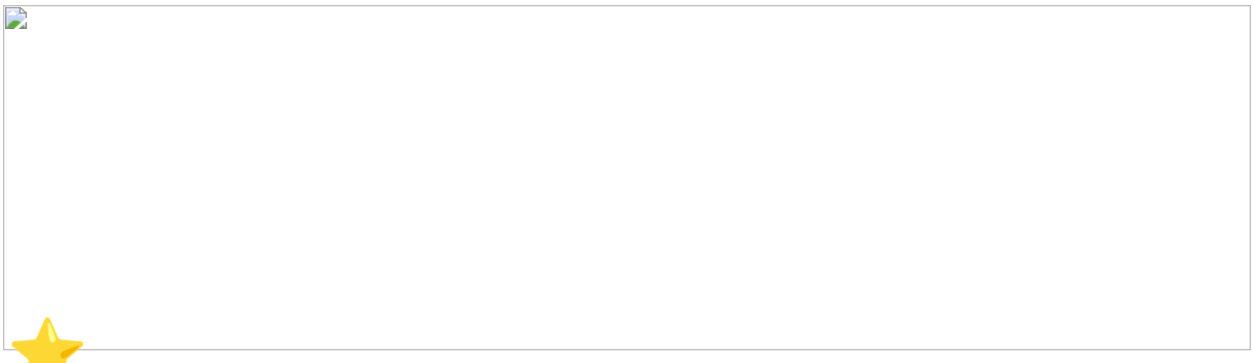


Aula PTech - HTML e CSS

- Projeto do Figma

  > https://www.figma.com/file/ewslOI0uVZFMN3M74W5BAB/Pok%C3%A9mon-Unite?node-id=52%3A2

- Abordar no projeto os conceitos básicos de:
  - HTML
  - CSS
    - Variáveis do CSS;
    - Filtros e efeitos gradientes;
    - Pseudo-elemento;
    - Pseudo-classes;
    - Grid;
    - Flex-box;
    - Responsividade;
    - Requisições para APIs.
- **Explicar sobre BEM CSS**
  1. O BEM é uma **convenção/padrão** de nomenclatura de classes para CSS.
  2. É um acrônimo para **B**loco **E**lemento **M**odificador.
  3. **Bloco**: é uma entidade independente, exemplo: Menu.
  4. **Elemento**: faz parte do bloco, como um elemento filho, exemplo: Item de Menu.
  5. **Modificador:** variante usada para mudar a aparência de um bloco ou de um elemento: exemplo Ativo ou Inativo.
  6. [BLOCO]__[ELEMENTO]--[MODIFICADOR]
- **Estrutura de “componentes” do nosso projeto**
  - Por que pensar antes de codar?
- **Criar projeto**
  - Criar pasta e arquivo `index.html`
  - Plugin Live Server

- **Criar estrutura inicial do HTML**
  - Digite `html:5` e pressione TAB para criar automaticamente a estrutura base do HTML.
  - Mudar titulo da página e linguagem.
  - Fazer um título de teste e rodar utilizando o plugin **Live Server.**
  - Ao final, seu arquivo index.html deve conter uma estrutura parecida com essa:

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="/assets/css/index.css">
  <title>Pokémon Unite</title>
</head>
<body>

</body>
</html>
```

- **Arquivo** `assets/css/reset.css`
  - Os navegadores, por padrão, já colocam estilos nos elementos. Porém, cada navegador coloca seus próprios estilos.
  - Isso pode causar inconsistências de layout ao abrir em navegadores diferentes.
  - Cria-se o reset.css para padronizar esses estilos para todos os navegadores, resetando os estilos padrões como margens, bordas, etc.

```
* {
  padding: 0;
  margin: 0;
  vertical-align: baseline;
  list-style: none;
  border: 0;
  font-size: 16px;
}
```

- **Arquivo** `assets/css/global.css`
  - Arquivo que todas as páginas irão usar, contendo estilos que serão globais em toda a aplicação.

```
@import url('https://fonts.googleapis.com/css2?family=Inter:wght@400;700&display=swap');

:root {
  --primary-color: #FDCF00;
  --secondary-color: #ffa800;
  --dark-color: #111;
  --container-size: 1024px;
  --card-title-background: #232323;
  --light-background: #242424
  --orange: #FFA800;
  --green: #96CB00;
  --yellow: #ECDA00;
  --purple: #7548C2;
  --pink: #D095AB;
}

body {
  font-family: 'Inter', sans-serif;
  background-color: var(--dark-color);
  color: white;
  padding-top: 50px;
}
```

- **Google Fonts** (fonte Inter): site mais utilizado para download e importação de fontes para utilizar em projetos, independente se são digitais ou não.

  Google Fonts
  Making the web more beautiful, fast, and open through great typography
  https://fonts.google.com/specimen/Inter?query=Inter

- Dentro do `:root` estamos informando para a raiz da árvore do nosso documento HTML as variáveis que iremos utilizar ao longo do desenvolvimento. Para criar uma variável, basta colocar: `--nome-da-variavel: valor`. Já definimos nesse passo qual serão algumas cores do nosso projeto e o tamanho do nosso `container`.
- Colocamos o `1024px` como tamanho máximo do `container` (elemento que irá "abraçar" o conteúdo da nossa página) do site por conta do tamanho da maioria das resoluções dos monitores.
- Mobile first: normalmente os sites e aplicações são desenvolvidas primeiro para funcionar e o layout se adaptar para dispositivos móveis e depois para desktop.
  - Depende do contexto em que sua aplicação vai ser usada: mais para desktop ou mais para mobile?
  - Vamos focar no layout para desktop primeiro para ficar melhor de visualizar e ser mais didático por conta do projeto proposto.
  - Mesmo começando pelo desktop, já podemos usar propriedades que sabemos que irá nos ajudar para deixar o site mais responsivo, como, por exemplo, medidas relativas ao tamanho da tela, etc.

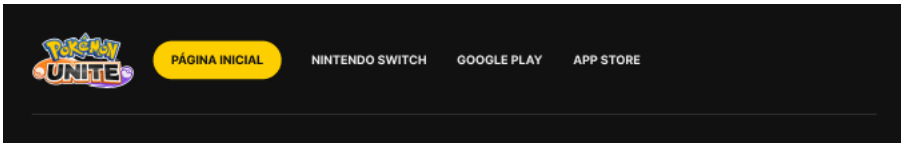
- **Criar arquivo /assets/css/index.css)**
  - Importar o arquivo **reset.css** e **global.css**
  - Chamar folha de estilo dentro da página inicial do HTML

```
<link rel="stylesheet" href="/assets/css/index.css">
```

- **Criar pasta assets/images**
  - Baixar as imagens do Drive e mandar para dentro da pasta /assets/images do projeto

  Ptech - Google Drive
  https://drive.google.com/drive/folders/1-HP--PtmXehU4AFEEMnQgAV1TlmoOhWW?usp=sharing

- **Criar estrutura HTML do Menu**



  - Na página `index.html` do projeto, adicionar logo após a tag `body` a seguinte estrutura HTML para criar o menu:

```
<nav class="container menu__container">
  <div class="menu__logo">
    <div>
      <img src="/assets/images/logo.svg" alt="Pokémon Unite">
    </div>
  </div>
  <ul class="menu">
    <li class="menu__item menu__item--active">
      <a class="menu__link" href="/">Página Inicial</a>
    </li>
    <li class="menu__item">
      <a class="menu__link" href="https://www.nintendo.com/pt_BR/games/detail/pokemon-unite-switch/" target="__blank">
        Nintendo Switch
      </a>
    </li>
    <li class="menu__item">
      <a class="menu__link" href="https://play.google.com/store/apps/details?id=jp.pokemon.pokemonunite"
        target="__blank">
        Google Play
      </a>
    </li>
    <li class="menu__item">
      <a class="menu__link" href="https://apps.apple.com/app/id1512321575" target="__blank">
        App Store
      </a>
    </li>
  </ul>
</nav>
```

- **Estilizar menu do nosso site**
  - **Flexbox**
    - O nome **flexbox** vem de **flexible box** que, em português, significa **caixa flexível**
    - É um conceito do CSS3 que nos ajuda a organizar, alinhar e distribuir os nossos elementos dentro de uma caixa (que podemos chamar de *container*), que nada mais é um elemento **PAI** que contem outros elementos dentro dele.
    - A principal ideia por trás do flexbox é fazer um *container* ter a capacidade de alterar sua largura/altura de acordo com os seus itens e a ordem deles, preenchendo melhor o espaço disponível.
    - O *container* vai expandir os itens para preencher melhor o espaço livre ou reduz para evitar que esses itens “quebrem” o nosso layout.
    - Trabalha com apenas **uma dimensão:** horizontal ou vertical no *container.*

  - Como o menu estará presente em todas as páginas, iremos inserir sua estilização dentro do arquivo `global.css`
  - Adicione dentro da folha de estilos `/assets/css/global.css` as seguintes propriedades:
    - Para que o nossos elementos nao ocupem toda a área da nossa tela, principalmente para monitores muito grandes, iremos estilizar nossa classe `container` respeitando o valor de `1024px` de largura:

```
.container {
  max-width: var(--container-size);
  margin: 0 auto; /* Centraliza na tela o container */
  padding: 0 24px;
}
```

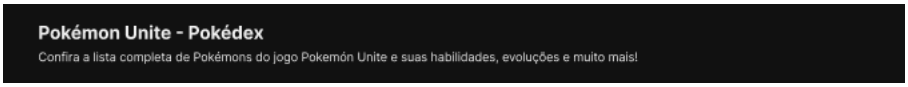
    - Estilizando as classes referentes ao menu:

```
.menu__container {
  display: flex;
  border-bottom: 1px solid rgba(255, 255, 255, .2);
  padding-bottom: 35px;
```

```
  margin-bottom: 35px;
  align-items: center; /* Alinha verticalmente ao centro o logo e os links */
}

.menu__logo {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-right: 24px;
}

.menu {
  width: 100%;
  box-sizing: border-box;
}

.menu__item {
  display: inline-block;
  margin: 0 12px;
  padding: 16px 24px;
}

.menu__link {
  color: white;
  text-decoration: none;
  text-transform: uppercase;
  font-weight: bold;
}

.menu__item--active {
  background: var(--primary-color);
  border-radius: 100px;
}

.menu__item--active .menu__link {
  color: var(--dark-color);
}
```

- `box-sizing: border-box`: essa propriedade com o valor `border-box` faz com que as margens (`margin` e `padding`) sejam incluídas no cálculo de largura e altura do nosso elemento. Para saber mais sobre a propriedade `box-sizing`, clique aqui.

- **Estrutura HTML e CSS do Título da Página + Descrição**



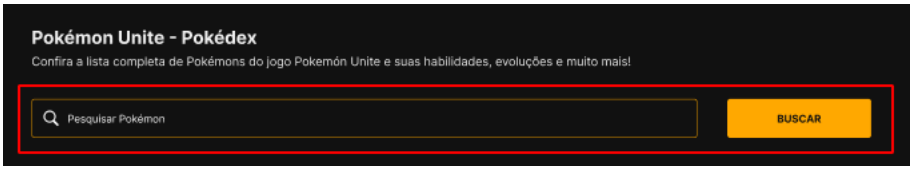
  - Logo após a tag `nav`, adicione a seguinte estrutura HTML

```
<nav>...</nav>
<main class="container">
  <h1>Pokémon Unite - Pokédex</h1>
  <h2>Confira a lista completa de Pokémons do jogo Pokemón Unite e suas habilidades, evoluções e muito mais!</h2>
</main>
```

  - No arquivo `/assets/css/global.css` vamos definir algumas propriedades para as tags `h1` e `h2` que devem ser aplicadas em todas as páginas:

```
h1 {
  font-size: 32px;
  margin-bottom: 8px;
}

h2 {
  font-size: 16px;
  font-weight: normal;
}
```

- **Criando estrutura HTML e CSS do campo de busca**



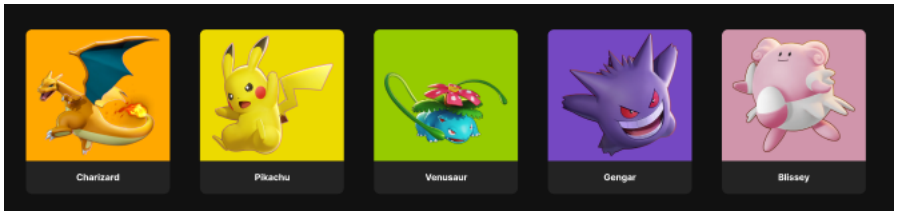
  - Adicione essa estrutura HTML da busca de Pokémons logo após do título e subtítulo:

```
<div class="search__container">
  <input class="search__field" type="text" placeholder="Pesquisar Pokémon" />
  <button class="search__button">Buscar</button>
</div>
```

  - Agora, em `/assets/css/index.css` adicione os seguintes estilos:

```
.search__container {
  margin: 40px 0;
  display: flex;
}

.search__field {
  border: 1px solid var(--primary-color);
  border-radius: 4px;
  background-color: transparent;
  color: white;
  padding: 16px;
  margin-right: 40px;
  width: calc(100% - 40px);
}

.search__field::placeholder {
  color: rgba(255, 255, 255, .8);
}

.search__button {
  padding: 16px 40px;
  background-color: var(--primary-color);
  color: var(--dark-color);
  border-radius: 4px;
  text-transform: uppercase;
  font-size: 14px;
  cursor: pointer;
  transition: background .3s ease-in-out;
}
```

- **Criando a estrutura de cards do Pokémon**



  - Logo após dos elementos da pesquisa de pokémons, adicione a segunte estrutura HTML:

```
<div class="card__container">
  <a id="charizard" class="card orange" href="/charizard.html">
```

```
    <img class="card__image" src="/assets/images/stat/stat-charizard.png" alt="Charizard">
    <div class="card__title">
      Charizard
    </div>
  </a>
</div>
```

- No arquivo `global.css` adicione uma nova variável, que será responsável por definir o tamanho da largura dos nossos cards:

```
:root {
  ...
  --card-size: calc(var(--container-size) / 4 - 32px);
}
```

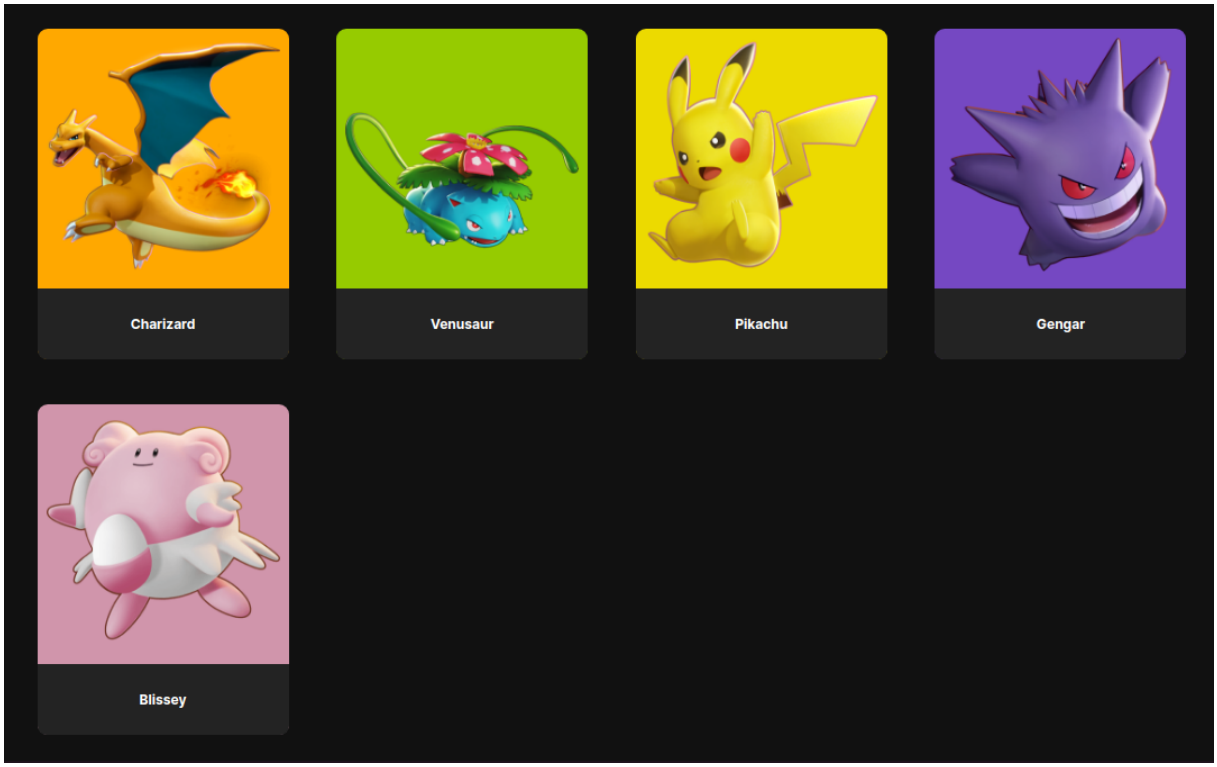
- Na área do nosso `container`, que possui uma largura de `1024px`, queremos que caibam 4 cards por linha, para que os Pokémons sejam exibidos. Por isso, nosso CSS irá dividir o valor de `1024px` em 4 (número de cards que definimos). Desse valor, será subtraído `32px` para que os cards possam ser espaçados.
- Em `/assets/css/index.css` estilize esses elementos:

```
.card__container {
    display: flex;
}

.card {
  width: calc(var(--container-size) / 4 - 32px);
  height: calc(var(--card-size) + 70px); /* Soma-se 70px para exibir o nome do Pokémon */
  border-radius: 9px;
  position: relative;
}

.card__image {
  width: 100%;
  object-fit: contain;
  padding: 8px;
  box-sizing: border-box;
}

.card__title {
  font-size: 12px;
  font-weight: bold;
  color: white;
  text-align: center;
  padding: 24px 0;
  position: absolute;
  bottom: 0;
  width: 100%;
  border-radius: 0 0 8px 8px;
  background-color: var(--card-title-background);
}

.card.orange {
  background-color: var(--orange);
}
```

- Para deixarmos os cards lado a lado e, quando não houver mais espaço, fazê-los ir para a próxima linha, adicionar as propriedades:

```
.card__container {
  display: flex;
  justify-content: space-between;
  flex-wrap: wrap;
  gap: 40px;
}
```

- **Adicionando mais cards de Pokémon**

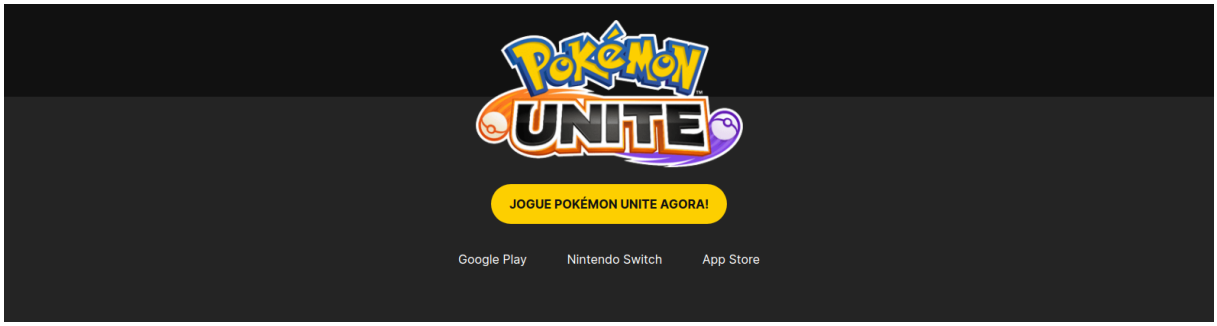



```
<div class="card__container">
  <a id="charizard" class="card orange" href="/charizard.html">
    <img class="card__image" src="/assets/images/stat/stat-charizard.png" alt="Charizard">
    <div class="card__title">
      Charizard
    </div>
  </a>
  <a id="venusaur" class="card green" href="/venusaur.html">
    <img class="card__image" src="/assets/images/stat/stat-venusaur.png" alt="Venusaur">
    <div class="card__title">
      Venusaur
    </div>
  </a>
  <a id="pikachu" class="card yellow" href="/pikachu.html">
    <img class="card__image" src="/assets/images/stat/stat-pikachu.png" alt="Pikachu">
    <div class="card__title">
      Pikachu
    </div>
  </a>
  <a id="gengar" class="card purple" href="/gengar.html">
    <img class="card__image" src="/assets/images/stat/stat-gengar.png" alt="Gengar">
    <div class="card__title">
      Gengar
    </div>
  </a>
  <a id="blissey" class="card pink" href="/blissey.html">
    <img class="card__image" src="/assets/images/stat/stat-blissey.png" alt="Blissey">
    <div class="card__title">
      Blissey
    </div>
  </a>
</div>
```

  - E agora precisamos definir a cor de fundo dos cards de acordo com cada Pokémon:

```
.card.orange {
  background-color: var(--orange);
}

.card.green {
  background-color: var(--green);
}

.card.yellow {
  background-color: var(--yellow);
}

.card.purple {
  background-color: var(--purple);
}

.card.pink {
  background-color: var(--pink);
}
```

- **Criando a estrutura HTML e CSS do rodapé**



  - Jogue Pokémon Unite agora!Logo após da tag `main`, adicione a estrutura HTML do rodapé:

```
<footer class="footer">
  <img src="/assets/images/logo.svg" alt="Pokémon Unite" class="footer__image">
  <a class="footer__button button__link" href="https://unite.pokemon.com/pt-br/">Jogue Pokémon Unite agora!</a>
  <div class="footer__links">
    <a class="footer__link" href="https://play.google.com/store/apps/details?id=jp.pokemon.pokemonunite"
      target="__blank">
      Google Play
    </a>
    <a class="footer__link" href="https://www.nintendo.com/pt_BR/games/detail/pokemon-unite-switch/" target="__blank">
      Nintendo Switch
    </a>
    <a class="footer__link" href="https://apps.apple.com/app/id1512321575" target="__blank">
      App Store
    </a>
  </div>
</footer>
```

  - Como o rodapé ficará disponível em todas as páginas, faremos sua estilização dentro do arquivo `global.css`:

```
.footer {
  position: relative;
  margin-top: 200px;
  background-color: var(--light-background);
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  min-height: 300px;
}
```

```
.footer__image {
  width: 350px;
  position: absolute;
  top: -100px;
}

.footer__links {
  display: inline-block;
  color: white;
  margin: 0 24px;
  text-decoration: none;
}

.footer__link {
  color: white;
  margin: 0 24px;
  text-decoration: none;
}

.footer__link:hover {
  color: var(--primary-color);
}

.footer__button {
  margin: 36px 0;
}

.button__link {
  padding: 16px 24px;
  background-color: var(--primary-color);
  border-radius: 100px;
  text-decoration: none;
  text-transform: uppercase;
  font-weight: bold;
  color: var(--dark-color);
  transition: .3s ease-in-out;
}

.button__link:hover {
  background-color: var(--secondary-color);
  transform: scale(1.1);
}
```

Detalhe Pokémon

- Criar arquivo do detalhe do pokemon, dentro da pasta principal `charizard.html`
- **Criar estrutura inicial do HTML**
  - Digite `html:5` e pressione TAB para criar automaticamente a estrutura base do HTML.
  - Mudar titulo da página e linguagem.

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Pokémon Unite | Charizard</title>
</head>
<body>

</body>
</html>
```

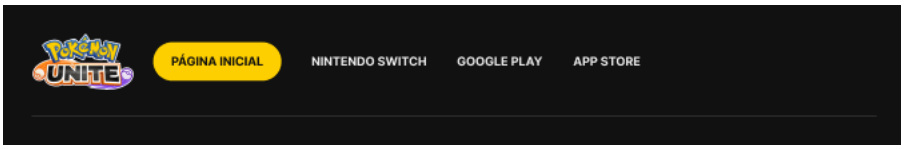
- **Criar arquivo /assets/css/pokemon.css**
  - Importar o arquivo **reset.css** e **global.css**
  - Chamar folha de estilo dentro da página inicial do HTML

```
<link rel="stylesheet" href="/assets/css/pokemon.css">
```

- **Criar style do** `pokemon-color`
  - Adicionar a cor do pokemon no root da página

```
<style>
  :root {
    --pokemon-color: #FFA800
  }
</style>
```

- **Criar estrutura HTML do Menu - IDÊNTICO AO INDEX.HTML**



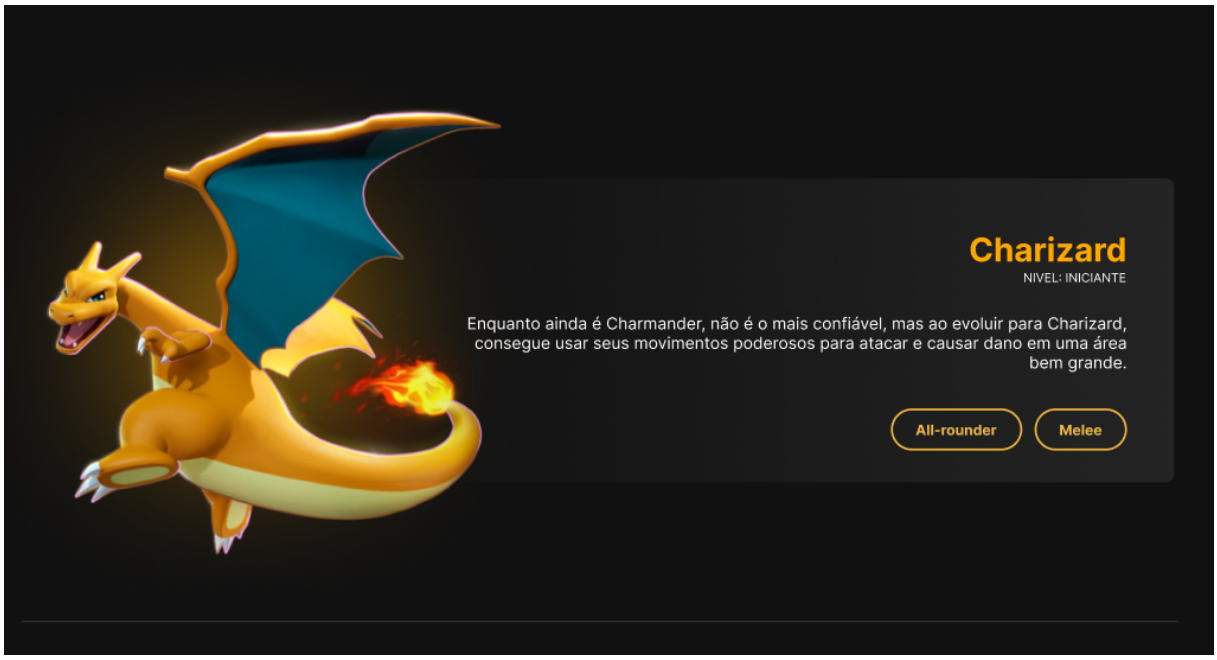
  - Como o menu foi estilizado anteriormente no `index.html` , agora iremos somente adicionar a mesma estrutura HTML que o mesmo estará pronto.
  - Na página `charizard.html` do projeto, adicionar logo após a tag `body` a seguinte estrutura HTML para criar o menu:

```
<nav class="container menu__container">
  <div class="menu__logo">
    <div>
      <img src="/assets/images/logo.svg" alt="Pokémon Unite">
    </div>
  </div>
  <ul class="menu">
    <li class="menu__item menu__item--active">
      <a class="menu__link" href="/">Página Inicial</a>
    </li>
    <li class="menu__item">
      <a class="menu__link" href="https://www.nintendo.com/pt_BR/games/detail/pokemon-unite-switch/" target="__blank">
        Nintendo Switch
      </a>
    </li>
    <li class="menu__item">
      <a class="menu__link" href="https://play.google.com/store/apps/details?id=jp.pokemon.pokemonunite"
        target="__blank">
        Google Play
      </a>
    </li>
    <li class="menu__item">
      <a class="menu__link" href="https://apps.apple.com/app/id1512321575" target="__blank">
        App Store
      </a>
    </li>
  </ul>
</nav>
```

- **Estrutura HTML e CSS**
  - Iremos adicionar o nosso conteúdo, todo dentro de `main`
  - Logo após a tag `nav`, adicione a seguinte estrutura HTML

```
<main class="container">
</main>
```

- **Descrição do pokemon**



- Logo após a tag main, adicione a seguinte estrutura HTML

```
<main class="container">
  <section class="preview__container">
      <div class="preview__pokemon">
        <img class="preview__image" src="/assets/images/stat/stat-charizard.png" alt="Charizardinho"/>
      </div>
      <div class="preview__details">
        <h1 class="pokemon-color">Charizard</h1>
        <p class="preview__level">Nível: iniciante</p>
        <p class="preview__description">
          Enquanto ainda é Charmander, não é o mais confiável, mas ao evoluir para Charizard, consegue usar seus movimentos poderosos para
        </p>
        <div class="preview__pills">
          <span class="preview__pill">All-rounder</span>
          <span class="preview__pill">Melee</span>
        </div>
      </div>
  </section>
...
```

- Agora, em `/assets/css/pokemon.css` adicione os seguintes estilos:

```
.pokemon-color {
  color: var(--pokemon-color);
}

h1 {
  margin-bottom: 0px;
}

h2 {
  font-weight: 700;
  font-size: 32px;
  margin-bottom: 8px;
}

.preview__container {
  display: flex;
  align-items: center;
  position: relative;
  justify-content: space-between;
  border-bottom: 1px solid rgba(255, 255, 255, 0.2);
  padding: 60px 0 100px;
```

```
}

.preview__pokemon {
  width: 35%;
  position: absolute;
}

.preview__pokemon::before {
  content: " ";
  position: absolute;
  width: 180px;
  height: 180px;
  left: calc(75% - 180px);
  top: calc(75% - 180px);
  background: var(--pokemon-color);
  filter: blur(100px);
}

.preview__image {
  width: 100%;
  height: 100%;
  object-fit: contain;
  box-sizing: border-box;
  position: relative;
}

.preview__details {
  background: linear-gradient(270deg, #242424 0%, rgba(36, 36, 36, 0) 95.6%);
  text-align: right;
  padding: 50px;
  border-radius: 8px;
  width: 100%;
  display: flex;
  flex-direction: column;
  align-items: flex-end;
}

.preview__level {
  font-size: 12px;
  text-transform: uppercase;
}

.preview__description {
  margin-top: 28px;
  max-width: 650px;
}

.preview__pills {
  display: flex;
  margin-top: 34px;
  justify-content: flex-end;
}

.preview__pill {
  border: 2px solid var(--pokemon-color);
  border-radius: 39px;
  color: var(--pokemon-color);
  margin-left: 12px;
  padding: 12px 24px;
  font-weight: 700;
  font-size: 14px;
}
```

- **Explicação CSS, conceitos**
  - **pseudo-elemento:**
    - Os pseudo-elementos nos permitem selecionar algumas áreas internas de um elemento HTML e customizá-las através de propriedades.
    - São criados elementos “virtuais”, que não são declarados em nosso HTML, mas que podemos adicionar propriedades CSS a eles.
    - Normalmente são declarados com “::” e o nome do pseudo-elemento, como o ::before (adicionado **antes** do elemento que estamos selecionando através de um seletor CSS) e ::after (adicionado **depois** do seletor que estamos usando em nosso CSS)

- **filter**
  - São propriedades que definem efeitos visuais (como desfoque e saturação, por exemplo) a um determinado elemento

- **Skills pokemon**
  - Logo após a section de `preview`, adicione a seguinte estrutura HTML

```
<section class="preview__container">
...
</section>
<section class="skills__container">
    <h2 class="skills__title pokemon-color" >Skills do Pokémon</h2>
    <p class="skills__description">Esse Pokémon possui as seguintes habilidades de batalha:</p>
    <div class="skills__list">
      <div class="skills__item">
        <label class="skills__label pokemon-color">Ataque</label>
        <progress class="skills__progress pokemon-color" value="50" max="100"></progress>
      </div>
      <div class="skills__item">
        <label class="skills__label pokemon-color">Resistência</label>
        <progress class="skills__progress pokemon-color" value="30" max="100"></progress>
      </div>
      <div class="skills__item">
        <label class="skills__label pokemon-color">Mobilidade</label>
        <progress class="skills__progress pokemon-color" value="40" max="100"></progress>
      </div>
      <div class="skills__item">
        <label class="skills__label pokemon-color">Pontuação</label>
        <progress class="skills__progress pokemon-color" value="90" max="100"></progress>
      </div>
      <div class="skills__item">
        <label class="skills__label pokemon-color">Apoio</label>
        <progress class="skills__progress pokemon-color" value="77" max="100"></progress>
      </div>
    </div>
</section>
```

Agora, em `/assets/css/pokemon.css` adicione os seguintes estilos:

```
.skills__container {
  margin: 60px auto;
  display: flex;
  flex-direction: column;
  align-items: center;
  max-width: 750px;
  justify-content: center;
}

.skills__title {
  margin-bottom: 8px;
}

.skills__description {
  margin-bottom: 40px;
}

.skills__list {
  width: 100%;
}

.skills__item {
  background-color: var(--light-background);
  border-radius: 8px;
  margin: 12px 0;
  padding: 24px;

  display: flex;
  justify-content: space-between;
  align-items: center;
}
```

```
.skills__label {
  font-size: 16px;
}

.skills__progress {
  width: 50%;
  height: 10px;
  box-shadow: 1px 1px 4px rgba(0, 0, 0, 0.2);
}

.skills__progress::-webkit-progress-bar {
  background: rgba(155, 155, 155, 0.21);
  border-radius: 10px;
}

.skills__progress::-webkit-progress-value {
  background-color: var(--pokemon-color);
  border-radius: 10px;
}
```

- **Evoluções**
  - Logo após a section de `preview`, adicione a seguinte estrutura HTML

```
<section class="preview__container">
...
</section>
<section class="evolutions__container">
  <h2 class="pokemon-color">Evoluções</h2>
  <p class="">Esse Pokémon pode evoluir em:</p>
  <div class="evolutions__list">
    <div class="evolutions__item">
      <img src="/assets/images/evolution/evolution-1-charmander.png" alt="Charmander" class="evolutions__image">
      <div class="evolutions__details">
        <p class="evolutions__name pokemon-color">Charmander</p>
        <p class="">Nível 1</p>
      </div>
    </div>
    <div class="evolutions__item">
      <img src="/assets/images/evolution/evolution-2-charmeleon.png" alt="Charmeleon" class="evolutions__image">
      <div class="evolutions__details">
        <p class="evolutions__name pokemon-color">Charmeleon</p>
        <p class="">Nível 4</p>
      </div>
    </div>
    <div class="evolutions__item">
      <img src="/assets/images/evolution/evolution-3-charizard.png" alt="Charizard" class="evolutions__image">
      <div class="evolutions__details">
        <p class="evolutions__name pokemon-color">Charizard</p>
        <p class="">Nível 9</p>
      </div>
    </div>
  </div>
</section>
```

Agora, em `/assets/css/pokemon.css` adicione os seguintes estilos:

```
.evolutions__container {
  display: flex;
  flex-direction: column;
  align-items: center;
}

.evolutions__list {
  width: 100%;
  display: flex;
  justify-content: space-evenly;
  margin-top: 32px;
  gap: 24px;
}

.evolutions__item {
  display: flex;
  flex-direction: column;
  align-items: center;
```

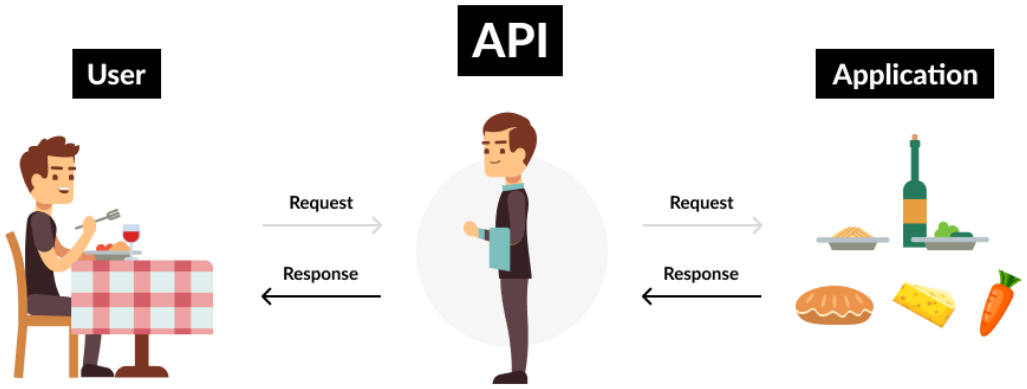
```
}

.evolutions__image {
  width: 100%;
  height: 100%;
  padding: 24px;
}

.evolutions__details {
  font-size: 12px;
  font-weight: bold;
  text-align: center;
  background-color: var(--light-background);
  width: 100%;
  border-radius: 8px;
  padding: 8px 0;
}
```

- **Footer**
  - Como o footer foi estilizado anteriormente no `index.html` , agora iremos somente adicionar a mesma estrutura HTML que o mesmo estará pronto.
  - Logo após da tag `main`, adicione a estrutura HTML do rodapé:

```
<footer class="footer">
  <img src="/assets/images/logo.svg" alt="Pokémon Unite" class="footer__image">
  <a class="footer__button button__link" href="https://unite.pokemon.com/pt-br/">Jogue Pokémon Unite agora!</a>
  <div class="footer__links">
    <a class="footer__link" href="https://play.google.com/store/apps/details?id=jp.pokemon.pokemonunite"
      target="__blank">
      Google Play
    </a>
    <a class="footer__link" href="https://www.nintendo.com/pt_BR/games/detail/pokemon-unite-switch/" target="__blank">
      Nintendo Switch
    </a>
    <a class="footer__link" href="https://apps.apple.com/app/id1512321575" target="__blank">
      App Store
    </a>
  </div>
</footer>
```

++ JS

- **Carregamento dos pokemons, a partir de uma API**
  - **O que é API?**
    - Conceito::
      - Conjunto de definições e protocolos para criar e integrar softwares de aplicações.
    - Em outras palavras::
      - nada mais é do que uma forma de comunicação entre sistemas (elas permitem a integração entre dois sistemas, em que um deles fornece informações e serviços que podem ser utilizados pelo outro, sem a necessidade de o sistema que consome a API conhecer detalhes da implementação do software.
    - Exemplo::
      - Podemos imaginar que uma API funciona como um garçom. Quando buscamos o que desejamos no menu e solicitamos ao garçom, ele encaminha esse pedido para a cozinha, que prepara o pedido. E, no fim, o garço nos traz esse prato pronto



- Crie uma nova pasta `/assets/js/index.js`
- Adicione a tag `script` no seu `head` index.html

```
<link rel="stylesheet" href="/assets/css/index.css">
<script src="/assets/js/index.js"></script>
```

- Adicione o seguinte código ao seu `index.js`

```
listPokemons();

function listPokemons(){
  //https://6283929f92a6a5e462260498.mockapi.io/pokemon
  fetch('https://bit.ly/3MJecd5')
  .then(response => response.json())
  .then(data => {
    const pokemons = data.map((item) => getContentCard(item));

    document.querySelector(".card__container").innerHTML = pokemons.join('');
  });
}

function getContentCard(data){
    return `
    <a id="${data.id}" class="card" href="/${data.id}.html" style="background-color:${data.color || 'red'}">
      <img class="card__image" src="/assets/images/stat/stat-${data.avatar}.png" alt="${data.name}">
      <div class="card__title"> ${data.name}</div>
    </a>
    `;
}
```

- Remova o trecho de código do `card__container` em `index.html`
- **Código síncrono x assíncrono?**
  - Síncrono ou assíncrono diz respeito ao fluxo de execução de um programa
  - Em um código **síncrono**, todas as funções e requisições trabalham em sincronia, sequencialmente. (começo, meio e fim)
  - **Assíncrono** significa que as coisas podem acontecer independemente do fluxo principal do programa.
  - Analogia::

- Quando realizamos uma chamada telefônica, poderíamos considerar como um processo síncrono, visto que ambos precisam estar em sincronia
- Se mudarmos para uma vídeo do youtube, ai consideramos como um processo assíncrono, já que o mesmo é gravado e pode ou não ser assistido posteriormente

```
console.log('Mensagem 1');
console.log('Mensagem 2');

setTimeout(() => console.log('Mensagem 3'), 200)
console.log('Mensagem 4')
```

```
"Mensagem 1"
"Mensagem 2"
"Mensagem 4"
"Mensagem 3"
```

- **O que é Promise?**
  - Conceito::
    - Promise é um objeto usado para processamento assíncrono. Um Promise (de "promessa") representa um valor que pode estar disponível agora, no futuro ou nunca.
  - Exemplo::
    - Podemos imaginar que uma API funciona como um garçom. Quando buscamos o que desejamos no menu e solicitamos ao garçom, ele encaminha esse pedido para a cozinha, que prepara o pedido. E, no fim, o garço nos traz esse prato pronto

++ Mobile

- **Corrigindo o menu**
  - No seu `global.css` adicione o seguinte código

```
.menu__toggle,
.icon {
  display: none;
}
@media only screen and (max-width: 600px) {
  .icon {
    display: inline-block;
  }

  .menu__container{
    flex-direction: column;
    justify-content: space-between;
  }

  .menu__logo{
    width: 90%;
  }

  .menu {
    display: none;

    width: 100%;
    flex-direction: column;
    align-items: center;
    margin-top: 24px
  }
```

```
  .menu, .menu__link {
    width: 100%;
  }

  .menu__toggle:checked ~ .menu {
    display: flex;

  }
}
```

- No `index.html` e no `charizard.html` adicione o seguinte código

```
...
<nav class="container menu__container">
  <input type="checkbox" id="menu__toggle" class="menu__toggle">
  <div class="menu__logo">
    ...
    ...
    <label for="menu__toggle" class="icon">
      <img src="/assets/images/hamburguer.svg" class="">
    </label>
  </div>
```

- **Corrigindo a listagem**
  - No `index.css` adicione

```
@media only screen and (max-width: 600px) {
  .card__container {
    justify-content: center;
  }

  .search__button--mobile{
    display: initial;
    padding: 8px 12px;
  }

  .search__button{
    display: none;
  }

  .search__field{
    margin-right: 12px;
  }
}
```

- **Corrigindo o footer**
  - No arquivo `global.css` juntamente com o media do menu, adicione o código

```
@media only screen and (max-width: 600px) {
...
...

.footer__image {
  width: 242px;
  top: -70px;
}

.footer__links {
  display: flex;
  flex-direction: column;
  align-items: center;
}

.footer__link {
  margin: 8px 24px;
}
```

- **Corrigindo o detalhe do pokemon**
  - No arquivo `pokemon.css` adicione o trecho

```
@media only screen and (max-width: 600px) {
  .preview__container {
    flex-direction: column;
  }

  .preview__pokemon {
    position: relative;
    margin-bottom: 36px;
    width: 75%;
  }

  .preview__details {
    width: initial;
  }

  .preview__pills {
    flex-wrap: wrap;
  }

  .preview__pill {
    margin-bottom: 8px;
  }

  .skills__description {
    text-align: center;
  }
}
```