05

Nossa Primeira App

Transcrição

Vamos criar o nosso primeiro aplicativo híbrido com HTML e CSS no Cordova. Iremos trabalhar com o restaurante **Só de Cenoura** que vende apenas bolo de cenoura e produtos do gênero.

Eles possuem um estabelecimento, que utiliza cardápios impressos com todas as opções para os clientes. Com isto, eles têm todos os problemas clássicos do mundo offline: quando existe alguma opção de preço ou nos pratos, eles precisam imprimir novamente os cardápios. O mesmo acontece quando o cardápio fica desgastado, além de dificuldade com os pedidos. Eles gostariam de modernizar o seus cardápios e nos convidaram para desenvolver algumas apps.

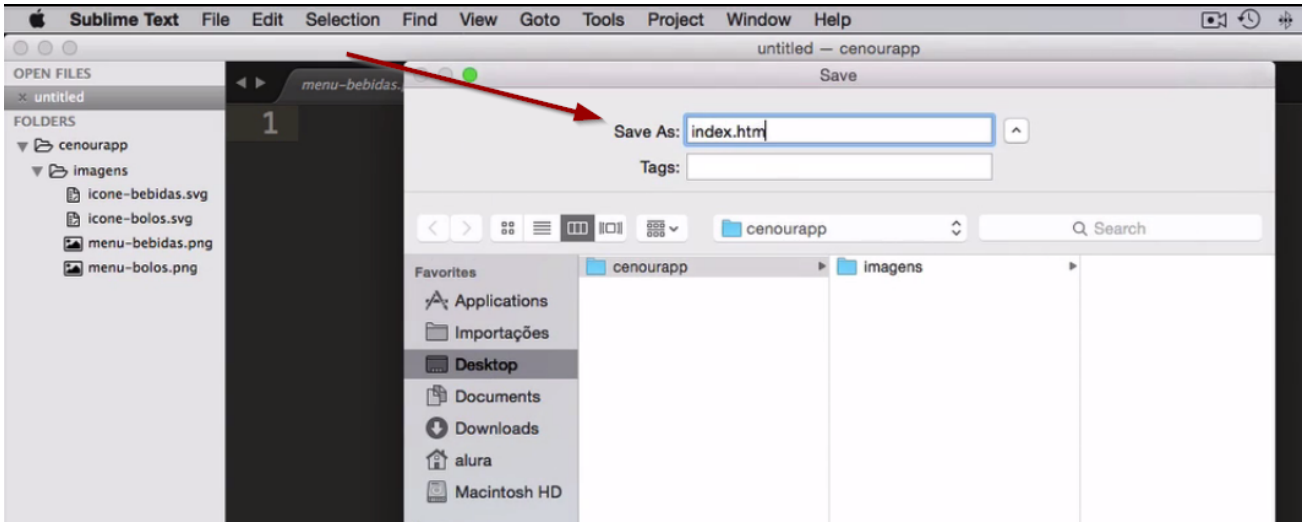
O primeiro cenário que resolveremos, será o do cardápio. Iremos criar uma versão digital para disponibilizar para os clientes do local, sem nenhuma função complicada, apenas para a visualização do cardápio. Assim poderemos otimizar a atualizações do cardápio, sem precisar reimprimi-lo frequentemente.

Eles disponibilizaram as folha do cardápio. São duas imagens feitas no Photoshop. Em uma das páginas temos diversas opções de bolos e em outra, teremos as bebidas oferecidas como café, chá e sucos. Este é o cardápio impresso atual.

Nosso primeiro passo será transformar isto em um app utilizando o Cordova, que irá permitir que o usuário visualize o cardápio. Quando forem necessárias atualizações, apenas faremos alterações nos arquivos das imagens.

Observe que estamos usando o ***Sublime*** como editor e que criamos o projeto **cenoura**. Nós incluímos as imagens do Menu e dois ícones (arquivos com extensão `.svg`) para as opções do cardápio.

O que precisamos fazer? Uma app Cordova é baseada e HTML, CSS e JS - ele trabalha como uma Web, porém dentro de um app. Iremos criar um arquivo `index.html` ao HTML, em que criaremos nossa página.



Começaremos a criar o nosso arquivo, com uma estrutura básica incluindo o `DOCTYPE html` , `meta` e o título `Só de Cenoura` .

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Só de Cenoura</title>
</head>
<body>

</body>
</html>
```

Vamos começar a implementar esta HTML. Antes, precisamos lembrar que estamos trabalhando com um app mobile que irá rodar como em um navegador móvel no celular. Então, iremos trabalhar com design responsivo. Logo, precisaremos incluir a famosa *meta tag* `viewport` :

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

O nosso código ficará assim:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Só de Cenoura</title>
</head>
<body>

</body>
</html>
```

Quem já trabalhou com design responsivo, viu esta meta tag várias vezes. Ela permitirá que o design que criaremos em HTML e CSS se adapte a resolução (a largura) da tela do dispositivo. Terá um uso padrão em todos os designs responsivos.

Agora, queremos que o app mostre as duas imagens do cardápio e que permita o usuário alternar entre **Bebidas** e **Bolos**. Vamos incluir os arquivos de imagem no HTML.

```
<img src="imagens/menu-bolos.png" alt="Menu Bolos">
```

Aproveitaremos para incluir alguns `ids` que serão úteis para referenciarmos estes arquivos no CSS. Iremos colocar uma classe chamada `menu` .

```
<img src="imagens/menu-bolos.png" alt="Menu Bolos"
    id="menu-bolos" class="menu">
```

Por que fizemos isto? Porque também teremos o menu bebidas. Iremos adicioná-la no código também:

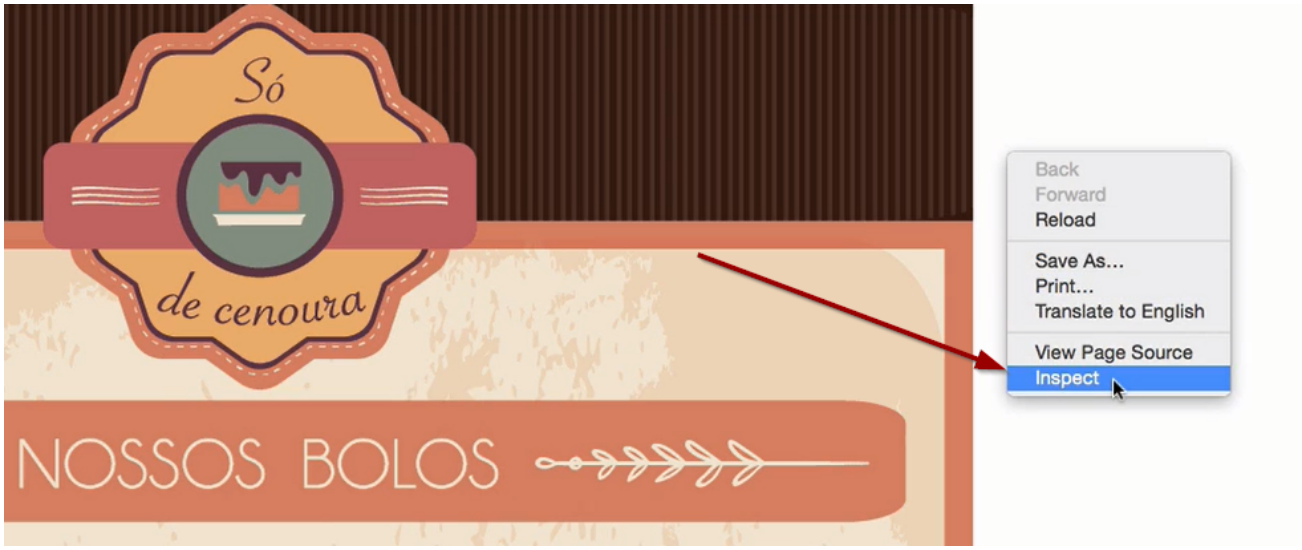
```
<img src="imagens/menu-bebidas.png" alt="Menu Bebidas"
    id="menu-bebidas" class="menu">
```

Os dois terão a classe `menu` , mas `ids` diferentes.

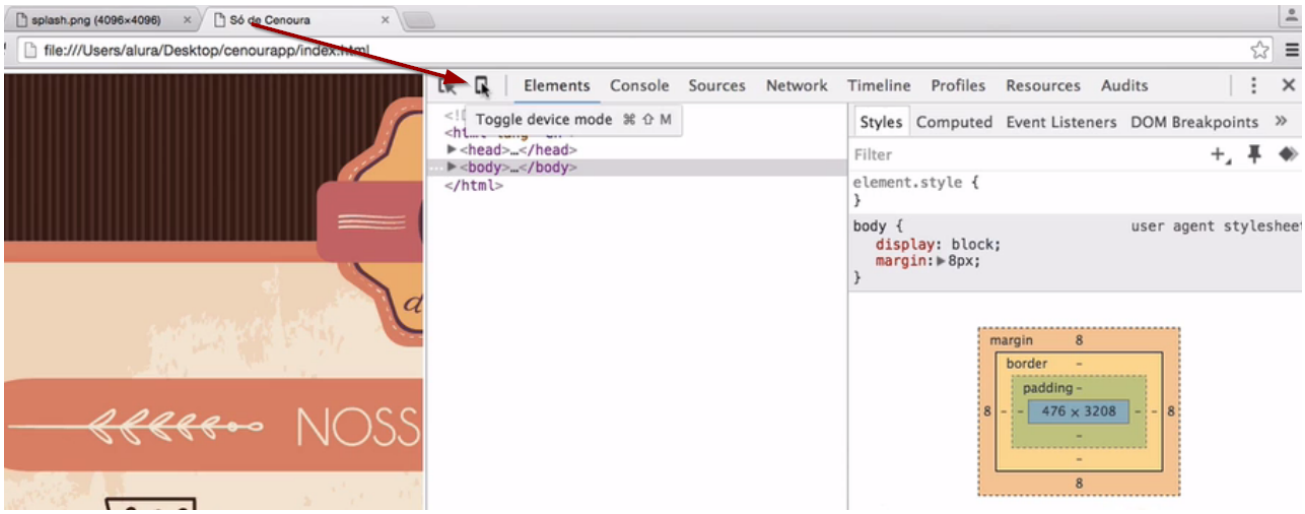
O nosso código está assim:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Só de Cenoura</title>
</head>
<body>
    <img src="imagens/menu-bolos.png" alt="Menu Bolos"
    id="menu-bolos" class="menu">

    <img src="imagens/menu-bebidas.png" alt="Menu Bebidas"
    id="menu-bebidas" class="menu">
</body>
</html>
```

Iremos jogar o nosso projeto no Chrome. Ele irá exibir as duas imagens, uma embaixo da outra. Veremos como é possível visualizar a página no celular. Vamos usar o inspetor de elementos, do Chrome.



O navegador irá abrir o inspetor, ele disponibilizará um ícone que é o ***Device mode***.



Ele dará a opção de escolher um dispositivo, por exemplo, o *Google Nexus 5* e nós poderemos como a página será visualizada no aparelho.

Não poderemos ver bem a imagem, porque ela ainda está gigante. Mas basicamente o que temos é um HTML com duas imagens.

Iremos arrumar a largura da imagem no CSS. Para isto, criaremos um arquivo novo que chamaremos de `estilo.css` .

Voltaremos no HTML e importaremos como `link` o arquivo .css.

```
<link rel="stylesheet" href="estilo.css">
```

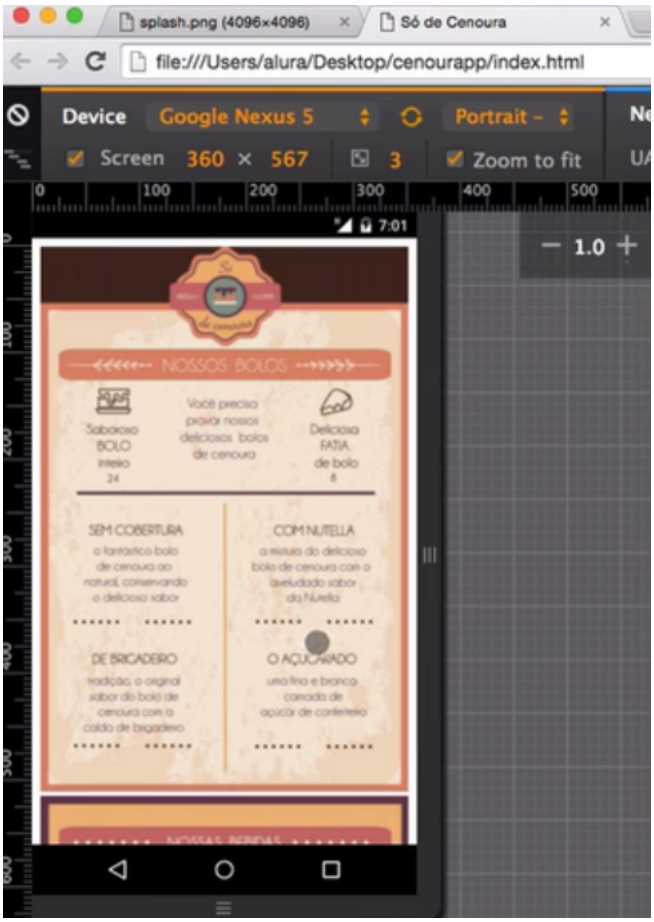
A linha estará localizada abaixo do título.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Só de Cenoura</title>
    <link rel="stylesheet" href="estilo.css">
</head>
<body>
    <img src="imagens/menu-bolos.png" alt="Menu Bolos"
    id="menu-bolos" class="menu">

    <img src="imagens/menu-bebidas.png" alt="Menu Bebidas"
    id="menu-bebidas" class="menu">
</body>
</html>
```

Vamos para aba do `estilo.css` e colocaremos algumas regras. Por exemplo, indicaremos que o `menu` não deve ultrapassar o limite da largura. Vamos indicar que o `width` das duas imagens será de 100%

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

Ao acessarmos novamente o navegador, veremos que a visualização do projeto está melhor.



As imagens estão posicionadas uma embaixo da outra. O aplicativo está feio, mas já é "usável".

De volta ao HTML, detalharemos no código que não queremos que as duas imagens apareçam simultaneamente. Através de um menu, o usuário tenha a opção de trocar de imagem. Como implementaremos isto com um HTML simples? Com dois *Radio Buttons* .

Teremos um `input type="radio"` que terá o nome de `opcao` . Criaremos uma `id` para a opção de bolos (`opcao-bolos`) e outra para `bebidas` (`opcao-bebidas`).

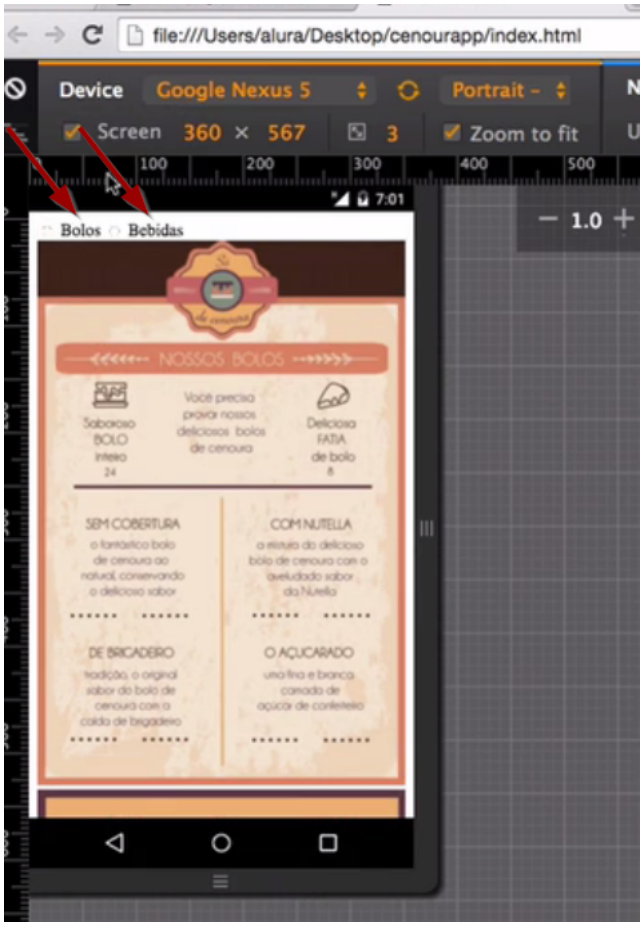
```
<body>
    <input type="radio" name="opcao" id="opcao-bolos">

    <input type="radio" name="opcao" id="opcao-bebidas">

    <img src="imagens/menu-bolos.png" alt="Menu Bolos"
        id="menu-bolos" class="menu">

    <img src="imagens/menu-bebidas.png" alt="Menu Bebidas"
        id="menu-bebidas" class="menu">
```

```
</body>
```

Para acompanhar o `input radio` é comum criarmos `label` s. Vamos incluir um label para `opcao-bolos` e outra para `opcao-bebidas` .

```
<input type="radio" name="opcao" id="opcao-bolos">
<label for="opcao-bolos">Bolos</label>

<input type="radio" name="opcao" id="opcao-bebidas">
<label for="opcao-bebidas">Bebidas</label>
```

Veremos novamente no navegador as alterações feitas até agora.



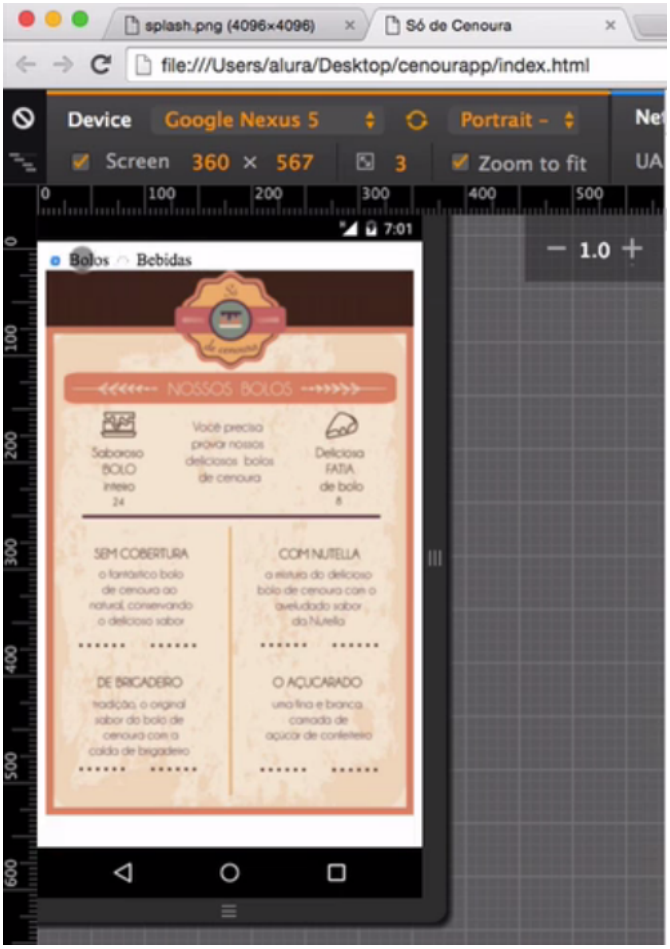
Temos `labels` com `input` , em que nos permite clicar nas opções "Bolos" e "Bebidas". Mas ainda não fazemos nada com isto. Queremos que quando clicamos na opção "Bolos", ele mostre apenas o cardápio de bolos e faça o mesmo com a opção de "Bebidas". Faremos isto com um CSS puro.

Na aba de `index.html` do terminal, iremos especificar no código que ele deve esconder a imagem que não está selecionada. Por exemplo, se tivermos a opção "Bolos" selecionada no app, queremos que o cardápio de bebidas fique oculto. Queremos que o contrário aconteça quando selecionarmos "Bebidas": o cardápio de bolo não deve aparecer. Na aba CSS conseguiremos fazer isto com alguns seletores avançados. Vamos criar isto no código.

Se tivermos a `opcao-bolos` selecionada usaremos a opção do seletor `checked` . Enquanto isso, para o `menu-bebidas` usaremos um `display: none` .

```
#opcao-bolos:checked ~ #menu-bebidas {
    display: none;
}

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

Estamos dizendo que se marcamos a `opcao-bolos`, esconderemos o `menubebidas`. O que significa o seletor `~`? Que a primeiro pedaço (`opcao-bolos`) da linha é **irmão** do segundo (`menu-bebidas`). Se retornarmos ao nosso HTML, veremos que todas nossas tags são irmãs. Isto significa que quando tivermos uma opção marcada, a tag irmã usará um `display: none`. Em seguida, testaremos se o código funciona:



Se marcarmos a opção "Bolos" no menu, o aplicativo irá esconder o cardápio de "Bebidas". Mas se selecionarmos "Bebidas", o cardápio de bolos ainda estará visível. Precisamos fazer com o menu de bebidas o mesmo que fizemos com o de bolos no CSS.

```
#opcao-bolos:checked ~ #menu-bebidas,
#opcao-bebidas:checked ~ #menu-bolos {
    display: none;
}

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

Assim, quando selecionarmos a `opcao-bebidas` ele irá esconder o `menu-bolos` . Se testarmos novamente no navegador, veremos que quando selecionamos o menu "Bebidas", o cardápio de bolos irá desaparecer.

Outra melhoria que podemos fazer é que, por padrão, a opção "Bolos" venha selecionada, quando o aplicativo carregar pela primeira vez. Assim, ele exibirá apenas o primeiro cardápio.

Acrescentaremos `checked` na linha do `input radio` da `opcao-bolos` .

```
<input type="radio" name="opcao" id="opcao-bolos" checked>
<label for="opcao-bolos">Bolos</label>

<input type="radio" name="opcao" id="opcao-bebidas">
<label for="opcao-bebidas">Bebidas</label>
```

Agora, ele irá carregar o aplicativo com a opção Bolos selecionada e depois, podemos trocar para o próximo cardápio. Nosso aplicativo tem uma funcionalidade simples, com um CSS simples.

Em seguida, queremos deixar o design mais bonito. Para isto, voltaremos no CSS e faremos alguns ajustes. No corpo (`body`) do app existia uma margem. Iremos retirá-la.

```
body {

    margin: 0;
}
```

Acrescentaremos também um `background` com uma cor que combine com meu menu do "Só de Cenoura". Faremos outros ajustes, como deixar o texto centralizado, trocar a fonte para `sans-serif` .

```
body {
    background: #3D1A11;
    margin: 0;
    font-family: sans-serif;
    text-align: center;
}
```

Se atualizarmos o navegador, já veremos mudanças.

Observe que o menu está localizado no meio, temos um design sem margem, com a cor do fundo que combina.

Próximo passo será incluir melhorias no label. Iremos mover o menu para a parte de baixo do aplicativo. Ele também será colorido e com ícones, sem as atuais bolinhas azuis.

Faremos as alterações no CSS, por partes. Começaremos tirando o `input radio` com `display: none` . Com isto, o nosso clique continuará funcionando e não será quebrado.

```
input[type=radio] {
    display: none;
}
```

Acrescentaremos as novas linhas abaixo do `body` no código.

```
body {
    background: #3D1A11;
    margin: 0;
    font-family: sans-serif;
    texte-align: center;
}

input[type=radio] {
    display: none;
}
```

Mesmo que o `input` esteja escondido, o próprio `label` já é clicável no HTML.

Também iremos fazer alterações do design do `label` , por exemplo, na cor. Adicionaremos um `background color` . Mudaremos a cor do `label` para branco e combinar com design escuro.

```
input[type=radio] {
    display: none;
}
label {
    background-color: #563429
}
```

A fonte do menu ficou branca.

Vamos mudar outras propriedades. Mudaremos o `display` , o `font-size` , o `padding` , `text-transform`

```
label {
    background-color: #563429
    color: white;
    display: block;
    font-size: 75%;
    padding: 4em 0 1em;
    text-transform: uppercase;
}
```

Nosso aplicativo está assim agora:

Queremos colocar um ícone no espaço entre os `label` s. Para isto, voltaremos a aba `estilo.css` e na parte referente ao `label` do nosso código, pegaremos o da `opcao-bolos` terá um `background-image` com a URL do ícone da opção "Bolos".

```
label[for=opcao-bolos] {
    background-image: url(imagens/icone-bolos.svg);

}
```

Faremos o mesmo com o ícone da opção "Bebidas".

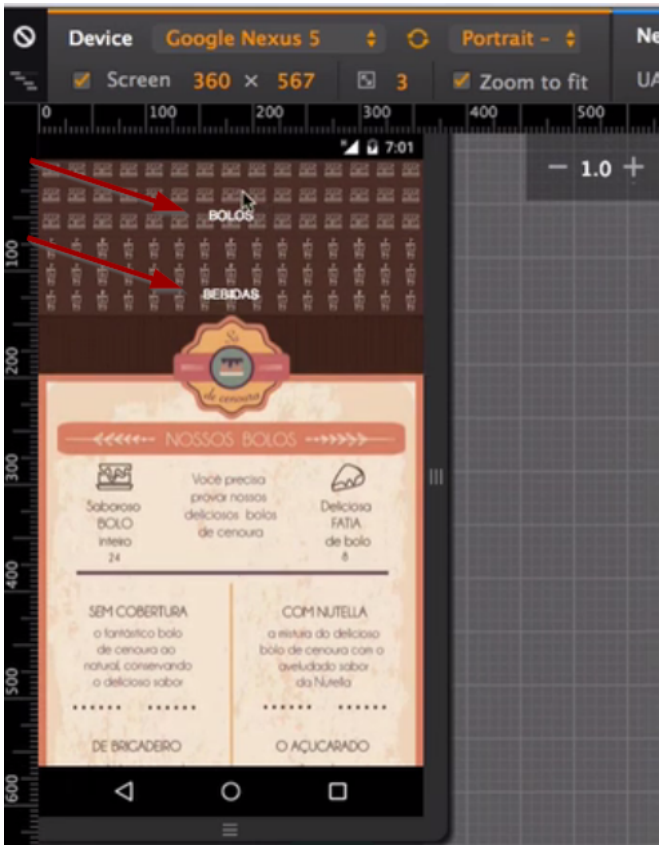
```
label[for=opcao-bebidas] {
    background-image: url(imagens/icone-bebidas.svg);

}
```

O código ficará assim:

```
label {
    background-color: #563429
    color: white;
    display: block;
    font-size: 75%;
    padding: 4em 0 1em;
    text-transform: uppercase;
}
    label[for=opcao-bolos] {
    background-image: url(imagens/icone-bolos.svg);
    }
    label[for=opcao-bebidas] {
```

```
    background-image: url(imagens/icone-bebidas.svg);
    }
```

No navegador, poderemos ver as mudanças.



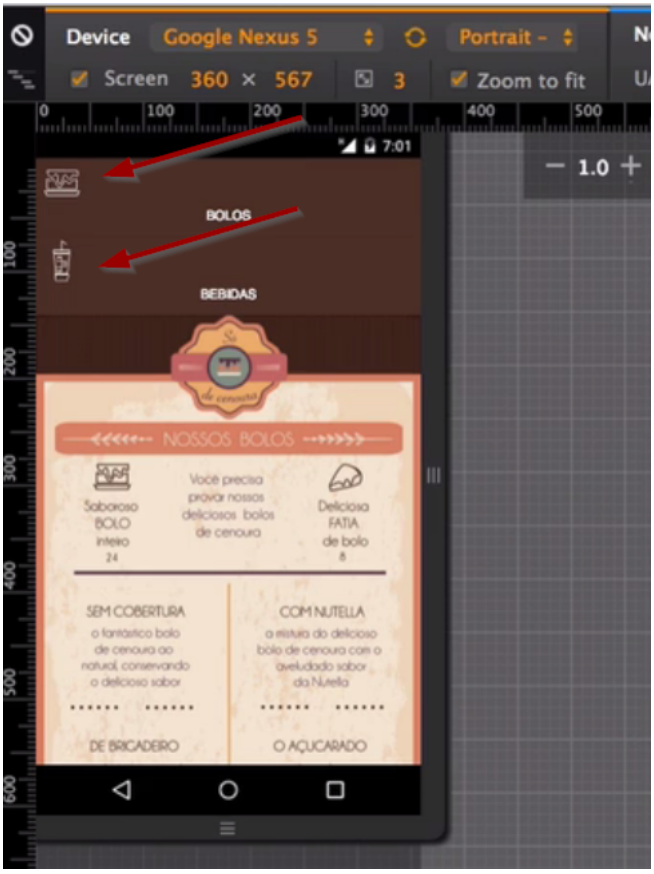
As imagens ficaram repetidas. O ícone ainda precisa de ajustes. Iremos trocar o `background-size` para `4em`.

```
label {
    background-color: #563429
    background-size: 4em;
    color: white;
    display: block;
    font-size: 75%;
    padding: 4em 0 1em;
    text-transform: uppercase;
```

O ícone já irá aumentar de tamanho. Porém, não queremos que ele repita (`no-repeat`).

```
background-repeat: no-repeat;
```

Agora ele aparecerá apenas uma vez em cada `label`.

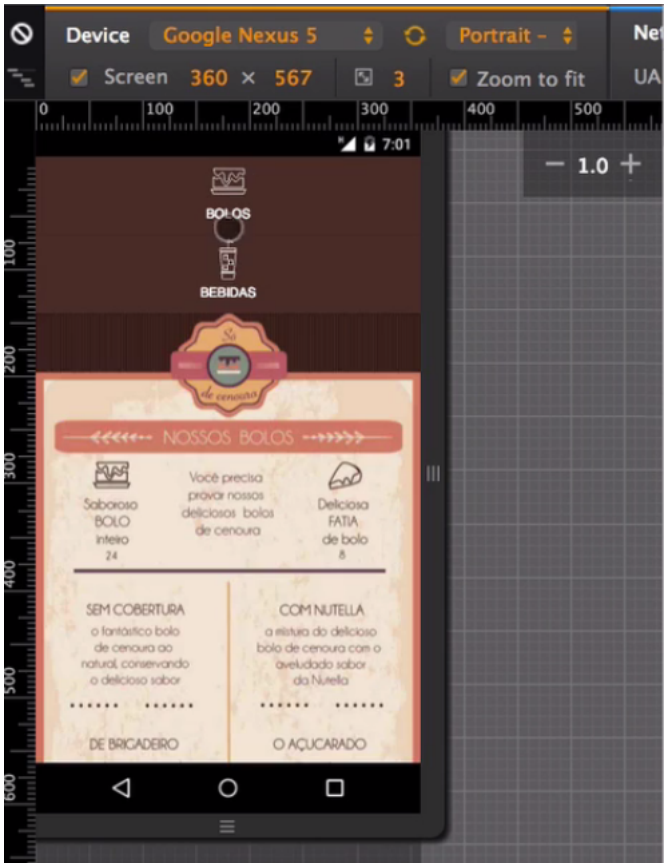


Em seguida, iremos deixar o ícone mais centralizado. Para isto usaremos o `background-position` .

```
background-position: center top;
```

Se quisermos, temos a opção de juntarmos todos os *backgrounds* em uma única propriedade.

Agora os ícones estão centralizados no aplicativo.



Iremos agora indicar qual ícone está selecionado, alterando a cor do `label`. Como saberemos se ele está selecionado? No HTML, o `radio button` posicionado antes do `label` estará marcado com um `checked`.

```
<input type=" radio" name="opcao" id="opcao-bolos" checked>
<label for="opcao-bolos">Bolos</label>
```

No outro caso, o `radio button` posicionado antes do `label` da opção de bebidas deverá ter um `checked`.

Então o que precisamos fazer é pintar o `label` cujo o `radio button` anterior esteja marcado com `checked`.

Como faremos isto? Na aba CSS, iremos adicionar o seletor `radio button`.

```
input[type=radio] label
```

Usaremos também um seletor de irmãos diferente do que utilizamos anteriormente. O seletor `+` significa **imediatamente adjacente**.

```
input[type=radio] + label {
    }
```

Ou seja, só iremos pintar o label que estiver logo na sequência do `input`. Se for o `input` da opção de bolos, usaremos o `label` relacionado com ele. Se for o `input` da opção de bebidas, usaremos o `label` relacionado com bebidas.

```
<input type=" radio" name="opcao" id="opcao-bebidas">
<label for="opcao-bebidas">Bebidas</label>
```

De volta ao CSS, incluiremos um `background-color` com uma cor diferenciada.

```
input[type=radio] + label {
    background-color: #E4876D
}
```

Mas queremos que a cor seja alterada no `input radio` que esteja atualmente selecionada.

```
input[type=radio]:checked + label {
    background-color: #E4876D
}
```

Se não acrescentarmos o `checked` , os dois `radios` ficarão pintados.

O `label` selecionado terá uma cor diferente.

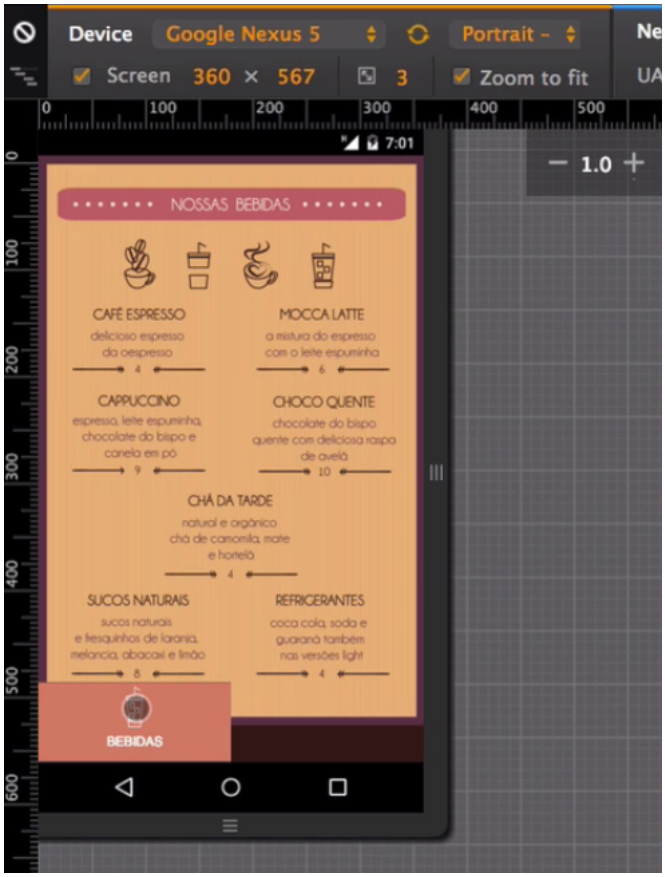
Agora, queremos mover os labels para baixo e deixá-los lado a lado.

Vou especificar a posição do menu, com definindo que cada um ocupe uma largura de 50%, assim eles dividirão a tela, que fiquem na parte debaixo, usaremos o `bottom` igual a 0. Também quero que ele apareça sobre as imagens do cardápio, por isso, usaremos o `z-index` igual a 1.

```
    input[type=radio]:checked + label {
        background-color: #E4876D
    }

label {
```

```
    width: 50%;

    position: fixed;
    bottom: 0;
    z-index: 1;
}
```

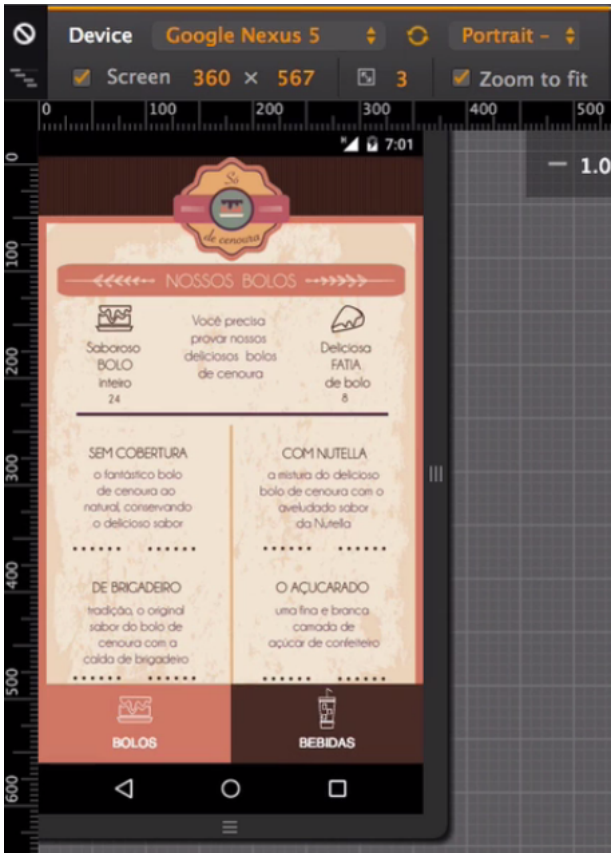
Se atualizarmos o navegador, veremos que o menu mudou de posição. Porém, os dois ícones estão um sobre o outro.



Precisamos posicioná-los corretamente. Faltou especificar no código que um precisa ficar à esquerda (`left`) e o outro à direita (`right`).

```
label[for=opcao-bolos] {
    left:0;

}

label[for=opcao-bebidas] {
    right: 0;
}
```

Agora, os botões "Bolos" e "Bebidas" estão lado a lado.



Está é o nosso primeiro app. Ele não é sofisticado, mas nosso objetivo é mostrar que qualquer HTML, CSS e, eventualmente, Java Script, pode criar um app mobile. Em seguida, veremos como fazer o app rodar em um dispositivo.