Imersão Dev - Aula 09

Rafaella: Olá, **Devs**! Estamos quase no final da nossa **Imersão Dev**. Meu coração até aperta de pensar nisso. Definitivamente, essas duas semanas têm sido incríveis e ainda está para ficar mais incrível ainda.

Hoje veremos muita coisa legal. Não somente com JavaScript, mas também com HTML e CSS, três tecnologias principais para o desenvolvimento front-end na web. Se você tiver qualquer dúvida, nos mencione no Discord. Não tenha medo de chamar, porque nós e todo mundo do Scuba Team, além de outros alunos que estão ali e já fizeram o projeto, poderão te ajudar.

Pegue o seu café, abra o **CodePen** e vamos para a nona aula.

Estamos chegando às nossas duas últimas aulas. Agora veremos a aula 9 e 10. Aprenderemos um pouco mais sobre HTML e CSS, que foram duas tecnologias que deixamos um pouco mais de lado até agora durante a **Imersão Dev**. Mas vamos ver como juntar JavaScript, que aprendemos até agora, com essas partes de elementos e de estilização.

O projeto de hoje vai ser um portfólio com todos os projetos que fizemos ao longo da Imersão Dev, que foram vários. Se não me engano, foram sete, contando com o Super Trunfo que demorou dois dias. Temos muita coisa legal para mostrar para outras pessoas. Esse projeto de portfólio vai ser o mais legal de você compartilhar nas redes sociais e com todas as pessoas que você conhece.

Faremos duas versões de portfólio, uma versão um pouco mais descritiva, com os títulos dos projetos e na próxima aula faremos algo que vai mostrar o nosso código mesmo. Todos os projetos, várias coisas vão aparecer e vai ser muito legal. Tenho certeza que vocês vão gostar muito de compartilhar isso para todo mundo poder ver.

Agora estamos com o **Figma** na tela. Figma é a plataforma que a galera de design utiliza para desenhar como será a página que queremos desenvolver. Depois, o pessoal de front-end pega esse design já pronto, da equipe de designers, e começa a investigar tudo que tem na página que teremos que transformar em código.

Eles fazem a imagem e nós temos que transformar cada elemento em código HTML, CSS e JavaScript. É exatamente isso que vamos fazer. Já temos aqui, no Figma, a página que queremos desenvolver e vocês terão acesso a exatamente esse link do Figma. O link vai ficar aqui embaixo, na descrição, para vocês poderem pegar e ter acesso a tudo.

Se não me engano, é preciso fazer uma conta no Figma para poder utilizar, mas é bem tranquilo e é tudo de graça. É só vocês aprenderem a mexer e entrar com um usuário. E

ao longo dessa aula vamos transformar cada um desses elementos em código. Vamos com calma para vocês entenderem tudo, vai ser tranquilo.

Paulo: Então é muito comum esse tal de Figma, ou até outros, onde um time de pessoas, ou uma pessoa que faz parte da sua equipe, vai lá e desenha a interface, a experiência que o usuário vai ter no nosso site, na nossa app, no nosso sistema. E as pessoas que trabalham com front-end, que têm esse cargo de transformar o que foi desenhado no papel, ou num protótipo, e implementar para transformar aquilo em algo tangível com o qual o usuário interage.

Essa ideia que a Rafaella trouxe é justamente para mostrar a você, que está estudando na Imersão Dev só há 10 dias, o funcionamento de uma empresa. Porque no funcionamento de uma empresa acontece exatamente essa proposta que a Rafaella trouxe.

Chega para esse time de pessoas que trabalham com JavaScript, HTML e CSS, chega de uma outra pessoa do seu time, de pessoas que trabalham com interface do usuário, com experiência do usuário, o UX (*User Experience*), normalmente elas usam essas ferramentas como Figma, Adobe XD, Sketch, entre outras, que servem para prototipar um site, uma app ou um sistema, por exemplo.

Alguém responsável por isso entrega o projeto na mão do Guilherme e da Rafa e fala: "Se vira aí". Então, entra de um lado um protótipo, um rabisco tipo um desenho de arquiteto, explicando mais ou menos como deve ser a experiência do usuário. E você coloca isso num site, por exemplo, deixa bonito e chama as coisas que precisam acontecer no servidor, no sistema.

No dia a dia da Alura temos usado essa abordagem, nós temos os **Alura Challenges**, onde tentamos trazer isso. Trazemos, inclusive, o Trello, que é onde colocamos o projeto e indicamos o que precisa ser feito.

Às vezes você pode falar "Paulo, não podia ficar só em JavaScript?". Podia, mas queremos que você enxergue como é na vida real mesmo, no ambiente de trabalho. Claro que aqui é só o começo. Vocês vão ter tempo de estudo, se aprofundar quando estiverem trabalhando e vendo isso acontecer.

Fica aqui meus parabéns à Rafa e ao Guilherme, que tiveram essa ideia de trazer mais um ponto que é da nossa realidade.

Guilherme: Isso aí. Agora, vamos começar a desenvolver o nosso projeto. Aos poucos vamos pegar as informações do Figma e passar para o nosso projeto. Nós não temos um modelo, um *template*, desse projeto. Nós vamos começar do zero. Então, no CodePen, vou clicar em "pen" para criar um novo pen. E intitular esse projeto como "Aula 9 - Portfólio".

Vou clicar no "Settings > Behavior" e deixar "não" nas opções de salvamento automático e de auto-update, e deixar "sim" para a formatação automática. E no "Pen Details", vou escrever no campo de tags: "imersaodev, alura" e, em seguida, vou clicar no botão "Salvar".

O que geralmente temos em uma página HTML? Temos aquela parte de cima, o cabeçalho. Podemos ver, por exemplo, na página inicial do site da Alura, na parte de cima onde tem o logo da Alura, a barra de busca e um menu horizontal, essa parte é um trecho do HTML, que é específico, sempre que eu quero alterar alguma coisa aqui da "cabeça" da página, tem essa estrutura.

Depois tem o corpo do HTML, que são os ícones, as imagens, tudo que aparece aqui no site. Nesse projeto temos algo semelhante. Abrindo lá no Figma, na parte de cima temos uma foto da Rafa, o nome dela e o texto "Instrutora e desenvolvedora front-end".

Nós podíamos separar uma parte do HTML só para isso.

Rafaella: Como estamos utilizando o CodePen, tem algumas coisas que não precisaremos escrever, coisas que ao usar outros editores precisamos escrever. Porque o CodePen já entende algumas coisas automaticamente, por exemplo, o HTML tem duas grandes divisões no código, a parte de *head*, que é a cabeça do HTML e a *body*, que é o corpo do HTML.

Essa parte *head*, do HTML, é necessária quando você está escrevendo código em outros editores, por exemplo, o Visual Code, entre outros nos quais precisamos escrever o HTML do zero. E o *head* geralmente contém os metadados da página, como por exemplo, título e a linguagem que utilizaremos. Porém não é preciso escrever aqui no CodePen por ele já ter isso como padrão e não é necessário mesmo escrever essa parte no CodePen. Podemos apenas escrever o *body*.

Então, o *head*, que é a parte de metadados e informação da página, nós não faremos, vamos focar no *body* que é onde escrevemos todos os elementos que queremos colocar na página. É a parte realmente estrutural, é o corpo desde lá de cima, como o Gui mostrou o cabeçalho da página, até o *footer*, o rodapé da página.

Então a primeira tag que mostraremos é a *body*, que é onde vamos englobar todos os elementos da página. E o Gui vai mostrar como dividiremos a página.

Guilherme: Isso aí, Rafa. Em outras palavras, quando usamos outro editor de código precisamos falar: "HTML, existe um CSS e o link do CSS é esse. HTML, tem um JavaScript e o link para os scripts é esse aqui, tem que passar esses endereços". Usando o CodePen isso já acontece automaticamente.

Para começar o nosso código, vou digitar o `<body>`, todo o conteúdo da nossa página estará dentro dele. Repare na sintaxe que estou usando, sempre colocando o sinal de

menor e depois o sinal de maior. Para fechar a tag `<body>`, fazemos a mesma coisa mas com uma barra, `</body>`.

Nossa página será dividida em duas estruturas. A primeira é o `<header>`, que no nosso projeto vai ser esse espaço que tem a foto da Rafa, o nome dela e o texto descrevendo o trabalho dela. Além disso, outra tag importante é a `<main>`, que vai ter o conteúdo com os links dos projetos de cada aula.

```
<body>
  <header>

  </header>
    <main>

   </main>
</body>
```

Guilherme: Vamos verificar, no Figma, os elementos que temos no cabeçalho. Temos uma imagem, um texto com uma fonte maior e a descrição em fonte menor. Vou colocar essas propriedades dentro do nosso `<header>`. Primeiro colocaremos um `<img>`, que representa a imagem, note que antes de fechar a tag aparece algumas coisas aqui no autocompletar, que podemos usar como propriedades para essa imagem que queremos colocar. Esse `src` significa "*source*" e indica qual é o caminho da imagem que queremos utilizar.

No projeto do Figma tem uma foto da Rafa. Rafa, onde conseguimos encontrar essa imagem?

Rafaella: Essa imagem está no meu perfil de uma plataforma chamada **GitHub**, vocês já devem ter ouvido falar sobre ela em algum momento durante seus estudos de programação. Se não ouviram, ainda falaremos sobre ela e tem um conteúdo superlegal sobre o GitHub que vamos deixar na descrição do vídeo.

GitHub é uma plataforma em que você consegue armazenar seus códigos, nela você consegue subir seus códigos e fazer diferentes versões dele. É uma plataforma que usaremos ao longo do tempo e é essencial para quem quer trabalhar na área de desenvolvimento.

Estamos na minha conta do GitHub. Uma coisa que eles fizeram e é muito legal é que basta colocar um `.png` no final da URL do meu perfil e já conseguimos acessar a minha foto.

Estamos utilizando o GitHub agora apenas para pegar o caminho da minha foto, a foto de vocês se já tiverem uma conta no GitHub. Recomendo que vocês já criem uma conta

lá para achar tranquilamente a sua foto simplesmente colocando `.png` no final da URL do seu perfil.

No JavaScript conseguimos, por exemplo, criar variáveis diferentes e podemos trocar o usuário dinamicamente e ir pegando foto de todos os usuários que quisermos. Coloca uma variável ali na barra (`/`) e vamos colocando as fotos que queremos puxar.

Não vamos fazer isso agora, mas é só para vocês entenderem algo legal que podemos fazer utilizando uma imagem que já está nessa plataforma.

Nós utilizaremos essa URL para a minha foto, não precisa salvar a imagem nem nada, pegamos direto da web e vamos voltar para o nosso código.

Guilherme: Lembrando, por questão de segurança, que quando colocamos o `.png` e pressionamos "Enter", a URL muda. Não é essa URL que pegaremos. Nós vamos pegar a URL do perfil mesmo, adicionando o `.png` no final. No caso do perfil da Rafaella, essa é a URL que pegaremos: `https://github.com/rafaballerini.png`. Vou colar a URL no nosso código e fechar a tag de imagem com barra e o sinal de maior (`/>`).

```
<body>
  <header>
        <img class="perfil-foto"
src="https://github.com/rafaballerini.png" />

  </header>

   <main>

  </main>
</body>
```

Paulo: Vamos rodar para ver como fica?

Guilherme: Sim. Vou salvar e clicar em "Run" para executar. Apareceu a foto da Rafa.

Rafaella: Para mudar o tamanho da imagem já faz parte da estilização que veremos daqui a pouco como fazer.

Guilherme: Posso criar uma nova divisão no HTML para colocar os textos que aparecem ao lado da sua foto no Figma.

Rafaella: Pode.

Guilherme: Então eu vou colocar uma `<div>`, porque eu quero dividir o nosso *header* para ficar fácil de conseguirmos manter. Dentro dessa `<div>` vou colocar aquele texto em uma tag `<h1>`.

Rafaella: Usamos muito o `<h1>` para título da página, para o que vai ser o texto principal da nossa página. Nesse caso, o título é o nosso próprio nome no portfólio.

Guilherme: Vou colocar aqui "Rafaella Ballerini".

Rafaella: Para o texto que aparece embaixo você pode usar um `<h2>` ou `<h3>`. O `<h1>` normalmente utilizamos apenas uma vez na página.

Guilherme: Vou colocar com `<h3>` o trecho "Instrutora e desenvolvedora front-end".

```
<body>
  <header>
        <img src="https://github.com/rafaballerini.png" />
            <div>
            <h1>Rafaella Ballerini</h1>
            <h3>Instrutora e desenvolvedora front-end #imersaodev</h3>
            </div>
   </header>
//código omitido
```

Guilherme: Vou salvar e executar, só para ver como fica.

Rafaella: Lembrando que é para você, aí de casa, criar o seu portfólio, ou seja, colocar seu nome e sua própria descrição.

Paulo: Mostra aí, Guilherme, que se trocarmos, na URL da imagem, o nome da Rafaella pelo meu nome de usuário do GitHub é a minha foto que aparece, `<img src="https://github.com/peas.png" />`. Ao salvar e rodar vai mudar a imagem.

Vocês podem colocar qualquer imagem nesse img src, podem fazer o mesmo procedimento que aprendemos para pegar as imagens na aula de Super Trunfo em que pegamos a imagem do Shiryu de Dragão. Mas como na próxima aula vamos usar mais o GitHub, já é legal você criar uma conta e ir se familiarizando.

Como esse aqui vai ser o site do seu portfólio, queremos que você demonstre que sabe usar o GitHub, que tem o CodePen, que você fez além do que tinha na Imersão Dev. Inclusive, Guilherme, aí na tag `<h3>`, coloca a hashtag #imersaodev. Uma pessoa reconhece a outra e fala: "Poxa, ela também começou na Imersão Dev?". É esse efeito que queremos criar.

Guilherme: Beleza. Nós colocamos a imagem, o `<h1>` e `<h3>`, mas ainda não está muito bonito. Nós sabemos que podemos usar o CSS para deixar isso mais bonito. Eu tenho uma dúvida que é comum. Em que momento fazemos essa parte do CSS? Primeiro criamos todo o HTML do site e depois arruma e deixa ele bem bonito ou fazemos uma parte e, em seguida, já faz o CSS?

Rafaella: Acho que isso vai mais do gosto da pessoa. Talvez quando você já tem um pouco mais de conhecimento de front-end é mais fácil fazer o HTML inteiro e depois ir estilizando. Mas no começo é mais recomendado ir estilizando cada parte logo após fazer o HTML. Para entender tudo que está acontecendo.

Talvez o ideal seja fazer o CSS dessa parte agora mesmo. Porque depois que fazemos todo o HTML fica bem “feio”, não fica muito legal. Fica muito simples. E também é difícil de se achar no código.

Guilherme: Pensando a partir do conhecimento que adquirimos na Imersão Dev, no JavaScript, sempre que precisávamos pegar determinado trecho do HTML usávamos um `document.GetElementById()`, entre outras formas, para falar que precisamos de determinado espaço do HTML. No CSS eu imagino que tenha algo parecido.

Rafaella: Exato. No CSS, normalmente, utilizamos as próprias tags. Existem três tipos de seletores principais que podemos utilizar para estilizar o HTML, por exemplo, se utilizamos a tag `<div>` vai ser aplicado para todas as `div`s do nosso projeto; a classe vai ser para os elementos que tiverem a classe que escreveremos, pode ser uma classe "perfil", uma classe "título", e os elementos que que colocarmos essa classe terão isso aplicado; e o id sempre utilizamos para estilizar apenas um elemento específico. Quando queremos apenas aquele elemento, daquela forma, colocamos um id nele que vai ser único e não será aplicado em nenhum outro elemento.

Nesse caso, vamos utilizar muito mais o `class` e as tags para estilizar nossa página. Então, vamos começar a estilizar?

Primeiro começamos estilizando as coisas maiores, começando pelo fundo e depois vamos para os elementos. Vamos começar pela tag `<body>`, que é essa que vai pegar todos os elementos. No CSS você escreve:

```
body {

}
```

Rafaella: E vamos escrever todas as coisas que queremos estilizar, por exemplo, o fundo, a cor da fonte, etc. Vamos lá para o Figma pensar qual é o mais externo de todos os elementos da nossa página. A primeira coisa que podemos pegar é essa cor azul-escura de fundo, que vai ser o nosso *background*. E como conseguimos descobrir

qual é a cor exata? Aliás, ela tem também um efeito gradiente. Para descobrir qual é a cor utilizaremos o próprio Figma.

Podemos clicar em cima da cor e lá na lateral direita tem uma aba vertical chamada "Inspect", de inspecionar, nela conseguimos ver alguma informação como a "colors", as cores, que nos informa que é um "Linear Gradient" (Gradiente Linear) e tem duas cores diferentes. Ele já nos informa quais cores estamos utilizando. Mas além disso, conseguimos ter o código CSS desse fundo. Nessa mesma aba, abaixo de "Colors", tem o "Code CSS" que passa a posição, largura, altura, e o nosso `background` que é exatamente o que queremos.

Preciso fazer uma observação: às vezes não é legal usar o posicionamento informado pelo Figma porque ele pode ter um posicionamento dependente de elementos de uma forma diferente. É importante termos controle do posicionamento ao estilizarmos o CSS, então não recomendo utilizar essa parte de posicionamento, altura e largura do Figma. Até porque no CodePen tem um tamanho diferente.

Mas faremos isso aos poucos, vou mostrar como recomendamos que isso seja feito. O que vamos pegar agora é apenas o trecho do código CSS do `background`. Podemos copiar esse trecho, voltar para o CodePen e colar dentro do `body` do CSS.

```
body {
  background: linear-gradient(236.85deg, #041832 27.26%, #3468A7
96.03%);
}
```

Rafaella: Vamos salvar e rodar para ver o que vai acontecer. Agora nosso fundo está com a cor certa e com o gradiente linear que vimos no design do Figma.

Paulo: Quem quiser pode mudar a cor também. Mexer nesses números do código de cor e testar.

Rafaella: Lógico. Esses números, na verdade, são o cálculo do ângulo do gradiente, que pode ser na vertical ou horizontal, pode ser a porcentagem das cores também, às vezes tem mais azul-escuro de um lado. Mas vocês podem estilizar como quiserem, estamos deixando aberto para vocês personalizarem.

Inclusive, se quiserem podem colocar uma imagem no fundo, algum *wallpaper* legal, por exemplo. Em relação às cores, vocês já sabem que utilizamos aquele código hexadecimal, que tem uma cerquilha (#) seguida por seis números. Vocês podem brincar com essas cores, basta pesquisar o código das cores que preferir. Tem vários lugares onde é possível brincar com isso, inclusive no próprio Figma.

Paulo: Essas cores sólidas, até se escrevermos `red` ou `pink` também funciona? Só as mais óbvias. Vamos rodar mas pode ser que dê errado, mas é justamente esse tipo de coisa que queremos que teste para saber se funciona mesmo, ainda que fique "feio".

```
body {
      background: linear-gradient(236.85deg, #041832 27.26%, red
96.03%);
}
```

Paulo: Então conseguimos colocar algumas cores além da forma em que escrevemos com o `#` seguido do valor.

Na sintaxe de `body`, não precisamos memorizar se é `linear-gradient()`, se é positivo ou relativo, pois no começo apenas vamos copiar as *tags* e explorar bastante.

Quem conhece e já trabalhou com Photoshop ou qualquer uma dessas ferramentas de edição de imagem, verá que parece muito com o menu que preenchemos para um gradiente de forma visual, com as barras por exemplo.

Agora, com apenas o código e sem uma caixa para preencher esses dados, fizemos a configuração de cor escrevendo apenas uma linha de linguagem declarativa, ou seja, declaramos que, para o corpo da página *web*, queremos um gradiente linear de uma cor a outra no *background* com uma certa opacidade.

Guilherme: Vamos copiar o valor da cor do nosso projeto no lugar de `red` para continuarmos.

```
body {
      background: linear-gradient(236.85deg, #041832 27.26%, #3468A7
96.03%);
}
```

Rafaella: É por isso que estamos usando o **Figma**, afinal é muito mais tranquilo já vermos de fato o código das coisas ao invés de termos que escolher e buscar uma cor.

Então, a segunda coisa é que precisamos também alterar a fonte que temos agora no CodePen, pois é bem diferente da fonte utilizada no nosso design.

Vamos voltar ao Figma e buscar a fonte certa. É a mesma para a página inteira, desde o nome até os nomes dos projetos. Na aba "Inspect" ao lado, temos a parte de tipografia chamada "Typography", e encontraremos a fonte "Roboto".

Para podermos trazer fontes para o nosso projeto, podemos utilizar uma plataforma chamada **Google Fonts**, basta procurar no buscador. Acessando-a, veremos que existem muitas opções e podemos escrever texto para testar se gostamos, e escolher.

Fiquem à vontade para explorar outras fontes, mas por enquanto usaremos a Roboto que é bem conhecida. Encontrando-a na plataforma, clicaremos sobre esta e selecionaremos os pesos ou estilos que queremos, o que também já está no nosso Figma.

O nome está com o número "700" no campo "weight", e o texto de descrição da nossa página está com peso "400", então vamos usar esses dois valores. Quando acessamos os pesos da fonte Roboto no Google Fonts, teremos uma lista com textos de exemplos e seus respectivos valores de peso e estilo no canto superior esquerdo de cada item.

Escolheremos o "Regular 400" e clicaremos no botão de "select this style" nesta opção. Depois, encontraremos o "Bold 700" e selecionaremos este estilo também.

Existe a possibilidade de importarmos a fonte em nosso HTML ou no CSS, pois o Google Fonts oferece a opção de "link" e "import" na opção "Use on the web" na aba "Selected family".

Usaremos no CSS, já que estamos aprendendo como estilizar. Selecionaremos a opção "import", copiaremos o código a partir da tag `<style>` até o final e depois colocaremos no início do nosso código CSS.

É como se estivéssemos de fato importando algo para usarmos ao longo do nosso código, então é importante colocarmos no topo do projeto para usarmos quando quisermos.

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

body {
      background: linear-gradient(236.85deg, #041832 27.26%, #3468A7
96.03%);
}
```

Rafaella: De volta ao Google Fonts, também encontraremos a informação de como utilizá-la, pois apenas a importamos para o nosso projeto, mas não a estamos usando ainda.

Para isso, iremos na aba lateral de "Selected family" e encontraremos o campo de "CSS rules to specify families" com `font-family: 'Roboto', sans-serif;`, que é a forma como falamos ao CSS que queremos utilizar esta fonte.

Então copiaremos essa linha e a colocaremos dentro do `body`, porque de fato vamos usar esta fonte para todos os elementos da página.

Vamos salvar e rodar para ver se a fonte está mudando.

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

body {
      background: linear-gradient(236.85deg, #041832 27.26%, #3468A7
96.03%);
      font-family: 'Roboto', sans-serif;
}
```

Rafaella: Assim já temos nosso fundo e nossa fonte correta.

Agora, precisaremos estilizar o nosso cabeçalho, que é de fato a parte que o Guilherme já fez contendo a imagem junto com a `<div>` que possui o nome e a descrição.

Para isso, podemos primeiro criar uma nova classe para esse conjunto de coisas - imagem, título e subtítulo -, porque sempre estilizaremos do maior para o menor.

Por exemplo, poderíamos puxar a tag `<header>` que é a única que estamos utilizando, mas começaremos usar de fato as classes.

Primeiramente, dentro do `<header>` do HTML, criaremos uma `class` para este conjunto do cabeçalho. Após o sinal de igualdade e entre aspas, escreveremos `container`.

O `"container"` é um agregado de elementos que temos, então essa classe representará o nosso cabeçalho.

Para chamarmos uma tag no CSS, só precisaremos escrevê-la, mas para chamarmos uma `class`, colocamos um ponto antes do nome, obtendo `.container` neste caso. Isso para quando quisermos estilizar uma classe.

Paulo: Se tivéssemos escrito somente `header`, só usaríamos essas características para o `<header>`, mas como fizemos usando a classe, todos os elementos que tiverem a `.container`, e talvez não seja somente o `<header>` mesmo, também pegarão essas características.

Rafaella: Exatamente, é por isso que as utilizamos mais, principalmente quando queremos reutilizar algum código e temos mais de um `container` por exemplo. Neste caso usaremos apenas este mesmo para entendermos bem o funcionamento das classes e tags.

Agora já temos o nosso `body` todo estilizado, ainda que não possamos ver muitas coisas na página ainda. Mas já temos estilizadas as coisas que usaremos nela inteira.

Vamos voltar ao Figma para sabermos o próximo passo. temos um fundo azul e um retângulo de fundo branco, o qual contém o cabeçalho do `<header>` e o `<main>`, que é o conteúdo principal.

Então precisaremos criar algo que englobe esses dois conjuntos de elementos. De volta ao nosso código, veremos o que podemos fazer para colocar este retângulo branco.

Temos o `<body>` com o `<header>` e o `<main>` separados no HTML, e precisamos englobá-lo em alguma divisão. Criaremos uma nova `<div>` que engloba desde antes do `<header>` até após o `</main>`.

Assim, poderemos estilizar essa `<div>` para termos o fundo branco do retângulo, pois se estilizarmos o `<body>`, será a parte mais externa do fundo azul. Portanto precisamos de uma divisão interna.

Em seguida, precisaremos criar uma nova classe para esta nova `<div>`. Podemos ter as tags para elementos no geral, mas as classes podem ser definidas para quando quisermos reutilizar a estilização ou definir estilizações diferentes, afinal estamos usando outras `<div>`.

Se estilizarmos a `<div>` da segunda linha do HTML, as linhas de `<h1>` e `<h3>` da `<div>` seguinte receberão essa mesma estilização, e não é o que queremos.

Portanto, criaremos uma nova `class` que terá o nome igual a `"container"` para o conjunto. Para estilizarmos uma classe, faremos de forma um pouco diferente em relação às tags.

```
<body>
      <div class="container">
            <header>
                  <img src;"https://github.com/rafaballerini.png" />
                  <div>
                        <h1>Rafaella Ballerini</h1>
                        <h3>Instrutora e desenvolvedora front-end
#imersaodev</h3>
                  </div>
            </header>

            <main>

            </main>
      </div>
</body>
```

Rafaella: Quando vamos estilizar tag, apenas colocamos seu nome, abrimos chaves e estilizamos.

Paulo: Se colocássemos uma `div` e depois abríssemos chaves, funcionaria também, porém todas as outras divisões mudariam o estilo para este mesmo, e não é isso que precisamos, pois queremos estilizar somente as `div` que "carimbamos" com as características do *container*, com especial, título e outra classe que faça sentido.

Rafaella: Para usarmos uma classe, precisamos pôr um ponto antes do nome da `class` que queremos. Portanto, usaremos `.container` mesmo e abriremos as chaves para podermos começar a estilizar.

Começaremos colocando a cor de fundo branca. Então vamos verificar qual é o valor desta cor para nosso background no "Rectangle 1" da aba "Inspect".

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

body {
      background: linear-gradient(236.85deg, #041832 27.26%, #3468A7
96.03%);
      font-family: 'Roboto', sans-serif;
}

.container {

}
```

Paulo: Há várias cores brancas diferentes, por incrível que pareça. Poderíamos escrever `background` com dois pontos e `white` apenas, mas a equipe de design que fez este Figma já falou que este branco não é puro, é mais um cinza super claro, então temos que buscar a informação lá no Figma.

Rafaella: Então pegaremos a cor que já está no nosso "Inspect" e colaremos no nosso código. Ela não tem um gradiente como a do fundo, então é mais tranquilo.

Podemos tanto copiar do código CSS que já vem com a propriedade do `background`, quanto escrever a propriedade e colocar a cor.

Guilherme: Iremos trazer só a cor ou outros elementos também?

Rafaella: Faremos apenas o *background* por enquanto.

Guilherme: Certo, então vamos escrever fazendo do modo *hard-coded* mesmo. Para mudarmos a cor, colocaremos `background:` em `.container`, seguido do valor `#ECF4FF` da cor de fundo que pegamos do Figma.

Rafaella: Em seguida, vamos salvar e rodar para vermos como nossa página ficou.

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

body {
      background: linear-gradient(236.85deg, #041832 27.26%, #3468A7
96.03%);
      font-family: 'Roboto', sans-serif;
}

.container {
      background: #ECF4FF;

}
```

Rafaella: O programa englobou nossa imagem e texto, mas já conseguimos ver que o azul de fundo ficou quase coberto.

Para aumentarmos essa área do fundo e diminuirmos o retângulo branco, podemos criar uma margem com a distância que queremos entre o elemento anterior e o atual.

Então aplicaremos a propriedade `margin:` com uns 64 pixels de espaço. Agora começamos de fato a trabalhar com tentativa e erro do CSS, pois temos que ir testando se a distância colocada é satisfatória.

Tem algumas dessas informações no Figma, mas às vezes é interessante termos o controle das posições dos elementos em nosso código.

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

body {
      background: linear-gradient(236.85deg, #041832 27.26%, #3468A7
96.03%);
```

```
      font-family: 'Roboto', sans-serif;
}

.container {
      background: #ECF4FF;
      margin: 64px;
}
```

Rafaella: Parece que a margem ficou boa com esse espaço entre o fundo azul e o branco, mas podemos alterar se quisermos. A medida em *pixels* é o tamanho que usamos para distâncias no código.

Desta forma, já temos nossa margem e nosso retângulo de fundo branco. Agora precisaremos alterar algumas coisas sobre o nosso design.

Na aba "Inspect" do Figma, encontraremos no código CSS as propriedades `box-shadow:` e `border-radius:`. Para entendermos essas palavras, começaremos pelo `border-radius:`, que em uma tradução livre do inglês seria o "raio da borda".

O utilizamos para fazermos a borda arredondada do nosso elemento. Quanto maior for o valor deste raio, mais curvados serão os vértices deste elemento.

Para um exemplo, colocaremos o `border-radius:` igual a vinte pixels. Vamos salvar nosso arquivo e rodar para vermos a diferença.

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

body {
      background: linear-gradient(236.85deg, #041832 27.26%, #3468A7
96.03%);
      font-family: 'Roboto', sans-serif;
}

.container {
      background: #ECF4FF;
      margin: 64px;
      border-radius: 20px;
}
```

Rafaella: Poderemos ver que as "esquinas" da nossa divisão `.container` já ficaram arredondadas com vinte pixels.

Continuando, voltaremos ao CSS do Figma para entendermos a propriedade `box-shaddow:`, a qual faz a "sombra" por debaixo do nosso elemento.

Podemos simplesmente copiar e colar no nosso código porque já vem com algumas configurações de tamanho, cor e transparência para aplicarmos.

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

body {
      background: linear-gradient(236.85deg, #041832 27.26%, #3468A7
96.03%);
      font-family: 'Roboto', sans-serif;
}

.container {
      background: #ECF4FF;
      margin: 64px;
      border-radius: 20px;
      box-shadows: 6px 6px 6px #0E1D2F;
}
```

Rafaella: Feito isso, rodaremos para ver que há uma pequena sombra sob o elemento de fundo branco, a qual é mais visível quando o fundo azul está em tom mais claro, dando uma certa profundidade para o elemento.

Agora precisaremos estilizar a cor da fonte. Às vezes a cor *defaut* é a preta mesmo, mas como dissemos sobre a cor branca, há vários tons de preto também.

Vamos voltar ao Figma para pegarmos o código da cor preta utilizada na página, e levá-lo para nosso `.container` mudar em todos os elementos que contém.

Guilherme: Escreveremos a propriedade `color:` que é a cor dos textos, seguido do código `#1C1C1C` da cor preta, diferente do `background:` que diz respeito à cor do fundo.

Rafaella: Exatamente. Salvaremos e rodaremos para vermos se a fonte mudou alguma coisa.

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

```
body {
      background: linear-gradient(236.85deg, #041832 27.26%, #3468A7
96.03%);
      font-family: 'Roboto', sans-serif;
}

.container {
      background: #ECF4FF;
      color: #1C1C1C;
      margin: 64px;
      border-radius: 20px;
      box-shadows: 6px 6px 6px #0E1D2F;
}
```

Rafaella: A mudança foi bem sutil, mas queremos que seja fiel ao design, então é importante sempre checarmos essas questões.

A última coisa que precisamos fazer é alterar a posição da foto, que está muito grande e totalmente colada na borda do container, e queremos que tenha algum espaçamento e fique mais para dentro.

Para isso, usamos o `padding:` que atua de uma forma diferente do `margin:` que utilizamos para termos uma margem do elemento externo de fundo azul para o elemento interno de fundo branco que estamos estilizando.

Já o `padding:` faz diferente; dá uma margem do nosso container para os elementos que estão dentro dele. Colocaremos um valor de sessenta e quatro pixels para vermos como fica.

Vamos salvar e rodar para observar a página.

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

body {
      background: linear-gradient(236.85deg, #041832 27.26%, #3468A7
96.03%);
      font-family: 'Roboto', sans-serif;
}

.container {
```

```
      background: #ECF4FF;
      color: #1C1C1C;
      margin: 64px;
      border-radius: 20px;
      box-shadows: 6px 6px 6px #0E1D2F;
      padding: 64px;
}
```

Rafaella: Já está melhorando!

Paulo: Agora já está ficando mais separado e apresentável! Estamos passando propositalmente o `background`, `color`, `margin` e etc., que são bem comuns no cotidiano.

Estamos criando um *card*, que é exatamente o nome do projeto, "CertifiCard".

Rafaella: Agora já temos o nosso container de fundo branco já estilizado da maneira como sabemos até agora, mas ainda trabalharemos em cada um dos elementos.

A primeira coisa que vamos estilizar logo depois da `<div>` é o nosso `<header>` no arquivo HTML. Também colocaremos uma `class` para ele, a qual poderá ser igual a `"perfil"` por exemplo.

```
<body>
      <div class="container">
            <header class= "perfil">
                  <img src;"https://github.com/rafaballerini.png" />
                  <div>
                        <h1>Rafaella Ballerini</h1>
                        <h3>Instrutora e desenvolvedora front-end
#imersaodev</h3>
                  </div>
            </header>

            <main>

            </main>
      </div>
</body>
```

Rafaella: Para a estilizarmos, iremos ao arquivo CSS. e a chamaremos com o ponto, `.perfil` e abre as chaves.

Vamos estilizar do nosso `<header>`, mas se formos ao Figma, não encontraremos nenhum elemento entre o container, a imagem e o cabeçalho que consigamos identificar apenas visualmente.

Precisaremos estilizar o posicionamento dos elementos que irão estar dentro do cabeçalho. Para isso, criaremos uma nova classe no nosso elemento-pai `<header>` que engloba a imagem e os títulos, e então estilizaremos os elementos-filhos internos.

Queremos posicioná-los da forma correta, e para isso utilizaremos o chamado **FlexBox**, que é uma forma de conseguirmos estilizar dinamicamente os elementos dentro de uma tag-pai, que neste caso é o cabeçalho `<header>`.

Por meio deste FlexBox conseguimos dizer se queremos os elementos na horizontal, alinhados verticalmente, centralizados e etc. É uma forma simples, mas há diversas outras de posicionar, como usando `position: absolute` ou `grid:`.

Mas o FlexBox é algo que facilita muito a vida, pois só dizemos se queremos centralizado, alinhado à direita e até se queremos um espaço entre eles por exemplo, e ele faz "sozinho".

Inclusive, aqui na **Plataforma Alura** temos um artigo neste link muito interessante e completo sobre todas as propriedades do FlexBox com todas as opções disponíveis para explorarmos. Vale conferir!

Paulo: O Flexbox é uma "caixa flexível" e parece mágica. Podemos colocar muitas coisas nessa caixa para termos resultados bem diferentes e personalizados.

Rafaella: Neste caso utilizaremos bem pouco, então ficará bem simples mesmo. Não precisaremos ter um elementos numa posição de "x" e "y", pois o FlexBox já posiciona da forma que queremos, basta dizer.

Para o usarmos, precisaremos colocar um `display: flex` em `.perfil` primeiro de tudo. Isso colocamos realmente no elemento-pai, que é o cabeçalho `<header>`.

Feito isso, estamos dizendo que vamos usar o FlexBox de fato. Em seguida, utilizaremos a propriedade `align-items:` que serve para "alinhar os itens" com `center` para os alinharmos ao centro.

Salvaremos e rodaremos para vermos a diferença na página.

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

body {
      background: linear-gradient(236.85deg, #041832 27.26%, #3468A7
```

```
96.03%);
      font-family: 'Roboto', sans-serif;
}

.container {
      background: #ECF4FF;
      color: #1C1C1C;
      margin: 64px;
      border-radius: 20px;
      box-shadows: 6px 6px 6px #0E1D2F;
      padding: 64px;
}

.perfil {
      display: flex;
      align-items: center;
}
```

Rafaella: Com isso, todos os nossos elementos ficaram centralizados na página, e antes o nome e a descrição estavam embaixo da imagem.

Falamos para o cabeçalho que temos elementos internos que devem ficar centralizados na página, então ficarão ao centro. Mas notaremos que o nome "Rafaella Ballerini" está alinhado com a descrição embaixo.

Para entendermos, iremos ao arquivo HTML novamente. Os elementos-filhos do `<header>` não são a imagem, o título e o subtítulo, e sim a imagem e a `<div>`, a qual engloba outros dois elementos, os `<h1>` e `<h3>`.

Então temos na verdade dois elementos dentro do cabeçalho que estamos estilizando, então este segundo tem duas coisas em comum: o título e o subtítulo, e por isso estão juntos um acima do outro e não centralizados lado a lado.

Estamos fazendo isso justamente porque este é o formato que está no nosso Figma. Portanto é bom termos cuidado quando formos utilizar o FlexBox, prestando atenção aos elementos-filhos e ao pai, mas com o tempo fica mais fácil.

Já centralizamos a imagem e o texto da maneira como queremos, e a primeira coisa que faremos depois será deixar a borda da foto redonda, como vemos no Figma.

Como já vimos, existe a propriedade `border-radius:` que arredonda os limites de um elemento, até o formato circular. Na aba lateral de "Inspect" com a parte "CSS", encontraremos o valor.

Guilherme: que é de quatrocentos pixels para esta propriedade.

Rafaella: Então, já copiamos e levamos para a nossa *tag* de imagem. Talvez seja interessante criarmos também uma classe para a nossa imagem. O que você acha, Gui?

Guilherme: Vamos!

Rafaella: Dentro da nossa *tag* `img`, criamos um `class` e, pode ser, por exemplo, `"perfil-foto"`, isto é, `<img class="perfil-foto"`.

Guilherme: Uma provocação, Rafa. Semelhante ao que o Paulo havia comentado antes, nós poderíamos colocar o "img", como fizemos no body, e colocarmos a nossa imagem. Mas, quem garante que teremos uma imagem só na nossa página?

Paulo: O estilo seria aplicado em outras imagens. Essa é a diferença de usar a *tag* ou "ponto" e o nome de uma classe.

Rafaella: O ideal é testar tudo. Analisar o processo todo e, depois, você entenderá o motivo de algumas escolhas nossas. No nosso CSS, colocaremos o `.perfil-foto {` e a deixaremos redonda com a propriedade `border-radius: 460px;`. Vamos salvar, rodar e atestar se ela, de fato, está redonda. Ótimo! Funcionou. Agora, precisamos diminuí-la um pouco, porque ela está gigantesca.

Para isso, podemos colocar uma altura máxima. Nós mesmos definiremos essa altura com *max-height*, "altura máxima", que pode ter 160px, isto é, `max-height: 160px;`. Inclusive, é possível definir tanto *height* quanto a largura. Como agora a foto é redonda, tanto faz mexer na altura ou na largura. Ficou bom!

Além disso, no Figma, existia outra propriedade que não pegamos ainda, o *box-shadow*.

Guilherme: É o *filter drop shadow*?

Rafaella: Na verdade, por vezes, o Figma tem um formato diferente de apresentar algumas informações. Ele está colocando a sombra como um filtro e nós podemos pegar as informações que queremos da nossa sombra, que são as que estão dentro dos parênteses e colocar a propriedade que já conhecemos no CSS, que é o *box-shadow*. Portanto, Gui, você pode copiar as informações que estão entre parênteses e colocar, no código, o *box-shadow*.

```css
.perfil-foto {
  border-radius: 460px;
  max-height: 160px;
  box-shadow: 0px 4px 4px rgba(0, 0, 0, 0.25);
}
```

Depois da nossa foto, a próxima coisa que estilizaremos é, de fato, os escritos. O nosso nome e, também, o nosso subtítulo. Agora, vamos ao nosso HTML verificar como ele está. E, após o `img`, temos a `div`, que engloba essas duas coisas. Podemos, também, colocar uma classe para a `div`, a classe título, `<div class="titulo">`.

Seguindo, vamos ao CSS. Escreveremos, `.titulo`, para estilizarmos nossa classe. No Figma, notaremos que uma das únicas coisas que precisamos estilizar em conjunto, título e subtítulo, é puxá-los um pouco para a esquerda, porque estão um pouco grudados no nosso CodePen com a imagem. É interessante, portanto, puxarmos um pouco a margem da foto, para o nosso título ficar mais para a esquerda.

Voltando ao CodePen, é possível verificar como está essa parte. Está muito grudado! Sendo assim, colocaremos margem, que é a distância do elemento exterior para o que estamos estilizando. Neste caso, `margin-left`, porque é apenas a da esquerda. Não queremos colocar margem para cima, para baixo ou para a direita. A nossa margem terá 16px.

```
.titulo {
  margin-left: 16px;
}
```

Separou um pouco mais! Ficou melhor. Agora, estilizaremos o nosso `h1` e o `h3`. Nós não utilizaremos o `h1` em outro lugar, porque, normalmente, precisamos apenas de um, e `h3` também não. Portanto, mais abaixo, nós usaremos outros tipos de *tags*. Podemos até usar a mesma *tag* para que vocês vejam usos distintos do `h1` e h3, mas, algo interessante é unir a classe do elemento pai, exterior.

Por exemplo, no nosso `h1` e `h3`, o elemento de cima é o `div`, isto é, `<div class="titulo">`, que é o título. Nós podemos utilizar a classe desse elemento, dar um espaço e colocar a *tag* que queremos estilizar. Assim, deixaremos um pouco mais específico que apenas a *tag* `h1` estará dentro da classe *título*.

```
 <header class= "perfil">
      <img class="perfil-foto"
src="https://github.com/rafaballerini.png" />
      <div class="titulo">
        <h1>Rafaella Ballerini</h1>
        <h3>Instrutora e desenvolvedora front-end #imersaodev</h3>
      </div>
    </header>
```

No nosso CSS, faremos `.titulo h1 {`. Nós queremos estilizar a *tag* `h1`, que está necessariamente dentro de um elemento com classe título. Estamos especificando de

uma forma um pouco diferente do que quando colocamos apenas uma classe para ela. É uma maneira bem interessante e amplamente utilizada.

Seguindo, vamos estilizar a nossa fonte, o nosso escrito. Primeiro, nós havíamos separado dois pesos diferentes de fonte, a 400 e 700. A 700 era, justamente, a do nosso título. Podemos até verificar no Figma. Portanto, a primeira coisa que faremos é o `font-weight: 700;`. Nós também consultaremos o tamanho da nossa fonte, porque, por padrão, existe um tamanho de fonte que o próprio CSS deixa na nossa página.

É interessante conferir qual foi o tamanho escolhido pela/o *designer*. Para isso, vamos ao Figma verificar o tamanho da fonte também.

Guilherme: O tamanho é 36px.

Rafaella: Perfeito! Então, vamos colocar `font-size:`, que é "tamanho da fonte", igual a 36 px, `font-size: 36px;`, salvar e executar.

Guilherme: No caso, mudará só o `h1`, não é?

Rafaella: Exato! Provavelmente, não mudou muito, porque o próprio padrão é exatamente isso. Os próprios `h1` e `h3` têm padrões de estilização, semelhante ao que já tínhamos. Agora, falta estilizarmos o `h3`. Faremos da mesma maneira.

Guilherme: Posso copiar todo o `h1`, substituir por `h3` e alterar o valor da fonte?

Rafella: Pode!

Guilherme: Eu só não sei qual é o tamanho, vou procurar no Figma. O tamanho é 24px e o peso, 400px.

```
.titulo h3 {
  font-weight: 400;
  font-size: 24px;
}
```

Acho que, nesse, teremos alguma diferença, já que ele está bem maior. Vamos ver?

Rafaella: O peso mudou bastante e o tamanho também. Ficou muito legal a parte do cabeçalho e temos ainda muitos detalhes pela frente, para deixar bem mais bonita parte abaixo. Vamos para a segunda parte! Sobre o HTML, o Gui nos contará quais elementos precisamos criar para, depois, estilizá-los.

Guilherme: Não vou me atentar muito à parte da estilização, quero apenas a parte da estrutura do HTML. Nós temos outra propriedade chamada projetos. Todo o trecho de

código do HTML, nós colocaremos na propriedade *main* e temos um link que funciona da seguinte maneira: quando clicamos no link do projeto, ele o abrirá no CodePen. Portanto, precisamos de duas coisas, escrever o link e inserir a parte de projetos.

Há, no HTML, uma estrutura para criamos uma lista, que podemos chamar de *ul*, que é uma lista não ordenada.

Rafaella: Existe a **ul**, não ordenada e a **ol**, ordenada. Neste caso, não há ordem, o 1 e 2 não são passos de uma receita que devemos seguir. Ou seja, ela não precisa estar ordenada, por isso, usaremos *ul*.

Guilherme: Vamos dar o título da nossa lista, `<ul>Projetos<ul>`, e, cada linha dessa nossa lista representará um projeto. Vamos colocar uma `<li></li>`, sem esquecer de fechá-la. Ao lado, colocaremos o *view* do projeto, para focarmos no HTML. A primeira coisa que faremos é criar uma **tag âncora**. Âncora se refere às questões que estamos super acostumados a fazer, mas nunca paramos para pensar no que de fato está acontecendo.

Quando selecionamos determinada palavra, que abre uma outra *tag*, estamos selecionando uma "tag âncora", representada pela letra "a". Essa *tag* `a`, nós também adicionaremos e fecharemos, `<li><a></a></li>`. Nós abrimos a *li*, abrimos a *tag a* e fechamos a *li*. Dentro, nós colocaremos texto. Repare que não estamos colocando o texto dentro a, mas, sim, entre eles. Nós colocaremos o nome do projeto que fizemos.

Particularmente, gostei muito de dois projetos que desenvolvemos durante a imersão, por isso os colocarei. Um deles é o "Conversor de moeda", `<li><a>Conversor de moeda</a></li>`. Já é possível ver como aparecerá na tela: "Projetos", e, na linha abaixo, um marcador em formato de círculo e, à frente, "Conversor de moeda". Não consigo clicar nele, porque ainda não adicionamos ainda o link para onde queremos ir no projeto.

Dentro da nossa *tag a*, colocaremos outra propriedade para indicar o link da tag âncora, que colocaremos com `href=""` e, entre aspas, o endereço do projeto. Para isso, abriremos o CodePen e pegaremos o link para que, ao clicar, a pessoa seja direcionada para a página do CodePen. Nós selecionaremos, portanto, este endereço com "Ctrl + C", o colocaremos no `href`, salvamos e rodamos.

```
<main class="projetos">
      <ul>Projetos<ul>
        <li><a
href="https://codepen.io/guilhermeonrails/pen/poPZGov?editors=0111">C
onversor de moeda</a></li>
```

```
                    </main>
                </div>
            </body>
```

Repare que a escrita "Conversor de moeda" aparece até de outra cor. Quando apertamos o link, somos direcionados para a aula do Conversor de Moeda. Vamos retornar, pois, não queremos ficar naquela aula agora. Podemos fazer isso para todas as outras aulas. A diferença desse código para a próxima aula, por exemplo, será apenas o link e o nome.

Vamos testar com o projeto do Aluraflix, que ficou bonito. Basta copiar e colar o código dele, salvar e executar.

```
 <main class="projetos">
      <ul>Projetos<ul>
        <li><a
href="https://codepen.io/guilhermeonrails/pen/poPZGov?editors=0111">C
onversor de moeda</a></li>
        <li><a href="https://codepen.io/guilhermeonrails/pen/yLbxpwm"
> Aluraflix</a></li>
    </main>
  </div>
</body>
```

Ao selecionar o primeiro link com o botão direito, acessamos o projeto da aula 9, "Conversor de Moeda". Selecionando o segundo link, da mesma forma, acessamos a aula 5, "Aluraflix". Nós fizemos a estrutura, mas, esteticamente, está feio. Especialmente se comparado com a parte do nosso *header*, em que a imagem está estilizada, circular, nossos links não estão com uma boa apresentação.

Rafaella: A primeira coisa que conseguimos verificar é: no nosso Figma, temos uma imagem para cada projeto que fizemos. Essas imagens não são exatamente imagens, mas, sim, *emojis*, que podemos utilizar no nosso sistema operacional. É como se estivéssemos copiando e colando o texto. Ele é lido, não é uma imagem, como a src, que precisávamos copiar e colar.

De fato, é algo que conseguimos copiar como texto e colar. Para isso, temos o site *bag emoji*, que conta com uma grande quantidade de emojis. Nós podemos usar, por exemplo, o emoji da calculadora de notas. Vamos pesquisar por "number". Podemos passar o mouse por cima e copiar o próprio emoji. Algo interessante para quem usa o Windows é que, basta apertar a tecla "windows" e o ponto para algo mágico acontecer. Testem em casa.

Paulo: Acho que no Chrome, no "Menu Edit" também temos emojis, uma lista com vários. Guilherme, você poderia acessar o seu menu do Chrome para verificarmos isso? No “Edit”, encontramos "Emojis e símbolos", é possível escolhê-los e usá-los.

Guilherme: No Conversor de Moeda, coloquei um emoji de número, mas, no Aluraflix, o que vocês acham que eu deveria colocar?

Paulo: Se você pesquisar por "movie", ele deve mostrar algum relacionado.

Guilherme: Vou usar um de claquete.

Rafaella: Essa foi a primeira coisa que colocamos para deixar um pouco melhor esteticamente. Vamos rodar e ver como ficou. Está ótimo, mas ainda padrão. Podemos estilizar, lembrando de sempre pensar primeiro nas coisas mais externas e depois nas mais internas. Após a nossa *tag* de *header*, o nosso cabeçalho, nós tínhamos, em seguida, a tag *main*, que tem o conteúdo principal da nossa página. Nós daremos uma classe para ela.

Para isso, voltaremos ao HTML e colocaremos a classe dentro da *main*, uma `classe="projetos"`, que será, de fato, o conteúdo dos projetos que temos.

```
<main class="projetos">
      <ul>Projetos<ul>
        <li><a
```

No CSS, nós estilizaremos o `.projetos` e temos muitas coisas a pensar pela frente. A primeira delas é, de fato, o nosso *flexbox*, que estávamos usando para posicionar os elementos que estão dentro dessa *tag*, que, no caso, formarão a nossa lista. Então, nós usamos o `display: flex;` para indicarmos que queremos usar o *flexbox*.

Seguindo, nós utilizaremos uma propriedade do *flexbox* que é o `flex-direction`, a direção dos elementos que estarão dentro. Como eu já havia comentado, pode ser horizontal, vertical, enfim, existem várias propriedades e nós disponibilizamos o link de um artigo para que vocês confiram todos.

Nesse caso, nós utilizaremos uma direção vertical, de coluna, por isso, adicionaremos `column`, pois queremos que a nossa lista fique estilizada nesse sentido, o *display: flex* normalmente vem como horizontal, portanto, é importante ajustar o *flex-direction* como `column`.

```
.projetos {
```

```
  display: flex;
  flex-direction: column;
}
```

Além disso, vamos alinhar todos os nossos elementos no centro. Usaremos a propriedade do `align-items: center;`, para deixá-los mais centralizados. Agora, basta salvar e rodar. Vamos ver como ficará com essa estilização.

Paulo: Escolha polêmica essa centralização. Achei mais bonito como estava antes, mas, cada um decide.

Rafaella: Podemos deixar também. Querem testar? Vamos deixar descentralizado, pode tirar o `align-items: center;`, Gui.

Guilherme: Engraçado que estamos falando sobre isso. Essa é uma questão interessante, eu gostaria de destacar um ponto: o time de designer manda para o Front-End, mandam algumas coisas que, às vezes, pensamos que não faz tanto sentido. É necessário que exista essa conversa.

Rafaella: É preciso cuidado, porque, pensando sobre o design, eu sei que eles estudam muitas coisas, então, por trás existem diversos motivos pelo qual é centralizado, por exemplo. Motivos que nós não entendemos, mas que estão relacionados com experiência do usuário e que, realmente, fazem muito sentido. O ideal é sempre perguntar, "será que não seria legal mudar?" e entender o motivo de estar de determinada maneira.

Guilherme: Ficou bem legal todo o nosso layout! A imagem, a questão de pegar um `h3` e um `h1` de uma determinada `class`, ficou muito interessante. Nós fizemos esse layout do zero. Amanhã, daremos continuidade ao layout e faremos coisas mais legais ainda. Nós estamos listando e, quando clicamos, abrimos o projeto do CodePen, mas, não seria bom se pudéssemos pegar o projeto CodePen e colocá-lo dentro desse nosso projeto?

E mais, será que não conseguimos colocar todo o visual que estamos vendo no ar com o link que mandamos para as nossas amigas e amigos, eles clicam e já conseguem ver tudo aberto, todos os projetos que fizemos com nome. Isso que faremos amanhã! Rafa e Paulo, vocês têm algum desafio?

Rafaella: Com certeza! Nós disponibilizaremos o Figma de base para a nossa aula. Para mim, o desafio é vocês, realmente, fazerem o que falta. Agora vocês já têm conhecimento suficiente para conseguir fazer isso com o CSS e de forma bem tranquila. Nós mudaremos algumas coisas, assim como comentamos, na próxima aula, mas é interessante vocês fazerem para treinar entenderem, de fato, a parte de estilização.

Paulo: Você pode até pensar na utilidade de ter um espaço como o que construímos, sendo que já existe o LinkedIn e outros espaços para guardar currículo. Mas, neste projeto, você está se apresentando como Dev. Está mostrando seus projetos, seu *portfolio*, enfim, tudo o que você está aprendendo conosco e que também aprendeu na internet e em outros lugares.

Usando o *img src* ou até outros mecanismos, você pode adicionar o logo do seu Instagram, Facebook, LinkedIn, e-mail, das suas redes, seu CodePen geral. Enfim, ficará bem bonito esse cartão com os links para as suas redes e com botões bonitos. É possível usar o logo do Instagram mais colorido, do que preto e branco e pesquisar no CSS e em outros lugares como fazer transições.

Colocar os links e as imagens, nós ensinamos, mas dá para você deixar mais aperfeiçoado, "redondo", e usar coisas que são importantes para você que está buscando o primeiro ou segundo emprego, ou para que você faça parte do Aluraverso. Vou até deixar algumas inspirações de CSSs e de projetos, porque alguns de vocês já estão na reta final, perguntando o que mais podem fazer, sobre outras ideias de projeto.

A minha dica é que, no CodePen, pelo "Search" (barra de busca), digitando "Imersão Dev", ele mostrará, por relevância, algumas das edições passadas, que foram muito votadas. Aliás, por isso, não deixe de votar no CodePen do seu amigo e amiga que você gostou muito, comentar.

Assim, é possível ter ideias usando, por exemplo, *if*, CSS, fazer conversores mais estilizados. Existem alguns certificados das edições passadas, para vocês perceberem que já foram um pouco diferentes, você pode se inspirar, seja no Layout ou nas ideias, para deixar seu *portfolio* com a sua cara. No Discord, também é possível encontrar outras ideias que não foram tão votadas, mas que são interessantes.

Nós vamos deixar, na aula, algumas ideias do que mais você pode fazer para complementar seu *portfolio* e mostrar não só o que você aprendeu, mas a sua capacidade de aprender. É isso que as empresas buscam, especialmente em um primeiro momento de contratação, em que ainda não conseguem te julgar por sua capacidade técnica, mas conseguem perceber sua dedicação.

Percebem que você foi além do proposto, pesquisou coisas novas do CSS e JavaScript, tirou dúvidas, ajudou outras pessoas no Discord, isso é muito importante em um profissional. É isso que buscamos!

Rafaella: Então é isso. Espero que tenham gostado. Hoje estudamos um pouco mais sobre essas duas tecnologias, HTML e CSS, que não havíamos nos aprofundado muito durante as outras aulas da imersão. Até gostaria de saber o que vocês gostaram mais, quem é mais do Front, que gosta da estilização, e quem é mais da lógica de programação, do JavaScript. É interessante também para vocês fazerem essa análise.

Amanhã aprenderemos algo muito legal! Nosso *portfolio* ficará cada vez melhor e, além disso, nós também o levaremos para fora do CodePen, em um lugar muito interessante, em que poderemos, de fato, compartilhar com qualquer pessoa da web. Enfim, nós o colocaremos no ar. Então, é isso! Estamos no Discord, vocês podem mandar mensagem, postar no LinkedIn, que olharemos tudo que vocês estão postando. Bons estudos e até amanhã!