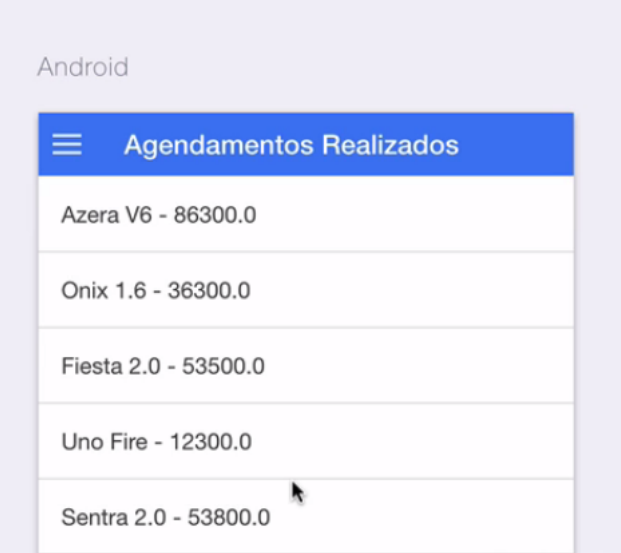
02

Sinalizando o erro na app

Transcrição

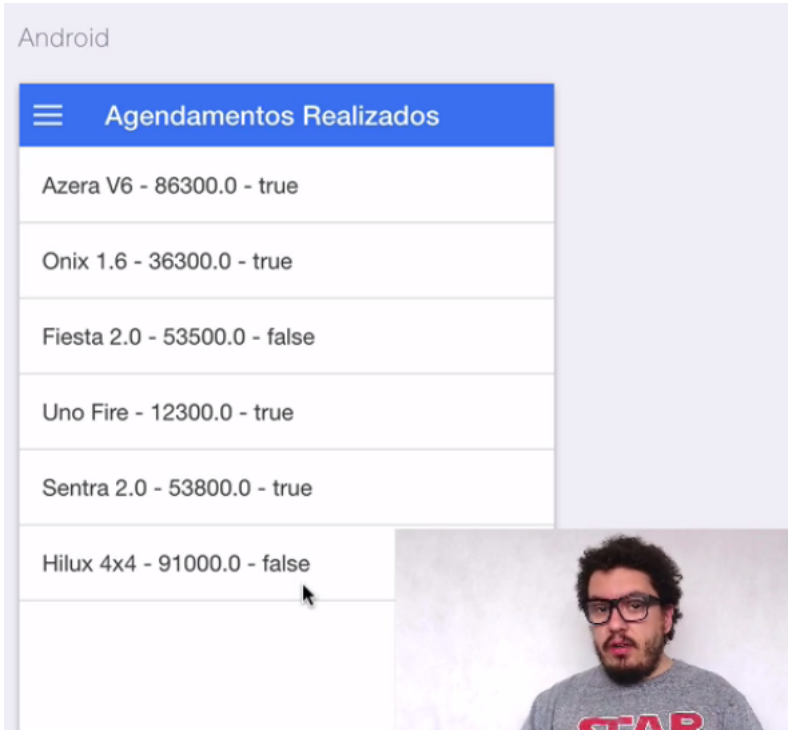
Nós temos uma tela nova, referentes aos agendamentos.



Agora, precisamos informar ao usuário quais agendamentos não deram certo. Qual esta como confirmado ou não. Vamos colocar o dado na tela, para mostrar para o usuário qual ele precisa reenviar. Vamos continuar trabalhando com o arquivo `agendamentos.html` . Começaremos adicionando o `{{agendamento.confirmado}}` :

```
<ion-view view-title='Agendamentos Realizados'>
    <ion-content>
        <ion-list>
            <ion-item ng-repeat="agendamento in agendamentos">
                {{agendamento.modelo}} - {{agendamento.preco}} - {{agendamento.confirmado}}
            </ion-item>
        </ion-list>
    </ion-content>
</ion-view>
```

Com estas alterações, já veremos o `true` e `false` no aplicativo.



Mas será que o usuário entenderia os termo boolenos `true` e `false` ? Não é interessante mostrar tais termos para o usuário. Seria interessante criar uma forma visual para que ele entenda que deu problema, por exemplo, colocar em vermelho os casos `falses` . Ou seja, usaremos um `class` dinâmico. Neste caso, adicionaremos uma classe CSS no projeto Ionic. Temos já criado um arquivo chamado `style.css` , dentro da pasta `css` :

```
.falha{
    color: red
}
```

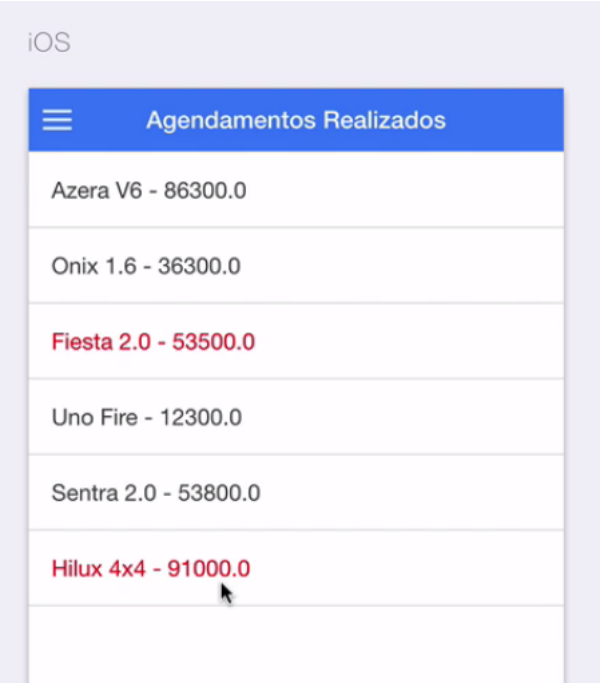
Este arquivo já é carregado no `index.html` , então toda classe adicionada no arquivo CSS, todo o projeto conseguirá enxergar.

Queremos que a classe seja usada apenas nos casos que forem `false` , então vamos adicionar o `ng-class` no `agendamentos.html` e trabalharemos com um objeto que será a classe recém criada `falha` .

```
<ion item ng-class="{'falha' : agendamento.confirmado == 'false'}" ng-repeat="agendamento in agendam
  {{agendamento.modelo}} - {{agendamento.preco}}
</ion-item>
```

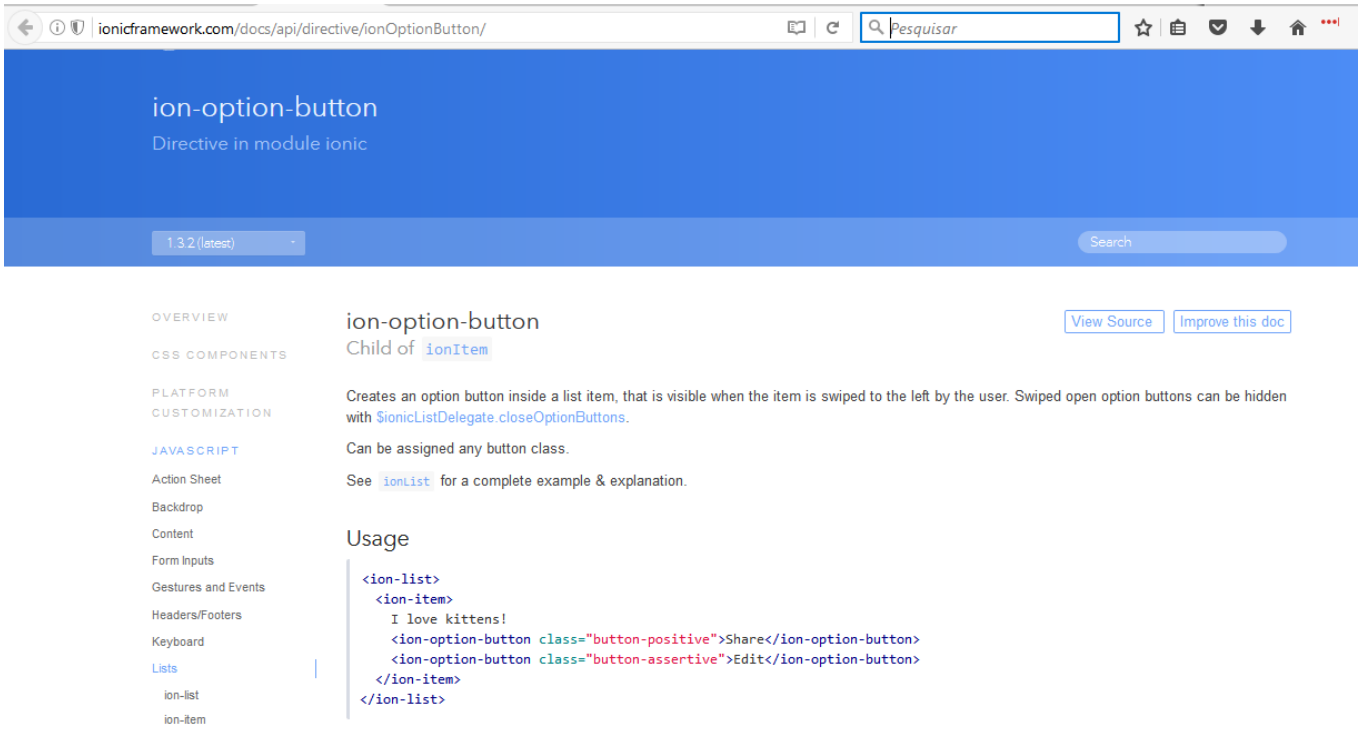
Como não precisamos mais do dado `{{agendamento.confirmado}}` , ele foi apagado.

Ao testarmos as alterações feitas no aplicativo, veremos que ficou mais claro para o usuário quais itens falharam no agendamento.



Agora, precisaremos adicionar um botão que será responsável pelo reenvio do pedido.

Quero apresentar para vocês o componente `ion-option-button`. Quando arrastamos o dedo para direita no item, o botão irá aparecer.



Vemos várias aplicações usando este componente, vamos usá-la também.

Adicionaremos a tag `<ion-option-button>` no `agendamentos.html`. O nome do botão será `Reenviar`:

```
<ion-view view-title='Agendamentos Realizados'>
    <ion-content>
        <ion-list>
      <ion item ng-class="{'falha' : agendamento.confirmado == 'false'}" ng-repeat="agendamento in a
        <ion-option-button class="button-positive">Reenviar</ion-option-button>
        {{agendamento.modelo}} - {{agendamento.preco}}
```

```
        </ion-item>

          </ion-list>

      </ion-content>

  </ion-view>
```

O botão "Reenviar" poderá ser visto na aplicação apenas quando arrastamos o item.

Temos o botão, agora, podemos criar um ação adicionando o `ng-click` no `<ion-option-button>`.

```
<ion item ng-class="{'falha' : agendamento.confirmado == 'false'}" ng-repeat="agendamento in agendam
  <ion-option-button ng-click="reenviar(agendamento)" class="button-positive">Reenviar</ion-option-b
  {{agendamento.modelo}} - {{agendamento.preco}}
</ion-item>
```

Nós não poderemos passar todo o *array*, então passaremos um único objeto que é o `agendamento`. Agora, seguiremos para o arquivo `agendamentos.controller.js`. Nele poderemos criar a função `$scope.reenviar`:

```
$scope.reenviar = function(agendamento){

  var agendamentoFinalizado = {
    params : {
      nome : agendamento.nome,
      endereco : agendamento.endereco,
      email : agendamento.email,
      carro : agendamento.modelo,
      preco : agendamento.preco

    }
  }
}
```

Nós montamos o objeto de forma semelhante a que foi feita no arquivo `finalizarPedido.controller.js`. Também injetamos o `CarroService` na `controller`, ainda no `agendamentos.controller.js`:

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

//...
```

E agora, podemos usar o `CarroService` dentro da função, passando como objeto `agendamentoFinalizado`.

```
$scope.reenviar = function(agendamento)

$scope.reenviar = function(agendamento){

  var agendamentoFinalizado = {
    params : {
      nome : agendamento.nome,
      endereco : agendamento.endereco,
      email : agendamento.email,
      carro : agendamento.modelo,
      preco : agendamento.preco

    }
  }

  CarroService.salvarPedido()
```

Seguiremos mostrando a chamada:

```
CarroService.salvarPedido(agendamentoFinalizado).then(function(dados){

  DatabaseValues.setup();
  DatabaseValues.bancoDeDados.transaction(function(transacao){
    transacao.executeSql("UPDATE agendamentos SET confirmado = 'true' WHERE id = ? ", [agendamento.i
  })
```

No caso de sucesso, precisamos atualizar o banco de dados, para informar que o agendamento foi enviado. Com o `transacao.executaSql()` executamos a transação necessária. O segundo parâmetro e um *array* que trocará todas as interrogações (`?`), pelos valores informados.

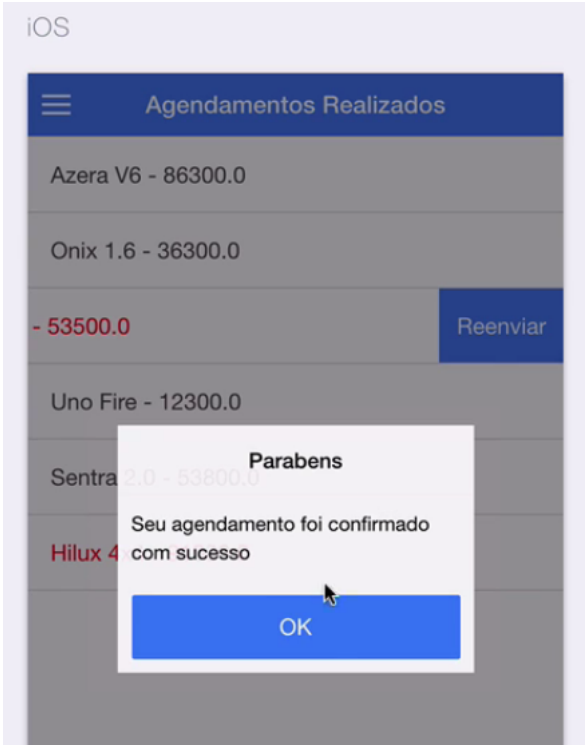
Adicionaremos também uma função de `erro` caso ocorra uma falha com o servidor. Usaremos um pop up tanto se tivermos ou não sucesso no agendamento.

```
CarroService.salvarPedido(agendamentoFinalizado).then(function(dados){

  DatabaseValues.setup();
  DatabaseValues.bancoDeDados.transaction(function(transacao){
    transacao.executeSql("UPDATE agendamentos SET confirmado = 'true' WHERE id = ? ", [agendamento.i
  })
```

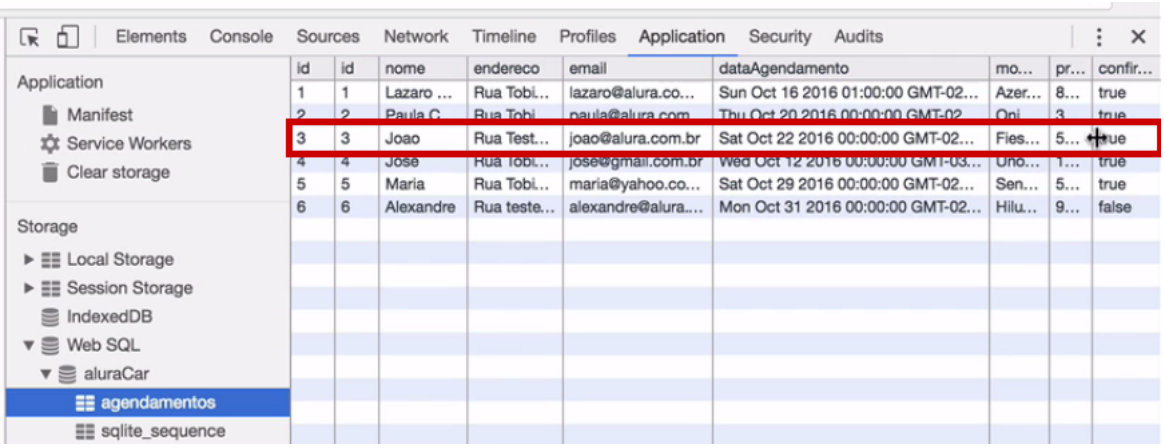
```
    $ionicPopup.alert({
      title : 'Parabens',
      template : 'Seu agendamento foi confirmado com sucesso'
    })


  }, function(erro){

    $ionicPopup.alert({
      title : 'Ops!',
      template : 'O servidor continua com erro. Tente mais tarde'
    })

  })
```

Então, quando tivermos sucesso, iremos alterar o banco e enviar a mensagem. No caso de erro, vamos apenas enviar a mensagem.

Vamos testar novamente a aplicação e ver o que acontece ao reenviarmos o agendamento:



Quando tivermos sucesso, receberemos a mensagem de `Parabéns` . E a tabela foi atualizada para `true` :



Conseguimos fazer o que queríamos!