

02

Sinalizando o erro na app

Transcrição

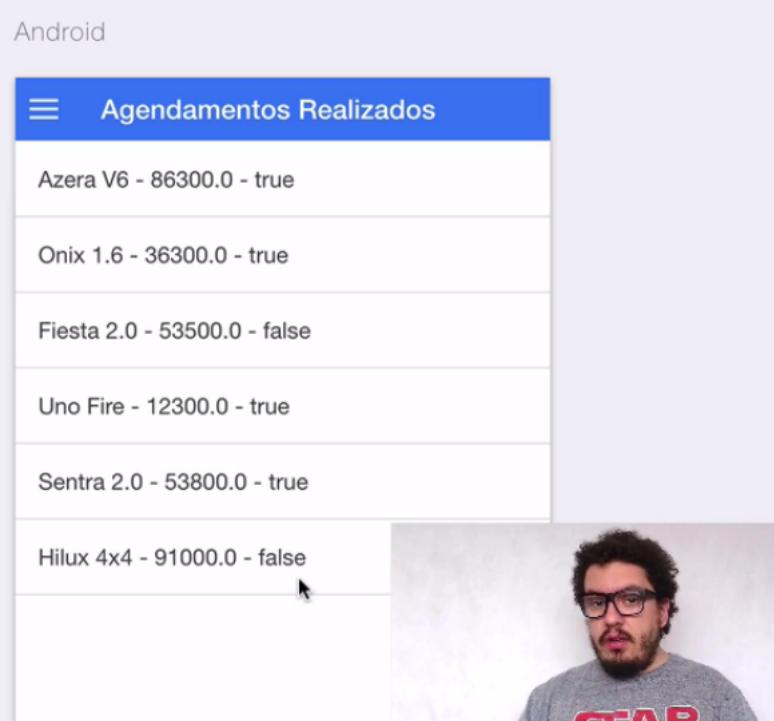
Nós temos uma tela nova, referentes aos agendamentos.



Agora, precisamos informar ao usuário quais agendamentos não deram certo. Qual está 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 termos booleanos `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 `false`s. 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 agendamentos"
  {{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.

ion-option-button
Directive in module ionic

OVERVIEW
CSS COMPONENTS
PLATFORM CUSTOMIZATION
JAVASCRIPT
Action Sheet
Backdrop
Content
Form Inputs
Gestures and Events
Headers/Footers
Keyboard
Lists
ion-list
ion-item

ion-option-button
Child of `ionItem`

Creates an option button inside a list item, that is visible when the item is swiped to the left by the user. Swiped open option buttons can be hidden with `$ionicListDelegate.closeOptionButtons`.

Can be assigned any button class.
See `ionList` for a complete example & explanation.

Usage

```
<ion-list>
  <ion-item>
    I love kittens!
    <ion-option-button class="button-positive">Share</ion-option-button>
    <ion-option-button class="button-assertive">Edit</ion-option-button>
  </ion-item>
</ion-list>
```

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 agenda">
        <ion-option-button class="button-positive">Reenviar</ion-option-button>
        {{agendamento.modelo}} - {{agendamento.preco}}
      </ion_item>
    </ion-list>
  </ion-content>
</ion-view>
```

```

</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 agendamentos">
  <ion-option-button ng-click="reenviar(agendamento)" class="button-positive">Reenviar</ion-option-button>
  {{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.id])
  })
})
```



No caso de sucesso, precisamos atualizar o banco de dados, para informar que o agendamento foi enviado. Com o `transacao.executeSql()` executamos a transação necessária. O segundo parâmetro é 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.id])
  })
})
```

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

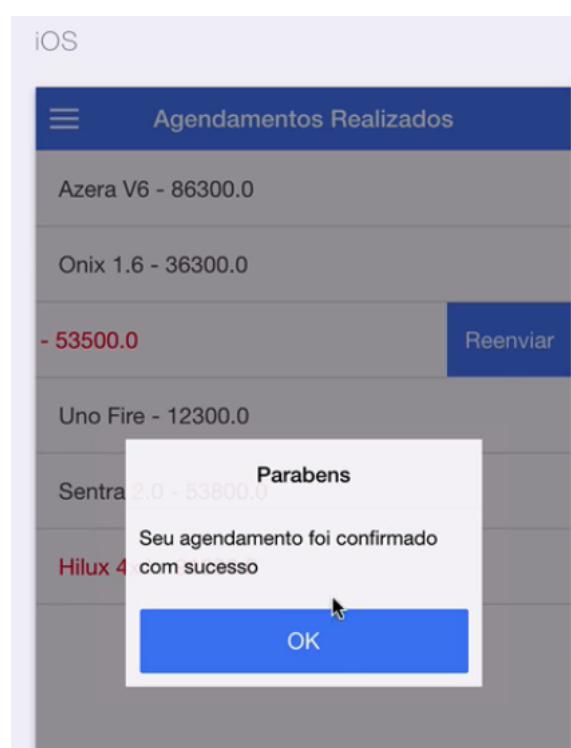
, function(erro){

$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`:

	id	id	nome	endereco	email	dataAgendamento	mo...	pr...	confir...
Application	1	1	Lazaro ...	Rua Tobi...	lazaro@alura.co...	Sun Oct 16 2016 01:00:00 GMT-02...	Azer...	8...	true
Manifest	2	2	Paula C	Rua Tobi...	paula@alura.com	Thu Oct 20 2016 00:00:00 GMT-02...	Oni...	3...	true
Service Workers	3	3	Joao	Rua Test...	joao@alura.com.br	Sat Oct 22 2016 00:00:00 GMT-02...	Fies...	5...	true
Clear storage	4	4	Jose	Rua rob...	jose@gmail.com.br	Wed Oct 12 2016 00:00:00 GMT-03...	Uno...	1...	true
Storage	5	5	Maria	Rua Tobi...	maria@yahoo.co...	Sat Oct 29 2016 00:00:00 GMT-02...	Sen...	5...	true
Local Storage	6	6	Alexandre	Rua teste...	alexandre@alura...	Mon Oct 31 2016 00:00:00 GMT-02...	Hilu...	9...	false
Session Storage									
IndexedDB									
Web SQL									
aluraCar									
agendamentos									
sqlite_sequence									

Conseguimos fazer o que queríamos!