04

Adicionando componentes do Ionic e refatorando com Angular

Adicionando componentes do Ionic e refatorando com Angular

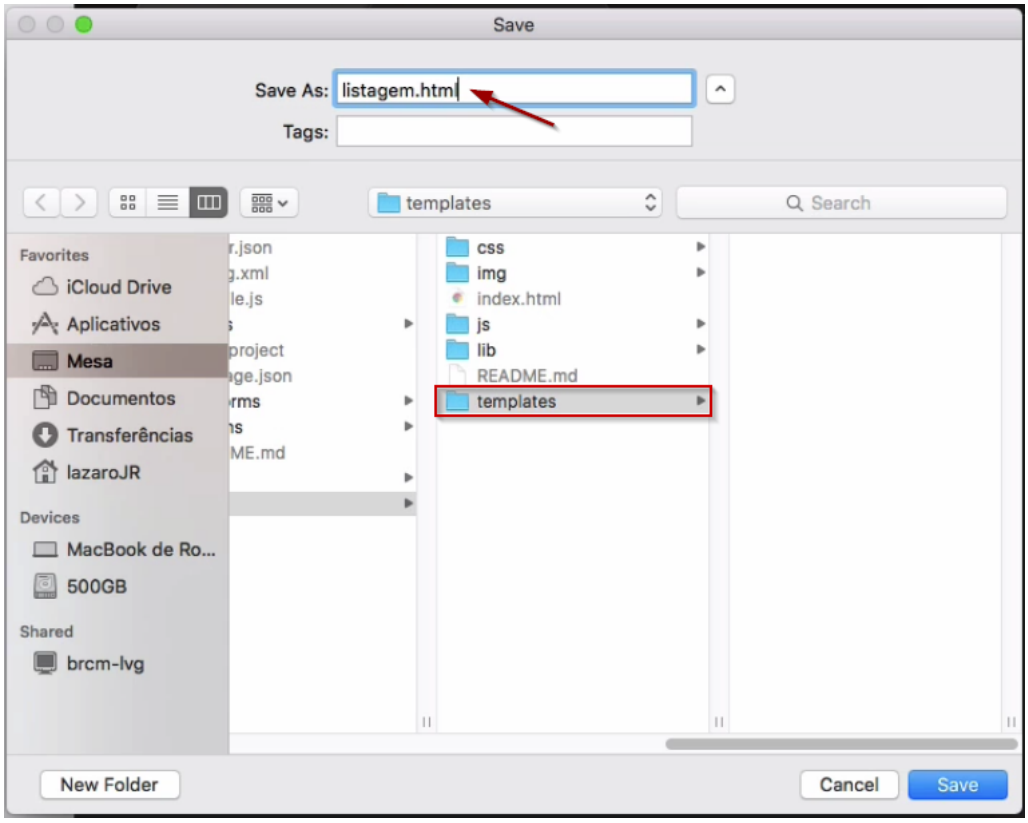
Vamos continuar adicionando componentes no Angular. Nós já temos nossa lista de carros. Agora, queremos que ao clicar no modelo de um carro, o usuário possa ir para uma segunda *View*, onde seja possível adicionar vários os acessórios disponibilizados pela concessionária.

A nossa aplicação já possui o comportamento de ir de uma tela para outra?

Se observarmos o nosso `index.html` não encontraremos no código nenhuma tag referente a este comportamento. Podemos usar este arquivo como uma "casca" que irá renderizar as *Views*, e cada uma destas será a responsável pelo conteúdo contido dentro delas.

Por exemplo, teremos um lista dos modelos de carros e outra com os acessórios, precisamos separar estes dois conteúdos e encontrar uma tecnologia que nos permita ir de uma tela para outra, com um clique. Como podemos fazer isto? Primeiramente, refatorando o nosso código.

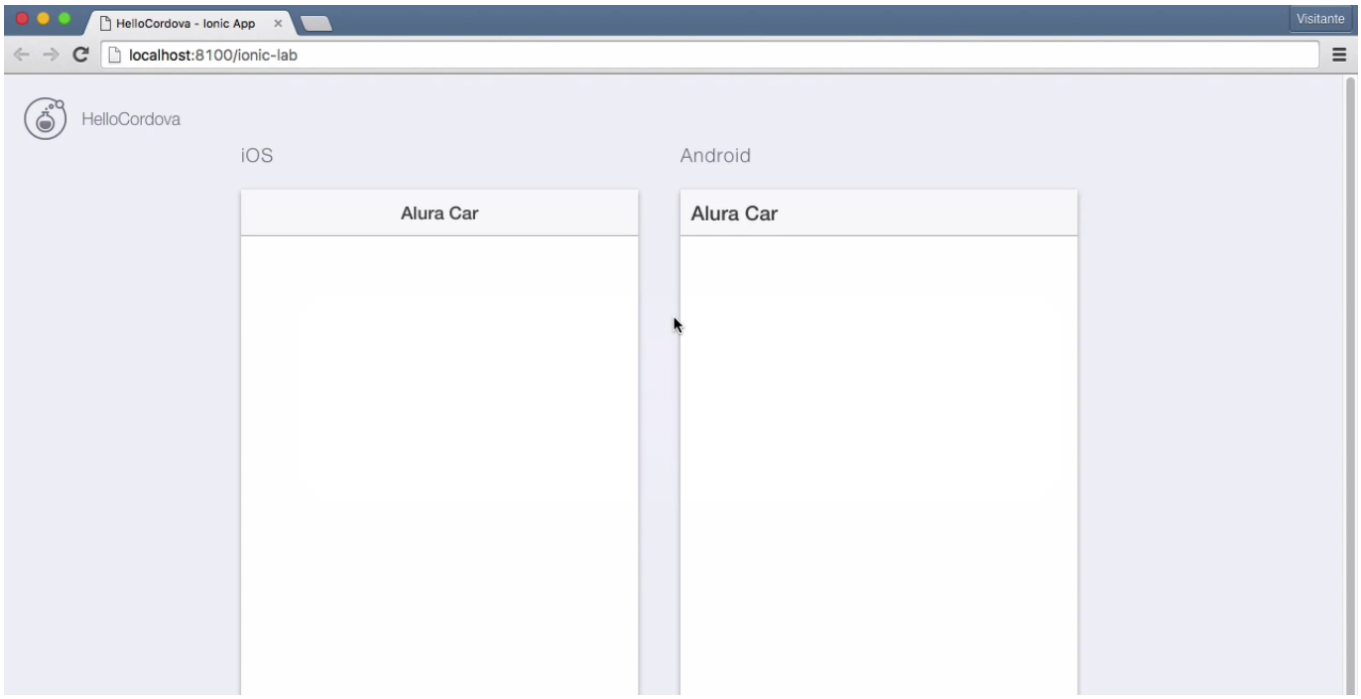
Nós tiraremos a lista do contéudo do `index` e separá-lo a parte. O que podemos fazer sobre isto? Criaremos uma lista nova. Dentro da pasta `www` podemos gerar uma nova, na qual colocaremos todas as *Views*. Uma boa prática para saber onde estão todas elas. Criaremos a nova pasta, clicando em `www` e depois, selecionando "new Folder". Iremos nomeá-la como "templates". Geraremos um novo arquivo que será salvo dentro de 'template'.



Salvamos o arquivo `listagem.html` , no qual iremos guardar o conteúdo da nossa *View*. Recortaremos tudo o que está dentro da tag `<ion-content>` e moveremos para o novo arquivo.

```
<ion-content>
          <ion-list>
            <ion-item ng-repeat="carro in listaDeCarros">
              {{carro}}
            </ion-list>
</ion-content>
```

Se conferirmos o resultado no browser, veremos que a lista ficou vazia.



Por que não conseguimos ver os itens da lista? Porque não informamos ao Ionic sobre a alteração feita e não fizemos nenhuma referência para o arquivo de listagem. Não faria sentido o Ionic carregar algo que ele não sabe onde está. Iremos informá-lo como chegar até à listagem. Para isto, criaremos um arquivo de rotas, na pasta `js`. Como é um arquivo JavaScript, é uma boa prática deixá-lo agrupado com os outros `js`. Vamos chamá-lo de `routes.js`.

Em seguida, criaremos o modo de criação de rotas:

```
angular.module('starter')
.config(function($stateProvider, $urlRouterProvider){

})
```

Observe que injetamos o `$stateProvider` e `$urlRouterProvider` que irá fazer as rotas.

Em seguida, para adicionar a rota, usaremos o `$stateProvider`,

```
$stateProvider

.state('listagem'){
      url : '/listagem',
      templateUrl : 'template/listagem.html',
      controller: 'ListagemController'
}
```

O primeiro parâmetro para `state` é a `listagem` e o segundo será o objeto com suas opções. Por fim, o `templateUrl` e o `cotroller`.

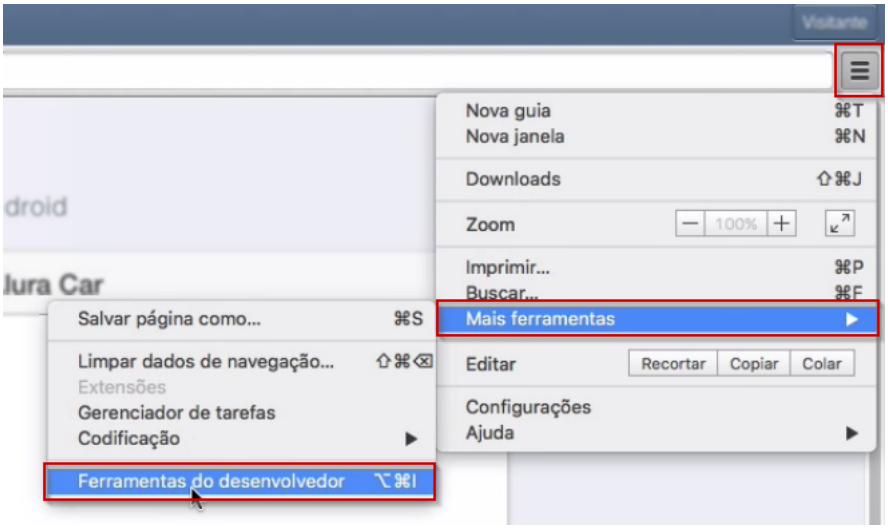
Agora, precisamos customizar uma rota padrão. Então, chamaremos a `urlRouterProvider` e passaremos o `otherwise()` para a listagem que acabamos de criar.

```
$urlRouterProvider.otherwise('listagem');
```

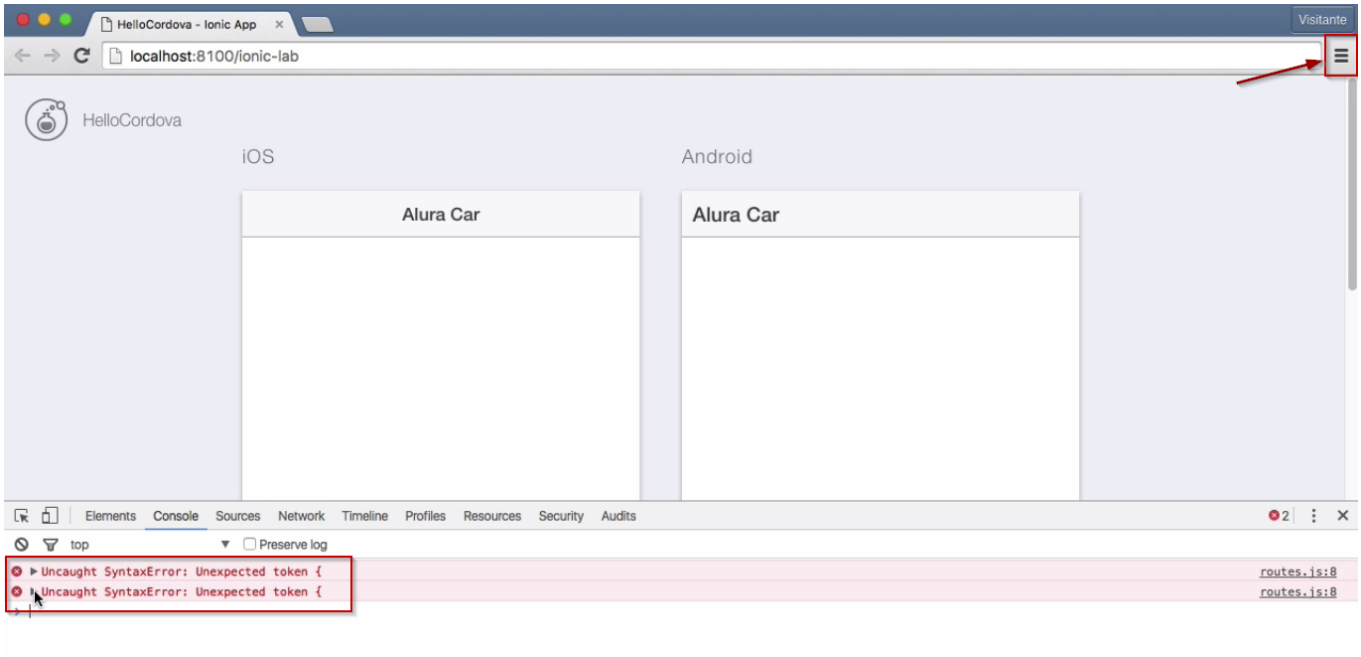
Como se trata de um arquivo JS, ele será carregado na primeira página da aplicação, no caso `index.html`. Nele, já estamos carregando o `app.js` e o `controllers.js`. Iremos adicionar o `routes.js`.

```
<!-- your app´s js -->
<script src="js/app.js"></script>
<script src="js/controllers.js"></script>
<script src="js/routes.js"></script>
```

Já indicamos para o Ionic como chegar até a nossa lista, mas se tentarmos visualizá-la no navegador, veremos que não aparecem os itens ainda. Vamos usar as ferramentas do Chrome Desenvolvedor para entender o que falta fazer. Para isto, vamos clicar no *SplitView* (também conhecido como **menu Hambúrguer**).



Na aba "Console", poderemos ver que existe um erro com o `token`, no arquivo `routes.js`.



Agora, sabemos onde está o erro. Esta é uma das vantagens de desenvolver na web.

No arquivo JS, a linha apontado com problema é a seguinte:

```
.state('listagem') {
}
```

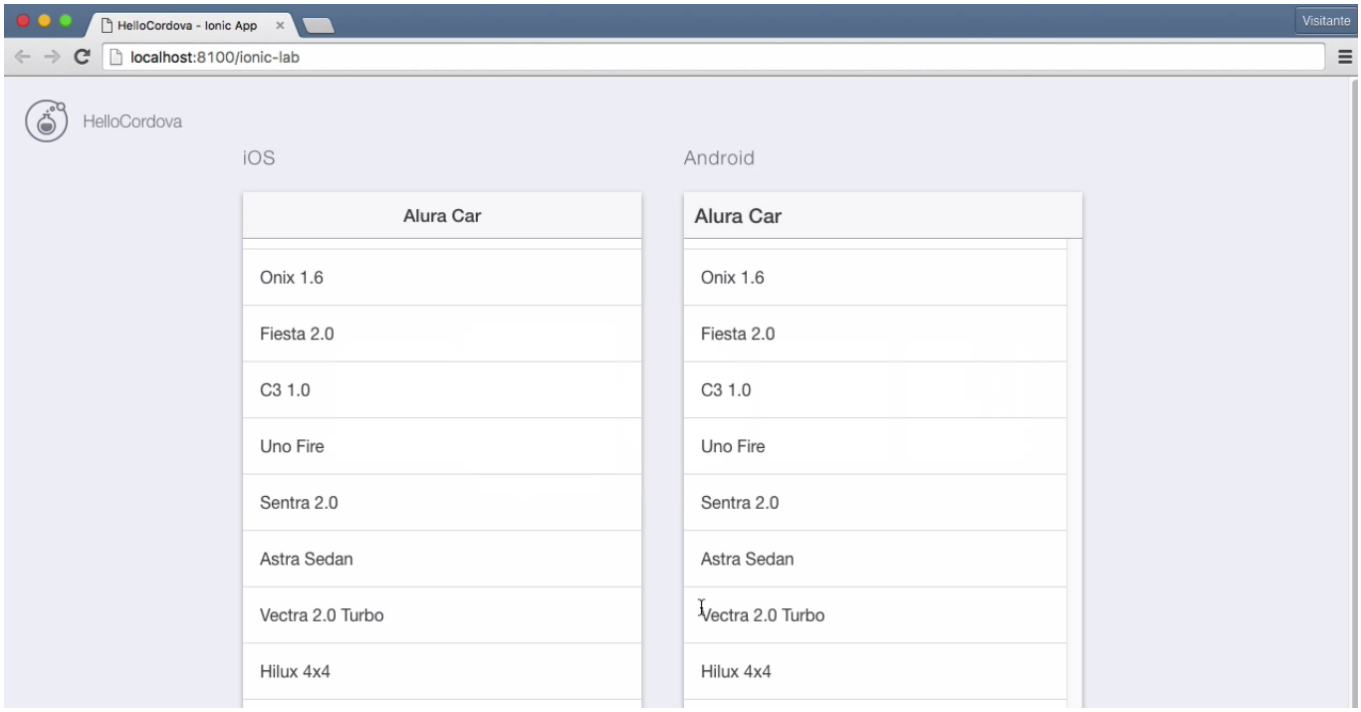
Nosso erro, foi ter fechado o parenteses, apesar de ainda vir o segundo parâmetro. Após removê-lo, o código ficará assim:

```
.state('listagem',{
      url : '/listagem',
      templateUrl : 'template/listagem.html',
      controller: 'ListagemController'
});
```

Fechamos o parenteses mais abaixo. Mas ainda não temos uma lista para ser carregada.

Abaixo da tag `<ion-header-bar>` vamos adiocionar a seguinte: .

```
<body ng-app="starter" ng-controller="ListagemController">
  <ion-pane>
    <ion-header-bar class="bar-stable">
      <h1 class="title">Alura Car</h1>
    </ion-header-bar>
    <ion-nav-view></ion-nav-view>
```



Agora, resolvemos o problema. A parte do `index.html`, ele irá injetar a rota que nós construímos: `listagem.html`. Porém, se você observar a nossa lista está começando com o Onix 1.6. Isto aconteceu porque no arquivo `listagem.html`, nós temos o conteúdo. No entanto, não informamos para o Ionic que o html é uma `View`. Para que isto aconteça, precisamos usar a tag `<ion-view>`.

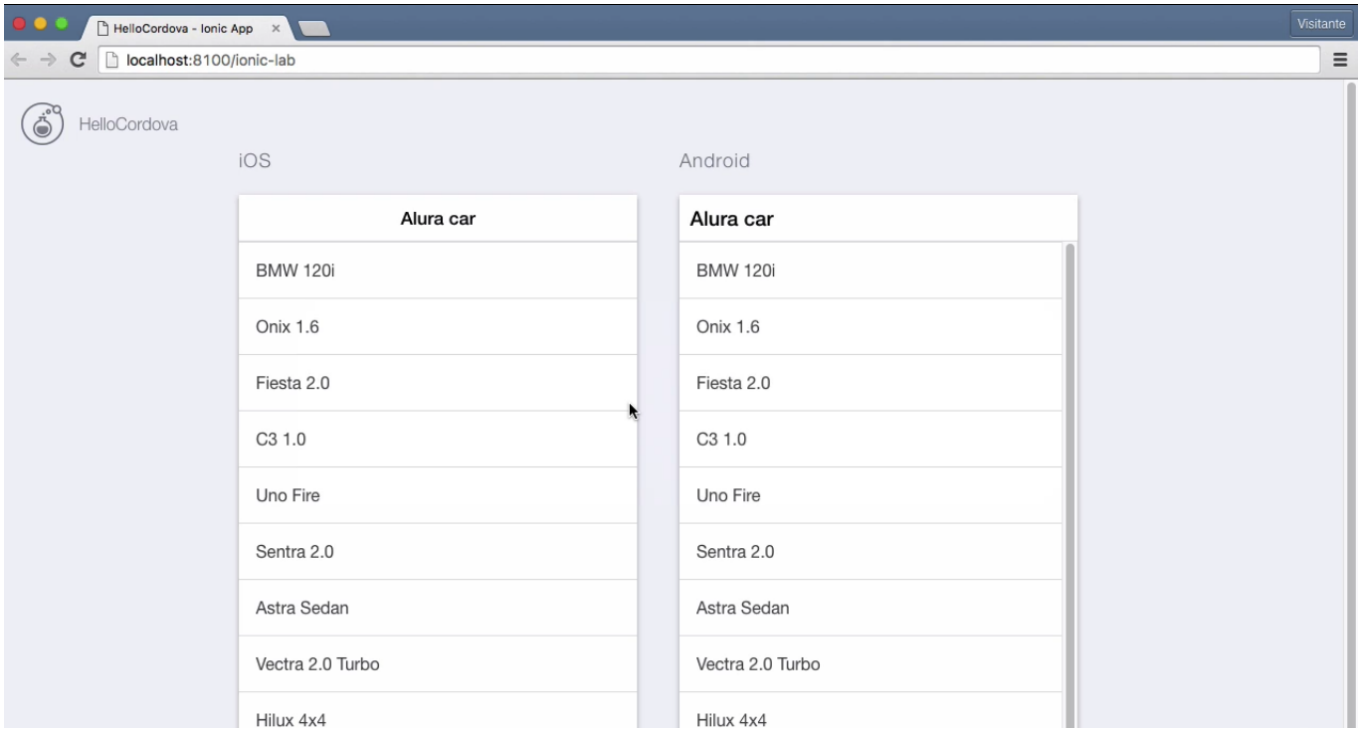
```
<ion-view>
    <ion-content>
        <ion-list>
          <ion-item ng-repeat="carro in listaDeCarros">
            {{carro}}
        </ion-list>
    </ion-content>
</ion-view>
```

Agora, informamos ao Ionic que temos uma View.

A lista não está mais sendo cortada. Mas falta o título da *View*, iremos adicioná-lo no código com a tag `<ion-nav-title>`.

```
<ion-view>
    <ion-nav-title>Alura car</ion-nav-title>
    <ion-content>
        <ion-list>
          <ion-item ng-repeat="carro in listaDeCarros">
            {{carro}}
        </ion-list>
    </ion-content>
</ion-view>
```

Conseguimos o resultado desejado da nossa *View*.



A *View* foi renderizada pelo arquivo `Listagem.html` .