

Índice

1 Herramientas 3

1.1 Herramientas necesarias 3

1.2 Configuración de herramientas 3

2 ANGULARJS 4

2.1 Descarga e Instalación 4

2.2 Tipos de Builds 4

3 HOLA MUNDO EN ANGULARJS 5

3.1 Definición 5

4 estructura de un proyecto 6

4.1 Definición 6

5 instalación de módulos 7

5.1 Definición 7

6 DIRECTIVAS 8

7 ANÁLISIS DE FICHEROS 9

8 Servicios 10

9 API REST 11

10 Login 12

11 Spinner 13

12 Ventanas modales 14

# Herramientas

## Herramientas necesarias

|  |  |
| --- | --- |
| Chrome | <https://www.google.com/chrome/browser/desktop/> |
| SublimeText | <http://www.sublimetext.com/3> |
| XAMPP | <https://www.apachefriends.org/es/index.html> |
| Package Control | <https://packagecontrol.io/installation> |

## Configuración de herramientas

En SublimeText, pulsamos en View 🡪 Show Console, pegamos el código y pulsamos ENTER

|  |
| --- |
| import urllib.request,os,hashlib; h = '6f4c264a24d933ce70df5dedcf1dcaee' + 'ebe013ee18cced0ef93d5f746d80ef60'; pf = 'Package Control.sublime-package'; ipp = sublime.installed\_packages\_path(); urllib.request.install\_opener( urllib.request.build\_opener( urllib.request.ProxyHandler()) ); by = urllib.request.urlopen( 'http://packagecontrol.io/' + pf.replace(' ', '%20')).read(); dh = hashlib.sha256(by).hexdigest(); print('Error validating download (got %s instead of %s), please try manual install' % (dh, h)) if dh != h else open(os.path.join( ipp, pf), 'wb' ).write(by) |

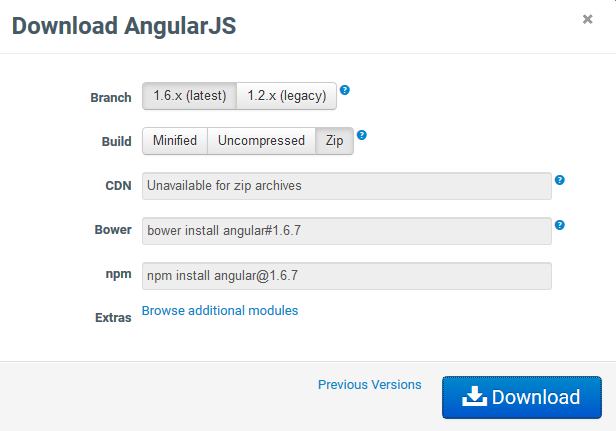
Ctrl + Shift + P 🡪 Install Package

Ctrl + Shift + P 🡪 AngularJS Snippets (Este nos muestra la ayuda de Angular cuando estemos desarrollando)

# AngularJS

## Descarga e Instalación

Descargaremos AngularJS desde la página: <https://angularjs.org/> Pulsamos sobre **Download AngularJS**



## Tipos de Builds

* ***Minified*** – Librería AngularJS en su versión mínima / comprimida
* ***Uncompressed*** – Librería AngularJS con comentarios, más completa y documentada para un mayor entendimiento
* ***Zip*** – API AngularJS con módulos y empaquetado como ZIP

# BootStrap

## Descarga e Instalación

Descargaremos BootStrap desde la página: <https://getbootstrap.com/> Pulsamos sobre **Download 🡪 Compiled CSS and JS 🡪 Download**

Copiamos la carpeta **css**, **fonts** y **js** a la carpeta de nuestro **Ejercicio 1**. Sobrescribimos si nos pide confirmación.

* En css nos podemos quedar sólo con **bootstrap.min.css** y **bootstrap.css.map**
* En js nos podemos quedar sólo con **bootstrap.min.js** y **bootstrap.min.js.map**

# Instalación de Componentes

Si escribimos *npm help npm* en la consola de dos podemos ver la ayuda de **npm**.

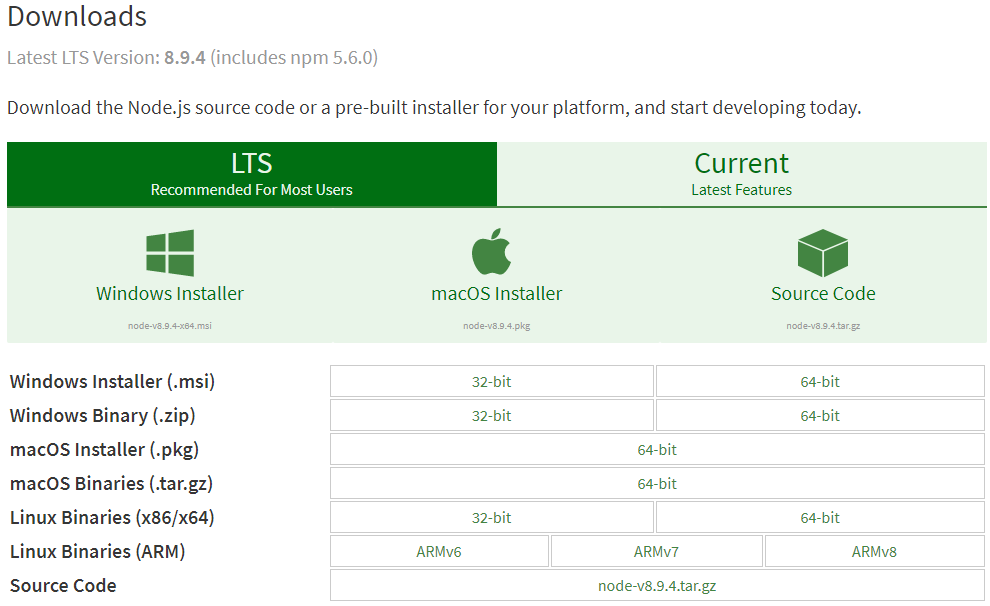
## Instalación de NPM

**npm** es el manejador de paquetes por defecto para **Node.js,** que es un entorno de ejecución para JavaScript construido con el [motor de JavaScript V8 de Chrome](https://developers.google.com/v8/).

**npm** se ejecuta desde la línea de comandos y maneja las dependencias para una aplicación. Está escrito en su totalidad en Javascript y el significado de las siglas es **Node Package Manager**.

Para instalar Node.js npm iremos a la web <https://nodejs.org/en/download/>

Y descargaremos la versión correspondiente.



## Instalar un Paquete con npm

*npm install <package-name>****--save***

**--save** hace que se incluya la dependencia en el **package.json**

## Instalación de Bower

**bower** es un gestor de paquetes para nuestro proyecto. Esto es, **bower** se va a encargar de descargar, instalar y tener organizadas todas nuestras dependencias.

*npm install bower*

## Instalar un paquete con bower

*bower install <package-name>* ***--save***

**--save** hace que se incluya la dependencia en el **bower.json**

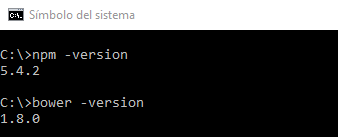
## Comprobando la instalación

Para confirmar que hemos realizado correctamente la instalación de npm y bower, abrimos una consola de dos y tecleamos.

npm –version

bower –version

Mostrará un mensaje con la versión instalada de cada uno.

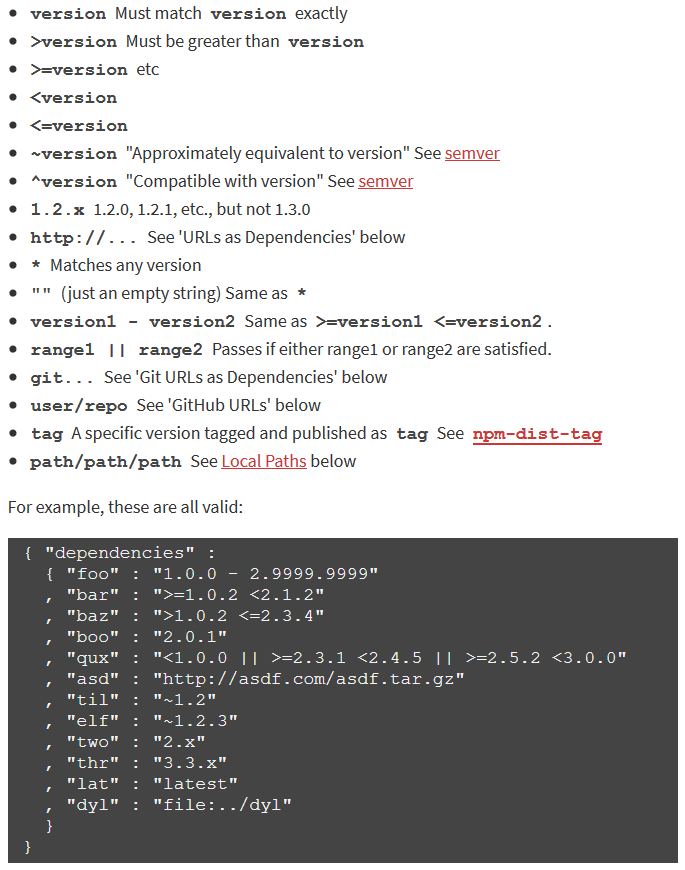


Para incluir a posteriori la dependencia en la app, es necesario incluir en el fichero js del componente el módulo instalado.

angular.module(‘myApp’, [‘package-module’]);

## Instalación de Módulos

Para la instalación de módulos vamos a tener varias posibilidades, veremos como instalar módulos con npm y con bower, como indicar con que versiones trabajar, etc..



## Instalación de Gulp

*npm install –g gulp --****save***

### Verificar que gulp está instalado correctamente

*gulp –v*

### Primera tarea con Gulp

<https://frontendlabs.io/1669--gulp-js-en-espanol-tutorial-basico-primeros-pasos-y-ejemplos>

Creamos una carpeta llamada: gulp-init, ingresamos a esa carpeta mediante terminal.

Luego allí dentro creamos nuestro archivo: gulpfile.js, que es el archivo que Gulp.js necesita para saber que tareas realizará y de momento no le podremos ningún contenido.

Escribimos npm init y pulsamos ***Enter*** para cada una de las preguntas que nos van a ir apareciendo. Con esto, **npm** nos debe haber creado un archivo llamado: package.json, que contendrá algo parecido a lo siguiente:

{

"name": "gulp-init",

"version": "1.0.0",

"description": "",

"main": "gulpfile.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "",

"license": "ISC"

}

# Hola mundo en AngularJS

## Ejercicio básico Hola Mundo

Como hemos ido viendo anteriormente, tenemos una carpeta llamada Ejercicio 1con el siguiente contenido:

* Crearemos un fichero **index.html** en el raíz.
* Incluir la directiva **ng-app** en la etiqueta *<html>*
* Incluimos la librería de **bootstrap**, en la etiqueta *<head>*

*<!-- Incluir la librería de Bootsrap -->*

*<link rel="stylesheet" href=”css/bootstrap.min.css”>*

* Incluimos la librería de **angular**, en la etiqueta *<head>*

*<!-- Incluir la librería de AngularJS -->*

*<script src="js/lib/angular.min.js"></script>*

* Incluimos contenido en el *<body>*

*<div class=”contanier”>*

*<h1>{{ “Hola Mundo en AngularJS” }}</h1>*

*</div>*

# Estructura de un Proyecto

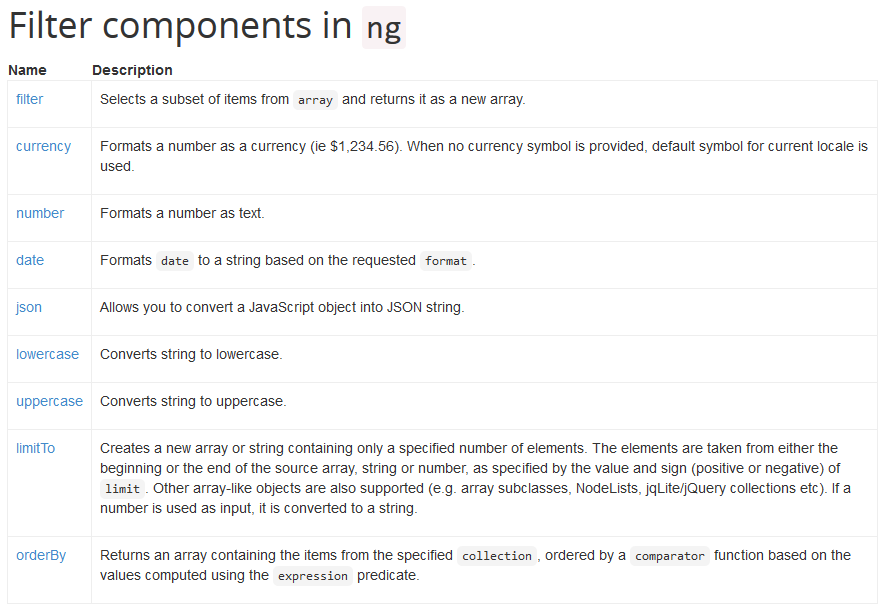
## Best Practices

Vamos a tratar de definir una serie de buenas prácticas para establecer una estructura ordenada y jerárquica en nuestros proyectos AngularJS, para ello nos vamos a basar en la estructura de un proyecto real, tendría la siguiente estructura:

A partir de este momento trabajaremos bajo este tipo de estructuras, en las que construiremos ejemplos de código que vamos a ir viendo detenidamente.

# Filtros

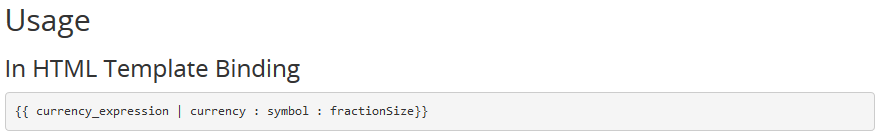
El uso de filtros nos facilitará el formateo de datos en la capa de presentación sin necesidad de tener que programar las funciones para tal fin. En definitiva no dejan de ser funciones ya construidas de las que podemos hacer uso desde el API de AngularJS.



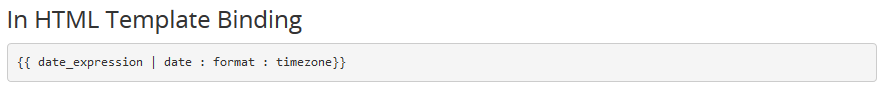
## Ejemplo de uso de filtros

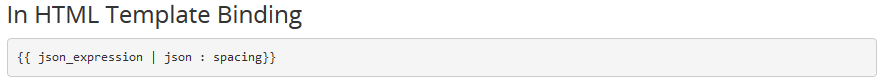
Para el uso de filtros, la sintaxis sería:

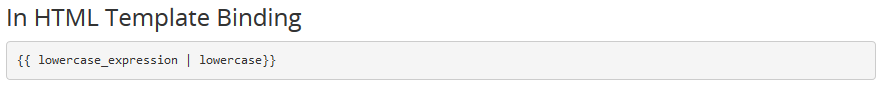
{{ expression [| filter\_name[:parameter\_value] ... ] }}

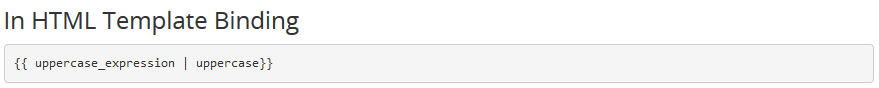








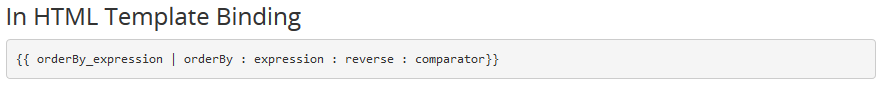




**Ej.**

*<h4>{{ “FerNanDo” | uppercase }}</h4>*





# Directivas

ng-model

ng-controller

ng-show

ng-hide

ng-click

ng-if

ng-repeat

ng-submit

ng-enabled

ng-disabled

ng-checked

# Análisis de Ficheros

## app.route.js

'use strict';

(function() {

angular

.module('app')

.config(routeConfig);

// function routeConfig($stateProvider, $urlRouterProvider){

//Se elimina el atributo stateProvider para no dar error en una regla de eslint

// no-unused-vars

function routeConfig($urlRouterProvider){

// For any unmatched url, send to /

$urlRouterProvider.otherwise('/home');

}

})();

## package.json

## bower.json

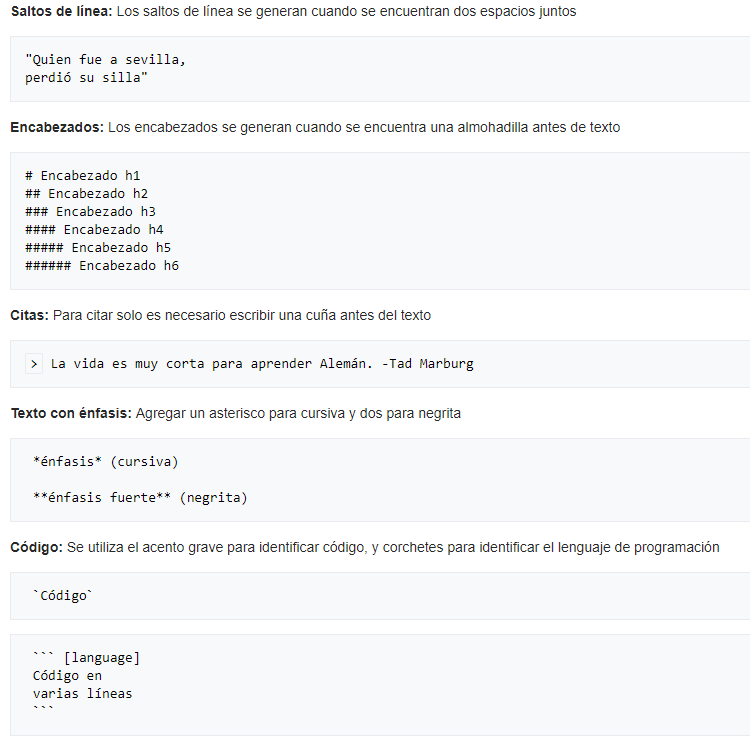
## index.html

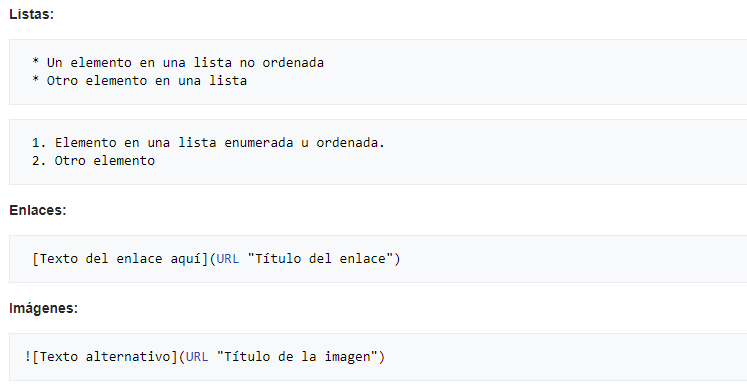
## gulpfile.js

## README.md

Un archivo **README** contiene información acerca de otros archivos en un directorio. Es una forma de documentación de software. La extensión **MD** significa [*M*](https://es.wikipedia.org/wiki/Markdown)*arkDown* o lenguaje de marcado ligero.

El nombre del archivo es generalmente escrito en mayúsculas. En GitHub cuando hay un fichero **README.md** este es convertido automáticamente y mostrado como HTML.





# MIFIDII - Costes y Gastos e Incentivos - Front v1.0.0.

Proyecto MIFIDII Costes y Gastos.

# Requirements

Features to starting project:

\* Install [NodeJS]( https://nodejs.org/en/)

\* Install [Bower](https://bower.io/)

How to install with proxy environment:

prerequisites.md#proxy-setup.

# Getting Started

## Installing

Download project from GIT repository

```

git clone https://gitlab.alm.gsnetcloud.corp/CosteGastoIncen/mifidII-ofi-front.git

```

Install NPM y Bower:

```

npm install & bower install

```

## Running

Deply app in local environment:

`gulp serve:dev`

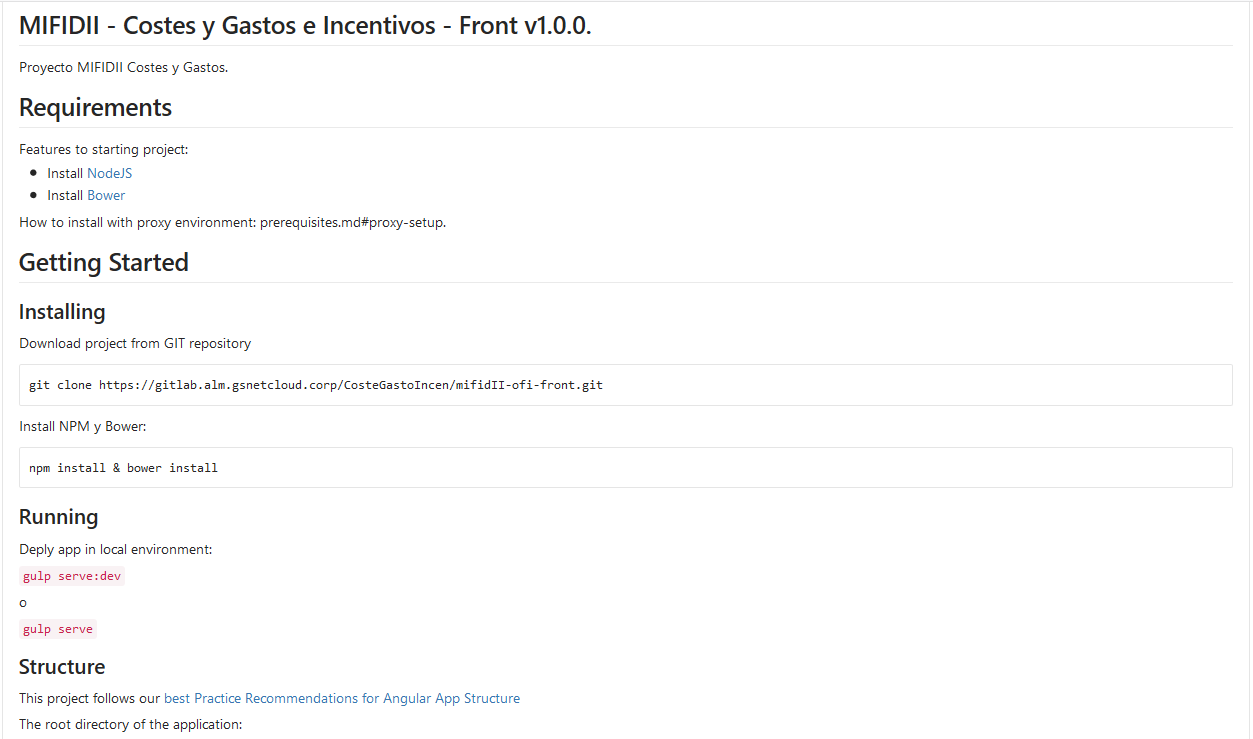
o

`gulp serve`

## Structure

This project follows our [best Practice Recommendations for Angular App Structure](https://github.com/Serenity-fronstack/angular-styleguide/blob/master/README.md)

The root directory of the application:



```

├── bower\_components/

├── conf/

├── conf.d/

├── docs/

├── e2e/

├── gulp/

├── nodes\_modules/

│

├── src/

│ ├── app/

│ │ ├── components/

│ │ │

│ │ ├── features/

│ │ │

│ │ ├── filters/

│ │ │

│ │ ├── services/

│ │ │

│ │ └── app.config.js

│ │ └── app.constants.js

│ │ └── app.environment.js

│ │ └── app.module.js

│ │ └── app.route.js

│ │ └── app.run.js

| |

│ ├── assets/

│ │ ├── css/

│ │ └── fonts/

│ │ └── images/

│ ├── json/

│ ├── favico.ico

│ └── index.html

│

├── .bowerrc

├── .csslintrc

├── .eslintignore

├── .eslintrc

├── .gitignore

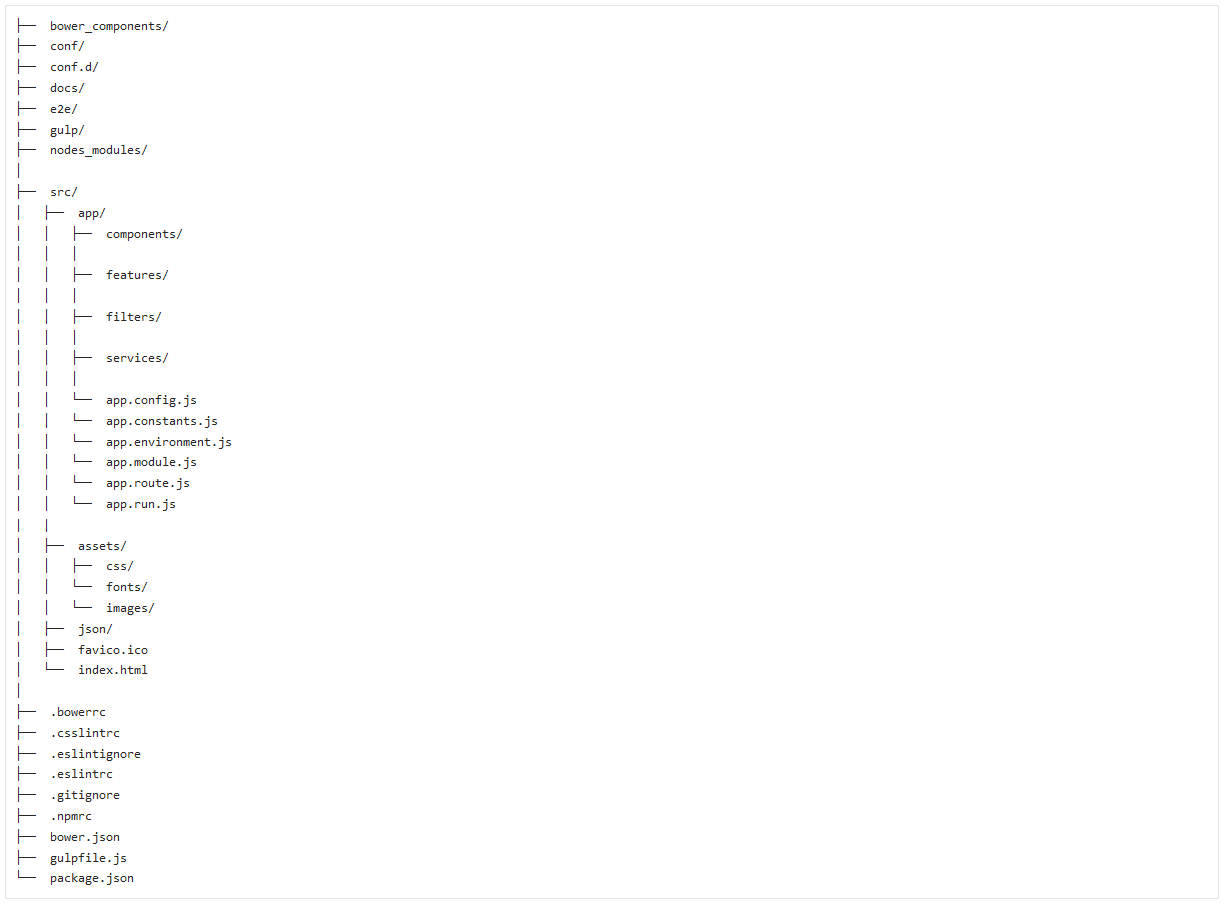
├── .npmrc

├── bower.json

├── gulpfile.js

└── package.json

```



## Configuration

The different gulp modules could be configured through a series of config files available in the `/config` folder. Settings such as Karma, Protractor or Coverage definition could be setup with these files.

### SOURCES

\* [John Papa Angular Styleguide](https://github.com/johnpapa/angular-styleguide/blob/master/a1/README.md)

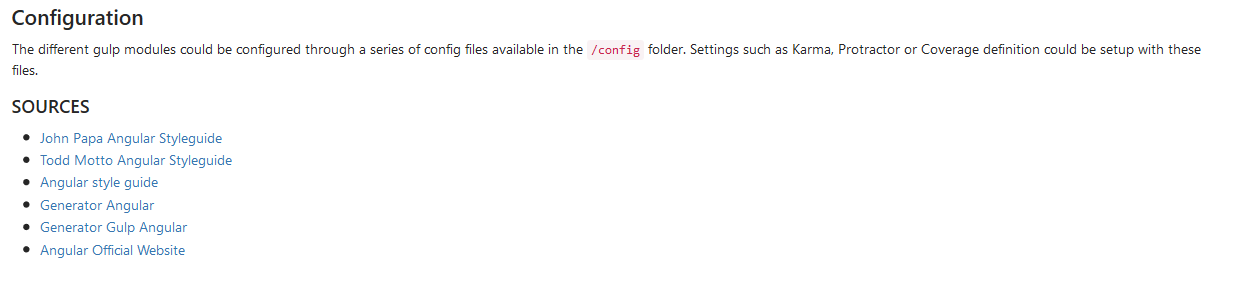
\* [Todd Motto Angular Styleguide](https://github.com/toddmotto/angular-styleguide#comment-standards)

\* [Angular style guide](https://github.com/mgechev/angularjs-style-guide)

\* [Generator Angular](https://github.com/yeoman/generator-angular)

\* [Generator Gulp Angular](https://github.com/FountainJS/generator-fountain-webapp)

\* [Angular Official Website](https://angularjs.org/)



## .gitignore

Cuando trabajamos con Git para gestionar el código fuente de las aplicaciones, es esencial configurar bien el archivo **.gitignore**. Este fichero define los archivos y directorios que no se suben al repositorio antes de hacer ***commit***. Este fichero ha de ser subido al repositorio de Git para ser compartido con el resto de usuarios que vayan a clonar nuestro repositorio. Se ha de ubicar en el raíz de nuestro proyecto.

Una particularidad a tener en cuenta es que si en el fichero usamos dos patrones que son contradictorios, se aplicaría siempre el último patrón.

**Ejemplo de contenido:**

#-- Carpetas autogeneradas por npm y bower con las librerías del proyecto

node\_modules/

bower\_components/

#-- Fichero de debug de npm

npm-debug.log

#-- Todos los ficheros con extensión .log

\*.log

### Otros recursos

El proyecto **gitignore** (<https://github.com/github/gitignore>) de GitHub contiene muchos ejemplos de archivos **.gitignore** para diferentes tecnologías y lenguajes de programación. En este repositorio se encuentran templates que nos pueden facilitar la configuración inicial de este tipo de ficheros.

Otra opción es la de usar la herramienta online en http://[gitignore.io](https://www.gitignore.io/), que básicamente permite escribir en un campo de búsqueda los nombres de todas las herramientas, sistemas, frameworks, lenguajes, etc. que puedas estar usando y generar el archivo de manera automática.

# Servicios

## Creando un servicio de utilidades (utils.service.js)

(function () {

'use strict';

angular

.module('app')

.factory('utils', ['$rootScope', '$q', '$state', '$window', '$uibModal', 'configuration', '$location', utils]);

/\*\* @ngInject \*/

function utils($rootScope, $q, $state, $window, $uibModal, configuration) {

var exported = {

openModalError: openModalError,

checkToken: checkToken,

checkSession: checkSession,

redirectToError: redirectToError,

getOptionsSpinner: getOptionsSpinner

};

return exported;

function openModalError(params) {

$uibModal.open({

component: 'abModal',

backdrop: true,

size: 'sm',

animation: true,

bindToController: true,

scope: $rootScope,

templateUrl: 'src/app/components/ab-modal/ab-modal.html', controllerAs: 'abm', resolve: {

datos: {

message: params.message,

accion: params.datos,

showBtnAcept: false,

showBtnCancel: false,

showBtnClose: true

}

}

});

}

function checkToken() {

var isValid = false;

setInterval(function () {

var theMoment = new Date().getTime();

//var fecha = new Date(); var secToken = sessionStorage.getItem("dsToken");

var objMoment = JSON.parse(secToken);

var tokenMoment = objMoment ? objMoment.localExpireTime : '';

if(tokenMoment=='') { return isValid;

}

if (!secToken && !isValid || secToken === null && !isValid) {

sessionStorage.clear();

isValid = true;

return isValid;

} else if (theMoment > tokenMoment && !isValid) {

sessionStorage.clear();

isValid = true;

return isValid;

}

}, 10000);

}

function checkSession(){

if(!$window.sessionStorage.getItem("dsToken")){

configuration.all().then(function(env){

$window.location.replace(env.security.auth.loginUrl);

});

}

}

function redirectToError(params) {

var urlError = params.urlError;

var errorTxt = 'Su sesión ha caducado o se ha producido un error en el acceso. Vuelva a iniciar sesión';

window.open(urlError + '?errorTxt=' + errorTxt, '\_self');

}

function getOptionsSpinner() {

var opts = {

lines: 15, // The number of lines to draw

length: 15, // The length of each line

width: 10, // The line thickness

radius: 30, // The radius of the inner circle

corners: 1, // Corner roundness (0..1)

rotate: 0, // The rotation offset

direction: 1, // 1: clockwise, -1: counterclockwise

color: '#B40404', // #rgb or #rrggbb

speed: 0.6, // Rounds per second

trail: 60, // Afterglow percentage

shadow: true, // Whether to render a shadow

hwaccel: false, // Whether to use hardware acceleration

className: 'spinner', // The CSS class to assign to the spinner

zIndex: 2e9, // The z-index (defaults to 2000000000)

top: 'auto', // Top position relative to parent in px

left: 'auto' // Left position relative to parent in px

};

return opts;

}

}

})();

## Invocando a una función del servicio de Utilidades

function showSpinner() {

…

// Set options for the spinner

var opts = utils.getOptionsSpinner();

// Create the spinner

new Spinner(opts).spin(document.getElementById('kintone-spin'));

}

# API Rest

Test con JSON Server (<https://my-json-server.typicode.com/>)



Crea un archivo JSON de ejemplo con algo de datos como tu requerimiento. Por ejemplo, necesito algunos datos JSON con información de usuario como id, nombre, ubicación, etc. Así que crearé un archivo llamado info.json con la siguiente información JSON:

{

    "users": [{

        "id": 1,

        "name": "roy",

        "location": "india"

    }, {

        "id": 2,

        "name": "sam",

        "location": "wales"

    }]

}

Desde la terminal, ejecuta el servidor json con info.json como fuente de información y deberías tener la API REST corriendo en [http://localhost:3000](http://localhost:3000/).



*json-server --watch info.json --port 4040*

Ya que nuestro servidor API REST falso está listo, veamos cómo acceder al API REST usando un cliente. Estoy usando el cliente Postman REST para hacer las llamadas API.

## Petición GET

Comencemos haciendo una petición  GET a la URL REST. Dentro del archivo JSON, hemos definido un punto final users que contiene información relacionada a los usuarios. Al hacer una petición GET a la URL <http://localhost:3000/users>, esta debería mostrar la información existente.

## Petición POST

Para poder agregar nueva información a la información existente, haremos una petición POST a la URL <http://localhost:3000/users>. Aquí está cómo se vería la petición POST.

Grid

Paginación

CallBack

# Login

# Spinner

Vamos a configurar un *spinner* en nuestra aplicación, para ello instalamos la librería con npm con la siguiente instrucción:

*npm install –g --save spin.js*

<http://spin.js.org/>

## Propiedades del Spinner

//Obtener las propiedades del spinner

function getOptionsSpinner() {

var opts = {

lines: 15, // The number of lines to draw

length: 15, // The length of each line

width: 10, // The line thickness

radius: 30, // The radius of the inner circle

corners: 1, // Corner roundness (0..1)

rotate: 0, // The rotation offset

direction: 1, // 1: clockwise, -1: counterclockwise

color: '#B40404', // #rgb or #rrggbb

speed: 0.6, // Rounds per second

trail: 60, // Afterglow percentage

shadow: true, // Whether to render a shadow

hwaccel: false, // Whether to use hardware acceleration

className: 'spinner', // The CSS class to assign to the spinner

zIndex: 2e9, // The z-index (defaults to 2000000000)

top: 'auto', // Top position relative to parent in px

left: 'auto' // Left position relative to parent in px

};

return opts;

}

## Mostrar el spinner

//Show the spinner

showSpinner();

function showSpinner() {

// Initialize

if ($('.kintone-spinner').length == 0) {

// Create elements for the spinner and the background of the spinner

var spin\_div = $('<div id ="kintone-spin" class="kintone-spinner"></div>');

var spin\_bg\_div = $('<div id ="kintone-spin-bg" class="kintone-spinner"></div>');

// Append spinner to the body

$(document.body).append(spin\_div, spin\_bg\_div);

// Set a style for the spinner

$(spin\_div).css({

'position': 'fixed',

'top': '60%',

'left': '50%'

});

$(spin\_bg\_div).css({

'position': 'absolute',

'top': '0px',

'left': '0px',

'z-index': '500',

'width': '100%',

'height': '100%',

'background-color': '#000',

'opacity': '0.5',

'filter': 'alpha(opacity=50)',

'-ms-filter': "alpha(opacity=50)"

});

// Set options for the spinner

var opts = utils.getOptionsSpinner();

// Create the spinner

new Spinner(opts).spin(document.getElementById('kintone-spin'));

}

// Display the spinner

$('.kintone-spinner').show();

}

## Ocultar el spinner

// Hide the spinner

$rootScope.$on('hideSpinner', function(event, obj){

$('.kintone-spinner').hide();

});



# Ventanas modales

Para hacer uso de ventanas modales, haremos uso de la directiva de angular **$uibModal**, vamos a ver el siguiente ejemplo:

## Llamada

var params = {

component: 'abAdviseModal',

backdrop: true,

size: 'sm',

animation: true,

bindToController : true,

controller : 'abAdviseModalCtrl',

scope: $scope,

templateUrl: 'app/components/ab-advise-modal/ab-advise-modal.html',

resolve: {

datos: {

isJSON: false,

objData: vm.formDataOper,

texto: texto,

accion: vm.situacionCONT,

showBtnAcept: true,

showBtnCancel: true,

showBtnClose: false

}

}

};

$uibModal.open(params);

## Controlador

(function () {

"use strict";

angular

.module('app')

.component('abAdviseModal', {

templateUrl: 'app/components/ab-advise-modal/ab-advise-modal.html',

controller: 'abAdviseModalCtrl',

controllerAs: 'vm',

bindings: {

resolve: '<',

close: '&',

dismiss: '&'

}

})

.controller('abAdviseModalCtrl', ['$scope', '$rootScope', 'utils', '$uibModal', 'rest', 'api', 'environment', '$window', '$stateParams', '$filter', abAdviseModalController])

;

function abAdviseModalController ($scope, $rootScope, utils, $uibModal, rest, api, environment, $stateParams) {

var vm = this;

var res = vm.resolve;

vm.isJSON = res.datos.isJSON,

vm.objData = res.datos.objData;

vm.texto = res.datos.texto;

vm.accion = res.datos.accion;

vm.showBtnAcept = res.datos.showBtnAcept;

vm.showBtnCancel = res.datos.showBtnCancel;

vm.showBtnClose = res.datos.showBtnClose;

vm.aceptarInforme = aceptarInforme;

vm.cancelar = cancelar;

**…**

## Fichero html que nos sirve de Template

<div>

<div class="modal-header">

<div class="navbar navbar-modal">

<div class="container">

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand">

&nbsp;&nbsp; {{ vm.accion }}

</a>

</div>

</div>

</div>

</div>

<div class="modal-body">

<div class="container" ng-show="vm.isJSON">

<p>Información {{vm.accion}}:</p>

<textarea style="overflow:auto;resize:none" rows="8" cols="30" name="calling" disabled="disabled">{{ vm.objData }}</textarea>

</div>

<div class="container" ng-hide="vm.isJSON">

<p>{{ vm.texto || vm.accion}}</p>

</div>

</div>

<div class="modal-footer">

<button class="btn btn-success" ng-click="vm.aceptarInforme(vm.accion)" ng-show="vm.showBtnAcept" style="width: 100px;">

<i class="glyphicon glyphicon-ok"></i>&nbsp;Aceptar

</button>

<button class="btn btn-danger" ng-click="vm.cancelar()" ng-show="vm.showBtnCancel" style="width: 100px;">

<i class="glyphicon glyphicon-remove"></i>&nbsp;Cancelar

</button>

<button class="btn btn-danger" ng-click="vm.cancelar()" ng-show="vm.showBtnClose" style="width: 100px;">

<i class="glyphicon glyphicon-remove"></i>&nbsp;Cerrar

</button>

</div>

</div>