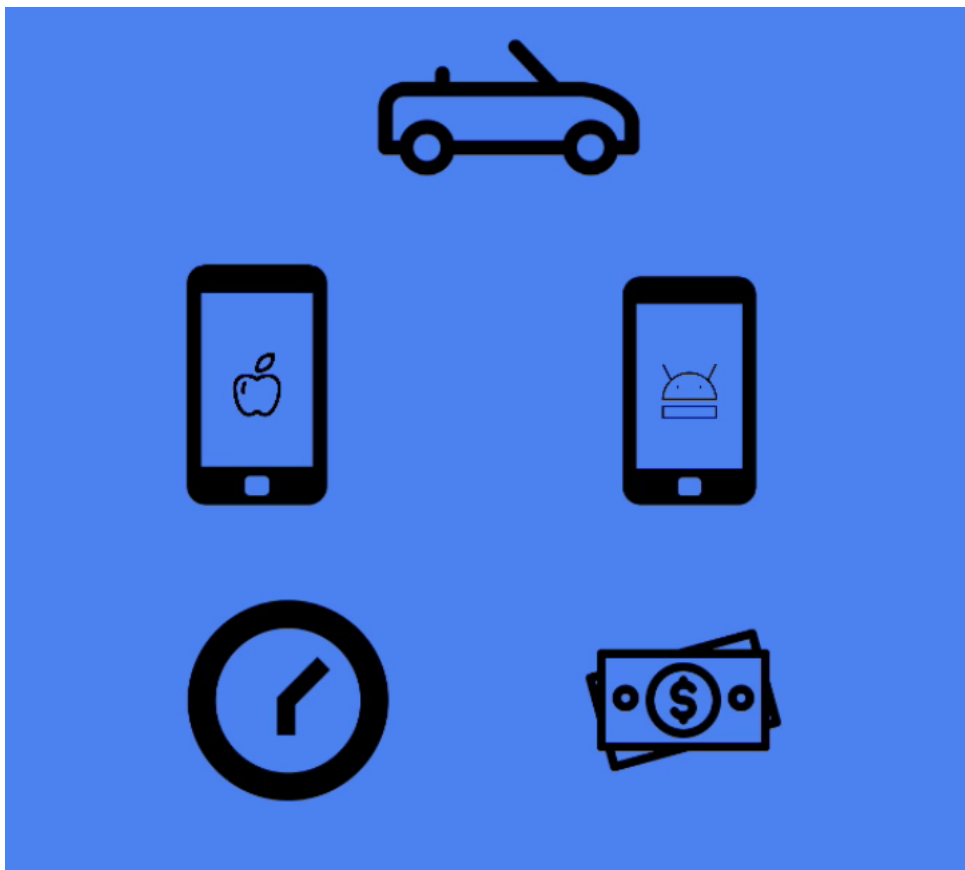
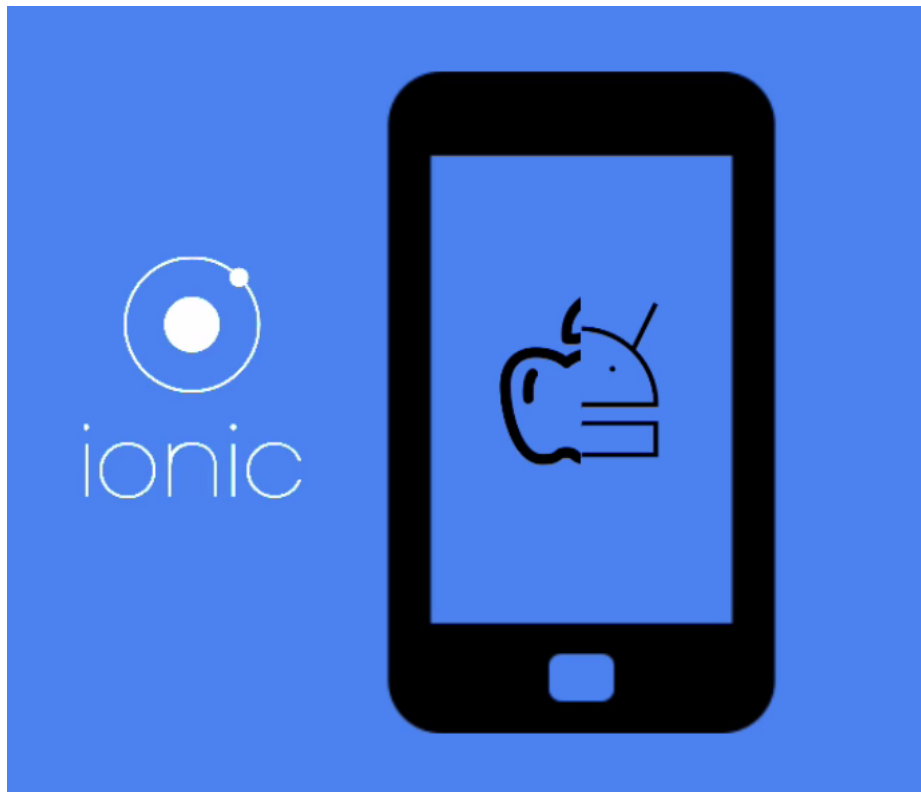


Criando a aplicação no Ionic

Meu nome é Lázaro e sou desenvolvedor na Alura. O nosso gerente de projeto chegou com um trabalho de um cliente para resolvermos. Ele é dono de uma concessionária e precisa de software mobile, que funcione tanto em Android como iOS. Mas temos que lidar com algumas questões logo no início do projeto: O primeiro é o **tempo** precisa ser curto e o segundo, são os **custos**.



Precisaremos encontrar uma solução. Uma delas é criarmos uma aplicação **híbrida**. Desta forma, iremos diminuir o tempo de desenvolvimento, ao fazer duas aplicações ao mesmo tempo sem perder a qualidade, além de conseguir cortar custos em 50%. Para desenvolvê-la, usaremos o **Ionic**.



Vamos colocar a mão na massa!

Começaremos abrindo o terminal, e vamos entrar na pasta destinada para a construção do projeto. Dentro dela, usaremos o comando `ls` para ser listado os itens. Veremos que a pasta está vazia.

```
Terminal  Shell  Editar  Visualizar  Janela  Ajuda
codigo — -bash — 80x24
Last login: Thu Jun 16 10:47:15 on ttys000
Alura-Verde:~ alura$ cd Documents/lazaro/curso-ionic/codigo/
Alura-Verde:codigo alura$ ls
Alura-Verde:codigo alura$
```

Limparemos tudo até aqui com o `clear`. Em seguida, iniciaremos a aplicação. Vamos chamar o `ionic`, que já está instalado na máquina. Caso você ainda não tenha instalado, poderá conferir como fazer a instalação [aqui](https://cursos.alura.com.br/course/ionic/section/1/task/5) (<https://cursos.alura.com.br/course/ionic/section/1/task/5>) ou nesse [vídeo](https://cursos.alura.com.br/course/ionic/section/1/task/4) (<https://cursos.alura.com.br/course/ionic/section/1/task/4>).

O idioma que o Ionic trabalha é em inglês, então, para iniciarmos a operação, usaremos o comando `start`. Também daremos um nome para a nossa aplicação, que chamaremos de `AluraCar`. E queremos usar um template em branco (`blank`).

```
Alura-Verde:codigo alura$ ionic start AluraCar blank
```

Se clicarmos em "enter", ele irá pedir para criarmos uma conta no ionic.io (<http://ionic.io>), para podermos usar o "Push Notifications". Mas neste momento, isto não é importante, então escreveremos `n` que significará **não**. Depois, ele irá continuar a instalação.

```
codigo — -bash — 80x24

To test your app on a device easily, try Ionic View:
http://view.ionic.io

Create an ionic.io account to send Push Notifications and use the Ionic View app
?
(Y/n): n

+-----+
+ Extra! Extra! Fresh Ionic updates for June 2016
+
+ Ionic 2 Beta is out! Try the next generation of Ionic
+ http://ionicframework.com/docs/v2/getting-started/installation/
+
+ Test and share your Ionic apps easily with Ionic View
+ http://view.ionic.io
+
+ Building enterprise apps? Ionic Enterprise Framework comes with the support and features you need
+ http://ionic.io/enterprise
+
+-----+
```

Em seguida, vamos limpar o console novamente com o comando `clear`.

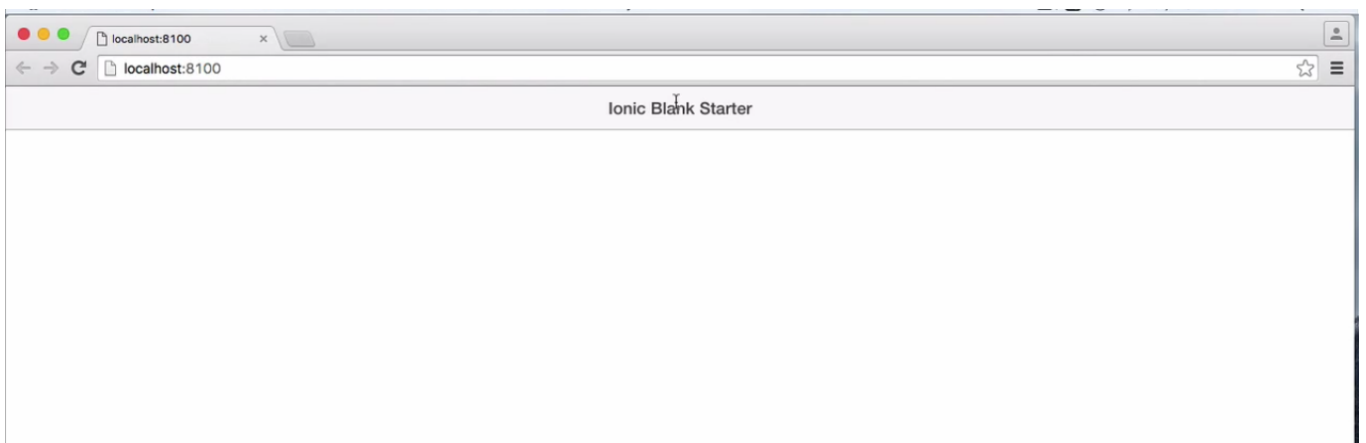
Mais adiante, pediremos para ele listar o que temos dentro da pasta e ele irá criar vários arquivos.

```
Alura-Verde:AluraCar alura$ ls
bower.json      gulpfile.js    ionic.project  platforms      css
config.xml     hooks          package.json   plugins        www
Alura-Verde:AluraCar alura$
```

Uma das principais vantagens oferecidas pelo Ionic é a possibilidade de testarmos a aplicação. Com o comando `ionic serve`, o Ionic subirá o servidor e que permitirá que ela já seja testada se quisermos. Não precisamos emular um sistema operacional como o Android ou iOS. Agora, temos a opção de emular dentro do browser. Nós podemos subir um servidor web e ver com está ficando a aplicação. É o que faremos:

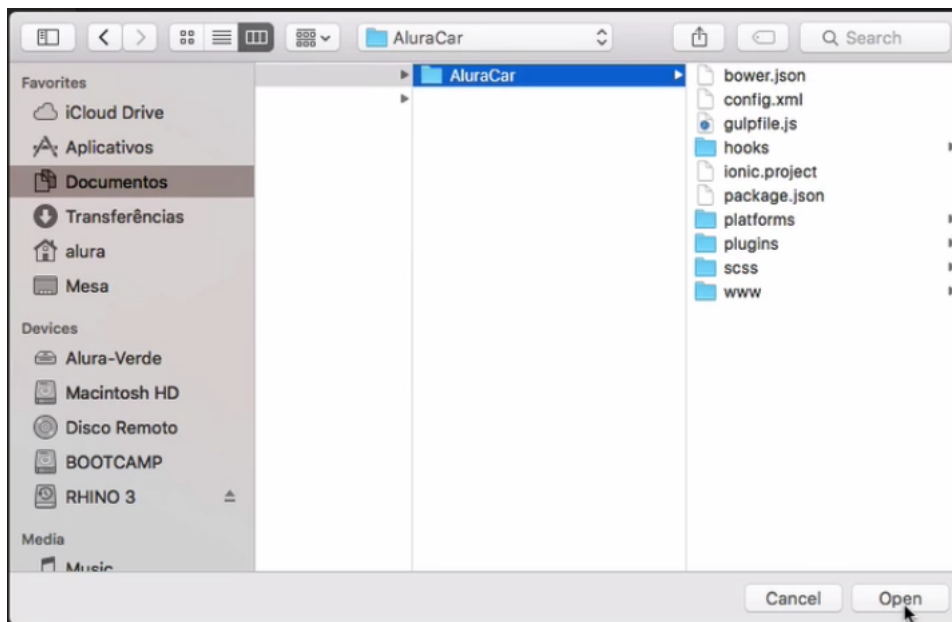
```
Alura-Verde:AluraCar alura$ ionic serve
```

Pronto! Com isso, ele já subiu o servidor na porta `8100` e já estamos usando a aplicação.

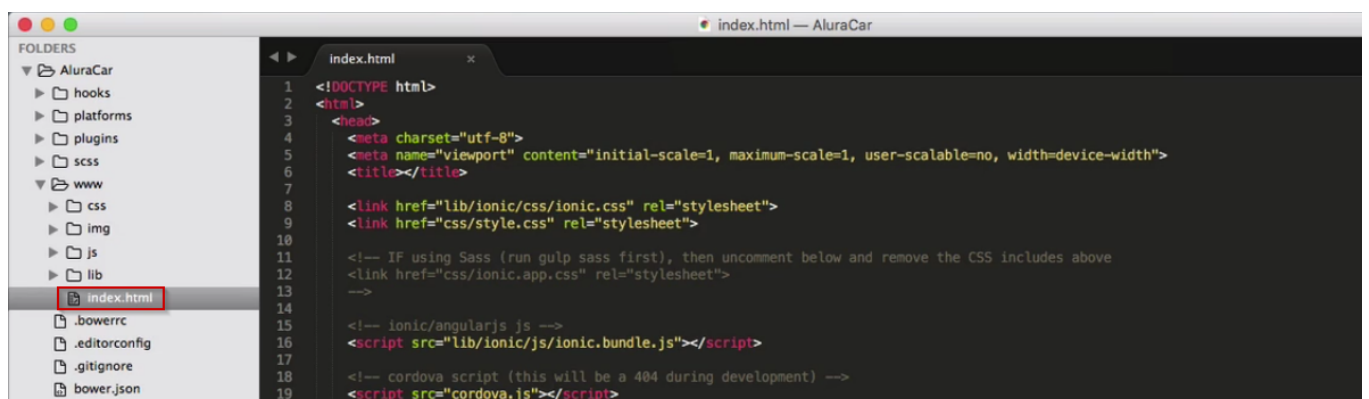


No curso, iremos construir algo, salvar e já ver acontecendo...

No terminal, veremos que o Ionic está rodando. Em seguida, abriremos o editor de texto Sublime para criarmos o código da aplicação. Você tem a opção de usar outros editores de texto. Vamos abrir o projeto AluraCar .



A pasta `www` será importante, porque nela ficará o código do nosso aplicativo. Nela encontraremos o arquivo `index.html` .



\...

```
<body ng-app="starter">
<ion-pane>
  <ion-header-bar class="bar-stable">
    <h1 class="title">Ionic Blank Starter</h1>
  </ion-header-bar>
  <ion-content>

    </ion-content>
</ion-pane>
```

```
</body>
</html>
```

O texto do cabeçalho da aplicação está localizado ali dentro.

```
<ion-pane>
  <ion-header-bar class="bar-stable">
    <h1 class="title">Ionic Blank Starter</h1>
  </ion-header-bar>
```

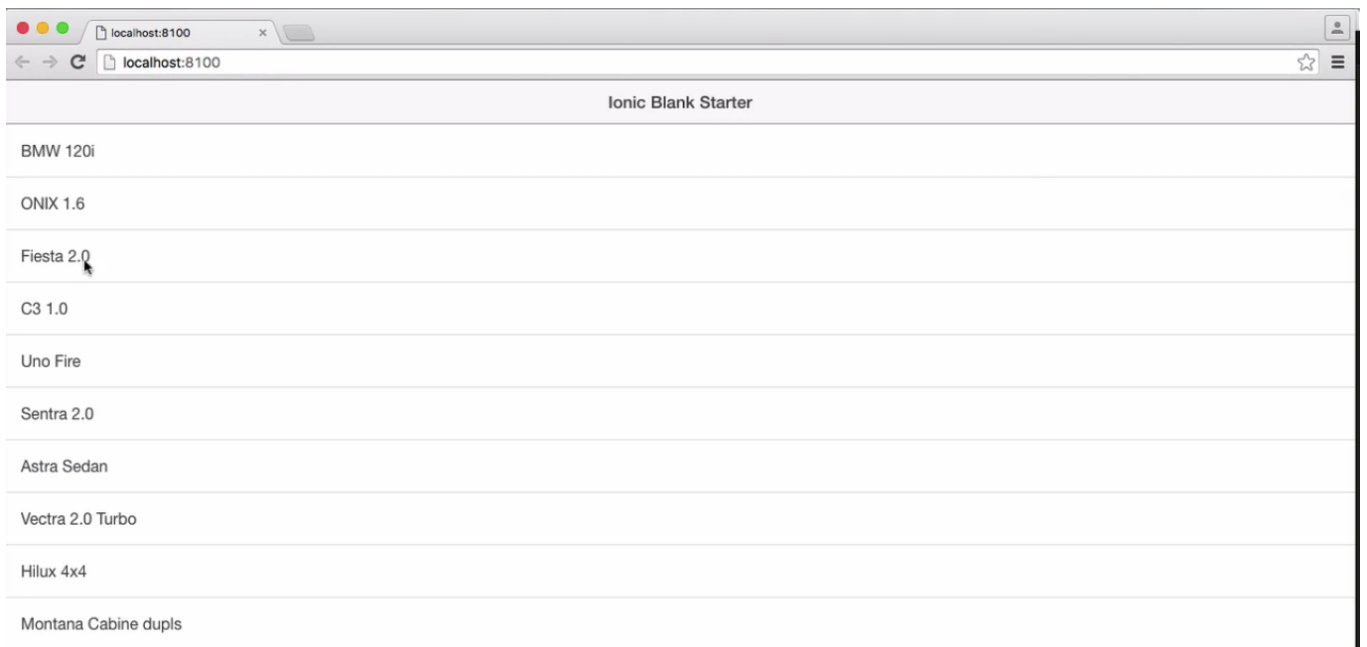
A lista de carros também será inserida neste aplicativo. Buscaremos a tag referente ao conteúdo `<ion-content>`, posicionada logo abaixo do `<ion-header-bar>`. Observe que as tags que utilizaremos são iniciada por `ion`. Para adicionarmos os itens da lista, incluiremos dentro do `<ion-content>`, a nova tag `<ion-list>`. Os itens também ganharão uma tag: `<ion-item>`.

```
\...
<ion-content>
  <ion-list>
    <ion-item> </ion-item>
  </ion-list>
</ion-content>
```

Em seguida, adicionaremos o conteúdo da lista. Para cada item, teremos que usar a tag `<ion-item>`.

```
\...
<ion-content>
  <ion-list>
    <ion-item>BMW 120i</ion-item>
    <ion-item>ONIX 1.6</ion-item>
    <ion-item>Fiesta 2.0</ion-item>
    <ion-item>C3 1.0</ion-item>
    <ion-item>Uno Fire</ion-item>
    <ion-item>Sentra 2.0</ion-item>
    <ion-item>Astra Sedan</ion-item>
    <ion-item>Vectra 2.0 Turbo</ion-item>
    <ion-item>Hilux 4x4</ion-item>
    <ion-item>Montana Cabine dupla</ion-item>
  </ion-list>
</ion-content>
```

Nossa lista já tem dez itens.



A aplicação começa a ganhar forma. Até aqui, começamos a usar muitos elementos do Ionic. Nós criamos uma aplicação do zero, subimos para o servidor. Também adicionamos as tags `<ion-list>` e `<ion-item>`. Criamos, subimos e testamos a aplicação.

A possibilidade de ver os resultados simultaneamente é uma das vantagens do Ionic.