Desenvolvimento Web

Relatório de Projeto Final

Meu Portfólio

Portfólio construído pelo aluno:

- Junior Nelson Dias Delgado

Relatório Completo do Código HTML

Estrutura Geral do Documento HTML (<!DOCTYPE html> e <html lang="pt-BR">)

- <!DOCTYPE html> - Declara o tipo de documento como HTML5, indicando ao navegador como interpretar o código.

- <html lang="en"> - Define a linguagem do documento como Português do Brasil, facilitando a correta interpretação por leitores automáticos e mecanismos de busca.

- <head>

Meta Tags:

- Meta Charset (<meta charset="UTF-8" />) - Define a codificação de caracteres como UTF-8, assegurando suporte para caracteres internacionais e especiais.

- Meta Viewport (<meta name="viewport" content="width=device-width, initial-scale=1.0" />) - Controla a escala inicial e a largura da página em dispositivos móveis, garantindo uma visualização adequada e responsiva.

Título da Página:

- <title>Junior Delgado Portfólio</title> - Define o título da página exibido na aba ou barra de título do navegador como "Junior Delgado Portfólio". Essencial para identificação da página e SEO.

Estilos (CSS):

- <link rel="stylesheet" href="style.css" /> - Importa o arquivo externo “style.css”, que contém estilos para a página. Separar o CSS do HTML facilita a manutenção e permite estilos consistentes em toda a aplicação.

```html
<head>
    <!-- Meta tags -->
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <!-- title -->
    <title>Junior Delgado Portfólio</title>

    <!-- CSS -->
    <link rel="stylesheet" href="style.css" />
  </head>
```

<body>

Navbar (<nav>):

- <nav class="navbar">...</nav> - Contém links de navegação para diferentes seções da página, como Início, Sobre, Habilidades, Projetos e Contato. Inclui uma versão móvel para menus responsivos.

```
<!-- navbar -->
    <nav class="navbar">
      <div class="navbar__content max-width mb-0">
        <a href="/">
          <img src="logo.png" alt="Junior Delgado" />
        </a>
        <ul class="navbar__links">
          <li><a href="#header"><strong>Início</strong></a></li>
            <li><a href="#about"><strong>Sobre</strong></a></li>
            <li><a href="#skills"><strong>Habilidades</strong></a></li>
            <li><a href="#projects"><strong>Projetos</strong></a></li>
            <li><a href="#footer"><strong>Contato</strong></a></li>
        </ul>
        <div class="navbar__mobile">
          <img src="menu.svg" alt="menu" class="burguer" />
          <ul class="mobile__links">
            <li><a href="#header"><strong>Início</strong></a></li>
            <li><a href="#about"><strong>Sobre</strong></a></li>
            <li><a href="#skills"><strong>Habilidades</strong></a></li>
            <li><a href="#projects"><strong>Projetos</strong></a></li>
            <li><a href="#footer"><strong>Contato</strong></a></li>
          </ul>
        </div>
      </div>
    </nav>
```

Header (<header id="header">...</header>)

- <header id="header">...</header> - Apresenta uma introdução pessoal com título, descrição e botão de chamada para ação (CTA), acompanhado de uma imagem ilustrativa.

```
<!-- header -->
  <header id="header" class="max-width">
    <div class="header__left">
      <h1>Olá! Sou Junior Delgado Desenvolvedor Fullstack</h1>
      <p>
        Desenvolvedor com ampla experiência em criar sites e aplicativos web que se adaptam e crescem conforme o necessário.
        <br>Me dedico a desenvolver soluções tecnológicas que solucionam desafios reais e trazem benefícios aos usuários.

      </p>
      <a href="#about" class="btn btn-primary">Saiba Mais!</a>
    </div>
    <div class="header__right">
      <div class="header__image">
        <img src="computador.png" id="computador" alt=""/>
      </div>
    </div>
  </header>

```

Sobre (<section id="about">...</section>)

- <section id="about">...</section> - Descreve informações pessoais, acompanhadas por uma imagem, sobre Junior Delgado como desenvolvedor fullstack.

```
<!-- Sobre -->
    <section id="about" class="about max-width">
      <div class="about__left">
        <h2 class="secondary-title">Mas afinal,<br> Quem sou eu ?</h2>
        <br>
        <a href="#"><img id="img-ju" src="Junior.jpeg" alt="Junior Delgado"></a>
      </div>
      <div class="about__right">
        <h3 class="tertiary-title mb-m">
          Fullstack developer apaixonado por tecnologia e soluções criativas
        </h3>
        <br>
        <p>
          Olá, eu sou Junior Delgado, um desenvolvedor fullstack que adora
           tecnologia e soluções criativas para qualquer necessidade !
        </p>
        <p>
          Tenho mais de 5 anos de experiência em desenvolvimento web
           e já desenvolvi soluções com várias tecnologias e plataformas
           para dar ao cliente a solução robusta e escalável que eles merecem !
        </p>
        <p>
          Eu amo encontrar soluções inovadoras e criativas para problemas
           complexos, e estou sempre aprendendo e me desenvolvendo como profissional !
        </p>
        <p>
          Acredite, suas soluções estão aqui comigo !
        </p>
      </div>
    </section>
```

Habilidades (<section id="skills">...</section>)

- <section id="skills">...</section> - Lista habilidades divididas em Front-end e Back-end, detalhando tecnologias e ferramentas em tabelas.

```html
<!-- habilidades -->
    <section id="skills" class="skills ">
      <div class="skills__content ">
        <h2 class="secondary-title">Minhas Habilidades</h2>
        <p class="description">
          Explore as competências e recursos que possuo e que me capacitam<br>
           a desenvolver soluções inovadoras e práticas para os meus clientes.
        </p>
        <br>
        <table border="1">
          <tr>
            <th>Front-end</th>
            <th>Back-end</th>
          </tr>
          <tr>
            <td>Fundamentos essenciais para desenvolver interfaces web interativas e responsivas.</td>
            <td>Proficiência em uma ou mais linguagens como JavaScript (Node.js), Python, Ruby, Java, C#, etc.</td>
          </tr>
          <tr>
            <td>Experiência em frameworks populares como React, Angular, Vue.js, etc.</td>
            <td>Conhecimento em frameworks como Express.js, Django, Spring, Ruby on Rails, etc.</td>
          </tr>
          <tr>
            <td>Conhecimento em ferramentas como Sass, Less para melhorar a produtividade e organização do código.</td>
            <td>Experiência com bancos de dados relacionais (MySQL, PostgreSQL) e não relacionais (MongoDB, Redis).</td>
          </tr>
          <tr>
            <td>Habilidade de criar interfaces que se adaptem a diferentes dispositivos.</td>
            <td>Familiaridade com implementações de CI/CD, containers (Docker), orquestração (Kubernetes), etc.</td>
          </tr>
          <tr>
            <td>Familiaridade com sistemas de controle de versão como Git.</td>
            <td>Compreensão dos princípios de segurança, autenticação e autorização.</td>
          </tr>
        </table>
      </div>
    </section>
```

Projetos (<section id="projects">...</section>)

- <section id="projects">...</section> - Apresenta projetos em destaque com imagens, títulos e descrições breves, cada um com um botão de "Leia Mais".

```html
<!-- Projetos -->
    <section id="projects" class="projects max-width">
      <div class="projects__content">
        <h2 class="secondary-title">Projetos em Destaque</h2>
        <p>
          De ideias abstratas a projetos bem-sucedidos, aqui estão alguns dos
          meus trabalhos mais recentes
        </p>
      </div>
      <ul>
        <li>
          <div class="image">
            <img src="cadastro.png" alt="Sistema de Cadastro Social Unico" />
          </div>
          <div class="projects__info">
            <h3 class="tertiary-title">
              Sistema de Cadastro Social Unico
            </h3>
            <p>
              Sistema web desenvolvido diretamente para a Camara Municipal de Sao Vicente, com funcionalidades
              como incriçao no CSU, consulta de informaçoes e
              acompanhamento de serviços em que o CSU pode ser utilizado.
            </p>
            <a href="#Registration-form"><button id="openFormBtn1" class="openFormButton">Leia Mais</button></a>
          </div>
        </li>
        <li class="projects__reversed-list">
          <div class="image">
            <img src="CVshop.jpg" alt="Loja Virtual CVshop.cv" />
          </div>
          <div class="projects__info">
            <h3 class="tertiary-title">
              Loja Virtual CVshop.cv
            </h3>
            <p>
              Site de e-commerce totalmente caboverdiano, desenvolvido do zero com tecnologias como React
              e Node.js, além de integração com diversos meios de pagamento e
              entrega.
            </p>
            <a href="#Registration-form"><button id="openFormBtn2" class="openFormButton">Leia Mais</button></a>
          </div>
        </li>
        <li>
          <div class="image">
            <img src="Correios-Cabo-Verde.jpg" alt="App de entregas dos Correios de Cabo Verde" />
          </div>
          <div class="projects__info">
            <h3 class="tertiary-title">
              App de entregas dos Correios de Cabo Verde !
            </h3>
            <p>
              Aplicativo mobile desenvolvido para a os Correios CV,
              permitindo a entrega com facilidade através da localizaçao
              exata do destinatario.
            </p>
            <a href="#Registration-form"><button id="openFormBtn3" class="openFormButton">Leia Mais</button></a>
          </div>
        </li>
      </ul>
    </section>
```

Footer (<footer id="footer">...</footer>)

- <footer id="footer">...</footer> - Inclui um vídeo, informações de contato, links para redes sociais e um formulário de registro para acessar projetos mais detalhadamente.

```
<!-- Footer -->
    <footer id="footer">
      <div class="footer__content max-width mb-0">
        <h2 class="tertiary-title">Veja o video:</h2>
        <video controls ><source src="programming.mp4" type="video/mp4"></video>
        <h2 class="tertiary-title mb-s">Caso queira contactar-me:</h2>
        <p class="mb-m">
          Ficarei feliz em saber mais sobre seus projetos e como podemos
          trabalhar juntos para torná-los realidade.
        </p>
        <div class="footer__contact mb-m"><br>
          <span class="mb-m"><u>juprogramador@gmail.com</u> </span>
          <span><u>+238 9571758</u></span><br>
          <p>
            Que tal se conectar comigo nas redes sociais abaixo e saber mais sobre
            meu trabalho?
          </p>
        </div>
        <ul id="links_socialmedia" class="mb-l">
          <li>
            <a href="#"><img src="linkedin.jpeg" alt="Linkedin" /></a>
          </li>
          <li>
            <a href="#"><img src="instagram.png" alt="Instagram" /></a>
          </li>
          <li>
            <a href="#"><img src="facebook.png" alt="Facebook" /></a>
          </li>
        </ul>
```

Script (<script src="script.js"></script>)

- <script src="script.js"></script> - Importa o arquivo JavaScript “script.js” para funcionalidades dinâmicas e interativas na página.

```
<!-- script -->
    <script src="script.js"></script>
  </body>
</html>
```

Relatório Completo do Código JavaScrip

1- Esse código JavaScript implementa funcionalidades interativas e de validação para um formulário de registro e também controla o comportamento dinâmico de uma barra de navegação móvel e outros elementos na página.

Este código está associado a um formulário de registro HTML com algumas validações básicas antes de enviar os dados para o servidor. Vamos analisar passo a passo o que ele faz:

Validação

```
document
    .getElementById("Registration-form")
    .addEventListener("submit", function(event){
        event.preventDefault();

        let username = document.getElementById("username").value;
        let email = document.getElementById("email").value;
        let password = document.getElementById("password").value;
        let confirmPassword = document.getElementById("confirmPassword").value;

        console.log(username + "" + email + "" + password + "" + confirmPassword );
        username = username.trim()
        if (username.trim() === ""){
            document.getElementById ("usernameError").innerHTML = "Nome de Utilizador é obrigatório"
            return;
        }

        if (email.trim() === ""){
            document.getElementById ("emailError").innerHTML = "Email é obrigatório"
            return;
        }

        if (password.trim() === ""){
            document.getElementById ("passwordError").innerHTML = "Password é obrigatório"
            return;
        }else if (password.length < 6){
             document.getElementById ("passwordError").innerHTML = "Deve conter mais de 5 caracteres"
             return
        }

         if (confirmPassword.trim() === ""){
            document.getElementById ("confirmPasswordError").innerHTML = "É obrigatório confirmar a password"
            return;
        }else if(confirmPassword != password){
            document.getElementById ("confirmPasswordError").innerHTML = "Passwords diferentes"
            return;
        }

        alert("Formulário submetido com sucesso")
    })
```

a) Evento de Envio (submit):

- O código começa selecionando o formulário de registro usando getElementById com o ID "Registration-form".

- Em seguida, adiciona um ouvinte de evento “submit”. Quando o formulário é enviado, a função fornecida como segundo argumento é acionada.

- event.preventDefault(), previne o comportamento padrão do formulário, que é enviar os dados para uma página específica e atualizá-la.

b) Obtenção de Valores:

- Aqui, os valores dos campos do formulário são obtidos usando getElementById para os IDs "username", "email", "password" e "confirmPassword".

c) Saída no Console:

- Este “console.log” simplesmente imprime os valores dos campos no console do navegador. É útil para depuração e verificar se os valores estão sendo capturados corretamente.

d) Validação do Nome de Usuário:

- Remove espaços em branco desnecessários do nome de usuário usando `trim()`.

- Verifica se o campo está vazio e, se estiver, exibe uma mensagem de erro no elemento com ID "usernameError" e interrompe o envio do formulário usando `return`.

e) Validação do Email:

- Similar ao nome de usuário, verifica se o campo de email está vazio. Se estiver, exibe uma mensagem de erro no elemento com ID "emailError" e interrompe o envio do formulário.

f) Validação da Senha:

- Verifica se o campo de senha está vazio. Se estiver, exibe uma mensagem de erro no elemento com ID "passwordError" e interrompe o envio do formulário.

- Além disso, verifica se a senha tem menos de 6 caracteres. Se sim, exibe uma mensagem de erro apropriada e interrompe o envio do formulário.

g) Validação da Confirmação de Senha:

- Verifica se o campo de confirmação de senha está vazio. Se estiver, exibe uma mensagem de erro no elemento com ID "confirmPasswordError" e interrompe o envio do formulário.

- Verifica se a confirmação de senha é diferente da senha digitada anteriormente. Se forem diferentes, exibe uma mensagem de erro apropriada e interrompe o envio do formulário.

h) Envio do Formulário:

- Se todas as validações passarem (isto é, nenhum dos `return` anteriores for acionado), exibe um alerta informando que o formulário foi submetido com sucesso.

Resumidamente, este código JavaScript assegura que todos os campos obrigatórios (`username`, `email`, `password` e `confirmPassword`) estejam preenchidos corretamente antes de permitir que o formulário seja enviado. Ele também fornece feedback imediato ao usuário caso algum campo não esteja conforme o esperado.

```
document.getElementById('Registration-form').addEventListener('submit', function(event) {
    event.preventDefault();

        const username =
        document.getElementById('username').value;
        const email =
        document.getElementById('email').value;
        const password =
        document.getElementById('password').value;
        const confirmPassword =
        document.getElementById('confirmPassword').value;

        if (username, email, password && confirmPassword) {

    document.getElementById('Registration-form').style.display = 'none';

                }else {

                    alert('Por Favor, preencha todos os campos.');
                }
    });
```

2- Esse código JavaScript está associado a um formulário de registro (`Registration-form`) em uma página HTML. Vamos analisar o que acontece passo a passo:

a) Seleção do Elemento e Adição de Evento:

- O código seleciona o elemento do formulário com o ID "Registration-form".

- Adiciona um ouvinte de evento “submit”, o que significa que quando o formulário for enviado, a função especificada será executada.

b) Prevenção do Comportamento Padrão:

- Esta linha previne o comportamento padrão do formulário, que é enviar os dados para uma página e atualizá-la.

c) Obtenção de Valores dos Campos:

- Estes trechos obtêm os valores dos campos de entrada (`input`) do formulário com os IDs "username", "email", "password" e "confirmPassword" e os armazenam nas variáveis correspondentes.

d) Condição de Verificação:

- Nesta linha, parece haver um erro na lógica. A expressão `username, email, password && confirmPassword` não verifica corretamente se todos os campos estão preenchidos. Na verdade, o operador `,` neste contexto avalia cada expressão separadamente e retorna o valor da última expressão avaliada (`confirmPassword` neste caso).

e) Ação se Todos os Campos Estiverem Preenchidos:

- Se todos os campos estiverem preenchidos (como deveria ser corretamente verificado), o formulário com o ID "Registration-form" terá seu estilo de exibição definido como 'none', o que faz com que ele desapareça da visualização na página.

f) Ação se Algum Campo Não Estiver Preenchido:

- Se algum dos campos estiver vazio, será exibido um alerta com a mensagem "Por Favor, preencha todos os campos.".

- Isso fornece um feedback para o usuário informando que é necessário preencher todos os campos antes de enviar o formulário.

Resumo:

Esse código JavaScript tenta verificar se todos os campos do formulário de registro estão preenchidos antes de permitir que o formulário seja enviado. No entanto, há um erro na condição de verificação (“if”), onde a expressão “username, email, password && confirmPassword” não faz a verificação correta dos campos. Além disso, após a verificação correta, ele esconde o formulário se todos os campos estiverem preenchidos ou exibe um alerta se algum campo estiver vazio.

Obs:

- Este trecho de código adiciona um listener de evento para o formulário com id "Registration-form".

- Ele intercepta a submissão do formulário (“event.preventDefault()”) para realizar validações personalizadas.

- Captura os valores dos campos do formulário, verifica se estão preenchidos e se a senha e a confirmação da senha correspondem.

- Caso haja erros de validação, exibe mensagens de erro ao lado dos campos correspondentes.

- Se todas as validações forem bem-sucedidas, exibe um alerta indicando que o formulário foi submetido com sucesso.

```
const navbar = document.querySelector('.navbar');
const mobileNavbar = document.querySelector('.navbar__mobile');
const button = document.querySelector('.burguer');

button.addEventListener('click', function () {
  mobileNavbar.classList.toggle('active');
});

window.addEventListener('scroll', function () {
  if (this.window.pageYOffset > 0) return navbar.classList.add('active');
  return navbar.classList.remove('active');
});
```

3- Esse código JavaScript manipula a interação com a barra de navegação (“navbar”) de uma página web. Vamos explicar linha por linha:

a) Evento de Clique no Botão: O código adiciona um ouvinte de eventos (event listener) ao botão selecionado (“button”). Esse ouvinte é disparado quando o botão é clicado (“click”).

- Quando o botão é clicado, ele executa uma função que utiliza “mobileNavbar.classList.toggle(“active”)”.

- “classList.toggle”'active”)” adiciona a classe “active” ao “mobileNavbar” se ela não estiver presente, e a remove se estiver presente. Isso permite alternar visualmente o estado do “mobileNavbar”, possivelmente para abrir e fechar um menu móvel.

b) Evento de Rolagem da Janela: O código adiciona um ouvinte de eventos para o evento “scroll” da janela (“window”).

- A cada vez que a página é rolada, a função associada a esse ouvinte é executada.

- Dentro dessa função, verifica-se se o deslocamento vertical da página (“window.pageYOffset”) é maior que 0.

- Se for maior que 0, a classe “active” é adicionada ao elemento “navbar” utilizando “navbar.classList.add(“activ”)”. Isso provavelmente indica que a barra de navegação deve ser alterada de alguma forma quando o usuário rola a página para baixo.

- Caso contrário, se o deslocamento vertical for 0 ou menor, a classe “active” é removida do “navbar” utilizando “navbar.classList.remove(“active”)”.

Resumindo, esse código JavaScript está controlando a interação com a barra de navegação da página. Ele permite abrir e fechar um menu móvel quando um botão é clicado e altera a aparência da barra de navegação com base na posição de rolagem da página.

```
document.addEventListener('DOMContentLoaded', function(){

document.getElementById('openFormBtn1').addEventListener('click', function() {
    document.getElementById('Registration-form').style.display = 'block';
    })

    document.getElementById('openFormBtn2').addEventListener('click', function() {
        document.getElementById('Registration-form').style.display = 'block';
        })

        document.getElementById('openFormBtn3').addEventListener('click', function() {
            document.getElementById('Registration-form').style.display = 'block';
            })
});
```

4- Esse código JavaScript está lidando com a exibição de um formulário de registro em resposta a cliques em diferentes botões na página. Vamos explicar o que cada parte faz:

Explicação Detalhada:

a) “document.addEventListener('DOMContentLoaded”, function() { ... })”:

- “DOMContentLoaded” é um evento que é disparado quando o documento HTML foi completamente carregado e parseado, sem esperar pelo CSS, imagens e outros recursos externos.

- O trecho “function() { ... }” define uma função anônima que será executada quando o evento “DOMContentLoaded” ocorrer. Dentro desta função estão os manipuladores de eventos para os botões.

b) “document.getElementById(“openFormBtn1”).addEventListener(“click”, function() { ... })”:

- “document.getElementById(“openFormBtn1”)” seleciona o elemento que possui o id “openFormBtn1”. Presumivelmente, este é um botão na página.

- “.addEventListener(“click”, function() { ... })” adiciona um ouvinte de eventos para o evento de clique (“click”) no botão selecionado.

- Quando o botão com id “openFormBtn1” é clicado, a função anônima definida (“function() { ... }”) é executada.

- Dentro dessa função, `document.getElementById('Registration-form').style.display = “block”;” é executado. Isso define o estilo CSS do elemento com id “Registration-form” para “display: block;”, tornando o formulário de registro visível na página.

c) document.getElementById(“openFormBtn2”).addEventListener(“click”, function() { ... })” e “document.getElementById(“openFormBtn3”).addEventListener(“click”, function() { ... })”:

- Estes dois blocos de código são semelhantes ao primeiro.

- Eles adicionam ouvintes de eventos de clique aos botões com ids “openFormBtn2” e `openFormBtn3”, respectivamente.

- Quando esses botões são clicados, a mesma função anônima é chamada, que também define o estilo do formulário de registro para “display: block;”, tornando-o visível na página.

Resumo:

O código JavaScript aguarda o carregamento completo do DOM e, em seguida, adiciona ouvintes de eventos de clique a três botões diferentes (“openFormBtn1”, “openFormBtn2”, “openFormBtn3”). Quando qualquer um desses botões é clicado, o formulário de registro (que tem o id “Registration-form”) é exibido alterando seu estilo para “display: block;”. Isso permite que os usuários revelem o formulário ao interagir com qualquer um dos botões designados.

Relatório Completo do Código CSS

Tag `*` (seletor universal)

```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  scroll-behavior: smooth;
  font-family: 'DM Sans', sans-serif;
  background: url();
}
```

- Margens e Preenchimentos: Define margem e preenchimento como zero para todos os elementos.

- Box-sizing: Define “box-sizing: border-box;”, o que inclui a borda e o preenchimento no tamanho total do elemento.

- Scroll-behavior: Define “scroll-behavior: smooth;”, proporcionando um comportamento suave ao rolar.

- Font-family: Define “font-family” como “"DM Sans”, sans-serif;” para todos os elementos.

- Background: Define um fundo padrão com “background: url();”, que provavelmente seria substituído por uma imagem específica.

“Tag html “

```css
html {
  font-size: 62.5%;
}
```

- Define o tamanho da fonte base como 62.5%, o que é comum para facilitar o cálculo de rem em valores mais simples (1 rem = 10 pixels).

Tag ”body”

```
body {
  font-size: 16px;
  margin-top: 150px;
}
```

- Define o tamanho da fonte para o corpo do documento como 16px.

- Adiciona uma margem superior de 150px ao corpo do documento.

Tag “ul”

```
 ul {
   list-style: none;
 }
```

- Remove os estilos padrão de lista (como marcadores ou numeração) de todas as listas não ordenadas (ul).

Tag “a”

```
a {
  text-decoration: none;
  color: inherit;
}
```

- Remove o sublinhado padrão (“text-decoration: none;”) e define a cor do link como herdada (“color: inherit;”) para todos os links (`a`).

Tag “h1”, Classe “.secondary-title”, Classe “.tertiary-title”:

```css
h1 {
  font-size: 64px;
  font-weight: 700;
  color: #02073e;
}

.secondary-title {
  font-size: 40px;
  font-weight: 700;
  color: #02073e;
}

.tertiary-title {
  font-size: 32px;
  font-weight: 500;
  line-height: 45px;
  color: #0f2137;
}
```

- Define estilos para cabeçalhos de nível “h1”, e para as classes “.secondary-title” e “.tertiary-title”. Cada um define tamanho da fonte, peso da fonte e cor específica.

Tag “p”, Classe “.p ”

```css
p,
.p {
  font-weight: 400;
  font-size: 16px;
  line-height: 35px;
}

p:not(:last-of-type),
.p:not(:last-of-type) {
  margin-bottom: 20px;
}
```

- Define estilos padrão para parágrafos (p) e elementos com a classe “.p “.

- Especifica o espaçamento entre parágrafos com “margin-bottom: 20px;”, exceto para o último parágrafo de seu tipo.

Classe “.btn” e suas variantes

```css
.btn {
  background-color: none;
  border: none;
  outline: none;
  padding: 10px 20px;
  display: inline-block;
  border-radius: 15px;
}

.btn.btn-primary {
  background-color: rgb(2, 27, 163);
  color: white;
  font-weight: 400;
  font-size: 14px;
}

.btn.btn-primary:hover {
  transform: scale(1.05);
  background-color: #0a4afa;
}
```

- Define estilos para botões genéricos e para botões com a classe “.btn-primary”.

- Especifica comportamento de hover para botões “.btn-primary”.

Classe ‘.max-width ‘

```css
.max-width {
  width: 100%;
  max-width: 1200px;
  margin: 0 auto;
  margin-bottom: 150px;
}
```

- Define largura máxima, largura e margens para elementos com a classe “.max-width”.

Media Queries

Existem várias regras de media queries que ajustam estilos para diferentes tamanhos de tela (max-width: 1024px, max-width: 768px, max-width: 425px), adaptando fontes, larguras, alinhamentos e visibilidade de elementos conforme necessário.

Id `#Registration-form`

```css
#Registration-form {
  text-align: center;
  justify-content: center;
  width: 100%;
  align-content: center;
  text-align: center;
}

#Registration-form h2 {
  text-align: center;
  margin-bottom: 10px;
}
```

- Define estilos para o formulário com o id “Registration-form”, centralizando e ajustando seu alinhamento.

- Estiliza um título (h2) dentro do formulário com margem inferior e centralização.

Outros Seletores e Classes

Há estilos definidos para elementos como “header”, “.navbar”, “.about”, “.skills”, “.projects”, “footer”, entre outros, cada um com suas próprias características de layout, cores, tamanhos de fonte e comportamentos específicos.

Essas regras CSS em conjunto definem a aparência e o comportamento de diferentes elementos em uma página web, adaptando-os para diferentes dispositivos e interações do usuário.