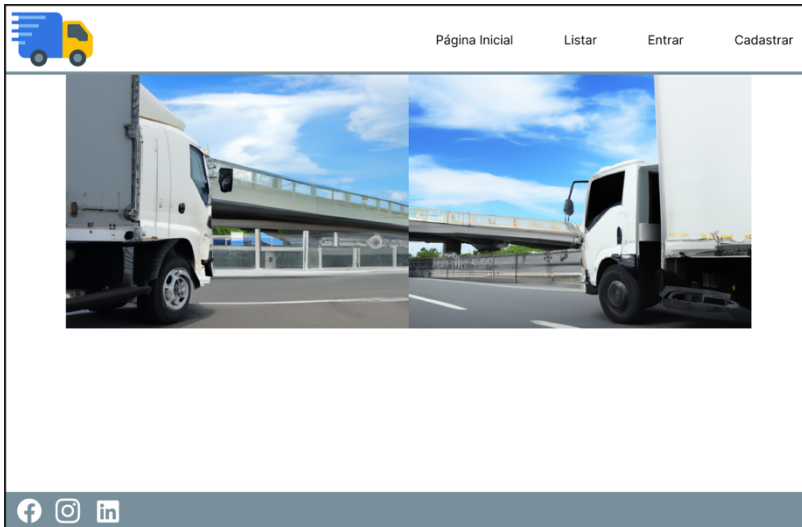
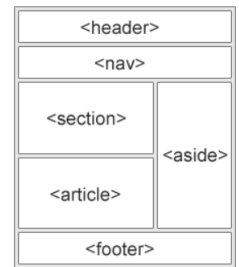


Nesta aula, vamos criar o HTML e o CSS do desenho que fizemos no figma. Para isso, vamos observar o desenho.



Conseguimos dividir de forma clara a área do cabeçalho, o conteúdo principal do site que estão com duas imagens e o rodapé. Para fazermos o HTML é importante utilizarmos as tags semânticas referente a cada região.

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>



Neste caso o arquivo HTML ficou com o conteúdo ao lado.

Observe que as três áreas principais foram separadas nas tags <header>, <main> e <footer>.

Dentro da tag <header> nós colocamos a tag <figure> para acomodar a logo e a tag <nav> para fazermos o menu de navegação.

Dentro da tag <nav> utilizamos a tag de lista para implementarmos os itens do menu, característica semântica importante.

Outra configuração importante que precisamos fazer foi colocar a tag <section> com a propriedade class img, para diferenciar as páginas, dentro da <main> e colocar as imagens dentro da <section> para utilizarmos como container flexbox para alinharmos as uma ao lado da outra.

A tag <section> neste caso foi importante também, para definirmos uma largura máxima para o conteúdo central do site para assim, alinharmos ao meio, o que estiver dentro dela.

Existem várias formas de realizar a mesma configuração. Estas inclusões de tags devem ser feitas sempre que necessário, para isso, é importante aprender como os recursos do HTML e do CSS funcionam.

```

index.html x
index.html > html
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5    <meta charset="UTF-8">
6    <meta http-equiv="X-UA-Compatible" content="IE=edge">
7    <meta name="viewport" content="width=device-width, initial-scale=1.0">
8    <title>Página Inicial</title>
9    <link rel="stylesheet" href="./css/estilo.css" />
10 </head>
11
12 <body>
13   <header>
14     <picture>
15       
16     </picture>
17     <nav>
18       <ul>
19         <li>
20           <a href="index.html">Página Inicial</a>
21         </li>
22         <li>
23           <a href="lista.html">Lista</a>
24         </li>
25         <li>
26           <a href="#">Entrar</a>
27         </li>
28         <li>
29           <a href="#">Cadastrar</a>
30         </li>
31       </ul>
32     </nav>
33   </header>
34   <main>
35     <section class="img">
36       
37       
38     </section>
39   </main>
40   <footer>
41     
42     
43     
44   </footer>
45 </body>
46
47 </html>
  
```

```
estilo.css x
css > estilo.css > ...
1 @import url('https://fonts.googleapis.com/css2?family=Inter:wght@400;700&display=swap');
2 * {
3   margin: 0;
4   padding: 0;
5   outline: 0;
6   box-sizing: border-box;
7   font-family: 'Inter', sans-serif;
8   font-weight: 400;
9 }
10 :root {
11   --primary: #2B75E2;
12   --secondary: #FFC107;
13   --secondary-light: #f0dea5;
14   --third: #78909C;
15   --third-light: #d9d9d9;
16   --black: #000;
17   --white: #FFF;
18 }
19 /* Cabeçalho */
20 header {
21   border-bottom: 0.4rem solid var(--third);
22   display: flex;
23   height: 6rem;
24 }
25 header picture {
26   margin-left: 0.5rem;
27   padding: 0.5rem;
28 }
29 header picture img {
30   height: 100%;
31 }
32 /* Navegação */
33 header nav {
34   display: flex;
35   align-items: center;
36   width: 100%;
37   justify-content: right;
38 }
39 header nav ul {
40   list-style-type: none;
41   display: flex;
42   align-items: center;
43   height: 100%;
44 }
45 header nav ul li {
46   padding: 1rem;
47   height: 100%;
48   display: flex;
49   align-items: center;
50 }
51 header nav ul li a {
52   text-decoration: none;
53   color: var(--black);
54 }
55 header nav ul li:hover {
56   background-color: var(--secondary-light);
57 }
58 /* Principal */
59 main {
60   min-height: calc(100vh - 9rem);
61 }
62 main section {
63   display: flex;
64   max-width: 1090px;
65   margin: 0 auto;
66 }
67 main section img img {
68   width: 50%;
69 }
```

Vamos para o CSS. Começando pelas configurações globais.

O comando @import permite que se carregue um CSS externo ao nosso site. Neste caso, o CSS da fonte do Google que utilizei no meu projeto.

O seletor * permite que se defina propriedades para todos os seletores.

O seletor :root define configurações globais.

A função var() permite aplicar um valor baseado em um nome. Este nome deve começar sempre com (--). No nosso projeto nós definimos as cores como variáveis, mas pode ser utilizado para qualquer configuração.

A parte do cabeçalho fizemos a borda embaixo, definimos sua altura, fixamos o espaço à esquerda da logo e entre a logo e

as bordas e configuramos para que a imagem da logo fique sempre com 100% da altura, para que ela ajuste de acordo com a altura do cabeçalho.

Na navegação a tag <nav> ocupa todo o espaço ao lado da logo e ela é o container para alinhar o menu à direita.

A tag é o container dos , os quais configuramos com flexbox.

A tag também precisou ser flexbox para alinharmos o texto que está dentro dela.

A tag <a> que possui o link dos menus nós definimos a cor e retiramos o underline padrão.

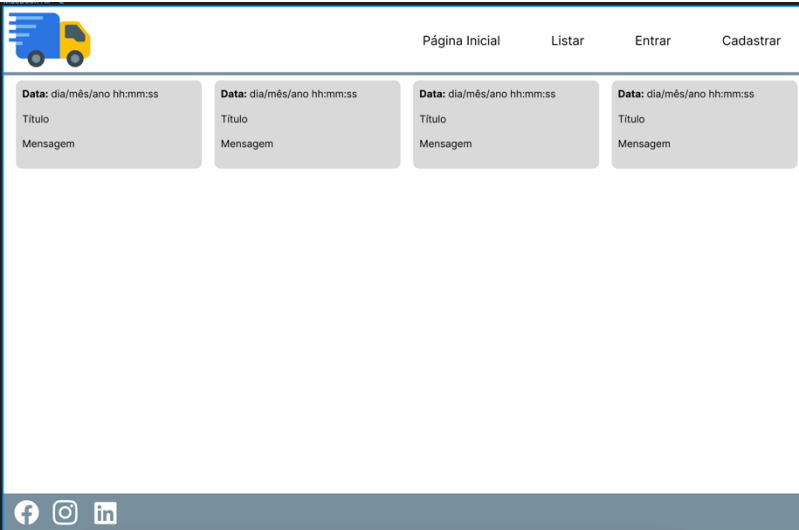
Para o efeito de passar o mouse, resolvemos fazer isso em todo o , trocando o fundo para um amarelo claro.

Para esta tela principal, utilizamos um recurso de calcular a altura mínima da tela subtraindo toda a altura pela altura do cabeçalho somada com a do rodapé.

Para alinhar as imagens no centro, definimos uma largura máxima de 1090px. A propriedade margin auto permitiu que as margens da esquerda e direita sejam aplicadas assim que a largura da tela ultrapassar a largura máxima.

As imagens ficaram com 50% de largura cada.

Antes de apresentar o CSS do rodapé, vamos trabalhar nas diferenças da parte principal da página que lista os quadrados.



As diferenças que teremos serão somente na parte do meio que faremos com grid. Não vou colocar todo o HTML, porque ele é igual ao da página index.html.

Para criar a grid, utilizamos a tag <section> como container e o grid-template-columns com a quantidade de auto de acordo com o número de colunas.

A tag <aside> faz o quadrado e utilizamos o h3 com strong para escolher uma região com o negrito.

```

42 <main>
43   <section class="list">
44     <aside>
45       <h3>
46         <strong>Data: </strong> dia/mês/ano hh:mm:ss
47       </h3>
48       <h4>Título</h4>
49       <p>Mensagem</p>
50     </aside>
51     <aside>
52       <h3>
53         <strong>Data: </strong> dia/mês/ano hh:mm:ss
54       </h3>
55       <h4>Título</h4>
56       <p>Mensagem</p>
57     </aside>
58     <aside>
59       <h3>
60         <strong>Data: </strong> dia/mês/ano hh:mm:ss
61       </h3>
62       <h4>Título</h4>
63       <p>Mensagem</p>
64     </aside>
65     <aside>
66       <h3>
67         <strong>Data: </strong> dia/mês/ano hh:mm:ss
68       </h3>
69       <h4>Título</h4>
70       <p>Mensagem</p>
71     </aside>
72   </section>
73 </main>
  
```

```

70 /* Listar */
71 main section.list {
72   display: grid;
73   grid-template-columns: auto auto auto auto;
74   gap: 1rem;
75 }
76 main section.list aside {
77   background-color: var(--third-light);
78   border-radius: 1rem;
79   padding: 1rem;
80   line-height: 2rem;
81 }
82 main section.list aside h3 strong {
83   font-weight: bold;
84 }
85 /* Rodapé */
86 footer {
87   background-color: var(--third);
88   height: 3rem;
89 }
90 footer img {
91   height: 100%;
92   padding: 0.4rem;
93 }
  
```

Por fim, o rodapé definimos uma altura fixa e as imagens com 100% da sua altura ajustada à altura do rodapé com um espaçamento entre a imagem e a borda do quadro do rodapé.

Com isso, nosso layout está pronto, porém, apesar de termos criado vários recursos nos preocupando com a proporcionalidade da largura e altura, nosso layout pode ficar mais responsivo. Para isso, por uma questão de organização, vou criar dois arquivos CSS, um para ajustar os quadros, tamanho dos itens e fontes, outro para montar o menu sanduíche e incluí-los no HTML. Nosso cabeçalho HTML vai ficar com 3 inclusões de CSS nesta ordem.

```

9 <link rel="stylesheet" href="./css/estilo.css" />
10 <link rel="stylesheet" href="./css/responsivo.css" />
11 <link rel="stylesheet" href="./css/menu.css" />
  
```

A ordem é importante, porque algumas propriedades serão alteradas de acordo com a largura da tela.

responsivo.css X

```
css > responsivo.css > {} @media (max-width: 500px)
1  @media (max-width: 1010px) {
2      main > section.list {
3          grid-template-columns: auto auto auto;
4      }
5  }
6  @media (max-width: 768px) {
7      header {
8          height: 4.5rem;
9      }
10     main > section.list {
11         grid-template-columns: auto auto;
12     }
13     main > section {
14         font-size: 0.9rem;
15     }
16     footer {
17         height: 2.5rem;
18     }
19 }
20 @media (max-width: 500px) {
21     main > section.list {
22         grid-template-columns: auto;
23     }
24 }
```

```
19 <nav>
20 <input id="menu-toggle" type="checkbox" />
21 <label for="menu-toggle">
22 <div class="menu-hamburger">
23 <span class="hamburger"></span>
24 </div>
25 </label>
26 </ul>
```

menu.css X

```
css > menu.css > header nav input#menu-toggle
1  header nav input#menu-toggle {
2      display: none;
3  }
4  @media (max-width: 500px) {
5      header nav {
6          position: fixed;
7          margin-top: 1rem;
8          align-items: baseline;
9      }
10     header nav ul {
11         display: none;
12         background-color: var(--white);
13     }
14     header nav ul li {
15         text-align: left;
16     }
17     header nav label[for="menu-toggle"] {
18         display: flex;
19         justify-content: flex-end;
20         cursor: pointer;
21     }
22     header nav label[for="menu-toggle"] div.menu-hamburger {
23         width: 35px;
24         height: 15px;
25         margin: 15px 2px 5px 5px;
26     }
27     header nav label[for="menu-toggle"] div.menu-hamburger span.hamburger {
28         position: relative;
29         display: block;
30         background-color: var(--black);
31         width: 30px;
32         height: 2px;
33     }
```

O arquivo responsivo.css possui o conteúdo ao lado.

A propriedade @media, permite aplicarmos um CSS baseado em uma condição. Nossa condição foi para tamanhos de tela de largura máxima de 1010px, diminuimos para 3 colunas.

Para tamanhos de tela de largura máxima de 768px, diminuimos a altura do cabeçalho e do rodapé, diminuimos para 2 colunas, diminuimos o tamanho da fonte.

Por fim, para tamanhos de tela de largura máxima de 500px, que já seria a largura dos celulares, diminuimos nossa grid para 1 coluna.

Essas foram as configurações que achei pertinente realizar, mas cada projeto irá pedir as suas.

Para verificar, diminua a largura da janela do navegador e verifique o que vai desconfigurando em telas de menor tamanho.

Para implementar o menu sanduíche, iremos incluir nos dois arquivos HTML, dentro da tag <nav> e antes da tag , a tag <input> do tipo checkbox para servir como um status se o menu está aberto ou fechado e a tag <label> que vincula no input através do atributo for. A tag <div> e serão utilizadas para montar os três tracinhos que compõem o menu.

No arquivo CSS, a primeira configuração, foi esconder o input.

A propriedade @media, define a largura da tela que o menu sanduíche irá começar a aparecer.

A tag nav tem position fixed, para que esse menu fique flutuante.

A tag ul é utilizada para montar um quadro branco atrás dos itens do menu.

A propriedade label[for="menu-toggle"] seleciona somente a label que possui esta propriedade.

A tag span foi utilizada para fazer os tracinhos do menu sanduíche. Observe que ela possui altura e largura fixas e fundo preto. Esta configuração cria o tracinho do meio, as próximas criam os de cima e de baixo.


```

menu.css X
css > menu.css > {} @media (max-width: 500px)
34 header nav label[for="menu-toggle"] div.menu-hamburger span.hamburger:before,
35 header nav label[for="menu-toggle"] div.menu-hamburger span.hamburger:after {
36   position: absolute;
37   display: block;
38   background-color: var(--black);
39   width: 100%;
40   height: 100%;
41   content: "";
42 }
43 header nav label[for="menu-toggle"] div.menu-hamburger span.hamburger:before {
44   top: -10px;
45 }
46 header nav label[for="menu-toggle"] div.menu-hamburger span.hamburger:after {
47   bottom: -10px;
48 }
49 header nav input:checked ~ ul {
50   display: block;
51   animation: fade-in 1s;
52 }
53
54 header
55   nav
56     input:checked
57     ~ label[for="menu-toggle"]
58     div.menu-hamburger
59     span.hamburger {
60       transform: rotate(45deg);
61       transition: 0.5s ease-in-out;
62     }
63 header
64   nav
65     input:checked
66     ~ label[for="menu-toggle"]
67     div.menu-hamburger
68     span.hamburger:before,
69 header
70   nav
71     input:checked
72     ~ label[for="menu-toggle"]
73     div.menu-hamburger
74     span.hamburger:after {
75       transform: rotate(90deg);
76       transition: 0.5s ease-in-out;
77     }
78 header
79   nav
80     input:checked
81     ~ label[for="menu-toggle"]
82     div.menu-hamburger
83     span.hamburger:before {
84       top: 0px;
85     }
86 header
87   nav
88     input:checked
89     ~ label[for="menu-toggle"]
90     div.menu-hamburger
91     span.hamburger:after {
92       bottom: 0px;
93     }
94 @keyframes fade-in {
95   from {
96     opacity: 0;
97   }
98   to {
99     opacity: 1;
100   }
101 }
102

```

A configuração `:before` e `:after`, permite criar outros 2 tracinhos. Eles terão a mesma largura e altura do anterior.

Para colocarmos antes `:before` e depois `:after` do tracinho do meio, separamos a configuração em duas.

A `:before` fica acima do tracinho do meio -10px

A `:after` fica abaixo do tracinho do meio -10px

O `input:checked ~ ul` indica o estilo que será aplicado ao `ul` quando o `input` estiver marcado.

Observe que o `display block` irá apresentar o menu que estava none na linha 11 do arquivo css.

A animação `fade-in` será apresentada mais abaixo.

A propriedade `transform` rotaciona os tracinhos em 45 graus.


O `transition` informa que esta rotação deve ocorrer em 0,5 segundos.



O `:before` e o `:after` irão girar mais 90 graus em 0,5 segundos, formando um X ao lado do menu.

Ainda para posicionar os tracinhos do `:before` e `:after`, aplicamos 0px do topo e do fundo respectivamente.

Por fim, a propriedade `@keyframes` permite definir de (from) e para (to) de uma animação.

No nosso caso, vamos aplicar opacidade no `fade-in` que é o momento em que o menu irá aparecer, ou seja, ele vai sair de totalmente escondido, opacidade 0, para totalmente aparente, opacidade 1.

	<p align="center">Centro Federal de Educação Tecnológica de Minas Gerais Campus VIII – Varginha Curso Técnico em Informática</p>	
<p><i>Disciplina</i> Lab. Aplicações Web 1</p>	<p align="center">HTML e CSS de Desenho no Figma</p>	<p><i>Professor</i> Lázaro Eduardo da Silva</p>

Data: dia/mês/ano hh:mm:ss

Título

Mensagem

Data: dia/mês/ano hh:mm:ss

Título

Mensagem

Data: dia/mês/ano hh:mm:ss

Título

Mensagem

Data: dia/mês/ano hh:mm:ss

Título

Mensagem






Página Inicial

Lista

Entrar

Cadastrar



Observe a implementação realizada e adapte para a realidade do seu projeto no Figma.

Bom trabalho!