Explicación Angular

Link de la entrega desplegada:

http://asw.delnido.es/

General

Lo más importante para empezar es saber que tenemos que definir un módulo, en nuestro caso lo hemos llamado *"routing"* , dentro del cual funcionan los controladores que usemos. Este módulo/app funcionará dentro de donde hayamos declarado el *ng-app="routing"* en el template. En nuestro caso la etiqueta *<body ng-app="routing"></body>* de nuestro index.html

En el controlador podemos definir diferentes elementos en la variable *$scope*. Que luego podemos obtener en el template escribiendo el nombre del elemento entre *{{""}}*.
Ejemplo:
Controlador.js: $scope.foo = "FooContent"
Template.html: {{foo}}

Lo mismo pasa con las funciones. Si definimos una función el controlador a través del $scope, luego podemos usarla en el template.
Ejemplo:
Controlador: $scope.myFunction = function(){}
Template: ng-click="myFunction"

Para la realización de bucles podemos usar *ng-repeat="item in list".*

Signin y SignOut

La comunicación que se hace de la vista html al controlador es con el input text, donde se escribe el id, y con el botón signIn, utilizando el parámetro $scope.
Cuando se va accede a la página se inicia todo otra vez, y se pone el input text a vacío, utilizando la variable ng-model="userID" en el html signIn.html y el código $scope.userID="" en el controlador signInCtrl.js.
Cuando se pulsa el botón signIn se llama la función signIn en el controlador signInCtrl.js utilizando el ng-click="signIn() que tenemos al signIn.html , dentro de esta se lee el contenido que tenemos en el input utilizando $scope.userID. Una vez tenemos el id, llamamos a la API para que nos devuelva información del usuario y podamos generar un token que guardamos en el localStorage.

Para el signOut, simplemente hay un botón dentro del signIn.html, dónde si lo clicas se comunica con la función ng-click="signOut() que tenemos en el html se comunica con la función $scope.signOut que está en el signInCtrl.js y elimina el token en el localStorage.

Editar perfil

El editar perfil tiene 3 campos de input, donde se inicializan con la información que la API nos retorna. Para ello hemos comunicado el controlador profileCtrl.js con profile.html mediante 3 ng-models llamados user.email, user.name y user.about y $scope.user.email, $scope.user.name y $scope.user.about, respectivamente. Al acceder a la pagina, se llama a la API y se rellena esta información con la actual del usuario, luego el usuario puede editar estos tres campos y al finalizar pulsa el botón modify perfil, que mediante la función ng-click="modifyProfile()" llama a la función del controlador $scope.modifyProfile y este se encarga de enviar a la API la nueva información del perfil.

Obtener contribuciones

En la pantalla principal se obtienen todas las contribuciones mediante la llamada correspondiente de la API, además para cada contribución se hacen las llamadas pertinentes para saber sus datos, por ejemplo se realiza una llamada para obtener el nombre de los autores de cada contribución, otra para obtener el número de comentarios de la contribución y otra para obtener los replies de cada comentario de la contribución. Todas estas llamadas se hacen para mostrar correctamente los datos de una contribución. Además se ha utilizado Angular Moment que nos permite poner el tiempo relativo a un contribución a partir del string que hay guardado en la base de datos.

Show de una contribution

Pantalla a la que le corresponde el controlador contributionCtrl.js de nuestro proyecto.
En esta pantalla se tienen que tener en cuenta varios casos.
Primero mostrar la contribución que corresponde según la url. Cuyo parámetro se define en la configuración de rutas (app.js) así → */contributions/:contributionId* , y el que podemos obtener desde nuestro controlador a través de la variable *$stateParams.contributionId.*
Tenemos que hacer la correspondiente petición GET para obtener los datos de esta contribución.
Por otro lado también tenemos que hacer que se guarden los nuevos comentarios. Para eso usamos un *form* con un input (que contendrá el comentario). Para obtener el texto de este comentario al input le asignamos un *ng-model="something"* y este "*something*" nos servirá para poder obtener el contenido desde el controlador con la variable *$scope.something.*
También necesitaremos otras 2 llamadas GET a la API, para obtener los comentarios y las replies. Y otra llamada POST para poder realizar una reply en alguno de los comentarios.
[En este caso, para no complicarnos, hemos puesto solo un formulario, con dos inputs, uno que contiene el texto de la reply, y otro, oculto, para asignarle el id del comentario sobre el que se quiere hacer la reply.]
Para la realización de las votaciones, se realiza una simple llamada PUT a la API cuando haces click en el flechita. Aprovechandonos de un *ng-click="functionA"* desde el template html y definiendo la función *"functionA"* en el controlador.

Lista de Asks

Al igual que al obtener las contribuciones, en el controlador de esta pantalla se obtienen todas las contribuciones mediante la llamada correspondiente de la API, además para cada contribución se hacen las llamadas pertinentes para saber sus datos, por ejemplo se realiza una llamada para obtener el nombre de los autores de cada contribución, otra para obtener el número de comentarios de la contribución y otra para obtener los replies de cada comentario de la contribución.
También se utiliza Angular Moment para saber el tiempo que lleva una contribución guardada en la base de datos.
En la vista únicamente se muestran los resultados cuya url sea vacía.

Lista de Threads

Esta pantalla solo se muestra si hay un usuario logueado, se muestran los comentarios y las replies hechas por el usuario actual.
Para hacer esto, en el controlador, hemos hecho las llamadas correspondientes a la API pasándole el id del usuario almacenado en el *localStorage*.
También en el controlador hemos creado una función, con su correspondiente llamada a la API, dónde dada una id de una contribución obtenemos otros detalles como por ejemplo el título para saber así el comentario a qué contribución hace referencia.

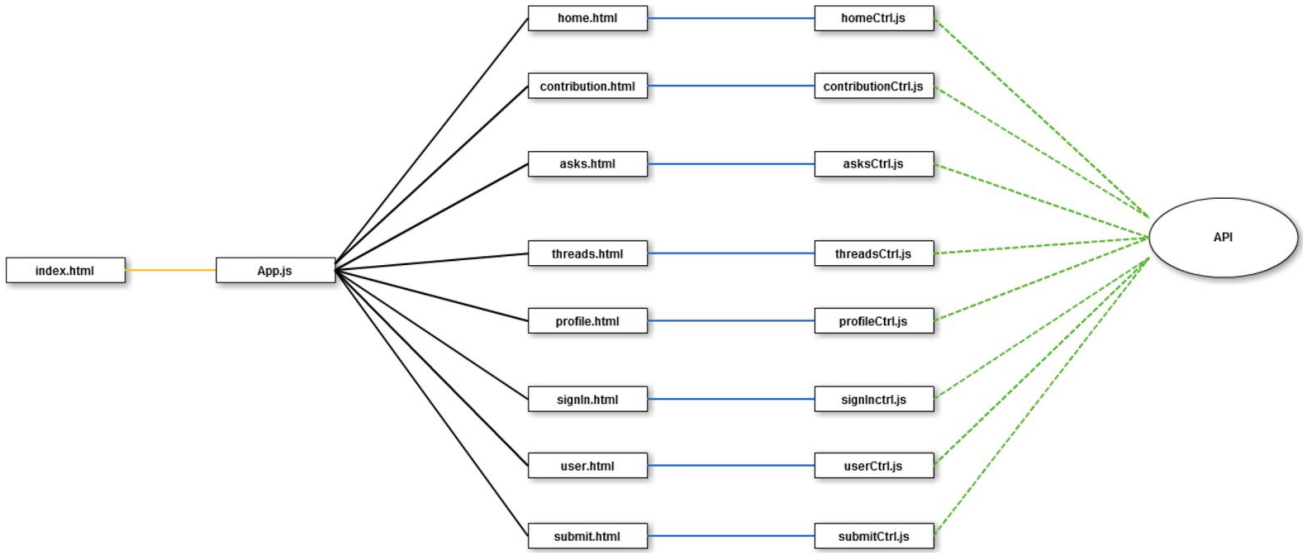
Enviar una contribución

En esta pantalla hemos creado un formulario donde cada campo tiene un atributo *ng-model*. En el controlador recogemos toda la información de estos campos, también obtenemos el id del usuario actual del *localStorage* y ponemos esta información como parámetros a la llamada a la API.
Una vez hecha la llamada, ésta nos puede devolver *success* o *error*. En el primer caso se redireccionará a la home donde podremos ver la contribución creada en la parte superior de la lista. En caso de *error*, se hace un reload de la página para que se vuelva a intentar introduciendo correctamente el formulario.

Funcionamento Login

El login funciona con la API que hicimos con el Swagger. Para loguearte tienes que ir a la página de SIgn In. Una vez ahí tienes que poner el id de tu usuario y pulsamos el botón Sign In. Esto llama a la función de signIn que se encuentra en el SignInCtrl.js. Esta se encarga de hacer una petición a la API que te devuelve el usuario en el caso que exista, en caso contrario devuelve un caso de error que sale como popup en la página web. Finalmente, se guarda el id del usuario en el localStorage en base 64, para así restringir su y preservar su significado original. Luego, cuando se necesita consultar cosas que se necesite el token del usuario simplemente se accede al token guardado en el localStorage. En el caso de Sign out, simplemente elimina el token del localStorage.

Diagrama


index.html
App.js
home.html
contribution.html
asks.html
threads.html
profile.html
signIn.html
user.html
submit.html
homeCtrl.js
contributionCtrl.js
asksCtrl.js
threadsCtrl.js
profileCtrl.js
signInctrl.js
userCtrl.js
submitCtrl.js
API