

EBOOK GRATUITO

# EVENTO DEVUP #02



Material de apoio para auxiliar no desenvolvimento  
do projeto das aulas gratuitas.

# Sumário

---

<b>Introdução.....</b>	<b>4</b>
Objetivo .....	5
Dicas de Leitura.....	6
Ambiente de Desenvolvimento.....	7
Protótipo .....	9
Projeto.....	12
<b>Tecnologias.....</b>	<b>13</b>
HTML.....	14
CSS .....	15
<b>HTML .....</b>	<b>16</b>
Estrutura básica .....	17
Tags HTML .....	19
Código HTML.....	23
<b>CSS.....</b>	<b>30</b>
Criando Folha CSS.....	31
Seletores CSS .....	32
Propriedades CSS.....	34
Código CSS .....	43

<b>Considerações .....</b>	<b>57</b>
Projeto.....	58



# Introdução

# Objetivo

---

A proposta deste ebook é auxiliar você a dar os seus primeiros passos com HTML e CSS. Essas tecnologias são essenciais no mercado do Desenvolvimento Web e formam o pacote de conhecimento que você precisa ter, independente de qual caminho sua carreira siga no futuro – FrontEnd ou BackEnd.

Em outras palavras, saber o conceito destas linguagens podem ser habilidades imprescindíveis para uma carreira de sucesso. E o melhor de tudo, é que vamos aprender já codificando um site super bonito e profissional!

# Dicas de Leitura

---

Este ebook pode ser acessado de qualquer dispositivo. Porém, para que seu aprendizado seja efetivo, você precisará interagir com os códigos HTML e CSS, e para isso será necessário o uso de um **computador** ou **notebook**.

Segundo **William Glasser**, psiquiatra americano que aplicou a **teoria da escolha** para a educação, quando lemos absorvemos apenas 10% do conteúdo, enquanto quando praticamos absorvemos 80%.

Então DEV, se você realmente quer entender e absorver este conteúdo, precisará praticar os códigos e conceitos que serão ensinados no decorrer do ebook.

# Ambiente de Desenvolvimento

---

Para conseguir desenvolver o site proposto no ebook, você precisará de apenas duas ferramentas instaladas no seu computador.

## Navegador (browser)

O navegador é um programa criado para permitir a navegação pela internet, através de um caminho que te leva até o que você procura na rede.

Ele é um programa básico. O Windows por exemplo, vem por padrão com o Internet Explorer, mas existem outros browsers como o Chrome, Safari, Opera, Firefox, por exemplo.

## Editor de texto

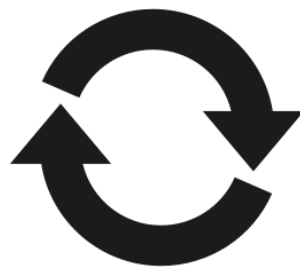
Caso você nunca tenha tido contato com as tecnologias que vamos usar, saiba que são arquivos como qualquer outro. O documento HTML é um arquivo **.html** que teremos que utilizar um navegador – seja ele qual for – para abri-lo.

Para criar ou editar este arquivo .html, você precisará de um editor de texto que interprete seu código HTML. Recomendamos a utilização do [Visual Studio Code](#), mas você pode escolher outros como o [Sublime Text](#), [Notepad++](#), enfim, o que você preferir.

Caso esteja usando o Visual Studio Code, realize a configuração abaixo para que você não precise ficar salvando o documento sempre que fizer uma alteração:

1. Clique em **"File"**;
2. Clique em **"Auto Save"**.

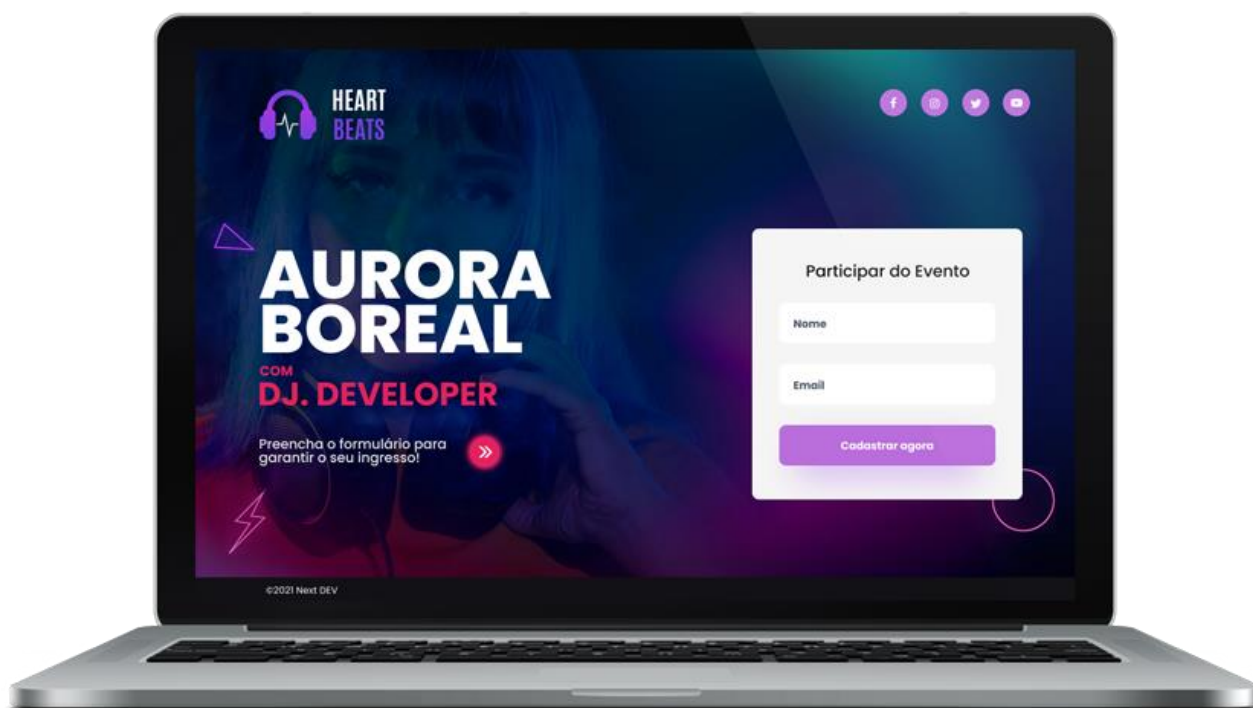
Caso esteja usando outro editor de texto, sempre que ver este símbolo abaixo, salve e recarregue o navegador para visualizar as alterações.





# Protótipo

Ao final deste ebook você terá construído uma página web do zero, igual este modelo abaixo! Legal, né?! 😊



O nome disso que fizemos é **protótipo**. O protótipo nada mais é do que o esboço do projeto, podendo ser de baixa ou alta fidelidade e tem o objetivo de validar a ideia do que será desenvolvido, nesse caso um site.

A prototipação é uma fase importante do desenvolvimento porque é através dela que conseguimos analisar instantaneamente falhas em nosso conceito de design.

Logo, um protótipo permite resolver esses problemas antes de passarmos para a codificação, pois assim já teremos propriedade no que será desenvolvido.

A ferramenta que escolhemos para prototipar este projeto é o Figma, mas existem diversas outras ferramentas que você pode utilizar para prototipar os seus projetos.

Caso queira ter acesso ao documento da prototipação que fizemos, será necessário criar uma conta no Figma. A grande vantagem desta ferramenta é que ela é totalmente online e gratuita, então você não precisará instalar pacotes ou softwares no seu computador.

## Criar a conta no Figma

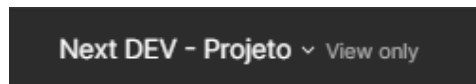
1. Acesse o link: <https://www.figma.com/>;
2. Clique no botão preto **"Try Figma for Free"**;
3. Na janela que surgir você pode logar usando a sua conta do Google ou criar uma conta digitando seu e-mail e uma senha;
4. Clique em **"Create account"**;
5. Na mesma janela, digite seu nome (é possível editá-lo depois) e escolha a área em que você estuda ou trabalha;
6. Por fim, clique em **"Create account"**.

## Copiar o protótipo do projeto

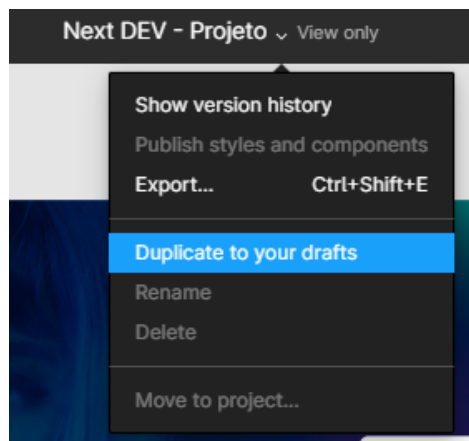
Agora que criou a sua conta, vamos copiar o protótipo do projeto. Quando clicar no link, apenas conseguirá visualizar o projeto, para conseguir editá-lo será necessário realizar o passo a passo abaixo:

1. Logado na sua conta do Figma, clique no link  
<https://www.figma.com/file/RGl3MfOsmiJTOSLQMRR85X/Next-DEV-Projeto>

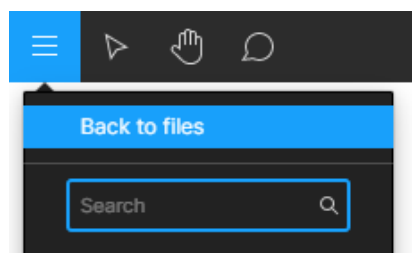
2. Na barra superior, localize a seta ao lado do nome do projeto e clique.



3. Ao clicar na seta, vai abrir um menu. Clique em **"Duplicate to your drafts"**.



4. Vai aparecer uma mensagem no rodapé do site, avisando que o arquivo foi duplicado.
5. Clique no primeiro menu da barra superior, e depois em **"Back to files"**.



6. Localize o projeto duplicado, e pronto! 😊

Agora com tudo configurado, aperta o cinto que a nossa jornada já vai começar!


# Projeto

Crie uma pasta na Área de Trabalho chamada **"Projeto DevUp #02"**.

Abra o seu editor de texto escolhido, crie um novo documento e salve-o dentro da pasta do projeto com o nome **index.html**, como abaixo:



Nome: index.html  
Tipo: All Files



Nome: index  
Tipo: HTML

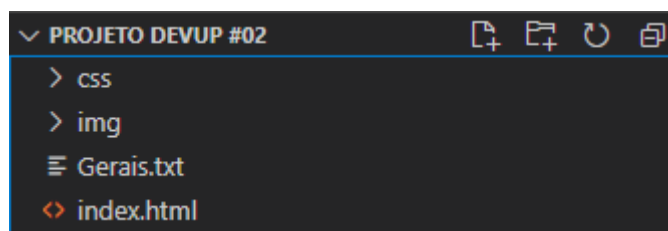
Ambos os modelos estão corretos.

Aproveite e crie uma pasta chamada **"css"**. Agora acesse o link abaixo e faça o download das imagens que serão utilizadas neste projeto:

## [Download imagens](#)

Depois de fazer o download, arraste o conteúdo da pasta para a raiz de "Projeto DevUp #02"

A sua pasta do projeto deve ficar semelhante a isto:





# Tecnologías

# HTML

---

HTML é uma das linguagens que utilizamos para desenvolver websites. O acrônimo HTML vem do inglês e significa Hypertext Markup Language ou em português **Linguagem de Marcação de Hipertexto**.

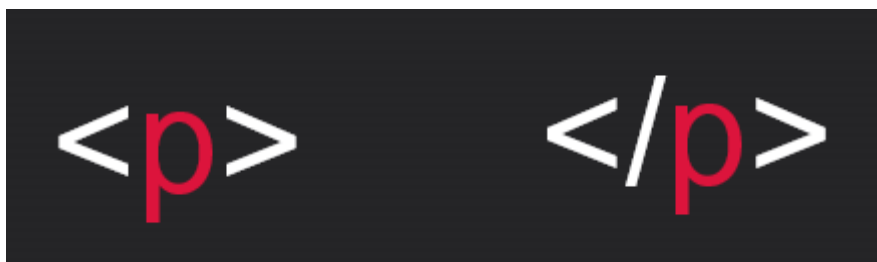
O HTML é a linguagem base da internet. Foi criada para ser de fácil entendimento por seres humanos e também por máquinas, como por exemplo o Google ou outros sistemas que percorrem a internet capturando informação.

Atualmente, o HTML encontra-se em sua quinta versão, o HTML5, sendo essa a versão mais utilizada na Internet.

## O que são as tags?

As tags são usadas para informar ao navegador a estrutura do site. Quando se escreve um código em HTML, as tags serão interpretadas pelo navegador, produzindo assim a estrutura e o conteúdo visual da página.

A principal característica das tags é estarem sempre dentro dos sinais de chevron (sinal de "menor que" e "maior que"), ou seja: **< >**.



# CSS

---

O CSS é uma linguagem de estilos que tem o objetivo de definir o layout de documentos escritos com linguagem de marcação – HTML. Ele adiciona estilo a uma página e altera a forma visual de como ela é apresentada.

O seu nome é um acrônimo que significa Cascading Style Sheets, que no português significa **Folha de Estilos em Cascata**.

O CSS é responsável pelo visual do site. Sendo assim, através dele aplicamos cores, definimos os tamanhos, bordas, efeitos e diversas outras coisas.

## Sintaxe CSS

Para escrever código CSS é necessário seguir uma regra. A regra é uma declaração que possui uma sintaxe própria bem simples que define a forma com que o estilo será aplicado aos nossos elementos HTML.

```
seletor {  
    propriedade: valor;  
}
```

**IMPORTANTE:** Neste ebook não iremos aplicar regras de CSS que irão tratar da responsividade da página, ou seja, a capacidade de adaptação de uma página a diferentes dispositivos, como tablets, celulares, etc. Isto porque não é o foco dessa aula introdutória, sendo um conteúdo mais avançado que abordamos apenas no curso **“Desenvolvendo Meu Primeiro Site”**.



# HTML

---



# Estrutura básica

Essa é a estrutura básica para construção de uma página HTML.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <title>Estrutura básica</title>
</head>
<body>
  <!-- Conteúdo da página -->
</body>
</html>
```



## DOCTYPE

Primeiramente, temos o comando DOCTYPE. Apesar da sintaxe parecida, ele não é uma tag HTML. O doctype é utilizado para informar ao navegador qual é a versão do HTML utilizada no arquivo.

A declaração do doctype é *case insensitive*, ou seja, não difere letras maiúsculas de minúsculas. Portanto, pode ser usado como preferir.

## HTML

A abertura e fechamento da tag **<html>** delimita o documento. Sendo assim, todas as demais tags dessa página devem estar nesse espaço.

## HEAD

A tag **<head>** serve para armazenar informações do seu site. Essas informações são invisíveis, ou seja, os usuários que estiverem navegando pelo seu site não verão o que existe na tag head.

Repare que a tag **<title>** não exibe nenhum conteúdo no corpo da sua página, mas sim, corresponde ao título que é mostrado na aba do navegador, ou seja, é uma informação sobre o seu documento html.

Essas informações são importantes para o seu navegador e para os serviços de busca, como o Google.

## META CHARSET="UTF-8"

Especifica o conjunto de caracteres que será usado para renderizar o texto da página, compreendendo a acentuação e outros caracteres especiais da região, no nosso caso o Brasil.

## TITLE

A tag **<title>** define o título da página, que será exibido na aba do navegador.

## BODY

A tag **<body>** define o espaço no qual deve estar contido o conteúdo visual da página, que será exibido para os visitantes do site. Portanto, todas as demais tags que representam textos, botões, etc. devem ser adicionadas neste intervalo.

# Tags HTML

---

Agora que você já aprendeu sobre a estrutura básica do HTML, vamos entender sobre outras tags importantíssimas, que inclusive, usaremos no nosso projeto.

## HEADER

Cuidado para não confundir com a tag `<head>`. O `<header>` tem a função de indicar que determinado conteúdo é um cabeçalho.

## FOOTER

Assim como a tag `<header>` que indica um cabeçalho, existe a tag `<footer>` que indica a existência de um rodapé.

## MAIN

A tag `<main>` tem uma única função semântica: definir o conteúdo principal da página ou da aplicação. Ele representa o conteúdo mais importante da página, que está diretamente relacionado ao tópico central do documento ou a funcionalidade principal de uma aplicação.

## DIV

O elemento `<div>` define uma divisão ou seção em um documento HTML. Ele é frequentemente usado como um contêiner para outros elementos, o que facilita na estilização de blocos. Portanto, ele é definido com um contêiner genérico para conteúdo de fluxo.

## UL

Utilizamos a tag `<ul>` quando queremos dar início a uma lista, neste caso, uma lista não ordenada. Caso desejássemos que fosse ordenada (1. 2. 3. ... N.), precisaríamos utilizar a tag `<ol> </ol>`.

Dentro das tags `<ul>` ou `<ol>`, utilizamos a tags `<li></li>` para cada novo elemento de nossas listas. Caracteristicamente, os itens em uma lista desordenada são exibidos com um marcador que pode ter várias formas, como um ponto, um círculo, ou um quadrado. O tipo de marcador não é definido na descrição HTML da página, mas na CSS associada, utilizando a propriedade `list-style-type`.

## LI

O elemento `<li>` (ou lista dos itens) é usado para representar um item que faz parte de uma lista. Este item deve estar contido em um elemento pai: uma lista ordenada (`<ol>`), uma lista desordenada (`<ul>`), ou um menu (`<menu>`) e representa uma única entidade dessa lista. Em menus e listas desordenadas a relação de itens é exibida, normalmente, usando pontos de marcação (as bolinhas). Em listas ordenadas eles são, comumente, mostrados com algum contador ascendente - como um número, ou letra - à sua esquerda.

## A

A tag `<a>` define um hiperlink, que é usado para vincular de uma página a outra. O atributo mais importante deste elemento é o `"href"`, que indica o destino do link.

## H1 – H6

Os elementos HTML `<h1>` a `<h6>` representam seis níveis de título de seção, em que `<h1>` é o nível de seção mais alto e `<h6>` é o mais baixo. Em outras palavras, o h1 é maior e mais importante que o h6.

Organizar a sua página de acordo com estas regras de importância é fundamental para otimizar a sua página para mecanismos de buscas (SEO), como o Google.

## SPAN

O elemento `<span>` é um conteúdo genérico em linha, ele pode ser usado para agrupar elementos para fins de estilo (usando os atributos `class` ou `id`), por exemplo. Ele deve ser usado somente quando nenhum outro elemento (HTML) semântico for apropriado.

## P

A tag `<p>` define um parágrafo. É uma das tags mais utilizadas, a cada novo parágrafo você precisará utilizar a tag `<p>`.

Os navegadores adicionam automaticamente uma única linha em branco antes e depois de cada `<p>` elemento.

## IMG

A tag `<img>` é utilizada quando você precisa inserir uma imagem em sua página. Repare que esta tag é acompanhada do atributo `"src"` que irá indicar o caminho dessa imagem no seu servidor web ou local.

## FORM

O elemento `<form>` é um contêiner para diferentes tipos de elementos de entrada, como: campos de texto, caixas de seleção, botões de opção, botões de envio, etc.

## INPUT

O elemento `<input>` HTML é o elemento de formulário mais usado.

Um elemento `<input>` pode ser exibido de várias maneiras, dependendo do seu atributo `"type"`.

No desenvolvimento do nosso projeto, iremos utilizar três diferentes: `text`, `email` e `submit`. Sendo respectivamente entrada de texto, email e envio do formulário.

# Código HTML

Agora que você já conhece sobre a estrutura básica do HTML e suas principais tags, podemos seguir estruturando o nosso documento.

Vamos adicionar dentro da tag **<head>** algumas configurações como o ícone do site (favicon), a fonte e a biblioteca de ícones de Redes Sociais que vamos utilizar.

```
<head>
  <meta charset="UTF-8">
  <title>Aurora Boreal | Next DEV</title>

  <!-- FAVICON -->
  <link rel="shortcut icon" href="img/favicon.png" type="image/
png">

  <!-- ÍCONES -->
  <link rel="stylesheet" href="https://unpkg.com/@coreui/icons@
2.0.0-beta.3/css/all.min.css">

  <!-- FONTE POPPINS -->
  <link rel="preconnect" href="https://fonts.gstatic.com">
  <link href="https://fonts.googleapis.com/css2?family=Poppins:
wght@500;600;700;800&display=swap" rel="stylesheet">
</head>
```



## Criando o Header

De acordo com o que queremos desenvolver, teremos algo similar a isso:

```
<body>
  <header>
    <div class="content">
      <div class="logo"></div>

      <ul>
        <li>
          <a href="#">
            <i class="cib-facebook-f"></i>
          </a>
        </li>

        <li>
          <a href="#">
            <i class="cib-instagram"></i>
          </a>
        </li>

        <li>
          <a href="#">
            <i class="cib-twitter"></i>
          </a>
        </li>

        <li>
          <a href="#">
            <i class="cib-youtube"></i>
          </a>
        </li>
      </ul>
    </div>
  </header>
</body>
```





No header, criamos uma `<div>` e adicionamos uma **"class"** chamada **"content"**. O objetivo desta div, que se repete tanto no `<main>` quanto no `<footer>` é limitar o tamanho do conteúdo. Entenderemos melhor quando chegarmos na parte de CSS.

Nós iremos adicionar o nosso logo através do CSS, por isso criamos uma `<div>` e adicionamos uma **"class"** chamada **"logo"**.

Com o elemento `<ul>`, criamos a nossa lista não ordenada, ou seja, a lista em que os `<li>` usam marcadores.

Podemos observar que dentro dos nossos itens de lista, tem elementos um dentro do outro. Isso se chama aninhamento, onde o elemento que está dentro é filho do elemento que está por fora.

Ex:

```
<li>
  <a href="#"></a>
</li>
```

A tag `<li>` é pai da tag `<a>`, portanto `<a>` é filho de `<li>`.

Utilizamos a tag `<a>` dentro dos nossos `<li>` pois esta é uma lista de botões, e é necessário esta tag para criar âncoras (redirecionamentos) para outras páginas, neste caso, as redes sociais.

E por fim, dentro de `<a>` inserimos a tag `<i>` com os ícones das redes sociais. É importante ressaltar que, para utilizar estes ícones é necessário adicionar a linkagem da biblioteca de ícones dentro do `<head>`, como fizemos anteriormente.

## Criando o Main

Agora, criaremos o conteúdo principal do nosso projeto:

```
<main>
  <div class="content">
    <div class="text">
      <h1>Aurora Boreal</h1>

      <div class="dj">
        <span>com</span>
        <p>DJ. DEVELOPER</p>
      </div>

      <div class="cta">
        <p>Preencha o formulário para garantir o
        seu ingresso!</p>
        <div class="button">
          
        </div>
      </div>
    </div>

    <div class="form">
      <p>Participar do Evento</p>
      <form action="#">
        <input type="text" placeholder="Nome">
        <input type="email" placeholder="Email">
        <input type="submit" value="Cadastrar agora">
      </form>
    </div>

    <div class="light01"></div>
    <div class="light02"></div>
    <div class="light03"></div>
  </div>
</main>
```



Essa estrutura é criada logo após o fechamento da tag **<header>** dentro do **<body>**.

Foram utilizadas diversas **<div>**'s dentro do nosso **<main>** para conseguirmos depois, trabalhar o posicionamento corretamente com o CSS.

Repare que, o título do nosso projeto "Aurora Boreal" foi definido com a tag **<h1>**, o título mais importante dentro deste projeto.

No formulário, definimos três campos de entrada. O atributo **"placeholder"** tem o objetivo de colocar um texto demonstrativo pro usuário saber o valor que este campo espera.

Ex.

As div's com as classes **".light01"**, **".light02"** e **".light03"** serão responsáveis por englobar as "luzes" do nosso projeto. Como essa:



## Criando o Footer

Para finalizarmos a nossa estrutura, vamos criar o footer:

```
<footer>
  <div class="content">
    <p>&copy;2021 Next DEV</p>
  </div>
</footer>
```



A tag **<footer>** deve ser adicionada abaixo da tag **<main>**.

Geralmente o conteúdo de footer são as redes sociais, direitos autorais e mapa do site. No nosso projeto, colocaremos os direitos autorais.

O código **&copy;** é responsável por criar o símbolo de copyright.

## Estrutura Projeto

A estrutura do seu projeto deve ser semelhante a isso:

```
<body>
  <header>
    ...
  </header>

  <main>
    ...
  </main>

  <footer>
    ...
  </footer>
</body>
```

Se estiver desta forma, parabéns DEV! Finalizamos a estrutura HTML do nosso projeto 😊

Nos links abaixo, você consegue baixar o código até esta parte do desenvolvimento:

[Github](#) ou [Mega](#)

Agora, bora desenvolver o visual?



# CSS

---

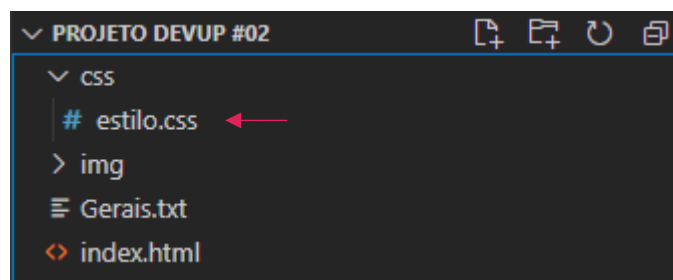
# Criando Folha CSS

Precisaremos criar um arquivo **.css** para conseguirmos estilizar a nossa estrutura HTML. Crie um novo documento e salve-o dentro da pasta **"css"** com o nome **estilo.css**. Confira abaixo:

Nome:	<input type="text" value="estilo.css"/>
Tipo:	<input type="text" value="All Files"/>

Nome:	<input type="text" value="estilo"/>
Tipo:	<input type="text" value="CSS"/>

A estrutura do projeto deve ficar assim:



# Seletores CSS

---

É através dos seletores que “selecionamos” o elemento HTML que queremos estilizar. Podemos fazer esta seleção de diversas formas, como pelo nome da tag, classe, id e outros.

Dentro do projeto usamos o seletor por tag, classe, atributo, universal e pseudoclassee. Vamos ver agora um pouco mais sobre cada um deles.

## Seletor de Tag

Este seletor básico escolhe todos os elementos que correspondem ao nome fornecido.

**Sintaxe:** nome-da-tag

**Ex:** header corresponderá a tag <header>

## Seletor de Classe

Este seletor básico escolhe elementos baseados no valor de seu atributo classe.

**Sintaxe:** .nome-da-classe

**Ex:** .index irá corresponder a qualquer elemento que tenha class="index"



## Seletor de Atributo

Este seletor básico irá escolher os elementos baseados no valor de um de seus atributos, ou até mesmo pelo próprio atributo. [Clique aqui](#) para ver o conteúdo que fizemos sobre.

**Sintaxe:** [atrib] [atrib=valor]\*

**Ex:** input[type="email"] irá corresponder a todos os campos de entrada com tipo email.

\* Existem outras formas de sintaxe

## Seletor Universal

O seletor universal atinge todos os elementos HTML. Normalmente é utilizado para resetar (zerar) algumas propriedades que vem como padrão nos navegadores.

**Sintaxe:** \*

## Pseudo-classe

Uma pseudo-classe CSS é uma palavra-chave adicionada a seletores que especifica um estado especial do elemento selecionado. [Clique aqui](#) para ver o conteúdo que fizemos sobre.

**Sintaxe:** seletor:pseudoclasse

**Ex:** a:hover pode ser usado para alterar a cor quando o usuário passar o cursor sobre ele.

# Propriedades CSS

---

Agora que você viu como selecionamos o elemento HTML que queremos estilizar, vamos ver um pouco sobre as propriedades do CSS, que são responsáveis por atribuir características aos elementos.

## MARGIN

A propriedade margin do CSS define a área de margem nos quatro lados do elemento. É uma abreviação que define todas as margens individuais de uma só vez: margin-top, margin-right, margin-bottom, e margin-left.

A propriedade margin pode ser especificada usando um, dois, três ou quatro valores.

[Clique aqui](#) para ver o conteúdo que fizemos sobre, serve tanto para o margin quanto o padding.

## PADDING

A propriedade padding define uma distância entre o conteúdo de um elemento e suas bordas. É um atalho que evita definir uma distância para cada lado separadamente: padding-top, padding-right, padding-bottom, padding-left.

O padding, assim como o margin também aceita de um a quatro valores.

[Clique aqui](#) para ver o conteúdo que fizemos sobre, serve tanto para o margin quanto o padding.

## WIDTH

A propriedade **width** define a largura de um elemento.

A largura de elemento por padrão, não inclui preenchimento, bordas ou margens. Para conseguir incluir o **"border"** e **"padding"** é necessário alterar a propriedade **"box-sizing"** para **"border-box"**.

## HEIGHT

A propriedade **height** define a altura de um elemento. Igual ao **width**, por padrão não inclui padding, border ou margin mas é possível alterar a propriedade box-sizing.

## BOX-SIZING

A propriedade CSS box-sizing é utilizada para alterar a propriedade padrão da box model, usada para calcular larguras (widths) e alturas (heights) dos elementos. [Clique aqui](#) para ver o conteúdo que fizemos sobre.

### content-box

Essa é o estilo padrão, conforme especificado pela norma CSS. As propriedades **width** e **height** são medidas incluindo só o conteúdo, mas não o padding, border ou margin.

### border-box

As propriedades de **width** e de **height** incluem o padding e border, mas não incluem a propriedade margin.

## FONT-FAMILY

A propriedade **font-family** especifica a fonte de um elemento. Esta propriedade pode conter vários nomes de fontes, se o navegador não suportar a primeira fonte, ele tentará a próxima fonte até encontrar uma fonte suportada.

Existem dois tipos de nomes de família de fontes:

- Nome de família - o nome de uma família de fontes, como "times", "courier", "arial" etc.
- Família-genérica - O nome de uma família-genérica, como "serif", "sans-serif", "cursiva", "fantasia", "monoespaço".

Comece com a fonte desejada e sempre termine com uma família genérica, para permitir que o navegador escolha uma fonte semelhante na família genérica, se nenhuma outra fonte estiver disponível.

## FONT-SIZE

A propriedade **font-size** define o tamanho do texto. Existem diversas unidades que podemos usar para definir os valores, as mais utilizadas são pixels e em.

Por padrão, 1em é equivalente a 16px, mas você pode alterar este valor.

## FONT-WEIGHT

A propriedade CSS **font-weight** especifica o peso ou a intensidade da fonte (ex.: negrito). Algumas fontes oferecem apenas as opções normal e negrito.

## TEXT-TRANSFORM

A propriedade **text-transform** especifica como capitalizar um texto de um elemento. Pode ser usado para que o texto apareça com todas as letras maiúsculas ou todas as minúsculas, ou com cada palavra em maiúscula.

## TEXT-ALIGN

A propriedade **text-align** especifica o alinhamento horizontal do texto em um elemento.

## LINE-HEIGHT

A propriedade **line-height** especifica a altura de uma linha.

Nota: Valores negativos não são permitidos.

## COLOR

A propriedade **color** é usada para definir a cor do texto. Pode ser definido através do nome da cor (em inglês), valores hexadecimais e rgb.

## BORDER

As propriedades **border** permite que você especifique o estilo, a largura e a cor da borda de um elemento.

## BORDER-RADIUS

A propriedade **border-radius** do CSS define o raio dos cantos do elemento. Assim, é possível fazer cantos arredondados nos elementos. [Clique aqui](#) para ver o conteúdo que fizemos sobre.

## OUTLINE

Um contorno é uma linha desenhada em torno dos elementos, FORA das bordas, para fazer o elemento "se destacar".

## CURSOR

A propriedade **cursor** especifica o cursor do mouse quando o ponteiro do mouse está sobre um elemento.

## BACKGROUND-IMAGE

A propriedade **background-image** define uma imagem de plano de fundo para um elemento. Sempre defina uma cor de fundo a ser usada se a imagem não estiver disponível.

## BACKGROUND-COLOR

A propriedade **background-color** define a cor de fundo de um elemento.

## BACKGROUND-SIZE

A propriedade **background-size** especifica o tamanho das imagens de fundo. O tamanho da imagem pode ser totalmente ou apenas parcialmente comprimido com o objetivo de preservar sua proporção.

## BACKGROUND-REPEAT

A propriedade **background-repeat** define se ou como uma imagem de fundo será repetida. Por padrão, uma imagem de fundo é repetida tanto vertical quanto horizontalmente.

## DISPLAY: FLEX

Antes do **flexbox**, como geralmente é chamado, havia quatro modos de layout:

- Bloco, para seções em uma página da web;
- Inline, para texto;
- Tabela, para dados de tabela bidimensional;
- Posicionado, para a posição explícita de um elemento;

O flexbox torna mais fácil projetar uma estrutura de layout responsiva e flexível sem usar flutuação ou posicionamento.

Propriedades como **justify-content** e **align-items** são usadas para fazer o alinhamento dos elementos, sendo respectivamente o alinhamento horizontal e vertical.

## DISPLAY: GRID

O CSS Grid oferece um sistema de layout baseado em grade, com linhas e colunas, facilitando o design de páginas da web sem ter que usar flutuadores e posicionamento.

## POSITION

O posicionamento permite retirar elementos do fluxo normal de layout do documento e fazer com que se comportem de maneira diferente, por exemplo um em cima do outro, ou sempre permanecendo no mesmo lugar dentro da janela de visualização do navegador.

Existe diversos valores para esta propriedade, vamos falar sobre as que utilizamos dentro do nosso projeto: **"absolute"** e **"relative"**.

### absolute

Um elemento com **"position: absolute"** é posicionado em relação ao ancestral posicionado mais próximo. Porém, se este elemento posicionado de forma absoluta não tiver ancestrais posicionados, ele usará o corpo do documento e se moverá junto com a rolagem da página.

Nota: Um elemento "posicionado" é aquele que possui qualquer posição que não seja static.

### relative

Um elemento com **"position: relative"** está posicionado em relação à sua posição normal.



Após definirmos o elemento como posicionado, movemos ele pela tela com as propriedades **top**, **left**, **bottom** e **right**.

## Z-INDEX

A propriedade **z-index** especifica a ordem da pilha de um elemento. Um elemento com uma ordem de pilha maior está sempre na frente de um elemento com uma ordem de pilha mais baixa.

Nota: z-index só funciona quando atribui a propriedade **"position"** ao elemento.

## BOX-SHADOW

O **box-shadow** é utilizado para adicionar efeitos de sombra em volta de um elemento.

## TRANSITION

As transições CSS permitem que você altere os valores das propriedades suavemente, durante um determinado período ou estado do elemento.

## TRANSFORM

A propriedade **transform** aplica uma transformação 2D ou 3D a um elemento. Esta propriedade permite girar, dimensionar, mover, inclinar, etc., elementos.

## ANIMATION

Esta propriedade do CSS permite a animar de elementos HTML.

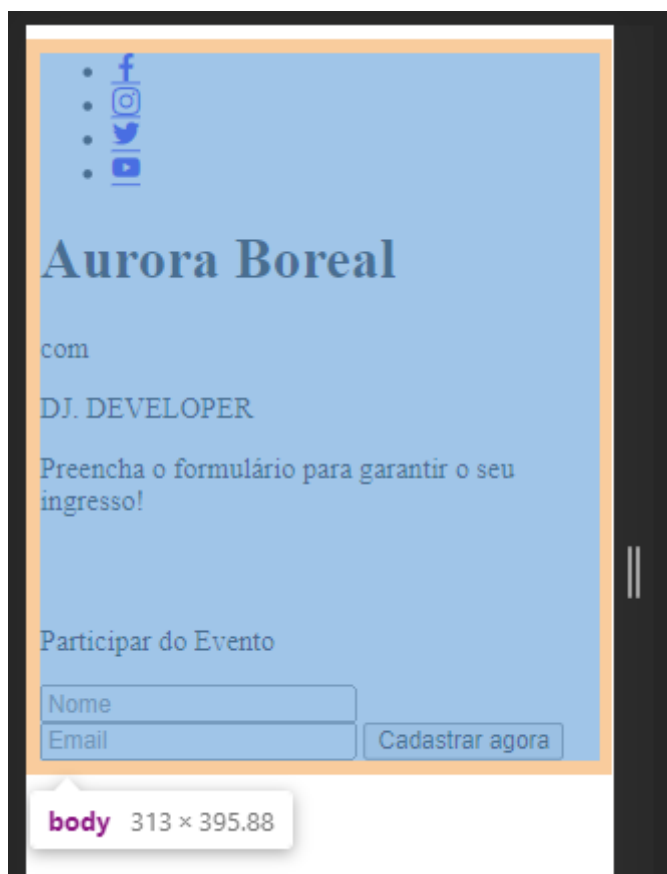
### @keyframes

Quando você especifica estilos CSS dentro da regra **@keyframes**, a animação mudará gradualmente do estilo atual para o novo estilo em determinados momentos.

# Código CSS

Primeiramente, precisamos resetar as propriedades que vem por padrão com o navegador, para assim conseguirmos estilizar nosso projeto com apenas o que nós definirmos.

Como podemos ver no exemplo abaixo, o azul representa o conteúdo de body enquanto o laranja é a margem que o navegador aplica.



O CSS separamos por partes, e é interessante você fazer cada parte e atualizar, para assim ver a diferença entre cada nova estilização.

## CSS Reset

No nosso CSS Reset, vamos tirar o enchimento e a margem padrão dos elementos, definir a família de fontes e com a propriedade **"box-sizing"**, alterar a forma como é calculada as larguras e alturas dos elementos.

Também iremos retirar o sublinhado dos links e, os marcadores dos itens de lista. Sendo assim, teremos algo semelhante a isso:

```
*{
  padding: 0;
  margin: 0;
  font-family: Poppins, sans-serif;
  box-sizing: border-box;
}
a{ text-decoration: none; }
li{ list-style-type: none; }
```



## BODY

No body, iremos adicionar uma imagem de fundo, juntamente com sua altura e largura.

```
body{
  background-image: url(../img/background.jpg);
  background-repeat: no-repeat;
  background-size: cover;
  width: 100%;
  height: 100vh;
}
```



## .CONTENT

Lembra que utilizamos essa classe diversas vezes?

Pois bem, o objetivo deste contêiner é limitar a largura máxima do conteúdo do site, sendo assim, quando a largura for maior que a definida ele irá se ajustar sempre ao meio, já que definimos a propriedade **"margin"** como **"auto"** nas laterais.

Ele também tem a função de ser a nossa caixa flexível, que engloba os flex-items.

```
.content{
  max-width: 1100px;
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin: 0 auto;
  position: relative;
}
```



## HEADER

Agora, que finalizamos as "configurações gerais" vamos começar a estilizar o projeto pelo header. Definimos então a distância entre ele e o topo do site, além da sua altura.

```
header{
  padding-top: 5vh;
  height: 10vh;
}
```



## .LOGO

Vamos adicionar o logo ao nosso projeto, linkando a imagem através do background e definindo os seus valores de altura e largura.

```
.logo{  
  background-image: url(../img/logo-190x75.png);  
  width: 190px;  
  height: 75px;  
}
```



## UL

Definimos a lista como **"flex"** para que os itens de lista fiquem um ao lado do outro.

```
header ul{  
  display: flex;  
}
```



## A

A tag `<a>` é o botão das redes sociais, definiremos sua altura e largura para transformar em um círculo.

Definimos ele como um contêiner grid para que o ícone fique alinhado ao centro tanto horizontal quanto verticalmente. As demais características são a cor de fundo e a transição.

```
header li a{
  width: 40px;
  height: 40px;
  background-color: #BB6BD9;
  border-radius: 50%;
  display: grid;
  place-items: center;
  transition: background-color .3s ease-in-out;
  margin-right: 16px;
}
```



I

E para finalizar, definimos o tamanho do ícone de sua cor.

```
header i{
  font-size: 14px;
  color: #fff;
}
```



## MAIN

No main, apenas teremos que afastar ele do topo.

```
main{
  padding-top: 20vh;
}
```



## H1

Vamos estilizar o título, que é um elemento com bastante destaque e importância da nossa página.

```
h1{
  font-size: 100px;
  line-height: .8;
  color: #fff;
  text-transform: uppercase;
  font-weight: 800;
}
```



## .DJ

Abaixo do título, temos um destaque que é o DJ, ambos os elementos tem a mesma cor, peso da fonte e estão com letras maiúsculas.

```
.dj{
  color: #E5215C;
  text-transform: uppercase;
  font-weight: 700;
  margin-top: 10px;
}
```



Precisaremos alterar apenas as fontes individualmente, e aproximar o **<p>** do **<span>** utilizando valores negativos.

```
.dj span{
  font-size: 20px;
```



```
}  
  
.dj p{  
  font-size: 48px;  
  margin-top: -15px;  
}
```



## .CTA

Nossa caixa de chamada para ação tem dois elementos filhos, portanto definimos ele como uma caixa flexível.

```
.cta{  
  margin-top: 20px;  
  display: flex;  
  align-items: center;  
}
```



Para que o **<p>** que por padrão ocupa a linha inteira, fique com uma linha abaixo da outra definiremos a sua altura. Além disso, definimos suas características textuais.

```
.cta p{  
  color: #e5e5e5;  
  font-size: 20px;  
  font-weight: 500;  
  width: 280px;  
  max-width: 280px;  
}
```



Agora, precisamos trabalhar o nosso botão pulsante. Ele tem a largura e altura definida, pois o transformaremos em um círculo, utilizamos a propriedade **"grid"** para alinhar a imagem que tem dentro do botão ao centro e, a propriedade **"animation"** fica responsável por definir a animação **"pulse"** e seus valores.

```
.cta .button{
  width: 45px;
  height: 45px;
  background-color: #E5215C;
  display: grid;
  place-items: center;
  border-radius: 50%;
  margin-left: 20px;
  animation: pulse 2s infinite;
}
```



Para finalizar, apenas precisamos alterar a largura da imagem de seta que está dentro do botão.

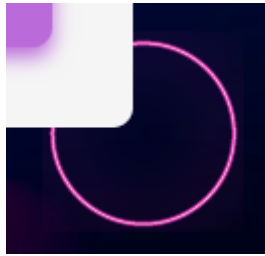
```
.cta img{
  width: 16px;
}
```



## **.FORM**

Essa é uma caixa flexível, com alinhamento em coluna, para que os elementos fiquem um abaixo do outro. Para isso definimos a **"flex-direction"** como **"column"**.

Também precisamos trabalhar com as propriedades **"position"** e **"z-index"**, para que o formulário fique na frente do círculo **".light03"**.



```
.form{
  padding: 40px;
  max-width: 400px;
  background-color: #f5f5f5;
  display: flex;
  flex-direction: column;
  border-radius: 10px;
  position: relative;
  z-index: 1;
}
```



Precisamos alterar o tamanho e o alinhamento do texto em destaque do formulário.

```
.form p{
  font-size: 24px;
  text-align: center;
}
```



Vamos definir a largura dos inputs como máxima, para que agora sim eles fiquem um abaixo do outro, como definimos com o **"flex-direction"**.

Precisamos zerar a borda, e o outline que aparece quando selecionamos o campo.

```
.form input{
  width: 100%;
  margin-top: 30px;
  padding: 15px;
  border-radius: 10px;
  border: 0;
  outline: 0;
}
```



O input de tipo **"submit"** é o nosso botão de enviar o formulário. Então, ele tem uma estilização diferente dos demais inputs.

Por padrão, o campo **"submit"** não aparece para o usuário como clicável (ponteiro do mouse), por mais que ele seja. Portanto precisamos definir a propriedade **"cursor"**, para mostrar ao usuário que ali temos um botão.

Selecionamos ele através do seu atributo, mas poderíamos também ter adicionado uma **"class"**, por exemplo.

```
.form input[type="submit"]{
  background-color: #BB6BD9;
  color: #fff;
  font-weight: 700;
  box-shadow: 0px 5px 10px #BB6BD9;
  cursor: pointer;
}
```



## FOOTER

O **<footer>** tem uma posição absoluta, então poderíamos colocar ele em qualquer canto da tela, mas como é um rodapé definimos que ele precisa estar no final.

```
footer{
  background: #121114;
  width: 100%;
  padding: 5px 0;
  color: #e5e5e5;
  position: absolute;
  bottom: 0;
}
```



## LIGHTS

As luzes terão características semelhantes, então podemos adicionar diversos seletores de uma vez só apenas separando-os por vírgulas.

Como podemos ver, tanto as luzes quanto o **<footer>** possuem a posição absoluta, mas existe uma diferença entre eles. O nosso **".content"** possui posição relativa, então as luzes terão posições absolutas, mas relativas ao **".content"**.

Isso significa que conseguimos colocá-las em qualquer posição, mas dentro do **".content"** e não da tela toda, como foi com o **<footer>**.

```
.light01, .light02, .light03{
  background-size: contain;
  position: absolute;
}
```



Para conseguirmos alinhar corretamente as luzes como no nosso protótipo, precisamos utilizar valores negativos.

Na **".light02"** precisamos utilizar a propriedade **"transform"** para rodar um pouco o elemento, ficando assim mais fiel ao protótipo.

Na **".light03"** adicionamos a propriedade **"z-index"** com o valor **"0"**, para que ele permaneça atrás do formulário, que definimos como **"1"**. Quanto maior o **"z-index"**, mais acima ele ficará dos demais elementos.

```
.light01{
  background-image: url(../img/luz1.png);
  width: 56px;
  height: 71px;
  bottom: -10vh;
  left: -3%;
}
.light02{
  background-image: url(../img/luz2.png);
  width: 71px;
  height: 56px;
  top: -5vh;
  left: -6%;
  transform: rotate(-15deg);
}
.light03{
  background-image: url(../img/luz3.png);
  width: 100px;
  height: 100px;
  bottom: -8vh;
  right: -5%;
  z-index: 0;
}
```



## :HOVER

Com o passar do mouse do usuário sobre os botões (redes sociais e formulário), apenas trocaremos a sua cor de fundo.

```
/* REDES SOCIAIS :HOVER */
header a:hover, .form input[type="submit"]:hover{
    background-color: #9938DF;
}
```



## PULSE

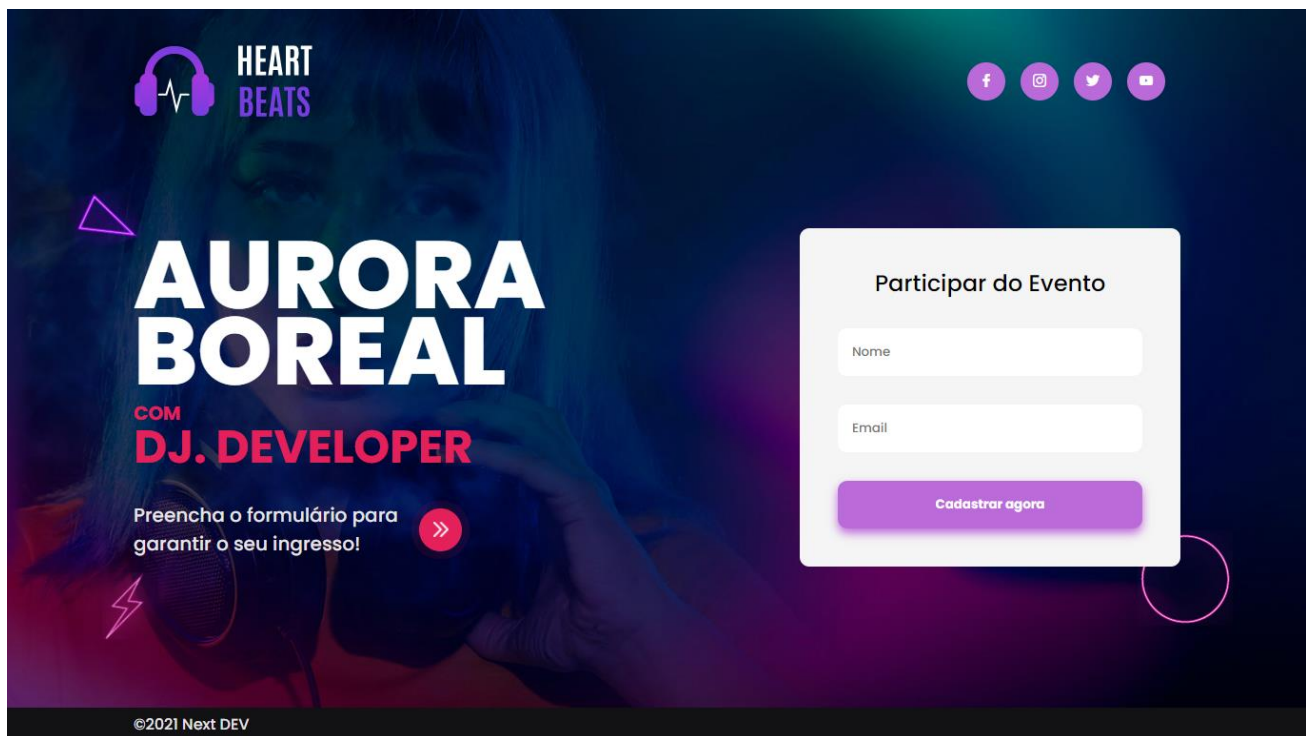
Nossa animação **"pulse"**, executará gradualmente estes valores em **"2s"** e **"infinitamente"**.

```
@keyframes pulse{
    0%{
        box-shadow: 0 0 0 0 rgba(229, 33, 92, .5);
    }
    70%{
        box-shadow: 0 0 0 10px rgba(229, 33, 92, 0);
    }
    100%{
        box-shadow: 0 0 0 0 rgba(229, 33, 92, 0);
    }
}
```



## Projeto Final

Agora é hora de ver como o projeto está DEV!



Se estiver desta forma, parabéns DEV! Finalizamos o nosso projeto, juntos



Nos links abaixo, você consegue baixar o código final deste projeto:

[Github](#) ou [Mega](#)





# Considerações

# Projeto

---

DEV, se você ficou um pouco confuso ao desenvolver este projeto, fica tranquilo tá?! Não se cobre demais, tudo isso faz parte do aprendizado.

O HTML e CSS são linguagens que você está se adaptando agora, interprete como se estivesse aprendendo um idioma diferente, totalmente novo pra você! E não é do dia pra noite que saímos falando japonês, por exemplo, não é mesmo? É preciso praticar, escrever e entender o que está fazendo.

Por isso que, como já passamos por esta situação, sabemos como ajudar você, da forma que gostaríamos que tivesse sido conosco. Então, criamos este ebook – **com muito carinho** - com toda a parte técnica do projeto, pra que você tenha um material mais completo para estudar. E isso pode fazer toda a diferença, já que às vezes a falta de uma informação faz com que percamos horas travados em um ponto do projeto.

O objetivo de desenvolver um projeto complexo, é mostrar na prática o que o estudo do HTML e CSS pode proporcionar pra você. E, logo você vai perceber que ele nem é complexo de verdade 😊

Espero que tenham gostado do conteúdo, abraços.



**Bianca Ramal**

📷 @bianca.ramal

Instrutora no curso  
“Desenvolvendo  
Meu Primeiro Site”