



# Desenvolvimento de Sistemas

---

## Responsividade: conceito, *front-end* para ambiente *mobile*, técnicas e ferramentas

*Front-end*, sendo uma camada de apresentação e de interação, precisa ser desenvolvido com especial atenção às diferentes telas e resoluções que os usuários possam aplicar. A ascensão dos *smartphones* popularizou um conceito que hoje é fundamental: a responsividade, ou seja, a capacidade de uma página ser apresentada correta e agradavelmente em diversas telas diferentes sem prejuízo à experiência do usuário. Este conteúdo discute esse conceito e aplica recursos de HTML (*hyper text markup language*), CSS (*cascading style sheets*) e biblioteca Bootstrap para alcançar responsividade.

### Conceito

Inicialmente, em um ambiente predominantemente de computadores, as páginas *web* preocupavam-se em utilizar o máximo da largura de tela possível. Com a popularização das telas pequenas e estreitas dos *smartphones*, uma preocupação tornou-se crescente no desenvolvimento *web*: como acomodar bem uma página para que seja corretamente visualizada e ofereça uma boa experiência, seja por uma tela grande, seja por uma tela menor?

O termo “responsividade”, ou “*design responsivo*”, foi cunhado por Ethan Marcotte em seu artigo *Responsive Web Design*, de 2010, em que une algumas técnicas de *design web* em uma abordagem unificada: *layout* em grade flexível, imagens flexíveis e consultas de mídia, usando elementos de HTML e CSS. Com base nisso, é preciso projetar soluções flexíveis e adaptativas, e não mais desenhar soluções que restrinjam a Internet.

Observou-se, ainda, que a responsividade não deve se limitar a alterar *layouts* de acordo com o tamanho das telas, mas, sim, inverter o pensamento corrente de desenhar um *site* pensando primeiro nas dimensões de uma tela de *desktop* para depois reescalar

para telas menores. Ao invés disso, a ideia seria pensar primeiro nas menores telas para progressivamente evoluir para as telas maiores – hoje esse conceito é conhecido como **mobile first** (mobile primeiro).

Responsividade é, portanto, uma técnica de montagem de páginas usando HTML e CSS em que o *site* se adapta ao navegador e ao tamanho de tela utilizados sem precisar definir diversos *layouts* e diversos códigos HTML e CSS para cada tamanho específico.

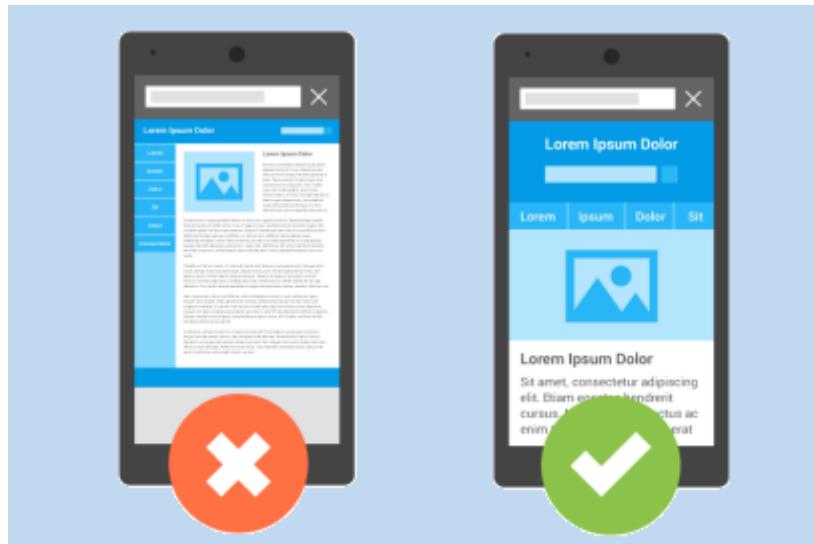


Figura 1 – Site não responsivo *versus* site responsivo em telas de *smartphone*

Fonte: PT Legal ([s. d.])

Algumas das características observadas em um *design* de *site* responsivo são as seguintes:

O *layout* da página deve se adaptar à resolução usada na visualização.

O *layout* deve ser fluido e evitar medidas fixas (quantidade de *pixels* para medir tamanho, por exemplo).

Os elementos da página precisam ser de fácil manipulação para o usuário, independentemente de resolução, podendo ser simplificados ou mesmo removidos se não forem necessários nos dispositivos menores

As imagens precisam ser otimizadas e redimensionadas quando necessário, tanto para adaptação à tela quanto para economia de dados de Internet.

Botões, *links* e *menus* precisam ser adaptados para que possam ser utilizados com toque, além de *mouse*.

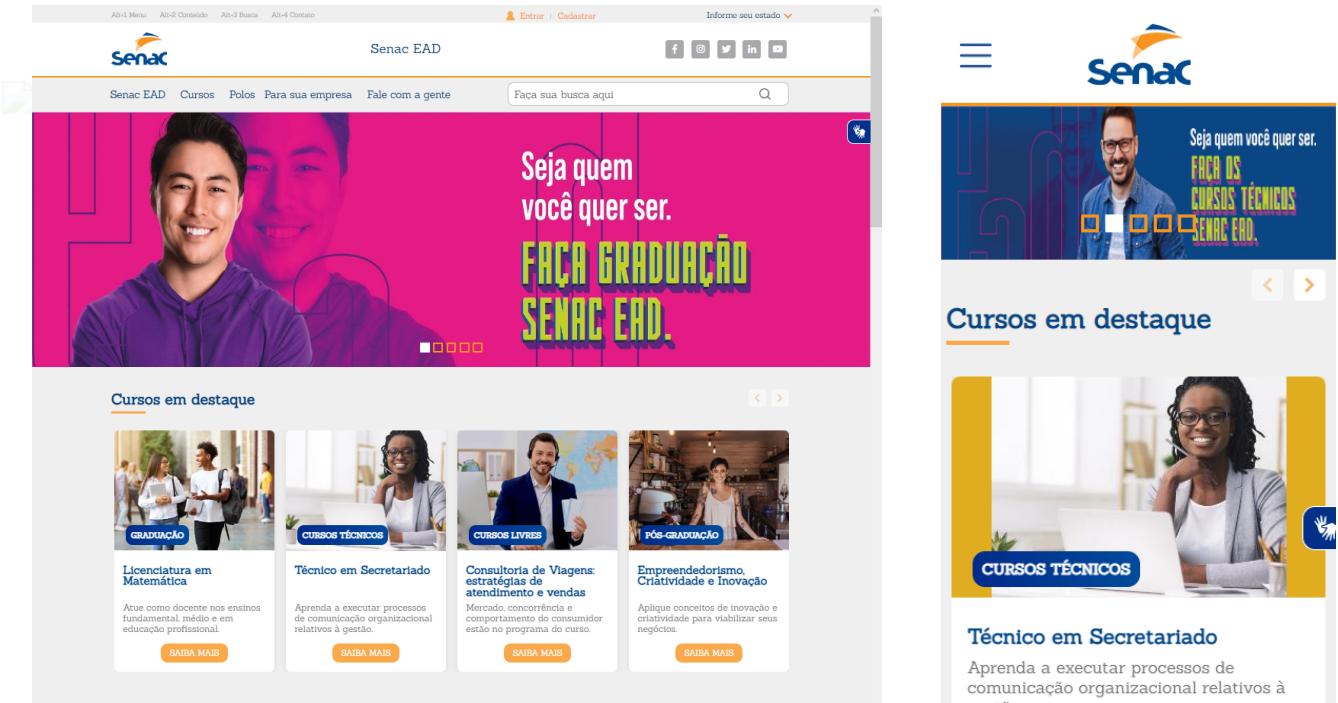


Figura 2 – O site do Senac EAD é responsivo: à esquerda em um *desktop* e à direita em uma tela de *smartphone*

Fonte: adaptado de Senac EAD (c2018)

Figura 3 – Site não responsivo: à direita em *desktop* e à esquerda em *mobile*

Fonte: Deque University (c2015)

Pelos exemplos das figuras 1, 2 e 3, é possível verificar que responsividade não significa mostrar um site exatamente da mesma maneira em uma tela e em outra. Pelo contrário, adaptações são necessárias para que o usuário acesse as mesmas informações com boa usabilidade.

# Front-end para mobile

Alguns elementos de HTML5 e de CSS3 ajudam na criação de páginas responsivas. Eles permitem redimensionar automaticamente, ocultar elementos e aumentar ou diminuir itens visuais de acordo com a tela em que o site é visualizado, deixando-o com boa aparência em diversos dispositivos.

## Tag viewport

O HTML conta com uma tag chamada **meta**, em que é possível incluir informações adicionais sobre os dados da página. A tag não aparece na tela, mas é detectada e interpretada pelo navegador.

O uso da tag **<meta>** para responsividade está ligado ao atributo **viewport**, na seguinte sintaxe:

```
<meta name="viewport" content="">
```

Antes de aprofundar o conhecimento nessa tag, cabe explicar o que é um **viewport**. Esse termo em inglês é usado para definir a área de exibição de conteúdo em um dispositivo e varia de acordo com o tamanho da tela do aparelho. No caso de um computador, o **viewport** para uma página depende do tamanho da janela do navegador – se a janela for redimensionada, o **viewport** aumentará ou diminuirá.

Voltando à tag HTML, usando **meta viewport**, é possível aplicar parâmetros como **width** para definir a largura do **viewport**, **height** para definir sua altura, e **initial-scale** para definir a escala inicial (o *zoom*) do **viewport**.

Para responsividade, é possível usar o valor **device-width** (largura do equipamento) e a escala inicial 1 (100%).

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Crie um arquivo HTML com o seguinte código para testar.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Teste Viewport
  </head>
  <body>
    <p><b>Abra esta página em um navegador modo desktop e depois em modo mobile</b></p>
    
    <p>
      Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.
      Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.
    </p>
  </body>
</html>
```

Note, pela figura a seguir, que a página se mantém mais ou menos com o mesmo *layout* no *desktop* e no *mobile*, tornando a experiência menos agradável na tela pequena.



Abra esta página em um navegador modo desktop e depois em modo mobile

Abra esta página em um navegador modo desktop e depois em modo mobile



Abra esta página em um navegador modo desktop e depois em modo mobile

Abra esta página em um navegador modo desktop e depois em modo mobile

Abra esta página em um navegador modo desktop e depois em modo mobile

Figura 4 – Página criada sem `<meta name="viewport">`

Fonte: Senac EAD (2023)

Agora, inclua a tag `<meta>` com **viewport** e depois visualize a diferença na figura 5.

```

<!DOCTYPE html>
<html>
    <head>
        <title>Teste Viewport</title>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <p><b>Abra esta página em um navegador modo desktop e depois em modo mobile</b>
        </p>
        
        <p>
            Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.
        </p>
    </body>
</html>

```

Abra esta página em um navegador modo desktop e depois em modo mobile



Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Abra esta página em um navegador modo desktop e depois em modo mobile



Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Figura 5 – Página criada com `<meta name="viewport">`

Fonte: Senac EAD (2023)

## Dimensões da imagem

Ao incluir imagens, como nos exemplos anteriores, é possível especificar suas dimensões por meio de unidades como pontos, centímetros ou *pixels*. No entanto, uma alternativa que pode ser mais adequada é usar a porcentagem da tela. Na página anterior,

se você quiser que a imagem tenha a mesma largura da tela, pode usar o atributo de CSS **width** com valor “100%” na tag HTML ou em uma folha de estilo.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Teste Viewport</title>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <p><b>Abra esta página em um navegador modo desktop e depois em modo mobile</b>
        </p>
        
        <p>
            Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.
        </p>
    </body>
</html>
```

Abra esta página em um navegador modo desktop e depois em modo mobile



Abra esta página em um navegador modo desktop e depois em modo mobile

Abra esta página em um navegador modo desktop e depois em modo mobile



Abra esta página em um navegador modo desktop e depois em modo mobile

Figura 6 – Imagem modificada para largura a 100% da tela

Fonte: Senac EAD (2023)

Pode haver um inconveniente em indicar a largura em 100%: se a imagem tem uma largura menor que a da tela, pode ser conveniente manter a dimensão original quando não for necessário. Para isso, é possível usar a propriedade de estilo **max-width** para definir a

largura máxima, de maneira que a imagem possa escalar para um valor menor, mas nunca maior ao especificado. Veja, a seguir, o trecho com a tag **<img>** do exemplo anterior usando esse recurso.

```

```

Abra esta página em um navegador modo desktop e depois em modo mobile



Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Abra esta página em um navegador modo desktop e depois em modo mobile



Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Figura 7 – Página com estilo **max-width** aplicado à imagem

Fonte: Senac EAD (2023)

Pela figura 7, note que, em *desktop*, a imagem assume sua largura original, já que é menor que a largura da janela. Já no *smartphone*, ela não ultrapassa os limites da largura da tela.

Geralmente **max-width** vem acompanhado da regra **height:auto**, definindo que a altura pode varia de acordo com a largura (o que já acontece por padrão quando se especifica apenas largura).

## Tamanho da imagem

Juntamente com a questão de dimensões, é possível aplicar recursos relativos a tamanho (ou “peso”) da imagem em *bytes*. Unindo essas duas questões, pode-se usar uma tag **<picture>** de HTML5 e definir que, de acordo com a largura da tela, use-se uma imagem de maior resolução ou outra de menor resolução (que também pode ser mais leve e demorar menos para carregar). Veja o destaque no exemplo a seguir, em que há duas versões para a imagem, uma com resolução 480 px x 320 px e outra com resolução 320 px x 213 px.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Teste Viewport</title>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <p><b>Abra esta página em um navegador modo desktop e depois em modo mobile</b></p>
        <picture>
            <source srcset="imagem-320x213.jpg" media="(max-width: 600px)">
            <source srcset="imagem-480x320.jpg" media="(max-width: 1000px)">
            <source srcset="imagem-teste.jpg">
            
        </picture>
        <p>
            Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.
            Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.
        </p>
    </body>
</html>
```

Dois itens são decisivos para usar essa solução. O primeiro é a tag **<source>**, que permite usar opções de figuras para uma mesma imagem. O outro é o atributo **media** de HTML5, em que se consegue especificar condições para usar uma figura ou outra de acordo com propriedades do dispositivo sendo usado. Nesse caso, usa-se como condição a largura da tela: se ela for de até 600 px, usará a imagem do arquivo “**“imagem-320x213.jpg”**; caso a largura da tela seja de até 1.000 px, a imagem virá de “**“imagem-480x320.jpg”**; caso contrário, por padrão, usará a imagem original “**“imagem-teste.jpg”** (que tem dimensões 640 x 425).

Abra esta página em um navegador modo desktop e depois em modo mobile



Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Abra esta página em um navegador modo desktop e depois em modo mobile



Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Abra esta página em um navegador modo desktop e depois em modo mobile



Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Figura 8 – Comportamento da imagem em três larguras diferentes de telas

Fonte: Senac EAD (2023)

Para ilustrar melhor o que você viu até agora, monte uma página e use a tag **<picture>** para mostrar três imagens completamente diferentes de acordo com a largura da tela. Também use a tag de dimensão **max-width** para que ajuste a 100% da largura da tela. Teste encolhendo e expandindo a tela do navegador e usando o recurso de simulação de tela de dispositivos da ferramenta de desenvolvedor do navegador.

## Media queries

No exemplo anterior, você experimentou um pouco do que é um *media query* (ou consulta de mídia): verificou por meio do atributo **media** um valor relativo ao dispositivo em que a imagem seria mostrada. *Media queries* são propriedades de CSS que permitem realizar testes sobre o dispositivo e aplicar regras específicas de estilo para esse dispositivo.

A sintaxe é a seguinte:

```
@media tipo-de-midia and (regra [ ] and|or|not (regra] ){  
    regras css;  
}
```

O **tipo-de-midia**, em questão de responsividade, tipicamente será *screen* (tela). Porém, é possível definir estilo para impressão (*print*) ou sintetizadores de voz (*speech*), por exemplo.

A **regra** são as condições para que as propriedades sejam aplicadas. Pode haver mais de uma regra e, nesse caso, usa-se *and*, *or* ou *not* para montar a expressão condicional.

Aplique *media query* em um exemplo simples usando a página com a qual está trabalhando até agora. Neste exemplo, será usada uma cor diferente para o fundo da tela e para o título na página quando estiver usando telas maiores.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Teste Viewport</title>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <style>
            @media screen and (min-width:600px){
                body{
                    background-color: #DAF7A6;
                }
            }
            @media screen and (min-width:1000px){
                p b{
                    color: #0D38A1 ;
                    font-family: Arial, Helvetica, sans-serif;
                    font-size: 2em
                }
            }
        </style>
    </head>
    <body>
        <p><b>Abra esta página em um navegador modo desktop e depois em modo mobile</b>
    </p>
        
        <p>
            Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.
            Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.
        </p>
    </body>
</html>
```

No trecho destacado, há a primeira consulta verificando se a tela tem tamanho mínimo de 600 px (**@media screen and (min-width:600px)**). Se sim, mudará a cor de fundo da página. Em seguida, é verificado se a tela tem o mínimo de 1.000 px de largura. Caso afirmativo, também se modifica a fonte do texto em negrito na página.



Figura 9 – Comparação da página mostrada em telas com 460 px, 665 px e 1.110 px

Fonte: Senac EAD (2023)

Veja um segundo exemplo em que se pode definir um *menu de links* horizontal por padrão (pensando em *mobile first*, esse seria o *layout* desejável) e, para telas maiores, manter o *menu* na lateral esquerda.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Teste Viewport</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
      /*por padrão: lista sem pontos e com cor de fundo...*/
      ul{
        list-style-type: none;
        overflow: hidden;
        background-color: #DAF7A6;
      }
      /*...e itens da lista ficarão lado a lado.*/
      li{
        float:left;
        padding: 5px;
      }
      /*Para telas maiores...*/
      @media screen and (min-width:600px){
        /*...os itens da lista ficam um abaixo do outro...*/
        li{
          float:none;
        }
        /*...o menu fica à esquerda...*/
        nav{
          float:left;
        }
        /*...e o conteúdo dá espaço ao menu*/
        main{
          padding-left: 110px;
        }
      }
    </style>
  </head>
```

```

<body>
    <nav>
        <ul>
            <li><a href="#">Home</a>
            <li><a href="#">Artigos
            <li><a href="#">Sobre
        </ul>
    </nav>
    <main>
        <p><b>Abra esta página em um navegador modo desktop e depois em modo mobile</b></p>
        
        <p>
            Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.
            Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.
        </p>
    </main>
</body>
</html>

```

No exemplo, estão sendo definidas regras-padrão para os elementos **ul** e **li**, que formam o *menu de links*, por meio das quais se define que o *menu* terá cor de fundo e *links* lado a lado. Com base no *media query*, se a tela tiver largura maior que 600 px, os *links* ficam alinhados na vertical (**float:none** sobrescreverá o **float:left** padrão) e posicionados à esquerda da página (**float:left** para o elemento **nav**)



[Home](#) [Artigos](#) [Sobre](#)

Abra esta página em um navegador modo desktop e depois em modo mobile

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.



[Home](#) [Artigos](#) [Sobre](#)

Abra esta página em um navegador modo desktop e depois em modo mobile

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Figura 10 – *Menu para desktop e para mobile*Fonte: Senac EAD (2023)

*Media queries* são essenciais para a construção de interfaces responsivas e são usadas pelas ferramentas que serão abordadas adiante neste conteúdo.

Crie uma página e use *media query* para mostrar todo o conteúdo em fonte Arial, quando estiver em tela de *mobile*, e em fonte Georgia com tamanho maior, quando estiver em tela de *desktop*.

## Técnicas e ferramentas

Algumas ferramentas ajudam na tarefa de montar *design web* responsivo de maneira mais prática. A principal e mais reconhecida no mercado é o Bootstrap, que será tratado com mais detalhes adiante.

Antes, veja algumas alternativas (recomenda-se que você busque e teste essas outras ferramentas).

### Pure CSS

Biblioteca *open-source* com coleção de pequenos componentes CSS que seguem a premissa *mobile first* e que podem ser usados em qualquer projeto *web*. Bastante leve, não tem dependência com bibliotecas JavaScript como o JQuery (e por isso não traz componentes mais complexos como o Bootstrap); baseia-se no CSS padrão e é extensível. Uma vantagem é ser disponível em módulos, que podem ser obtidos separadamente para o projeto.

The screenshot shows the Pure.CSS website. On the left is a dark sidebar with a white header 'PURE' and a list of links: Get Started, Layouts, Base, Grids, Forms, Buttons, Tables, Menus, Tools, Customize, Extend, and Releases. Below these are two small icons. The main content area features a large 'Pure.CSS' logo with a blue 'P' icon. Below the logo is a subtitle: 'A set of small, responsive CSS modules that you can use in every web project.' Underneath this is a snippet of CSS code showing a link to the minified file. At the bottom of the main content area is a horizontal bar divided into six colored segments, each representing a different CSS module: Base (blue), Grids (purple), Forms (green), Buttons (red), Tables (orange), and Menus (yellow). Each segment shows its size: 0.9KB, 0.6KB, 1.4KB, 0.7KB, 0.5KB, and 0.7KB respectively. Below the bar is a sub-subtitle: 'CSS with a minimal footprint.' followed by a paragraph explaining the file size and optimization. A small note at the bottom states: '\* We can add correctly :) the numbers above are individual module sizes; when grouped together they compress (gzip) even more.'

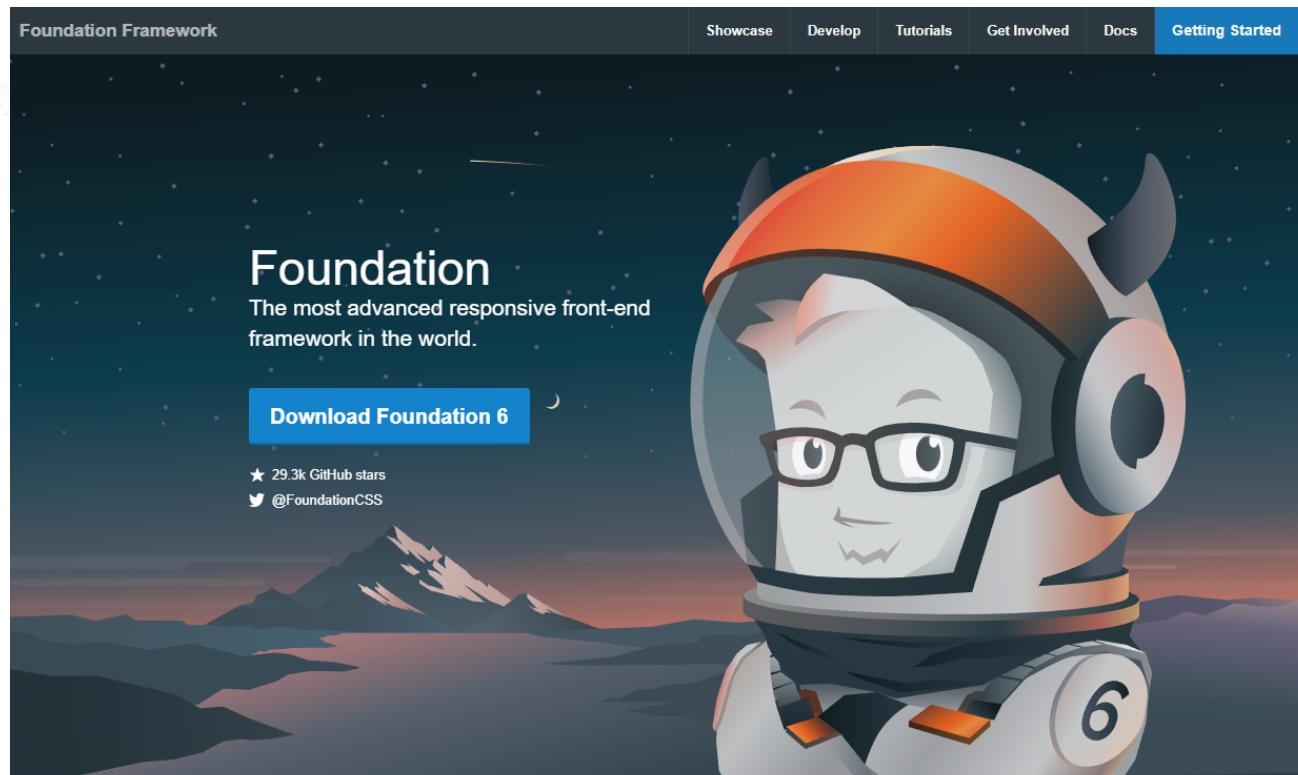
Figura 11 – Página inicial do Pure CSS

Fonte: Pure CSS (c2014)

Página disponível em [purecss.io](http://purecss.io). Mostra título “Pure.css”, abaixo links para obtenção do código, e abaixo um texto explicativo. Na lateral esquerda, menu com links diversos.

## Foundation

*Framework open-source* flexível com funcionalidades semelhantes ao Bootstrap, oferecendo uma gama alta de ferramentas de *design* e componentes reutilizáveis. Conta com sistema de *grids*, tipografia, componentes UI, *plugins* JavaScript e outros. Pode ser customizado, mas sua curva de aprendizado pode ser mais acentuada por ter muitos recursos.



## Responsive design gets a whole lot faster

A Framework for any device, medium, and accessibility. Foundation is a family of responsive front-end

Figura 12 – Página inicial do Foundation

Fonte: Foundation (c1998-2023)

Página disponível em <https://get.foundation>. Mostra no topo alguns links, abaixo uma grande imagem e um link para download e mais abaixo texto explicativo sobre a ferramenta.

## Cascade

É outra ferramenta CSS *open-source* que usa esquema de *grids* para responsividade e conta com uma série de componentes, como tabelas, navegação e tipografia. Dividido em módulos, o desenvolvedor pode escolher quais funcionalidades deseja importar ao projeto, incluindo apenas as folhas de estilo necessárias para isso. Também é pensado para que funcione bem em navegadores novos e antigos.

Grid   Typography   Icons   Components   Templates   Download v1.6

**Cascade Framework**  
Putting back the C in CSS

Star 273 Fork 52 Tweet Follow

**Feature Rich** Semantic and non-semantic grid layouts, base templates, table designs, navigation elements, typography and lots, lots more.

**Universal** Build high performance websites in no time for old browsers and new browsers alike without worrying about browser quirks.

**Atomic** An optimised atomic design gives you full control of the look and feel of your site while keeping code bloat minimal.

**Modular** Don't need certain features? Don't include them. Unlike other frameworks, you can include only the components you need.

### Different from other CSS frameworks

Although the overall look and feel are most definitely inspired by Twitter Bootstrap, Cascade framework is not just another Bootstrap clone. Where Twitter Bootstrap puts its focus on delivering shiny user elements that can be dropped into any project and takes control of your project's overall look-and-feel, Cascade Framework is intended to do the opposite. By splitting your CSS into separate files based on features rather than selectors as well as by implementing [atomic design](#), Cascade Framework puts **you** in control!

Also different from Twitter Bootstrap or other CSS Frameworks out there, Cascade Framework can be used for modern browsers and older browsers alike. All

Figura 13 – Página inicial do Cascade

Fonte: Cascade (c2023)

Página disponível em <http://jslegers.github.io/cascadeframework/index.html>. Mostra alguns links no topo, abaixo área com logo e título “Cascade Framework”. Abaixo, textos explicativos sobre a ferramenta.

## Bootstrap

Lançado em 2011 por iniciativa do Twitter, o *framework open-source* Bootstrap estabelece uma série de padrões, componentes e classes CSS que elevam a consistência do projeto, facilitam a manipulação do estilo, agilizam a programação de funcionalidades comuns de *front-end* e auxiliam na comunicação do código-fonte (alguém com conhecimento em Bootstrap saberá o que o seu código intenciona ao ler as classes no HTML, o que nem sempre é verdade quando se criam estilos CSS próprios). Essas facilidades em conjunto com a necessidade de adaptar a web para a realidade dos *smartphones*, que se expandia na época, tornaram a ferramenta uma das mais utilizadas em *design* de páginas desde então.

The screenshot shows the official Bootstrap website. At the top, there's a purple header bar with the Bootstrap logo (a stylized 'B' inside a bracket), navigation links for 'Docs', 'Examples', 'Icons', 'Themes', and 'Blog', and social media icons for GitHub, Twitter, and LinkedIn. To the right of the header are links for 'v5.3' and a dark mode switch. Below the header, a yellow banner at the top left says 'New in v5.3' followed by 'Color mode support, expanded color palette, and more!'. The main content area has a large purple gradient background featuring the Bootstrap logo. The tagline 'Build fast, responsive sites with Bootstrap' is centered in a large, bold, dark font. Below the tagline is a brief description: 'Powerful, extensible, and feature-packed frontend toolkit. Build and customize with Sass, utilize prebuilt grid system and components, and bring projects to life with powerful JavaScript plugins.' There are two buttons at the bottom: a grey one with '\$ npm i bootstrap@5.3.0-alpha1' and a purple one with 'Read the docs'. At the very bottom, there's a small advertisement for OVHcloud services.

Figura 14 – Página inicial do Bootstrap

Fonte: Bootstrap ([s. d.])

As principais características do Bootstrap são suas classes CSS, usadas para aplicar estilo ou para embutir alguma funcionalidade nos elementos da página, seus componentes de interface gráfica, como barras de navegação, *menus* expansíveis e janelas modais, e sua integração com Javascript, permitindo comportamento dinâmico.

Por exemplo, se você quiser que um botão tenha uma aparência específica definida pelo Bootstrap, basta que se inclua a classe CSS correta no elemento de HTML e as funcionalidades e o estilo associados estarão prontamente aplicados ao elemento. Grosseiramente, é possível dizer que usar o Bootstrap é incluir as classes certas nas *tags* de HTML.

## Usando o Bootstrap

Há duas maneiras de aplicar Bootstrap a uma página HTML:

1. Baixando a biblioteca e fazendo referência aos arquivos.

## 2. Usando *links* CDN (*content delivery network*).



Se você optar por **baixar a biblioteca**, na página principal do projeto (procure por “get Bootstrap”), há uma linha que mostra o número da versão (na data de produção deste material, a versão mais atual é 5.3) e um *link* para *download*. Ele levará a uma página de *downloads*. Nessa página, há uma seção **Compiled CSS and JS** e um botão **Download**. Clica-se nesse botão para baixar um arquivo de nome **bootstrap-<versão>-dist.zip** ou algo semelhante.

The screenshot shows the Bootstrap download page. On the left, there's a sidebar with navigation links for 'Getting started', 'Customize', and 'Layout'. The main content area has a heading 'Download' and a sub-section 'Compiled CSS and JS' which is highlighted with a red rounded rectangle. This section contains a list of what's included in the download and a prominent blue 'Download' button. To the right of the main content, there's a sidebar titled 'On this page' with links to various documentation sections like 'Compiled CSS and JS', 'Source files', and 'Package managers'.

Figura 15 – Página de *download* do Bootstrap

Fonte: Senac EAD (2023)

No arquivo compactado que foi baixado, serão encontrados todos os arquivos CSS e JavaScript necessários para o *framework*. Extraia o conteúdo diretamente para a pasta onde ficarão as páginas HTML e troque o nome da nova pasta extraída para **bootstrap**. É importante que dentro dessa pasta **bootstrap** estejam as pastas **css** e **js**.

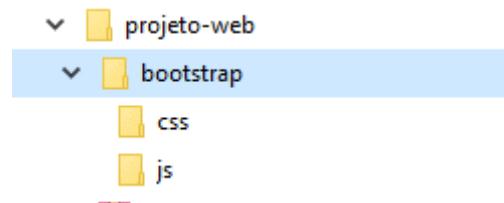


Figura 16 – Hierarquia de pastas para o Bootstrap

Fonte: Senac EAD (2023)

Na pasta principal (**projeto-web** neste exemplo), será criado um arquivo HTML com o seguinte código:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Primeiro Bootstrap</title>
  </head>
  <body>
    <h1>Estamos usando bootstrap?</h1>
    <p>Se este texto e o botão abaixo estão estilizados, estamos sim.</p>
    <button type="button">Click me!</button>
  </body>
</html>
```

Abra no navegador e você verá uma página bastante simples.

## Estamos usando bootstrap?

Se este texto e o botão abaixo estão estilizados, estamos sim.

**Click me!**

Figura 17 – Página para testes criada, sem estilo

Fonte: Senac EAD (2023)

Adicione o Bootstrap incluindo uma referência ao arquivo CSS principal (**bootstrap.min.css** presente na pasta **bootstrap/css**) e também a um arquivo de JavaScript (**bootstrap.min.js**, presente na pasta **bootstrap/js**). Note a diferença recarregando a página no navegador.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Primeiro Bootstrap</title>
    <link href="bootstrap/css/bootstrap.min.css" rel="stylesheet">
  </head>
  <body>
    <h1>Estamos usando bootstrap?</h1>
    <p>Se este texto e o botão abaixo estão estilizados, estamos sim.</p>
    <button type="button" class="btn btn-primary">Click me!</button>
    <script src="bootstrap/js/bootstrap.bundle.min.js"></script>
  </body>
</html>
```

## Estamos usando bootstrap?

Se este texto e o botão abaixo estão estilizados, estamos sim.

Click me!

Figura 18 – Página de teste agora com Bootstrap

Fonte: Senac EAD (2023)

Perceba que sem necessitar de alterações no HTML, o texto já foi reestilizado. O botão, no entanto, precisou levar duas classes CSS próprias do Bootstrap: **btn** e **btn-primary**. Essa é uma amostra de como trabalhar com Bootstrap.

Outra maneira de usar o Bootstrap é **por meio de links CDN** (*links da web* que permitem referenciar os arquivos do *framework* sem a necessidade de *download* de todo o pacote). Nesse caso, ao invés de referenciar arquivos locais, você apontará para

endereços específicos na web. Esses endereços são disponibilizados pela página do Bootstrap.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Primeiro Bootstrap</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-GLh1TQ8iRABdZLl603oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD" crossorigin="anonymous">
  </head>
  <body>
    <h1>Estamos usando bootstrap?</h1>
    <p>Se este texto e o botão abaixo estão estilizados, estamos sim.</p>
    <button type="button" class="btn btn-primary">Click me!</button>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js" integrity="sha384-w76AqPfDkMBDXo30js1Sgez6pr3x5MlQ1ZAGC+nuZB+EYdgRZgiwxhTBtkF7CXvN" crossorigin="anonymous">
  </body>
</html>
```

Para a versão 5.3.0, as referências serão as destacadas acima. Elas podem ser encontradas também na página inicial do Bootstrap, na seção **Include via CDN**. Os atributos **integrity** e **crossorigin** são opcionais, mas conferem mais segurança ao uso do CDN – os valores devem ser obtidos na página do Bootstrap, pois mudam a cada versão.

Abra a página desenvolvida no navegador e note que o resultado é exatamente o mesmo de antes. A vantagem em usar CDN é que pode tornar o projeto um pouco mais leve por não necessitar baixar recursos desnecessários no carregamento da página.

Antes de realizar novos experimentos, veja alguns dos conceitos mais marcantes do *framework* Bootstrap.

## Sistema de grade

O Bootstrap usa um sistema de grade (ou *grid system*) para posicionamento de elementos da página HTML, organizando *containers* como **<div>** em uma disposição de linhas e colunas. Imagine que uma seção do site precisa mostrar três imagens e um texto

abaixo de cada uma. Nesse caso, seria definido, usando classes do Bootstrap, que você quer três colunas para essa região da página. A grade é dividida em 12 colunas e é nisso que seu *layout* deverá se basear.

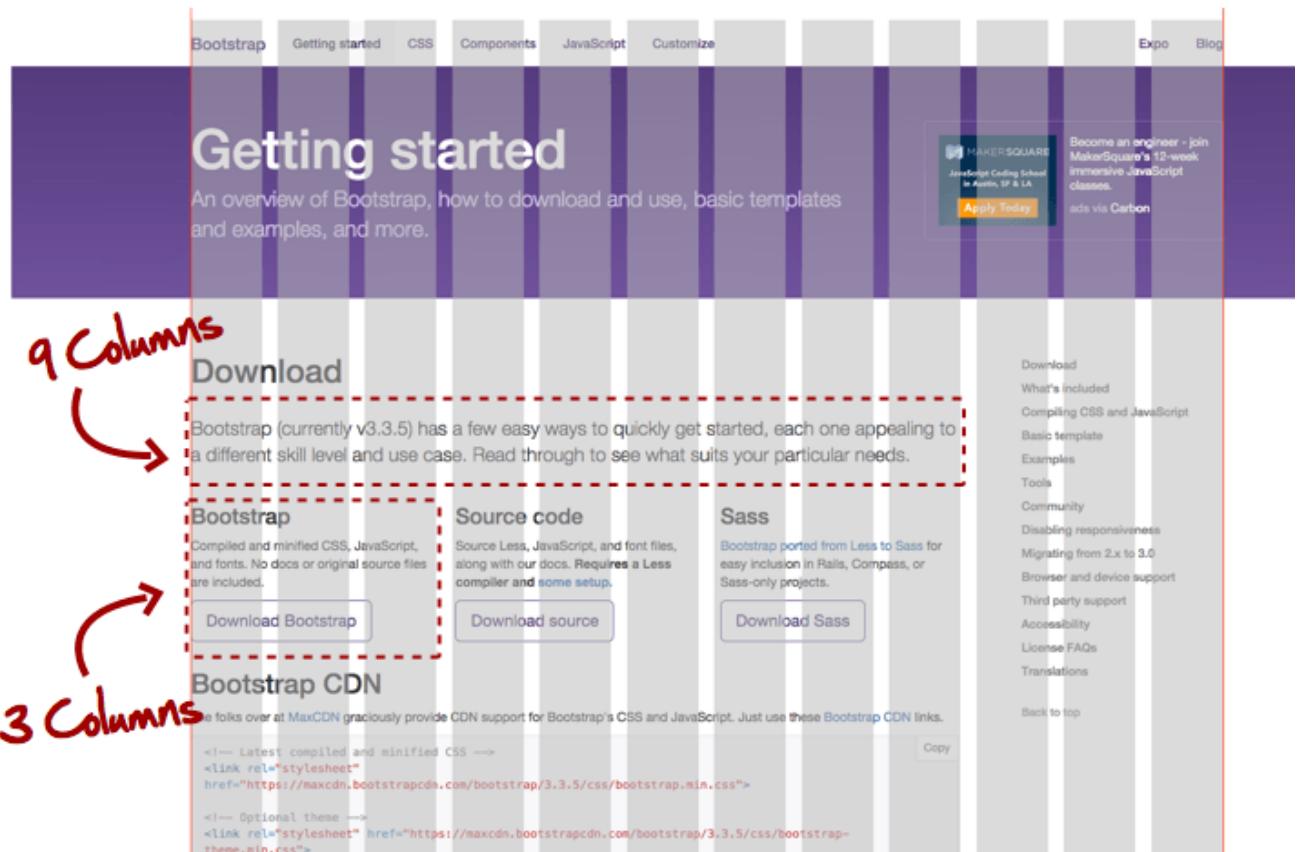


Figura 19 – Grid do Bootstrap aplicado a uma página. O elemento destacado superior ocupa nove colunas, enquanto o elemento destacado inferior ocupa três colunas

Fonte: Kind PNG (2019)

Algumas instruções devem ser seguidas na hora de montar página com *grid*:

O *grid* estará em um elemento HTML com classe **container**.

O **container** incluirá elementos com classe **row**.

Os elementos **row** incluem colunas, cuja largura é definida especificando-se o número de células que ela abrange usando uma classe **col-sm-\***, na qual \* é esse número. Por exemplo, quatro elementos **<div>** com classe **col-sm-3** resultariam em quatro colunas igualmente espaçadas (o número 3 em **col-sm-3** indica que **<div>** está usando o espaço de 3 das 12 colunas definidas pela estrutura de *grid* do Bootstrap).

.col-sm1											
----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------

col-sm-8		col-sm-4
.col-sm-4	.col-sm-4	.col-sm-4
.col-sm-6		.col-sm-6

Tabela 1 – Esquema de grid do Bootstrap

Fonte: Senac EAD

Veja na tabela 1 que, se você quiser usar 12 colunas, aplicará a classe **col-sm-1** para cada um dos 12 elementos que representarem essas colunas. Por outro lado, se quiser uma seção que ocupe 8 das 12 colunas, **usará col-sm-8**, e assim por diante.

Os *grids* se adaptam de acordo com a resolução da tela. Para telas mais largas, as células aparecem dispostas lado a lado como mostrado na figura 1. Em telas mais estreitas, elas aparecem uma abaixo da outra.

O termo **sm**, usado no nome da classe do exemplo, é relativo à responsividade e indica que o *layout* é pensado para telas pequenas (*small*) e ficará em células lado a lado em resoluções de, no mínimo, 540 px – resoluções menores transformarão as células em linhas. Além de **sm**, tem-se também **md**, **lg** e **xl**, como mostra a tabela 1, que se comportam de maneira análoga dependendo da resolução que representam.

	<b>Extrapequeno (extra small)</b> <b>Resolução &lt; 576 px</b>	<b>Pequeno (small devices)</b> <b>Resolução ≥ 576 px</b>	<b>Médio (medium devices)</b> <b>Resolução ≥ 768 px</b>	<b>Grande (large devices)</b> <b>Resolução ≥ 992 px</b>	<b>Extragrande (extra large)</b> <b>Resolução ≥ 1.200 px</b>
<b>Tamanho máximo do container</b>	Nenhum (auto)	540 px	720 px	960 px	1.140 px
<b>Prefixo de classe</b>	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl

Tabela 2 – Esquemas de classes de coluna do Bootstrap para cada faixa de resolução de tela

Fonte: Senac EAD (2023)

Experimente o sistema de *grid* com um exemplo simples:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Grids Bootstrap</title>
        <link href="bootstrap/css/bootstrap.min.css" rel="stylesheet">
    </head>
    <body>
        <div class="container">
            <div class="row">
                <div style="background-color: grey;" class="col-md-6" >
                    <p>Conteúdo de uma coluna</p>
                </div>
                <div style="background-color: green;" class="col-md-6">
                    <p>Conteúdo de outra coluna</p>
                </div>
            </div>
        </div>
        <script src="bootstrap/js/bootstrap.bundle.min.js"></script>
    </body>
</html>
```

Note que há uma **<div>** com classe **container** e, dentro dele, outro com classe **row** (a linha). Dentro dessa **<div>**, há duas outras **<div>** com classe **col-md-6**, especificando que cada coluna ocupará metade do **container**.

Figura 20 – Teste com colunas do Bootstrap

Fonte: Senac EAD (2023)

Teste a página em tela *mobile* e veja que as colunas se tornam linhas quando a resolução é pequena, ou seja, adaptam-se à visualização.

Em um exemplo um pouco mais prático, imagine que você precisa fazer uma vitrine de produtos em uma página. É possível tirar benefício do sistema de *grids*.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Vitrine </title>
        <link href="bootstrap/css/bootstrap.min.css" rel="stylesheet">
```

```
<style>
    .produto{
        max-height:150px;
        max-width:100%
    }
    .destaque{
        float:left;
        max-height:200px;
        max-width:100%
    }
</style>
</head>
<body>

<div class="container">
    <div id="linha-titulo" class="row">
        <div class="col-sm-12">
            <h3>Nossos produtos</h3>
        </div>
    </div>
    <div id="linha-destaques" class="row">
        <div class="col-sm-6">
            
            <p><strong>OFERTA RELÂMPAGO</strong></p>
            <p>Smartphone de última geração</p>
            <p><strong>Apenas R$ 1000,00</strong></p>
        </div>
        <div class="col-sm-6">
            
            <p><strong>OFERTA RELÂMPAGO</strong></p>
            <p>Tablet de última geração</p>
            <p><strong>Apenas R$ 800,00</strong></p>
        </div>
    </div>
    <div id="linha-produtos" class="row">
        <div class="col-sm-3">
            
            <p>HD de 500 GB <strong>R$ 350</strong></p>
        </div>
        <div class="col-sm-3">
            
            <p>Monitor de 20 polegadas <strong>R$ 400</strong></p>
        </div>
        <div class="col-sm-3">
            
            <p>Notebook com 8GB de memória<strong>R$ 1950</strong></p>
        </div>
        <div class="col-sm-3">
            
            <p>PC com processador 9 núcleos <strong>R$ 2500</strong></p>
        </div>
    </div>
    </div>
    <script src="bootstrap/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

Observe com atenção o código acima. Nele, estão sendo definidos um **container** e três linhas. A **<div>** com **id "linha-título"** usa a classe **col-sm-12**, pois deseja-se que o título ocupe toda essa linha (ou seja, as 12 colunas do *grid*). Já a **<div>** de **id "linha-destaques"** tem dois elementos, cada um ocupando 6 colunas (ou seja, metade do **container**). Na terceira linha, com **id "linha-produtos"**, desejam-se quatro itens e, por isso, a classe **col-sm-3** (pois 12 dividido por 4 é igual a 3) é usada. Note, pela figura 21, que os itens se rearranjam em telas menores.

Figura 21 – Página de teste em tela *desktop* e em tela *mobile*

Fonte: Senac EAD (2023)

É possível também haver **container** dentro de **container**, expandindo as possibilidades de divisões do conteúdo na página. Por exemplo, se você quiser que, em telas menores, a descrição dos produtos de destaque fique abaixo em vez de ficar ao lado da imagem, é possível ajustar da seguinte maneira (o código a seguir mostra apenas o trecho, do código anterior, que sofre alteração):

```
<div id="linha-destaques" class="row">
    <div class="col-sm-6">
        <div class="container">
            <div class="row">
                <div class="col-sm-4">
                    
                </div>
                <div class="col-sm-8">
                    <p><strong>OFERTA RELÂMPAGO</strong></p>
                    <p>Smartphone de última geração</p>
                    <p><strong>Apenas R$ 1000,00</strong></p>
                </div>
            </div>
        </div>
    </div>
    <div class="col-sm-6">
        <div class="container">
            <div class="row">
                <div class="col-sm-4">
                    
                </div>
                <div class="col-sm-8">
                    <p><strong>OFERTA RELÂMPAGO</strong></p>
                    <p>Tablet de última geração</p>
                    <p><strong>Apenas R$ 800,00</strong></p>
                </div>
            </div>
        </div>
    </div>

```

```
</div>
</div>
</div>
</div>
</div>
```

Em cada coluna de **linha-destaques** é incluído um **container**, uma linha e duas colunas, que se ajustarão dinamicamente de acordo com a largura da tela.

Figura 22 – Página vitrine ajustada

Fonte: Senac EAD (2023)

Usando o sistema de *grid* do Bootstrap, crie uma página como a da imagem a seguir. Use **<divs>** com propriedades **height** e **background-color** configuradas.

Figura 23 – Exemplo

Fonte: Senac EAD (2023)

## Texto e tipografia

A seguir, alguns recursos que Bootstrap oferece com relação ao conteúdo de texto.

Bootstrap já traz estilização para as *tags* de **<h1>** a **<h6>**. Não é necessário aplicar classe específica.

É possível alinhar o texto de uma **<div>** ou outro **container** usando as classes **text-center** (centralizado) e **text-end** (alinhado à direita).

```
<div>
    <p>texto à esquerda</p>
</div>
<div class="text-center">
    <p>texto centralizado</p>
</div>
<div class="text-end">
    <p>texto à direita</p>
</div>
```



Figura 24 – Alinhamentos de texto

Fonte: Senac EAD (2023)

Pode-se montar uma lista **<ul>** de itens dispostos horizontalmente com as classes **list-inline** e **list-inline-item**.

```
<ul class="list-inline">
    <li class="list-inline-item">item 1
</li>
    <li class="list-inline-item">item 2
</li>
    <li class="list-inline-item">item 3
</li>
</ul>
```

A classe **lead** dá destaque a um parágrafo.

```
<p class="lead">Este parágrafo está diferente
</p>
<p>Este parágrafo é comum</p>
<p>E este parágrafo também é comum</p>
```

Figura 25 – Parágrafo com ênfase

Fonte: Senac EAD (2023)

## Imagens

Bootstrap tem uma classe própria que aplica a propriedade **max-width:100%** discutida anteriormente neste conteúdo. Trata-se da classe **img-fluid**.

```

```



Também é possível alinhar imagens à esquerda ou à direita do **container** com as classes **float-start** e **float-end**.

```


```

Para deixar uma imagem centralizada, pode-se usar a classe **text-center**.

Figura 26 – Imagem centralizada e com borda arredondada

Fonte: Senac EAD (2023)

## Tabelas

A estilização de tabelas fica facilitada com o Bootstrap. Além disso, a ferramenta aplica elementos de responsividade à estrutura.

A classe **table** já é capaz de estilizar a tabela sem necessidade de outras classes.

```
<table class="table">
  <thead>
    <tr>
      <th>ID</th>
      <th>Nome</th>
      <th>Idade</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>João Silva</td>
      <td>50</td>
    </tr>
  </tbody>
</table>
```



```
</tr>
<tr>
    <td>2</td>
    <td>Maria Aparecida</td>
    <td>45</td>
</tr>
<tr>
    <td>3</td>
    <td>Quitéria Santana</td>
    <td>30</td>
</tr>
</tbody>
</table>
```

É possível aplicar borda à tabela com a classe **table-bordered**.

```
<table class="table table-bordered">
    ...
</table>
```

É possível fazer uma tabela zebraada (uma linha com cor e outra não) usando a classe **table-striped**.

```
<table class="table table-striped">
    ...
</table>
```

Figura 27 – Tabela zebraada com borda

Fonte: Senac EAD (2023)

## Formulários

O Bootstrap traz várias melhorias aos formulários tradicionais. Cada campo do formulário deve usar a classe **form-control**. Rótulos para os campos devem usar a classe **form-label**. Os tamanhos dos campos são adaptáveis com a lógica de grades do Bootstrap, permitindo responsividade.

```
<div class="container">
  <form method="post" action="#">
    <div class="mb-3">
      <label for="txtNome" class="form-label">Nome:</label>
      <input id="txtNome" class="form-control" type="text">
    </div>
    <div class="mb-3">
      <label for="txtSenha" class="form-label">Senha:</label>
      <input id="txtSenha" class="form-control" type="password">
    </div>
    <div class="mb-3">
      <input type="submit" class="btn btn-primary" value="Acessar" >
    </div>
  </form>
</div>
```

Figura 28 – Formulário estilizado com Bootstrap

Fonte: Senac EAD (2023)

Campos **<select>** contam com uma classe específica **form-select**.

```
<form method="post" action="#">
  <div class="mb-3">
    <label class="form-label" for="selRegião">Selecione a região </label>
    <select id="selRegiao" class="form-select">
      <option>Norte </option>
      <option>Nordeste </option>
      <option>Centro-Oeste </option>
      <option>Sudeste </option>
      <option>Sul </option>
    </select>
  </div>
  <div class="mb-3">
    <input type="submit" class="btn btn-primary" value="OK" >
  </div>
</form>
```

*Inputs de checkbox* também têm uma classe própria: **form-check-input**. Para o **<label>** associado a esse elemento, usa-se a classe **form-check-label**. Ambos os elementos precisam estar em um **container** com classe **form-check**.

```
<form method="post" action="#">
    <div class="form-check">
        <input class="form-check-input" type="checkbox" value="" id="chkEmail">
        <label class="form-check-label" for="chkEmail">
            Deseja receber nossos emails?
        </label>
    </div>
</form>
```

As mesmas classes são aplicadas a **radio-buttons**.

```
<form method="post" action="#">
    <div class="form-check">
        <input class="form-check-input" type="radio" name="radioSexo" id="rbMasculino">
        <label class="form-check-label" for="rbMasculino">
            Masculino
        </label>
    </div>
    <div class="form-check">
        <input checked="" class="form-check-input" type="radio" name="radioSexo" id="rbFeminino">
        <label class="form-check-label" for="rbFeminino">
            Feminino
        </label>
    </div>
</form>
```

## Menu de navegação

A classe **nav** pode ser aplicada às tags **<ul>** para formar uma lista horizontal de navegação. Para cada item **<li>** também é preciso usar a classe **nav-item**.

```
<ul class="nav">
```

```
<li class="nav-item">
    <a class="nav-link" href="#">Página Inicial</a>
</li>
<li class="nav-item">
    <a class="nav-link" href="#">Produtos</a>
</li>
<li class="nav-item">
    <a class="nav-link disabled" href="#">Sobre a Empresa</a>
</li>
</ul>
```

Figura 29 – *Menu de navegação com três links*

Fonte: Senac EAD (2023)

É possível usar em conjunto com **nav** a classe **justify-content-center** para manter o *menu* no centro da tela ou **justify-content-end** para alinhá-lo à direita. E se preferir uma navegação vertical, basta adicionar a classe **flex-column**.

## Modais

Modal é um painel ou uma caixa de diálogo que surge à frente do conteúdo de uma página de modo que a interação aconteça apenas com esse painel. É bastante útil para mostrar mensagens ao usuário ou complementar uma ação. Com Bootstrap, para criar um modal, primeiro é preciso definir alguns dados no botão que o aciona:

```
<button type="button" class="btn btn-primary" data-bs-toggle="modal" dat
a-bs-target="#meuModal">
    Abrir Modal
</button>
```

É necessário usar os atributos de dados **data-bs-toggle** com valor **modal** e **data-bs-target** com valor correspondente ao **id** do **container** que será o painel modal – neste caso, haverá uma **<div>** com **id = "meuModal"**.

É preciso, então, criar o painel em si:

```
<div class="modal" id="meuModal">
    <div class="modal-dialog">
        <div class="modal-content">

            <!-- Cabeçalho -->
            <div class="modal-header">
                <h4 class="modal-title">Este é um modal</h4>
                <button type="button" class="btn-close" data-bs-dismiss="modal"></button>
            </div>

            <!-- Conteúdo -->
            <div class="modal-body">
                <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
                <p>Maecenas sit amet lorem suscipit, pulvinar nisl sit amet, eleifend ligula.</p>
                <p>Nulla a ornare lectus, nec pellentesque metus.</p>
            </div>

            <!-- Rodapé -->
            <div class="modal-footer">
                <button type="button" class="btn btn-danger" data-bs-dismiss="modal"> Fechar </button>
            </div>

        </div>
    </div>
</div>
```

A classe CSS **modal** define o painel modal como um todo.

Dentro do elemento com essa classe, há um **container** com a classe **modal-dialog** e, dentro desse, outro **container** com **modal-content** que forma o conteúdo.

No **container** com **modal-content** é possível incluir uma **<div>** com classe **modal-header** para uma barra de título. A classe **modal-title** pode ser usada para a estilização do título e **btn-close** para um botão de fechar em formato de X.

Importante notar o atributo **data-bs-dismiss** com valor **modal**. Caso não tenha esse atributo, o botão não será capaz de fechar o painel.

A classe **modal-body** define o conteúdo principal do painel. Aqui poderia ser incluída uma mensagem de erro, uma orientação ao usuário, imagens ou mesmo um formulário.



A classe **modal-footer** define um rodapé para o painel, podendo trazer botões de interação. Note mais uma vez o atributo **data-bs-dismiss** com valor **modal**.

Figura 30 – Modal em ação

Fonte: Senac EAD (2023)

Também é possível usar JavaScript para abrir um modal sem a necessidade do clique do usuário. Basta, em algum evento (**ready** do jQuery, por exemplo), criar um objeto e invocar um método **show()**, como no código a seguir.

```
constmyModal = new bootstrap.Modal('#meuModal', { });
myModal.show();
```

Dentre as chaves, podem ser informadas algumas opções, como possibilitar ou bloquear o fechamento do modal pela tecla **Esc** (**keyboard: true** ou **keyboard: false**).

Pesquise na documentação do Bootstrap e teste os recursos a seguir um a um:

Alerts

Navbar

Jumbotron

Cards

Carousel

## Encerramento

Com base no conteúdo estudado, fica clara a importância de pensar em responsividade ao desenvolver páginas para *front-end*. Elementos de HTML5 com CSS3 já garantem boas adaptações às páginas para que se ajustem adequadamente na maioria das telas. No entanto, ferramentas como o Bootstrap, de grande presença no mercado, são de grande auxílio por trazerem soluções práticas de responsividade e por incluírem, ainda, recursos muito úteis no desenvolvimento de páginas *web*. A recomendação é que você continue estudando essa ferramenta vasta em recursos e mantenha-se de olho nas atualizações que forem disponibilizadas.