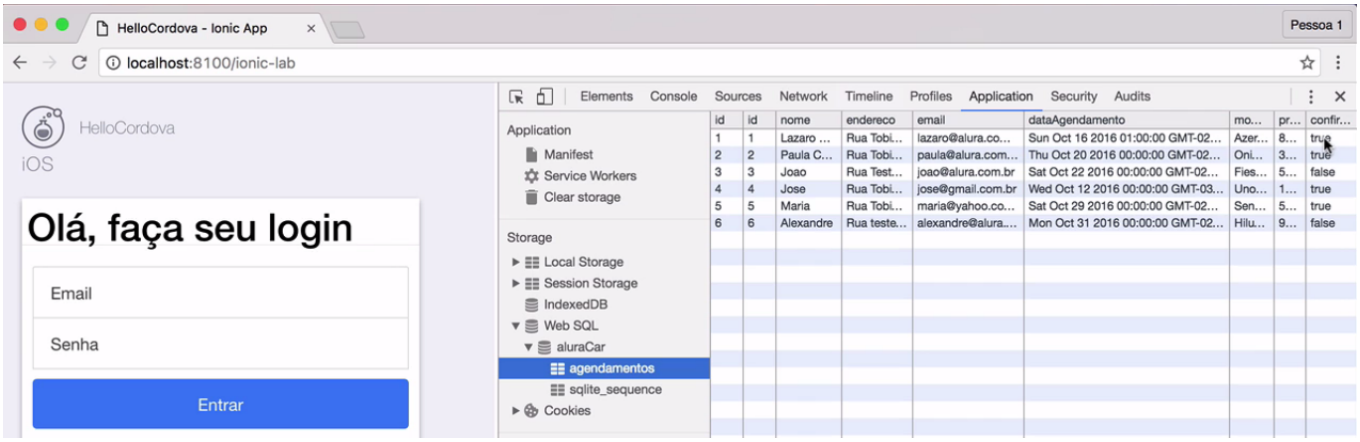
01

Listagem de agendamentos

Transcrição

Estou com o banco de dados abertos, vamos acessar a aba do desenvolvedor e temos os agendamentos que forma feitos. Lembrando que a cada três requisições um está falhando no Back-End.



Nós temos que mostrar para o usuário que o agendamento não deu certo. Para fazer o envio, ele pode novamente fazer todo o agendamento, mas isso não é muito habitual. Nas aplicações não costuma ser assim. Já que temos os dados salvos, podemos mostrar quais deram certos e os que não deram, podemos reenviar para o servidor. Para isto, temos que dar um `SELECT` no banco e colocar numa tela - que ainda não foi criada. Então, iremos adicionar a opção "Agendamento" no menu e em seguida, criaremos a tela nova.

Dentro da pasta `templates`, vou criar o arquivo `agendamentos.html`. Adicionaremos as tags de estrutura básica e como título da view usaremos `Agendamentos Realizados`.

```
<ion-view view-title='Agendamentos Realizados'>
    <ion-content>

    </ion-content>
</ion-view>
```

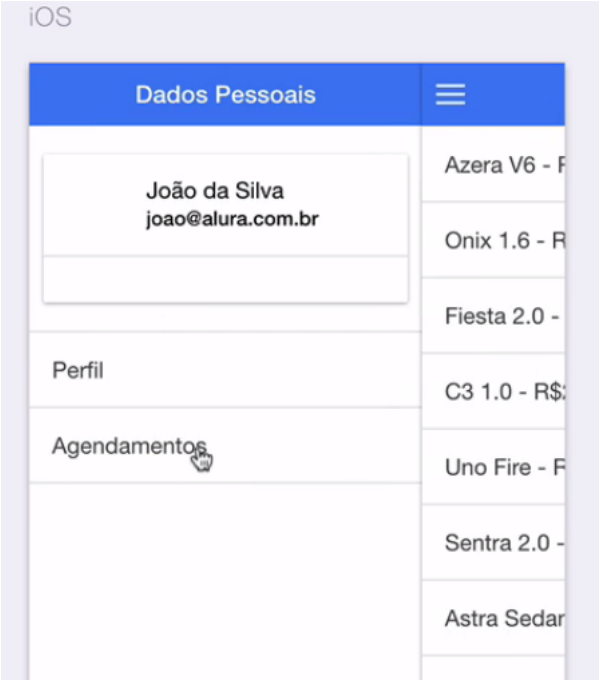
Quando criamos uma tela, precisamos criar uma rota. Faremos isto no `routes.js`. Abaixo do `app.finalizarpedido`, adicionaremos `app.agendamentos`.

```
.state('app.agendamentos' {
    url : '/agendamentos',
    view : {
        'menuContent' : {
            templateUrl : 'templates/agendamentos.html'
        }
    }
  })
```

Para chegarmos a esta tela, precisaremos criar um item no menu. No arquivo `menu.html`, adicionaremos a tag `<ion-item>`:

```
<ion-list>
  <ion-item menu-close href="#/app/perfil">
    Perfil
  </ion-item>
  <ion-item menu-close href="#/app/agendamentos">
    Agendamentos
  </ion-item>
```

No aplicativo já veremos o "Agendamentos" listado no menu:



É uma boa prática, criarmos o código e já testá-lo para verificarmos se não temos nenhum erro. Desta forma, fica mais fácil verificar em qual componente está o problema.

No nosso caso, está tudo tudo funcionando corretamente, podemos continuar. Agora, nós queremos pegar a lista de agendamentos do nosso banco de dados. Teremos em seguida, que criar uma variável para pegar a lista e exibi-la na tela. Precisaremos de uma `controller`. Mas já a adicionaremos no `routes.js`:

```
.state('app.agendamentos' {
    url : '/agendamentos',
    view : {
        'menuContent' : {
            templateUrl : 'templates/agendamentos.html'
            controller : 'AgendamentosController'
        }
    }
})
```

Depois, criaremos o arquivo `agendamentos.controller.js`. Após criá-lo, vamos adicioná-lo no `index.html`.

```
<!-- your app's js -->
<script src="js/app.js"></script>
<script src="js/controllers/listagem.controller.js" ></script>
<script src="js/controllers/carroEscolhido.controller.js" ></script>
<script src="js/controllers/finalizarPedido.controller.js" ></script>
```

```
    <script src="js/controllers/login.controller.js" ></script>
    <script src="js/controllers/menu.controller.js" ></script>
    <script src="js/controllers/perfil.controller.js" ></script>
    <script src="js/controlles/agendamentos.controller.js"></script>
```

E no `agendamentos.controller.js`, vamos adicionar o `controller` que chamaremos `AgendamentosController`:

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

    $scope.agendamentos = [];
  })
```

O `$scope` será um *array*.

Agora, preciso fazer um Select para pegar todos os dados. O arquivo `database.value.js` foi criado para configurar o banco de dados, vamos injetá-lo no `agendamento.controller.js`.

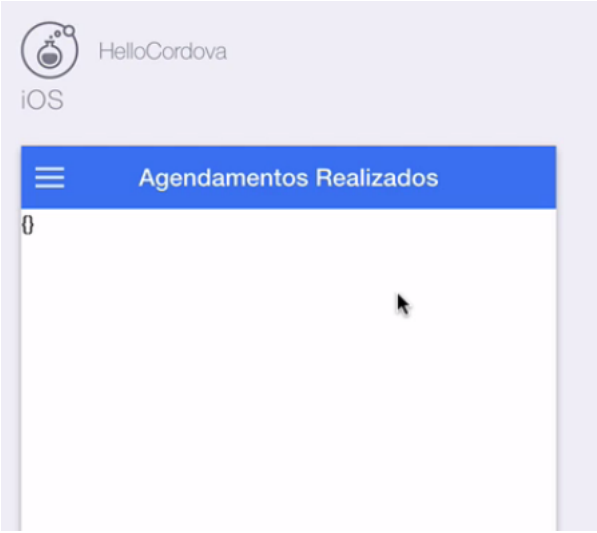
```
angular.module('starter')
.controller('AgendamentosController', function($scope, DatabaseValues){

    $scope.agendamentos = [];

    DatabaseValues.setup();
    DatabaseValues.bancoDeDados.transaction(function(transacao){
        transacao.executeSql('SELECT * FROM agendamentos', [], function(transacao, resultados))

        $scope.agendamentos = resultados;

      })
  })
```

Com o `setup`, configuramos o banco de dados e, então, fizemos o mesmo com a variável `bancoDeDados`. Com a transação adicionada, nós podemos executar um SQL que pegará os dados. Também informamos os três parâmetro, incluindo o segundo que será um *array* que passaremos para os campos dinâmicos. Por último, pegamos a variável criada e dentro, colocamos os `resultados`, que será mostrado na View.

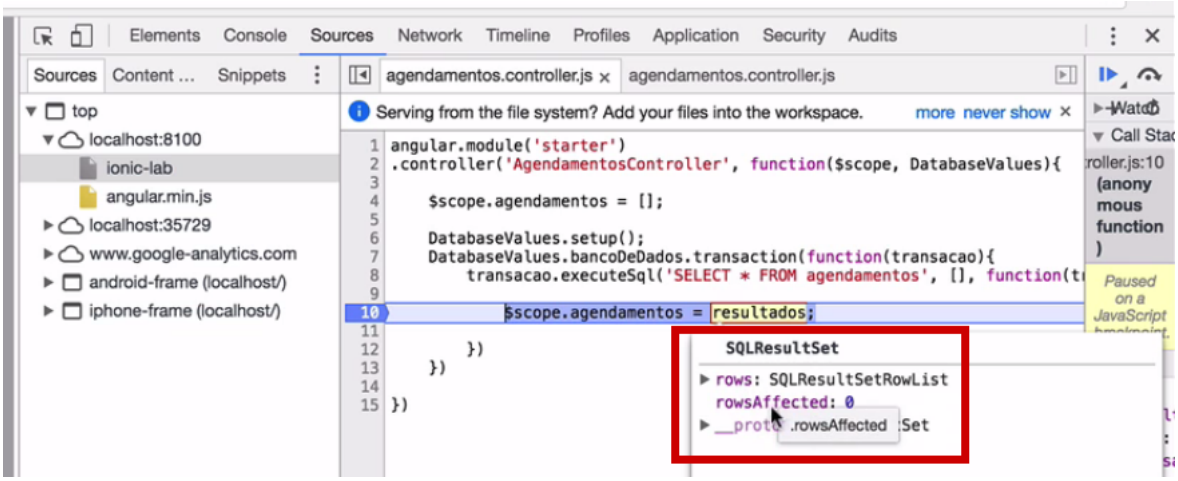
Agora, no `agendamentos.html`, adicionaremos o `{{agendamentos}}`:

```
<ion-view view-title='Agendamentos Realizados'>
    <ion-content>
        {{agendamentos}}

    </ion-content>
</ion-view>
```

Se testarmos o aplicativa, veremos que o resultado estará presente na View.



Mas ainda não é o resultado que queremos mostrar para o usuário. Felizmente, podemos usar as ferramentas do browser, e "debugar" para ver qual resultado ele trouxe.



Na aba "Sources", vemos que a variável `$scope.agendamentos` está vazia e `resultados` possui dois atributos: `rows` e `rowsAffected`. Dentro do `rows` encontramos seis objetos. Ou seja, o `resultados` trouxe o dados certos, porém não usamos o atributo certo. Nós teremos que criar um laço que percorrerá os objetos e pegará cada um deste, colocando-os dentro do *array*.

Na `controller`, adicionaremos um `for`:

```
for(var i = 0; i < resultados.rows.length; i++){
  $scope.agendamentos.push(resultados.rows[i]);

}
```

Podem ocorrer erros de compatibilidade caso exista uma diferença entre as versões dos componentes.

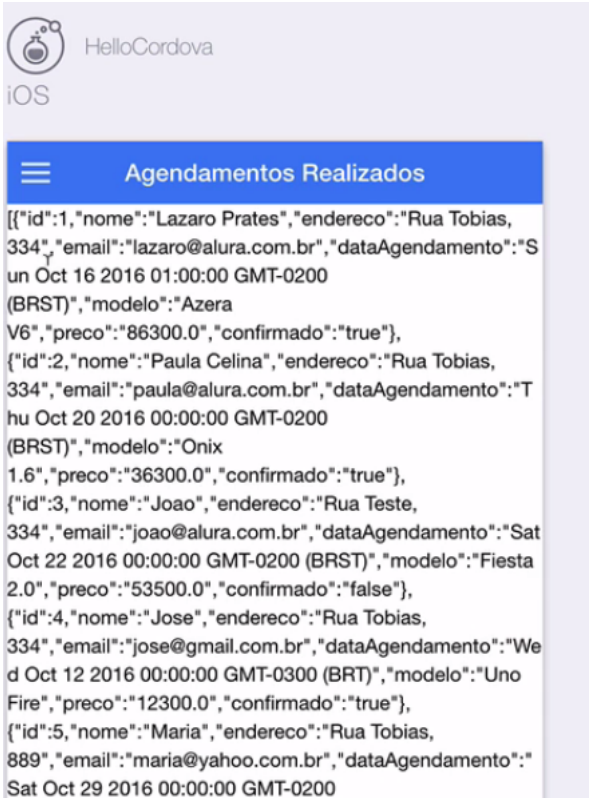
Cada item será colocado no *array*, como queremos colocar dentro de um objeto usaremos o `push()`.

Até aqui, o nosso código ficou assim:

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

```
    $scope.agendamentos = [];

    DatabaseValues.setup();
    DatabaseValues.bancoDeDados.transaction(function(transacao){
        transacao.executeSql('SELECT * FROM agendamentos', [], function(transacao, resultados))

        for(var i = 0; i < resultados.rows.length; i++){
          $scope.agendamentos.push(resultados.rows[i]);

        }

      })
  })
```

Quando testarmos e chegarmos na tela "Agendamentos Realizados", conseguiremos visualizar o resultados:



Estamos trazendo os resultados, mas não podemos mostrá-los assim para o usuário. Já viu alguma aplicação mostrando um JSON como resultado final? Para melhorarmos isto, podemos colocar cada cadastro como um item de uma lista. Vamos fazer algo semelhante ao que fizemos no `listagem.html`, usando o `ng-repeat`:

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

            </ion-item>

        </ion-list>

    </ion-content>
```

```
</ion-view>
```

Vamos conferir o resultado:

Perfeito! Fizemos um SQL no banco, buscamos os dados e disponibilizamos de uma forma mais visual para o nosso usuário.

Até a próxima!