**Dados inconsistentes? Precisamos removê-los ou alterá-los!**

Começando deste ponto? Você pode fazer o [DOWNLOAD](https://s3.amazonaws.com/caelum-online-public/angular-1/stages/07-alurapic.zip) completo do projeto do capítulo anterior e continuar seus estudos a partir deste capítulo.

**Temos dados inconsistentes, precisamos removê-los!**

Quem viu e quem vê nossa aplicação! O único problema é que temos dados inconsistentes cadastrados que precisam ser removidos, aliás, a funcionalidade de remoção é fundamental para o usuário, assim como poder alterar fotos já cadastradas. Vamos atacar primeiro a remoção, mas já aproveitaremos para adicionar o botão de edição.

Vamos adicionar um link para o botão editar e um botão para o botão remover dentro da diretiva meu-painel, abaixo da diretiva minha-foto:

<!-- public/partials/principal.html -->

<!-- código anterior omitido -->

<div class="row">

<meu-painel class="col-md-2 painel-animado" ng-repeat="foto in fotos | filter: filtro" titulo="{{foto.titulo}}">

<minha-foto url="{{foto.url}}" titulo="{{foto.titulo}}">

</minha-foto>

<a class="btn btn-primary btn-block" href="">Editar</a>

<button class="btn btn-danger btn-block" >Remover</button>

</meu-painel>

</div>

Usamos um link para edição porque uma navegação será gerada e usamos botão para remoção porque a ação de remover manterá o usuário na mesma página.

Sabemos que o Angular interage com a interface de eventos do JavaScript através de diretivas. Vamos adicionar a diretiva ng-click no botão "remover" que chamará a função remover em FotosController:

<!-- public/partials/principal.html -->

<!-- código anterior omitido -->

<div class="row">

<meu-painel class="col-md-2 painel-animado" ng-repeat="foto in fotos | filter: filtro" titulo="{{foto.titulo}}">

<minha-foto url="{{foto.url}}" titulo="{{foto.titulo}}">

</minha-foto>

<a class="btn btn-primary btn-block" href="">Editar</a>

<button class="btn btn-danger btn-block" ng-click="remover()">Remover</button>

</meu-painel>

</div>

Agora, vamos implementar em FotosController a função remover:

// public/js/controllers/fotos-controller.js

angular.module('alurapic').controller('FotosController', function($scope, $http) {

$scope.fotos = [];

$scope.filtro = '';

$http.get('/v1/fotos')

.success(function(retorno) {

$scope.fotos = retorno;

})

.error(function(erro) {

console.log(erro)

});

$scope.remover = function() {

// como saber qual foto será removida?

};

});

Sabemos que o click do botão remover chamará nossa função, mas qual foto deverá ser removida? Aquela que nossa função receber como parâmetro! Voltando para a view principal.htmL, passaremos foto como parâmetro de remover:

<!-- public/partials/principal.html -->

<!-- código anterior omitido -->

<div class="row">

<meu-painel class="col-md-2 painel-animado" ng-repeat="foto in fotos | filter: filtro" titulo="{{foto.titulo}}">

<minha-foto url="{{foto.url}}" titulo="{{foto.titulo}}">

</minha-foto>

<a class="btn btn-primary btn-block" href="">Editar</a>

<!-- novidade aqui! -->

<button class="btn btn-danger btn-block" ng-click="remover(foto)">Remover</button>

</meu-painel>

</div>

Lembre-se que a diretiva ng-repeat constrói nosso template, repetindo o elemento no qual foi adicionada para a quantidade de elementos da lista, e que podemos dar um apelido para termos acesso ao elemento, em nosso caso, usamos foto. Sendo assim, quando o usuário clicar em remover, nossa função receberá o objeto fotocorrespondente! Maravilha! Agora basta alterarmos nosso controller e utilizarmos $http.delete para apagarmos nosso foto no servidor. Já temos uma rota no servidor criada para tal tarefa com a seguinte estrutura:

/v1/fotos/IDdaFotoQueDesejamosApagar

Caso você trabalhasse em outra empresa, o endereço poderia ser diferente, o importante aqui é saber que o desenvolvedor front-end que não cria seu próprio back-end deve estar em sintonia com a equipe de back-end e solicitar os pontos de acesso. Ninguém fez curso "mãe Diná" para saber que endereços são esses!

// public/js/controllers/fotos-controller.js

angular.module('alurapic').controller('FotosController', function($scope, $http) {

$scope.fotos = [];

$scope.filtro = '';

$http.get('/v1/fotos')

.success(function(retorno) {

$scope.fotos = retorno;

})

.error(function(erro) {

console.log(erro)

});

$scope.remover = function(foto) {

$http.delete('/v1/fotos/' + foto.\_id)

.success(function() {

console.log('Foto ' + foto.titulo + ' removida com sucesso!');

})

.error(function(erro) {

console.log('Não foi possível apagar a foto ' + foto.titulo);

});

};

});

Antes de testarmos, que tal exibirmos também em principal.html uma mensagem de sucesso ou erro de acordo com o resultado da operação? Simples, da mesma maneira que fizemos em foto.html:

<!-- public/partials/principal.html -->

<div class="jumbotron">

<h1 class="text-center">Alurapic</h1>

</div>

<!-- novidade aqui! -->

<p ng-show="mensagem.length" class="alert alert-info">

{{mensagem}}

</p>

<!-- código posterior omitido -->

Agora, vamos alterar nosso controller para exibir as mensagens no lugar de imprimi-las no console. Porém, quando temos uma mensagem de erro, esta sim, exibimos no console:

// public/js/controllers/fotos-controller.js

angular.module('alurapic').controller('FotosController', function($scope, $http) {

$scope.fotos = [];

$scope.filtro = '';

$scope.mensagem = '';

$http.get('/v1/fotos')

.success(function(retorno) {

$scope.fotos = retorno;

})

.error(function(erro) {

console.log(erro)

});

$scope.remover = function(foto) {

$http.delete('/v1/fotos/' + foto.\_id)

.success(function() {

$scope.mensagem = 'Foto ' + foto.titulo + ' removida com sucesso!';

})

.error(function(erro) {

console.log(erro);

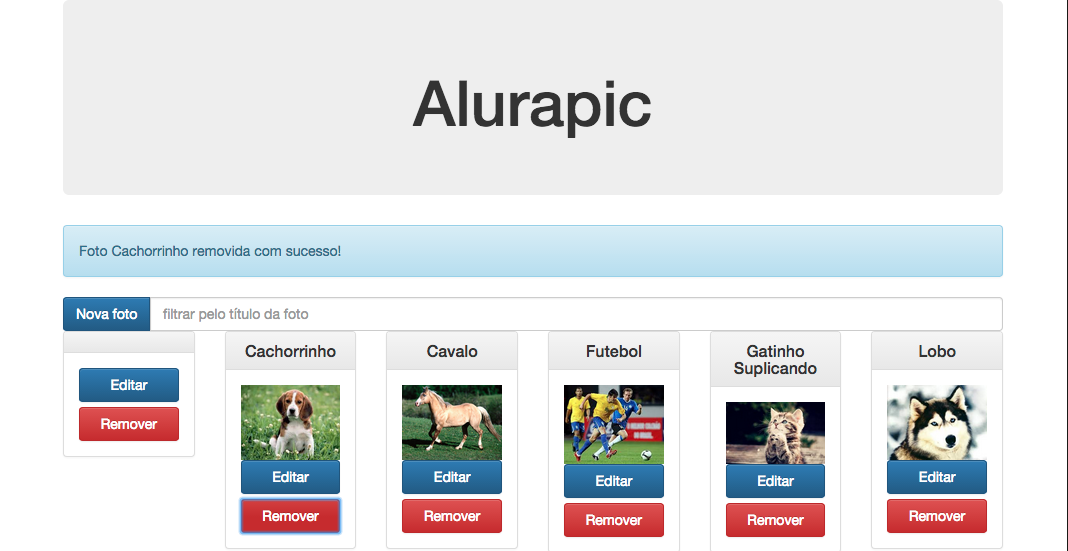
$scope.mensagem = 'Não foi possível apagar a foto ' + foto.titulo;

});

};

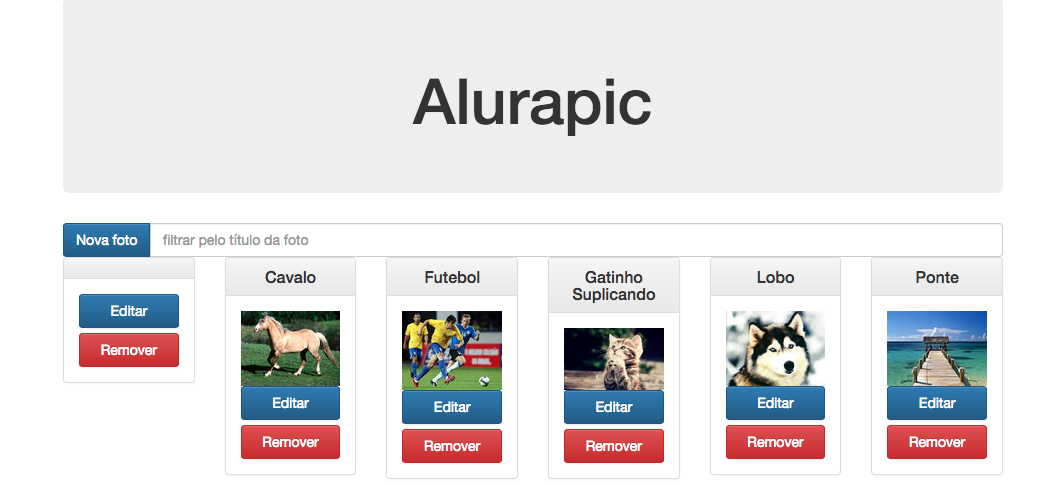
});

E agora? O que falta? Testar!



**Não removeu? Não se preocupe, a solução será performática**

Opa! Recebemos uma mensagem de sucesso dizendo que a foto com o título "Cachorrinho" foi removida, porém ela continua sendo exibida. E se recarregarmos a página?



O problema é que quando removemos a foto do servidor, $scope.fotos ainda contém nossa foto. Podemos resolver isso facilmente solicitando novamente a lista de fotos do servidor, porém estaríamos realizando uma requisição extra e evitar o número de requisições é sempre uma boa ideia, ainda mais se o usuário estiver numa rede móvel de alta latência.

Que tal removermos a foto da lista quando a operação de remoção for bem sucedida? Evitaríamos assim a requisição extra. Como $scope.fotos nada mais é do que uma array, podemos usar a tão conhecida funçãosplice para remover a foto:

// public/js/controllers/fotos-controller.js

angular.module('alurapic').controller('FotosController', function($scope, $http) {

$scope.fotos = [];

$scope.filtro = '';

$scope.mensagem = '';

$http.get('/v1/fotos')

.success(function(retorno) {

$scope.fotos = retorno;

})

.error(function(erro) {

console.log(erro)

});

$scope.remover = function(foto) {

$http.delete('/v1/fotos/' + foto.\_id)

.success(function() {

var indiceDaFoto = $scope.fotos.indexOf(foto);

$scope.fotos.splice(indiceDaFoto, 1);

$scope.mensagem = 'Foto ' + foto.titulo + ' removida com sucesso!';

})

.error(function(erro) {

console.log(erro);

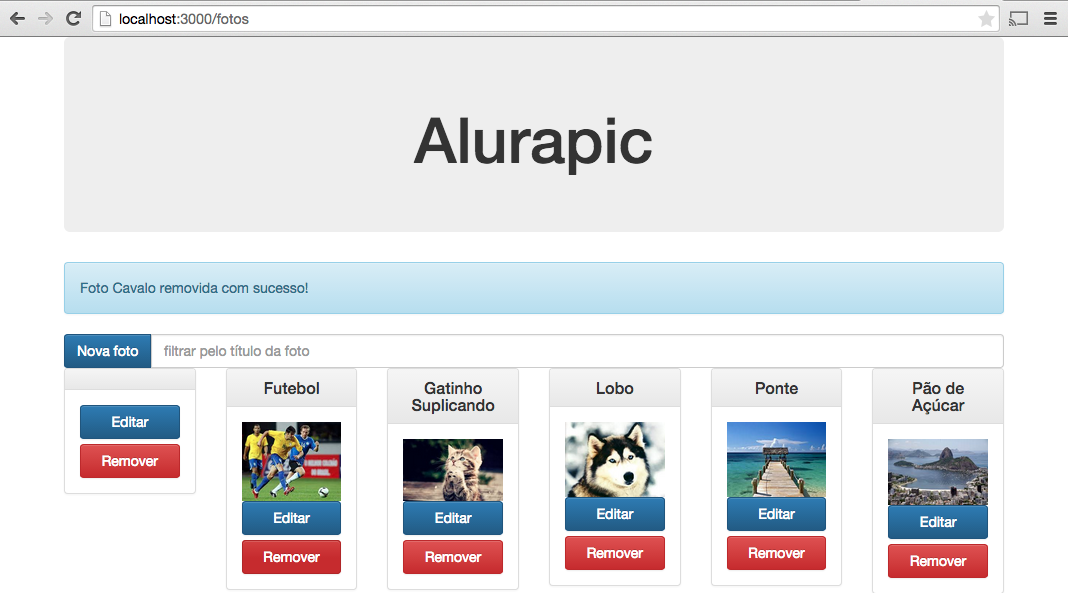
$scope.mensagem = 'Não foi possível apagar a foto ' + foto.titulo;

});

};

});

Agora, vamos remover o cavalo:



Excelente, funcionou, inclusive a remoção foi animada (ngAnimate)! Agora podemos remover os dados inválidos que cadastramos durante nossos testes. Não estranhe se receber a mensagem foto undefined foi removida, porque existem fotos cadastradas sem título! Agora, precisamos atacar a edição fotos.

**Quero editar! Como chegar até minha foto?**

A estratégia de edição será a seguinte: quando o usuário clicar no botão editar, iremos para a parcial foto.html, porém enviaremos o ID da foto como parâmetro. Em FotoController, capturaremos este ID. Quando o ID for passado, buscaremos a foto através deste ID atualizando o $scope.foto com seus dados.

O primeiro passo é registrarmos uma nova rota no Angular que saiba lidar com o ID da foto:

// public/js/main.js

angular.module('alurapic', ['minhasDiretivas', 'ngAnimate', 'ngRoute'])

.config(function($routeProvider, $locationProvider) {

$locationProvider.html5Mode(true);

$routeProvider.when('/fotos', {

templateUrl: 'partials/principal.html',

controller: 'FotosController'

});

$routeProvider.when('/fotos/new', {

templateUrl: 'partials/foto.html',

controller: 'FotoController'

});

// novidade aqui! Nova rota!

$routeProvider.when('/fotos/edit/:fotoId', {

templateUrl: 'partials/foto.html',

controller: 'FotoController'

});

$routeProvider.otherwise({redirectTo: '/fotos'});

});

Repare que registramos nossa nova rota da seguinte maneira:

'/fotos/edit/:fotoId'

No final da rota, usamos o curinga **:fotoId** que serve para duas coisas: indicar que a rota pode aceitar qualquer valor na posição do curinga e para indicar como teremos acesso ao parâmetro passado em nossos controllers. Veja também que estamos utilizando o mesmo controller e mesma view.

**Criando URLs dinamicamente**

Agora que já temos nossa rota cadastrada, vamos alterar o botão editar da nossa view parcial principal.html. Precisamos construir um endereço diferente para cada foto, levando em consideração seu ID. Isso é fácil, basta usarmos uma AE:

<!-- public/partials/principal.html -->

<!-- código anterior omitido -->

<div class="row">

<meu-painel class="col-md-2 painel-animado" ng-repeat="foto in fotos | filter: filtro" titulo="{{foto.titulo}}">

<minha-foto url="{{foto.url}}" titulo="{{foto.titulo}}">

</minha-foto>

<!-- novidade aqui! -->

<a class="btn btn-primary btn-block" href="/fotos/edit/{{foto.\_id}}">

Editar

</a>

<button class="btn btn-danger btn-block"

ng-click="remover(foto)">

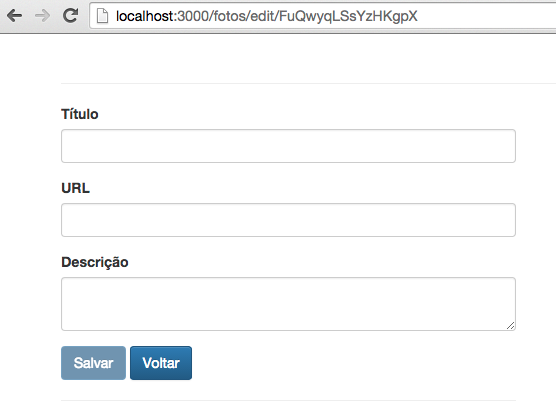
Remover

</button>

</meu-painel>

</div>

Depois de recarregarmos a página para que nossas alterações surtam efeito, quando clicamos em uma das fotos somos jogados para a view parcial foto.html, mas com a diferença de que o ID da foto é exibido como parte da URL:



**Achei! mas onde estão os dados da foto?**

Excelente, mas ainda precisamos ter acesso ao ID da foto em FotoController para que possamos buscá-la em nosso servidor. Existe um serviço especializado do Angular que nos fornecerá este parâmetro, o **$routeParams**. Como qualquer serviço no Angular, ele é recebido como parâmetro na função que define nosso controller:

// public/js/controllers/foto-controller.js

angular.module('alurapic')

.controller('FotoController', function($scope, $http, $routeParams) {

// código do controller omitido

});

Como utilizar o serviço? Basta sabermos o nome do curinga que usamos em nossa rota e utilizá-lo como propriedade de $routeParams:

// public/js/controllers/foto-controller.js

angular.module('alurapic')

.controller('FotoController', function($scope, $http, $routeParams) {

$scope.foto = {};

$scope.mensagem = '';

if($routeParams.fotoId) {

// busca a foto no servidor

}

$scope.submeter = function() {

if ($scope.formulario.$valid) {

$http.post('/v1/fotos', $scope.foto)

.success(function() {

$scope.foto = {};

$scope.mensagem = 'Foto cadastrada com sucesso';

})

.error(function(erro) {

console.log(erro);

$scope.mensagem = 'Não foi possível cadastrar a foto';

})

}

};

});

Se o parâmetro foi passado, buscamos a foto no servidor através de $http.get. O importante é saber que receberemos na função passada para success a foto que buscamos. Mais uma vez o programador front-end precisa conhecer os endereços e os dados retornados, caso ele não tenha sido o responsável pelo back-end:

// public/js/controllers/foto-controller.js

angular.module('alurapic')

.controller('FotoController', function($scope, $http, $routeParams) {

$scope.foto = {};

$scope.mensagem = '';

if($routeParams.fotoId) {

$http.get('/v1/fotos/' + $routeParams.fotoId)

.success(function(foto) {

$scope.foto = foto;

})

.error(function(erro) {

console.log(erro);

$scope.mensagem = 'Não foi possível obter a foto'

});

}

$scope.submeter = function() {

if ($scope.formulario.$valid) {

$http.post('/v1/fotos', $scope.foto)

.success(function() {

$scope.foto = {};

$scope.mensagem = 'Foto cadastrada com sucesso';

})

.error(function(erro) {

console.log(erro);

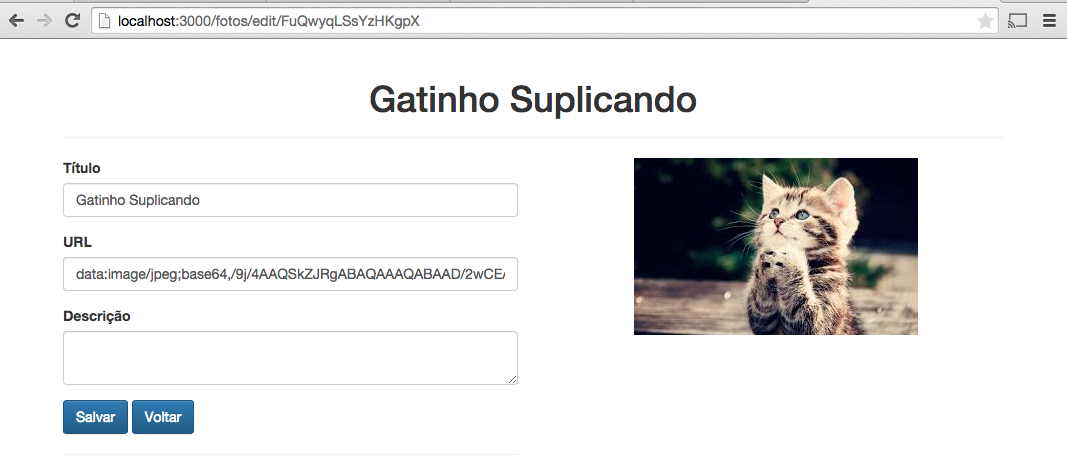
$scope.mensagem = 'Não foi possível cadastrar a foto';

})

}

};

});



Ótimo! Agora, precisamos alterar a lógica de $scope.submeter para que consiga tomar uma ação diferente para inclusão e alteração de fotos. Na alteração, normalmente se usa o verbo HTTP PUT, é por isso que utilizaremos$http.put:

// public/js/controllers/foto-controller.js

angular.module('alurapic')

.controller('FotoController', function($scope, $http, $routeParams) {

$scope.foto = {};

$scope.mensagem = '';

if($routeParams.fotoId) {

$http.get('/v1/fotos/' + $routeParams.fotoId)

.success(function(foto) {

$scope.foto = foto;

})

.error(function(erro) {

console.log(erro);

$scope.mensagem = 'Não foi possível obter a foto'

});

}

$scope.submeter = function() {

if ($scope.formulario.$valid) {

if($routeParams.fotoId) {

$http.put('/v1/fotos/' + $scope.foto.\_id, $scope.foto)

.success(function() {

$scope.mensagem = 'Foto alterada com sucesso';

})

.error(function(erro) {

console.log(erro);

$scope.mensagem = 'Não foi possível alterar';

});

} else {

$http.post('/v1/fotos', $scope.foto)

.success(function() {

$scope.foto = {};

$scope.mensagem = 'Foto cadastrada com sucesso';

})

.error(function(erro) {

console.log(erro);

$scope.mensagem = 'Não foi possível cadastrar a foto';

})

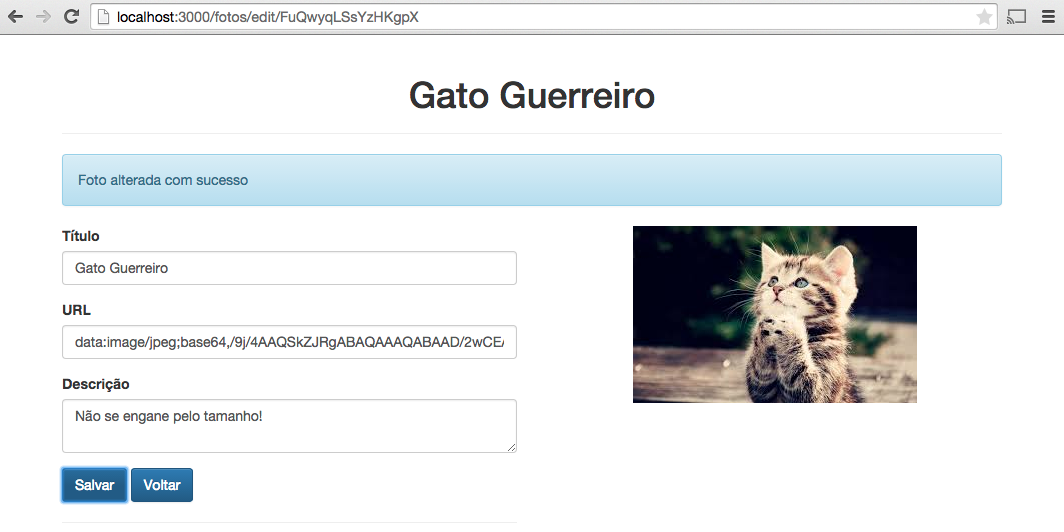
}

}

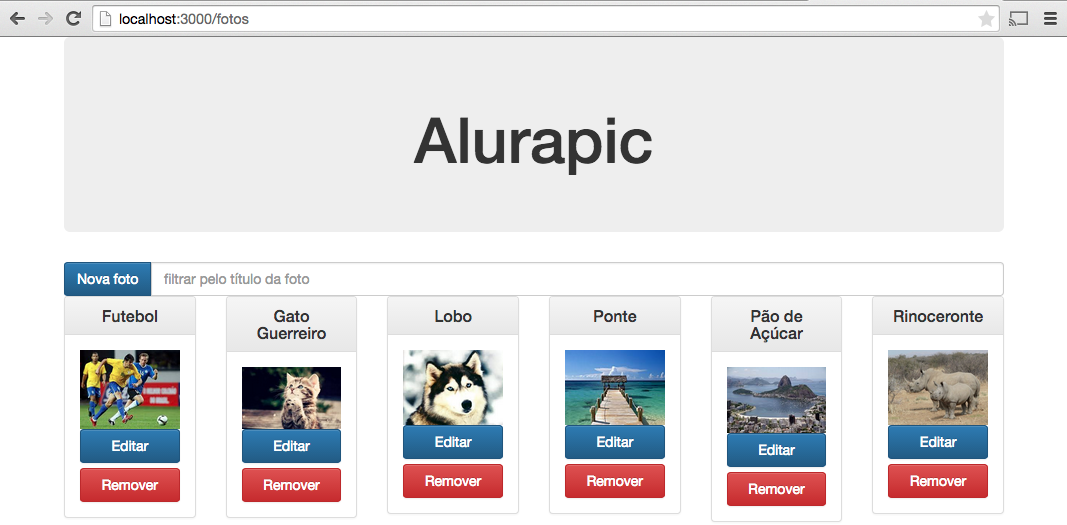
};

});

Agora só nos resta testar. Vamos alterar o nome da foto e colocar uma descrição qualquer. Quando salvamos, recebemos a mensagem de sucesso:



E agora? Se clicarmos em "voltar', nossa parcial principal.html buscará as fotos novamente no servidor, inclusive a que alteramos:



**O que aprendemos neste capítulo?**

* a diretiva ng-click
* deleção de recurso com $http.delete
* o truque com a função splice para evitar uma nova requisição
* rotas do Angular com curingas (parametrizadas)
* $routeParams e acesso a parâmetros
* alteração de recurso com $http.put
* único formulário para inclusão e alteração