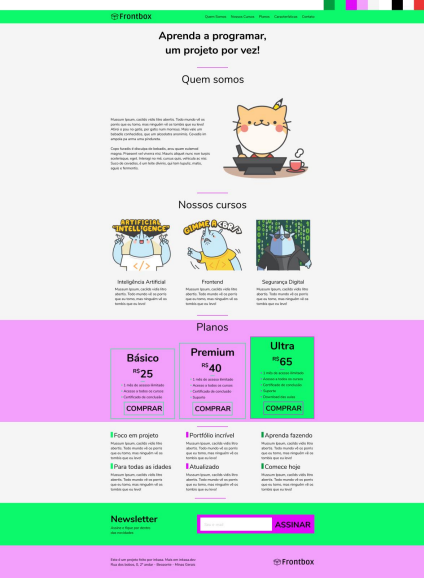
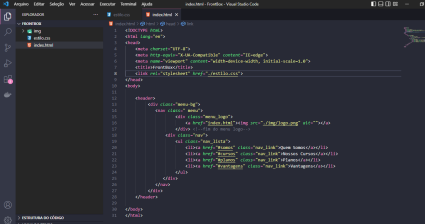
**Projeto 1**

Neste projeto vamos aprender como criar um site Responsivo.

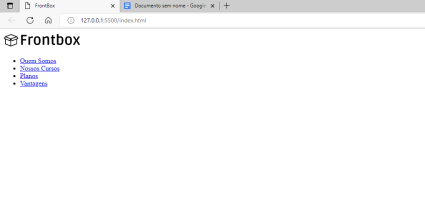


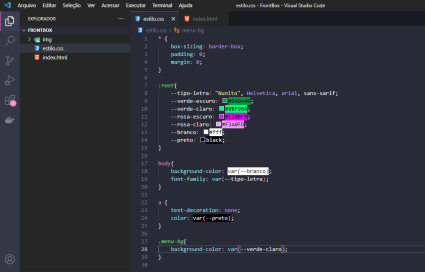
Aplicando tudo que aprendemos até agora: html, css e javascript. Esse é o protótipo do site que vamos desenvolver. Devemos ter a capacidade de imitar a página, usando nossos códigos.

Agora vamos colocar a mão na massa, meus queridos colegas, **vamos criar uma pasta e vamos dar-lhe o nome de FrontBox**. Dentro da pasta, vamos colocar imagem para nós poder usar a desenvolver o site.



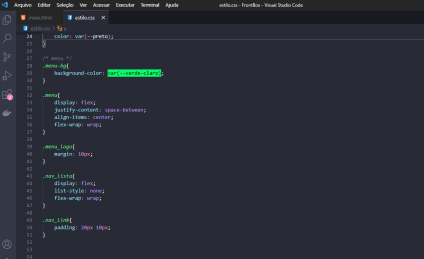
Nesse momento já temos a nossa pasta no visual studio code. Já temos a nossa pasta imagem, ficheiro **index.html** e o **estilo.css.** No ficheiro html vamos criar a nossa estrutura básica para fazer o nosso menu do nosso site. Pelo que estamos a ver na imagem, estamos a usar as tags semântica de modo organizar o nosso projecto. No nosso menu html, temos o logo do nosso site e a nossa lista desordenada. cada disse que criamos está a receber uma classe, de modo eu poder mexer no css.É importante colocar classes ou id nas tags, para te facilitar quando tiveres a mexer no css de cada elemento do html.

Utilizando apenas html, o nosso site fica dessa forma. A única coisa que temos até ao momento é o logo da página e a nossa lista. Agora vamos mexer com css, de modo a dar estilo a nossa página.

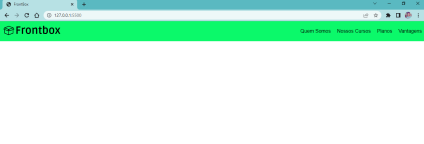


Nesse momento estamos a mexer no css, assim como estão a ver na imagem. criamos um selector global, onde aplicamos um **box-sizing: border-box;** de modo a controlar a largura e altura do meu navegador da minha página. Tivemos que aplicar um **padding: 0; e margin:0;** de modo eliminar todas as margem interna e externa. O :**root** é o mesmo que dizer **html**, é a raiz da nossa página. Na nossa raiz aplicamos o tipo de letra e as cores que vamos utilizar durante a construção do site. Aplicamos uma cor de fundo branca no corpo da nossa página. Isso quer dizer, que o fundo inteiro da nossa página é branco. Aplicamos uma cor verde no fundo da nossa página.

O nosso site está assim nesse momento. Vamos dar continuidade a nossa página.

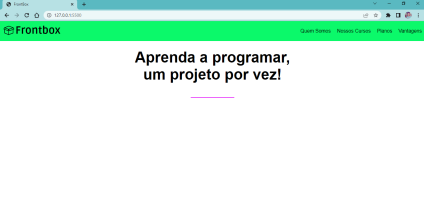


Agora vamos utilizar o flexbox de modo a tornar o nosso site responsivo. No menu do nosso site tivemos que aplicar um display: flex; de modo a separar elemento logo do site e os menus da página. Para isso tivemos que aplicar justify-content: space-between; aplicamos também margem internas entre os itens da lista de 20px para parte **superior** e **inferior**. 10px para **esquerda** e **direita**. Nem todos os códigos serão, explicado, porque são códigos que já foram explicados na aula. Agora vamos ver como ficou o site na internet.

Agora vamos mexer no html, de modo a colocar mais conteúdo.

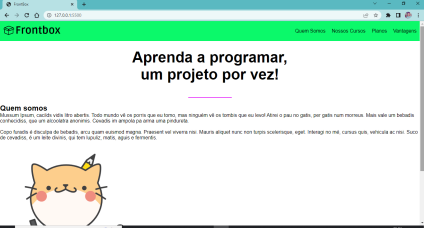
Depois de terminar a div do menu, criamos um titulo da página **“Aprenda a programar, um projeto por vez!”** depois disso, criamos uma linha de baixo do titulo, usando a tag <hr>. Agora vamos levar isso no css.

O nosso titulo recebeu uma classe = “ intro”, que vamos utilizar agora no css. O nosso titulo todo queremos colocar como negritado. Para isso tivemos que usar o **font-weight: bold;** aplicamos um tamanho de 3rem no titulo, e centralizamos o nosso texto no centro. Para colocar a linha, tivemos primeiro que dar uma cor de fundo, depois colocar um tamanho e uma altura. Agora vamos como ficou no navegador.



Pelo visto a nossa página está a ficar bonito. Uma forma de memorizar os códigos é praticar sempre. Vamos continuar a codificar a nossa página.

Acabamos de criar a sessão quem somos, criamos parágrafo e colocamos imagem. Agora vamos visualizar como ficou no navegador.



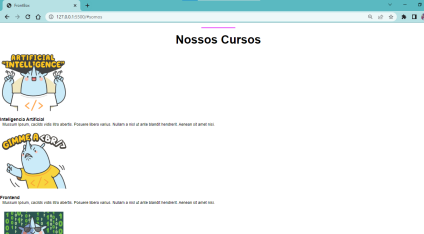
Nesse momento já temos o título, o parágrafo e a imagem. Agora devemos mexer no css, de modo organizar os elementos no site.

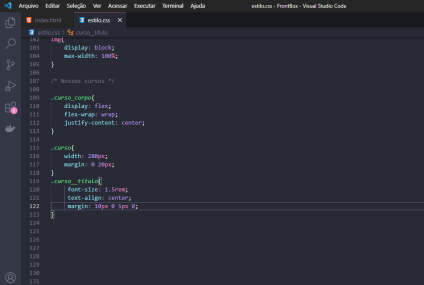
Até ao momento a única coisa de novo é **flex: 6 1 160px;** essa é uma propriedade do flexbox de modo a controlar o tamanho dos elementos filhos. O valor 6 pertence a propriedade **flex grow:6; flex-shrink: 1;** e o **flex-basics: 160px;** O quando **colocarmos flex-grow: 6,** estamos a dizer que o texto deve ser, 6 vezes maior que os outros itens do elemento pai. Nesse caso a

imagem. Então o tamanho que margem ocupa do texto é 6 vezes maior que a imagem. **Flex shirink:1;** está a querer dizer que o texto deve reduzir o tamanho, para caber na margem principal que é do **somos\_corpos. Flex-basis: 160px;** está a querer dizer que o tamanho inicial do texto é de 160px.

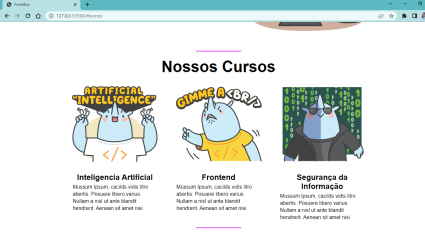
Vamos dar continuidade ao código do site que estamos a fazer.

Nesse momento criamos uma nova sessão que tem como nome “**nossos cursos**”. Vamos poder visualizar agora no navegador.

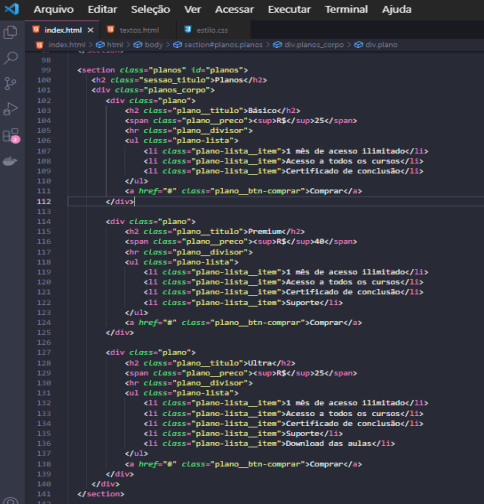


Agora vamos mexer no css de modo a deixar o nosso curso centralizado e bem organizado. 

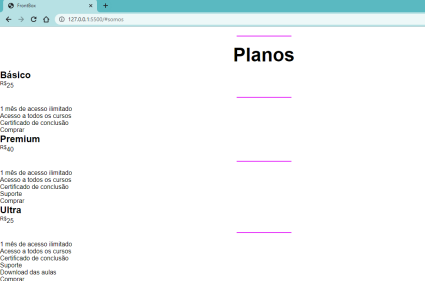
No elemento pai, aplicamos um **display:flex**; de modo que os elementos filhos ficam de forma de linha e não de coluna. Diminuímos também a largura dos cursos. E fizemos a separação entre os elementos. Vamos ver como ficou.

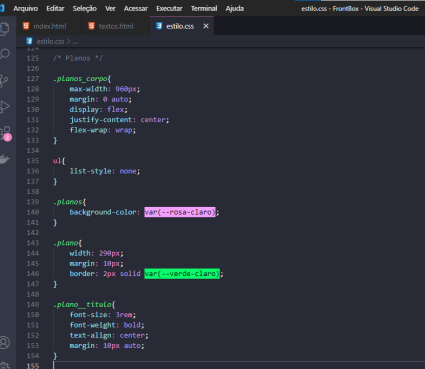


Nesse momento estamos a ver os nossos cursos centralizados, utilizando o **justufy-content: center;** vamos continuar a codificar.



A maioria dos códigos são repetidos. A única coisa que aumentou é a tag <a> que vai servir como botão. Vamos visualizar e ver como está.

Esse é o resultado da nossa página sem o css. Agora vamos mexer no css e ver como vai ficar.







Pelo visto o nosso o código css ficou bastante grande. Mas isso não é um problema. Se você o que cada propriedade do css faz, vai facilitar a tua compreensão. Porque a maioria das propriedades estamos a usar várias vezes. Caso tenhas dúvida, volta na aula de css, onde explica cada propriedade css. Única propriedade é o **::before** Insira algo antes do conteúdo de cada elemento. **: last-child** Seleciona todos os elementos que é o último filho do seu elemento pai.



Esse é resultado do código que acabemos de o fazer. Vamos continuar com os códigos no vs code.

Nesse momento temos a vantagens do curso do site. Agora vamos ver como está sem o css.



Esse é o resultado das vantagens do site. Mas não temos o css. Agora vamos mexer no css e ver como está o nosso site.



A única coisa que temos de diferente **:nth-of-type(3n + 1),** essa propriedade escolhe a posição do elemento de acordo com o número que está entre parentes. Inicialmente n = 0; n = 0, 3(0) + 1 = 1; então ele vai selecionar o primeiro elemento. Depois quando n = 1, vai selecionar o ultimo elemento.



E estamos a haver o resultado, como ficou no navegador. Já estamos quase a finalizar o site.

Nesse momento criamos o nosso newsletter e o footer. Agora vamos ver o código css.





Nesse momento terminamos com os códigos html, css. Vamos ver como está a página. Depois vamos mexer no javascript e podemos visualizar o site.



Essa foi o ultima parte do retoque no site sobre html e css. Conseguimos imitar a página do layout.



Nesse momento estamos a ver página inteira do layout que foi imitado. Agora vamos mexer no **javascript**, de modo a criar um menu que vai ficar dentro do icon. Desse modo vamos ter que mexer no **HTML**, **CSS** e vamos criar um ficheiro **javascript**. Entretanto, agora mexer no **HTML**.



Nesse código temos, mas conteúdo dentro da nossa tags. Além de ter um menu normal, também teremos um **menu mobile.** De modo que quando eu minimizar a minha tela do navegador, ela fica no formato de um telemóvel. Por isso tivemos que criar um **menu mobile.**

Dentro da tags **button**, criamos um evento javascript chamado **onclick** de modo que quando eu dar um clique no icon, **ele possa abrir-me um menu no formato mobile.** A imagem está no formato svg. E dentro do evento onclick vamos criar uma função chamado **MenuShow().** Agora vamos mexer no ficheiro css.



Nesse código a única coisa nova é só a classe .**mobile-menu-icon** e o .**mobile-menu** que tem como propriedade **display: none;** de modo que o meu icon menu não seja mostrado enquanto estiver em tela cheia.



Agora aqui temos um css no formato mobile com num tamanho de tela de **730px.** Todos os códigos que vamos aplicar nesse momento, servirá apenas para efeito numa tela de 730px. Para isso criamos um media e aplicamos um tamanho para ela. E nesse código a única coisa que mexemos foi o menu e colocar no formato de telemóvel. Quando aplicamos uma media screen, vamos apenas selecionar os elementos que queremos colcoar no formato mobile, Porque que não podemos colocar tudo dentro media? Porque já usamos o flexbox para organizar os elementos. Agora vamos mexer no ficheiro javascript. Porque já criamos uma função chamado **menshow().**

****

Nesse código javascript criamos uma função com nome **menuShow().** Onde dentro dessa função criamos uma variável com nome **menuMobile**, de modo que ela seleciona o documento que ela vai precisar manipular. Nesse caso é a classe **.mobile-menu** que encontra-se em uma tag html. Depois disso, criamos uma estrutura de condição onde a variável menuMobile vai receber um classlist que está no css de um tipo booleano, que vai retornar verdadeiro ou falso. Agora vamos explicar o que estar a acontecer dentro da estrutura de condição.

if (menuMobile.classList.contains('open'))

Se o menuMobile tiver uma classlist open no meu css e estiver aberto, assim que demos um clique, então vamos remover ou fechar o menu.

menuMobile.classList.remove('open');

Se não tiver um menu open, ele vai adicionar.

else {

menuMobile.classList.add('open');

document.querySelector('.icon').src =

"./img/close\_white\_36dp.svg";

}



Esse é o resultado do site aplicando HTML, CSS e JAVASCRIPT.