UNIVERSITAT DE BARCELONA

Trabajo final de Grado

GRADO DE INGENIERÍA INFORMÁTICA

Facultad de Matemáticas e Informática
Universidad de Barcelona

Co-gobierna
Aplicación de participación ciudadana desarrollada en AngularJS.

Autor: Edward Osorio Crispin
Directora: Dra. Maite López Sánchez

Realizado en: Departamento de Matemáticas e Informática

Agradecimientos

El presente proyecto es la culminación de una etapa muy esencial en mi vida: la etapa universitaria. Por esta razón, quiero dar las gracias a los que me acompañaron a lo largo de estos años y forman parte de esta historia. A cada profesor que tuve en la carrera, de quienes aprendí todo aquello que ignoraba y asentaron las bases de la carrera en mí. Lo bueno es poder haber trabajado codo con codo al lado de mis compañeros, formando equipos de trabajo y entendiéndonos para un mejor avance en los temas que teníamos que presentar.

A los míos, mi familia, quienes, desde su condición y rol, me han sabido apoyar tanto moralmente como sentimentalmente. Valoro cada detalle en animarme cuando las cosas iban difíciles en la carrera.

A mi tutora de este proyecto final, quien confió en mis posibilidades para poder llevar a cabo este tema, sin olvidar la motivación que existe en mí para poder desarrollarlo, su apoyo y paciencia es bueno recordar.

Finalmente, mis agradecimientos sinceros a cada uno de ustedes quienes decidieron ser partícipe de mi presentación en este proyecto final. Gracias por su interés y ganas de aprender. Hasta aquí se cierra un ciclo, esperando con optimismo nuevas etapas por superar. Muchas gracias.

Sumario

1. Introducción

El presente proyecto se basa en la idea original de los consensos de normas en comunidades virtuales que conllevan la participación ciudadana, en forma democrática, usando internet como herramienta para entender las formas de decisión popular (las populares elecciones sociales).

Como toda idea, puede evolucionar, y a partir de aquí nace la idea actual de la cual se plasma en este proyecto: Procesos participativos, vinculantes y transparentes. Siguiendo con esta idea, entonces podemos diferenciar estas tres grandes características.

La primera característica se centra en los procesos participativos orientados a desarrollar unos planes municipales teniendo como base unas políticas públicas. La segunda característica, los procesos vinculantes que escapan del patrón: informar-consultar-decidir. Y como última característica, los procesos transparentes que explican justificadamente las acciones de una cierta propuesta después de haberse dado a conocer.

Buscando una forma de poder presentar este portal, con una funcionalidad y diseño de cara al usuario (frontend), se ha decidido usar la herramienta AngularJS para conseguir los objetivos marcados en el presente proyecto. Este framework es de código abierto y se usa para crear y mantener aplicaciones web de una sola página. Mantenido por Google, está programado en JavaScript.

Asimismo, y tratando de conseguir más estabilidad y compatibilidad en el desarrollo de la web, se determinó usar la tecnología Stack Mean. En esta tecnología, cuya información más detallada se explicará más adelante, participa activamente AngularJS, por esta razón van de la mano y es más cómodo poder tener el *front-end* y el *back-end* compactado en las funcionalidades.

Ante este panorama, la misión es clara: poder hacer crecer el portal tanto en calidad de software, como de usabilidad y de funcionalidad. De esta forma, los usuarios (los grandes protagonistas del portal), pueden participar en las discusiones del portal de una forma correcta unido a las mejores características que pueden venir de cara al futuro. Estas características pueden ser la interacción a tiempo real entre los usuarios, como por ejemplo un chat de discusión entre ellos o la capacidad de proponer más proyectos que vean necesarios.

1.1 El proyecto web

El presente proyecto, como hemos dicho, sienta sus bases en poder mostrar al usuario una página web que tenga la función de participar, vincularse y mostrar transparencia en ello, haciendo que la ciudadanía participe activamente de los planes estratégicos de un ayuntamiento, en nuestro caso de Madrid.

Tratándose de un proyecto final, se desarrollan temas y funcionalidades vistos en algunas asignaturas de la carrera. Cabe mencionar:

- Coweb: esta materia optativa se explicó temas sobre el desarrollo de páginas web's, usando herramientas para su construcción. El lenguaje HTML, las hojas de estilo CSS para el diseño, JQuery y base de datos, funciones escritas en el lenguaje JavaScript en donde particularmente sentamos las bases para nuestro interés personal en la programación web. Principalmente lo que se hizo fue montar una página web con todas estas herramientas. Con nuestro proyecto todo esto va relacionado, ya que tratamos de reunir todos los conocimientos necesarios que se explicaron en esta materia para poder agregar unos nuevos (AngularJS, MongoDB, etc) y así ampliar mis conocimientos sobre herramientas que son usadas en la actualidad.

- Diseño del Software: materia en donde se dictan temas sobre como diseñar un proyecto antes de su programación. Fue bien tener conocimientos sobre como visionar el proyecto y tener claro los escenarios y usuarios que participan en él. A través de casos de uso y diagramas de secuencia, podemos entender más el proyecto y su funcionamiento. En nuestro caso explicamos y detallamos las funciones más importantes de nuestra web, como por ejemplo sería el registro e inicio de sesión de usuarios, o el mecanismo de participación.

- Ingeniera del Software: otra materia protagonista y que juega un papel muy importante en cualquier proyecto. En su temario, se rescata la explicación de cómo planificar en un periodo de tiempo cualquier desarrollo de trabajo. He aquí que se comprueba cómo trabajar en equipo desarrollando un proyecto para la vida real y de qué forma se organizan las diferentes responsabilidades de los desarrolladores. Al tratarse un proyecto que desarrollamos desde cero, utilizamos la técnica de Scrum, que consiste en una reunión con el tutor y el equipo de diseño después de márcanos un hito de tiempo para el desarrollo evolutivo de nuestro proyecto web y planificar el siguiente hito.

- Factores Humanos: materia en donde se imparte como saber hacer un buen test de usabilidad.

1.2 Motivación del proyecto

La realización de este proyecto tiene una serie de motivaciones que fueron el impulso para que se lleve a término. Principalmente el interés y ganas de poder desarrollar desde cero una página web con la tecnología elegida por mí, me llevo a pensar que tan importante es saber y tener la experiencia necesaria en este tema. Hablamos de AngularJS, un framework que en la actualidad está siendo muy usado por grandes empresas para el desarrollo web en la parte frontend.

Meses atrás de empezar con el proyecto, había tenido un contacto inicial con el framework, el cual me pareció sumamente interesante. El hecho de que haya sido desarrollado por Google, el gigante informático, y que sea programado en JavaScript, sumó puntos a favor. Además, que AngularJS lleva como filosofía el patrón MVC (modelo, vista, controlador), algo muy insistido en la carrera en las diversas materias de software, hizo que me declinara a escoger este framework. Además de AngularJS, y por intereses y gustos dados al diseño web, quise poner en práctica todo lo relacionado con Bootstrap. Bootstrap es un framework de código abierto para diseñar aplicaciones web. Apoyándome en esta herramienta y AngularJS fue mi idea inicial de cómo poder construir un sitio web sobre participación ciudadana y así ayudar a la población de tener un espacio para sus decisiones con respectos a proyectos que conseguirán beneficiarlos.

Refiriéndose a la temática de la web, colaborar y construir el sitio web de una herramienta que centra el tema en procesos participativos, vinculantes y transparentes llamada Co-gobierna, me sirve de motivación extra ya que desarrollo una web para un medio real, la ciudadanía. Al tratarse de un proyecto presente y de cara al futuro, me siento parte de este plan y el conocerlo me da la satisfacción para seguir aprendiendo sobre la participación ciudadana, como por ejemplo los portales como Decide Madrid, Considerit, etc., que tienen esta temática.

Desde mi terreno, como desarrollador web, tengo comunicación directa con el equipo de diseño y el haberme reunido con este equipo de la herramienta, me ha servido como punto de inicio para poder entender que es lo que ellos querían realmente plasmar en la web. Este equipo fue quien inició el proyecto desde el Laboratorio de Inteligencia Colectiva para la Participación Democrática de Medialab-Prado en Madrid. Allí se pensó en cómo desarrollar nuevos mecanismos digitales que aborden retos relacionados con la participación democrática en internet. Finalmente, tanto sea para el usuario como para el técnico, la persona se ha de sentir familiarizado con el portal y debe de tener una experiencia de usuario agradable. Es allí donde mi motivación radica más, en el desarrollo *frontend* de la página web.

1.3 Objetivos

Listando los objetivos de nuestro proyecto, principalmente y el más básico es ofrecer una herramienta de labor que sea funcional y útil. Esto lo conseguimos elaborando un sitio web para la herramienta Cogobierna, que muestra a la ciudadanía propuestas para el beneficio de una comunidad. Las características y funcionalidades de Cogobierna nos lo indicó el equipo de proyecto de esta herramienta. De esta forma, tenemos como misión:

- Integrar todo el diseño hecho por el equipo, desarrollando una web que sea funcional y útil para el usuario, sea el dispositivo que use. He aquí el tema central de mi proyecto: trabajar principalmente con el *frontend*, basándome en la herramienta AngularJS.

- Poder montar una backend sencilla y compatible con nuestra frontend, por esto decidimos usar la tecnología Stack Mean.

- Separar funcionalidades de un usuario ciudadano de las de un usuario técnico. Esto quiere decir que el usuario técnico tendrá más acceso a ciertos apartados que un ciudadano no podrá visualizar.

- Tratar en detalle las 3 características de los procesos de nuestra web: participativos, vinculantes y transparentes. Para esto el objetivo fijado fue trabajar desde cuando se participa en un proyecto hasta cuando se ve los resultados de dicha participación, dicho en otras palabras, empezar con los procesos participativos y acabar los procesos transparentes, así tenemos una fácil organización y sabemos en qué estamos trabajando realmente.

1.4 Planificación de trabajo

Hemos diseñado dos diagramas de Grantt para visualizar el tiempo dedicado a este proyecto *(Ver figura 1 y 2).*

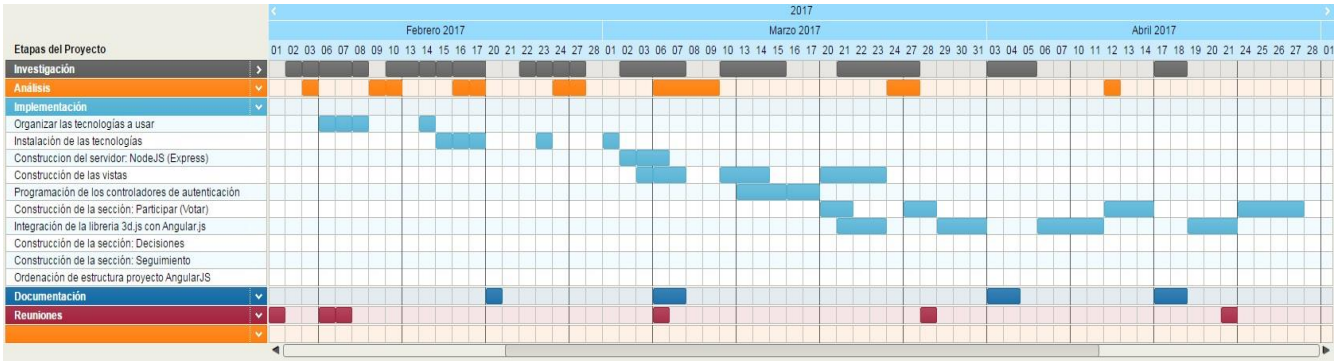

Figura 1 - Diagrama de Gantt de Febrero hasta abril

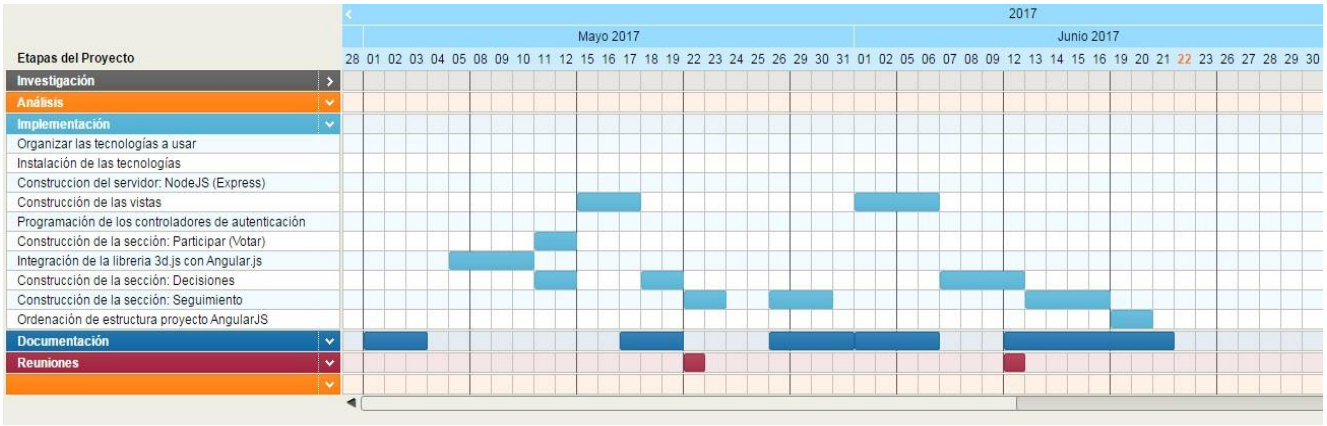

Figura 2 - Diagrama de Gantt de Mayo hasta junio

2. Análisis

Nuestro proyecto tal y como se ha mencionado anteriormente, se basa en lo que son denominados: portales web de participación ciudadana.

2.1 Participación ciudadana

El término de participación ciudadana nace bajo la necesidad de poder hacer partícipe a los principales sectores sociales en los varios y múltiples proyectos y planes que están ligados con los temas de una convivencia social: ecología, movilidad, sanidad, urbanismo, etc[1]. Teniendo un esquema de ejemplo, las áreas de Ecología, Movilidad y Urbanismo tienen el común denominador en donde se da consejos sectoriales, procesos de seguimiento, talleres y jornadas de trabajo. Entendemos entonces que esta participación es una parte esencial del sistema democrático que impulsa la arquitectura de una sociedad que debe estar activa. Esta sociedad al estar relacionada con los temas públicos, da una dosis de crecimiento al Gobierno y promueve su eficacia. Teniendo en consideración esto, el ejercer su derecho el ciudadano, resultará un equipo de gobierno exigente porque los ciudadanos esperaran mucho de este gobierno.

La participación ciudadana también conlleva un proceso participativo en donde se genera un diálogo constructivo y con una buena argumentación entre las instituciones y la sociedad *(Ver figura 3).*

Hemos de aclarar también que existe el término de participación social y que comúnmente es confundido con la participación ciudadana. Definimos la participación social como la participación que es fomentada por las instituciones autonómicas, locales y estatales, siendo ellos mismos los encargados de facilitar las herramientas para que la ciudadanía tenga acceso a los temas decisivos del gobierno de forma independiente sin que esté ligado a ninguna fuerza política. La diferencia entre la participación ciudadana radica en que la participación social engloba todos los tipos y niveles de participación en actividades propiamente sociales o de estilo comunitario.

Un punto importante en este apartado, es definir bien lo que sería el espacio local de una participación ciudadana. Este espacio será necesario para que el ciudadano se sienta familiarizado. Si bien hemos dicho que la participación es el total de la relación sociedad civil más estado, es entonces aquí donde el ciudadano puede expresarse, que sienta suyo aquel territorio para que pueda manifestarse y ejercer sus derechos.

La participación ciudadana es la llave para revolucionar el ambiente local por un ambiente público. Además, con esto, se contribuye a crear las condiciones para asentar una gobernabilidad democrática. Siguiendo la idea de separar definiciones, la participación ciudadana a diferencia de otras maneras de participación, hace una llamada a todos los habitantes de una ciudad para que intervengan en los planes y proyectos públicos representando intereses específicos, marcando como primera instancia en el ámbito de lo común, diario y en el espacio local (donde se produce el mayor acercamiento entre las autoridades y ciudadanos).

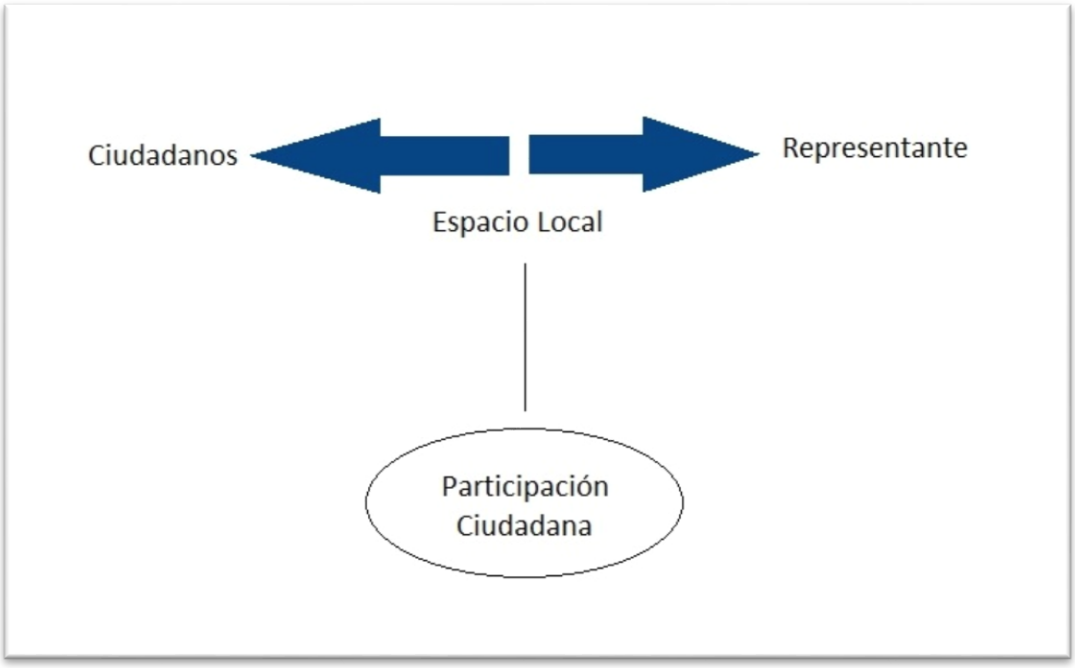

Figura 3 - Participación Ciudadana en el Espacio Local

Considerando la evolución y los objetivos de la participación ciudadana, a partir de la década de los ochenta la participación en grupo está dirigida por temas comunes de incentivar los procesos de democratización. Con esto, queremos decir que, si los ciudadanos saben que su voz ha sido escuchada, y si averiguan que su participación ha influenciado en las decisiones finales, entonces se sienten satisfechos al haber cumplido su objetivo.

La participación ciudadana ligada tan fuertemente a la democracia, explica entonces porque los ciudadanos quieren participar. El mayor motivo es porque los representantes no cumplen siempre su papel de unión entre el gobierno y los inconvenientes que suele tener una sociedad. Como bien dice un famoso texto (Mauricio Merino, La participación ciudadana en la Democracia, 1995): “Participamos, en una palabra, para corregir los defectos de la representación política que supone la democracia, pero también para influir en las decisiones de quienes nos representan y para asegurar que esas decisiones realmente obedezcan a las demandas, las carencias y las expectativas de los diferentes grupos que integran la nación.

2.2 Angular JS

En el presente apartado abarcaremos lo que realmente es el tema central del proyecto: Angular JS. Angular JS es una tecnología basada en el patrón modelo-vista-controlador (MVC) del lenguaje JavaScript para el desarrollo web Front End que da la posibilidad de confeccionar aplicaciones de una sola página, las llamadas Single-Page Applications (SPA). AngularJS se ubica de esta forma en la capa de cliente. Desde aquí como hemos indicado, al usar el patrón MVC, separa de forma eficaz la responsabilidad de cada tecnología en su área: CSS, HTML y Javascript. Teniendo esto una vez montado, el propio AngularJS las relaciona cuando considere oportuno[2].

La principal característica que tiene AngularJS es que para acceder al valor de un elemento no es necesario construir una cierta arquitectura que a la larga sería muy compleja y posiblemente obtendría resultados erróneos. Por ejemplo, aquí una muestra de la implementación *(Ver figura 4).*

```
// Muchos accesos en el árbol DOM para acceder al valor del elemento
var variable =
$("#elemento").up("li").first("p").down("span.texto").val();

// Con AngularJS todo queda más elegante y específico
var variable = $scope.variable;
```

Figura 4 - Diferencias de acceso al valor de un elemento

Todo este acceso tan eficaz es posible ya que AngularJS actualiza la vista cuando se cambia el modelo, y de igual modo en forma inversa (cuando cambia el modelo, se cambia la vista). Este proceso en donde residiría la magia de AngularJS es llamado: Two-way data binding.

El binding sería enlazar la información que se tiene en el “scope” con lo que enseñamos en el HTML. En el caso de “two-way data binding” *(ver figura 5)*, la información fluye desde el “scope” hacia la parte visual, y viceversa, desde la parte visual hasta el scope. La ventaja de usar el “binding” es ahorrar líneas de código ya que si se realiza una aplicación común con javascript sin usar librería ninguna (o puede ser con JQuery), estaríamos trabajando para cada evento, definirlos, etc. Una ventaja más de usar el “binding” sería el hecho de hacer una limpieza de código muy valorable. Esto pasa porque ya no se tiene que estar programando varios eventos ni tampoco ya se necesita enviar datos de un lugar a otro. Por ejemplo, es muy fácil y rápido si se recibe un JSON de una llamada a un servicio web, se relaciona al “scope” y automáticamente ya se tiene disponible en la vista.

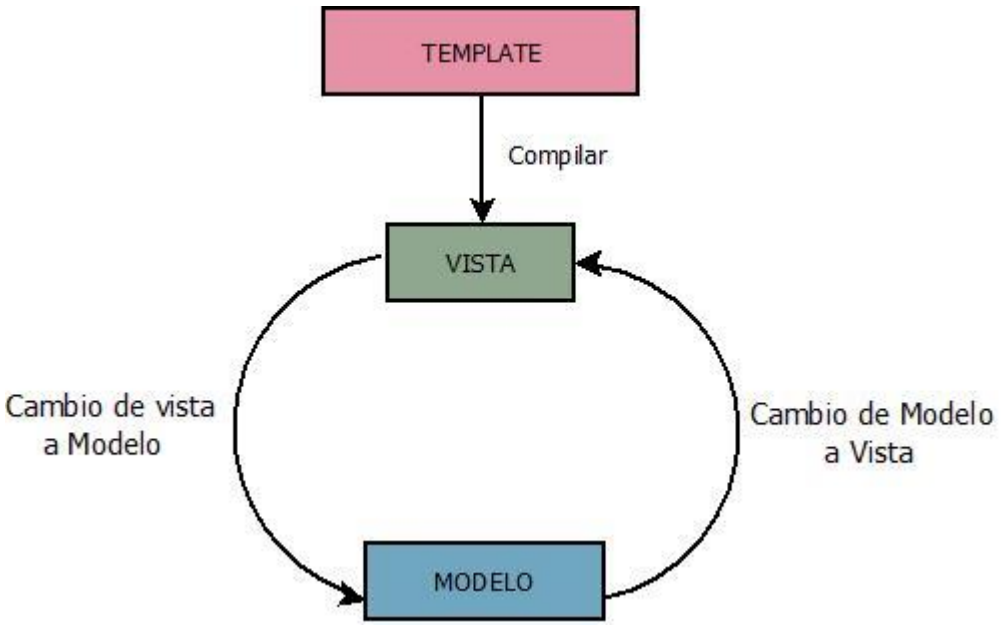

Figura 5 - Diagrama de flujo: Two-Way Data Binding

La decisión de desarrollar la aplicación web usando la tecnología AngularJS radica en los siguientes puntos principales:

- Reusabilidad: Da la opción de crear componentes (directivas) que son muy fáciles de reutilizar. Esto quiere decir, que permiten separar totalmente su función y no entran en conflicto con otros componentes.

- Testeo: En AngularJS los componentes están separados, de esta forma se puede testear su comportamiento de una forma aislada e independiente.

- Inyección de dependencias: Para el caso de tener que hacer el uso de un servicio, se puede inyectar directamente en el controlador y con ello funciona a la perfección.

Anteriormente, se ha mencionado términos propios de AngularJS (scope, controlador, modelo, vistas, etc.). Definiremos cada término brevemente para posteriormente ampliarlo en los siguientes apartados:

- Scope: El scope es el que tiene la responsabilidad de detectar los cambios en el modelo y proporciona el contexto a las plantillas. El scope está organizado en una estructura muy parecida a la estructura DOM de una aplicación. Las acciones del scope es que puede ver expresiones y propagar eventos. La traducción literal de scope sería “ámbito”. Si nos ceñimos a esta traducción, tendría sentido decir que

el scope es el “ámbito” de los datos en donde estamos confeccionando en las vistas.

- Controlador: El Controlador es toda la parte del código con la lógica que tiene la tarea de comunicar al modelo con la vista.

- Modelo: Es toda la parte de datos, que conjuntamente con la plantilla, producen todas las vistas.

- Vistas: La vista es todo lo que el usuario, el cliente final puede visualizar. Tiene como base una plantilla, se une con el modelo y se renderiza en el árbol DOM *(Ver figura 6)*.

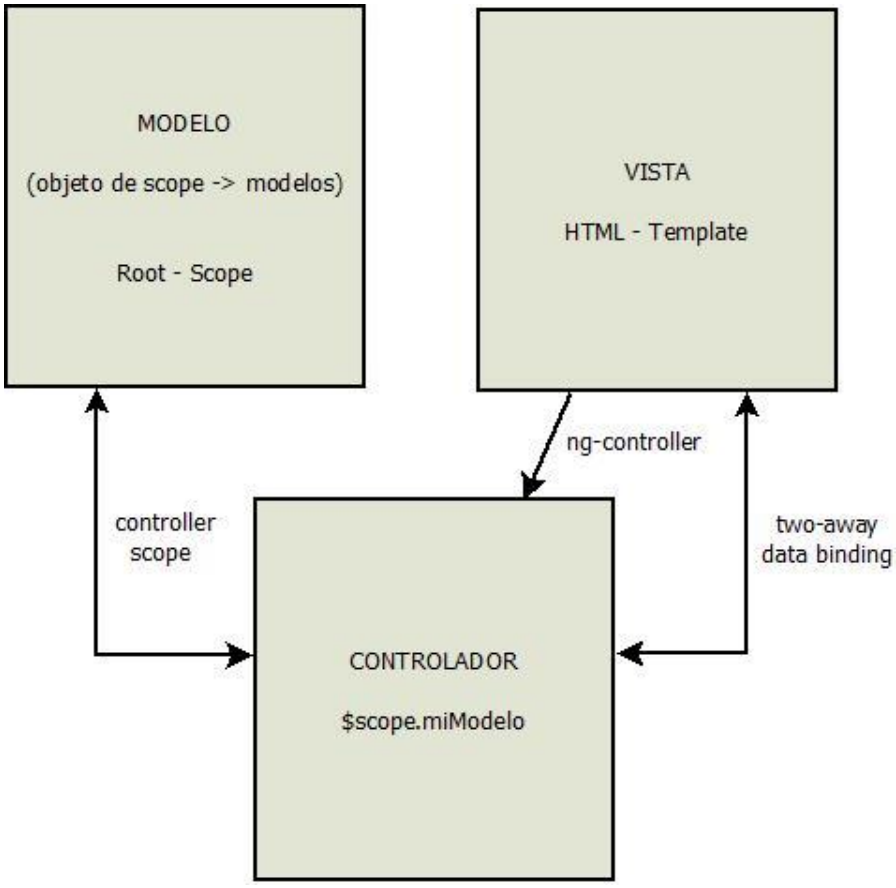

Figura 6 – Diagrama de interacción con la vista

2.2.1 Angular UI Router

Sabiendo la buena característica que tiene AngularJS de ser tan extensible que permite instalar módulos adicionales, nos centraremos en un módulo muy usado: ng-route. Este módulo se encarga de gestionar la aplicación en diferentes vistas. El único inconveniente es que no deja cargar a la vez varias vistas en una misma página. Para hallar la solución a este problema se usa: Angular ui-router *(Ver figura 7).* En este proyecto usamos este módulo ya que necesitamos tener la gestión de una vista más compleja[3].

Angular UI-Router se apoya en el patrón Composite View, ya que es uno de los patrones fijos para desarrollar la capa de presentación y establece que la vista comprendería varias subvistas que se actualizan de manera independiente.

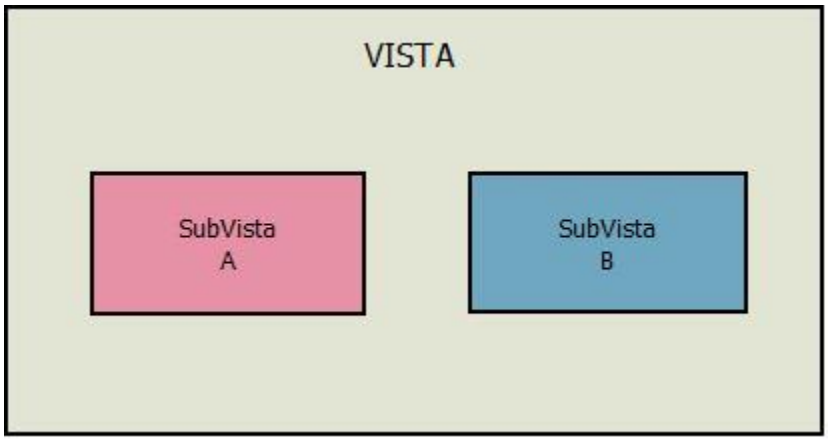

Figura 7 - Estructura de la Vista (con Angular UI-Router)

El módulo UI-Router se define en el concepto de “estado” para poder explicar la navegación entre vistas.

Las diferencias que vemos entre estado y las clásicas URL de rutas es que las vistas y rutas no están vinculadas a la propia URL del sitio. Con esto se consigue poder cambiar algunas partes del sitio usando enrutamiento, aún inclusive si la URL cambia. En cambio, si usamos rutas clásicas con ngRoute, se tendría que incluir otros métodos como ng-include y podría resultar muy difícil de entender. Con UI-Router esto se aclara ya que todos los estados, enrutamiento y vistas se controlan desde una sola configuración. De esta forma se podría tener una única vista en una sola aplicación si quisiéramos.

2.2.2 Controladores en AngularJS

El término Controlador en AngularJS define a los objetos JavaScript que participan en el desarrollo de la lógica de la aplicación. Tiene la misión de enlazar el scope (ámbito) con la vista permitiendo con esto tener el control total de los datos de las aplicaciones AngularJS[4]. Una ventaja de los controladores es poder inyectarles valores o constantes. Las constantes no cambiaran en el proceso de la aplicación a diferencia de los valores que pueden cambiar sus datos durante la ejecución de la aplicación. Además, aparte de valores y constantes, también se le puede inyectar servicios y factorías. Para la creación del controlador *(Ver figura 8)* se sigue los siguientes puntos:

- Se crea un módulo (module de Angular JS). Este objeto contendrá el controlador.
- Usar el método controller () del módulo que hemos creado para asignarle una función constructora que Angular usará cuando cree el controlador.
- Usar la directiva ng-controller, para que le asignemos el nombre propio del controlador en el HTML. Luego, esta propia directiva se colocará en el DOM donde queramos tener acceso al scope.

```
var app = angular.module("miapp", []);
app.controller("miappCtrl", function(){
    var scope = this;
    scope.datoScope = "valor_del_dato";

    scope.metodoScope = function(){
        scope.datoScope = "otro_valor";
}
});
```

Figura 8 - Código de creación de un Controlador

En aplicaciones de mayor tamaño, como es nuestro caso, es muy común tener los controladores en archivos JavaScript externos. Hay que tener en cuenta para que casos es óptimo usar el controlador y para qué casos no. En este caso, enumeramos para que no se debe de utilizar el controlador:

- No utilizar el controlador para manipular el DOM de una página. El controlador no debe tener conocimiento de cómo está constituido el HTML del documento en donde se estará desarrollando.
- No utilizar el controlador para hacer un filtro a la salida de datos.
- No utilizar el controlador para hacer un formateo de la entrada de datos.
- No utilizar el controlador para cambiar estados entre los múltiples controladores que pueden haber.
- No utilizar el controlador para supervisar el ciclo de vida de componentes. Un ejemplo claro sería crear servicios o instancias.

2.2.3 Servicios en AngularJS

Otro elemento que usamos para la elaboración de nuestro proyecto, es el artefacto llamado: Servicios. En AngularJS podemos encontrar cinco tipos distintos de servicios: Constantes, Servicios, Valores, Factorías y Proveedores[5].

Ahora definamos lo que es un servicio. En el aspecto global diríamos que un servicio son las múltiples actividades que responden a las necesidades del usuario. En este caso, el servicio provee múltiples funciones para realizar una tarea o muchas tareas específicas. Dentro de la perspectiva de AngularJS, un servicio es un objeto JavaScript que permite el acceso para obtener información. Los objetos aquí tendrán métodos que sirven para mantener los datos durante el ciclo de vida de la aplicación y que tienen comunicación entre ellos a través de varios componentes.
Una característica importante es que todos los servicios en AngularJS son en realidad Singleton. Singleton significa que AngularJS una vez haya construido un objeto de servicio, la misma instancia es reutilizada por toda la aplicación. De esta forma se asegura que no hay dos instancias de un mismo servicio. Por esta razón, la utilidad de los servicios radica en que se usan para compartir datos a través de múltiples componentes. Podemos construir nuestros propios servicios peo AngularJS ya tiene construido muchos servicios que interactúan constantemente.

Cómo mencionamos antes, existen cinco tipos de servicios, iremos a explicarlos brevemente:

- Constantes: Este tipo de servicio usa una constante al cual le pasamos directamente el valor del servicio dicho.
- Valores: Tiene la misma característica que la constante. Se le pasa directamente el valor de dicho servicio.
- Servicios: El servicio es un tipo que recibe un parámetro de una clase JavaScript y AngularJS se encargará de crear una instancia de aquella clase.
- Factorías: Podríamos decir que es lo mismo que un Servicio, pero la Factoría es más configurable y detallado. Con esto se define el concepto de libertad que ofrece las factorías a los servicios de AngularJS. La Factoría sería como una caja de herramientas para un servicio.
- Proveedores: El proveedor permite que se configure antes de crear el valor del Servicio, tendría el comportamiento de una factoría salvo por ese detalle.

2.2.4 Vistas en AngularJS

En este apartado veremos cómo AngularJS se reafirma en la idea de página de aplicación individual mediante múltiples puntos de vista en una sola página. Para esta finalidad, AngularJS tiene la etiqueta ng-view, directivas como ng-plantilla y servicios como $routeProvider. Definiremos el uso de cada elemento:

- Ng-vista: la etiqueta ng-vista solamente lo que hace es crear un marcador de posición en una vista dada (html por ejemplo). La etiqueta ng-vista está para colocarse sobre la base de la configuración *(Ver figura 9).*

```
<div ng-app = "miApp">
     ...
     <div ng-view></div>

</div>
```

Figura 9 - Uso del ng-view dentro del módulo principal

- Ng-Plantilla: Esta directiva se usa para crear vistas usando las etiquetas de html. Contiene el atributo "id" que es usado por $routeProvider para delegar una vista con un controlador. *(Ver figura 10)*

```
<div ng-app = "miApp">
   <script type = "text/ng-template" id = "accion.html">
      <h1> Accion </h1>
      {{mensaje}}
   </script>
</div>
```

Figura 10 - Uso del ng-plantilla basado en el id

- $routeProvider: la configuración de urls de la aplicación pasa por este servicio. Cuando se define una url con parámetro, este envía el html que corresponde dado con ng-template y lo asocia en el elemento ng-view.

Un ejemplo de este uso, estaría dividido en dos partes, la primera ya la definimos anteriormente en ng-plantilla. La segunda nos centraríamos en el bloque de script en el módulo principal y es en donde se ajustaría la configuración de enrutamiento *(Ver figura 11).*

```
var miApp = angular.module("miApp", ['ngRoute']);

miApp.config(['$routeProvider', function($routeProvider) {
   $routeProvider.

   when('/accion', {
      templateUrl: 'accion.htm', controller: 'accionController'
   }).

   when('/viewAccion', {
      templateUrl: 'viewAccion.htm', controller: 'ViewAccionController'
   }).

   otherwise({
      redirectTo: '/accion'
   });

}]);
```

Figura 11 - Uso del Route Provider

Aclaremos algunos puntos del anterior código:

- $RouteProvider está definido como una función que tiene configuración del módulo myApp usando la clave: $ routeProvider

- La línea $RouteProvider.when define la URL “acción”, a la cual luego se le asignará “addAccion.htm”. En este caso, la plantilla addAccion.html debería de estar presente en la misma ruta que el html.

- Cada controlador se usa para establecer la vista predeterminada de cada ruta.

2.3 Stack Mean (MongoDB, Express, AngularJS y NodeJS)

Entramos en un apartado muy importante a tratar ya que la realización de nuestro proyecto hacemos uso de esta tecnologia: “Stack Mean”. Definimos al Stack Mean como el paquete de software de tecnologia basada en JavaScript. La palabra MEAN proviene de los cuatro softwares más populares que se utilizan para crear cualquier aplicación web. Estos softwares son: MongoDB, ExpressJS, AngularJS y NodeJS[6]. Daremos una pequeña definición ahora para más adelante ampliarlo:

- MongoDB: base de datos que almacena documentos JSON.
- Express: es una infraestructura de aplicaciones web basado en NodeJS que permite por ejemplo crear API Rest.
- AngularJS: el principal protagonista del proyecto, simplemente decir que crea la parte cliente de la aplicación en formato Single Page.
- NodeJS: es una herramienta JavaScript que ofrece funcionalidades básicas para la aplicación bajo un modelo asíncrono de eventos.

La finalidad de la tecnología Stack Mean es la reordenación de códigos para cambiar la plataforma base de una aplicación. Cabe decir que, en la actualidad, es la tecnología más avanzada en el mundo empresarial. La tecnología Stack Mean es sencilla de usar y tiene la ventaja que funciona muy rápido, más que otras tecnologías.

Si buscamos un poco de historia de cómo nació la tecnología Stack Mean, hemos de decir que un hecho trascendente fue la revolución del AJAX. Debido a este hecho, el lenguaje JavaScript dejó de ser un lenguaje pequeño y ganó mucha popularidad. En ese proceso fue en donde nació NodeJS, que es un ambiente de ejecución JavaScript (desarrollado por Google). Con este acontecimiento, ya se tiene JavaScript también en el lado del Servidor. Se juega con más ventajas, ya que NodeJS sigue el paradigma de la programación orientada a eventos y el no bloqueo de entrada y salida de datos, con resultados óptimos para escenarios de aplicaciones en tiempo real.

Así pues, gracias a toda esta revolución, hoy en día se puede crear aplicaciones distribuidas utilizando el mismo lenguaje JavaScript en todas sus vertientes y capas. En un principio desde el lado cliente el JavaScript fue el denominador, pero después ya también en el servidor y en la capa de almacenamiento. En poco tiempo, ya tendríamos JavaScript listo para ser usado en todas las capes del desarrollo. De esta forma y gracias a las tecnologías que l

o posibilitan se le denominó: MEAN. De forma curiosa, cabe decir que por orden lógico debería de tener el nombre de ANEM o quizás también MENA. Mas, el acrónimo MEAN queda más claro y es muy llamativo, ya que en el inglés de la calle significaría: desconsiderado, presumido, y por otro lado también algo sofisticado.

Ahora pasemos a definir con mayor amplitud cada herramienta de lo que compone Stack Mean:

- Node.js: Como curiosidad, cabe mencionar que JavaScript apareció en el lado servidor casi al mismo tiempo que el navegador web. Netscape, empresa que creó el lenguaje, lo añadió también en el servidor web bajo el nombre de Netscape Enterprise Server, pero con poco éxito. Después de esto, nuevamente un proyecto puso al JavaScript nuevamente en el lado servidor, y este sería Node.js. Node.js, de código abierto, es un entorno de ejecución de aplicaciones multiplaforma. Node.js hace servir como base el motor de JavaScript de Google, denominado V8. Asimismo, provee de una arquitectura orientada a eventos, así como una serie de APIs no-bloqueantes (asíncronas), además de incorporar un módulo para poder comportarse como si fuera un servidor web. Esto último es especialmente óptimo para crear aplicaciones web, y es algo que hemos hecho en nuestro proyecto.

- Express: este framework nació para ayudar a crear aplicaciones web de una forma más sencilla. Escrito en JavaScript, está ligado con Node.js. Decimos esto porque Node.js es muy complejo, costoso y se tiene que hacer el trabajo a bajo nivel si queremos configurarlo. Para eso está Express, para que no tengamos que reinventar nada cada vez que se quiera crear una aplicación web, dando soporte para las principales necesidades: gestión de peticiones y respuestas, cabeceras, rutas, vistas, etc. Sin Express, actualmente sería muy costoso construir aplicaciones web con JavaScript, así que por suerte ambos van muy unidos para crear aplicaciones web de forma más sencilla.

- AngularJS: el motivo de este presente proyecto es esta herramienta. Cabe decir que en el navegador se podría desarrollar solo con HTML, CSS y JavaScript, pero tener una biblioteca que nos dé funcionalidades ya hechas, ahorra mucho el trabajo y da facilidades. Una de estas bibliotecas de funciones y conocida sería JQuery. Pero con el boom del patrón de diseño MVC (Modelo-Vista-Controlador) trasladado al navegador, aparecieron muchas bibliotecas especializadas en hacer fácil su uso. Y he aquí que entra en escena nuestro gran protagonista: AngularJS, creada y soportada por Google, gratuita y de código abierto. Como apunte, decir que muchos han definido a Angular como “lo que HTML debería haber sido si se hubiera diseñado para crear aplicaciones”. Ya se ve entonces la importancia que tiene.

- MongoDB: Centrándonos en la base de datos, normalmente se usaban base de datos relaciones. Pero en la actualidad al existir y desarrollar aplicaciones web que solicitan mayor flexibilidad y mayor capacidad escalar. Por esta situación, nació la tecnología en almacenes de datos que se pasó a llamar NoSQL. Un almacén de datos no-relaciones hay uno especialmente que surge y tiene éxito entre los demás: MongoDB, y que es la “M” del stack MEAN.

Y es justo mencionar y detallar de otro elemento que no está dentro del acrónimo MEAN pero que juega un papel muy importante también en esta tecnología: JSON (JavaScript Simple Object Notation). *(Ver figura 12).*

Tal y como su nombre lo indica JSON, permite representar objetos (estructuras complejas) en forma de código JavaScript que luego se podrá revisar. JSON sigue la línea de formato ligero y que es muy sencillo leer tanto para una máquina como para la persona que desarrolla. Además, está basado en JavaScript. Diríamos que JSON es el pegamento de todas las capas. El formato en el que se comunican los datos entre todos los niveles de la aplicación: navegador, servidor web y servidor de datos.

Con toda esta información, vemos que esta tecnología de la cual nos hemos hecho servir para el control de usuario de nuestra aplicación, es eficaz para tener una buena organización y optimización de todo nuestro proyecto, basado únicamente en JavaScript.

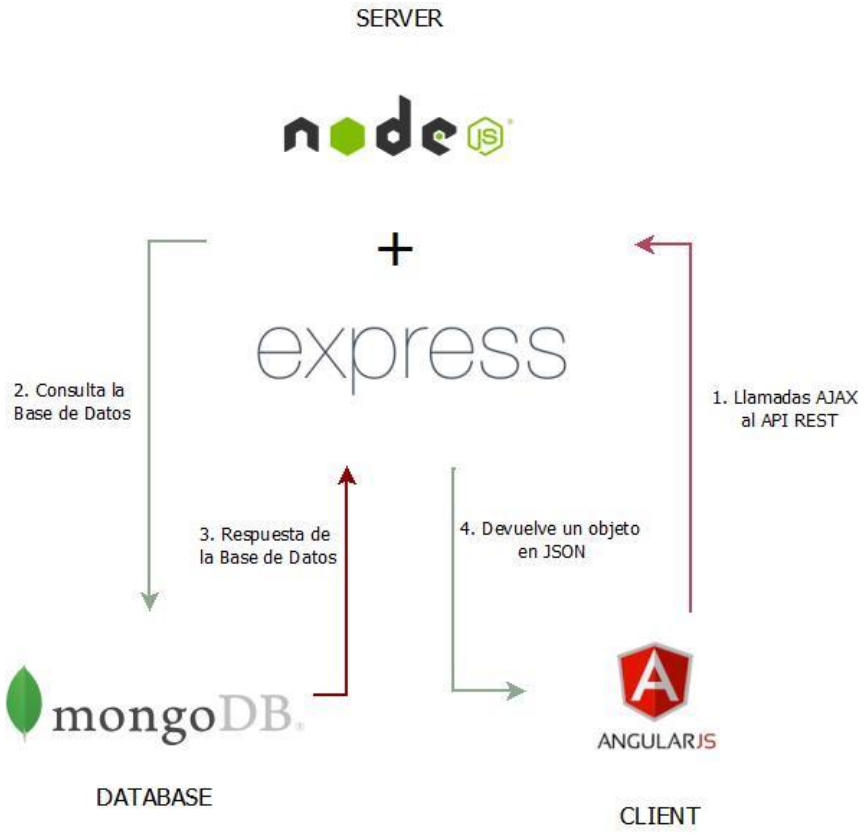

Figura 12 - Organización y Desarrollo de la tecnología Mean Stack

2.4 Autenticación basada en token: JWT Token

Toda aplicación web debe de tener una parte esencial llamada: autenticación. En nuestro proyecto por supuesto lo hemos incluido, pero de una forma especial, ya que nos hemos basado en autenticación en token. Existen diferencias lógicas con los sistemas tradicionales de login y explicaremos las diferencias [7].

Comentaremos cómo funcionan los sistemas de autenticación tradicionales:

1) El usuario introduce un nombre de usuario y contraseña en el formulario de login para iniciar sesión.

2) Cuando se hace la petición, se valida el usuario en el back-end mediante una consulta a la base de datos. En caso de que la petición sea exitosa, se crea una sesión usando la información de usuario que extrae de la base de datos. Después se devuelve la información de la sesión en el encabezado de la respuesta, para que de esta forma se almacene el ID de sesión en el navegador.

3) Se ofrece la información de sesión para acceder a ciertas rutas de la aplicación web que estarán restringidos.

4) Si la información es válida, se permite al usuario acceder y ver rutas específicas, y se responde con el contenido HTML renderizado.

Con todo esto, se puede montar un buen sistema de autenticación, con usuarios registrados, que pueden iniciar sesión, registrarse y además tener rutas específicas de acceso, pero el problema está en la incompatibilidad que se tiene con los clientes móviles y dotarles también de restricción. Existen motivos por los cuales esto ocurre:

1) Las cookies y las sesiones son ilógicas en aplicaciones móviles. Es imposible compartir sesiones o cookies creadas en el servidor con los clientes móviles.
2) En una aplicación tradicional siempre se devuelve un HTML renderizado. En un cliente de móvil, para dar respuesta se necesita incluir un JSON o XML.

Y es entonces que lo que se necesita es tener una aplicación independiente del cliente, y es aquí en donde empleamos la autenticación basada en token.

En la autenticación basada en token, las sesiones y las cookies no tienen uso. Por el contrario, se usa un token para autenticar al usuario en cada petición que se hace al servidor. Su funcionamiento sería el siguiente:

1) El usuario introduce un nombre de usuario y contraseña en el formulario de login y solicita el ingreso.
2) Cuando se hace la petición, se valida el usuario en el back-end mediante una consulta a la base de datos. Si esta petición es válida, se crea un token usando la información de usuario que otorgó la base de datos, y esa información se devuelve en el encabezado de la respuesta, para que de esta forma se guarde el token en el almacenamiento local.
3) Se le da la información del token en el encabezado de cada petición para acceder a rutas restringidas de la aplicación.
4) En caso que el token del encabezado sea válido, se permite al usuario acceder a la ruta específica, y se responde con JSON o XML.

Así entonces, hemos visto que no se ha devuelto ninguna sesión ni ninguna cookie, a la vez que tampoco se ha devuelto contenido HTML. Esto nos da a entender que se puede usar esta arquitectura para cualquier tipo de cliente en una aplicación que se presente.

La arquitectura de esta autenticación (*Ver figura 13).*

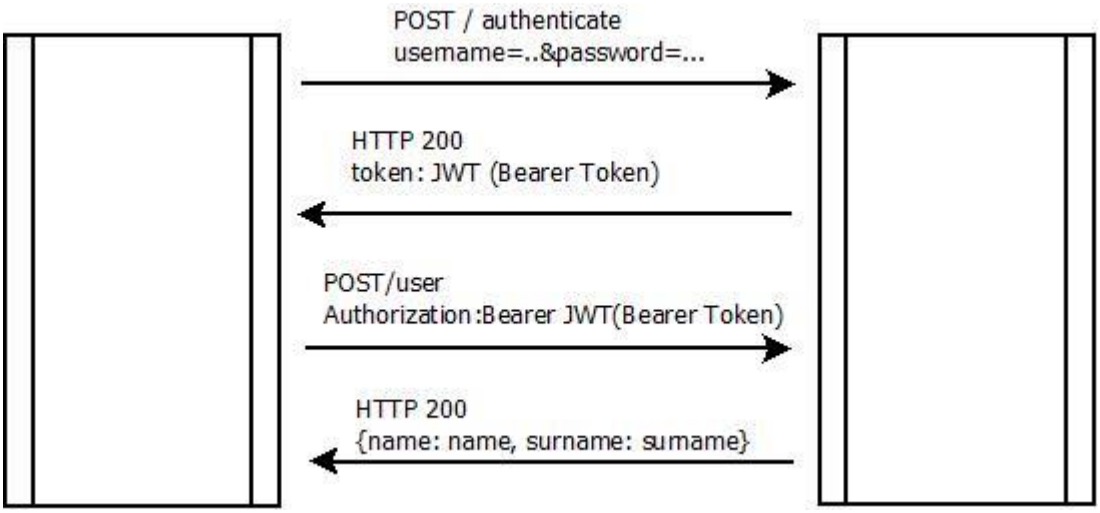

Figura 13 - Arquitectura de Autenticación

Por otro lado, mencionamos antes la sigla que acompañaba a Token: JWT. JWT es la sigla de Json Web Token, y es un formato de token utilizado en encabezados de autorización. Este token es muy funcional para la comunicación entre dos sistemas de forma segura. Llamaríamos a JWT como si fuera un token que transporta algo, un token portador. Definimos que un token portados consiste en 3 partes diferenciadas:

- El header que es la parte del token en donde se almacena el tipo de token y el método de encriptación.
- Toda la información se encuentra en la carga útil. Esto puede contener cualquier tipo de datos como información de usuario, información de producto, etc.
- La firma que es la combinación del encabezado, carga útil y clave secreta. La clave secreta debe ser almacenada del lado del servidor.

Poseer una autenticación basada en token, puede llegar a solucionar conflictos muy severos. Las ventajas más resaltantes serían:

- Servicios independientes del Cliente: el token es transferido por medio de los encabezados de petición, en vez de almacenarse en las sesiones o cookies. De esta forma diríamos que no existe estados, y así se puede enviar una petición al servidor desde cualquier tipo de cliente que haga peticiones HTTP.

- CDN: Una muy buena ventaja es este punto, el cual destierra la dependencia de la lógica front-end con el del código back-end. En algunas aplicaciones de web actuales, las vistas se renderizan en el back-end y el contenido HTML se devuelve al navegador. Como hemos dicho, esta dependencia vuelve a la aplicación inestable si se desea en el futuro migrar una parte. Por eso, en la autenticación basada en token, se puede desarrollar un proyecto front-end separado del código back-end. El código back-end devolverá un JSON en vez de un HTML renderizado, y se podrá colocar en el CDN la versión modificada y descomprimida del código front-end. El funcionamiento será que cuando se navegue por la web, el contenido HTML se servirá desde el CDN, y el contenido de la página se cargara a través de servicios de la API usando el token en los encabezados de autorización.

- No Sesion Cookie (No CSRF): Las fuentes de peticiones pueden provenir de sitios no tan confiables y CSRF no comprueba estos datos. La solución pasaría por usar un banco de tokens para enviar aquel token en cada envío de formulario. En la autenticación basada en token, aquel token se usa en los encabezados de la autorización.

- Almacenamiento Persistente de Token: El problema radica en el momento en el que se hace una lectura, escritura o borrada de una sesión en la aplicación, también se realiza una operación de archivo en la carpeta temporal del sistema operativo. En el caso que se tenga múltiples servidores y que se crea una sesión en el primero de ellos, cuando se realiza otra petición y esta va a otro servidor, la información de sesión no existe aún, con lo cual se obtiene una respuesta de: “no autorizado”. En la autenticación basada en token, este conflicto se resuelve de forma natural, no existe este problema, ya que el token de la petición se intercepta en cada petición, y en cualquier servidor.

2.5 Casos de Uso

En el diagrama de Casos de Uso se representan todas las acciones disponibles que tiene el Usuario. *(Ver figura 14).*

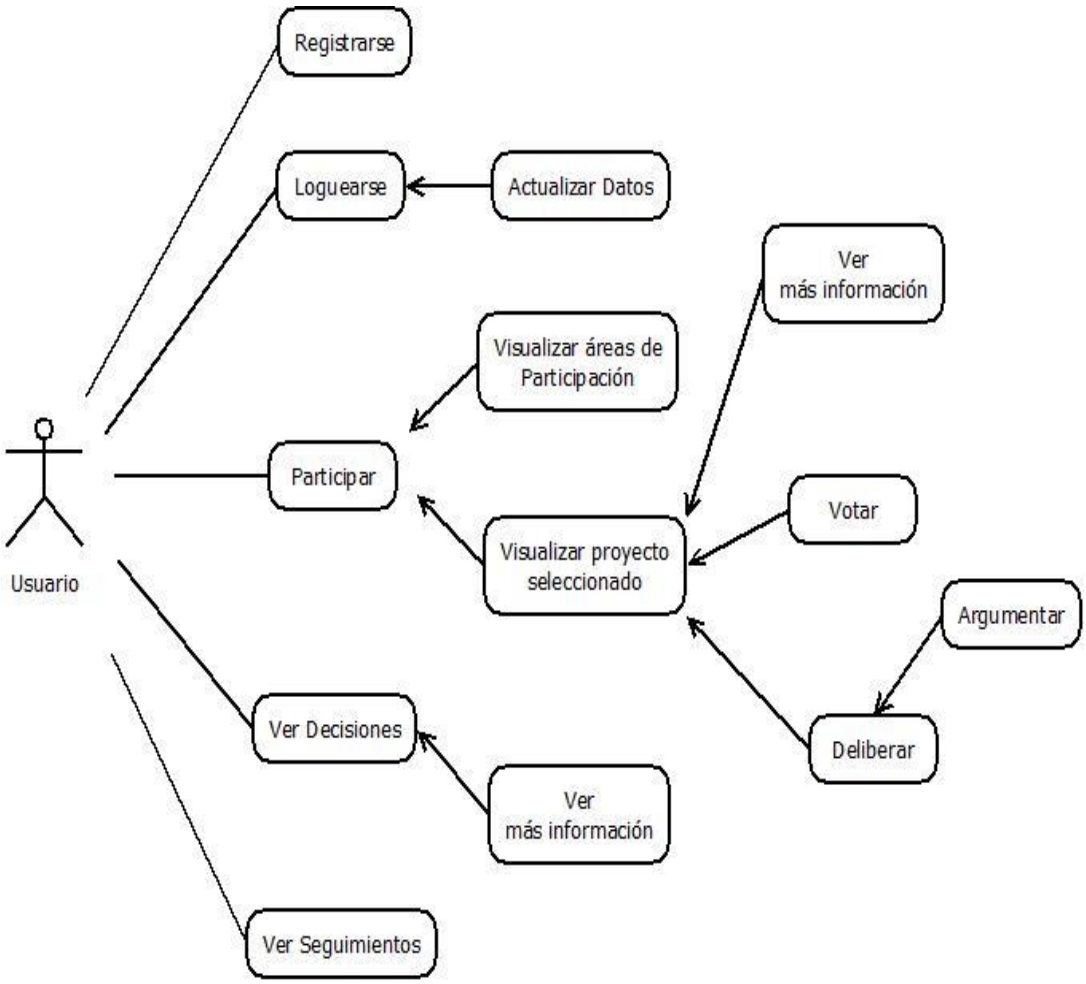

Figura 14 - Diagrama de Casos de Uso

3. Diseño.

En este punto, explicaremos el uso de la tecnología que hemos hecho servir para la construcción de nuestro proyecto, además de cómo se ha organizado la estructura del código y que diferencias existen entre la herramienta empleada con las demás que existen en la actualidad.

3.1 Uso de la tecnología

Tal y como hemos descrito en los puntos anteriores, AngularJS es nuestro gran protagonista y nos hemos decantado por este framework partiendo de la concepción que se basa en el patrón MVC, un patrón que hemos visto a lo largo de la carrera en materias de software y que ya conocíamos. Además, es atrayente ya que aprendimos con ello:

- Desacoplar la manipulación del DOM de la lógica de aplicación.
- Desacoplar el lado cliente del lado servidor en una aplicación.
- Construir una estructura para el desarrollo de una aplicación.

Nuestro uso de AngularJS podríamos resumirlo en la creación de SPA's (Single Page Apps), ya que con esto conseguimos mayor fluidez posible en el diseño. La comunicación entre el cliente y servidor es constante y transparente y da la sensación que no salimos de la página principal de nuestra aplicación. Las ventajas que nos aporta Angular y las cuales ya hemos comentado anteriormente sería el Two way data binding (vista y modelo están en relación constante, y por ello todo cambio visual se actualiza a tiempo real en el modelo y viceversa). Las directivas también es una buena ventaja para su uso, ya que, gracias a ellos, se puede trabajar muy fácil a nivel de componentes, siendo estos componentes reutilizables en toda la aplicación global. Las directivas en si no son más que marcadores en un elemento de DOM que indican al compilador de Angular que aquel elemento tiene un comportamiento específico.

AngularJS está en expansión y es ahí también uno de los motivos por los cuales nos decidimos en usarla. Las principales plataformas que usan AngularJS hoy en día son: Virgin Mobile, Amazon, Forbes, PayPal, Gmail, etc.

Además de la principal tecnología que utilizamos, también nos hacemos servir de las siguientes:

- Hoja de Estilo CSS:

  La sigla de este nombre proviene de "Cascading Style Sheets" que traducido sería Hoja de Estilo en Cascada. Es un lenguaje simple que describe de qué manera se va mostrar un documento en la pantalla o de qué forma se va imprimir. El uso principal que proporciona CSS es para dar estilo a los documentos HTML y XML

dividiendo el contenido de la presentación. El lenguaje CSS permite a los desarrolladores web tener el control total del estilo y el formato de varias páginas web al mismo tiempo. El funcionamiento del CSS se define por reglas: declaraciones sobre el estilo de uno o más elementos. Las hojas de estilo están formadas por una o más de aquellas reglas que se aplican a un documento HTML o XML. Una regla tiene dos partes: un selector y la declaración. Y, además, la declaración está formada por una propiedad y el valor que se le asigne. Por ejemplo:

P {color: blue;} → P es el selector, {color: blue} es la declaración, color es la propiedad y blue es el valor que se le asignó.

En este proyecto incluimos tanto librerías de CSS de las propias tecnologías que hacemos servir, así como CSS personalizados construidos para ciertos diseños propios.

- Bootstrap:

Y he aquí en donde para encontrar la solución de presentar este proyecto como una página web con diseño responsivo, nos hacemos valer de este poderoso framework de front-end gratuito llamado Bootstrap. Bootstrap incluye plantillas de diseño que tienen su base en HTML y CSS para formularios, tablas, navegación, tipografía, etc. Además, incluye complementos JavaScript opcionales. Antes, habíamos mencionado el término diseño web responsivo a lo que se refiere en poder ajustar automáticamente los sitios web para verse correctamente en todos los dispositivos, desde los teléfonos pequeños hasta las grandes pantallas de escritorio.

La facilidad de su uso hizo que nos fuera muy óptimo usarlo, ya que basta con conocimiento básico de HTML y CSS para poder partir bien. La base de Bootstrap y el marco central son los dispositivos móviles. *(Ver figura 15).*

Figura 15 - CSS y Bootstrap, muy ligados al diseño web

- D3 JS:

  Librería de JavaScript el cual hacemos uso para graficar los datos que presentamos en nuestra aplicación. Esta librería sirve para producir a partir de los datos (Data Driven Documents), gráficos dinámicos e interactivos en un navegador web. Para ello, hace servir tecnologías como HTML5, CSS y SVG. Diríamos que su principal ventaja es que D3.js posee el control total sobre el resultado final. *(Ver figura 16).*

```
d3.select("#var ul").selectAll("li")
    .data([1,2,3,4])
    .enter().append("li")
    .text(function(d){return d;});
```

Figura 16 - Código básico de uso de la librería 3D.js

  Esta librería es absorbida dentro de una página web de HTML e utiliza funciones JavaScript predefinidas construidas para seleccionar elementos, se crea objetos SVG, se les da estilo, añade transiciones y efectos dinámicos. Cabe decir que a estos objetos inclusive se les puede aplicar estilos usando CSS[8].

3.2 Organización y arquitectura del sistema.

La arquitectura y organización de carpetas que hemos seguido, tiene como referencia cualquier proyecto hecho bajo el sello de AngularJS. Nuestra idea era poder organizar el proyecto para que se puedan encontrar de una forma rápida y sencilla las cosas. Como es un proyecto basado en AngularJS, los elementos que podríamos tener serán:

- Vistas: Aquí almacenaríamos todos los archivos HTML, EJS con contenido estático y las directivas de angular.
- Controladores: Se reunirá a todos los controladores de Angular
- Servicios: Factoría de Angular que realiza las llamadas a los servicios Web.
- Una carpeta de inicio de aplicación donde contendrá el corazón de angular, así como la página de inicio y los demás archivos de aplicación como suelen ser imágenes, hojas de estilo, scripts propios o de las librerías y tecnologías usadas, etc.

Sabiendo esto, ya podemos ver como iría la estructura de nuestro proyecto. Organizado por carpetas y archivos que permitan ubicar a cada elemento o ítem en un lugar específicamente predeterminado *(Ver figura 17).*

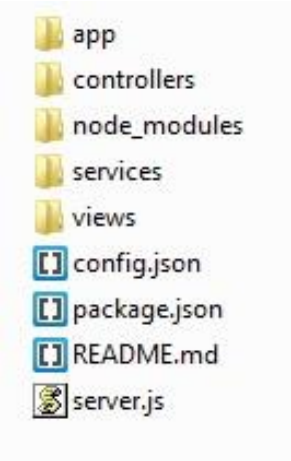

Figura 17 - Estructura de Proyecto

En esta estructura de carpetas, organizamos los elementos que anteriormente los hemos definido:

- App: contendrá el index, y el javascript central de angular, además de las carpetas de cada ítem de la aplicación, junto con la carpeta de contenido: imágenes, hojas de estilo, fuentes, etc.

- Controllers: ficheros de controladores, como por ejemplo del login y registro de usuario.
- Services: fichero que contendrá los servicios del usuario.

- Views: archivos ejs con contenido estático, como por ejemplo la cabecera o el pie de página de la aplicación.

- Config.json: es el archivo de configuración Express. Contiene los datos de acceso para la conexión del Node.js con la aplicación.

- Package.json: archivo en donde definimos las dependencias, repositorio, etc.

- Server.js: es el archivo del servidor Express en donde iniciaría la aplicación Node.js. Además, se define la configuración de la aplicación, enlaza los controladores con las rutas e inicia el servidor http.

3.3 Comparativa entre otras aplicaciones.

Considerando que nuestro proyecto es una evolución de la idea de una herramienta de participación ciudadana para el consenso de normas, entendemos que tiene ciertas diferencias ahora que son procesos participativos, vinculantes y transparentes. Las principales aplicaciones existentes sobre participación ciudadana a comparar son:

- Decide Madrid: en el portal Decide Madrid el sistema de votaciones comienza por responder si o no a unas preguntas sobre el proyecto en el cual se decide participar. Además, se ofrece los detalles de los proyectos finalistas que han sido seleccionados. Este proceso no es anónimo.

- Considerit.it: en esta aplicación se considera que el usuario inicia la acción enviando preguntas o ideas a la comunidad que elija. Después se le irá mostrando las áreas que tienen mayor acuerdo y menor acuerdo y una explicación razonada para una toma de decisión. Entre los usuarios existirá debate para que la participación crezca y sea más inteligente. La forma de votación es haber aprendido a dialogar con los demás usuarios de la comunidad para que entre todos busquen un proyecto que sea resuelto. En el caso de Considerit.it hay la opción de declarar anónima una opinión.

- Pol.is: Esta herramienta de participación ciudadana, permite que cualquier usuario pueda contribuir antes, durante y después de un debate iniciado. Los usuarios que participan de la discusión pueden valorar y volver a debatir a tiempo real, y de esta manera, observar qué proyectos o propuestas tienen mayor éxito entre la comunidad de participantes. El proceso de votación no es anónimo y se asemeja con nuestra aplicación en el número de votos posibles: de acuerdo, no de acuerdo, indeciso.

- Appgree: en esta plataforma se ofrece la posibilidad de hacer debates y votaciones a tiempo real (como Pol.is), pero teniendo el recuento de votos muy rápido. Esto quiere decir que, aunque haya participaciones masivas, los debates serán ágiles. Su funcionamiento se basa en la división de usuarios mediante un reparto aleatorio en tantos grupos como proyectos a valorar existan. Con esto, cada grupo valora un proyecto y se asume que ese grupo representa el parecer de todo el conjunto de usuarios. En Appgree el proceso de votación es anónimo y la votación tiene un voto positivo o negativo.

4. Desarrollo

El presente proyecto se construyó para ser una aplicación web en donde su uso sea intuitivo y fácil para el usuario. El sitio web está alojado en un servidor web, así que los usuarios pueden conectarse a él mediante una navegación normal. Los programas para su desarrollo que hemos utilizado los describimos a continuación.

4.1 Lista de programas para el desarrollo de la aplicación

- Brackets.io: es un editor de código abierto para el desarrollo diseño web construido con las tecnologías HTML, CSS y JavaScript. Fue creado y es mantenido por Adobe. El motivo para decantarnos por este programa es la facilidad que tiene para mostrar el código específico de acuerdo al contexto que usamos y además que nos permite trabajar directamente en el navegador editando el código inmediatamente.

- SourceTree: como todo proyecto que tiene un desarrollo largo, es necesario subirlo a un repositorio de Git, de esta forma podemos trabajar con el código y tenerlo disponible en cualquier momento y lugar. Para facilitarnos este trabajo, utilizamos SourceTree. Nos permite visualizar nuestro repositorio mediante su interfaz sencilla, además de permitirnos revisar los cambios provisionales que hemos hecho en nuestro código y así elegir los archivos que queremos subir o descartar. Como hemos trabajado en una misma rama, no tenemos problemas de conflictos entre los ficheros editados.

- MongoDB: explicado anteriormente, solo basta decir que es una base de datos orientada a documentos y el cual no es necesario que se siga un esquema. Los datos de los usuarios registrados en nuestra aplicación son almacenados y el servidor es iniciado desde la consola de Windows con el comando mongod.exe. Con este comando arrancamos el servicio mongod que comenzará a escuchar peticiones por el puerto 27017.

4.2 Implementación e integración de las tecnologías

Tal y como hemos descrito en el apartado 3.2 (Organización y arquitectura del sistema), el proyecto une las tecnologías AngularJS, NodeJS, Express y MongoDB. Además de esto, también usamos CSS, JQuery, Bootstrap y la librería de gráficos D3.js. Como toda aplicación web, tiene su punto inicial en como un usuario puede registrarse en la aplicación, tener su cuenta y poder iniciar sesión en el portal.

- Controlador de la cuenta de usuario:

  Este controlador será para la sección de cuenta de usuario en nuestra aplicación angular. El objeto de usuario lo pondremos visible para la vista e implementamos métodos para actualizar o eliminar la cuenta del actual usuario. *(Ver figura 18)*

```
angular
    .module('app')
    .controller('Account.IndexController', Controller);

function Controller($window, UserService, FlashService) {
    ...
    function saveUser() {
        ...
    }
    function deleteUser() {
        ...
    }
}
```

Figura 18 - Controlador de la cuenta Usuario

- Vista de la cuenta de usuario:

  Esta vista será para la sección de cuenta de usuario. Contiene un formulario para actualizar los datos del usuario o eliminarlo. *(Ver figura 19)*

Figura 19 - Vista de la cuenta de usuario

- Servicios para mensajes informativos y para los usuarios en la aplicación:

  Estos ficheros javascript's que actuarán como servicios estarán en la carpeta app-services. Todos los accesos API y la lógica de negocio estarán en esta carpeta para así mantener aparte los controladores y respetar esta separación delegando responsabilidades a quien toca. En el caso de los mensajes informativos, el servicio de mensajes "flash" se usa para mostrar mensajes de éxito o error en la aplicación angular a través de $rootScope. Este mensaje lo expone en el archivo principal: (app/index.html). *(Ver figura 20)*

Figura 20 - Mensaje flash de error en dato incorrecto

Y en el caso de los usuarios, el servicio encapsula la interacción con el api web para las operaciones CRUD que tienen que ver con el usuario (Crear, Leer, Actualizar y Borrar). *(Ver figura 21).*

```
angular
    .module('app')
    .factory('UserService', Service);

function Service($http, $q) {
    var service = {};
    ...
    service.Create = Create;
    service.Update = Update;
    service.Delete = Delete;

    return service;

    function Create(user) {
        return $http.post('/api/users', user).then(handleSuccess, handleError);
    }
    function Update(user) {
        return $http.put('/api/users/' + user._id, user).then(handleSuccess, handleError);
    }
    function Delete(_id) {
        return $http.delete('/api/users/' + _id).then(handleSuccess, handleError);
    }
}
```

Figura 21 - Servicio del usuario

- Controlador para la página de inicio: bienvenida:

  El controlador de la página de bienvenida es el controlador para la sección de inicio. Este controlador obtiene el usuario actual y lo muestra en la vista. *(Ver figura 22).*

```
angular
    .module('app')
    .controller('Home.IndexController', Controller);

function Controller(UserService) {
    var vm = this;
    vm.user = null;
    initController();

    function initController() {
        UserService.GetCurrent().then(function (user) {
            vm.user = user;
        });
    }
}
```

Figura 22 - Controlador: inicio de la aplicacion

- Vista para la página de inicio: bienvenida:

  Esta página de inicio es la vista predeterminada para la sección de inicio. Mostramos el mensaje de bienvenida para el usuario, así como las opciones a realizar *(Ver figura 23).*

Figura 23 - vista de bienvenida a la aplicación

- Apps.js: Punto de entrada para nuestra aplicación AngularJS

  En este fichero javascript se inicia nuestra aplicación angular. Definimos a nuestro modulo con el nombre de “app” y junto con las dependencias que necesitamos y explicaremos más adelante, contendrá la lógica de configuración y de inicio para cuando se cargue la aplicación por primera vez. La función de configuración se encarga de definir las rutas de la aplicación mediante Router Angular, explicado anteriormente. Cabe mencionar también que agregamos la función run que añade token JWT como autorización predeterminada para todas las solicitudes http, necesario además para autenticarse en el api web *(Ver figura 24).*

```
angular
    .module('app', ['ui.router'])
    .config(config)
    .run(run);

function config($stateProvider, $urlRouterProvider) {
    // default route
    $urlRouterProvider.otherwise("/");

    $stateProvider
        .state('home', {
            url: '/',
            templateUrl: 'home/index.html',
            controller: 'Home.IndexController',
            controllerAs: 'vm',
            data: { activeTab: 'home' }
        })
        ...
}

function run($http, $rootScope, $window) {
    ...
}
```

Figura 24 - app.js: el fichero más importante en Angular

- Vista principal de la aplicación angular

  Este archivo index.html es el archivo principal para nuestra aplicación angular. Contiene la plantilla general para la aplicación, con lo cual todas las vistas de la aplicación tendrán el mismo molde *(Ver figura 25).*

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Cogobierna</title>

</head>
<body class="container">
    <!-- header -->
    <header>
    <!-- creacion de la cabecera de la aplicacion -->
    </header>

    <!-- main : llamada a cada vista de nuestra aplicacion -->
    <main ui-view></main>

    <!-- creacion del pie de pagina de la aplicacion -->
    <footer></footer>

    <!-- external scripts -->
    <!-- llamadas a scripts necesarios para la aplicacion -->
    <script src="app-content/js/angular/angular.min.js"></script>
</body>
</html>
```

Figura 25 - Vista principal de nuestra aplicación

- Vista: Participa

  Nos adentramos en la aplicación y en la parte funcional de votación. Al usuario se le presenta las seis áreas en las cuales puede participar. Estas seis áreas están distribuidas en un gráfico construido con la librería 3d.js. Para esto, incluimos la dependencia de aquella librería en específico: angular-d3-sunburst en nuestro fichero app.js. En este fichero, en la función configuración añadimos nuevos estados para agregar su respectivo controlador que obtendrá los datos dependiendo del área en el que se encuentra el usuario. Estas áreas son las siguientes: Urbanismo, Transporte, Cultura y Deporte, Salud, Movilidad y Educación. *(Ver figura 26)*

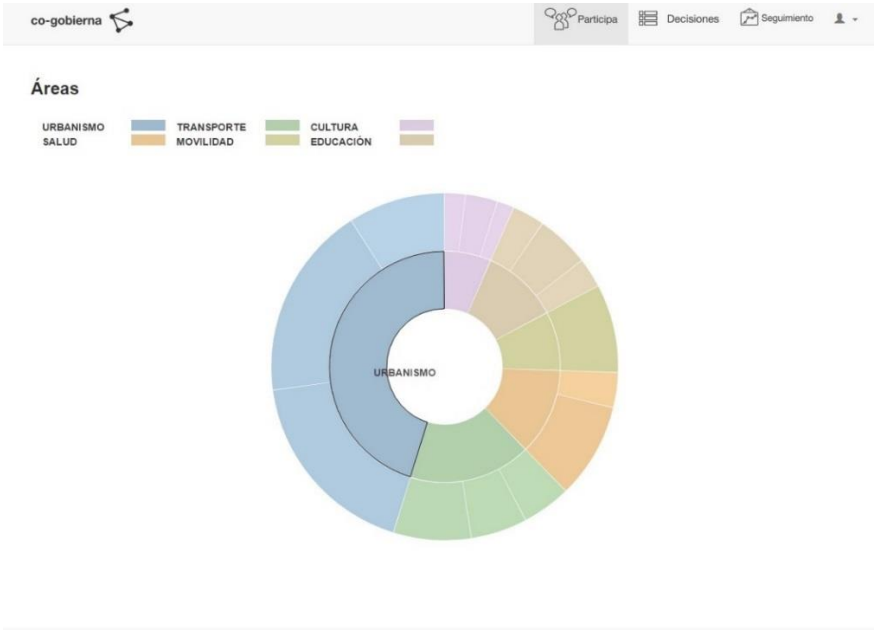

Figura 26 - Áreas de participación ciudadana

La funcionalidad del gráfico tiene la característica de poder seleccionar o bien el área para ver los proyectos en una segunda gráfica generada o bien entrar en cada proyecto directamente. Ambos enlaces nos dirigen en la vista de cada proyecto. Por ejemplo, en el proyecto "El Cañaveral" del área Urbanismo tendríamos la vista de ese proyecto indicándonos con el mismo gráfico el área donde nos encontramos además de la breve información que sirve como introducción *(Ver figura 27).*

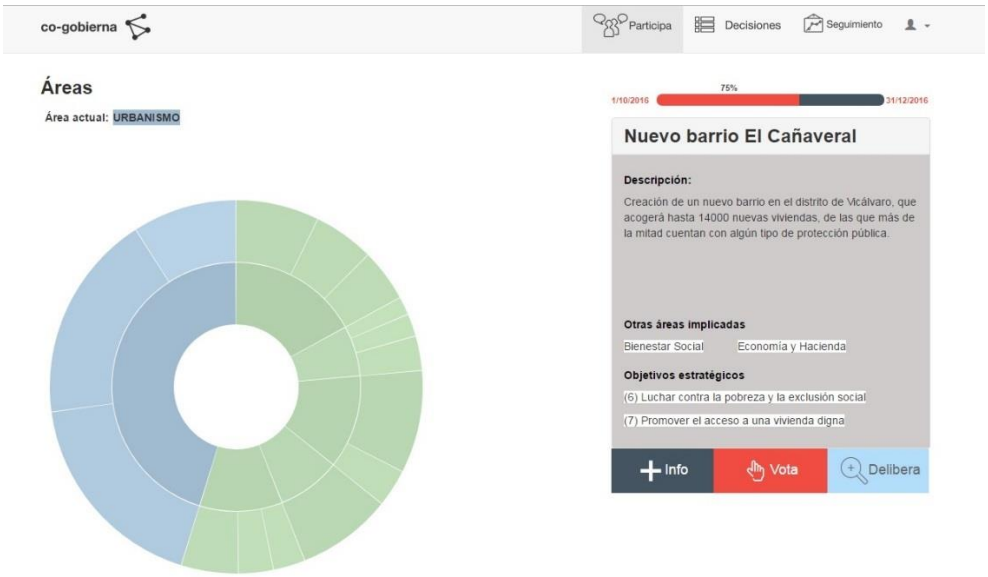

Figura 27 - Vista del proyecto seleccionado

Cabe señalar que cada área lo hemos diferenciado por un color distinto. Para esta diferencia, hemos indicado un nuevo módulo en la librería angular-d3-sunburst:

d3.scaleOrdinal(d3.schemePastel1);

Este módulo proporciona esquemas de colores secuenciales. Este esquema de colores puede trabajar con d3.scaleOrdinal. Para su instalación usaremos las siguientes librerías:

<script src="https://d3js.org/d3-color.v1.min.js"></script>
<script src="https://d3js.org/d3-interpolate.v1.min.js"></script>
<script src="https://d3js.org/d3-scale-chromatic.v1.min.js"></script>

Volviendo a la vista de participación, podemos seleccionar las tres alternativas propuestas: más información del proyecto, votar el proyecto o deliberar el proyecto. Estas tres alternativas las definimos como estados dentro del fichero app.js. *(Ver figura 28)*

```
.state('participa/proyectoUrba1/masinfo',{
    url: "/participa/proyectoUrba1/masinfo",
    templateUrl: 'participa/proyectos/urbanismo/proyecto1info.html',
    controller: 'ParticipaUrbanismo.ActionController',
    controllerAs: 'vm'

})
.state('participa/proyectoUrba1/vota',{
    url: "/participa/proyectoUrba1/vota",
    templateUrl: 'participa/proyectos/urbanismo/proyecto1vota.html',
    controller: 'GaugeCtrl.ActionController',
    controllerAs: 'vm'

})
.state('participa/proyectoUrba1/delibera',{
    url: "/participa/proyectoUrba1/delibera",
    templateUrl: 'participa/proyectos/urbanismo/proyecto1delibera.html',
    controller: 'GaugeCtrl.ActionController',
    controllerAs: 'vm'

})
```

Figura 28 - Estados que describen las opciones de un proyecto

Con esta organización, definimos cada proyecto existente en nuestra aplicación usando estados. Esto es posible gracias al módulo ui-router que nos ayuda mucho en nuestra navegación de vistas que es un tanto compleja.

La primera opción (más información del proyecto), también posibilita poder votar desde aquella vista o bien deliberar *(Ver figura 29)*

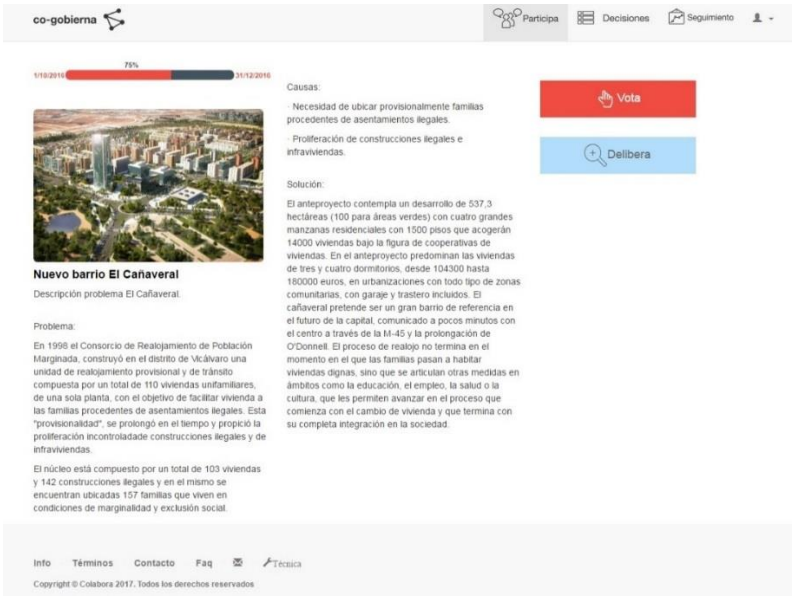

Figura 29 - Vista sobre más información de un proyecto

La segunda opción (votar por un proyecto), nos ofrece el gráfico de porcentaje de decisión sobre ese proyecto. Tenemos tres casos: A favor, no decide y En Contra. Este gráfico también está construido usando la librería d3.js. Seguimos con el mismo ejemplo que en el anterior gráfico, incluimos su dependencia llamada ui.gauge en nuestro fichero app.js:

angular.module('app', ['ui.router','angular-d3-sunburst','ui.gauge'])

Es importante decir que la integración de esta librería d3.js con Angular.js se hizo manualmente separando el controlador, la vista y los datos independientemente, ya que por defecto los script's de esta librería d3.js viene todo unificado. Esto aplicado a AngularJS directamente no es funcional, por ello se tiene que estudiar el código de la librería 3D.js para su integración con el modelo de AngularJS. Su controlador como observamos en la figura 24, se llama GaugeCtrl y se aplica en cada vista del gráfico de porcentajes para votar en un proyecto seleccionado *(Ver figura 30)*

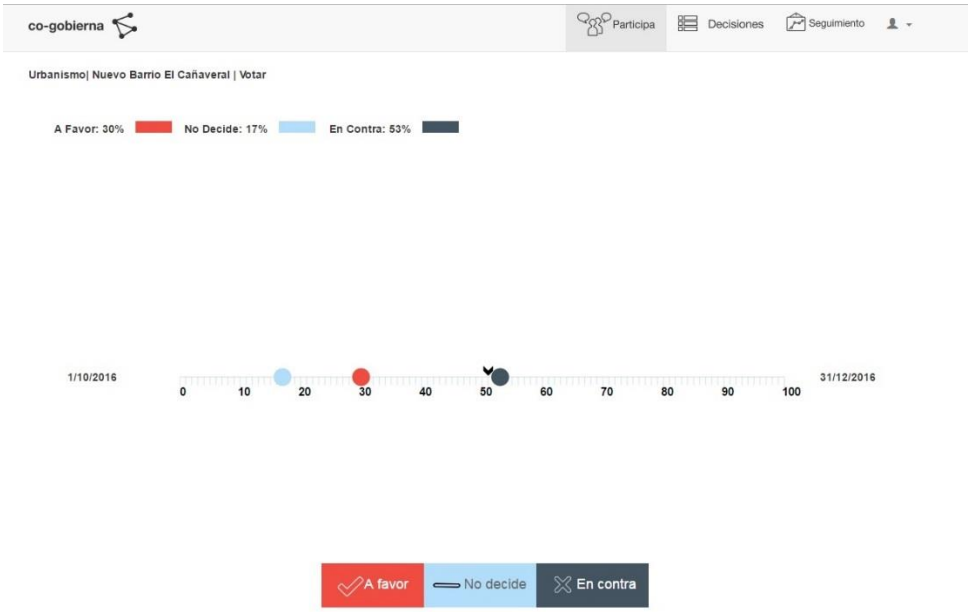

Figura 30 - Vista de un proyecto para votar

Según el color indica el porcentaje de cada caso para votar en el gráfico, el cual obtiene los datos desde el controlador. Cabe decir que en el caso de no decidir no afectará en ningún caso las votaciones a favor ni en contra. Una vez votado el proyecto se vuelve a la página de inicio para ver otras propuestas *(Ver figura 31)*

```
angular
    .module('app')
    .controller('GaugeCtrl.ActionController', function($scope) {

    $scope.unit = 100;
    $scope.minorUnit = 1;
    $scope.majorUnit = 10;
        $scope.options = {
            size: 800,
            scale: {
                baseLineColor: "silver",
                majorUnitColor: "grey",
                minorUnitColor: "rgba(52,152,219,.5)"
            },
            markers: [{ unit: 30, color: "#f04c41" }, { unit: 17, color: "#b2defb" }, { unit: 53, color: "#445461" }]
        };
```

Figura 31 - Controlador del sistema de datos para votar

Para deliberar, tenemos la opción de visualizar nuevamente el gráfico de porcentaje de votos y así poder argumentar a favor o en contra el proyecto. La funcionalidad de poder introducir comentarios lo basamos en un nuevo controlador de AngularJS: FrmController. Este controlador añade un comentario a través de la variable $scope que se activa al pulsar el botón de añadir *(Ver figura 32)*

```
(function () {
    'use strict';

angular
    .module('app')
    .controller('FrmController', function($scope) {
        $scope.comment = [];
                $scope.btn_add = function() {
                    if($scope.txtcomment !=''){
                    $scope.comment.push($scope.txtcomment);
                    $scope.txtcomment = "";
                    }
                }

});
```

Figura 32 - Controlador para agregar nuevo comentario

La vista muestra el proyecto en el cual estamos actualmente y argumentos hechos con anterioridad si es el caso. Cada proyecto tiene esta visualización: argumentos a favor y argumentos en contra *(Ver figura 33).*

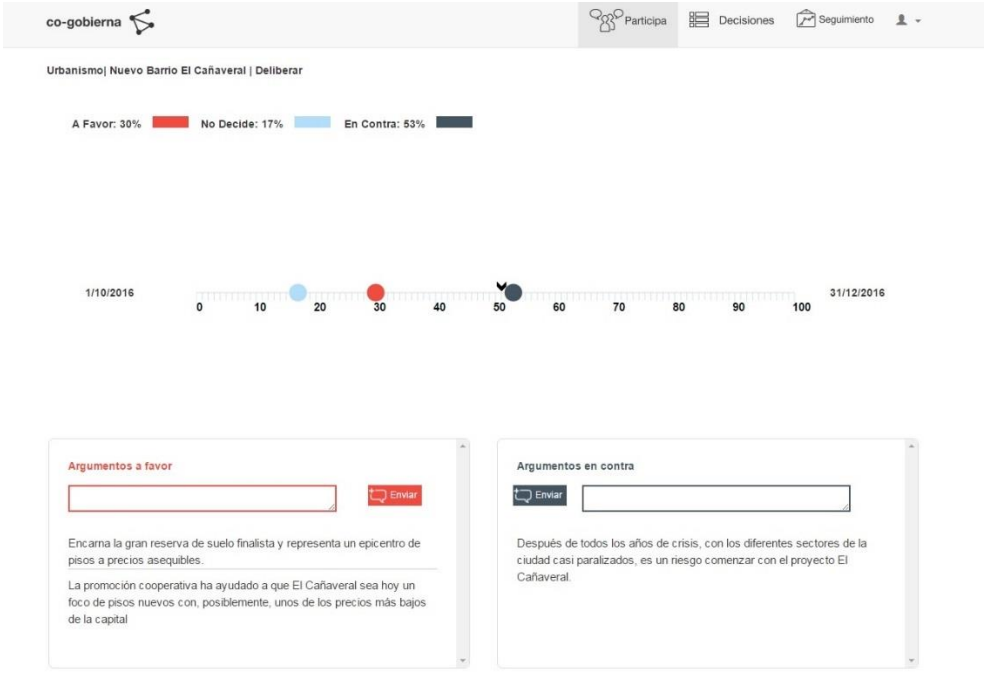

Figura 33 - Vista para añadir argumentos en un proyecto

- Vista: Decisiones

  En esta vista, se presenta un listado de todos los proyectos indicando cuales son los seleccionados y cuáles no. A su vez, hay un enlace para cada proyecto para ver el motivo de aquella decisión. Habrá un estado diferente para cada enlace de proyecto *(Ver figura 34).*

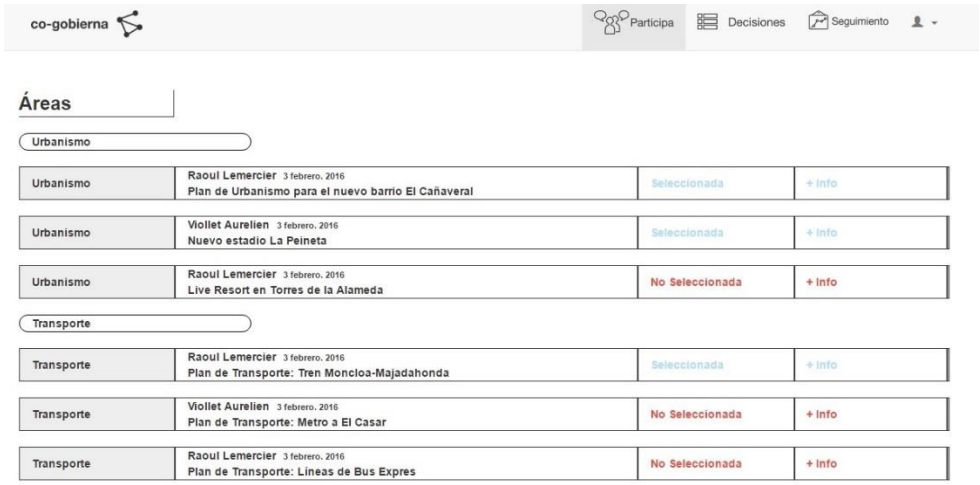

Figura 34 - Vista del listado de proyectos en Decisiones

- Vista: Información de la decisión sobre un proyecto

  Visualiza toda la información sobre el proyecto que se decidió llevarse a cabo. Asimismo, también el porcentaje de decisión que se obtuvo. *(Ver figura 35)*

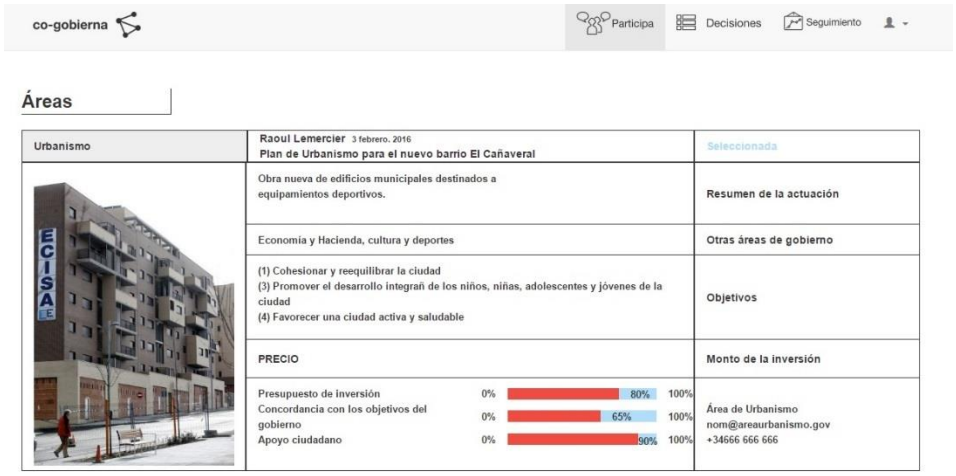

Figura 35 - Vista con la información detallada sobre la decisión en un proyecto

- Vista: Seguimiento

  En el apartado de Seguimiento, vemos organizados los proyectos por las seis áreas existentes. *(Ver figura 36)*

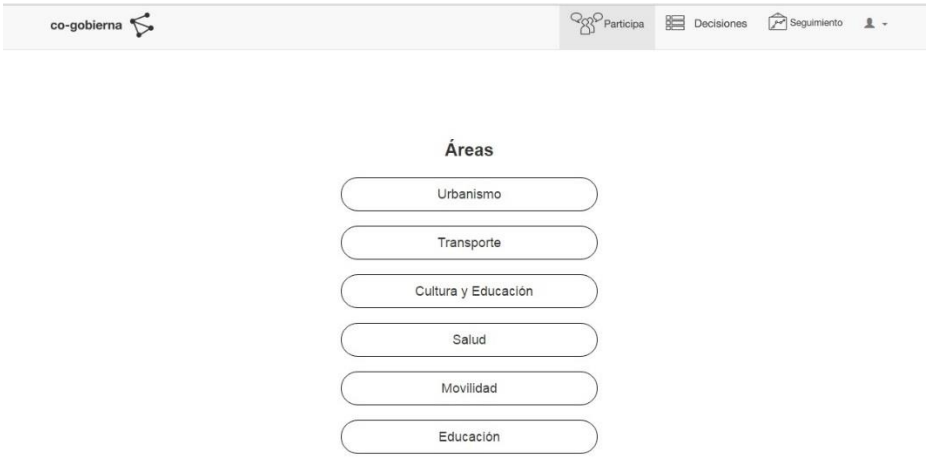

Figura 36 - Vista inicial del apartado Seguimiento

Seleccionamos por ejemplo el área de Urbanismo, y nos muestra el listado de los proyectos que hay en esta área y en qué porcentaje de desarrollo se encuentran. *(Ver figura 37)*

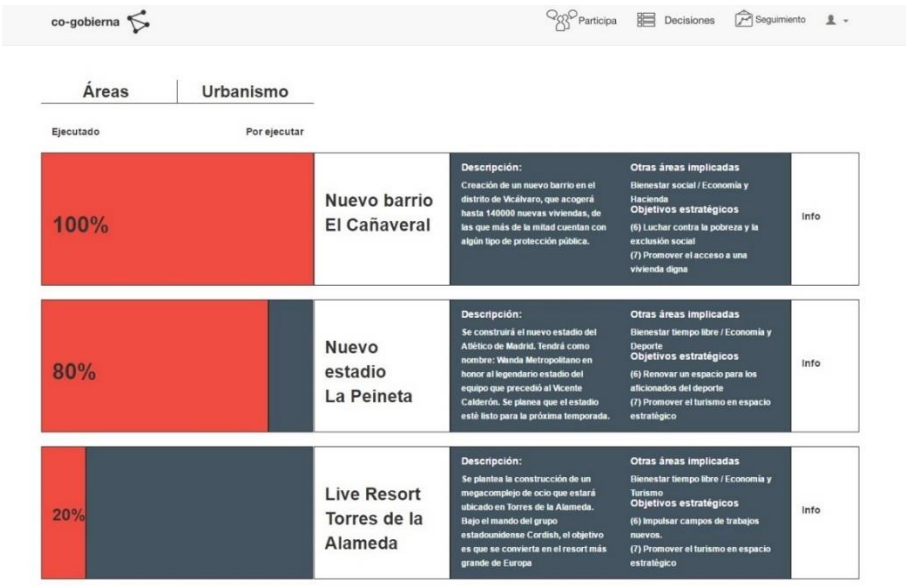

Figura 37 - Seguimiento del área de Urbanismo

- Lado Servidor

  En nuestra estructura de proyecto, tenemos la carpeta “controllers” que contendrá todos los controladores del lado servidor para la aplicación node.js. En la carpeta controllers ubicamos la carpeta “api” que contendrá todos los controladores para el web API (Application Programming Interface). Estos controladores se ubican en el fichero users.controller.js. En este fichero se definen las rutas responsables de operaciones relacionadas con el usuario, como, por ejemplo: autenticación, registro, actualización y eliminación de datos de usuario.

  Volviendo en la raíz de la carpeta “controllers” creamos 3 ficheros con código javascript: app.controller.js, login.controllers.js y register.controllers.js.
  En nuestro fichero app.controllers.js el controlador express controla todo el acceso de los ficheros clientes en la aplicación angular: usa la autenticación por sesión/cookie para mantener protegidos los ficheros de angular y expone el token JWT para ser usado por la aplicación angular.
  En el segundo fichero, login.controllers.js, se define rutas para mostrar en la vista de inicio de sesión y autenticar al usuario según sus credenciales. Para ello usa la web API, así mantiene separadas las capas de la API web con las de la base de datos. Cuando la autenticación es correcta, el token jwt que se devuelve desde la API web, se almacena en la sesión de usuario para que esté disponible en la aplicación angular cuando se carga.
  En el último script, register.controller.js, se define rutas para mostrar la vista de registro y así añadir un nuevo usuario. El funcionamiento es idéntico como en el de inicio de sesión (login.controller.js), ya que utiliza la API web para registrar nuevos usuarios con la finalidad de separar las capas de la API web con las de la base de datos.

- Servicios de usuario

  Nuevamente en nuestra estructura de proyecto, describimos el contenido de la carpeta “services”. Esta carpeta está hecha para la capa de servicios de la aplicación Node.js. Aquí es donde se encuentra toda la lógica de negocio y el acceso a los datos. Contiene el fichero “user.service.js”. En este fichero se implementa el encapsulamiento de todo el acceso a los datos y la lógica de negocio para los usuarios mediante una interfaz fácil de entender. Contiene funciones para operaciones CRUD (crear, leer, actualizar y borrar) y la autenticación de usuario.
  Se ha implementado todos los métodos de servicio mediante “promises”. En JavaScript, el objeto Promise (Promesa) es utilizado para operaciones asíncronas. Una promesa representa un valor que puede estar disponible ahora mismo, en el futuro o nunca. De esta forma, hemos mantenido coherentemente el controlador de los usuarios y con esto todos los métodos de servicio se pueden llamar con el patrón: then.

5. Uso y resultados finales.

En este apartado trataremos de mostrar el funcionamiento de la aplicación, desde el registro hasta las opciones de votación o deliberación.

- Registrarse
- Elegir área y proyecto
- Ver información proyecto y votar
- Deliberar un proyecto y argumentar a favor o en contra

- Registrarse:

  Desde la página de login de la aplicación, entramos al enlace de registrarse. Solo necesitamos el nombre, apellidos, nombre de usuario y contraseña *(Ver figura 38)*

Figura 38 - Vistas de iniciar sesión y registrarse

- Elegir área y proyecto:

  Una vez registrado y haber iniciado sesión podemos visualizar las áreas con las que se trabajará. Seleccionamos un área y se nos despliega los proyectos que esta contiene. Una vez aquí podemos navegar entre los distintos proyectos del área. Clicando en el enlace del medio del gráfico, volvemos en la visualización de todas las áreas *(Ver figura 39).*

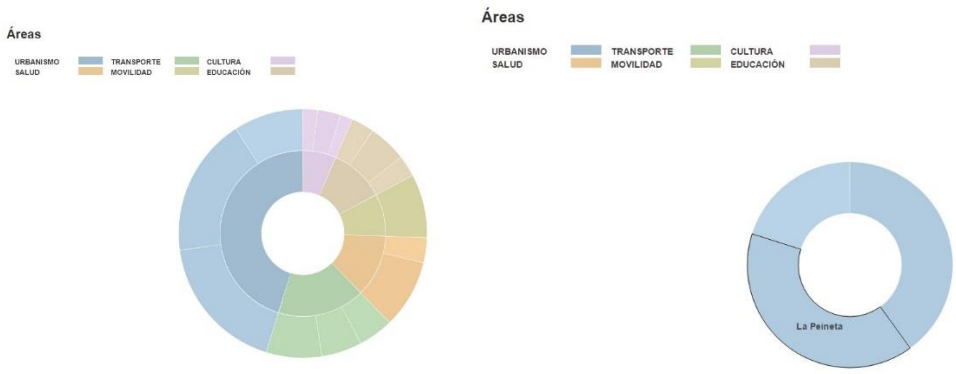

Figura 39 - Área seleccionada y proyecto

- Ver información proyecto y votar:

  Seleccionado el proyecto nos enseña la información del mismo y podemos votar viendo el porcentaje de decisiones *(Ver figura 40).*

Figura 40 - Información de proyecto y votar

- Deliberar un proyecto y argumentar en contra o favor

  En vez de votar, podemos deliberar ese proyecto, argumentado comentarios a favor o en contra de ese proyecto. *(Ver figura 41 y 42)*

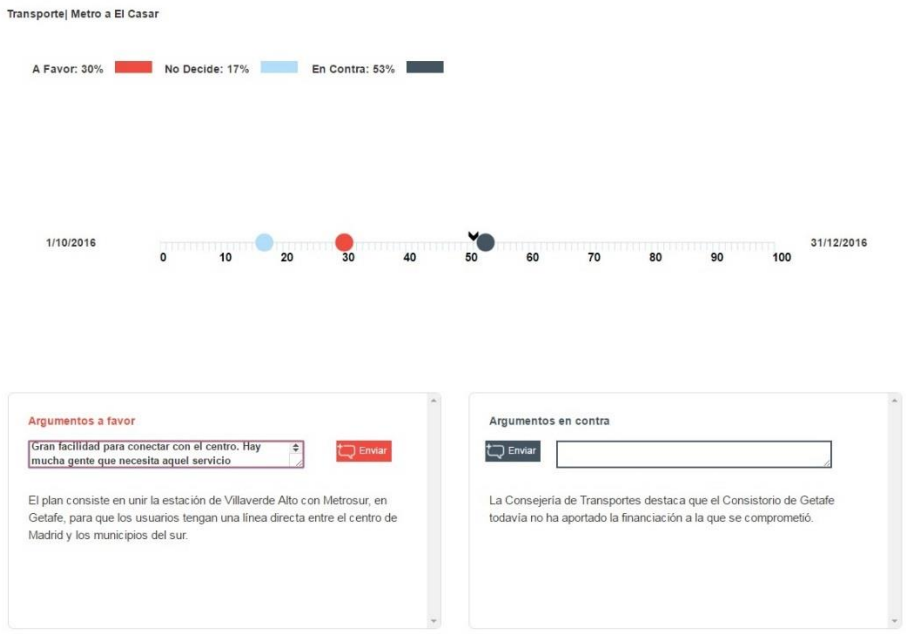

Figura 41 - Añadiendo comentario a favor

Figura 42 - Argumento a favor añadido

6. Tiempo de desarrollo del proyecto

Las etapas por las cuales ha pasado el proyecto desde su inicio han sido:

- Investigación: Es la etapa en donde se buscó información sobre las tecnologías a desarrollar
- Análisis: Etapa en donde se aplicó la información buscada: trabajo a desarrollar
- Implementación: Es el tiempo que se tomó para el desarrollo de la aplicación.
- Documentación: Etapa en la cual recogemos toda la información y requisitos para poder generar el trabajo. Es la etapa que está presente a lo largo de todo el tiempo dedicado al proyecto*. (Ver figura 43)*

Etapas	Horas
Investigación	180
Análisis	130
Desarrollo	250
Documentación	90
Total	650

Figura 43 - Tabla de etapas y horas dedicadas

6.1 Balance del coste económico del proyecto

Aquí enumeramos los costes de análisis, desarrollo y documentación. Dejamos fuera de la contabilización los costes de formación y también los costes del alojamiento web. Nuestro cliente nos ha proporcionado acceso ya. *(ver figura 44).*

Etapas	Precio por hora
Ánalisis	8
Desarrollo	7
Documentación	4

Figura 44 - Tabla de precios por etapas

En esta tabla final vemos en detalle los costes totales de cada etapa*. (Ver figura 45)*

Etapas	Horas realizadas	Precio / hora	Total
Ánalisis	130	8	1040 €
Desarrollo	250	7	1750 €
Documentación	90	4	360 €
Total	470		3150 €

Figura 45 - Tabla de precios totales

7. Conclusiones

El proyecto al iniciarse, se listó una serie de objetivos que se ha tratado de realizar y alcanzar, con orientación de muchas horas de investigación, feedback de mi tutora y apoyo del equipo de diseño. Los objetivos a realizar fueron:

a) Integrar todo el diseño hecho por el equipo, desarrollando una web que sea funcional y útil para el usuario: se logró al tener todo el material proporcionado por el equipo de diseño, además de trabajar con la herramienta Bootstrap para que sea compatible con los diferentes tamaños de ventana.

b) Poder montar una backend sencilla y compatible con nuestra frontend: Se consiguió a media parte ya que la backend fue montada, programada e implemetada con NodeJs y la base de datos MongoDB, todo usando JavaScript. Intentamos no mezclar lenguajes como PHP del lado del servidor, mas, por el lado de las votaciones fue complejo no poder visualizar los datos actualizados de cada votación sin adentrarnos en poder construir una backend más potente.

c) Separar funcionalidades de un usuario ciudadano de las de un usuario técnico: La falta de cumplimiento de este objetivo viene precedido por el anterior objetivo. Al no tener una base de datos más extensa, no pudimos crear un usuario técnico que sea el que maneje esta parte de datos. Al tratar de integrar la librería 3D.js de gráficos con AngularJS (parte front-end y tema central de nuestro proyecto), llevamos muchas horas de programación para que AngularJS funcione perfectamente.

d) Tratar en detalle las 3 características de los procesos de nuestra web: Objetivo entendido y cumplido al adentrarnos en las 3 características con AngularJS. La teoría de unos procesos participativos, vinculantes y transparentes han quedado correctos.

7.1 Actualizaciones futuras del proyecto.

Podemos enumerar algunas actualizaciones que se han quedado afuera en la realización del portal. Comencemos por las vistas de la parte técnica. Programando una parte servidor más extensa, esta tendría más facilidad en el desarrollo.
El sistema de votaciones actualizado para que los porcentajes se reflejen en tiempo real es otro punto a favor para una actualización. Tener toda la información en la parte servidor para así poder crear más proyectos de una forma más rápida.

Entendemos que es la primera versión de un portal que tiene futuro y que está recién iniciándose bajo el soporte de la Comunidad MediaLab-Prado. En este punto, muy funcional para la población de Madrid y la forma de expresarse de la ciudadanía para tener una mejor ciudad.

8. Bibliografía

[1] http://ajuntament.barcelona.cat/

[2] https://tombatossals.github.io/angularjs-tutorial/

[3] https://scotch.io/tutorials/angular-routing-using-ui-router

[4] https://desarrolloweb.com/articulos/controladores-controller-angularjs.html

[5] http://java-white-box.blogspot.com.es/2015/06/angularjs-que-es-un-servicio-que-es-un.html

[6] https://www.campusmvp.es/recursos/post/Que-es-el-stack-MEAN-y-como-escoger-el-mejor-para-ti.aspx

[7] https://www.udemy.com/curso-de-nodejs-y-angular-2-crea-webapps-con-el-mean-stack-2/

[8] http://gcoch.github.io/D3-tutorial/