03

Edição dos dados

Transcrição

O perfil já está recebendo os dados do login. Mas agora, queremos conseguir alterar os dados que o Back-End nos trouxe.

Adicionamos uma `label` para cada campo que precisava ser preenchido. Agora, adicionaremos um `input` para o "Editar".

```
<input type="text" ng-model="usuarioLogado.dataNascimento">
```

O valor que usaremos é o `model`. Vamos inserir outro `input` no trecho referente ao telefone:

```
<input type="text" ng-model="usuarioLogado.telefone">
```

E um último, no trecho referente ao e-mail.

```
<input type="text" ng-model="usuarioLogado.email">
```

Até aqui, o nosso código está assim:

```
<ion-view view-title="Perfil do Usuário">
    <ion-content>
        <div class="list card">
            <div class="item item-avatar">
                <h2> {{usuario.Logado.nome}} </h2>
            </div>
            <div class="item item-body">
                <div class="list">
                <label class="item item-input">
                    <span class="input-label"> {{usuarioLogado.dataNascimento}}
                    </span>
                    <input type="text" ng-model="usuarioLogado.dataNascimento">
                </label>
                <label class="item item-input">
                    <span class="input-label"> {{usuarioLogado.telefone | brPhoneNumber}} </span>
                    <input type="text" ng-model="usuarioLogado.telefone">
                </label>
                <label class="item item-input">
                    <span class="input-label"> {{usuarioLogado.email}} </span>
                    <input type="text" ng-model="usuarioLogado.email">
                </label>
                <button class="button button-full button-positive">Editar</button>
              </div>
            </div>
          </div>
      </ion-content>
  </ion-view>
```

Desta forma, o usuário já tem um lugar para inputar dados. Nossa tela ficou da seguinte forma:

Ao lado do telefone, já conseguimos ver o input. Resolvemos um problema, mas ele ficou estranho. Não é o formato que costumamos encontrar nas aplicações atualmente. Nós queremos que ao clicar no botão "Editar", possamos trocar o `span` pelo `input`. Como estamos trabalhando com Angular, para fazermos isso, usaremos a diretiva `ng-show` que receberá uma expressão. Com ela indicaremos se deverá aparecer´, nesse caso ela será `true`, caso contrário, será `false`. Precisaremos usar uma variável para saber quando está verdadeiro ou não o `input`. Vamos criar a variável `estaEditando` no `controllers.js`. Adicionaremos dentro do escopo:

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

    $scope.estaEditando = false;

    $scope.usuarioLogado = $rootScope.usuario;
})
```

Na primeira vez, não precisamos que o `input` esteja aparecendo, mas precisamos mostrar o `span`. Vamos especificar isso no

```
<label class="item item-input">
    <span ng-show="!estaEditando" class="input-label"> {{usuarioLogado.dataNascimento}}
    </span>
    <input ng-show="estaEditando" type="text" ng-model="usuarioLogado.dataNascimento">
  </label>
```

Usamos o sinal de exclamação (`!`) para negar. Repetiremos o mesmo com os `span` s e `input` s do `telefone` e `email`:

```
<ion-view view-title="Perfil do Usuário">
    <ion-content>
        <div class="list card">
            <div class="item item-avatar">
```

```
                <h2> {{usuario.Logado.nome}} </h2>
            </div>
            <div class="item item-body">
                <div class="list">
                <label class="item item-input">
                    <span ng-show="!estaEditando" class="input-label"> {{usuarioLogado.dataNascimen
                    </span>
                    <input ng-show="estaEditando" type="text" ng-model="usuarioLogado.dataNascimento
                </label>
                <label class="item item-input">
                    <span ng-show="!estaEditando" class="input-label"> {{usuarioLogado.telefone |
                    <input ng-show="estaEditando" type="text" ng-model="usuarioLogado.telefone">
                </label>
                <label class="item item-input">
                    <span ng-show="!estaEditando" class="input-label"> {{usuarioLogado.email}} </s
                    <input ng-show="estaEditando" type="text" ng-model="usuarioLogado.email">
                </label>
                <button class="button button-full button-positive">Editar</button>
            </div>
            </div>
        </div>
    </ion-content>
</ion-view>
```

Agora, não veremos mais o input nos campos das nossas telas. Mas também não conseguimos mais editar os campos, porque não fizemos a chamada do botão "Editar". Em `perfil.html`, usaremos a diretiva `ng-click` dentro do `button`.

```
<button ng-click ng-click="acaoBotao()" class="button button-full button-positive">Editar</button>
```

A função `acaoBotao()` terá que trocar o valor da variável para aparecer o `input`. Vamos usá-la no `controllers.js`:

```
angular.module('starter')
.controller('PerfilController', function($rootScope, $scope){
    $scope.estaEditando = false;

    $scope.usuarioLogado = $rootScope.usuario;

    $scope.acaoBotao = function(){
       $scope.estaEditando = true;
    }
})
```

Após salvarmos as alterações, na tela "Perfil do Usuário", ao clicarmos no botão "Editar" os campos terão se transformado em `input`.

No entanto, como faremos para salvar as alterações dos dados? Precisamos que o texto botão também modifique de acordo com a ação que quisermos realizar.

```
angular.module('starter')
.controller('PerfilController', function($rootScope, $scope){
    $scope.estaEditando = false;
    $scope.textoBotao = 'Editar';

    $scope.usuarioLogado = $rootScope.usuario;

    $scope.acaoBotao = function(){
       $scope.estaEditando = true;
    }
})
```

O botão terá uma expressão, no lugar de um nome fixo "Editar". Vamos fazer isto no `perfil.html`

```
<button ng-click ng-click="acaoBotao()" class="button button-full button-positive">{{textoBotao}}</
```

O botão mudará de nome de acordo com a função, para isto criaremos a regra dentro do `acaoBotao` no `controllers.js` :

```
$scope.acaoBotao = function(){
    if($scope.estaEditando) {
       $scope.estaeditando = false;
       $scope.textoBotao = 'Editar';
    } else {
        $scope.estaEditando = true;
        $scope.textoBotao = 'Salvar';
    }

}
```

Quando ele estiver no `if`, o texto do botão `Editar`, mas quando cair no `else`, o texto será `Salvar`. O arquivo `controller.js`, ficará assim:

```
angular.module('starter')
.controller('PerfilController', function($rootScope, $scope){
    $scope.estaEditando = false;
    $scope.textoBotao = 'Editar';

    $scope.usuarioLogado = $rootScope.usuario;

    $scope.acaoBotao = function(){
        if($scope.estaEditando) {
           $scope.estaeditando = false;
           $scope.textoBotao = 'Editar';
        } else {
            $scope.estaEditando = true;
            $scope.textoBotao = 'Salvar';
        }

    }

})
```

Mas observe que tenho duplicações de código. Se fôssemos novamente declarar o `false`, eu recomendaria você a refatorar as duas chamadas em uma função. Mas como nós estamos declarando para depois trocar o valor, vamos deixar como está. Você tem liberdade para refatorar e colocar as linhas parecidas em um função.

Veja que agora, quando alteramos os dados, temos o botão "Salvar" para gravar os dados modificados.

Assim que salvarmos as alterações, o botão voltará a exibir o texto "Editar". Alcançamos todos os objetivos desta aula.