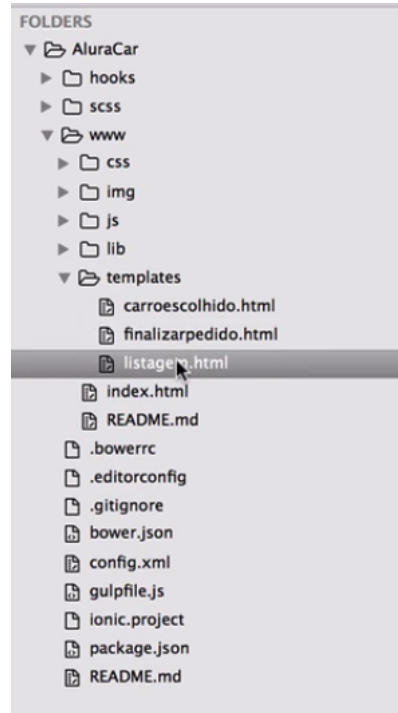


## Criando nossa tela de login

### Transcrição

Olá, pessoal. Estou com o código aberto, lembrando que na pasta `www` estão os arquivos da aplicação.



Na pasta `template`, encontramos as três Views que criamos. A pasta `js`, estão todos os JavaScripts.

Vamos subir a aplicação para podermos visualizá-la. No Terminal, após conferirmos as pastas do arquivo, usaremos o seguinte comando:

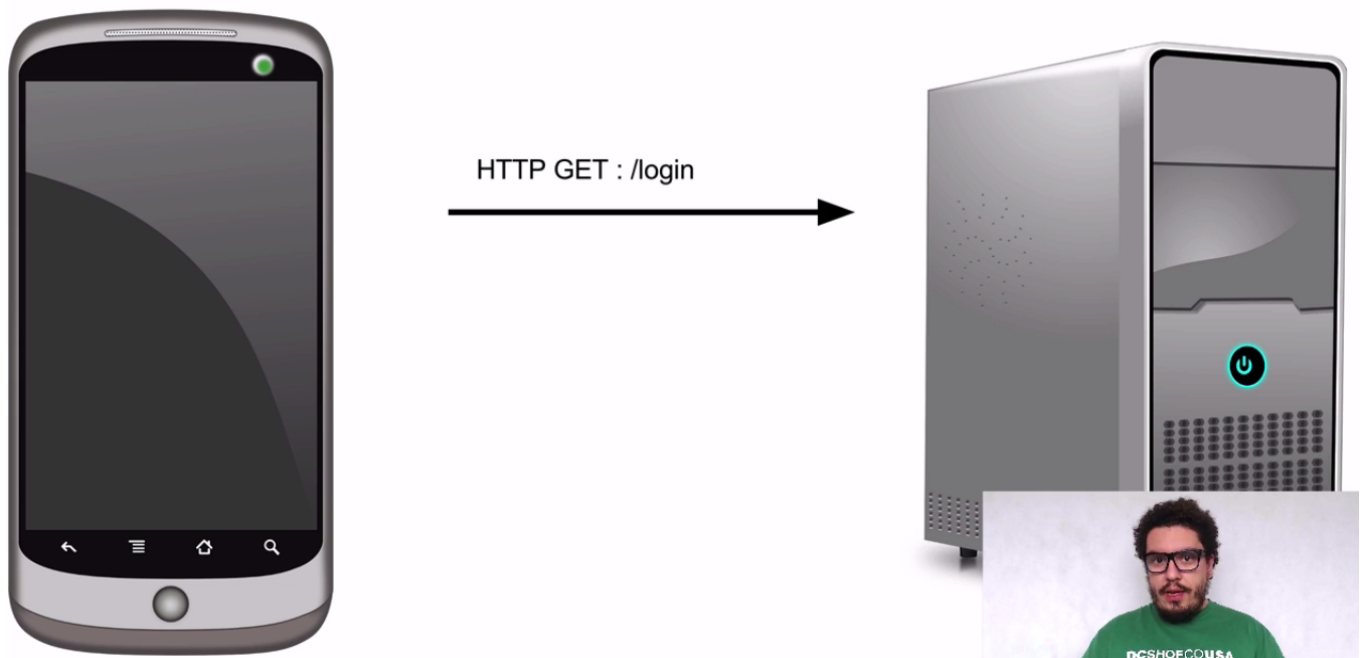
```
AluraCar ionic serve -l
```

Atualmente, ela está assim:



O aplicativo AluraCar está bastante semelhante no Android.

Em seguida, começaremos criando a funcionalidade de login. A arquitetura por trás do login é feito da seguinte forma: temos o nosso telefone (iOS ou Android), fazemos uma requisição para o Back-End do cliente, na rota `/login`. Lembrando, como foi construído esse Back-End para o mobile, não interessa se foi construído com Java, PHP, Ruby, existem vários frameworks que você pode usar para fazer isso.



O importante é que seja feita a requisição do servidor, que deve nos disponibilizar a rota, e que após a realização do processo, seja enviado um "OK" de retorno - ou não. Para nós, o servidor não nos interessa. O que efetivamente é importante é enviarmos os dados e retornar para fazer o login - ou não. A regra de negócio do login, se a senha ou o usuários estão corretos não ficarão no mobile. Ficará na API construída pelo cliente.

Voltando para nossa aplicação. O que vamos construir primeiro, a regra de negócio ou vamos construir a tela? Tanto faz, mas eu prefiro começar pela tela, para já termos algo visual, e depois construiremos a regra de negócio.

Criaremos um novo arquivo na pasta `templates` que receberá o nome de `login.html` e precisaremos adicionar a estrutura de uma View de uma tela do Ionic. Começaremos a estrutura com a tag `<ion-view>` e dentro, teremos o `<ion-content>`.

```
<ion-view>
  <ion-header-bar>
  </ion-header-bar>
  <ion-content>

</ion-content>
</ion-view>
```

Com a tag `<ion-header-bar>` incluímos o cabeçalho. Dentro dela, em seguida colocaremos um HTML.

```
<ion-view>
  <ion-header-bar>
    <h1>Olá, faça seu login</h1>
  </ion-header-bar>
  <ion-content>
    <form>

    </form>
  </ion-content>
</ion-view>
```

Começaremos a criar o formulário adicionando o `<form>`. Na [documentação do Ionic](http://ionicframework.com/docs/components/#inset-forms) (<http://ionicframework.com/docs/components/#inset-forms>) podemos ver um layout interessante para usar no nosso formulário:

The screenshot shows the Ionic Framework documentation page for "Inset Forms". The page title is "Inset Forms". The text explains that by default each input item will fill 100% of the width of its parent element (the list). However, you can inset the list using either the `list list-inset` or `card` classnames. The `card` classname applies a lower box shadow while `list-inset` does not. Additionally, if the list's parent element has `padding` assigned then this will also give the form an inset appearance.

```
<div class="list list-inset">
  <label class="item item-input">
    <input type="text" placeholder="First Name">
  </label>
  <label class="item item-input">
    <input type="text" placeholder="Last Name">
  </label>
</div>
```

On the right side of the page, there is a mobile app preview showing a login form titled "Inset Forms". The form has two input fields: "First Name" and "Last Name", both with placeholder text. Below the input fields is a blue "Submit" button.

Podemos pedir que o usuário preencha com seu "Email" e "Senha". Adicionaremos as tags `<div>` e `<label>` para cada input.

```
<ion-view>
  <ion-header-bar>
    <h1>Olá, faça seu login</h1>
  </ion-header-bar>
  <ion-content>
```

```

<form>
  <div>
    <label>
      <input type="" name="">
    </label>
    <label>
      <input type="" name="">
    </label>
  </div>
</form>
</ion-content>
</ion-view>

```

Em seguida, adicionaremos o CSS, dentro das tags <div> :

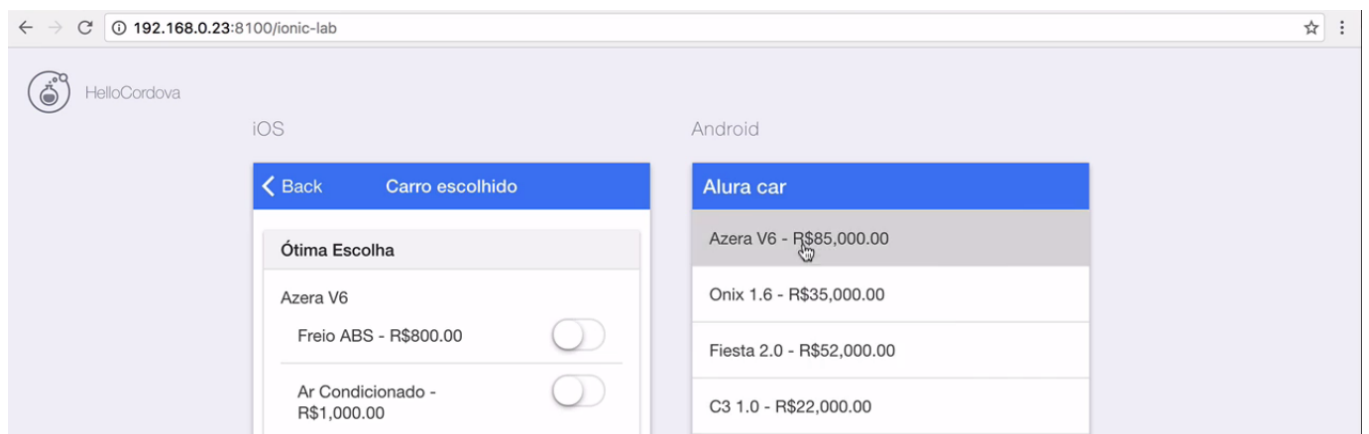
```

<div class="list list-inset">
  <label class="item item-input">
    <input type="text">
  </label>
  <label class="item item-input">
    <input type="text">
  </label>
</div>

```

Observe que usamos a classe `list list-inset` e `item item-input` como foi mostrado na documentação.

Até aqui, já criamos algo e podemos testar.



Nós criamos as telas, mas não criamos as rotas. Faremos isto, no arquivo `routes.js` do nosso aplicativo, localizado dentro da pasta `js`. Logo abaixo da rota `finalizarPedido`, nós adicionaremos um novo `state` :

```

.state('login', {
  url : '/login',
  templateUrl: 'templates/login.html'
})

```

Adicionamos um `state` para `login` e que deve levar para a rota `/login`. O outro atributo é a `controller`, mas não precisamos adicioná-lo.

Falta ainda criar o **botão** para chegar até a rota. Queremos que a primeira rota, seja já o login. Por exemplo, o Facebook, a primeira tela que nós entramos é no login. No nosso app, não será diferente. A primeira tela que queremos que ele carregue é o login - lembrando, que anteriormente era a listagem.

Nós temos a rota, mas caso ninguém a chame, ele cairá no `otherwise` :

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

  $urlRouterProvider.otherwise('listagem');

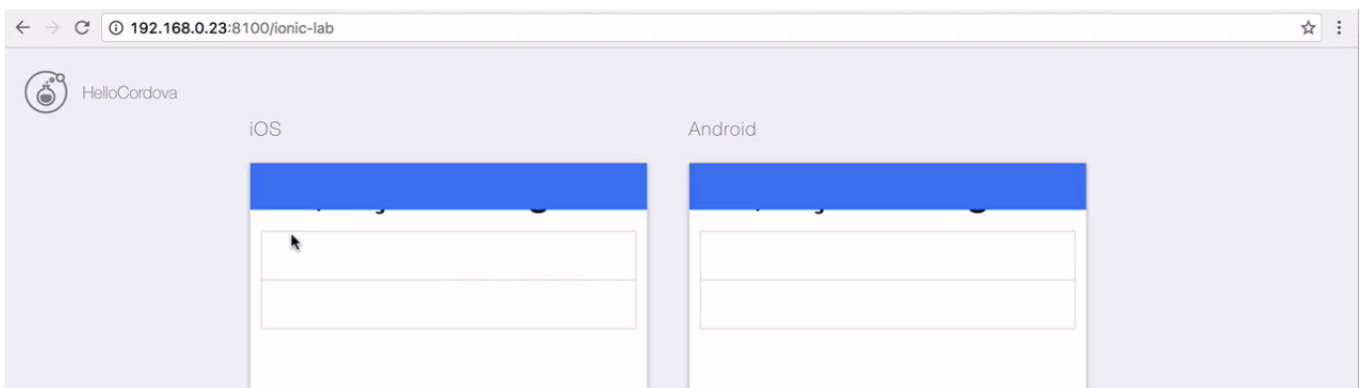
  $stateProvider

  <!-- ... -->
```

Por isso, precisamos alterar a rota padrão de `listagem` para `login` .

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

Já conseguiremos visualizar o que estamos construindo na tela.



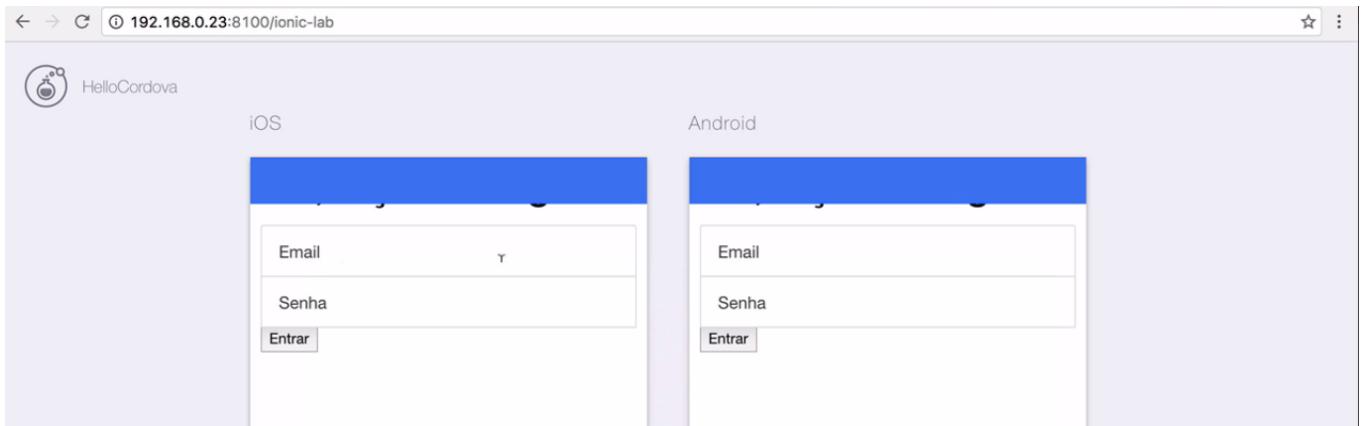
Não foi carregada a listagem e apareceram os `inputs` . Mas ainda não sabemos qual dos dois campos devemos preencher com o email ou com a senha. Precisaremos adicionar o botão, para que os dados sejam enviados.

Em seguida, adicionaremos o `span` referente ao E-mail e à Senha .

```
<div class="list list-inset">
  <label class="item item-input">
    <span class="input-label">E-mail</span>
    <input type="text">
  </label>
  <label class="item item-input">
    <span class="input-label">Senha</span>
    <input type="password">
  </label>
</div>
```

Logo, abaixo da `label` , adicionaremos o `button` .

```
<button>Entrar</button>
```



Já conseguimos visualizar o botão entrar.

Nós ainda vamos arrumar o botão.

```
<button class="button button-block button-positive"> Entrar</button>
```

Até aqui, o nosso código ficou assim:

```
<ion-view>
  <ion-header-bar>
    <h1>Olá, faça seu login</h1>
  </ion-header-bar>
  <ion-content>
    <form>
      <div class="list list-inset">
        <label class="item item-input">
          <span class="input-label">E-mail</span>
          <input type="text">
        </label>
        <label class="item item-input">
          <span class="input-label">Senha</span>
          <input type="password">
        </label>

        <button class="button button-block button-positive"> Entrar</button>
      </div>
    </form>
  </ion-content>
</ion-view>
```

Com a classe `positive`, o botão ficará com a cor azul.

! [botao azul] ([https://s3.amazonaws.com/caelum-online-public/Ionic+2/1.2\\_6\\_botao+azul.png](https://s3.amazonaws.com/caelum-online-public/Ionic+2/1.2_6_botao+azul.png))  
([https://s3.amazonaws.com/caelum-online-public/Ionic+2/1.2\\_6\\_botao+azul.png](https://s3.amazonaws.com/caelum-online-public/Ionic+2/1.2_6_botao+azul.png))

Nós construímos a tela, agora podemos criar a lógica.

Até a próxima!