02

Refatoração e mensagem de erro

Transcrição

Nós já temos o nosso banco de dados configurado e a tabela está criada, podemos salvar os dados quando o usuário finaliza o pedido. Então, como veremos no `finalizarPedido.controller.js` quando o Back-End recebe a requisição com sucesso, salvamos o campo da tabela agendamentos como `true`. Caso o retorno seja com erro, nós colocamos com false. Nós temos que fazer este tratamento na `controller` do pedido. Então, quando o usuário clicar em `salvarPedido`, ele chamará a função no service, que terá a função `function(dados)`.

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

  $scope.salvarDadosNoBancoDeDados('true');


  $ionicHistory.nextViewOptions({
    disableBack : true
  })

  $ionicPopup.alert({
    title: 'Parabens',
    template: 'Você acaba de comprar um carro.'
  }).then(function(){
    $state.go('app.listagem');
  });
}.
```

Em seguida, criaremos a `function(erro)`:

```
}, function(erro){

}
```

E dentro do `function(dados)`, vamos precisamos pegar a tabela e inserir os dados que estão na tela (nome, email, endereço, data de agendamento...).

Vamos injetar o `DatabaseValues` na `controller`:

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

Na função `finalizarPedido()`, vamos configurar o banco de dados adicionando o `setup`. Depois, colocaremos a variável `bancoDeDados` e a `transacao` para colocarmos o SQL para dentro do banco de dados.

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

  DatabaseValues.setup();
```

```
    DatabaseValues.bancoDeDaods.transaction(function(transacao){
        transacao.executeSql('INSERT INTO agendamentos(nome, endereco, email, dataAgendamento, modelo,
      })


    $ionicHistory.nextViewOptions({
      disableBack : true
    })
```

O `confirmado` é um dado importante. Nos colocaremos `true` ou `false` para ele. Neste caso, já que estamos na parte referente ao **sucesso** da função, vamos colocar `true` . Em seguida, vamos inserir o valor da função. Como os `VALUES` serão dinâmicos, usaremos a interrogação (`?`) para cada um dos setes campos que trabalharemos. Iremos separá-las com vírgulas (`,`). Depois, para cada `?` , adicionaremos um valor dentro do *array*, que serão as variáveis.

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

  DatabaseValues.setup();
  DatabaseValues.bancoDeDados.transaction(function(transacao){
    transacao.executeSql('INSERT INTO agendamentos(nome, endereco, email, dataAgendamento, modelo, pre

  })
```

Observe que as variáveis estão na mesma ordem em que os campos correspondentes foram definidos. E no fim, colocamos o `true` .

E quando ocorrer o erro, colocaremos os mesmo itens, mas mudaremos o confirmado de `true` para `false` .

```
  //...
  }, function(erro){

  DatabaseValues.setup();
  DatabaseValues.bancoDeDados.transaction(function(transacao){
    transacao.executeSql('INSERT INTO agendamentos(nome, endereco, email, dataAgendamento, modelo, pre

    })
  }
```

Queremos que ele faça as mesmas configuramos, mas o confirmado ficou 'false'. Mas desta forma, nós praticamente duplicamos todo o trecho do código, só diferenciando um campo do `insert` . É melhor refatorarmos o código e deixar o parâmetro diferente em outra função. Primeiramente, adicionaremos um novo `$scope` .

```
  $scope.salvarDadosNoBancoDeDados = function(confirmado){

  }
```

E vamos recortar o trecho repetido e colar logo abaixo da linha:

```
$scope.salvarDadosNoBancoDeDados = function(confirmado){

  DatabaseValues.setup();
  DatabaseValues.bancoDeDados.transaction(function(transacao){
    transacao.executeSql('INSERT INTO agendamentos(nome, endereco, email, dataAgendamento, modelo, p
}
```

Onde estava o `true` fixo, trocaremos pela variável `confirmado`. Podemos também remover o trecho do `function(erro)`. Mais acima, chamaremos a função que acabamos de criar e como trata-se da parte em que teremos sucesso, passaremos o parâmetro `true`.

```
CarroService.salvarPedido(pedidoFinalizado).then(function(dados){
    $scope.salvarDadosNoBancoDeDados('true');
}
```

E no erro, usaremos o `false`:

```
//...
}, function(erro){
    $scope.salvarDadosNoBancoDeDados('false')
});
```

Após a refatoração, vamos testar a aplicação. No browser, vamos testar o banco de dados. Vou começar preenchendo os campos de agendamento com os meus dados. Veremos que eles aparecerão na tabela ao lado.

A última coluna é referente ao "confirmado" e está preenchida com `true`.

Faremos um segundo teste, desta vez com os dados da minha esposa "Paula". Teremos sucesso novamente.

Mas se fizermos o terceiro teste, veremos que desta vez teremos um caso de erro. Agora, a coluna de "confirmado" da tabela estará preenchido com o `false`.

Até aqui, conseguimos fazer tudo o que queríamos. Mas ficou estranho... Em caso de erro, o usuário não é avisado. Então, vamos adicionar um pop up que avise o usuário que ele deve tentar novamente. Nós já adicionamos um pop up no caso de sucesso, no mesmo arquivo `finalizarPerdido.controller.js`:

```
$ionicPopup.alert({
  title: 'Parabens',
  template: 'Você acaba de comprar um carro.'
}).then(function(){
  $state.go('app.listagem');
});
```

Faremos o mesmo no caso em que ocorrer o erro:

```
//...

}, function(erro){
$scope.salvarDadosNoBancoDeDados('false');

$ionicPopup.alert({
  title : 'Ops!',
  template : 'Servidor com problemas. Tente mais tarde.'
}).then(function(){
  $state.go('app.listagem');
})

});
```

Agora, após a exibição da mensagem, o usuário será direcionado para a tela de listagem. Quando clicarmos o "OK" (`then`), recebemos uma `function()` e usamos o `$state.go` .

Salvaremos as alterações e depois, faremos um novo teste. Teremos problemas com o agendamento do Alexandre e já conseguiremos ver a mensagem avisando o usuário.

Seremos redirecionados para a tela de listagem e os dados do erro aparecerão na tabela.

Agora, conseguimos salvar o dado localmente.