Tutorial: Ejemplo de aplicación web con Angular JS y una API REST con Node[1]

Sin duda este va a ser el año del despegue de AngularJS[2] y hay que ponerse las pilas. Por ello, siguiendo los pasos de **éste tutorial de Scotch.io[3]**, Este tutorial nos servirá para empezar con algo sencillo pero que toca bastantes aspectos de Angular en la parte Frontend de una *Single Page Application*[4], añadiendo un API con Node[5] y Mongo para la parte Backend y completando así el Stack tecnológico de moda, MEAN[6].

Vamos a ello!

La estructura de archivos va a ser muy sencilla, no vamos a modularizar ni añadir tareas con Grunt[7] para enfocarnos en los conceptos de Angular. Estos son los ficheros que tendremos

```
- public
----- index.html
----- main.js
server.js
package.json
```

- **main.js** contendrá toda la lógica del frontend, es donde tendremos los controladores de Angular JS y llamaremos via AJAX al API para pedir contenido, borrarlo, etc..
- **index.html** será nuestro único fichero html y por tanto nuestra única página, toda la funcionalidad será en ella.
- **server.js** es nuestro fichero Node donde estará la configuración del servidor y las rutas a nuestro API.

- **package.json** es el fichero donde están los datos de la aplicación y las dependencias utlizadas, como toda aplicación Node.

Empezaremos por **package.json** para indicar que dependencias vamos a necesitar, que simplemente serán *Express* y *Mongoose*:

```
{
    "name": "angular-todo",
    "version": "0.0.1",
    "description": "Simple Angular TODO app based in MEAN stack",
    "main": "server.js",
    "dependencies": {
        "express": "~3.x",
        "mongoose": "latest"
    }
```

Después de esto, en una terminal ejecutamos "npm install" y se nos instalarán las dependencias para poder empezar a utilizarlas.

Ahora pasaremos al archivo **server.js** que será el fichero donde esté la configuración del servidor, así como la conexión a la base de datos y las rutas de nuestro API.
En los comentarios del código he explicado a grandes rasgos que hace cada línea.

Como todo fichero de servidor de Node, primero añadimos las librerías que necesitamos (express y mongoose).

```
//server.js

 express        = require('express');
 app            = express();
 mongoose       = require('mongoose');

// Conexión con la base de datos
mongoose.connect('mongodb://localhost:27017/angular-todo');

// Configuración
app.configure(function {
```

```
    // Localización de los ficheros estáticos
    app.use(express.static(__dirname + '/public'));
    // Muestra un log de todos los request en la consola
    app.use(express.logger('dev'));
    // Permite cambiar el HTML con el método POST
    app.use(express.bodyParser());
    // Simula DELETE y PUT
    app.use(express.methodOverride());
});

// Escucha en el puerto 8080 y corre el server
app.listen(, function {
    console.log('App listening on port 8080');
});
```

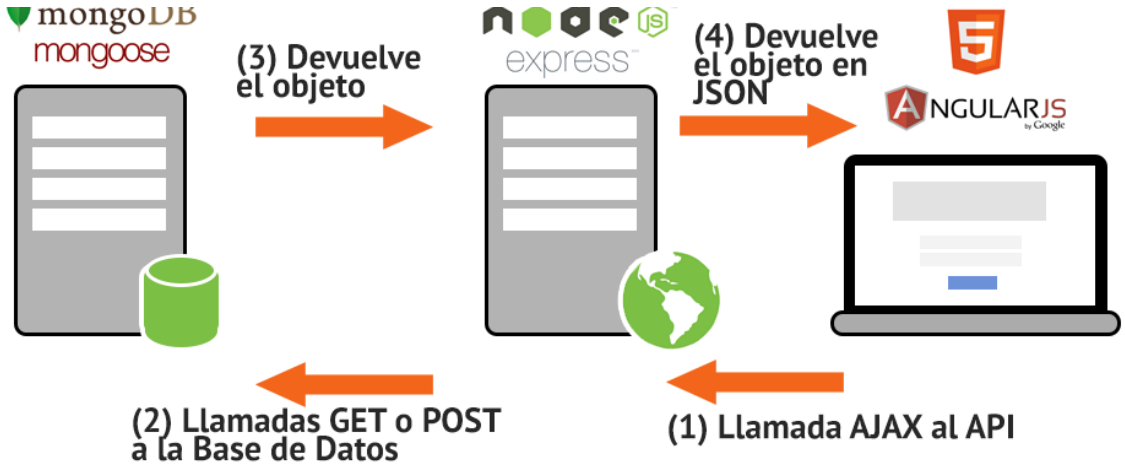
El siguiente paso es construir el modelo de la base de datos que modele las tareas o "ToDos". Esto lo hacemos con *Mongoose* y nuestro modelo será muy sencillo ya que solo cuenta con un atributo "Text" que define la tarea. Este código lo insertamos en **server.js** antes de la línea donde se inicia el servidor con app.listen

```
// Definición de modelos
 Todo = mongoose.model('Todo', {
    text: String
});
```

Tras esto nos queda construir las rutas que llamarán a nuestro API y que utilizaremos desde el frontend. En esta tabla se muestran las 3 llamadas que vamos a implementar y que definirán nuestra API:

HTTP	URL	Descripción
GET	/api/todos	Devuelve todas las tareas de la BD
POST	/api/todos	Crea una tarea
DELETE	/api/todos/:todo	Borra una tarea

Y antes de seguir con el código, veamos un diagrama visual del flujo que va a seguir aplicación con las tecnologías que empleamos en cada parte:



Desde el frontend, con Angular hacemos llamadas AJAX a nuestra API en el servidor Node. Este consulta a la base de datos (Mongo) dependiendo de la llamada realizada. La BD devuelve el objeto como respuesta a Node y este lo sirve como JSON a Angular que lo muestra en el frontend sin necesidad de recargar la página, creando así una *Single Page Application*.

Veamos las rutas. Estas irán también en el archivo server.js, justo antes de cuando se inicia el servidor y escucha en el puerto con app.listen

```
// Rutas de nuestro API
// GET de todos los TODOs
app.get('/api/todos', function(req, res) {
    Todo.find(function(err, todos) {
        (err) {
            res.send(err);
        }
        res.json(todos);
    });
});

// POST que crea un TODO y devuelve todos tras la creación
app.post('/api/todos', function(req, res) {
    Todo.create({
        text: req.body.text,
        done: false
```

```
    }, function(err, todo){
        (err) {
            res.send(err);
        }

        Todo.find(function(err, todos) {
            (err){
                res.send(err);
            }
            res.json(todos);
        });
    });
});

// DELETE un TODO específico y devuelve todos tras borrarlo.
app.delete('/api/todos/:todo', function(req, res) {
    Todo.remove({
        _id: req.params.todo
    }, function(err, todo) {
        (err){
            res.send(err);
        }

        Todo.find(function(err, todos) {
            (err){
                res.send(err);
            }
            res.json(todos);
        });

    })
});

// Carga una vista HTML simple donde irá nuestra Single App Page
```

```
// Angular Manejará el Frontend
app.get(, function(req, res) {
    res.sendfile('./public/index.html');
});
```

Gracias a Mongoose podemos buscar(**find**), borrar (**remove**) y crear(**create**) de una manera muy sencilla. La última ruta no corresponde al API, si no que será la encargada de mostrar el html donde ejecutaremos toda la lógica del Frontend.

Todo esto que hemos hecho corresponde al Backend de la aplicación. Ahora empezaremos con lo que de verdad importa, el desarrollo frontend con Angular.

Tendremos toda la lógica en el fichero **main.js,** Primero crearemos un modulo que será el que defina toda nuestra aplicación

```
 angularTodo = angular.module('angularTodo', []);
```

y seguidamente la función mainController que será el controlador de la aplicación

```
function mainController($scope, $http) {
    $scope.formData = {};

    // Cuando se cargue la página, pide del API todos los TODOs
    $http.get('/api/todos')
        .success(function(data) {
            $scope.todos = data;
            console.log(data)
        })
        .error(function(data) {
            console.log('Error: ' + data);
        });

    // Cuando se añade un nuevo TODO, manda el texto a la API
    $scope.createTodo = function{
        $http.post('/api/todos', $scope.formData)
            .success(function(data) {
                $scope.formData = {};
```

```
                $scope.todos = data;
                console.log(data);
            })
            .error(function(data) {
                console.log('Error:' + data);
            });
    };

    // Borra un TODO despues de checkearlo como acabado
    $scope.deleteTodo = function {
        $http.delete('/api/todos/' + id)
            .success(function(data) {
                $scope.todos = data;
                console.log(data);
            })
            .error(function(data) {
                console.log('Error:' + data);
            });
    };
}
```

Pasemos a explicar algunos conceptos de esta parte.

En el objeto **$scope** se almacenan todas las variables dentro del ámbito del controlador. En el HTML, todo lo que se encuentre dentro de la directiva **ng-controller=”mainController”** es controlable desde el objeto **$scope**.

Y el objeto **$http** es el que hace toda la magia, ya que nos permite hacer llamadas AJAX a nuestro API con pocas líneas de código.

Con estos dos objetos creamos las 3 funciones que hacen las 3 peticiones que acepta nuestra API, el **GET** de todas las tareas almacenadas, el **POST** de creación de una nueva tarea y el **DELETE** de una tarea.

Y por último nos queda el **HTML** en el que maquetaremos los resultados que nos trae el API.
Necesitamos indicar que parte de la página corresponde a la aplicación Angular, eso lo

hacemos con la directiva **ng-app**. En nuestro caso lo hemos puesto en el tag **<html>** ya que todo el HTML es la aplicación

```
1. <html  ng-app"angularTodo"...</html>
```

```
<html lang="en" ng-app="angularTodo">...</html>
```

Una aplicación Angular puede tener varios controladores, en este ejemplo solo tenemos uno, el **mainController**, y debemos decir en el HTML que parte es la corresponde a esta función, eso lo hacemos con la directiva **ng-controller**. En nuestro caso la hemos puesto en el body porque no hay más. Si tuviesemos más controladores, se pueden poner en otros **section**, **article** o **div** y tener varios en la página.

```
1. <body ng-controller"mainController"...</body>
```

```
<body ng-controller="mainController">...</body>
```

Para mostrar la lista de tareas que devuelve el GET, utilizaremos la directiva **ng-repeat** que nos permite crear una iteración al estilo de un **for**

```
1. <div class"checkbox" ng-repeat"todo in todos"
2. <label>
3. <input "checkbox" ng-click"deleteTodo(todo._id)"
4. {{ todo.text }}
5. </label>
6. </div>
```

```
<div class="checkbox" ng-repeat="todo in todos">
<label>
<input type="checkbox" ng-click="deleteTodo(todo._id)">
{{ todo.text }}
</label>
</div>
```

Con esto creamos un input de tipo checkbox por cada objeto que nos devuelve la llamada al API. Y con la directiva **ng-click** creamos un evento que escucha cuando marquemos el checkbox para llamar a la función deleteTodo() a la cual se le pasa como parámetro el id de la tarea para que llame al DELETE del API.

Por último, tenemos un formulario con un input de tipo texto donde escribimos tareas nuevas y las mandamos por POST al API. Aquí usamos una nueva directiva, **ng-model**, que es la que controla el Modelo en este caso la tarea y su texto, y de nuevo **ng-click** en el botón de submit para llamar a la función **createTodo()** del controlador que hace el POST al API y a la Base de datos.

```
1. <form>
2. <div class"form-group"
3. <input "text" class"form-control input-lg text-center"
4. placeholder"Inserta una tarea nueva"
5. ng-model"formData.text"
6. </div>
7. <button class"btn btn-primary btn-lg" ng-click"createTodo()"Añadir</button>
8. </form>
```

```
<form>
<div class="form-group">
<input type="text" class="form-control input-lg text-center"
placeholder="Inserta una tarea nueva"
ng-model="formData.text">
</div>
<button class="btn btn-primary btn-lg" ng-click="createTodo()">Añadir</button>
</form>
```

Con esto estaría todo. Solo nos queda incluir las librerías de **jQuery** y **Angular** como scripts al final de la página y y también una hoja de estilos para que no sea tan fea la aplicación.

Hemos usado un CDN[8] para ello y así no tenemos que preocuparnos en bajarnos la libreria y añadirla al proyecto, para centrarnos en entender los conceptos

El código HTML completo sería así:

```
1. <!doctype html>
2. <html  ng-app"angularTodo"
3. <head>
4. <meta charset"UTF-8"
5. <meta "viewport" content"width=device-width, initial-scale=1"
6. <title>Angular TODO app</title>
7. <link "stylesheet" "//netdna.bootstrapcdn.com/bootstrap/3.0.3/css/bootstrap.min.css"
```

```
</head>
<body ng-controller"mainController"
<div class"container"
<!--Cabecera-->
<div class"jumbotron text-center"
<h1>Angular TODO List <span class"label label-info"{{ todos.length }}</span></h1>
</div>
<!--Lista de Todos-->
<div "todo-list" class"row"
<div class"col-sm-4 col-sm-offset-4"
<div class"checkbox" ng-repeat"todo in todos"
<label>
<input "checkbox" ng-click"deleteTodo(todo._id)"
{{ todo.text }}
</label>
</div>
</div>
</div>
<!--Formulario para insertar nuevos Todo-->
<div "todo-form" class"row"
<div class"col-sm-8 col-sm-offset-2 text-center"
<form>
<div class"form-group"
<input "text" class"form-control input-lg text-center"
placeholder"Inserta una tarea nueva"
ng-model"formData.text"
</div>
<button class"btn btn-primary btn-lg" ng-click"createTodo()"Añadir</button>
</form>
</div>
</div>
</div>
<script "//ajax.googleapis.com/ajax/libs/jquery/2.0.3/jquery.min.js"></script>
<script "//ajax.googleapis.com/ajax/libs/angularjs/1.0.8/angular.min.js"></script>
<script "main.js"></script>
</body>
</html>
```

```
<!doctype html>
```

```
<html lang="en" ng-app="angularTodo">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Angular TODO app</title>
<link rel="stylesheet" href="//netdna.bootstrapcdn.com/bootstrap/3.0.3/css/bootstrap.min.css">
</head>
<body ng-controller="mainController">
<div class="container">
<!--Cabecera-->
<div class="jumbotron text-center">
<h1>Angular TODO List <span class="label label-info">{{ todos.length }}</span></h1>
</div>

<!--Lista de Todos-->
<div id="todo-list" class="row">
<div class="col-sm-4 col-sm-offset-4">
<div class="checkbox" ng-repeat="todo in todos">
<label>
<input type="checkbox" ng-click="deleteTodo(todo._id)">
{{ todo.text }}
</label>
</div>
</div>
</div>

<!--Formulario para insertar nuevos Todo-->
<div id="todo-form" class="row">
<div class="col-sm-8 col-sm-offset-2 text-center">
<form>
<div class="form-group">
<input type="text" class="form-control input-lg text-center"
placeholder="Inserta una tarea nueva"
ng-model="formData.text">
</div>
<button class="btn btn-primary btn-lg" ng-click="createTodo()">Añadir</button>
</form>
</div>
</div>
```

```
</div>

<script src="//ajax.googleapis.com/ajax/libs/jquery/2.0.3/jquery.min.js"></script>
<script src="//ajax.googleapis.com/ajax/libs/angularjs/1.0.8/angular.min.js"></script>
<script src="main.js"></script>
</body>
</html>
```

Y ya tenemos nuestra aplicación de ToDos. Solo tenemos que correr el servidor en un terminal con node server.js e ir a un navegador a la URL http://localhost:8080 y tendremos algo como esto

Puedes ver una versión en inglés de este ejemplo en el Blog Scotch.io[9], escrito por

Chris Sevilleja[10].

The following two tabs change content below.

cazaustre[11]

Front End Developer at Chefly[12]

Desarrollador Front-End y Diseñador Gráfico Freelance. Apasionado de la tecnología HTML5 y el mundo JavaScript. Geek, adicto a las series y a las camisetas.

1. http://carlosazaustre.es/blog/tutorial-ejemplo-de-aplicacion-web-con-angular-js-y-api-rest-con-node/
2. http://carlosazaustre.es/blog/empezando-con-angular-js/
3. http://scotch.io/tutorials/javascript/creating-a-single-page-todo-app-with-node-and-angular
4. http://en.wikipedia.org/wiki/Single-page_application
5. http://carlosazaustre.es/blog/como-crear-una-api-rest-usando-node-js/
6. http://mean.io/
7. http://carlosazaustre.es/blog/automatizar-tareas-en-javascript-con-grunt-js/
8. http://cdnjs.com/
9. http://scotch.io/tutorials/javascript/creating-a-single-page-todo-app-with-node-and-angular
10. https://twitter.com/sevilayha/
11. http://carlosazaustre.es/
12. http://chefly.co/