vendedor -->
182     <div class="info">
183
184         <!-- Parágrafo que mostra a posição do vendedor no
185             | ranking seguida pelo nome do vendedor -->
186         <p>${index + 1} - ${vendedor.vendedor}</p>
187
188         <!-- Parágrafo que mostra o total de vendas do vendedor
189             | formatado como moeda brasileira -->
190         <p>${vendedor.total.toLocaleString('pt-BR', { style: 'currency', currency:
191             | 'BRL' })}</p>
192
193     </div>
194     `).join(''); // Combina todos os elementos do array em uma única
195         | // string HTML sem separador entre eles.
196
197
198     // Carregar resumo completo
199     // Seleciona o corpo da tabela dentro do elemento com o ID 'tabela-resumo'
200         | // para inserir os dados das vendas.
201     const containerResumoVendas = document.querySelector('#tabela-resumo tbody');
202
203     // Atualiza o conteúdo HTML do corpo da tabela com os dados de cada venda.
204     containerResumoVendas.innerHTML = dadosVendedores.map(venda => `
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

JS scripts.js > document.addEventListener('DOMContentLoaded') callback > then() callback > dadosVendedores.map() callback

```
205
206          <!-- Cria uma linha de tabela (tr) para cada objeto de
207          | venda no array dadosVendedores --&gt;
208          &lt;tr&gt;
209
210          <!-- Célula de tabela (td) contendo o nome do vendedor --&gt;
211          &lt;td&gt;${venda.Vendedor}&lt;/td&gt;
212
213          <!-- Célula de tabela contendo o nome do produto vendido --&gt;
214          &lt;td&gt;${venda.Produto}&lt;/td&gt;
215
216          <!-- Célula de tabela contendo o total das vendas formatado
217          | como moeda do Brasil (Real) --&gt;
218          &lt;td&gt;${venda.Total.toLocaleString('pt-BR', { style: 'currency', currency: 'BRL' })}&lt;/
219          td&gt;
220
221          &lt;/tr&gt;
222      `).join(''); // Junta todos os elementos do array resultante em uma
223      | // string única, sem separador, formando o HTML para inserir na tabela.
224  }
225</pre>
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

JS scripts.js > document.addEventListener('DOMContentLoaded') callback > then() callback > dadosVendedores.map() callback

```
226     // Adiciona um tratamento de erro para o caso de falha ao
227     // carregar ou processar o arquivo Excel.
228     .catch(error => console.error('Erro ao carregar o arquivo Excel:', error));
229     /* O método .catch() é chamado se uma exceção é lançada em qualquer
230      parte do encadeamento de Promises (promessas),
231      aqui captura e registra erros que podem ocorrer durante a
232      requisição do arquivo Excel ou seu processamento.
233      'console.error' é usado para imprimir a mensagem de erro no
234      console, facilitando o diagnóstico de problemas. */
235
236
237     // Adiciona um ouvinte de eventos para lidar com cliques no
238     // botão que mostra os top 3 vendedores.
239     document.getElementById('mostrar-top-3').addEventListener('click', () => {
240
241         // Muda o título principal da página para refletir a
242         // visualização selecionada.
243         document.getElementById('titulo-principal').innerText = 'Melhores Vendedores do Mês';
244
245         // Ajusta a visibilidade dos contêineres de conteúdo para
246         // mostrar apenas os top 3 vendedores.
247         document.getElementById('top-3-vendedores').style.display = 'flex'; // Mostra os top 3
248         vendedores.
249         document.getElementById('ranking-vendedores').style.display = 'none'; // Oculta o ranking
250         completo de vendedores.
251         document.getElementById('resumo-vendas').style.display = 'none'; // Oculta o resumo de vendas.
252         document.getElementById('detalhes-vendedor').style.display = 'none'; // Oculta os detalhes dos
253         vendedores.
254     });
255
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

JS scripts.js > document.addEventListener('DOMContentLoaded') callback

```
253     // Adiciona um ouvinte de eventos para lidar com cliques no
254     |     // botão que mostra o ranking completo de vendedores.
255     document.getElementById('mostrar-ranking').addEventListener('click', () => {
256
257         // Muda o título principal da página para 'Ranking de Vendas'.
258         document.getElementById('titulo-principal').innerText = 'Ranking de Vendas';
259
260         // Ajusta a visibilidade dos contêineres de conteúdo para mostrar
261         |     // apenas o ranking de vendedores.
262         document.getElementById('top-3-vendedores').style.display = 'none'; // Oculta os top 3
263         vendedores.
264         document.getElementById('ranking-vendedores').style.display = 'flex'; // Mostra o ranking
265         completo de vendedores.
266         document.getElementById('resumo-vendas').style.display = 'none'; // Oculta o resumo de vendas.
267         document.getElementById('detalhes-vendedor').style.display = 'none'; // Oculta os detalhes dos
268         vendedores.
269
270         });
271
272         // Adiciona um ouvinte de eventos para lidar com cliques no
273         |     // botão que mostra o resumo de vendas.
274         document.getElementById('mostrar-resumo').addEventListener('click', () => {
275
276             // Muda o título principal da página para 'Resumo de Vendas'.
277             document.getElementById('titulo-principal').innerText = 'Resumo de Vendas';
278
279             // Ajusta a visibilidade dos contêineres de conteúdo para
280             |     // mostrar apenas o resumo de vendas.
281             document.getElementById('top-3-vendedores').style.display = 'none'; // Oculta os top 3
282             vendedores.
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

```
JS scripts.js > ⚡ document.addEventListener('DOMContentLoaded') callback > ⚡ addEventListener('click') callback
279     document.getElementById('ranking-vendedores').style.display = 'none'; // Oculta o ranking
280     completo de vendedores.
281     document.getElementById('resumo-vendas').style.display = 'block'; // Mostra o resumo de vendas.
282     document.getElementById('detalhes-vendedor').style.display = 'none'; // Oculta os detalhes dos
283     vendedores.
284
285
286     // Adiciona um ouvinte de evento de clique ao container
287     // que lista todos os vendedores.
288     document.querySelector('#ranking-vendedores').addEventListener('click', (event) => {
289
290         // O metodo 'closest' é usado para encontrar o elemento
291         // ascendente mais próximo que corresponde ao
292         // seletor '.item-ranking'.
293         // Isso é útil para garantir que o clique dentro de
294         // elementos filhos também seja considerado.
295         const item = event.target.closest('.item-ranking');
296
297         if (item) {
298
299             // Extraí o nome do vendedor do atributo 'data-vendedor' do item clicado.
300             const vendedorNome = item.getAttribute('data-vendedor');
301
302             // Filtra os dados de todos os vendedores para obter
303             // apenas aqueles que correspondem ao vendedor selecionado.
304             const detalhesVendedor = dadosVendedores.filter(venda => venda.Vendedor === vendedorNome);
305
```

JS scripts.js > document.addEventListener('DOMContentLoaded') callback > addEventListener('click') callback

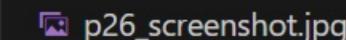
```
306     // Seleciona o corpo da tabela onde os detalhes do
307     // vendedor serão exibidos.
308     const tabelaDetalhes = document.querySelector('#tabela-detalhes tbody');
309
310     // Atualiza o conteúdo HTML da tabela com os detalhes
311     // das vendas do vendedor específico.
312     tabelaDetalhes.innerHTML = detalhesVendedor.map(venda =>
313         <tr>
314             <td>${venda.Vendedor}</td>
315             <td>${venda.Produto}</td>
316             <td>${parseFloat(venda.Total).toFixed(2)}</td>
317         </tr>
318     ).join('');
319
320     // Atualiza o título principal para refletir a
321     // visualização dos detalhes do vendedor.
322     document.getElementById('titulo-principal').innerText = `Detalhes de Vendas - ${vendedorNome}`;
323
324     // Ajusta a visibilidade dos diferentes contêineres
325     // para mostrar apenas os detalhes do vendedor.
326     document.getElementById('top-3-vendedores').style.display = 'none';
327     document.getElementById('ranking-vendedores').style.display = 'none';
328     document.getElementById('resumo-vendas').style.display = 'none';
329     document.getElementById('detalhes-vendedor').style.display = 'block';
330
331 }
332 });
333
334 
```

✓ P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html



JS scripts.js

Vendedor.xlsx

JS scripts.js >  document.addEventListener('DOMContentLoaded') callback

```
335 // Adiciona um ouvinte de evento de clique ao
336     // botão 'voltar-ranking'.
337     document.getElementById('voltar-ranking').addEventListener('click', () => {
338
339         // Quando o botão é clicado, o título principal da
340             // página é alterado para 'Ranking de Vendas'.
341         document.getElementById('titulo-principal').innerText = 'Ranking de Vendas';
342
343         // Os seguintes comandos ajustam a visibilidade dos
344             // contêineres de conteúdo da página:
345         document.getElementById('top-3-vendedores').style.display = 'none'; // Oculta a seção dos top 3
346             vendedores.
347         document.getElementById('ranking-vendedores').style.display = 'flex'; // Exibe o ranking de
348             vendedores.
349         document.getElementById('resumo-vendas').style.display = 'none'; // Oculta o resumo de vendas.
350         document.getElementById('detalhes-vendedor').style.display = 'none'; // Oculta os detalhes do
351             vendedor.
352
353     });
354
355     document.querySelectorAll('.filtros input').forEach(input => {
356
357         input.addEventListener('input', filtrarTabelaResumo);
358
359     });
360
361     const filtroVenda = document.querySelector('#filtro-venda');
362     const filtroVendedor = document.querySelector('#filtro-vendedor');
```

JS scripts.js > ⚡ document.addEventListener('DOMContentLoaded') callback

```
360     // Adiciona ouvintes de eventos a todos os campos de
361         // entrada dentro do contêiner '.filtros'.
362     document.querySelectorAll('.filtros input').forEach(input => {
363
364         // Para cada campo de entrada, adiciona um ouvinte
365             // para o evento 'input'.
366         input.addEventListener('input', filtrarTabelaResumo);
367
368         // 'filtrarTabelaResumo' é uma função que será chamada cada vez
369             // que o usuário alterar o texto em qualquer um
370                 // dos campos de entrada.
371             // Esta função é responsável por filtrar os dados na tabela
372                 // de resumo com base no texto inserido.
373
374     });
375
376
377     function filtrarTabelaResumo() {
378
379         // Obtém o valor do campo de filtro do vendedor e o
380             // converte para letras minúsculas para ignorar a
381                 // diferença entre maiúsculas e minúsculas.
382         const filtroVendedor = document.getElementById('filtro-vendedor').value.toLowerCase();
383
384         // Obtém o valor do campo de filtro do produto e o converte
385             // para letras minúsculas para ignorar a diferença
386                 // entre maiúsculas e minúsculas.
387         const filtroProduto = document.getElementById('filtro-produto').value.toLowerCase();
388
```

P26 - DASHBOARD DE VENDAS

- > images
- # estilos.css
- <> index.html
- 🖼️ p26_screenshot.jpg
- JS scripts.js

Vendedor.xlsx

JS scripts.js > ⚡ document.addEventListener('DOMContentLoaded') callback

```
389     // Obtém o valor do campo de filtro total e o converte para
390     // letras minúsculas para ignorar a diferença
391     // entre maiúsculas e minúsculas.
392     const filtroTotal = document.getElementById('filtro-total').value.toLowerCase();
393
394     // Seleciona todas as linhas da tabela resumo para verificar
395     // cada uma contra os filtros aplicados.
396     const linhas = document.querySelectorAll('#tabela-resumo tbody tr');
397
398     // Itera sobre cada linha da tabela para aplicar os filtros.
399     linhas.forEach(linha => {
400
401         // Obtém o texto do vendedor da linha corrente e o
402         // converte para minúsculas.
403         const vendedor = linha.children[0].innerText.toLowerCase();
404
405         // Obtém o texto do produto da linha corrente e o
406         // converte para minúsculas.
407         const produto = linha.children[1].innerText.toLowerCase();
408
409         // Obtém o texto do total da linha corrente e o
410         // converte para minúsculas.
411         const total = linha.children[2].innerText.toLowerCase();
412
413         // Verifica se o texto do vendedor na linha contém o
414         // texto filtrado pelo usuário.
415         const correspondeVendedor = vendedor.includes(filtroVendedor);
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

```
JS scripts.js > ⚡ document.addEventListener('DOMContentLoaded') callback > ⚡ filtrarTabelaResumo > ⚡ linhas.forEach() callback
417     // Verifica se o texto do produto na linha contém o
418     // texto filtrado pelo usuário.
419     const correspondeProduto = produto.includes(filtroProduto);
420
421     // Verifica se o texto do total na linha contém o
422     // texto filtrado pelo usuário.
423     const correspondeTotal = total.includes(filtroTotal);
424
425     // Se todos os filtros correspondem, a linha é
426     // exibida, caso contrário, é ocultada.
427     if (correspondeVendedor && correspondeProduto && correspondeTotal) {
428
429         // A linha é mantida visível se todos os filtros correspondem.
430         linha.style.display = '';
431
432     } else {
433
434         // A linha é ocultada se algum filtro não corresponde.
435         linha.style.display = 'none';
436
437     }
438 });
439 }
440
441 // Adiciona um ouvinte de evento de clique ao botão 'exportar-resumo'.
442 document.getElementById('exportar-resumo').addEventListener('click', () => {
443
```

JS scripts.js > document.addEventListener('DOMContentLoaded') callback > addEventListener('click') callback

```
444     // Chama a função 'exportarTabelaParaExcel' quando o botão é clicado.
445     // Passa o ID da tabela que contém o resumo das vendas e o
446     // nome do arquivo que será gerado.
447     exportarTabelaParaExcel('tabela-resumo', 'resumo_vendas.xlsx');
448     // 'tabela-resumo' é o ID da tabela de onde os
449     // dados serão exportados.
450     // 'resumo_vendas.xlsx' é o nome do arquivo Excel
451     // que será gerado e oferecido para download.
452
453 );
454
455
456 // Adiciona um ouvinte de evento de clique ao botão 'exportar-detalhes'.
457 document.getElementById('exportar-detalhes').addEventListener('click', () => {
458     // Chama a função 'exportarTabelaParaExcel' quando o
459     // botão é clicado.
460     // Passa o ID da tabela que contém os detalhes das
461     // vendas de um vendedor específico e o nome do
462     // arquivo que será gerado.
463
464     exportarTabelaParaExcel('tabela-detalhes', 'detalhes_vendas.xlsx');
465     // 'tabela-detalhes' é o ID da tabela de onde os dados serão exportados.
466     // 'detalhes_vendas.xlsx' é o nome do arquivo Excel que será
467     // gerado e oferecido para download.
468
469 );
470
471
```

P26 - DASHBOARD DE VENDAS

> images

estilos.css

<> index.html

p26_screenshot.jpg

JS scripts.js

Vendedor.xlsx

JS scripts.js > document.addEventListener('DOMContentLoaded') callback

```
472 // Define a função 'exportarTabelaParaExcel' que aceita
473 // dois parâmetros: o ID da tabela HTML e o nome do
474 // arquivo Excel a ser criado.
475 function exportarTabelaParaExcel(tabelaId, nomeArquivo) {
476
477     // Obtém a tabela pelo seu ID do DOM.
478     const tabela = document.getElementById(tabelaId);
479
480     // Obtém todas as linhas ('tr') da tabela e as converte em
481     // um array para facilitar a manipulação.
482     const linhas = Array.from(tabela.querySelectorAll('tr'));
483
484     // Clona a tabela original para evitar alterações na
485     // exibição atual no DOM.
486     // 'cloneNode(true)' é utilizado para fazer uma cópia
487     // profunda, incluindo o elemento e todos os seus filhos.
488     const tabelaClone = tabela.cloneNode(true);
489
490     // Da mesma forma que com a tabela original, obtém todas
491     // as linhas do clone em forma de array.
492     const linhasClone = Array.from(tabelaClone.querySelectorAll('tr'));
493
494     // Utiliza a biblioteca XLSX para converter a tabela
495     // clonada em um livro Excel ('workbook').
496     // 'table_to_book' é um método que aceita uma tabela HTML e
497     // opções, criando um objeto de livro que representa um arquivo Excel.
498     const workbook = XLSX.utils.table_to_book(tabelaClone, { sheet: "Sheet1" });
499
```

EXPLORER ... <> index.html # estilos.css JS scripts.js X

...

✓ P26 - DASHBOARD DE VENDAS

- > images
- # estilos.css
- <> index.html
- ☒ p26_screenshot.jpg
- JS scripts.js
- ☒ Vendedor.xlsx

JS scripts.js > ⚒ document.addEventListener('DOMContentLoaded') callback > ⚒ exportarTabelaParaExcel

```
500     // Escreve o 'workbook' como um arquivo Excel físico e
501     // inicia o download.
502     // 'writeFile' é um método que aceita o 'workbook' e o
503     // nome do arquivo, salvando o arquivo no sistema do usuário.
504     XLSX.writeFile(workbook, nomeArquivo);
505
506 }
507
508});
```