04

Salvando o pedido

Salvando o pedido

Olá! Agora que a lista de carros vem do servidor, nosso próximo objetivo é salvar o pedido. Depois que o cliente escolhe o carro e os acessórios e finaliza o pedido com seus dados, esse pedido precisa ser salvo. Então todos os dados da tela "Estamos quase lá" devem ser enviados para a concessionária quando o botão "Realizar um sonho" for pressionado.

Como na aula anterior, faremos isso com um arquivo de serviço. Entretanto, as informações serão passadas por uma URL diferente: `https://aluracar.herokuapp.com/salvarpedido`. No arquivo `service.js` já temos uma função para obter os carros, e abaixo dela criaremos a função para salvar os pedidos, aproveitando a url que já salvamos nesse arquivo.

```
salvarPedido : function (pedido){
    return $http.get(url + "salvarpedido", pedido).then(function(response){
      return "Deu certo."
      });
}
```

Para que a aplicação pegue os dados da tela "Estamos quase lá" e mande para o servidor, precisamos ir para a função dessa tela no arquivo `controllers.js`. Por enquanto, ela está assim:

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

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

    $scope.finalizarPedido = function (){
        $ionicPopup.alert({
            title: 'Parabéns',

            template: 'Você acaba de comprar um carro.'
        }).then(function(){
            $state.go('listagem';)
        )};
    }
  });
```

A função que enviará os dados da tela para o servidor é a `finalizarPedido`. Entretanto, ainda não associamos a função aos dados que o comprador digitará na tela. Para colocá-la no `$scope` do Angular para receber os dados na *controller* e enviar para o serviço, é preciso criar uma variável. Vamos chamá-la de "pedido" e inseri-la entre o `$scope` de carro finalizado e o de finalizar pedido:

```
$scope.pedido = {};
```

Criada a variável, podemos usá-la na *view* da tela, para usar o valor digitado em cada campo e colocar no modelo com `ng-model`. Assim, os dados digitados em cada campo estarão no `$scope` do Angular.

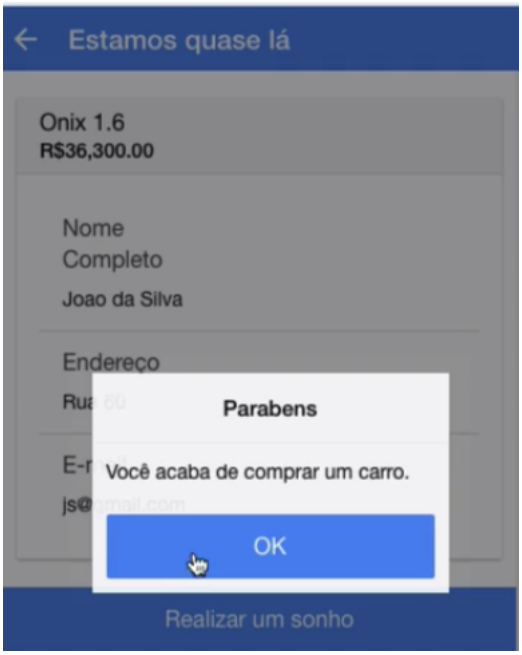
```
<div class="card">
```

```
        <div class="item item-divider">
            <h2>{{carroFinalizado.nome}}</h2>
            <h3>R{{carroFinalizado.preco | currency}}</h3>
        </div>
        <div class="item item-text-wrap">
            <div class="list">

                <label class="item item-input item-stacked-label">
                    <span class="input-label">Nome Completo</span>"
                    <input type="text ng-model="pedido.nome">
                </label>
                <label class="item item-input item-stacked-label">
                    <span class="input-label">Endereço</span>
                    <input type="text" ng-model="pedido.email">
                </label>
            </div>
        </div>
    <a class = "button button-positive button-full ng-click" finalizarPedido()>Realizar um sonho</a>
```

Quando o comprador usar o botão (`ng-click` na *view*), ativará um serviço que se chamará `CarroService` . Para isso, precisamos injetar o serviço na *controller* e colocá-lo no `$scope.finalizarPedido` . Nesse serviço colocaremos os objetos `pedido` e o `carroFinalizado` , afinal são esses os dados de que o cliente precisa. Uniremos os dois na variável `pedidoFinalizado` , que entrará no `CarroService` . O código ficará assim:

```
angular.module('starter')
.controller}('FinalizarPedidoController', function($stateParams, $scope, $ionicPopup, $state, Carros

    $scope.carroFinalizado = angular.fromJson($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
          }
      }
      CarroService.salvarPedido(pedidoFinalizado).then(function(dados){
          $ionicPopup.alert({
              title: 'Parabéns',
              template: 'Você acaba de comprar um carro.'
          }).then(function(){
              $state.go('listagem')
          });
      });

    }

});
```

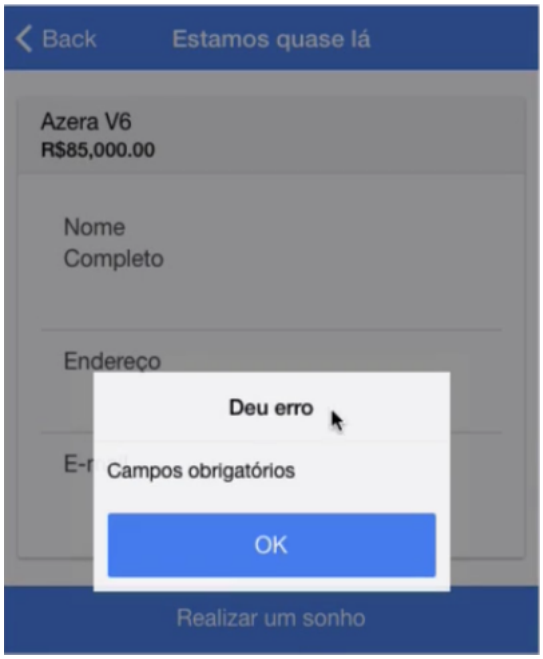
Note que o pop-up será chamado quando terminar a função e o pedido for de fato finalizado, por isso foi deslocado para dentro do serviço. No simulador, quando preenchemos os campos e clicamos em "Realizar um sonho", teremos o seguinte resultado:



Se não preenchermos os campos e clicarmos no botão, não recebemos nenhum retorno. Isso acontece porque os dados são obrigatórios e o servidor acusa erro. Podemos programar para esse caso uma mensagem de *callback* de erro. Voltando ao código do serviço, podemos inseri-la logo abaixo da função de sucesso. Para chamar a atenção do usuário, colocaremos outro pop-up.

```
\\...
}, function(erro){
    $ionicPopup.alert({
      title: 'Deu erro',
      template: 'Campos obrigatórios'
      });
\\...
```

Voltando para o simulador para testar o novo pop-up, obteremos a tela seguinte:



O que nos mostra que o servidor reconheceu o erro e está lidando com ele corretamente. Portanto, o nosso objetivo da aula foi cumprido. Até a próxima!