# **Precisamos alimentar esse sistema**

Começando deste ponto? Você pode fazer o [DOWNLOAD](https://s3.amazonaws.com/caelum-online-public/angular-1/stages/06-alurapic.zip) completo do projeto do capítulo anterior e continuar seus estudos a partir deste capítulo.

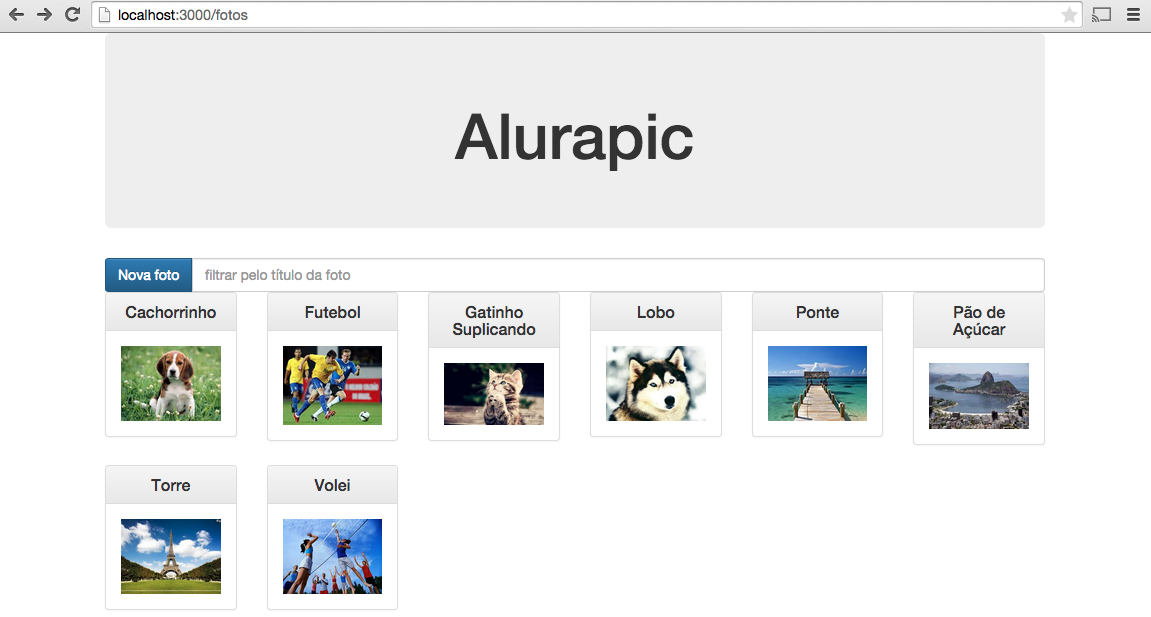
Não podemos postergar ainda mais, precisamos concluir o cadastro de novas fotos! Sabemos que a rota do Angular localhost:3000/fotos/new exibirá a view, mas não é nada elegante pedir que o usuário digite esse endereço toda vez quando for cadastrar uma foto, não?

## **Navegando entre views**

Podemos melhorar sua experiência adicionando um link que, ao ser clicado na parcial principal.html, chamará nossa rota que exibirá nossa tela de cadastro.

Vamos editar public/partias/principal.html para adicionarmos um link que terá um visual de botão graças ao Bootstrap:

<!-- public/partials/principal.html -->  
  
<div class="jumbotron">  
 <h1 class="text-center">Alurapic</h1>  
</div>  
  
<div class="row">  
 <div class="col-md-12">  
 <form>  
  
 <!-- Novidade! -->  
  
 <div class="input-group">  
 <span class="input-group-btn">  
 <a href="/fotos/new" class="btn btn-primary" type="button">  
 Nova foto  
 </a>  
 </span>  
 <input class="form-control" placeholder="filtrar pelo título da foto" ng-model="filtro" ng-model-options="{ debounce: 500 }">  
 </div>   
  
 <!-- fim novidade! -->  
  
 </form>  
 </div> <!-- fim col-md-12 -->  
</div> <!-- fim row -->  
  
<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>  
 </meu-painel>  
</div>



Agrupamos nosso botão e nosso campo de pesquisa dentro de um input-group do Bootstrap, inclusive nosso botão decora o campo de pesquisa com a classe input-group-btn.

Repare também que nosso botão, na verdade um link, aponta para o endereço /fotos/new, justamente a rota que já temos registrada. Agora que já conseguimos navegar entre as parciais principal.html e foto.html, podemos atacar a marcação desta última.

## **Nossa primeira view de cadastro**

Vamos criar nosso formulário de cadastro, mas ainda sem nos preocuparmos com expressões do Angular, inclusive já vamos adicionar os botões "salvar" e "voltar":

<!-- public/partials/foto.html -->  
  
<div class="page-header text-center">  
 <h1>TITULO DA FOTO AQUI</h1>  
</div>  
  
<form name="formulario" class="row">  
 <div class="col-md-6">  
 <div class="form-group">  
 <label>Título</label>  
 <input name="titulo" class="form-control">   
 </div>  
 <div class="form-group">  
 <label>URL</label>  
 <input name="url" class="form-control">  
 </div>  
 <div class="form-group">  
 <label>Descrição</label>  
 <textarea name="descricao" class="form-control">  
 </textarea>  
 </div>  
  
 <button type="submit" class="btn btn-primary">  
 Salvar  
 </button>  
 <a href="/" class="btn btn-primary">Voltar</a>  
 <hr>  
 </div>  
 <div class="col-md-6">  
 <minha-foto></minha-foto>  
 </div>  
</form>

Excelente, já podemos verificar o resultado!



Queremos agora que cada input do nosso formulário alimente as propriedades de um objeto foto sem qualquer informação. Vamos adicionar a já conhecida diretiva ng-model para cada um dos inputs. No título da página, usaremos uma AE:

<!-- public/partials/foto.html -->  
  
<div class="page-header text-center">  
 <h1>{{foto.titulo}}</h1>  
</div>  
  
<form name="formulario" class="row">  
 <div class="col-md-6">  
 <div class="form-group">  
 <label>Título</label>  
 <input name="titulo" class="form-control" ng-model="foto.titulo">   
 </div>  
 <div class="form-group">  
 <label>URL</label>  
 <input name="url" class="form-control" ng-model="foto.url">  
 </div>  
 <div class="form-group">  
 <label>Descrição</label>  
 <textarea name="descricao" class="form-control" ng-model="foto.descricao">  
 </textarea>  
 </div>  
  
 <button type="submit" class="btn btn-primary">  
 Salvar  
 </button>  
 <a href="/" class="btn btn-primary">Voltar</a>  
 <hr>  
 </div>  
 <div class="col-md-6">  
 <minha-foto></minha-foto>  
 </div>  
</form>

Quando recarregamos nossa página, o título some e o formulário ainda é exibido. Lembre-se que uma AE não avaliada não resulta em erro, mas apenas na ausência de valor no local onde é utilizada. A mesma coisa acontece com a diretiva ng-model.

Agora que já temos o "esqueleto" da nossa tela de cadastro, precisamos de um controller que nos dê suporte para a operação de cadastro. Vamos criar o arquivo public/js/controllers/foto-controller.js e definir o controller **FotoController** (no singular), importando-o logo em seguida na view principal da aplicaçãopublic/index.html:

// public/js/controllers/foto-controller.js  
angular.module('alurapic')  
 .controller('FotoController', function($scope) {  
  
 $scope.foto = {};  
  
 });

<!-- public/index.html -->  
<!DOCTYPE html>  
<html lang="pt-br" ng-app="alurapic">  
 <head>  
 <base href="/">  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width">  
 <title>Alurapic</title>  
 <link rel="stylesheet" href="css/bootstrap.min.css">  
 <link rel="stylesheet" href="css/bootstrap-theme.min.css">  
 <link rel="stylesheet" href="css/efeitos.css">  
 <script src="js/lib/angular.min.js"></script>  
 <script src="js/lib/angular-animate.min.js"></script>  
 <script src="js/lib/angular-route.min.js"></script>  
 <script src="js/main.js"></script>  
 <script src="js/controllers/fotos-controller.js"></script>  
  
 <!-- importando o novo controller -->  
  
 <script src="js/controllers/foto-controller.js"></script>  
 <script src="js/directives/minhas-diretivas.js"></script>  
 </head>  
 <body>  
 <div class="container">  
 <ng-view></ng-view>  
 </div><!-- fim container -->  
 </body>  
</html>

Veja que não importamos o novo script como último script. Por uma questão de organização apenas, ele foi importado imediatamente após o controller já existente.

Vamos voltar nossa atenção para FotoController. Veja que $scope disponibiliza a propriedade foto, um objeto JavaScript, porém sem qualquer propriedade. Não se preocupe, como estamos usando a diretiva ng-model, a propriedade indicada será criada automaticamente no objeto, sendo assim, se usamos ng-model="foto.titulo", o Angular criará automaticamente em $scope.foto a propriedade titulo, inclusive atribuindo o valor digitado pelo usuário.

## **Preparando o terreno para cadastrarmos novas fotos**

O que precisamos agora é implementar o botão salvar. O que ele deve fazer? Submeter o formulário, claro, mas precisamos acessar $scope.foto no momento da submissão para que possamos enviar os dados assincronamente através do serviço $http. JavaScript possui o evento **submit** justamente para isso.

O evento submit é disparado quando um formulário é submetido e nele podemos adicionar uma função que permite a execução de um código arbitrário que pode cancelar sua submissão caso haja algum problema, como o de um campo que não foi preenchido. Mas estamos usando Angular, e agora? Como interagir com a interface de eventos do JavaScript?

O Angular suporta a interface de eventos do JavaScript através de diretivas. Por exemplo, se quisermos o evento click, usamos a diretiva ng-click, o evento mouseouver, a diretiva ng-mouseover e assim por diante. Sendo assim, para lidarmos com o evento submit disparado pelo formulário adicionamos a diretiva **ng-submit** diretamente na tag form:

<!-- public/partials/foto.html -->  
  
<!-- código anterior omitido -->  
  
<form name="formulario" class="row" ng-submit="submeter()">  
  
<!-- código posterior omitido -->

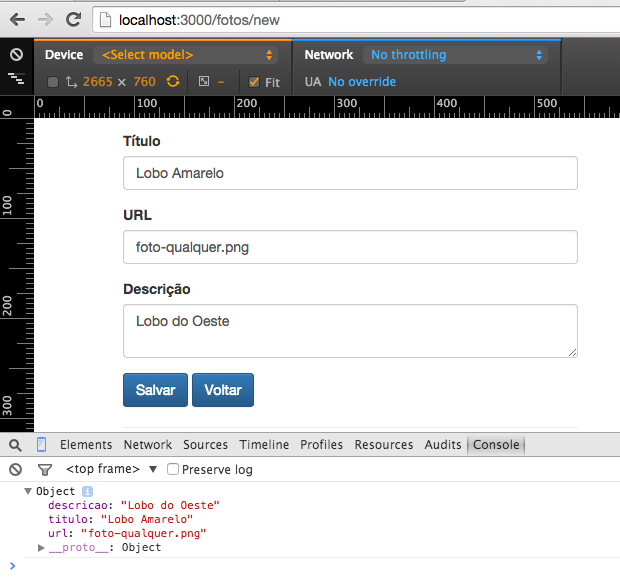
Note que o valor da diretiva ng-submit chama uma função que deve ser definida na propriedade$scope.submeter. Vamos adicioná-la em nosso controller, porém exibiremos apenas os dados da foto no console do navegador:

// public/js/controllers/foto-controller.js  
  
angular.module('alurapic')  
 .controller('FotoController', function($scope) {  
  
 $scope.foto = {};  
  
 $scope.submeter = function() {  
 console.log($scope.foto);  
 };  
  
 });

Se testarmos nosso código, nada acontecerá, por quê? O motivo é simples: não associamos FotoController à view parcial foto.html. Lembram onde realizamos essa associação? Na configuração de rotas! Vamos editar o arquivo public/js/main.js para adicionarmos a chave controller que faltava para a foto /fotos/new:

// 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'  
 });  
  
 // adicionando a propriedade controller que faltava.   
  
 $routeProvider.when('/fotos/new', {  
 templateUrl: 'partials/foto.html',  
 controller: 'FotoController'  
 });  
  
 $routeProvider.otherwise({redirectTo: '/fotos'});  
  
 });

Agora já podemos recarregar nossa página, clicar no botão nova foto, preencher alguma informação e clicar no botão salvar. Para vermos os dados da foto, precisamos abrir o console do navegador com F12 (CMD + SHIFT + C, no MAC):



Funciona! Agora só nos resta enviar os dados capturados para uma rota **back-end** especializada nesta tarefa, usando o serviço $http. Porém, não é incomum validarmos os dados do usuário verificando a obrigatoriedade de algum campo ou aplicando alguma regra mais específica de validação. Em nossa aplicação não será diferente e faremos isso através do Angular.

## **Validando nosso formulário**

Quando queremos que o Angular tome conta da validação do formulário para nós, precisamos abdicar do sistema de validação do HTML5. Apesar de extremamente funcional, ele não se integra perfeitamente com o Angular e não é tão flexível quanto este último.

Para desabilitar a validação do HTML5, adicionamos o atributo **novalidate** na tag form:

<!-- public/partials/foto.html -->  
<!-- código anterior omitido -->  
  
<form novalidate name="formulario" class="row" ng-submit="submeter()">  
  
<!-- código posterior omitido -->

Pronto, agora vamos tornar todos os campos do nosso formulário obrigatórios, exceto o campo de descrição. Fazemos isso adicionando o mesmo atributo que é usado no HTML5, o atributo **required**:

<!-- public/partials/foto.html -->  
  
<div class="page-header text-center">  
 <h1>{{foto.titulo}}</h1>  
</div>  
  
<form novalidate name="formulario" class="row" ng-submit="submeter()">  
 <div class="col-md-6">  
 <div class="form-group">  
 <label>Título</label>  
 <input name="titulo" class="form-control"   
 ng-model="foto.titulo" required>   
 </div>  
 <div class="form-group">  
 <label>URL</label>  
 <input name="url" class="form-control"   
 ng-model="foto.url" required>  
 </div>  
 <div class="form-group">  
 <label>Descrição</label>  
 <textarea name="descricao" class="form-control" ng-model="foto.descricao">  
 </textarea>  
 </div>  
  
 <button type="submit" class="btn btn-primary">  
 Salvar  
 </button>  
 <a href="/" class="btn btn-primary">Voltar</a>  
 <hr>  
 </div>  
 <div class="col-md-6">  
 <minha-foto></minha-foto>  
 </div>  
</form>

Desligamos a validação do HTML, mas se deixarmos o título em branco e clicarmos em salvar, já teríamos que receber uma mensagem e não recebemos. Diferente do HTML, que já existe uma mensagem por padrão, o Angular precisa que você defina essa mensagem. A vantagem é que temos a flexibilidade de exibir mensagens de validação da forma que desejarmos.

Vamos adicionar, imediatamente após o campo título do nosso formulário, uma tag span com as classes form-control alert-danger

<!-- public/partials/foto.html -->  
  
<div class="page-header text-center">  
 <h1>{{foto.titulo}}</h1>  
</div>  
  
<form novalidate name="formulario" class="row" ng-submit="submeter()">  
 <div class="col-md-6">  
 <div class="form-group">  
 <label>Título</label>  
 <input name="titulo" class="form-control"   
 ng-model="foto.titulo" required>  
  
 <!-- novidade -->  
  
 <span class="form-control alert-danger">  
 Título obrigatório  
 </span>   
 </div>  
 <div class="form-group">  
 <label>URL</label>  
 <input name="url" class="form-control"   
 ng-model="foto.url" required>  
 </div>  
 <div class="form-group">  
 <label>Descrição</label>  
 <textarea name="descricao" class="form-control" ng-model="foto.descricao">  
 </textarea>  
 </div>  
  
 <button type="submit" class="btn btn-primary">  
 Salvar  
 </button>  
 <a href="/" class="btn btn-primary">Voltar</a>  
 <hr>  
 </div>  
 <div class="col-md-6">  
 <minha-foto></minha-foto>  
 </div>  
</form>

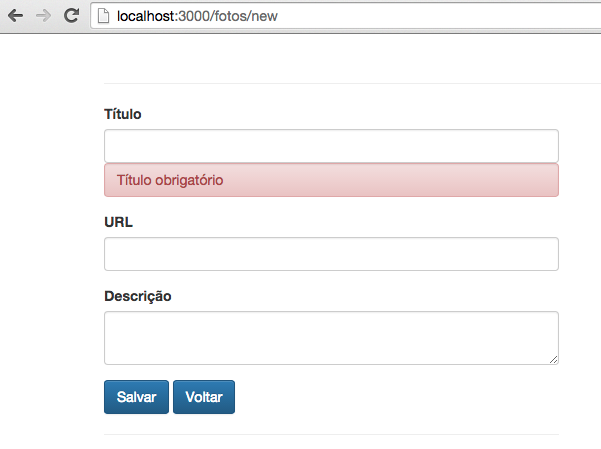
Ainda não funciona conforme esperado, porque se recarregarmos a página a mensagem de erro será exibida. Para que funcione, sua exibição deve ser condicional. Algo do tipo se o campo titulo é inválido, exiba a tag span. O Angular possui a diretiva ng-show, que permite a exibição condicional de elementos da tela. Quando seu valor é true, o elemento no qual a diretiva está aplicada é exibida, caso contrário não é exibido.

A questão toda é: quem fornecerá o valor da diretiva ng-show? A resposta mora em um objeto criado implicitamente que representa nosso formulário. Qual o nome deste objeto? Seu nome é o valor do atributoname do formulário, em nosso caso, **formulario**. É através dele que temos acesso a todos os campos do formulário, contanto que cada um deles também tenham definido um valor para o atributo name. Sendo assim, podemos fazer para o campo título:

// apenas exemplo, não entra em nenhum lugar por enquanto  
  
ng-show = "formulario.titulo.$error.required"

Acessamos formulario.titulo.$error, que nos dá acesso à interface de erros do Angular. Como queremos saber o status da validação required, usamos formulario.titulo.$error.required:

<!-- public/partials/foto.html -->  
  
<div class="page-header text-center">  
 <h1>{{foto.titulo}}</h1>  
</div>  
  
<form novalidate name="formulario" class="row" ng-submit="submeter()">  
 <div class="col-md-6">  
 <div class="form-group">  
 <label>Título</label>  
 <input name="titulo" class="form-control"   
 ng-model="foto.titulo" required>  
  
 <!-- novidade -->  
  
 <span ng-show = "formulario.titulo.$error.required"   
 class="form-control alert-danger">  
 Título obrigatório  
 </span>   
 </div>  
 <div class="form-group">  
 <label>URL</label>  
 <input name="url" class="form-control"   
 ng-model="foto.url" required>  
 </div>  
 <div class="form-group">  
 <label>Descrição</label>  
 <textarea name="descricao" class="form-control" ng-model="foto.descricao">  
 </textarea>  
 </div>  
  
 <button type="submit" class="btn btn-primary">  
 Salvar  
 </button>  
 <a href="/" class="btn btn-primary">Voltar</a>  
 <hr>  
 </div>  
 <div class="col-md-6">  
 <minha-foto></minha-foto>  
 </div>  
</form>



Hum, nosso formulário já exibe a mensagem de validação assim que é recarregado. Está certo? Depende do que desejamos. Se queremos exibir mensagens de erro sempre que um campo do formulário estiver errado, está certo. Porém, se quisermos validar os campos apenas quando o formulário for submetido, não. Aliás, vamos alterar o valor da diretiva ng-show para exibir a mensagem de erro de validação apenas quando o formulário for submetido. Basta adicionarmos mais uma condição, a formulario.$submitted que retorna verdadeiro apenas se o formulário foi submetido.

<!-- public/partials/foto.html -->  
<!-- código anterior omitido -->  
  
 <span ng-show = "formulario.$submitted && formulario.titulo.$error.required"   
 class="form-control alert-danger">  
 Título obrigatório  
 </span>   
<!-- código posterior omitido -->

Veja que agora ng-show só exibirá o elemento se as duas condições forem verdadeiras: o formulário for submetido e o campo inválido. Agora, ao recarregar a página, nosso formulário só será validado quando salvarmos o formulário, ação que disparará sua submissão.

Vamos deixar agora campo URL obrigatório e também preparar a mensagem de erro:

<!-- public/partials/foto.html -->  
  
<div class="page-header text-center">  
 <h1>{{foto.titulo}}</h1>  
</div>  
  
<form novalidate name="formulario" class="row" ng-submit="submeter()">  
 <div class="col-md-6">  
 <div class="form-group">  
 <label>Título</label>  
 <input name="titulo" class="form-control"   
 ng-model="foto.titulo" required>  
 <span ng-show = "formulario.$submitted && formulario.titulo.$error.required"   
 class="form-control alert-danger">  
 Título obrigatório  
 </span>   
 </div>  
 <div class="form-group">  
 <label>URL</label>  
 <input name="url" class="form-control"   
 ng-model="foto.url" required>  
  
 <!-- novidade -->  
  
 <span ng-show = "formulario.$submitted && formulario.url.$error.required"   
 class="form-control alert-danger">  
 URL obrigatória  
 </span>   
 </div>  
 <div class="form-group">  
 <label>Descrição</label>  
 <textarea name="descricao" class="form-control" ng-model="foto.descricao">  
 </textarea>  
 </div>  
  
 <button type="submit" class="btn btn-primary">  
 Salvar  
 </button>  
 <a href="/" class="btn btn-primary">Voltar</a>  
 <hr>  
 </div>  
 <div class="col-md-6">  
 <minha-foto></minha-foto>  
 </div>  
</form>

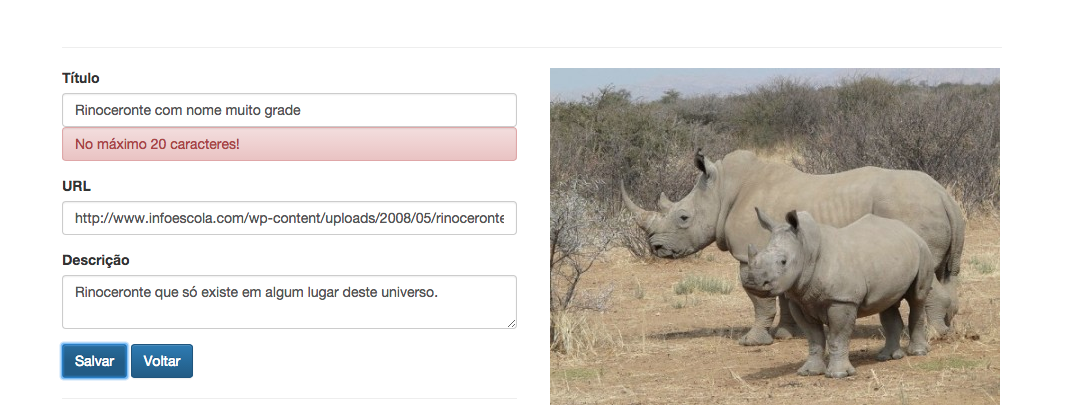
Excelente, mas o Angular permite fazer mais do que simplesmente considerar a obrigatoriedade de um campo. Existem diretivas específicas para validação. Por exemplo, vamos estipular que o campo título não pode passar de 20 caracteres através da diretiva **ng-maxlength**.

<!-- public/partials/foto.html -->  
  
<div class="page-header text-center">  
 <h1>{{foto.titulo}}</h1>  
</div>  
  
<form novalidate name="formulario" class="row" ng-submit="submeter()">  
 <div class="col-md-6">  
 <div class="form-group">  
 <label>Título</label>  
 <input name="titulo" class="form-control"   
 ng-model="foto.titulo" required  
 ng-maxlength="20">  
 <span ng-show = "formulario.$submitted && formulario.titulo.$error.required"   
 class="form-control alert-danger">  
 Título obrigatório  
 </span>   
 <span ng-show="formulario.$submitted && formulario.titulo.$error.maxlength" class="form-control alert-danger">  
 No máximo 20 caracteres!  
 </span>   
 </div>  
 <div class="form-group">  
 <label>URL</label>  
 <input name="url" class="form-control"   
 ng-model="foto.url" required>  
  
 <span ng-show = "formulario.$submitted && formulario.url.$error.required"   
 class="form-control alert-danger">  
 URL obrigatória  
 </span>   
  
 </div>  
 <div class="form-group">  
 <label>Descrição</label>  
 <textarea name="descricao" class="form-control" ng-model="foto.descricao">  
 </textarea>  
 </div>  
  
 <button type="submit" class="btn btn-primary">  
 Salvar  
 </button>  
 <a href="/" class="btn btn-primary">Voltar</a>  
 <hr>  
 </div>  
 <div class="col-md-6">  
 <minha-foto></minha-foto>  
 </div>  
</form>

Vamos aproveitar e fazer o data binding da URL da foto com nossa diretiva minha-foto:

<!-- public/partials/foto.html -->  
<!-- código anterior omitido -->  
  
<div class="col-md-6">  
 <minha-foto url="{{foto.url}}" titulo="{{foto.titulo}}">  
 </minha-foto>  
 </div>  
</form>

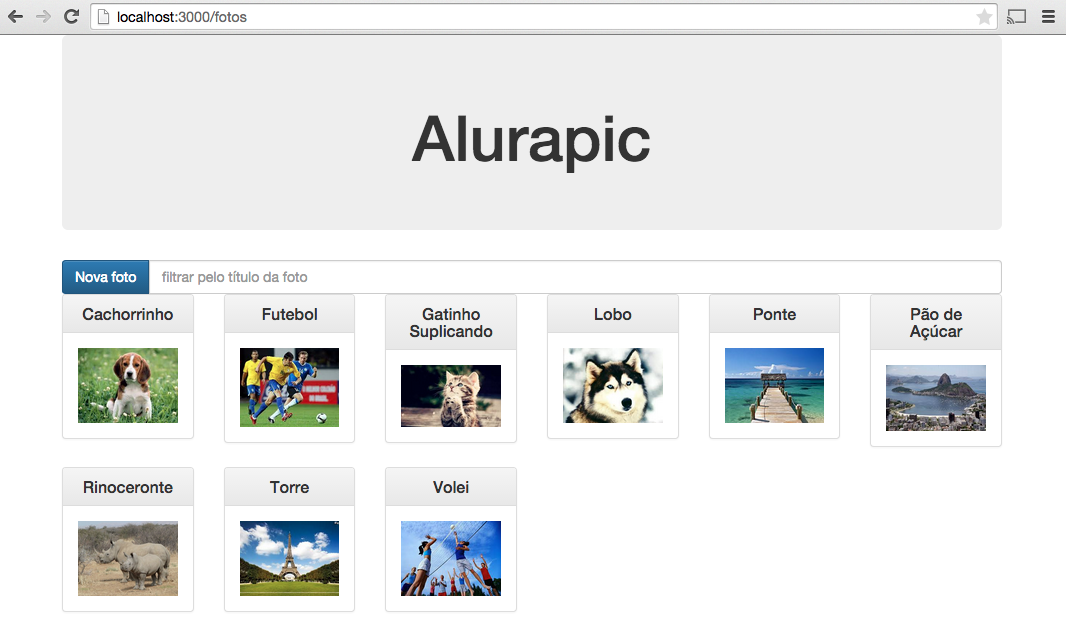
Agora vamos testar nossa validação. Vamos aproveitar e digitar uma URL válida, isso fará com que nossa diretiva minha-foto exibe a foto instantaneamente:



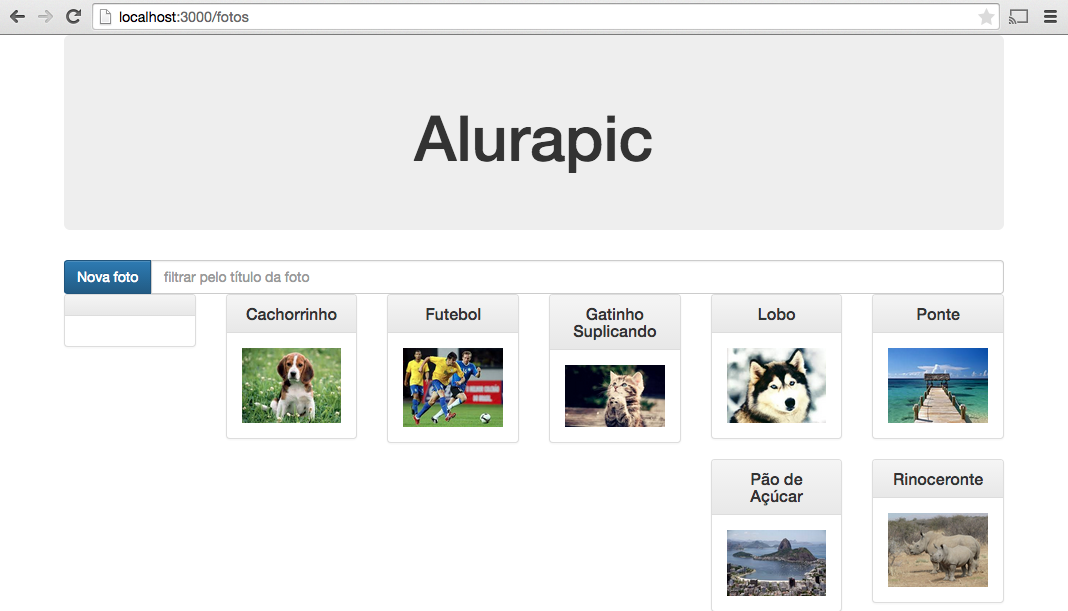
Perfeito! Agora já podemos alterar a função $scope.submeter e utilizar o serviço $http para gravar nosso produto. Como já dizemos, pedimos $http ao serviço de injeção de dependências do Angular. Como queremos enviar os dados, usamos $http.post, que recebe dois parâmetros. O primeiro é a URL do nosso server,/v1/fotos, e como segundo os dados que serão enviados, no caso, $scope.fotos. O restante é igual já fizemos:

// public/js/controllers/foto-controller.js  
  
angular.module('alurapic')  
 .controller('FotoController', function($scope, $http) {  
  
 $scope.foto = {};  
  
 $scope.submeter = function() {  
  
 $http.post('/v1/fotos', $scope.foto)  
 .success(function() {  
 console.log('Foto adicionada com sucesso');  
 })  
 .error(function(erro) {  
 console.log('Não foi possível cadastrar a foto');  
 })  
 };  
  
 });

Já podemos testar. Quando digitarmos dados válidos e clicamos em salvar, nosso formulário continua preenchido e nem sequer recebemos uma mensagem de sucesso! Bom, atacaremos isso em breve, mas se as informações foram enviadas e salvas, basta clicarmos no botão voltar e verificarmos se nossa foto aparece na lista, e realmente aparece!



Muito perfeito! Mas se clicarmos no botão salvar com dados inválidos? Por exemplo, com o tamanho do título sendo maior do que 20 caracteres e deixando o campo URL em branco? O resultado será este:



"Oh my god!", os dados inválidos do nosso formulário foram submetidos e uma foto cadastrada indevidamente. Mas por que o nome ficou em branco? Isso porque dados consideramos inválidos pelo Angular não são aplicados do formulário para o atributo relacionado em $scope. Então, no momento do envio dos dados,$scope.foto.titulo ficou vazio.

Para resolver isso, basta consultarmos em nosso $scope o status do formulário com a sintaxe**$scope.formulario.$valid**. Isso mesmo, através de $scope podemos acessar nosso formulário através do seuname e perguntarmos se ele é válido consultando a propriedade $valid.

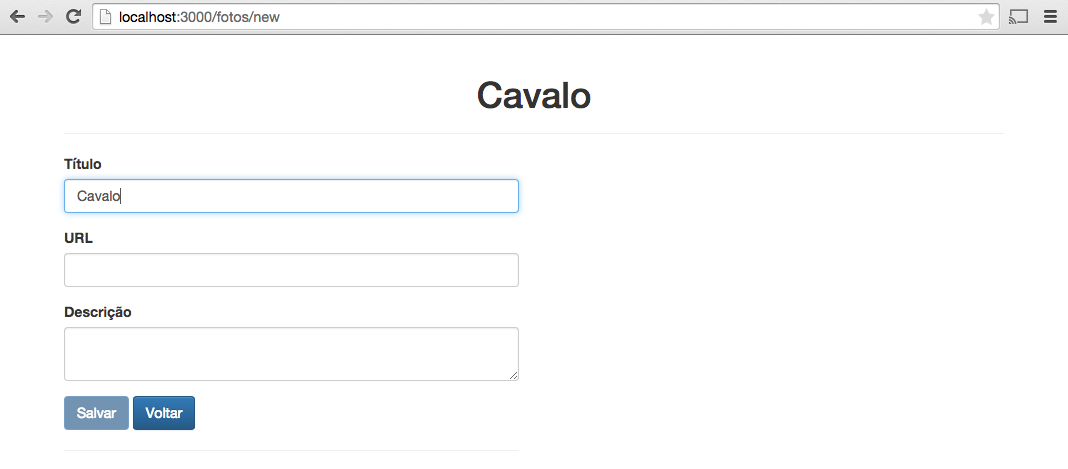
Alterando nosso FotoController:

// public/js/controllers/foto-controller.js  
  
angular.module('alurapic')  
 .controller('FotoController', function($scope, $http) {  
  
 $scope.foto = {};  
  
 $scope.submeter = function() {  
  
 if ($scope.formulario.$valid) {  
  
 $http.post('/v1/fotos', $scope.foto)  
 .success(function() {  
 console.log('Foto adicionada com sucesso');  
 })  
 .error(function(erro) {  
 console.log('Não foi possível cadastra a foto');  
 })  
 }  
 };  
  
 });

Agora, nossa lógica de envio das informações só será executada caso o formulário seja válido. Antes de testarmos, podemos melhorar ainda mais a experiência do usuário habilitando a exibição do botão salvarapenas se o formulário estiver válido.

<!-- public/partials/foto.html -->  
  
<!-- código anterior omitido -->  
 <button type="submit" class="btn btn-primary" ng-disabled="formulario.$invalid">  
 Salvar  
</button>  
<!-- código posterior omitido -->

A cada interação do usuário com nosso formulário, a diretiva ng-disabled consultará o status do formulário para saber se ele é invalido, caso seja, o botão ficará desabilitado:



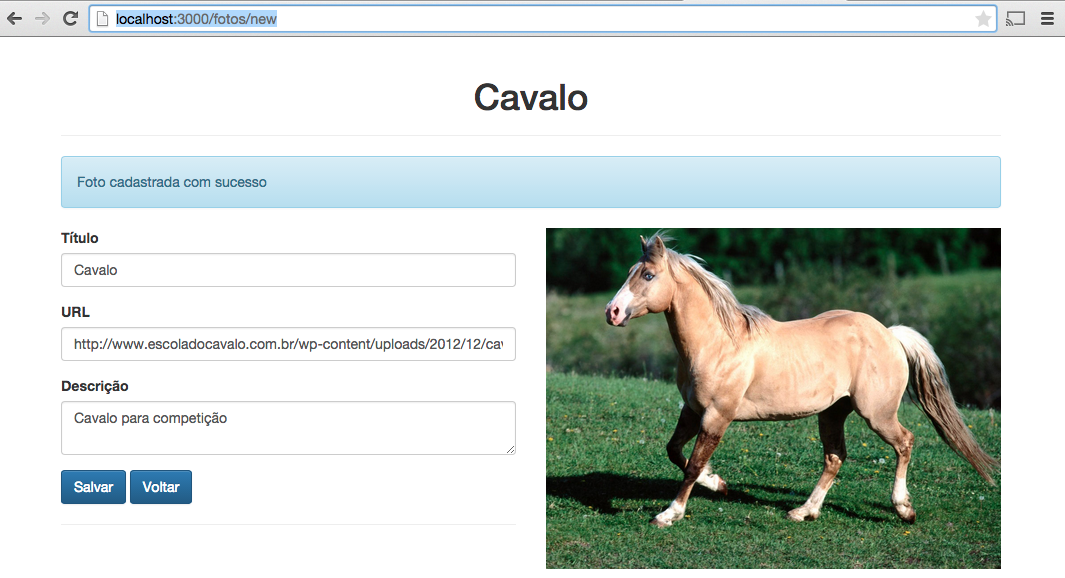
Agora, para deixar ainda melhor nosso formulário, vamos exibir uma mensagem de fracasso ou sucesso para indicar o status da operação com o servidor. Inclusive vamos limpar os dados do formulário quando a operação for bem sucedida. Vamos adicionar um parágrafo que consultará $scope.mensagem. O parágrafo só será exibido se exibir alguma mensagem:

<div class="page-header text-center">  
 <h1>{{foto.titulo}}</h1>  
</div>  
  
<!-- novidade! Aqui será exibida mensagens para o usuário -->  
  
<p ng-show="mensagem.length" class="alert alert-info">{{mensagem}}</p>  
  
<form novalidate name="formulario" class="row" ng-submit="submeter()">  
 <div class="col-md-6">  
 <div class="form-group">  
 <label>Título</label>  
 <input name="titulo" class="form-control"   
 ng-model="foto.titulo" required  
 ng-maxlength="20">  
 <span ng-show = "formulario.$submitted && formulario.titulo.$error.required"   
 class="form-control alert-danger">  
 Título obrigatório  
 </span>   
 <span ng-show="formulario.$submitted && formulario.titulo.$error.maxlength" class="form-control alert-danger">  
 No máximo 20 caracteres!  
 </span>   
 </div>  
 <div class="form-group">  
 <label>URL</label>  
 <input name="url" class="form-control"   
 ng-model="foto.url" required>  
  
 <span ng-show = "formulario.$submitted && formulario.url.$error.required"   
 class="form-control alert-danger">  
 URL obrigatória  
 </span>   
  
 </div>  
 <div class="form-group">  
 <label>Descrição</label>  
 <textarea name="descricao" class="form-control" ng-model="foto.descricao">  
 </textarea>  
 </div>  
  
 <button type="submit" class="btn btn-primary" ng-disabled="formulario.$invalid">  
 Salvar  
 </button>  
 <a href="/" class="btn btn-primary">Voltar</a>  
 <hr>  
 </div>  
 <div class="col-md-6">  
 <minha-foto url="{{foto.url}}" titulo="{{foto.titulo}}">  
 </minha-foto>  
 </div>  
</form>

Agora, alterando nosso controller para popular a $scope.mensagem:

// public/js/controllers/foto-controller.js  
  
angular.module('alurapic')  
 .controller('FotoController', function($scope, $http) {  
  
 $scope.foto = {};  
 $scope.mensagem = '';  
  
 $scope.submeter = function() {  
  
 if ($scope.formulario.$valid) {  
  
 $http.post('/v1/fotos', $scope.foto)  
 .success(function() {  
 $scope.mensagem = 'Foto cadastrada com sucesso';  
 })  
 .error(function(erro) {  
 console.log(erro);  
 $scope.mensagem = 'Não foi possível cadastrar a foto';  
 })  
 }  
 };  
  
 });

Muito bem, agora basta cadastrarmos uma nova foto e verificar a exibição da nossa mensagem:



Quase lá! Precisamos limpar o formulário quando a mensagem for adicionada com sucesso. Basta atribuirmos um objeto vazio à $scope.foto:

angular.module('alurapic')  
 .controller('FotoController', function($scope, $http) {  
  
 $scope.foto = {};  
 $scope.mensagem = '';  
  
 $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';  
 })  
 }  
 };  
  
 });

Agora, para cada foto cadastrada com sucesso, o formulário será limpo automaticamente, tudo por causa do data binding do Angular.

## **O que aprendemos neste capítulo?**

* navegação entre views
* a diretiva ng-submit e a interface de eventos do Angular
* executar uma ação no controller através de ações do usuário
* validação de formulário com Angular
* envio de dados para o servidor com $http.post
* mensagens de aviso para o usuário