03

# Fazendo a lógica da tela de login

### Transcrição

Estou com o formulário da aplicação aberto, com o campos de preenchimento e o botão de "Entrar" e então, fazer o Submit do formulário. Depois de fazer toda a regra de negócio - ir no Back-End, verificar se o usuário está logado ou não -, passar ou não para a tela de listagem. Caso ele não tenha logado, exibir para o usuário o aviso de que o e-mail ou a senha estão incorretos. Esta será a lógica que seguiremos.

Então, precisamos submeter o formulário quando clicarmos no botão.

Atualmente, o nosso código está 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>
```

Em seguida, adicionaremos o `type` do botão.

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

E para onde vai o formulário? Podemos fazer o bind com o Angular. No momento de submeter o formulário, eu quero chamar uma função Angular. Nós adicionaremos no `form`, o `ng-submit`. A função receberá o nome `realizarLogin()`. Para que ele consiga pegar os dados na controller, ele precisará fazer o bind entre o escopo da controller e o HTML. Para isto, usaremos no `input` do `E-mail` e da `Senha`, o `ng-model`, que receberá o objeto seguido do atributo.

```
<form ng-submit="realizarLogin()">
  <div class="list list-inset">
    <label class="item item-input">
      <span class="input-label">E-mail</span>
      <input type="text" ng-model="login.email">
```

```
      </label>
      <label class="item item-input">
        <span class="input-label">Senha</span>
        <input type="password" ng-model="login.senha">
      </label>
  <!-- ... -->
```

Quando ele clicar no botão `submit`, a função `realizarLogin()` será chamada.

Antes de seguirmos, vamos tentar melhorar a clareza do código da tela que ficou com muitos componentes, acrescentando comentários.

Adicionaremos nas tags de fechamento de `<div>`, `<form>` e `<ion-content>`.

```
      </div><!-- fim componente formulario -->
    </form><!-- fim do formulario -->
  </ion-content><!-- fim da tag content -->
```

É uma boa prática fazermos isso quando o arquivo HTML começa a ficar grande. Saber o que temos no código facilita a manutenção.

Até aqui, o nosso código está assim:

```
<ion-view>
  <ion-header-bar>
    <h1>Olá, faça seu login</h1>
  </ion-header-bar>
    <ion-content>
    <form ng-submit="realizarLogin()">
      <div class="list list-inset">
        <label class="item item-input">
          <span class="input-label">E-mail</span>
          <input type="text" ng-model="login.email">
        </label>
        <label class="item item-input">
          <span class="input-label">Senha</span>
          <input type="password" ng-model="login.senha">
        </label>

        <button class="button button-block button-positive" type="submit"> Entrar</button>
      </div><!-- fim componente formulario -->
    </form><!-- fim do formulario -->
  </ion-content><!-- fim da tag content -->
</ion-view>
```

Agora, podemos continuar. Vamos chamar a função `realizarLogin()` na nossa controller - mas, antes, precisaremos criar uma. Declararemos a `controller` no arquivo `routes.js`. Dentro do `login`, adicionaremos a `controller`:

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

```
    controller: 'LoginController'
})
```

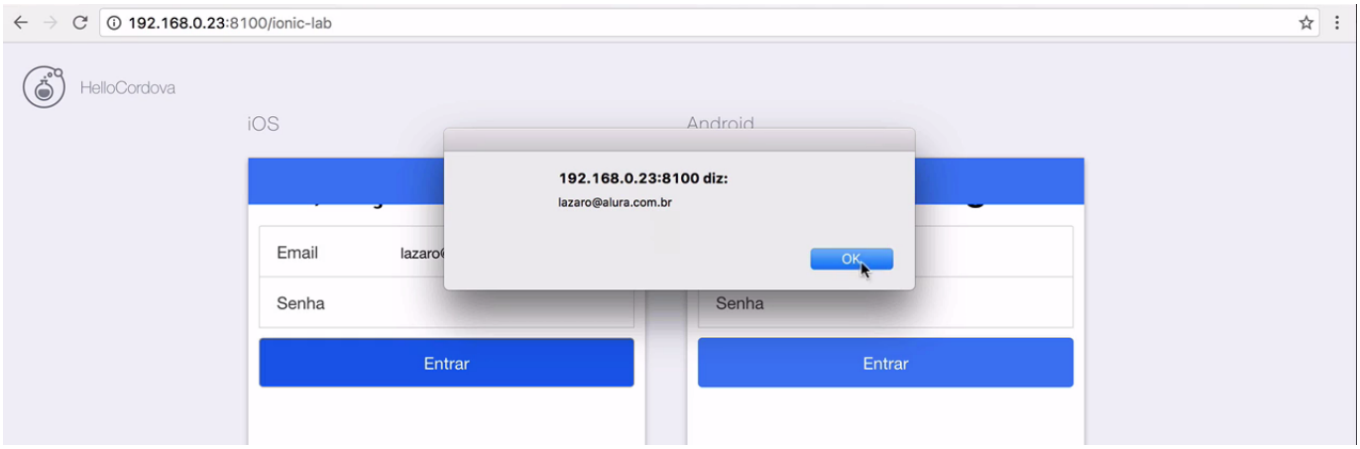
Em seguida, criaremos a `controller` no arquivo `controllers.js`. Nele, já encontraremos o `ListagemController`, `CarroEscolhidoController` e `FinalizarPedidoController`. Após todos eles, adicionaremos outro `module` chamado `stater`. Depois, criaremos a `.controller`.

```
angular.module('starter')
.controller('LoginController', function(){

});
```

Passamos dois parâmetros para a `.controller`. Então, precisaremos receber dois dados da tela. Precisaremos construir o objeto no escopo ( `$scope` ).

```
angular.module('starter')
.controller('LoginController', function($scope, CarroService, $ionicPopup, $state){

    $scope.login = {};

    $scope.realizarLogin = function(){
      alert($scope.logi.email);
  }
});
```

A função `realizarLogin()` receberá uma `function()`. Com o `alert()`, podemos testar se está tudo funcionando corretamente.




Ao preenchermos o formulário com um email, visualizamos uma mensagem.

Continuaremos construindo a função. Após apagar o `alert`, adicionaremos o objeto `dadosDoLogin` e dentro dele, o `params`, usado para fazer o envio para o servidor.

```
angular.module('starter')
.controller('LoginController', function($scope, CarroService, $ionicPopup, $state){

    $scope.login = {};
```

```
    $scope.realizarLogin = function(){

        var dadosDoLogin = {
            params : {
                email : $scope.login.email,
                senha : $scope.login.senha
            }
        }
```

Fizemos o mesmo que com a controller `finalizarPedido`, tem dentro o objeto com vários atributos e enviamos para o nosso serviço.

```
angular.module('starter')
.controller('FinalizarPedidoController', function($scope,CarroService){

    $scope.carroFinalizado = angular.fromJson($stateParams.carro);

    $scope.pedido = {};

    $scope.finalizarPedido = function(){

        var pedidoFinalizado = {
            params : {
                carro : $scope.carroFinalizado.nome,
                preco : $scope.carroFinalizado.preco,
                nome :  $scope.pedido.nome,
                endereco : $scope.pedido.endereco,
                email : $scope.pedido.email
            }
        }
```

No `LoginController`, vamos ter que criar um serviço também no `service.js`. Logo abaixo do `salvarPedido`, adicionaremos o `realizarLogin`

```
realizarLogin : function(dadosDoLogin){
  return $http.get(url + "login", dadosDoLogin).then(function(response){
    return response.data;
  });
}
```

Ele irá retornar uma chamada `$http`. Nós concatenamos a `url` do Back-End ('https://aluracar.herokuapp.com/ (https://aluracar.herokuapp.com/)') com o `login`. Quando fizermos a chamada precisaremos chamar o objeto `dadosDoLogin` - o e-mail e a senha. A resposta será o `response.data`.

Já temos o serviço podemos passá-la agora na controller, por meio da injeção de dependência do Angular. Usamos o `CarroService`, que aparece no alto do arquivo `service.js`, como podemos ver abaixo:

```
angular.module('starter')
.service('CarroService', function($http){

    var url = 'https://aluracar.herokuapp.com/';
```

De volta ao `controller.js`, o `CarroService`, como segundo parâmetro no `function`:

```
angular.module('starter')
.controller('FinalizarPedidoController', function($stateParams, $scope
    , $ionicPopup, $state, CarroService){
<!-- ... -->
```

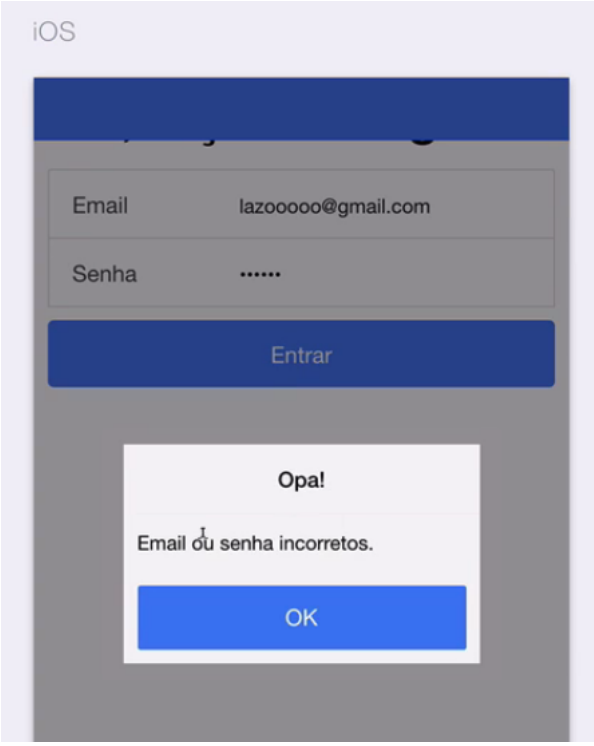
Depois, abaixo das variáveis, ele irá fazer a chamada dos dados do login.

```
CarroService.realizarLogin(dadosDoLogin).then(function(dados){

  }, function(erro){
    $ionicPopup.alert({
      title : 'Opa!',
      template : 'E-mail ou senha incorretos.'
    })

  })
```

Caso o usuário tenha informado algum dado incorreto, ele cairá na função de `erro` e, então, será mostrado um pop-up com a mensagem: `E-mail ou senha incorretos`. Criamos a pop-up utilizando o `ionicPopup`. Ele será adicionado também na `controller`:

```
angular.module('starter')
.controller('FinalizarPedidoController', function($scope, CarroService, $ionicPopup){

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

Podemos testar no navegador e veremos que o pop-up irá funcionar.



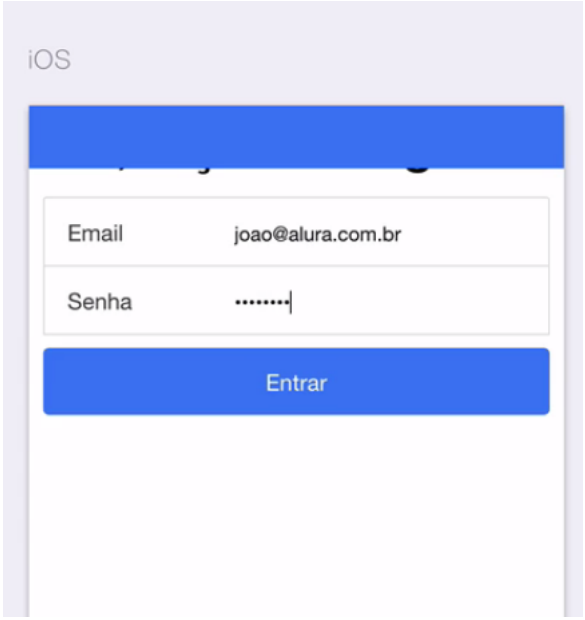

Realmente, não existe o usuário no Back-End, ele precisa mostrar a mensagem. Mas se o e-mail e senha estiverem corretos, eles devem nos direcionar para a listagem. Para isto, adicionaremos o `$state` na `controller`:

```
angular.module('starter')
.controller('FinalizarPedidoController', function($scope, CarroService, $ionicPopup, $state){

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

Depois, mais abaixo, adicionaremos o `$state.go()` e dentro, a `listagem`.

```
CarroService.realizarLogin(dadosDoLogin).then(function(dados){
    $state.go('listagem');
```

Para fazer o teste, se você for usar o Back-End construído para o curso, só teremos uma opção válida: e-mail é `joao@alura.com.br` e a senha é `alura123`. Mas você tem a opção de usar um Back-End com vários usuários.




Após clicarmos no botão "Entrar", seremos levados para a listagem. Se fizermos o mesmo tipo de test no Android, teremos os mesmo resultados.

Então, primeiro nós construímos a tela, depois, construímos a funcionalidade por trás. Com isso, adicionamos mais um funcionalidade ao nosso aplicativo.