Gestión de un servicio de citas municipales con AngularJS

Memoria de Proyecto Final de Máster

Máster de Ingeniería Informática

Desarrollo de aplicaciones web

Autor: Borja Fernández Ruiz

Consultor: Ignasi Lorente Puchades

Profesor responsable: César Pablo Córcoles Briongos

Enero 2016

Dedicatoria/Cita

Han sido varias las personas que me han acompañado en este viaje, como en muchos otros, desde que me propuse empezar este máster hasta hoy, que estoy cerca de su finalización.

Han pasado 3 años en los que mi vida ha ido cambiado, en muchos aspectos, pero en el fondo no ha cambiado en nada porque las personas importantes de mi vida siguen estando ahí.

Tengo que agradecer a mis padres su ayuda y apoyo interminables, desde siempre, para llegar hasta donde he llegado, en todos los aspectos de mi vida. Son mi ejemplo de cómo ser buenos padres, buenas personas y cómo con mucho esfuerzo y mucha constancia se pueden cumplir tus objetivos. Gracias por tanto y por todo, en mayúsculas.

Gracias a mi mujer, Tere, que cuando empezamos esta aventura éramos “sólo” novios. Ahora, casados y haciendo una vida en común sigue siendo un vital apoyo y la persona que me hace ser feliz cada día. Al igual que mis padres, mi espejo diario de cómo con tesón y esfuerzo se pueden conseguir tus objetivos.

Gracias a mis compañeros de trabajo, que me han animado para llevar a buen puerto este proyecto.

Gracias a los compañeros de máster con los que he tenido el placer de coincidir. Excelentes compañeros que me han hecho ayudado y con los que he colaborado a lo largo de estos años.

Y por último gracias a mis amigos, que se han interesado todo este tiempo por mis exámenes, prácticas y en particular por este proyecto. En especial a Víctor Gil y Ana Hontoria por ayudarme con sus nociones.

Abstract

This project is about an application which manages appointments for the Social Services centers in any city or village of Spain. Its realization is doing so generally as possible to be used in different contexts or activities.

In this particular case we try to solve needs in one of the Social Services centers in a specific area, which needs to have an application that allows citizens to make an appointment with the desired specialty, being able to choose the better timetable and the specialist or specialists that work in the same center they are assigned.

In this way, we can have many centers in the same area, depending on the size of the center and the neighborhoods that the area has, in which citizens are provided of different social services easily, such as: Family Care, Immigration Issues, Community Help, Seniors Care, etc…

This services, whether in chats, meetings or activities, will take place at different times each week by a specialist or instructor and it will be administrator´s responsibility to keep the appointments on the calendar and the availability of each service.

The citizens, at the same time, will have to be registered in the system to be able to make the appointment with the services they want.

Índice

1. Introducción/Prefacio

Para completar el TFM del Máster de Ingeniería Informática el área de Desarrollo de aplicaciones web era uno de las áreas que más me atraían. Al final, la vorágine del día a día, tanto laboral como personal, no siempre nos permite dedicar el tiempo preciso a aprender todos los nuevos paradigmas y nuevas herramientas que aparecen y se consolidan en este sector de la informática.

Este Máster está siendo una buena plataforma para estar al día y tomar contacto con algunos de los aspectos que hoy en día y de cara al futuro serán prácticos y muy útiles para intentar estar lo más actualizado posible, en un sector que no para ni un segundo de crecer y evolucionar.

Por tanto, este TFM se presenta como una oportunidad única de seguir indagando nuevos frameworks, nuevas ideas y nuevas técnicas que se están imponiendo en las aplicaciones web poco a poco. Aunque sólo fuera a modo de estudio nunca está de más seguir aprendiendo.

Para poner en práctica todos los conocimientos que estoy aprendiendo y reforzando en este Máster detecto algunas necesidades como ciudadano, para solicitar algunos servicios municipales de manera online. Lógicamente las Administraciones no pueden dar cabida a todo y es complejo tener todos tus servicios de manera telemática pero, como veremos en este documento, sí existen algunas carencias que siempre se pueden mejorar. Para aportar valor añadido surge este proyecto.

2. Descripción/Definición/Hipótesis

El proyecto consiste en llevar a cabo una aplicación de gestión de citas para los centros de Servicios Sociales de un municipio de la geografía española. Su realización se lleva a cabo de manera lo más genérica posible para que pueda ser extrapolable a diferentes contextos e incluso actividades.

En el caso concreto que nos ocupa se pretende dar solución a las necesidades de los centros de Servicios Sociales de un municipio, que necesita contar con una aplicación web que permita a los ciudadanos solicitar cita en la especialidad deseada, en el horario disponible que sea de su agrado y con el especialista o especialistas que trabajen en el centro concreto al que están asignados.

De esta forma, dentro de un municipio es factible contar con diferentes centros, en función del tamaño del mismo y los barrios que tenga, en el que se faciliten a los ciudadanos diferentes servicios sociales, a saber: Atención a Familias, Sensibilización sobre Inmigración, Ayuda Comunitaria, Ayudas a personas mayores, etc.

Estos servicios, en forma de charlas, reuniones o actividades, se darán en diferentes horarios cada semana por un especialista o instructor y será responsabilidad de los administradores mantener correctamente el calendario y la disponibilidad de cada servicio.

El ciudadano, a su vez, deberá estar registrado en el sistema para poder solicitar las citas con los servicios que desee.

3. Objetivos

En este apartado y de manera inicial, se enumeran en un listado los objetivos del proyecto, ordenados por orden de relevancia. En primer lugar se encuentran los objetivos principales, que son objetivos clave que se han de cumplir para validar el proyecto y en un segundo punto del apartado los objetivos adicionales que le dan un valor añadido al proyecto y que pueden sufrir variaciones.

Principales

- El acceso al sistema debe ser mediante la autenticación del usuario, es decir, deberá estar registrado en el mismo.
- Una vez accedido al sistema, el usuario deberá poder consultar los horarios disponibles en cada semana (de la que exista información) para cada actividad.
- En cada horario disponible se visualizará el especialista/trabajador que tutela la actividad en cuestión.
- Se utilizarán mensajes de confirmación que mejoren la usabilidad del sistema y la experiencia del usuario con la aplicación.
- El usuario podrá consultar el listado de citas reservadas.
- El usuario podrá cancelar citas ya reservadas previamente.
- La información de los horarios disponibles será actualizada por los administradores del sistema a través de la aplicación web.
- Se utilizarán herramientas de software libre para el desarrollo y el despliegue del sistema.
- El sistema respetará los principales estándares de accesibilidad web.
- El cliente deberá utilizar un navegador web para acceder al sistema.
- Los datos de los usuarios deberán registrarse en la Agencia de Protección de Datos, garantizando los derechos que exige la Ley Orgánica de Protección de Datos - LOPD - (objetivo no aplicable en el presente ámbito del proyecto).

Secundarios

- Las categorías de las actividades serán actualizadas por los administradores del sistema a través de la aplicación web.
- Serán los usuarios administradores los únicos que podrán dar de alta nuevos usuarios asignándoles sus perfiles correspondientes.
- Se incluirán filtros de las citas en función de los especialistas trabajadores de cada centro.
- Se pretende ofrecer una solución multidispositivo en la medida de lo posible.
- A nivel técnico se pretenden utilizar arquitectura web basada en lenguajes de programación, frameworks y APIs recientes, tales como node.js, angularJS o similares.

4. Marco teórico/Escenario

En el siguiente apartado se describe el marco teórico y el contexto en el que se encuadra el proyecto.

- Antecedentes y motivaciones que dan lugar al proyecto

El presente proyecto surge con el objetivo de aplicar los conocimientos adquiridos a lo largo de estos años en el Máster de Ingeniería Informática, focalizando en el área de Desarrollo de aplicaciones web.

Aunque ya se entrará en detalle en el apartado de Estado del Arte, este área de Desarrollo de aplicaciones web se encuentra en un momento de gran equilibrio entre el rendimiento actual de las aplicaciones web pero especialmente el crecimiento y el futuro que tienen gracias a la variedad de tecnologías que están aportando valor al sector. Tecnologías que ofrecen mayor rendimiento, facilidades de desarrollo y un sinfín de aspectos positivos que hacen de este área de conocimiento, al menos desde mi punto de vista, en uno de los más atractivos, gracias a estos aspectos y sobre todo al enorme uso que se hace hoy en día de ellas.

Conocedor, como ciudadano y por motivos laborales, de las necesidades tecnológicas de algunos municipios de la Comunidad de Madrid, en España, para gestionar las citas de algunos de los servicios municipales, decido asumir el reto, gracias al contexto que me ofrece este Trabajo Fin de Máster, de plantear una solución tecnológica básica enfocada a las actividades de los Servicios Sociales de algunos de estos municipios anteriormente nombrados.

- Estado del arte / Escenario

La arquitectura de aplicación web podríamos decir que hoy en día es un valor seguro, conocido, estudiado, de moda y sobre todo muy usado para aportar soluciones web. Lejos de estancarse es un área que se afianza y crece, mejorando la arquitectura con nuevas tecnologías, productos y sistemas que ofrecen mejor rendimiento, mayor capacidad de almacenamiento, mayores facilidades de desarrollo, implementación y despliegue y recorte en los gastos económicos que suponen desarrollar una aplicación.

Sin remontarnos en detalle a las primeras arquitecturas y el primer concepto de mainframe del siglo pasado, éstas tenían un diseño de instalación bastante sencillo: un ordenador central voluminoso y potente que lo era todo, y al que se conectaban los usuarios mediantes terminales denominados actualmente como "terminales tontos" que no eran más que un teclado y una pantalla.

Estos sistemas fueron evolucionando gracias a la popularización de los PCs, su bajada de precio, los nuevos diseños e ideas pasando a tener un servidor central pero que repartía los esfuerzos y las tareas entre los equipos conectados a él.

Llegamos así a la arquitectura cliente-servidor. El servidor central ya no tenía por qué ser tan potente, ni tan caro pues los terminales conectados a él tenían capacidad de procesamiento y almacenamiento para realizar tareas y repartirse el esfuerzo. A nivel empresarial, con esta arquitectura, una empresa necesita un servidor (no mucho más caro que un PC normal) que centraliza la información y aporta mayor seguridad y velocidad que las arquitecturas más antiguas.

No obstante, este modelo habitualmente precisa instalar en los equipos clientes un programa (un paquete de instalación) para acceder a los datos y aplicaciones del servidor (cliente pesado).

En este sentido surge el modelo y el auge de la arquitectura de las aplicaciones web, en la que la primera ventaja es que ya no es necesario instalar y mantener todos los equipos conectados. En el presente apartado se detallarán otras ventajas que nos sitúan en el estado del arte actual pero, a nivel objetivo, sí es conveniente no tratar la arquitectura cliente-servidor como obsoleta o poco útil pues para determinados usos en la actualidad se sigue utilizando y puede aportar un buen rendimiento.

No obstante, la arquitectura de aplicaciones web se ha impuesto gracias a todas las ventajas que tiene y al espléndido futuro que se le espera con las nuevas tecnologías que van apareciendo y estabilizándose.

Entre sus principales ventajas destaca su facilidad de uso para los usuarios, pues sólo necesitan un navegador para acceder a la aplicación. Eso sí, es responsabilidad del desarrollador implementar una aplicación accesible, compatible y usable para todos los usuarios. Por tanto, una de las ventajas es su facilidad de uso.

Otra ventaja es el ahorro de coste de hardware y software. La proliferación de las herramientas OpenSource hacen que la implementación y despliegue sea más barato. Además los usuarios clientes no precisan un hardware especialmente potente y complejo. Esto deriva en aplicaciones con menos errores, menor probabilidad de cuelgues y más fáciles de solucionar en caso de fallos.

La accesibilidad de las aplicaciones web son otra de sus ventajas, actualmente los desarrollos se realizan para multidispositivos por lo que puedes conectarte desde un PC, una tablet o un móvil desde cualquier lugar del mundo. En este sentido el marco colaborativo también se ve beneficiado.

La escalabilidad de las aplicaciones web es mucho mayor también. Sólo es necesario actualizar el servidor y los procesos de actualización suelen ser más rápidos, limpios y seguros.

Gracias a las tecnologías actuales los datos están muy seguros gracias a los servicios redundantes y de backup de las empresas proveedoras de hosting y almacenamiento.

Además, desde hace unos años han surgido nuevas tecnologías y conceptos como Cloud Computing, el Internet de las Cosas (Internet of Things) o Green Computing que

están íntimamente relacionados con las aplicaciones web y se presentan como el presente y el futuro de éstas.

Son tecnologías que quizá excedan el alcance del presente proyecto pero, al igual que otras, es importante tenerlas en cuenta por las buenas ventajas que ofrece. Mejoras del rendimiento de las aplicaciones, ahorro de costes, mayor seguridad y replicación de la información, optimización de recursos, de memoria, minimización del impacto ambiental y mayor inteligencia y accesibilidad son algunas de las características de estas técnicas.

Por último, a nivel de desarrollo son varias las tecnologías que no paran de mejorar, crecer y ofrecer mayores ventajas a la hora de implementar y desplegar una aplicación web. Tecnologías clásicas como HTML, CSS o JavaScript sufren actualizaciones y mejoras que se traducen que una mayor accesibilidad, usabilidad, ubicuidad y mejor experiencia para el usuario. Nacen nuevos frameworks y entornos de desarrollo como node.js, AngularJS, EmbedJS y nuevas tendencias e ideas en bases de datos NoSQL como MongoDB que hacen de este sector de las aplicaciones web un área en continuo crecimiento, mejora y expansión.

- Otros proyectos/estudios/productos similares o relacionados

Este proyecto, como se indicó anteriormente, surge de las necesidades detectadas en el ámbito de la gestión de citas de determinados servicios sociales en municipios (Administraciones Locales) de la geografía española, concretamente de la Comunidad de Madrid.
Desde la publicación y puesta en marcha de la Ley 11/2007, de 22 de junio, de acceso electrónico de los ciudadanos a los Servicios Públicos [1] se reconoce a los ciudadanos su derecho a relacionarse electrónicamente con las administraciones públicas, así como la obligación de éstas a garantizar ese derecho.

De esta forma, desde las Administraciones Públicas se pretende impulsar el uso de los servicios electrónicos en la Administración, pues tienen la obligación de posibilitar el acceso a todos sus servicios electrónicos, incluyendo registros, pagos, notificaciones, consultas de tramitación de sus procedimientos, etc.

Este trabajo de adaptación a la nueva norma conlleva un importante estudio de análisis e implementación de herramientas que permitan el cumplimiento de la misma. No obstante, esta adaptación, en función de las distintas Administraciones se está realizando de manera paulatina.

En general, las Administraciones (y en particular por el caso que nos ocupa, las Administraciones Locales o Ayuntamientos) están resolviendo esta disyuntiva con la puesta en marcha de lo que la ley denomina "Sede Electrónica" (o Portal del Ciudadano) en la que se engloban e incluyen diferentes servicios electrónicos. Estos sistemas tratan cada servicio como expedientes, por lo que a efectos prácticos y en general, funcionan como tramitadores de expedientes.

Este tipo de sistemas tienen la ventaja de ofrecer una solución muy válida de manera genérica que es cómoda para los usuarios (por ser la manera de tramitar las solicitudes muy parecida) y también muy cómoda para los trabajadores y administradores de la Administración, pues en realidad, sea cual sea la solicitud no deja de ser un trámite de un aspecto concreto.

No obstante, esta solución, adoptada por Administraciones de mayor enjundia (Ministerios, Ayuntamientos españoles importantes, la Agencia Tributaria, la DGT, etc.) hasta de "menor" calado (Juntas de Comunidades Autónomas, Diputaciones Forales o Ayuntamientos de municipios con menor número de habitantes) no siempre ofrecen todos los servicios y además no siempre lo hacen de una manera particular y específica con el servicio en cuestión, lo que redunda en servicios incompletos o demasiado estandarizados.

Para mejorar este aspecto nace la idea de este proyecto, con el objetivo de abordar de una manera más específica y particular un servicio de gestión de citas, que pueda ser extrapolable a diferentes servicios de una Administración concreta.

5. Arquitectura de la aplicación

En este apartado se describe la arquitectura de la aplicación. Cabe destacar que la principal tecnología utilizada en la aplicación es la llamada AngularFire [2].

AngularFire es el nombre utilizado para el uso de AngularJS y Firebase, de los que se entrará en detalle en el apartado Plataforma de desarrollo del presente documento. AngularFire es una librería de código abierto, soportado por el equipo de Firebase y su comunidad de desarrolladores. Angular ha sido definido como un framework (desarrollado por Google) Model-View-Whatever, es decir, Modelo-Vista-Lo-que-sea. Pretende de esta forma huir de la discusión entre MVVM vs MVC vs MVP, pero está enmarcado como un estilo MVVM (Modelo Vista VistaModelo), o como se puede encontrar en algunos ejemplos como un TodoMVC [3].

Angular hace que las aplicaciones se organicen en torno a Controllers que construyen los ViewModels que se enlazarán a las vistas. Estos ViewModels se enlazan con las vistas, es decir, los modelos Javascript y el DOM, mediante un sistema de databinding declarativo bidireccional.
A este modelo se une Firebase, que como se verá más adelante es una API para almacenar y sincronizar datos en tiempo real, haciendo también la función de base de datos de tipo NoSQL. Contiene "bindings" para la mayoría de frameworks, de ahí la unión con Angular para formar AngularFire. De esta forma, se consigue que los datos de la aplicación se sincronicen en tiempo real con todos los clientes, es decir, si cambian los datos de un usuario o cliente, los cambios persisten en Firebase y se sincronizan y actualizan con el resto de usuarios inmediatamente.
El siguiente diagrama se puede encontrar en el blog de Firebase [4], y explica lo mencionado anteriormente.

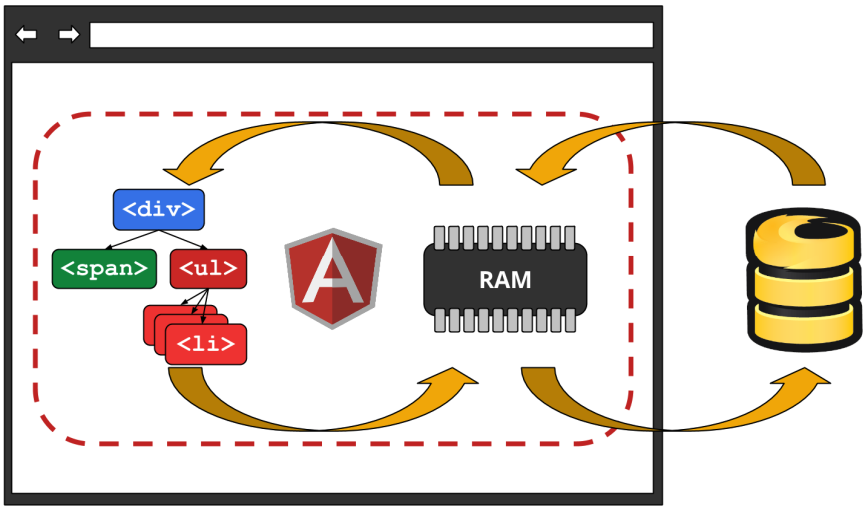


Ilustración 1. El modelo de Angular + Firebase: Three-Way Data Binding

Como se puede observar, según el diagrama cuando los datos se actualiza en la Vista, el Modelo, o en Firebase (cualquiera de los tres), los cambios se propagan en tiempo real entre todos ellos.
Esto hace de este modelo una solución realmente interesante.

6. Plataforma de desarrollo

En este apartado se describen las tecnologías utilizadas para llevar a cabo esta aplicación. Tal y como se detalla en el punto anterior de Arquitectura de la aplicación del presente documento, la base tecnológica de la aplicación es el llamado AngularFire, código abierto mantenido por Firebase que integra AngularJS con él.

Por tanto, como principales tecnologías en este proyecto se podría destacar AngularJS como framework de desarrollo (basado en el lenguaje de programación Javascript, tal y como veremos a continuación) y Firebase como servicio de base de datos noSQL. A su vez, se han utilizado Bower, Grunt, Yeoman, npm-Node.js y Bootstrap para el apartado visual de la interfaz (CSS), sin destacar ningún elemento hardware que merezca ser mencionado por su importancia.

AngularJS

AngularJS es un framework MVC de JavaScript para el Desarrollo Web Front End que permite crear aplicaciones SPA (*Single-Page Applications*).

Actualmente es uno de los frameworks más de moda. Aunque su primera versión es de 2009, desde 2012 está en pleno auge, consolidándose año tras año. La tendencia ahora es hablar de una nueva tecnología que ha adoptado el nombre de MEAN (MongoDB/Mongoose + ExpressJS + AngularJS + NodeJS), lo que también se traduce a aplicaciones Javascrip end-to-end.

Una de las ventajas y motivos que han hecho que AngularJS se haya consolidado es que está mantenido por Google y por bastante comunidad en la red. Es relativamente sencillo encontrar documentación, vídeos y ejemplos de calidad. Otro punto a favor es que AngularJS le da especial importancia a la realización de tests unitarios, disponiendo para ello funcionalidades concretas.

AngularJS está construido en torno a la creencia de que la programación declarativa es la que debe utilizarse para generar interfaces de usuario y enlazar componentes de software, mientras que la programación imperativa es excelente para expresar la lógica de negocio. Este framework adapta y amplía el HTML tradicional para servir mejor contenido dinámico a través de un 'data-binding' bidireccional ("Two Way Data Binding") que permite la sincronización automática de modelos y vistas, es decir, permite que cualquier cambio en el modelo se refleje en la vista y viceversa de manera automática. Como resultado, AngularJS pone menos énfasis en la manipulación del DOM y mejora la testeabilidad y el rendimiento.

Además, con el uso de la inyección de dependencias, Angular lleva servicios tradicionales del lado del servidor, tales como controladores dependientes de la vista,

a las aplicaciones web del lado del cliente, es decir, al crear un módulo, una fábrica o cualquier otro componente AngularJS, tan sólo basta indicarle las dependencias que necesita y el framework se encarga de proporcionarlo. En consecuencia, gran parte de la carga en el backend se reduce, lo que conlleva a aplicaciones web mucho más ligeras.

Son muchas otras las características que definen a AngularJS, así como el patrón Modelo-Vista-VistaModelo en el que se le engloba, tal y como se ha indicado en el apartado anterior. Todas estas virtudes hacen que se haya extendido rápidamente y aunque también existen voces críticas con este framework [5], se le considera como una de las mejores opciones para procesamiento JavaScript [6].

Firebase

Firebase se presenta como una poderosa API para almacenar y sincronizar datos en tiempo real, una base de datos de tipo NoSQL. Pero Firebase es un servicio diferente a una simple base de datos NoSQL, un servicio dirigido tanto a aplicaciones web como aplicaciones móviles, el cual permite realizar, además del almacenamiento de datos, una sincronización en directo de los mismos en cada uno de los dispositivos que se encuentran conectados a la aplicación. Además, permite el funcionamiento offline y una posterior sincronización de los datos cuando el dispositivo vuelve a tener conexión a internet.

Además, cuenta con librerías para la mayoría de las plataformas web y móviles, y 'bindings' para la mayoría de frameworks, entre los que se encuentra AngularJS, haciendo de Firebase un candidato ideal para este proyecto.

En cuanto a la seguridad de los datos, Firebase requiere encriptado SSL 2048-bit para toda la transferencia de datos.

Firebase se puede utilizar a través de su página web, registrando una cuenta (se puede vincular con tu cuenta de gmail de Google por ejemplo) se puede acceder al servicio y crear nuevas bases de datos, pudiendo utilizar todas sus herramientas.

Como ya se ha indicado, Firebase es una base de datos NoSQL [7], un concepto de base de datos diferente a las típicas bases de datos relacionales. Las bases de datos NoSQL surgen por los problemas de escalabilidad detectados con la llegada del software como servicio, los servicios en la nube y las startups de éxito con millones de usuarios.

Si bien los modelos relacionales se pueden adaptar para hacerlos escalar incluso en los entornos más difíciles, sí que es cierto que, a menudo, se hacen cada vez menos intuitivos a medida que aumenta la complejidad. No es extraño por tanto el uso de por ejemplo triples y cuádruples JOINs en consultas SQL y el uso de cachés para acelerar peticiones y evitar ejecutar consultas pesadas.

Los sistemas NoSQL intentan atacar este problema proponiendo una estructura de almacenamiento más versátil, aunque sea a costa de perder ciertas funcionalidades como las transacciones que engloban operaciones en más de una colección de datos, o la incapacidad de ejecutar el producto cartesiano de dos tablas (también llamado JOIN) teniendo que recurrir a la desnormalización de datos [8].

Algunas implementaciones conocidas de estas bases de datos NoSQL (y que se han visto en algunas asignaturas de este máster) son MongoDB, Cassandra, BigTable, Dynamo y Hadoop, entre muchas otras.

Las principales diferencias de estas bases de datos con las relacionales son la ausencia de esquema en los registros de datos, escalabilidad horizontal sencilla (frente a la vertical de las relacionales) y mayor velocidad de procesamiento siempre y cuando el sistema sea estable y maduro.

La ausencia de esquema en los registros de datos hace que cada registro pueda contener una información con diferente forma cada vez y almacenar estructuras de datos complejas en un único registro, mejorando la claridad (al tener todos los datos relacionados en un mismo bloque) y el rendimiento (no hay que hacer JOINs).

A su vez, en cuanto a escalabilidad horizontal, se puede aumentar el rendimiento del sistema simplemente añadiendo más nodos.

En cuanto a la velocidad de procesamiento, estos sistemas NoSQL suelen realizar operaciones en memoria directamente, y sólo vuelvan los datos a disco cada cierto tiempo. Esto permite que las operaciones de escritura sean muy rápidas aunque también conlleva riesgos de durabilidad en los datos y de posible pérdida de consistencia en los datos en caso de fallo en el sistema.

Yeoman, Grunt, Bower y npm

Tal y como se indica en su sitio oficial [9] Yeoman es una herramienta o plugin que permite generar la estructura o esqueleto de aplicaciones completas o partes útiles de la misma. Para ello cuenta con multitud de generadores oficiales y estables que permiten al desarrollador crear el esqueleto de una aplicación web rápidamente. De esta forma mediante Yeoman se puede crear la aplicación para empezar a trabajar desde cero evitando problemas de configuración y despliegue.

Para mejorar la productividad y satisfacción creando la aplicación web, Yeoman se apoya en diferentes herramientas o utilidades. Para el presente proyecto se han utilizado las siguientes: la herramienta que genera el esqueleto (yo), la herramienta de construcción como tal (Grunt) y los gestores de paquetes (Bower y npm).

Grunt.js es una librería JavaScript que nos permite configurar tareas automáticas y así ahorrar tiempo en el desarrollo y despliegue de aplicaciones webs. Para el caso de este proyecto hace también la función de servidor que compila y despliega los cambios sin necesidad de tener que reiniciarlo.

Con un simple fichero JS que llamaremos Gruntfile, indicamos las tareas que queremos automatizar con un simple comando y las escribimos en él en formato JSON. En posteriores secciones se citará como desplegar la aplicación con Grunt y el fichero Gruntfile con algunas tareas automatizadas por defecto al generar Yeoman.

Bower y npm son manejadores de paquetes para el front-end de tu aplicación, es decir, tienen una colección de herramientas software que automatizan y facilitan los procesos de instalación, actualización, configuración y eliminación de programas y servicios para una aplicación web. De hecho a Bower en un entorno tecnológico como el que se presenta en este proyecto se le conoce como un gestor de dependencias de la aplicación, que gestiona los componentes front-end como los html, css, etc. Bower realmente depende de npm y estos a su vez de Node.js.

Bootstrap

Es uno de los frameworks CSS más famosos y uno de los más utilizados, en la actualidad muchas de los sitios y aplicaciones web que se crean se apoyan en un diseño creado con Bootstrap [10].

Bootstrap se diseñó inicialmente por Twitter. Permite dar forma a nuestra aplicación web mediante librerías CSS que incluyen botones, tipografías, cuadros, menús, etc. Fue liberado bajo licencia MIT en el año 2011 y su desarrollo continúa en un repositorio de GitHub [11].

Actualmente Bootstrap está implementado en su versión 3 (aunque la versión 4 se encuentra en fase alpha) [12].

Los puntos fuertes de Bootstrap son la gran cantidad y calidad de sus librerías, con increíbles jQuery plugins, decenas de componentes HTML y CSS totalmente personalizables y una documentación excelente que hace que la experiencia de uso por parte de diseñadores y desarrolladores sea muy enriquecedora.

A su vez, el otro gran punto fuerte de Bootstrap es que gracias a su uso se pueden conseguir sitios web y aplicaciones web totalmente adaptables a todo tipo de pantallas y dispositivos (diseño responsive). En este sentido son varias las características que se han reforzado para hacer compatible al máximo ese diseño responsive con navegadores web. Destaca que casi al completo es compatible con HTML5 y CSS3, el uso de GRID de 12 columnas donde incluir el contenido y la posibilidad de incluir imágenes responsive con la clase ‘img-responsive’.

Por último cabe destacar que Bootstrap es compatible con la mayoría de navegadores web del mercado, tales como Google Chrome, Mozilla Firefox, Opera, Safari e Internet Explorer.

¿Por qué se han elegido todas estas plataformas y tecnologías?

El denominador común de estas tecnologías es la evolución que están teniendo las aplicaciones web y los sistemas multidispositivo en los últimos años. En un contexto en continuo crecimiento, la búsqueda de nuevas soluciones más ágiles, más dinámicas, sencillas de desarrollar y compatibles con navegadores y diferentes dispositivos hacen que surjan nuevas tecnologías que puedan desarrollar dichas soluciones.

Para el presente proyecto la idea es utilizar algunas de estas nuevas tecnologías que se han visto de manera no muy detallada en el máster. Por tanto, el objetivo, dado el tiempo con el que se cuenta, es realizar una aplicación web lo más completa posible utilizando estas tecnologías recientes.

AngularJS al ser un framework tan conocido y tan utilizado es un candidato perfecto. Es mucha la documentación existente y aunque la curva de aprendizaje se pueda considerar compleja, especialmente al principio, se ha considerado de manera personal por el alumno como la herramienta perfecta para realizar este proyecto. Se utiliza para este proyecto la última versión estable de AngularJS, es decir, la versión 1.4.8 (la 1.5.0 se encuentra en fase beta).

A su vez, con el objetivo de aprender y seguir la tendencia de tecnologías recientes, el uso de bases de datos NoSQL también resulta de interés y más contando con un servicio sencillo como es Firebase que mediante la creación de una cuenta en el sistema permite contar con una base de datos NoSQL accediendo fácilmente a su web. Para este proyecto se utiliza la versión 0.9 de AngularFire (Angular + Firebase). Aunque en la actualidad dicha versión se encuentra en la 1.1.3 (la página oficial de Firebase cuenta con documentación referencial de la API) no existen todavía demasiados ejemplos ni mucha experiencia con esta versión, pues es relativamente reciente, de hace unos meses. En este sentido se decide empezar por una versión estable y muy madura.

Por último, Yeoman se utiliza gracias a la investigación realizada para la puesta en marcha del proyecto. Son varias las bondades con las que cuenta para crear una aplicación web AngularJS desde cero por lo que es otro candidato perfecto para el contexto de este proyecto. Yeoman, tal y como se ha indicado anteriormente, precisa Grunt, Bower y npm de ahí su uso.

En definitiva, se han utilizado tecnologías recientes por una idea de concepto desde el principio, porque se utilizan mucho en la actualidad (con mucho éxito en proyectos de envergadura similar al actual) y por enriquecimiento personal del alumno.

7. Planificación temporal

En este apartado se describe, de una manera generalista y a alto nivel, un esbozo de planificación temporal para el proyecto. Se toma como fecha de inicio el 16 de septiembre de 2015 (fecha de comienzo del TFM). El calendario base del proyecto se ha personalizado a la disponibilidad del alumno, en función de sus circunstancias laborales y personales, de manera que éste trabajará en el proyecto 6 días a la semana, a razón de 4 horas cada día. El horario será de lunes a viernes de 18:00 a 22:00 y sábados de 10:00 a 14:00. No se han contemplado posibles excepciones pero naturalmente podrán producirse. Como planificación que es, podría verse modificada en el transcurso del proyecto.

Cada fase tiene un nombre general con unas tareas asociadas que también se conciben de manera general, no tendrán por qué ser las que se produzcan exactamente en el proyecto.

Como por motivos académicos el TFM se entregará en enero, la planificación se estructura para que se termine a principios de enero de 2016.

En la primera fase (Estudio de Viabilidad), intentando adecuarla al calendario de la PEC1, se puede observar que se dilata en el tiempo varios días. A partir de la segunda fase (Análisis del Sistema) el proyecto planifica su rumbo natural.

Teniendo en cuenta la excepción anterior, la fase de mayor duración es la fase de Implementación. Las fases de Análisis y Diseño del Sistema juntas supondrían más de la mitad de la fase de Implementación. Se ha de tener en cuenta que las necesidades y los requisitos que tiene el sistema ya estaban detectados con anterioridad por el alumno por lo que el esfuerzo no será tan grande en estos aspectos.

A continuación se describe el listado de fases y tareas que forman la planificación temporal del proyecto. Se ha utilizado una base del tiempo continua, es decir, las fases y las tareas no se solapan aunque es lógico pensar que en un ciclo de vida real del proyecto esto no siempre es así y en ocasiones es necesario volver a modificar tareas que ya estaban completadas.

El proyecto finalizaría el miércoles 6 de enero de 2016. Como la entrega se produciría a finales de mes, se puede contar con días del mes de enero para posibles contingencias en forma de modificaciones, actualizaciones, etc.

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		**Estudio de viabilidad**	**15 días**	**mié 16/09/15**	**vie 02/10/15**	
2		Definir el proyecto y establecer el alcance	1 día	mié 16/09/15	mié 16/09/15	
3		Estudio de la situación actual	1 día	mié 30/09/15	mié 30/09/15	2
4		Definición de requisitos/objetivos	1 día	jue 01/10/15	jue 01/10/15	3
5		Estudio y valoración de estudios o proyectos similares	1 día	vie 02/10/15	vie 02/10/15	4
6		**Análisis del Sistema**	**16 días**	**lun 05/10/15**	**jue 22/10/15**	**5**
7		Definición del sistema	6 días	lun 05/10/15	sáb 10/10/15	
8		Establecimiento de requisitos	4 días	lun 12/10/15	jue 15/10/15	7
9		Definición de interfaces de usuario	4 días	vie 16/10/15	mar 20/10/15	8
10		Especificación del plan de pruebas	2 días	mié 21/10/15	jue 22/10/15	9
11		**Diseño del Sistema**	**9 días**	**vie 23/10/15**	**lun 02/11/15**	
12		Especificación de estándares, normas de diseño y construcción	3 días	vie 23/10/15	lun 26/10/15	
13		Elección de infraestructuras y herramientas de desarrollo	2 días	mar 27/10/15	mié 28/10/15	12
14		Especificaciones de desarrollo	2 días	jue 29/10/15	vie 30/10/15	13
15		Requisitos de implantanción	2 días	sáb 31/10/15	lun 02/11/15	14
16		**Implementación**	**49 días**	**mar 03/11/15**	**mar 29/12/15**	
17		Planificación de las actividades de desarrollo e integración de sistema	7 días	mar 03/11/15	mar 10/11/15	
18		Desarrollo e implementación	30 días	mié 11/11/15	mar 15/12/15	17
19		Documentación	12 días	mié 16/12/15	mar 29/12/15	18
20		**Implantación**	**7 días**	**mié 30/12/15**	**mié 06/01/16**	
21		Implantación del sistema	1 día	mié 30/12/15	mié 30/12/15	
22		Pruebas	3 días	jue 31/12/15	sáb 02/01/16	21
23		Aceptación del sistema	1 día	lun 04/01/16	lun 04/01/16	22
24		Defensa del proyecto	2 días	mar 05/01/16	mié 06/01/16	23

Ilustración 2.Planificación temporal del proyecto

Con el diagrama de Gantt se puede comprobar la planificación espacio temporal.

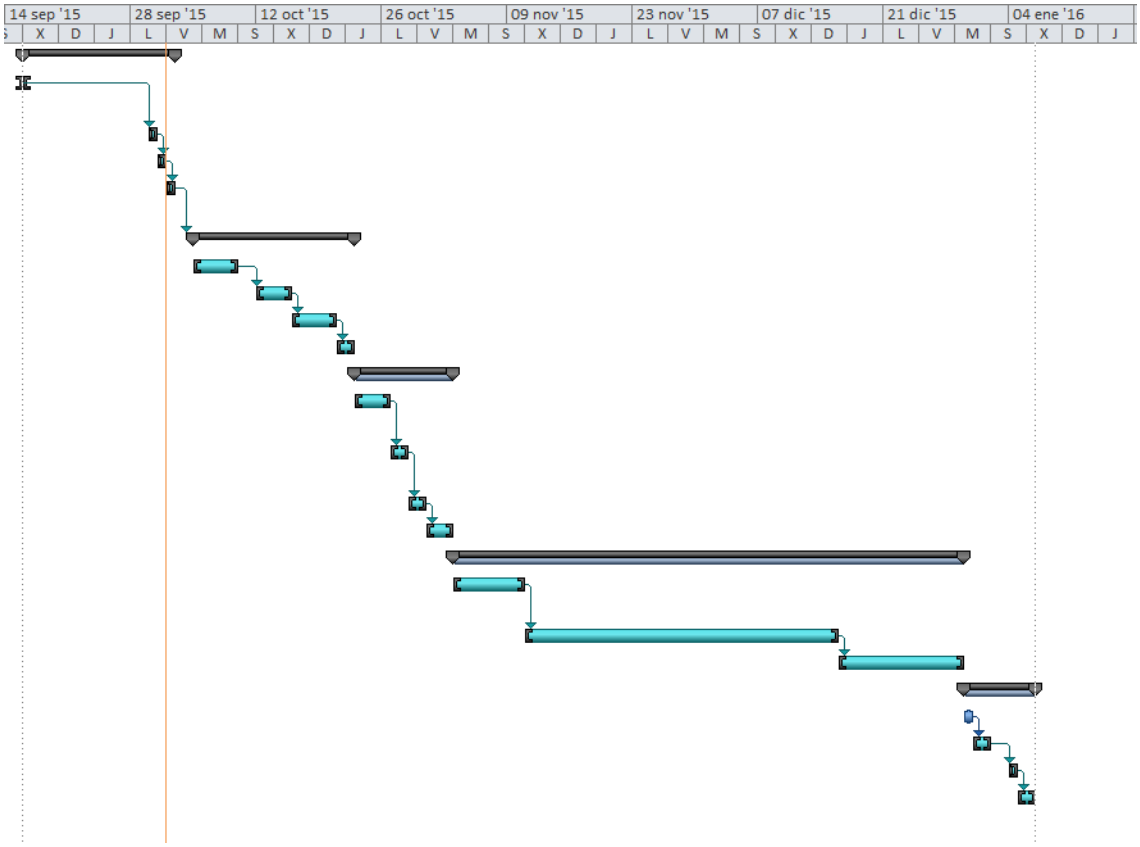


Ilustración 3. Diagrama de Gantt

8. Planificación económica

En este apartado se desglosa en términos económicos la planificación temporal descrita en el apartado anterior. Se ha de tener en cuenta que no es más que una planificación económica inicial, a efectos de previsiones, pero lógicamente está sujeta a cambios. Se toma como precio estimación el de 15 euros/hora, teniendo en cuenta un único recurso humano que soy el alumno que está llevando a cabo el proyecto.

De momento no se puede estimar el coste de hardware y software aunque se pretende utilizar herramientas de software libre que lógicamente ahorraría el coste final del producto.

No se tiene en cuenta gastos materiales de oficina, luz, Internet, etc., porque el proyecto se realizará a cuenta del alumno.

El coste por fases será el siguiente:

Estudio de Viabilidad: 4 días de trabajo, es decir, 16 horas. Total: *240 euros*.

Análisis del Sistema: 16 días, es decir, 64 horas. *960 euros*.

Diseño del Sistema: 9 días, es decir, 36 horas. *540 euros*.

Implementación: 49 días, es decir, 196 horas. *2940 euros*.

Implantación: 7 días, es decir, 28 horas. *420 euros*.

Coste total del proyecto (sin tener en cuenta impuestos como el IVA): 5100 euros.

9. Diagramas UML y casos de uso

A continuación se describe el diagrama de clases de la aplicación.

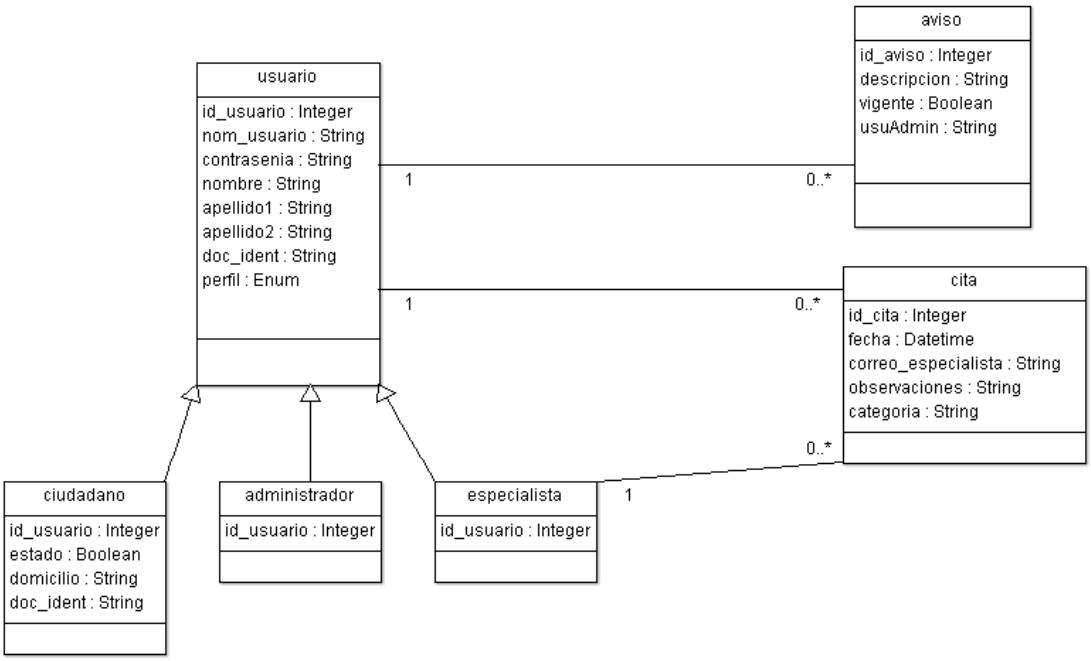


Ilustración 4. Diagrama de clases

Como se puede observar la entidad de usuario tiene una herencia en sus tres perfiles, ciudadano, administrador y especialista. Existen varios atributos comunes, entre los que destaca el atributo perfil, que se puede entender como un enumerado para que luego en el modelo de base de datos sea el atributo que discrimine a unos u otros, más allá de las relaciones existentes. Además de éstas tenemos las entidades aviso y cita.

El usuario se relaciona con los avisos porque es el usuario administrador el que puede poner de 0 a N avisos. A su vez, un aviso puede estar dado de alta por un usuario.

El usuario se relaciona con la cita porque un usuario ciudadano puede tener de 0 a N citas, y a su vez una cita está asignada a un único usuario. Para su mejor visualización se ha relacionado directamente a la entidad especialista con la cita, para reflejar que un especialista puede tener de 0 a N citas y una cita es dada por un único especialista.

Definición de los casos de uso

En esta sección se incluye la definición formal de los casos de uso que tiene la aplicación web.

IDENTIFICADOR	CU-001
NOMBRE	Login en el sistema
PRIORIDAD	1
DESCRIPCIÓN	Caso de uso para acceder al sistema mediante usuario y contraseña
ACTORES	Administrador Ciudadano Especialista
PRE-CONDICIONES	1.- El usuario/actor debe estar dado de alta en el sistema
INICIADO POR	
FLUJO	1.- El usuario accede al enlace de la aplicación web y le aparece una pantalla requiriendo usuario y contraseña para entrar. El usuario cumplimenta ambos campos. 2.- El usuario acepta pulsando el botón entrar. a) Si hay errores en los datos o en la validación ocurre una excepción. El sistema notifica con un mensaje de error al usuario. Vuelve al paso 1 para solicitar de nuevo los datos de usuario y contraseña. 3.- La operación termina con éxito.
POSTCONDICIONES	El usuario accede al sistema, accediendo a la pantalla correspondiente a su perfil.
NOTAS	

Tabla 1. Caso de uso - Login en el sistema

IDENTIFICADOR	CU-002
NOMBRE	Solicitar cita
PRIORIDAD	1
DESCRIPCIÓN	Caso de uso para solicitar una cita
ACTORES	Administrador Ciudadano
PRE-CONDICIONES	1.- El usuario debe estar dado de alta en el sistema 2.- El usuario debe estar logado. 3.- Deben existir horarios con fechas disponibles
INICIADO POR	
FLUJO	1.- El usuario accede a la solicitud de nueva cita. 2.- El usuario observa en una lista las fechas y especialistas disponibles. 3.- El usuario solicita la cita que desee. 3.1.- Si el usuario es administrador podrá pedir una cita para un usuario ciudadano introduciendo en la solicitud el correo del usuario al que reserva la cita. a) Si hay errores en los datos o en la validación ocurre una excepción. El sistema notifica con un mensaje de error al usuario. Volver al paso 1 para volver a solicitar fechas corrigiendo los datos o fin de caso de uso. 4.- La operación termina con éxito.
POSTCONDICIONES	El usuario tendrá una nueva cita reservada, es decir, a nivel de base de datos se actualizará el registro de la cita relacionando al usuario con ella.A su vez, el horario elegido dejará de estar disponible.
NOTAS	

Tabla 2. Caso de uso - Solicitar cita

IDENTIFICADOR	CU-003
NOMBRE	Cancelar cita
PRIORIDAD	1
DESCRIPCIÓN	Caso de uso para eliminar una cita ya dada de alta previamente
ACTORES	Administrador Ciudadano
PRE-CONDICIONES	1.- El usuario debe estar dado de alta en el sistema. 2.- El usuario debe estar logado. 3.- La cita debe estar dada de alta en el sistema.
INICIADO POR	
FLUJO	1.- El usuario accede a consultar sus citas. 2.- Localiza la cita que quiere eliminar. Una vez que obtiene la cita en cuestión procede a intentar eliminarla. 3.- El sistema advierte al usuario con un mensaje si de verdad desea eliminar la cita. 4.- La operación termina con éxito y el horario de la cita pasa a estar disponible.
POSTCONDICIONES	El usuario habrá eliminado la cita, por lo que la cita pasa a estar disponible para volver a ser solicitada por otro usuario.
NOTAS	

Tabla 3. Caso de uso - Cancelar cita

IDENTIFICADOR	CU-004
NOMBRE	Crear horario
PRIORIDAD	1
DESCRIPCIÓN	Caso de uso para crear horarios disponibles en la agenda
ACTORES	Administrador
PRE-CONDICIONES	1.- El usuario administrador debe estar dado de alta en el sistema. 2.- El usuario debe estar logado.
INICIADO POR	
FLUJO	1.- El usuario accede desde el menú al formulario de creación de horario. 2.- El usuario cumplimenta los datos del nuevo horario y acepta la operación. a) si hay errores en los datos introducidos se muestra un mensaje indicando este hecho y se vuelve al paso 2. 3.- La operación termina con éxito. Se notifica al usuario con un mensaje de que su solicitud se ha realizado con éxito.
POSTCONDICIONES	Se crea un nuevo horario de cita en la base de datos, que quedará disponible para que los usuarios las puedan solicitar.
NOTAS	

Tabla 4. Caso de uso - Crear horario

IDENTIFICADOR	CU-005
NOMBRE	Consulta de citas
PRIORIDAD	1
DESCRIPCIÓN	Caso de uso para consultar la agenda de citas
ACTORES	Administrador Ciudadano Especialista
PRE-CONDICIONES	1.- El usuario debe estar dado de alta en el sistema. 2.- El usuario debe estar logado. 3.- Para obtener algún resultado deben existir alguna cita dada de alta.
INICIADO POR	
FLUJO	1.- El usuario utiliza la funcionalidad de consulta para encontrar citas ya existentes. 2.- Tras acceder se muestran los resultados obtenidos: 3.- Si existe una cita o varias se mostrará un listado con las citas obtenidas.
POSTCONDICIONES	Se habrá realizado una consulta a la base de datos que devuelve las citas que cumplan los criterios de búsqueda, presentándose en forma de listado en una tabla.
NOTAS	

Tabla 5. Caso de uso - Consulta de citas

IDENTIFICADOR	CU-006
NOMBRE	Añadir avisos de interés
PRIORIDAD	2
DESCRIPCIÓN	Caso de uso para que los administradores incluyan noticias o avisos de interés que se mostrarán en la pantalla de inicio de los usuarios
ACTORES	Administrador
PRE-CONDICIONES	1.- El usuario debe estar dado de alta en el sistema. 2.- El usuario debe estar logado.
INICIADO POR	
FLUJO	1.- El usuario administrador desde el punto de menú correspondiente accede a dar de alta un nuevo aviso. 2.- Se cumplimenta el formulario de nuevo aviso y pulsa añadir. a) Si no se cumplimentan correctamente los datos obligatorios o existen problemas en la validación se muestra un mensaje de error. 3.- La operación se realiza con éxito. Se redirige al usuario a la pantalla de avisos y se comprueba que el aviso de interés se ha añadido correctamente.
POSTCONDICIONES	Se añadirá un nuevo registro en la tabla de Avisos de la base de datos. Si es un aviso con estado vigente se mostrará en la página principal del usuario
NOTAS	

Tabla 6. Caso de uso - Añadir avisos de interés

IDENTIFICADOR	CU-007
NOMBRE	Modificar avisos
PRIORIDAD	2
DESCRIPCIÓN	Caso de uso para que los administradores modifiquen noticias o avisos de interés
ACTORES	Administrador
PRE-CONDICIONES	1.- El usuario debe estar dado de alta en el sistema. 2.- El usuario debe estar logado.
INICIADO POR	
FLUJO	1.- El usuario administrador desde el punto de menú correspondiente accede a consultar los avisos. 2.- Se busca el aviso que se quiere modificar y se accede a su modificación. 3.- Se modifican los campos que se deseen modificar y se acepta la operación. a) Si no se cumplimentan correctamente los datos obligatorios o existen problemas en la validación se muestra un mensaje de error y se vuelve al paso 2. 3.- La operación se realiza con éxito. El usuario automáticamente puede comprobar que los datos se han modificado correctamente.
POSTCONDICIONES	Se modificará el registro en cuestión en la tabla de Avisos de la base de datos. Si es un aviso con estado vigente se mostrará en la página principal del usuario
NOTAS	

Tabla 7. Caso de uso - Modificar avisos

IDENTIFICADOR	CU-008
NOMBRE	Eliminar avisos
PRIORIDAD	2
DESCRIPCIÓN	Caso de uso para que los administradores eliminen noticias o avisos de interés
ACTORES	Administrador
PRE-CONDICIONES	1.- El usuario debe estar dado de alta en el sistema. 2.- El usuario debe estar logado. 3.- El aviso no puede tener estado "Vigente".
INICIADO POR	
FLUJO	1.- El usuario administrador desde el punto de menú correspondiente accede a consultar los avisos. 2.- Se busca el aviso que se quiere modificar y se accede a su borrado pulsando el botón Borrar/Eliminar. 3.- El sistema advierte al usuario con un mensaje si de verdad desea eliminar el aviso. 4.- La operación termina con éxito. El usuario automáticamente podrá comprobar en pantalla que el aviso de interés ya no aparece.
POSTCONDICIONES	El usuario habrá eliminado el aviso, con su correspondiente borrado del registro en la base de datos.
NOTAS	

Tabla 8. Caso de uso - Eliminar avisos

IDENTIFICADOR	CU-009
NOMBRE	Modificar contraseña
PRIORIDAD	2
DESCRIPCIÓN	Caso de uso para que los usuarios modifiquen su contraseña de acceso al sistema
ACTORES	Administrador Ciudadano Especialista
PRE-CONDICIONES	1.- El usuario debe estar dado de alta en el sistema. 2.- El usuario debe estar logado.
INICIADO POR	
FLUJO	1.- Para acceder al formulario de modificación de contraseña, el usuario debe clicar encima de su nombre de usuario en el menú y le aparecerá la opción de menú de modificación de contraseña para acceder al formulario. 2.- El usuario cumplimenta los datos de la nueva contraseña y acepta la operación. a) si hay errores en los datos introducidos se muestra un mensaje indicando este hecho y se vuelve al paso 2. 3.- La operación termina con éxito. Se notifica al usuario con un mensaje de que su solicitud se ha realizado con éxito.
POSTCONDICIONES	La contraseña del usuario se ha modificado y se consolida el cambio en la base de datos.
NOTAS	

Tabla 9. Caso de uso - Modificar contraseña

IDENTIFICADOR	CU-010
NOMBRE	Crear usuario
PRIORIDAD	1
DESCRIPCIÓN	Caso de uso para que los usuarios administradores den de alta nuevos usuarios en el sistema
ACTORES	Administrador
PRE-CONDICIONES	1.- El usuario administrador debe estar dado de alta en el sistema. 2.- El usuario debe estar logado.
INICIADO POR	
FLUJO	1.- El usuario accede desde el menú al formulario de alta de usuario. 2.- El usuario cumplimenta los datos del nuevo usuario y acepta la operación. a) si hay errores en los datos introducidos se muestra un mensaje indicando este hecho y se vuelve al paso 2. 3.- La operación termina con éxito. Se notifica al usuario con un mensaje de que su solicitud se ha realizado con éxito.
POSTCONDICIONES	Se genera un nuevo usuario en el sistema y en la base de datos
NOTAS	

Tabla 10. Caso de uso - Crear usuario

IDENTIFICADOR	CU-011
NOMBRE	Logout del sistema
PRIORIDAD	1
DESCRIPCIÓN	Caso de uso para abandonar el sistema
ACTORES	Administrador Ciudadano Especialista
PRE-CONDICIONES	1.- El usuario debe estar dado de alta en el sistema. 2.- El usuario debe estar logado.
INICIADO POR	
FLUJO	1.- El usuario utilizará el botón de desconexión del sistema. 2.- Se cierra el sistema llevando al usuario a la pantalla de login.
POSTCONDICIONES	El botón de desconexión cerrará la sesión activa del usuario y le redirigirá a la pantalla de login del sistema.
NOTAS	Se reiniciarán todas las variables globales $scope y $rootScope

Tabla 11. Caso de uso - Logout del sistema

10. Prototipos

En base a los casos de uso descritos anteriormente, y a los flujos de interacción que se detallarán en el siguiente apartado del documento, se han llevado a cabo los prototipos de la aplicación. En primer lugar se muestran prototipos hechos a mano alzada, 'sketches' escaneados, que dan una primera idea de lo que se quiere reflejar (aunque finalmente la implementación no tenga por qué ser estrictamente igual a los prototipos). Cabe destacar que para no incluir todas las posibles páginas y navegaciones posibles se han incluido los casos más relevantes. Por ejemplo, dado que hay varias funcionalidades de 'Alta', sólo se ha incluido la 'Solicitud de citas'. El resto de altas y funcionalidades de modificación y borrado serían análogas, variando en todo caso los campos o atributos de la funcionalidad.

El mismo comportamiento se ha llevado a cabo con las consultas, sólo se ha incluido la consulta "Ver mis citas" que puede realizar uno de los perfiles de usuario.

Por tanto, los prototipos serían los siguientes.

Ilustración 5. Login

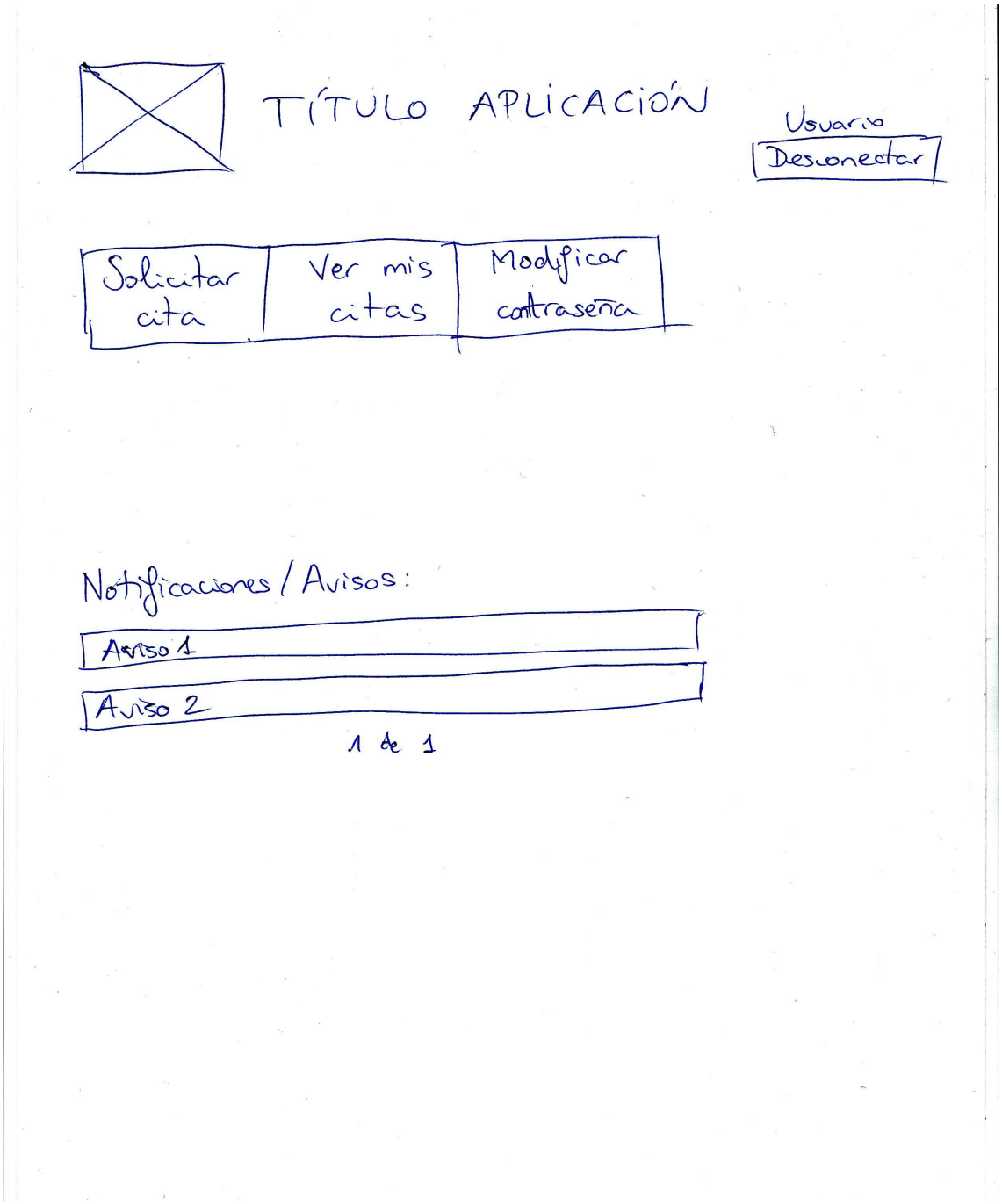


Ilustración 6. Pantalla de inicio de usuario Ciudadano

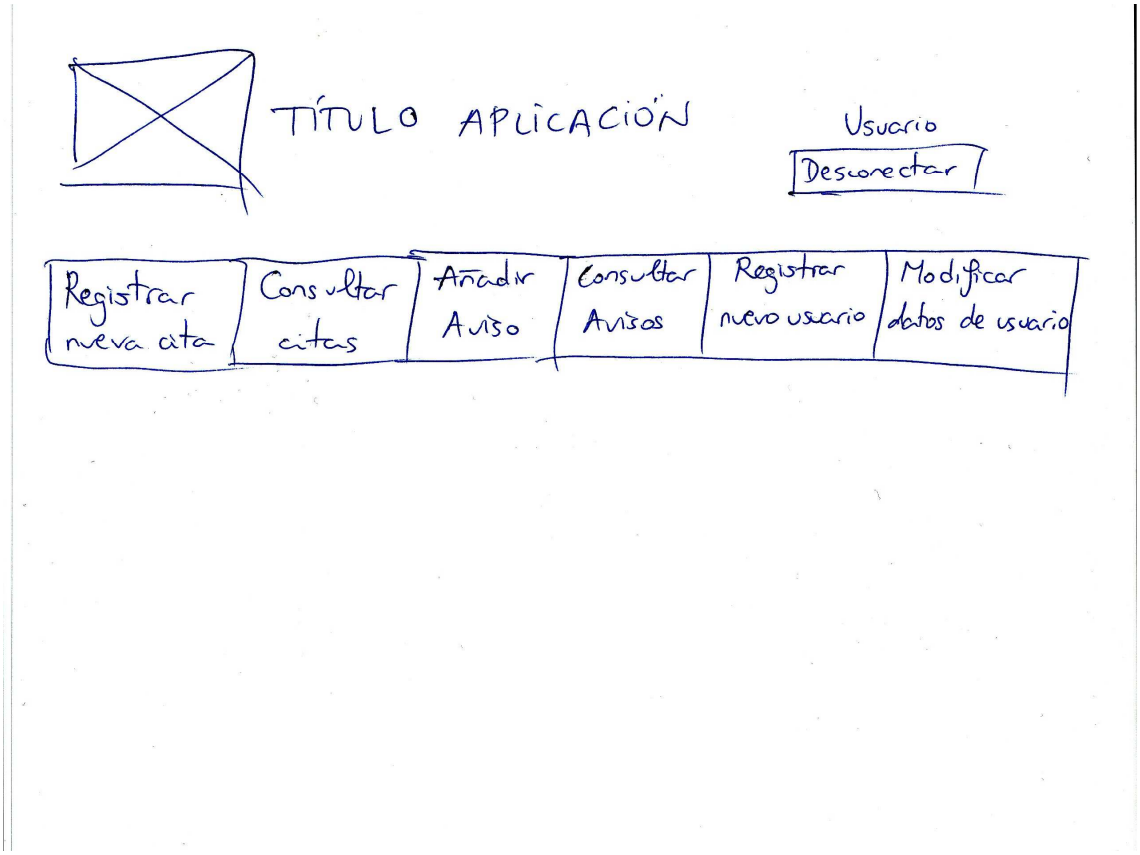


Ilustración 7. Pantalla de inicio de usuario Administrador

En esta pantalla de administrador se puede observar como el menú contiene todas las funcionalidades que puede realizar. Aunque no esté incluido en el prototipo se puede tener en cuenta la posibilidad de incluir los avisos de interés también en este perfil y en este contenido.

Ilustración 8. Pantalla de inicio de usuario Especialista

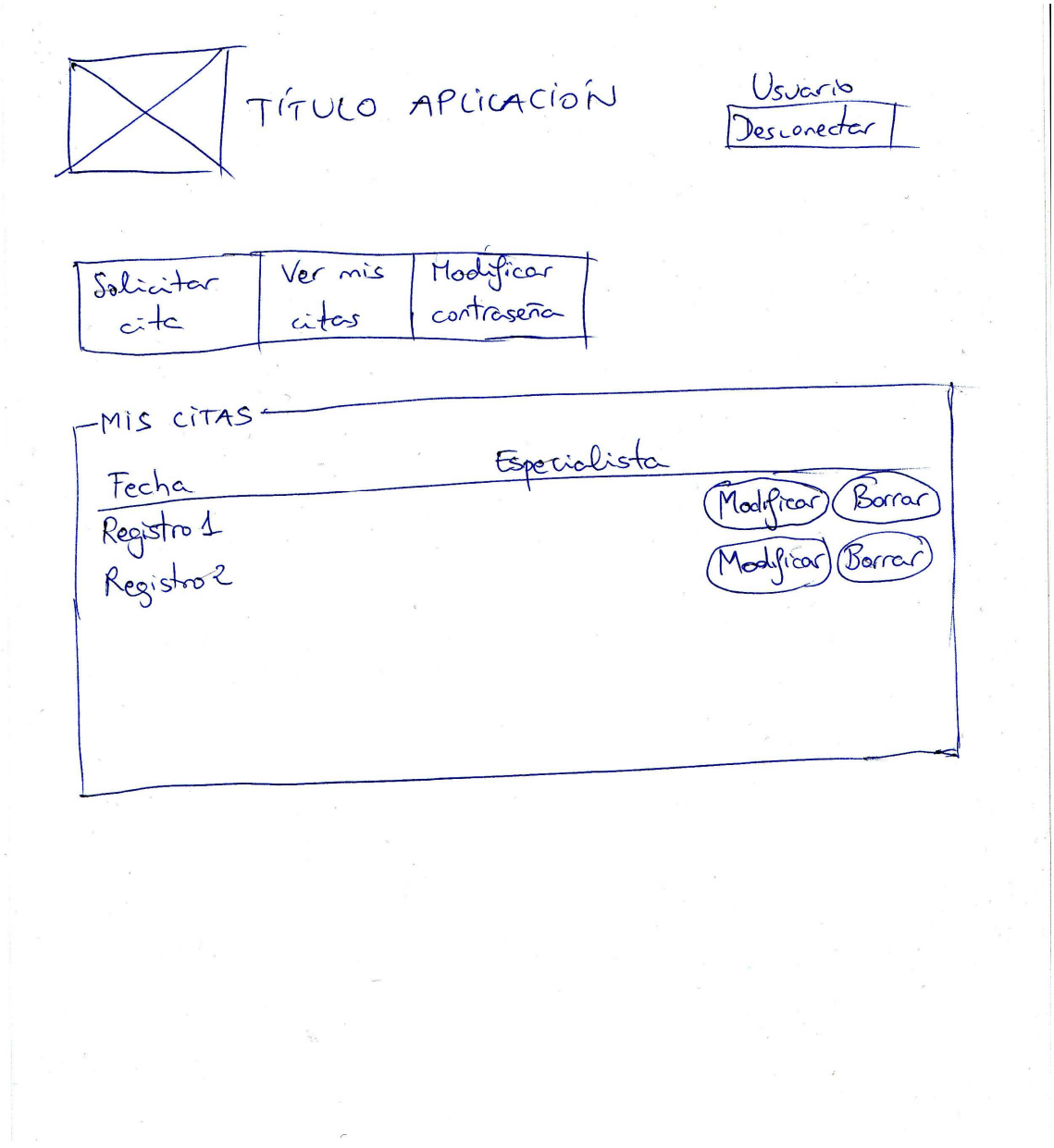


Ilustración 9. Pantalla en la que el usuario Ciudadano puede consultar sus citas

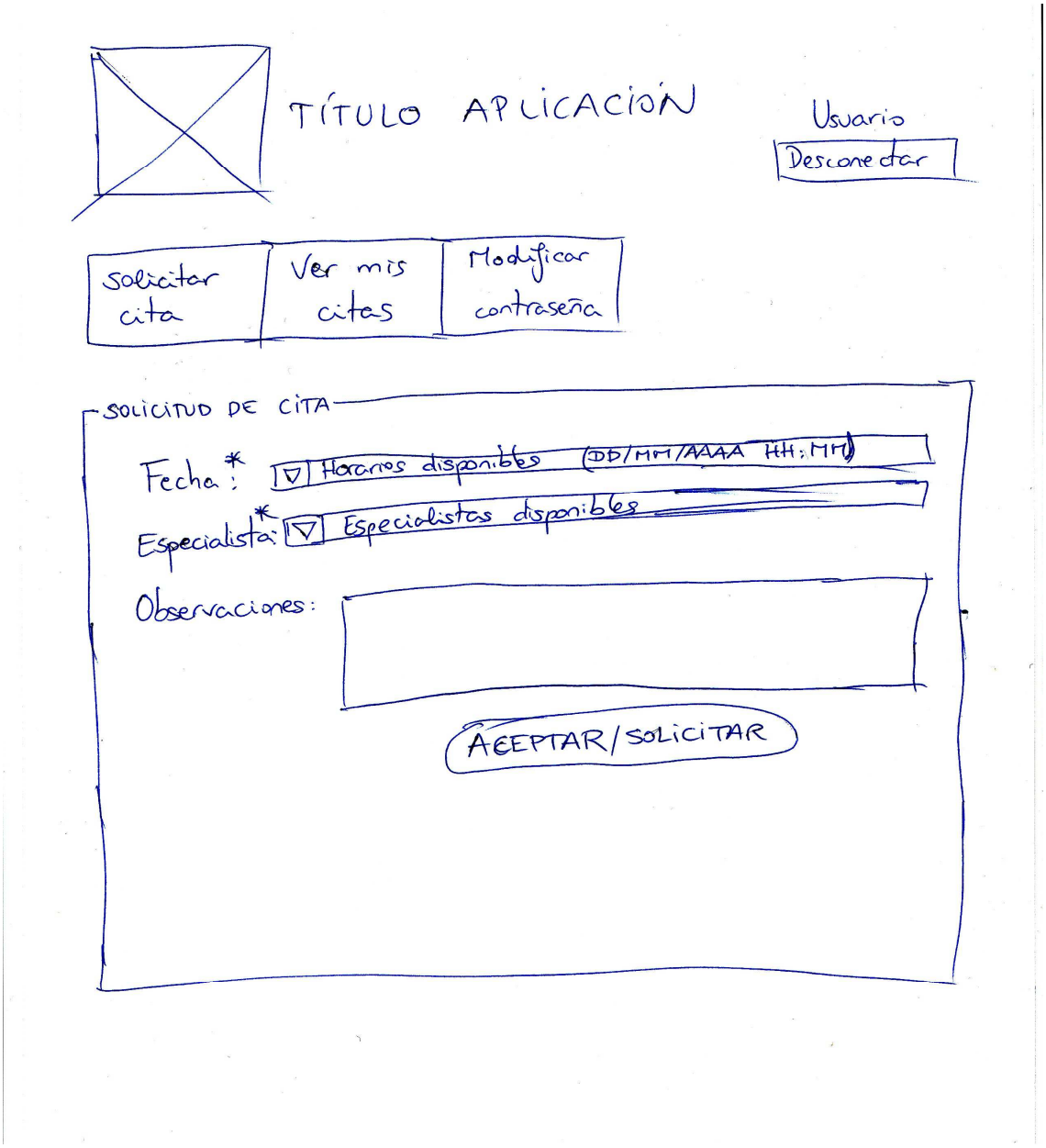


Ilustración 10. Pantalla de Solicitud de nueva cita del usuario Ciudadano

Como se decía, además de los prototipos a mano alzada, se han replicado en prototipos horizontales de alta fidelidad, utilizando la herramienta WireFrameSketcher [13]. Son los siguientes:

Ilustración 11. Prototipo horizontal de alta fidelidad: Login

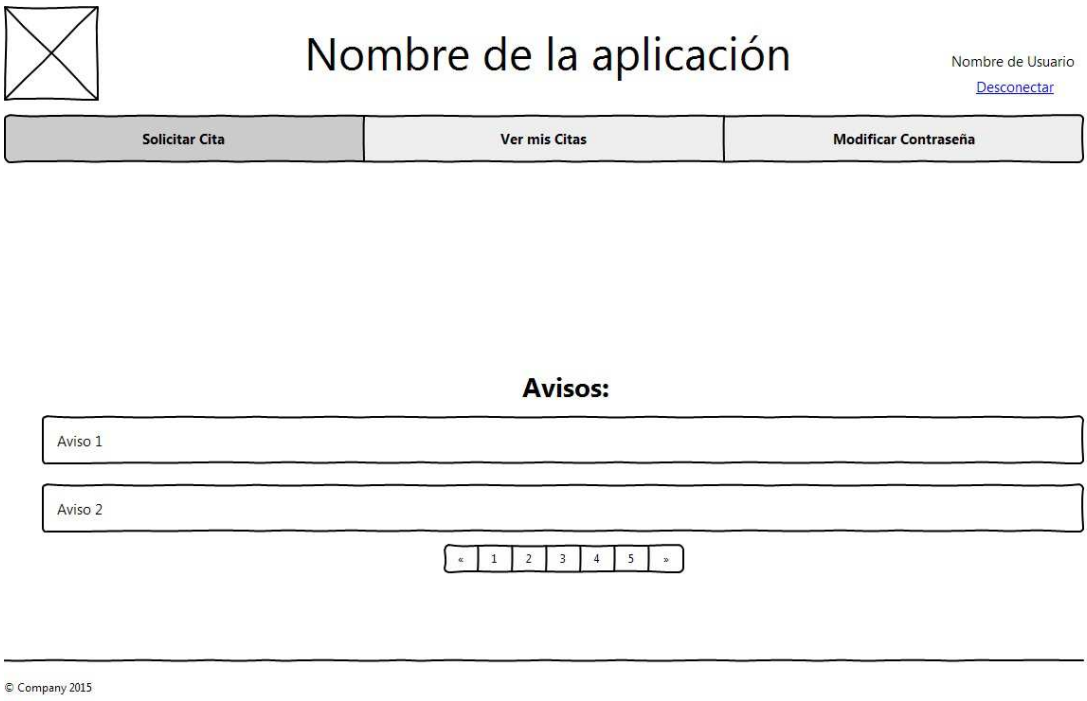


Ilustración 12. Prototipo horizontal de alta fidelidad: Pantalla de inicio de usuario Ciudadano

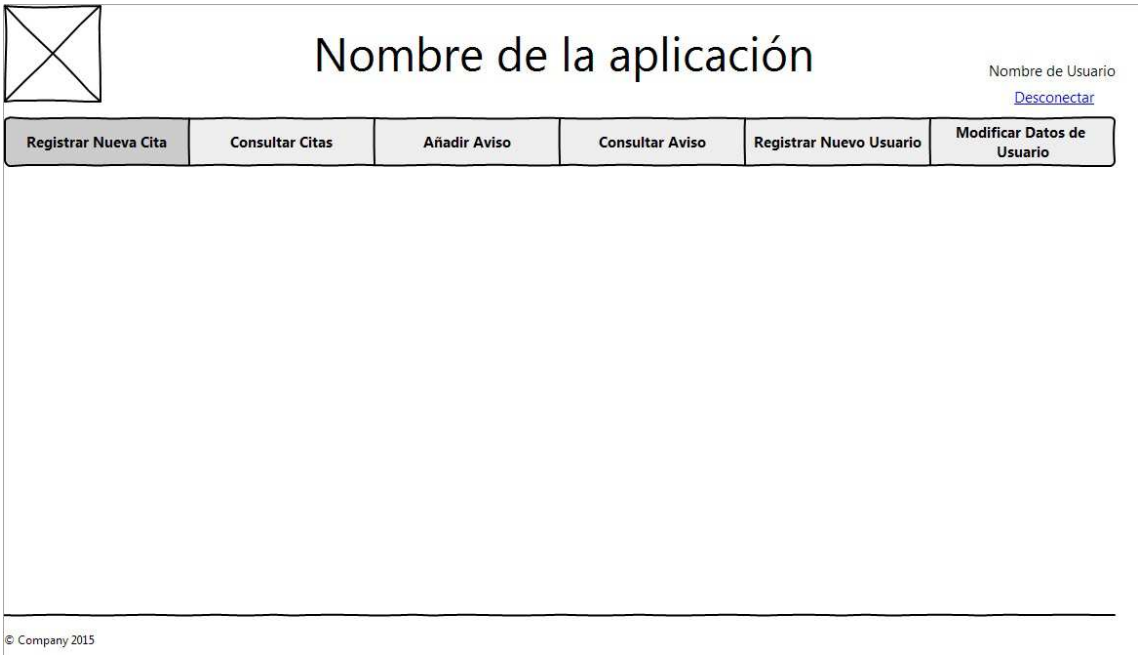


Ilustración 13. Prototipo horizontal de alta fidelidad: Pantalla de inicio de usuario Administrador

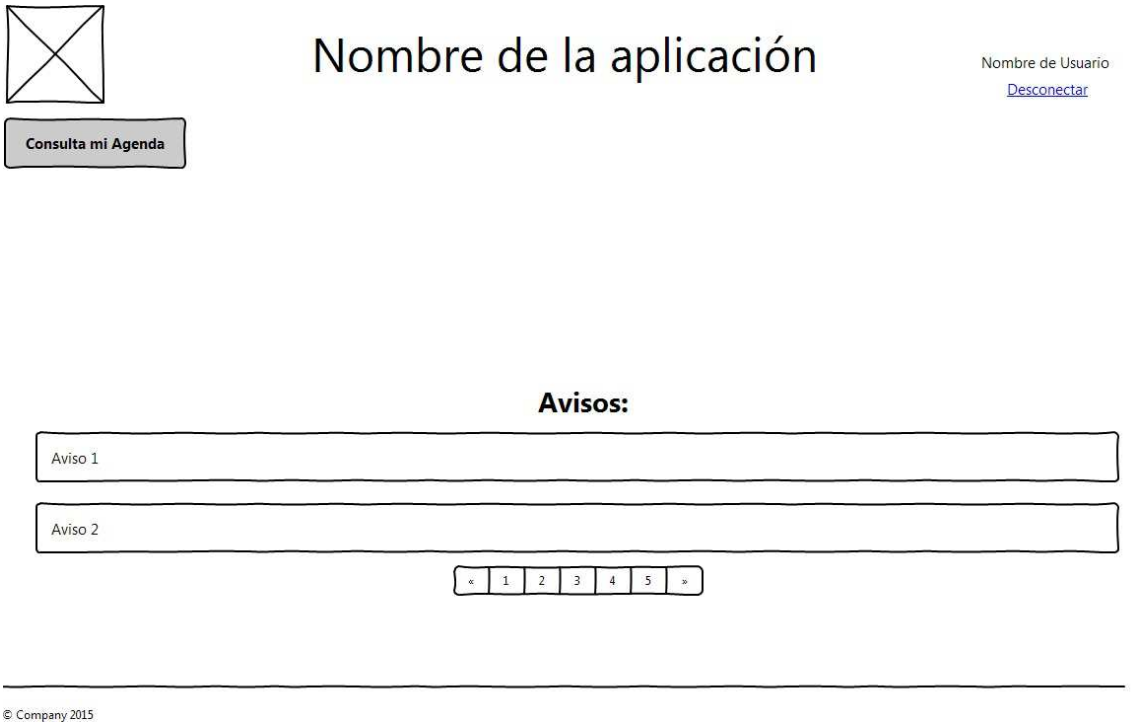


Ilustración 14. Prototipo horizontal de alta fidelidad: Pantalla de inicio de usuario Especialista

Ilustración 15. Prototipo horizontal de alta fidelidad: Listado de citas del usuario Ciudadano

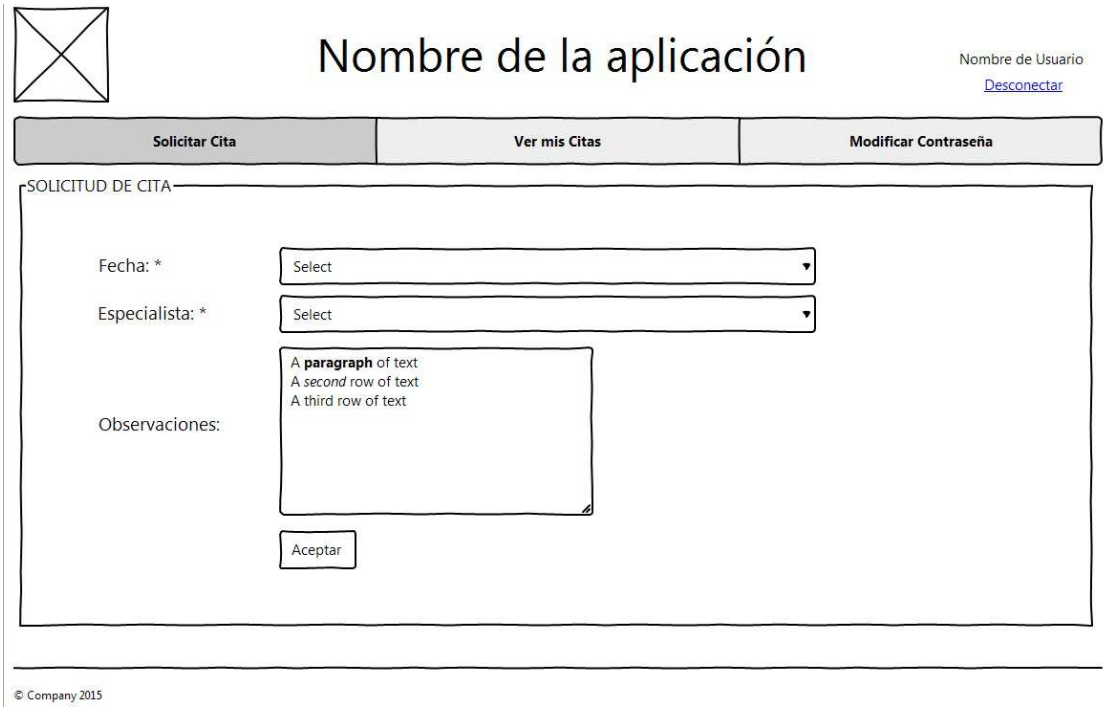


Ilustración 16. Prototipo horizontal de alta fidelidad: Pantalla de solicitud de nueva cita del usuario Ciudadano

Decisiones del diseño del prototipo

Para la elaboración de los prototipos ha sido preciso tomar decisiones de diseño. La idea del alumno en cuanto a la interfaz es un diseño sencillo, atractivo y no demasiado sobrecargado. De esta forma, en la parte superior de cada página se pretende incluir un pequeño logo de la aplicación, el nombre de la misma (o incluirla en la ventana del navegador web), el menú correspondiente al perfil del usuario, el correo de éste y un enlace para desconectar o salir de la aplicación.

A su vez, en la parte inferior de la página se tendrá un pie con información relevante. En los prototipos se ha incluido un pie estándar para una empresa pero se podrá cumplimentar con información del alumno, la versión de la aplicación, etc.

Por último, entre ambos, se incluirá el contenido de cada página. La mayoría de las páginas serán formularios de introducción de datos aunque no se descarta el uso de tablas para mostrar listados. En los prototipos se han utilizado 'fieldsets' para encuadrar los formularios, aunque parte únicamente como idea.

Evaluación del prototipo

Para evaluar el prototipo diseñado se utiliza la técnica del test con usuarios, con la que preguntamos a los usuarios su opinión sobre diversos aspectos del mismo. Este test se realiza con cinco trabajadores del centro de Servicios Sociales, con perfiles diferentes, para intentar extraer las mejores conclusiones y opiniones posibles.

En este sentido, y siendo esta fase posterior a la toma de requisitos y puesta en marcha de la técnica de indagación, resulta conveniente que al menos un grupo de los usuarios que ya participaron en esa fase, participen en esta evaluación, pues ya se conocen sus contextos de uso.

Además, también se conoce información de dichos usuarios gracias a las encuestas realizadas anteriormente en la toma de requisitos; preguntas relacionadas con su perfil de trabajo, experiencia, uso de herramientas web, etc. Esta información obtenida es importante y necesaria para esta prueba de evaluación.

Para poder completar esta prueba los usuarios realizarán diferentes tareas y tendrán que contestar a una serie de preguntas sobre estas tareas. En función de las respuestas y el comportamiento que se detecte durante la prueba se podrán analizar los puntos fuertes y débiles del prototipo, con el objetivo de mejorarlo de cara al producto final.

Las tareas propuestas a realizar durante esta prueba están en algún caso íntimamente relacionadas con los escenarios de uso, y son las siguientes:

- Escenario: un ciudadano usuario del sistema quiere solicitar una nueva cita. Tareas: el usuario tiene que acceder e identificarse en el sistema y solicitar una nueva cita en la fecha que desee.

- Escenario: Un ciudadano usuario del sistema desea cancelar una cita. Tareas: el usuario tiene que acceder e identificarse en el sistema, buscar sus citas y cancelar la que desee.

- Escenario: un usuario presencialmente en el centro solicita un recordatorio de su cita.

  Tareas: con perfil administrador acceder e identificarse en el sistema y, conocido el documento de identificación del usuario, buscar sus citas para podérselas comunicar.

- Escenario: un especialista quiere consultar su agenda en un momento determinado.
  Tareas: con perfil especialista acceder e identificarse en el sistema y buscar sus citas para consultarlas.

Posterior a la prueba, los usuarios deben responder a una serie de preguntas. Se prepara el siguiente cuestionario:

PREGUNTAS POST-TEST					
• Nombre y Apellidos:					
• ¿Crees que el prototipo en real será fácil de utilizar?	1	2	3	4	5
1- muy difícil 5-muy fácil					
• ¿Te has sentido perdido en la navegación?	1	2	3	4	5
1- nunca 5-siempre					
• ¿Consideras que la apariencia gráfica a priori facilita la navegación?	1	2	3	4	5
1- poco 5-mucho					
• Teniendo en cuenta que es un prototipo, ¿Has podido finalizar las tareas descritas en el test?	1	2	3	4	5
1- poco 5-mucho					
• ¿Has precisado de ayuda para alguna de las tareas?	1	2	3	4	5

• 1- nunca 5-siempre					
• ¿Has conseguido encontrar algún elemento de ayuda?	1	2	3	4	5
• 1- nunca 5-siempre					
• ¿La estructura del prototipo te ha parecido correcta?	1	2	3	4	5
• 1- nunca 5-siempre					
• ¿Qué es lo que más te ha gustado del prototipo?					
• ¿Qué es lo que menos te ha gustado del prototipo?					
• Observaciones					

Tabla 12. Cuestionario post-test

11. Usuarios y contextos de uso

Para la etapa de requisitos se necesitan llevar a cabo una serie de tareas y actividades que involucren a los usuarios con el objetivo de obtener información sobre el producto o producto que se precisa desarrollar.

Esta información nos permitirá conocer a los usuarios, sus contextos de uso, sus necesidades y objetivos, por tanto se obtendrá información muy valiosa de la aplicación que necesitan. Para recopilar esta información se utilizan métodos de indagación. Sin entrar en detalle de todos los posibles y diferentes métodos existentes, para el proyecto que nos ocupa se van a utilizar dos métodos de indagación:

- Observación, en concreto en su modalidad de investigación contextual.

- Encuestas.

El método de observación es un método cualitativo que permite, mediante la observación de los usuarios en su entorno habitual, conocer qué hacen estos, cómo lo hacen, qué comportamiento tienen, etc. Utilizando su vertiente de investigación contextual se pueden observar esta serie de características in situ, mientras el usuario interacciona con su entorno.

Las encuestas son un método de indagación cuantitativo y permite, mediante la realización de las mismas, obtener respuestas y alternativas ante las dudas que hayan podido surgir.

¿Por qué se han elegido ambos métodos?

Se han elegido estos métodos porque se considera que se complementan muy bien y ofrecen un espectro completo de las necesidades de los usuarios y el flujo de actuación que necesita la aplicación.

Como el sistema de gestión de cita de Servicios Sociales no está automatizado, métodos como el Logging o el Análisis competitivo podrían resultar insuficientes e inadecuados.

Observación, investigación contextual. Desarrollo y conclusiones.

Como método cualitativo, la observación nos ha permitido desplazarnos al entorno de trabajo y comprobar en persona el día a día del funcionamiento de este servicio en un municipio concreto de la geografía española. Los empleados atienden las solicitudes de los usuarios de manera presencial o telefónica. La agenda de horarios se mantiene y gestiona mediante una base de datos ofimática a la que accede el personal trabajador que recepciona las solicitudes.

Lejos de ser un funcionamiento cómodo, dependiendo del volumen de solicitudes y de la simultaneidad de las mismas no es raro encontrarse con casos de duplicidades y necesidad de correcciones en las citas. Además, los usuarios, llamando por teléfono o

desplazándose presencialmente, no saben a priori los horarios disponibles, por lo que cada consulta, sea por el medio que sea, tiene una dilatación en el tiempo que ocasiona mayor tiempo de espera para los usuarios y por tanto menor agilidad en el servicio.

Esta fue una de las principales necesidades que constatamos realizando el método de observación. Además fue una de las principales necesidades obtenidas en la entrevista en profundidad que realizamos mediante este método entre los trabajadores y usuarios.

En términos de gestión, los trabajadores agradecerían también una funcionalidad que les permita gestionar las citas y solicitudes de manera más amigable. Aunque con el paso del tiempo han conseguido mejorar y depurar la funcionalidad ofimática que utilizan, el servicio sigue creciendo, el volumen de solicitudes y de usuarios crece al ritmo de la ciudad y esta funcionalidad se les queda pequeña para agilizar la atención del servicio.

Además, la comunicación con los especialistas que dan las actividades se realiza de manera diaria, vía presencial, telefónica o correo electrónico, para trasladar la agenda de horarios de cada uno de ellos. Aunque estos especialistas pueden acceder a la aplicación ofimática se pretende que este acceso sea sólo por las personas que gestionan las citas directamente con el usuario, con el objetivo de evitar una gestión con demasiados trabajadores implicados en la gestión. Por tanto, salvo un posible cambio en la estructura de trabajo, sí parece necesario que al menos los especialistas tengan un ámbito de consulta que les permita estar informados puntualmente de la agenda sin necesidad de preguntarle a nadie.

Por último, los usuarios, muy contentos con el servicio en general, con la atención y las actividades que se ofrecen, sí muestran cierta insatisfacción en la poca agilidad de la gestión de las solicitudes de las citas, tal y como se indicaba anteriormente.

La diversidad de público y rangos de edades precisan una aplicación sencilla, fácil de utilizar y pensada en la comodidad y usabilidad de todos los usuarios. Aunque la ventana presencial, e incluso telefónica, se mantendrá operativa para los usuarios que lo necesiten, con esta aplicación que se desea desarrollar se pretende centralizar todo el servicio y dotarle de la agilidad que ahora no tiene. No obstante, será imprescindible la sencillez de interfaz y funcionamiento para que la aplicación no caiga en desuso y los usuarios vuelvan a los métodos de comunicaciones vigentes hasta ahora. Aunque lógicamente algunos usuarios mantengan la rutina de solicitud actual, se pretende que un gran porcentaje del volumen de usuarios actual y potencial utilicen de manera habitual la nueva aplicación.

Encuestas. Desarrollo y conclusiones.

La riqueza de información que nos aportó la utilización del método de observación fue impagable. Pero tras realizar un análisis de esta información se pensó que el uso de encuestas podrían potenciar y dotar de más calidad todos los requerimientos y necesidades que se habían recogido, especialmente porque hubo algunas preguntas

realizadas con el método anterior que realizadas en un entorno de trabajo no tenían el reposo necesario para que las respuestas fueran pensadas y cotejadas perfectamente.

Por tanto, se ha elaborado el siguiente formulario de preguntas, para en primera instancia enviárselo a 3 trabajadores que el servicio de Servicios Sociales ha tenido a bien que colabore con nosotros. No obstante, este formulario se mandará después a una muestra de trabajadores mayor, para tener mayor variedad de respuestas.

ENCUESTA
• Nombre y Apellidos: • Edad: • Puesto de trabajo: • ¿Cuántos años lleva trabajando en este servicio de gestión de citas? • ¿Cuánto tiempo dedica diariamente a la atención de solicitudes de citas? • ¿Aproximadamente cuál es el número de solicitudes recibidas diariamente en el servicio? • En términos de porcentaje, indíquenos el volumen de solicitudes recibidas presencialmente y vía teléfono, teniendo ambas que completar un 100%. • ¿Utiliza normalmente formularios vía Internet o aplicaciones web telemáticas para completar solicitudes? • Indique las observaciones o puntualizaciones que considere convenientes.

Tabla 13. Encuesta a trabajadores

Con esta encuesta se pretende conocer el tipo de trabajadores que realizan la encuesta, la experiencia que tienen, el volumen de atención y solicitudes que llegan al

servicio y una pregunta, no demasiado relevante, que nos indique si el trabajadores suele realizar solicitudes vía Internet como la aplicación que se plantea poner en marcha. Además, se les permite puntualizar lo que necesiten en la última pregunta.

Los resultados obtenidos de la primera muestra, 3 trabajadoras, nos muestran que estas trabajadoras son las personas que recepcionan las solicitudes de citas, con edades de los 25 a los 45 años, tal y como veremos más adelante.

Aproximadamente dedican cada una 4 horas en atender solicitudes, en horario de mañana o tarde.

Normalmente se reciben unas 50 solicitudes diarias, y semanalmente, en función de la semana, unas 250 ó 300 solicitudes.

El volumen de atención es de un 60% de atención presencial y un 40% de atención telefónica. Muchos de los usuarios aprovechan para acudir al centro personalmente pero en la atención al público son varios los servicios que se atienden por lo que la tendencia está cambiando. Se nos ocurre que podría ser interesante destinar algunos puestos con ordenadores para que, con la aplicación publicada, cada usuario pueda realizar in situ las solicitudes sin necesidad de esperar tanto tiempo.

Por último, dos de las tres trabajadoras nos indican que tienen cierto manejo en aprovechar los servicios telemáticos que pueden para realizar solicitudes.

Perfiles de usuario identificados

De los dos métodos de indagación, especialmente del primero, se han identificado diferentes perfiles de usuario que interactúan con el sistema. A continuación, se indica cada perfil con una ficha que muestra las principales características de cada uno de ellos.

PERFIL DE USUARIO	
USUARIO	Usuario ciudadano/a.
EDAD	Usuarios/as entre los 25 y 80 años, con algunos casos de menor y mayor edad a dicho rango.
SEXO	Hombres y mujeres sin distinción.
PROFESIÓN	Profesiones variopintas y estudios académicos de todo tipo.
MOTIVACIONES e INTERESES	Las motivaciones e intereses de estos usuarios son los de disfrutar de un buen servicio, mejorando los aspectos de solicitud de las actividades, en los que no siempre reciben la

	inmediatez que desearían.
EXPERIENCIA EN EL USO DE APLICACIONES WEB	Cierta experiencia en el uso de aplicaciones y herramientas web especialmente en usuarios con edades menores de 65 años. A partir de esa edad, sin apenas experiencia. Mucha experiencia en usuarios menores de 40 años.
CONTEXTO DE USO	Este perfil de usuario es el principal actor de esta aplicación. Es el usuario objetivo por el que se lleva a cabo la aplicación. Su contexto de uso es el de consulta y solicitud de citas para disfrutar de las actividades que dispone el servicio
TAREAS	Las tareas que realizará este perfil de usuario son: - Acceder al sistema mediante credenciales de usuario y contraseña - Cambiar la contraseña - Consulta de horarios disponibles y ocupados - Solicitud de cita - Cancelación de cita - Consulta de citas previas

Tabla 14. Perfil de usuario ciudadano/a

PERFIL DE USUARIO	
USUARIO	Trabajador/a con perfil administrador/a
EDAD	Empleados/as entre los 30 y 55 años.
SEXO	Hombres y mujeres sin distinción
PROFESIÓN	Empleados con categoría de administrativos o auxiliares administrativos de Servicios Sociales del municipio
MOTIVACIONES e INTERESES	Las motivaciones e intereses de este perfil de usuario es la de ofrecer el mejor servicio posible, teniendo las mejores herramientas para que así sea.

EXPERIENCIA EN EL USO DE APLICACIONES WEB	Cierta experiencia en el uso de aplicaciones y herramientas web debido al uso para otro tipo de aspectos en sus puestos de trabajo y con notoria experiencia en herramientas ofimáticas.
CONTEXTO DE USO	El contexto de uso es el de administración, por tanto el de gestión de la agenda. No obstante, para la atención telefónica precisarán realizar solicitudes en nombre de los usuarios que les llamen. Este tipo de función no es seguro que se encuentre en la primera versión del sistema.
TAREAS	Las tareas que realizará este perfil de usuario son: - Acceder al sistema mediante credenciales de usuario y contraseña - Gestión de la agenda (creación de horarios, consulta de citas, etc.) - Gestionar usuarios (dar de alta nuevos usuarios) -Tareas relacionadas con los usuarios, haciéndose pasar por ellos en momentos puntuales (solicitud de citas, cancelación, consulta, etc.). - Gestionar los avisos del sistema (alta, baja y modificación).

Tabla 15. Perfil de usuario administrador/a

PERFIL DE USUARIO	
USUARIO	Especialista con perfil Consultor
EDAD	Empleados/as entre los 30 y 50 años.
SEXO	Hombres y mujeres sin distinción
PROFESIÓN	Empleados de Servicios Sociales del municipio con distintas categorías (psicólogos, docentes, psiquiatras, etc.).

MOTIVACIONES e INTERESES	Las motivaciones e intereses de este perfil de usuario es la de ofrecer el mejor servicio posible, teniendo las mejores herramientas para que así sea.
EXPERIENCIA EN EL USO DE APLICACIONES WEB	Cierta experiencia en el uso de aplicaciones y herramientas web.
CONTEXTO DE USO	El contexto de uso es el de la consulta de la agenda. No serán los actores que den de alta, modifiquen o cancelen citas en la aplicación.
TAREAS	Las tareas que realizará este perfil de usuario son: - Acceder al sistema mediante credenciales de usuario y contraseña - Modificar contraseña de usuario - Consulta de la agenda.

Tabla 16. Perfil de usuario especialista

Escenarios de uso

Con los escenarios de uso podemos suponer situaciones hipotéticas que consideramos puedan darse en un momento concreto. Con ellos se suelen identificar aspectos importantes que afectan a la utilización de una aplicación o un producto en el mundo real y que no pueden identificarse ni tenerse en cuenta de otro modo.

De esta forma, suponiendo que la aplicación ya ha sido desarrollada, se plantean a continuación algunos escenarios con los que se observa cómo interactúan los distintos usuarios con la aplicación en un contexto determinado. No obstante, no se puede dar por sentado que todos los usuarios se van a comportar de la misma manera, cada uno tiene experiencias y expectativas diferentes por lo que los siguientes escenarios deben tomarse como ejemplo.

Escenario 1: Un usuario presencialmente en el centro solicita un recordatorio de su cita

Durante el transcurso de una mañana cualquiera un usuario acude al centro de Servicios Sociales solicitando que le recuerden la próxima cita que tiene ya solicitada previamente. Aunque lo ideal es que dicho usuario consulte su cita en la aplicación mediante los ordenadores del centro dispuestos para ello, insiste en que le realicen la gestión. La trabajadora del mostrador de atención al ciudadano decide, a su juicio y en vistas de la cola que hay en los ordenadores del centro, realizar la gestión.

Para llevar a cabo la gestión la trabajadora, que tiene perfil administradora del sistema, necesita los datos personales del usuario, en este caso el documento de identificación del mismo. Una vez el usuario le indica su DNI, la trabajadora accede al sistema.

En primer lugar, la trabajadora se identifica introduciendo su usuario y contraseña en los dos campos habilitados para ello en la página principal del sistema. Pulsa Aceptar y el sistema después de validar los datos accede a la gestión permitida para los usuarios con perfil Administrador.

Seguidamente, la trabajadora observa los campos del filtro de búsqueda de la agenda y en el campo "Documento de Identificación" introduce el DNI que el usuario le había indicado previamente. A continuación pulsa Consultar.

El sistema le devuelve una serie de resultados, en este caso un único registro. En dicho registro, en forma de tabla, se pueden observar los datos de esa cita.

La trabajadora traslada al ciudadano la información con la fecha y los datos de la cita y éste agradeciendo la gestión abandona el centro satisfecho.

Escenario 2: Un ciudadano usuario del sistema desea cancelar una cita

Un ciudadano que es usuario del sistema necesita cancelar una cita que tenía solicitada para realizar otra la semana siguiente. Tiene constancia de que existe una funcionalidad para cancelar una cita ya solicitada por lo que decide acceder al sistema a intentar realizar la gestión. Únicamente necesitará sus datos credenciales para acceder y saber la cita que quiere modificar.

Una vez accede a la URL del sistema le aparece la pantalla para introducir los datos de usuario. Introduce su usuario y contraseña en los campos dispuestos para ello y pulsa el botón de Aceptar. Una vez el sistema valida los datos introducidos, accede a la aplicación con el perfil del usuario.

En esa pantalla de bienvenida, en el menú superior el usuario observa que hay un punto de menú que se llama "Mis Citas", por lo que pulsa sobre él. El sistema le devuelve al usuario todas sus citas en forma de registros en una tabla.

Cada registro, junto con su información tiene un enlace llamado Cancelar, por lo que el usuario pulsa sobre él. A continuación se le presenta una pantalla modal con un mensaje de aviso que le indica que va a proceder a dar de baja la cita. El usuario pulsa el botón Aceptar para consolidar los cambios. El sistema le devuelve a la pantalla con todas las citas en la que podrá observar que la cita cancelada ya no está registrada. Comprobado este hecho la gestión se ha realizado con éxito por lo que da por concluida su tarea pulsando en el botón "Salir" o "Desconectar" de la parte superior.

Escenario 3: Un especialista quiere consultar su agenda en un momento determinado.

Uno de los trabajadores especialistas necesita consultar su agenda de citas. Para ello necesita sus credenciales para acceder al sistema. Una vez accede a la URL del sistema le aparece la pantalla para introducir los datos de usuario. Introduce su usuario y contraseña en los campos dispuestos para ello y pulsa el botón de Aceptar. Una vez el sistema valida los datos introducidos, accede a la aplicación con el perfil de usuario correspondiente.

En esa pantalla de bienvenida, en el menú superior el usuario observa que hay un punto de menú que se llama "Mis Citas", por lo que pulsa sobre él. El sistema le devuelve al usuario todas sus citas en forma de registros en una tabla, ordenados por fecha.

Cada registro contiene la información correspondiente de la cita, por lo que de un vistazo (con paginación de registros si fuese necesario) el especialista puede consultar todas sus citas y ver por tanto su agenda para los días que necesite.

Flujos de interacción

Los flujos de interacción ayudan a visualizar la estructura general de la aplicación en forma de diagrama.

Flujo de interacción 1 : Login

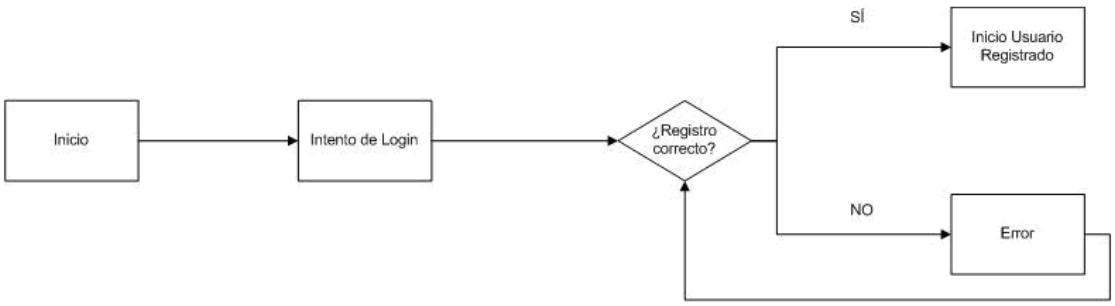


Ilustración 17. Flujo de interacción de Login

En este flujo de interacción se describe la funcionalidad de Login. El usuario accede a la url de inicio de la aplicación web, cumplimenta sus datos de usuario (nombre de usuario y contraseña) y procede al intento de inicio de sesión. Si los datos son correctos el sistema le redirige a la pantalla de inicio como usuario registrado (en función del perfil que tenga, tal y como se verá en el siguiente flujo de interacción) y si no se le mostrará un error.

Flujo de Interacción 2 : Inicio de Usuario Registrado

Este flujo parte de la operación exitosa del flujo anterior. Aunque se describa en tres flujos diferentes en el presente documento, dado su tamaño, se debe entender como uno solo. Desde el inicio del usuario registrado, éste puede tener tres perfiles diferentes: ciudadano, administrador o especialista y en función de ellos podrá realizar diferentes tareas y funcionalidades, tal y como se describe en el flujo. De esta forma, a continuación se incluyen los flujos de los tres perfiles.

Flujo de Interacción 2.1: funcionalidades del usuario Ciudadano.

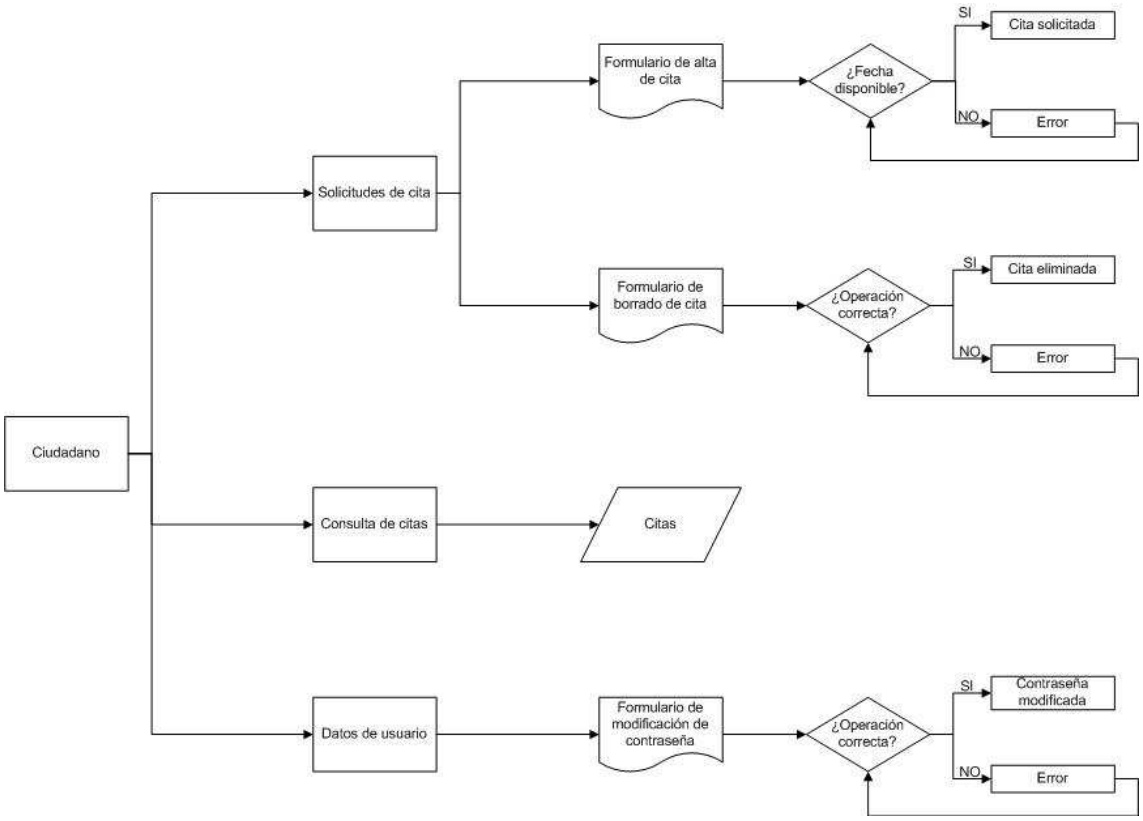


Ilustración 18. Flujo de interacción de funcionalidades del usuario Ciudadano

El usuario ciudadano puede gestionar sus citas mediante solicitudes de alta, baja y consulta de las mismas. Además puede modificar sus datos de usuario, concretamente modificando su contraseña. No se contempla que pueda modificar otro tipo de datos, para ello precisaría acudir presencialmente a uno de los centros.

Flujo de Interacción 2.2: funcionalidades del usuario Administrador.

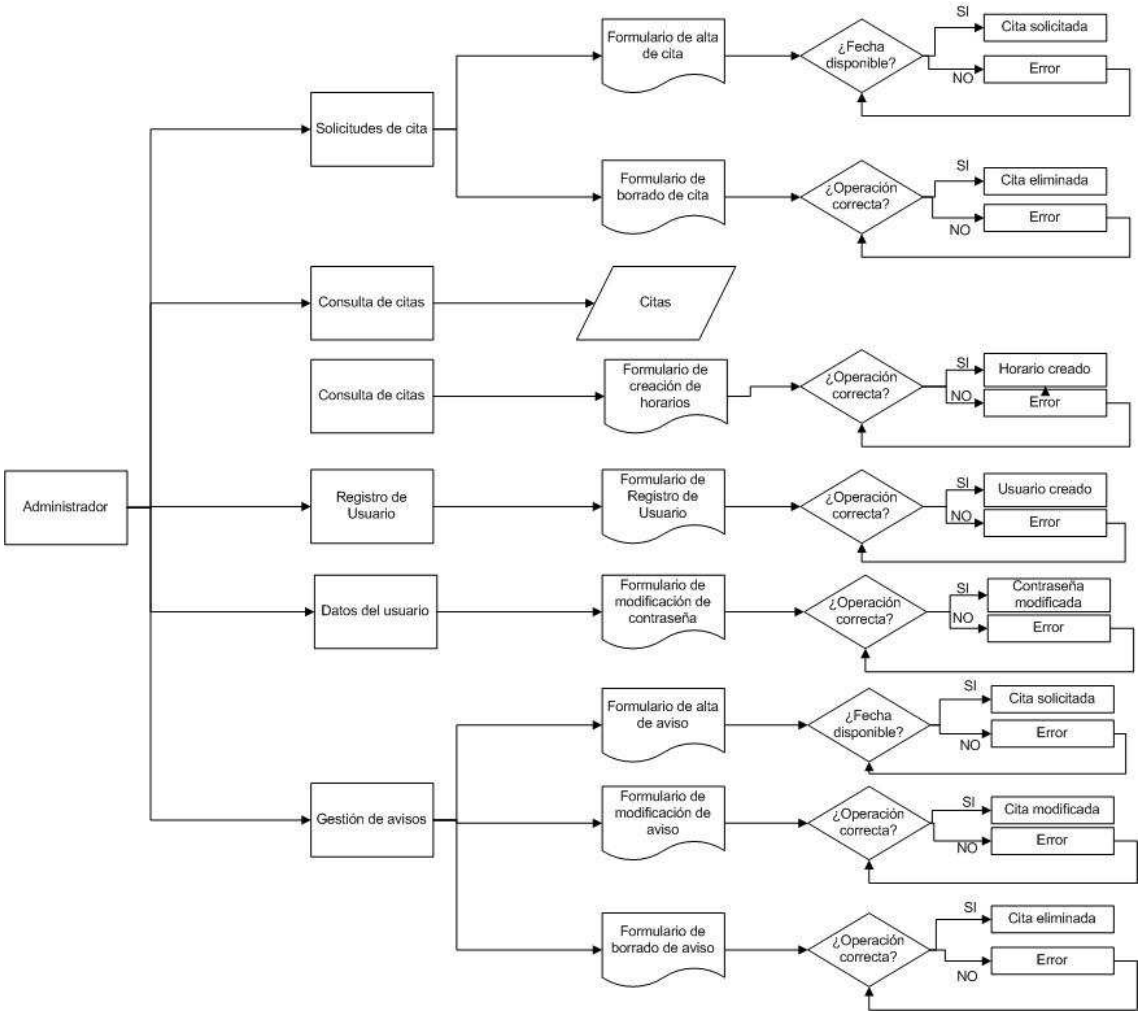


Ilustración 19. Flujo de interacción del usuario Administrador

El usuario administrador puede gestionar las citas mediante solicitudes de alta, baja y consulta de las mismas, aplicando la gestión de alta y baja sobre un usuario ciudadano concreto.

Además, se encarga del registro de usuario pues los usuarios ciudadanos deben acudir presencialmente para llevar a cabo su registro. A su vez este perfil es el encargado de modificar datos personales del ciudadano, como su domicilio, etc.

Por último, es el perfil encargado de gestionar los avisos o novedades que se muestran en la aplicación.

Flujo de Interacción 2.3: funcionalidades del usuario Especialista.

El usuario especialista podrá consultar las citas que tiene a su nombre y modificar su contraseña de usuario.

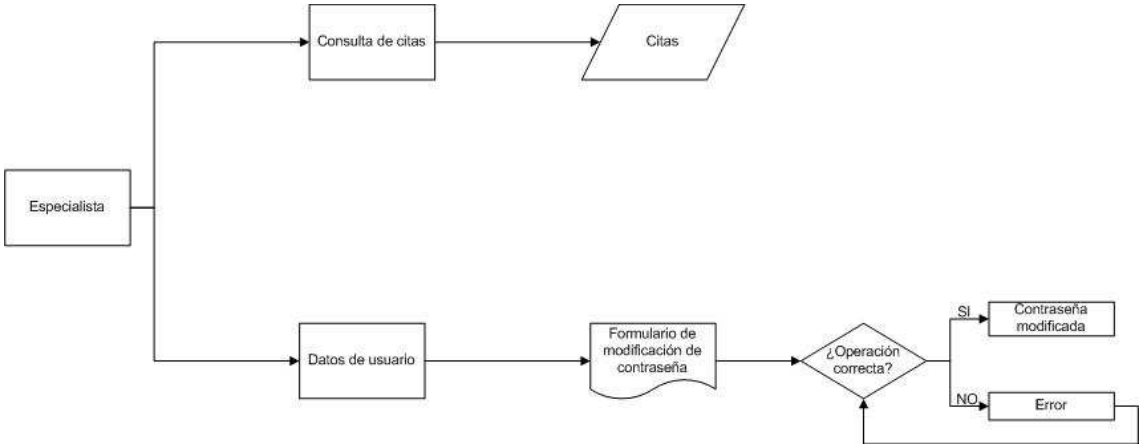


Ilustración 20. Flujo de interacción del usuario Especialista

12. Usabilidad/UX

En el proyecto se ha intentado dotar al sistema de la mejor usabilidad y accesibilidad posibles con el fin de que la experiencia del usuario sea lo mejor posible.

Con el estudio de los usuarios y los contextos de uso, así como con la colaboración de los usuarios en los escenarios de uso, se puede obtener información muy valiosa que se puede aplicar posteriormente en las decisiones de usabilidad y experiencia de usuario de la aplicación web.

Uno de los aspectos importantes que se ha implementado en el sistema, y que era un objetivo a conseguir, es que la aplicación web fuera compatible con diferentes dispositivos y tuviera comportamiento adaptable a diferentes pantallas y resoluciones. Para ello, gracias a la utilización de Bootstrap comentada anteriormente, se ha podido dotar de este comportamiento a los templates HTML.

De esta forma, por ejemplo en un PC, si la ventana del navegador se va haciendo más pequeña, el contenido de la aplicación se adapta a la misma para poder visualizarse con cualquier resolución. En los dispositivos móviles o tabletas, con sus limitaciones de resolución, la aplicación web adapta también su contenido para su correcta visualización.

Destacar también que, tal y como se indica en la sección de 'Guía de usuario', se mejora la usabilidad y experiencia del usuario en la aplicación con aspectos como el implementado en el menú, pues si el usuario accede a través de él a cualquiera de las opciones posibles, ésta se sombrea para que el usuario sepa en todo momento dónde se encuentra en la aplicación.

Ilustración 21. Home de la aplicación en dispositivo Samsung Galaxy S4

En la imagen superior se puede observar cómo se visualizaría la aplicación haciendo más pequeña la ventana de un navegador Google Chrome hasta un tamaño de pantalla igual que un Samsung Galaxy 4 (se ha simulado esta ventana mediante las herramientas de desarrollador del citado navegador).

Para la visualización de las citas, en las que se ha usado una tabla, se ha identificado con la etiqueta 'table responsive', de manera que el contenido de la tabla se adapta a la resolución generando un 'scroll'.

Otro de los aspectos que se ha intentado cuidar es el de la claridad en las acciones que realice el usuario, y que en ningún momento se sienta perdido en la navegación. Por ello, cada perfil de usuario tiene un menú de acciones (también con comportamiento responsive lógicamente), resaltándose cada punto de menú cuando se pasa el ratón por encima. Además en cada punto de la aplicación se acompaña el contenido con un título representativo. Se contempló la posibilidad de incluir migas de pan (breadcumb) pero la aplicación no tiene tanta profundidad en la navegación como para que merezca la pena llevarse a cabo.

A su vez, después de realizar operaciones de guardado o borrado se muestran mensajes en ventanas modales que informan al usuario de sus acciones y sus consecuencias.

Por último, la interfaz implementada en el sistema se ha llevado a cabo con la intención de ser visualmente clara y cómoda, sin texto innecesario y sin demasiados colores que puedan entorpecer la navegación y la experiencia de usuario.

13. Tests

Las pruebas unitarias son una tarea imprescindible para el buen devenir de un proyecto de desarrollo software. Es preciso definir cómo se van a realizar, qué tipo de pruebas se van a llevar a cabo, qué recursos se van a utilizar y cuándo se van a llevar a cabo.

Normalmente, una vez concluido el desarrollo software se preparan una batería de pruebas que permitan verificar la implementación, detectar errores o posibles problemas así como validar los objetivos funcionales fijados previamente. No obstante, también son comunes utilizar pruebas para módulos o funcionalidades concretas durante la fase de implementación, con el objetivo de poder verificar que la funcionalidad es correcta a pequeña escala, sin necesidad de probar el conjunto del código completo.

En lo que a nuestro proyecto se refiere, AngularJS apuesta claramente por el testing y la facilidad de realizar pruebas y test unitarios y funcionales. Para ello, dispone de diferentes herramientas, como son Karma, Jasmine o Protractor [14].

Tras un pequeño estudio e investigación sobre qué herramientas serían las más adecuadas se ha decidido instalar y utilizar Karma y Jasmine porque Protractor parece a priori una solución más compleja (aunque más completa porque aborda el testing de manera end-to-end).

Sobre Karma y Jasmine, son las que Yeoman aborda en su tutorial de inicio (ya que Karma fue creada por AngularJS) y son de las que más referencias de calidad se pueden encontrar en la red.

Antes de definir las funcionalidades y detallar el estudio de pruebas realizado, es interesante indicar que, como se indicó anteriormente, Karma es un test runner creado por AngularJS para la realización de pruebas [15], y Jasmine es un framework de pruebas [16] que se entiende muy bien con Karma, pues éste ofrece herramientas para llamar de manera sencilla a nuestros test Jasmine mediante código. Ambos se complementan mediante el adaptador "karma-jasmine", que sirve para utilizar el framework Jasmine bajo Karma.

Sin entrar en detalle en métodos de instalación, cabe destacar que para este proyecto se han utilizado tests automatizados sencillos con Karma para la creación de avisos de interés y para el login. El resto de funcionalidades han sido testeadas mediante pruebas funcionales por el alumno, al no ser una aplicación excesivamente compleja y extensa. De esta forma, se ha intentado controlar por parte del alumno que la funcionalidad funciona correctamente.

Pruebas unitarias con Karma y Jasmine

Como se indicaba, se ha utilizado Karma para realizar pruebas unitarias de las funcionalidades de login y de creación de avisos de interés. Realmente Jasmine se ha instalado pero no se han utilizado sus funciones 'spy'. Después de instalar Karma, y en nuestro caso utilizando Yeoman, éste genera una carpeta de test en la que se deben incluir todos los test y en el que se crea un fichero que se llama karma.conf.js que es en el que se configura qué frameworks se van a utilizar (por ejemplo Jasmine), qué plugins, qué navegador web o browser o qué ficheros se van a testear (en este proyecto los ubicados dentro de test/spec/**/*.js), entre otros.

Los ficheros de pruebas con Karma tienen una estructura parecida a la siguiente (fichero de test addAviso.js).

```
describe('Controller: AddAvisoCtrl', function () {

  // load the controller's module
  beforeEach(module('tfmgestionCitasApp'));
  var AddAvisoCtrl,
    scope;

  // Initialize the controller and a mock scope
  beforeEach(inject(function ($controller, $rootScope) {
    scope = $rootScope.$new();
    AddAvisoCtrl = $controller('AddAvisoCtrl', {
      $scope: scope
      // place here mocked dependencies
    });
  }));

    it('Comprobar que el scope está definido', function () {
                        expect(scope).toBeDefined();
            });
    it('Añadir nuevos avisos', function () {
                        scope.AniadirAviso('Título Test 1', 'Contenido Test 1');
                        expect(scope.testGuardado).toBeTruthy();
            });
});
```

Se debe incluir un 'describe()' del controlador (en este caso el de creador de avisos de interés), se debe cargar el módulo del controlador, inicializarlo y realizar los test necesarios mediante 'it()'.
Como se puede observar para este test se ha comprobado en primer lugar que el scope está definido `expect(scope).toBeDefined();` y en segundo lugar que se pueden añadir nuevos avisos:

```
it('Añadir nuevos avisos', function () {
                    scope.AniadirAviso('Título Test 1', 'Contenido Test 1');
                    expect(scope.testGuardado).toBeTruthy();
 });
```

En este segundo caso se llama a la función de añadir aviso y se espera que la variable testGuardado asignada al scope sea verdadera. Esta variable está declarada en el fichero controlador de creador de avisos (addAviso.js) y tiene valor true cuando se ha añadido el aviso.

Para ejecutar las pruebas se arranca Karma con Grunt mediante el comando grunt test. Si el resultado es satisfactorio se obtendrá un resultado como el siguiente, el que se informa que ambos 'expect' tienen resultado exitoso.

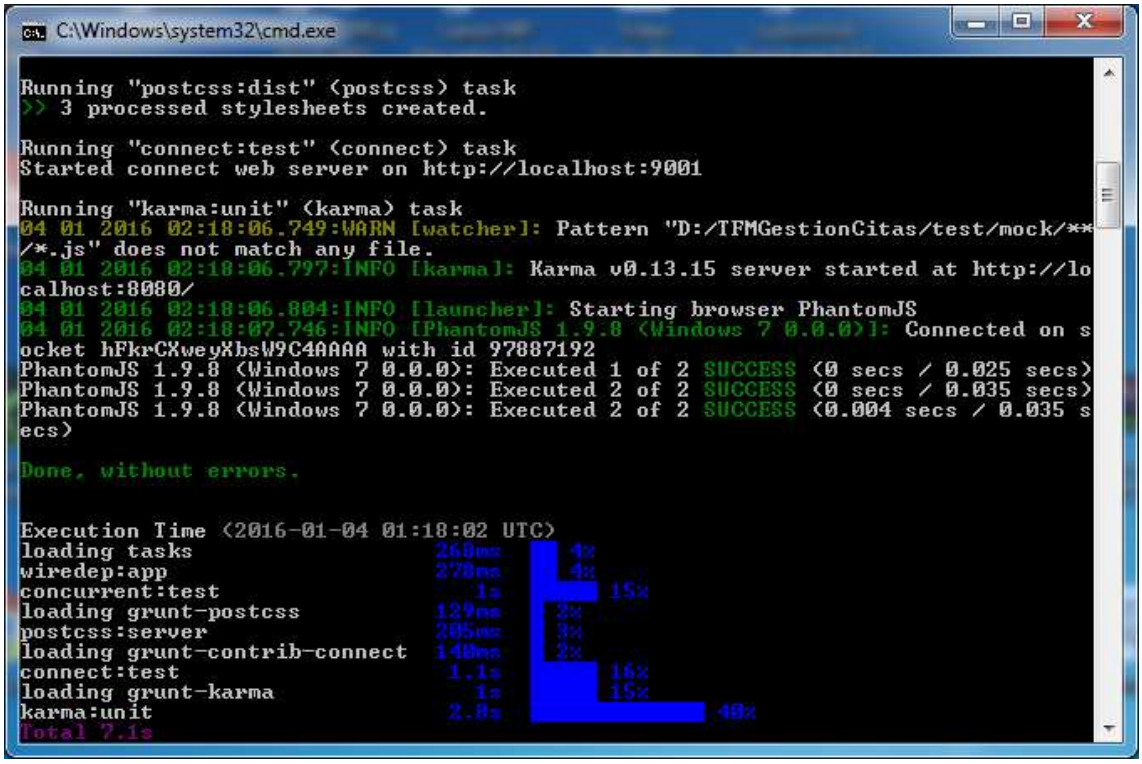


Ilustración 22. Resultado de la ejecución de un test exitoso con Grunt

Si se tienen más ficheros de prueba, realizará el testing automático sobre todos ellos, de manera que se evidencia lo cómodo que puede resultar realizar tests con esta herramienta.

En cuanto a la funcionalidad de Login, se han observado problemas por el uso de Firebase y, aunque la funcionalidad funciona correctamente, el test no verifica el hecho esperado que se ha incluido en el código por parte del alumno.

```
it('Realizar el login correctamente', function () {
        scope.usuario.email = 'a@a.com';
        scope.usuario.password = '123456';
        scope.signIn();
       expect($location.path()).toBe('/misavisos');
});
```

Como se puede observar, en el test se incluyen el email y el password de acceso, que pertenecen a un usuario de perfil especialista y se llama a la función de login. Se espera que el sistema redirija al path '/misavisos' una vez se haya realizado el login. El resultado obtenido no es exitoso pues el uso de las funciones de Firebase (en el que se comprueba en Firebase que existe usuario con ese email y password) no parecen ser compatibles con el test de la manera que se está realizando por el alumno al menos. Obviando esas funciones el 'expect' funciona correctamente.

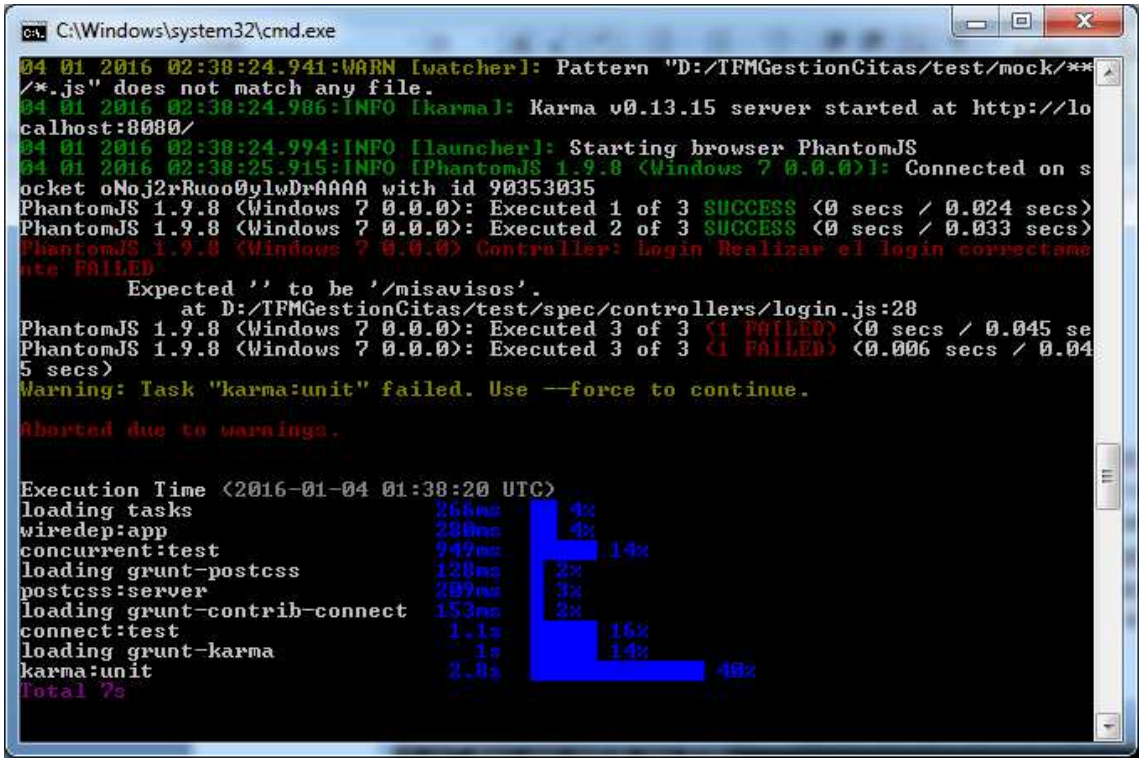


Ilustración 23. Resultado de la ejecución de un test con errores con Grunt

Como se puede observar el tercer test 'expect' no funciona correctamente.

Por ello y por la complejidad, especialmente al principio, que supone realizar los test para todas las funcionalidades se ha decidido probar la aplicación en vivo por parte del alumno, con el objetivo de comprobar su funcionamiento y detectar posibles bugs.

En este sentido la aplicación se ha probado al completo (se podrá observar su funcionamiento especialmente en la guía de usuario del presente documento) y por particularizar en algunas funcionalidades, se ha incidido en la parte de validación y control de duplicidades.

Por ejemplo, en la creación de horarios para las citas se comprueba si un horario se ha dado de alta ya con una fecha y un especialista concretos. Si es así se muestra una pantalla modal informando este hecho.

```
firebaseObj.once("value", function(snapshot){
        snapshot.forEach(function(childSnapshot){
                var data = childSnapshot.val();
                if ((fecha.getTime() == data.fecha.getTime()) && (especialista == data.especialista))
                {
                        duplicado = true;
                }
        }
        )
});

if (duplicado = false){

fb.$push({
        fecha: fecha.getTime(),
        categoria: categoria,
        observaciones: observaciones,
        especialista: especialista.seleccionado,
        disponible: true
}).then(function(ref) {
        console.log(ref);
        $('#newModal').modal();
        //$location.path('/misavisos');
}, function(error) {
        console.log("Error:", error);
});
 }
 else
 {
         $('#editModal').modal();
```

14. Versiones de la aplicación

Se entrega una versión etiquetada en el pie de la aplicación como Versión 0.1.

En una anterior entrega (PEC 3) se realizó una entrega en fase Beta con algunas funcionalidades, pero para esta versión se han completado todas las funcionalidades previstas, siendo las siguientes:

- Guardado en base de datos Firebase de usuarios, citas (horarios) y avisos.

- Distinción de los tres perfiles de usuario: Administrador, Especialista y Ciudadano

- Login de usuario

- Modificación de contraseña de usuario

- Creación de usuario.

- Creación de horario para cita.

- Consulta, creación y cancelación de citas.

- Consulta, creación, modificación y borrado de avisos.

- Menú diferente en función del perfil

- Filtro en las citas: a cada usuario sólo se le mostrarán sus citas automáticamente y en el listado tiene un filtro de búsqueda para ordenar por fecha.

- Filtro en los avisos: sólo se muestran los avisos con estado vigente. A su vez, sólo el usuario con perfil Administrador puede modificar y eliminar avisos. El resto sólo puede consultarlos.

- Validaciones los formularios, bien con validaciones en el propio HTML o validaciones en código javascript.

- Interfaces responsive.

- Logout del sistema

Por último destacar que para las pruebas y desarrollo realizado en local, se han utilizado tres usuarios de prueba y diversa información que se ha ido almacenando mediante la aplicación en la base de datos Firebase.

15. Instrucciones de instalación/implantación

Aunque para anteriores entregas (PECs) el proyecto se subió a la plataforma GitHub (que es muy útil para entornos colaborativos, para compartir el código entre varios desarrollados, realizar anotaciones, modificaciones, etc.) en esta versión final se ha subido el código fuente completo.

Teniendo este código fuente, para poner el proyecto en marcha se precisa tener instalado npm y bower. Este proyecto ha sido generado con Yeoman, que tiene un pequeño tutorial para generar y arrancar el proyecto [17]. No obstante, para arrancar el proyecto únicamente necesitaremos arrancar un terminal de nuestro equipo y ejecutar los siguientes comandos:

$ npm install (para instalar npm)

$ npm install --global bower grunt-cli

Con estos comandos se actualizan npm, Bower y Grunt.

Para ejecutar Grunt, desde la ruta donde esté la aplicación, en la consola/terminal introducimos el comando $ grunt serve que automáticamente desplegará la aplicación, normalmente en el puerto 9090. Además, una vez desplegado, abrirá la pantalla de inicio (la de Login en este caso) en el navegador web por defecto, con lo que ya se estará en disposición de probar la aplicación.

16. Proyección a futuro

La implementación de este proyecto nos permite contar con una aplicación que incluye las funcionalidades básicas para un gestor de citas, en este caso para petición de citas de servicios sociales. A nivel técnico son varias las implementaciones que podrían ser interesantes para la aplicación de cara al futuro. Son las siguientes:

- Actualizar a la última versión de AngularFire.

  Como se indicó en el apartado de tecnologías utilizadas, para esta versión se ha decidido no utilizar la última versión de AngularFire (1.x.x, en el momento de la entrega del proyecto se encuentra por la v1.1.3) por varios motivos que ya se indicaron en dicho apartado.

  Por tanto podría ser interesante actualizar dicha versión. El sitio web oficial de Firebase tiene documentación sobre el impacto que ocasionaría en el código fuente esta actualización y los pasos que se han de seguir [18].

  A nivel práctico con la actualización la aplicación a priori mejoraría los aspectos de sincronización y de gestión, uso y manejo de datos en arrays, objetos y listas.

- En relación al punto anterior, sería muy interesante incluir una API de calendario que se integre perfectamente en la aplicación y mejore la usabilidad. Aunque para esta versión se ha intentado integrar un calendario con la API datetime-picker de bootstrap y AngularJS no ha sido posible que funcionase perfectamente por la incompatibilidad de versiones por lo que podría ser una mejora interesante, especialmente para la creación de horarios por parte de los usuarios con perfil Administrador. La puesta en marcha podría no ser demasiado costosa, existe documentación por la red y en el periodo de este proyecto ya se ha adquirido algo de experiencia intentando implementarlo.

- Aviso por correo electrónico y/o SMS a los usuarios: podría enviarse un correo electrónico y/o un SMS a los usuarios informando de las solicitudes y cancelaciones de sus citas, así como de recordatorios de las próximas citas que tienen. De esta forma tendrían más información fuera de la propia aplicación.
  Para hacer esto posible el servidor donde esté montada la aplicación debería contar con la configuración correcta vía SMTP para enviar correos electrónicos. Para enviar SMS debería ser necesario montar una plataforma que permita el envío de los mismos, así como la contratación de un buen servicio de envío de SMS con alguna operadora que no encarezca la inversión en exceso.

- Implementar un módulo de gestión de usuarios. Se podría implementar una funcionalidad más profunda y sólida en cuanto a la gestión de usuarios, en la que se puedan modificar datos personales del usuario y no sólo su contraseña de acceso.

- En este sentido, a la hora de crear un usuario se podría incluir un generador de contraseñas (se pensó en utilizar 'Math.random()' para este proyecto pero no se incluyó finalmente por algunas de las carencias de seguridad que tiene) con el objetivo de que el usuario administrador no tenga que incluir la contraseña manualmente (con los problemas de seguridad que puede conllevar).

- Mejorar el uso de desplegables. Son varios los desplegables que se utilizan en la aplicación (para elegir especialistas al crear horarios, para que los administradores elijan usuarios y solicitar una cita para ellos, etc.) pero su uso puede no ser demasiado bueno si existen muchos datos. En ese caso debería modificarse por algún otro elemento que permita visualizarlos y seleccionarlos de mejor manera.

- Implementar un módulo de gestión de zonas y sus centros correspondientes. Sería interesante incluir una funcionalidad que permita englobar los servicios en sus zonas del municipio y por tanto en los centros asignados a esas zonas. De esta forma, se podrían almacenar las citas en función de la zona del municipio y el centro en el que se ofrezca el servicio.

- Implementación de indicadores y explotación estadística. Resultaría muy interesante la implementación de indicadores estadísticos que permitan contar con un mayor detalle de las citas, usuarios, etc, según los criterios deseados. Con la explotación de esta información estadística se mejoraría el servicio que se ofrece a los ciudadanos pues se podría conocer, entre otros, el volumen de solicitudes, cuánto tiempo tarda en ocuparse el calendario, etc. Además, en este sentido, el punto anterior sobre las zonas y los centros del municipio podrían ser todavía más interesantes para usos estadísticos, de manera que se pueda conocer el volumen de solicitudes por zonas, categorías por zonas, etc.

  Para implementar este punto sería preciso contemplar los indicadores en base de datos y realizar las consultas correspondientes.

- Reportes de los especialistas. Podría resultar interesante que los especialistas pudiesen realizar un reporte de una cita ya pasada, en el que recojan datos interesantes de lo sucedido en la misma. Para ello sería necesario implementar un formulario que por cada cita permita completar los datos y almacenarlos.

En cuanto a su uso en el futuro, esta aplicación podría ser utilizada en un servicio concreto para la gestión de citas. Si sustituyese a una aplicación que ya ha estado en funcionamiento sería preciso estudiar el impacto que supondría una posible migración, puesta en marcha, formación, etc., teniendo en cuenta las particularidades de dónde se va a implantar la aplicación, pues precisaría un proyecto aparte.

No obstante, se puede destacar que la aplicación se ha desarrollado con cierta flexibilidad para que, con no demasiado esfuerzo de programación, se pueda adaptar a otro tipo de servicios de gestión tales como:

- Solicitud de reserva de pistas deportivas
- Solicitud de inscripción en diferentes actividades (culturales, educativas, etc.).
- Gestión de atención pública de algún servicio como por ejemplo un servicio que permita obtener un plano gracias al Servicio de Información Geográfica, un servicio que permita realizar una consulta al departamento de Urbanismo, una consulta al departamento de Consumo, una solicitud de cita para fines educativos, y un largo etcétera.

17. Conclusiones

Las conclusiones que se pueden extraer de la elaboración de este proyecto son muy positivas.

El planteamiento personal inicial para este proyecto era el de poner en práctica los conocimientos aprendidos a lo largo de estos últimos años de máster y poder realizar un buen trabajo en este trabajo fin de máster.

Además, mi intención era la de implementar un sistema que, aunque fuese en una versión no excesivamente madura y completa, pudiera dar una solución a algún problema o carencia que hubiese detectado, en cualquier ámbito, personal o laboral.

Por último, sentía la necesidad, y esta era una gran oportunidad, de iniciarme y aprender conceptos y tecnologías nuevas que me pudiesen servir en un futuro, fuese cual fuese también el ámbito en el que lo pudiese aplicar. No obstante, no negaré los problemas que me han surgido estos meses de trabajo en los que la curva de aprendizaje de las herramientas utilizadas ha sido compleja al principio y por el poco tiempo disponible para llevar a cabo el proyecto, por circunstancias no sólo de calendario del proyecto sino también personales. No obstante, he sido capaz de solventar y solucionar estos problemas con esfuerzo, constancia y aprendiendo nuevos aspectos que he podido aplicar.

Finalmente, como decía, las conclusiones son muy buenas porque he cumplido los objetivos inmediatos que pretendía, realizando un proyecto desde cero con una idea inicial propia, que considero puede ser la base para ser mejorada en un futuro y aplicada en un ámbito interesante, como es el de mejorar los servicios a ciudadanos. A su vez he utilizado sistemas y herramientas actuales, que son el presente de las aplicaciones web, y que me han permitido ponerme al día en este aspecto, hecho que no desaprovecharé para el futuro.

Anexo 1. Entregables del proyecto

En un proyecto web se ha de entregar el material correspondiente a la finalización del mismo. Este material debe incluir la documentación oficial del proyecto realizada y según el caso los fuentes de la aplicación, normalmente compartidos mediante repositorios como CVS, Subversion (SVN) o similares.

Se ha entregado el código fuente completo, que contiene todos los ficheros de configuración, templates, código javascript, etc., aunque cabe destacar que como aprendizaje, en anteriores entregas, se utilizó la plataforma Github, subiendo el proyecto que por entonces estaba realizado, mediante comandos siguiendo algún tutorial [19].

Con el objetivo de detallar el entregable de código fuente y para que sirva de conocimiento previo al anexo siguiente, en el que se detallan los aspectos más importantes del código y la funcionalidad de la aplicación, a continuación se indica la estructura de ficheros de la aplicación y los aspectos mas relevantes.

Como se ha indicado a lo largo del proyecto, para generar la aplicación se ha utilizado Yeoman. Mediante un wizard con diversas opciones Yeoman genera el esqueleto de la aplicación con las características elegidas. El esqueleto de una aplicación Angular estándar creada con Yeoman sería el de la ilustración que se muestra a continuación.

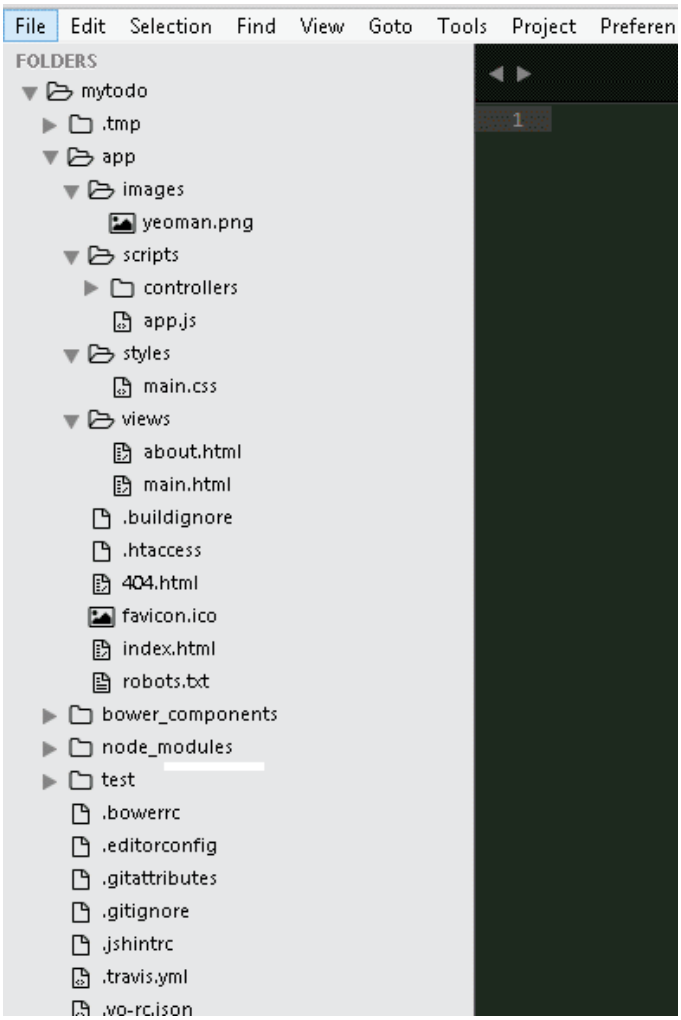


Ilustración 24. Esqueleto de una aplicación estándar generada con Yeoman

Tal y como se puede observar en el código fuente entregado, la estructura es idéntica, sólo que las ramas de cada carpeta están lógicamente más completas, con las funcionalidades de la aplicación.

Siguiendo esa estructura los elementos importantes son:

- carpeta app: carpeta raíz de la aplicación web
    - index.html: el HTML base para nuestra aplicación Angular
    - 404.html, favicon.ico y robots.txt: ficheros para error, para el icono del navegador o para buscadores que crea automáticamente.
    - carpeta scripts: en la que se incluye prácticamente el código de la funcionalidad de la aplicación, es decir, los controladores (controllers), los servicios (services) y el fichero app.js (que contiene el código principal de nuestra aplicación Angular).
    - carpeta styles: en el que se incluyen los ficheros CSS.

    - carpeta views: en el que se incluyen los ficheros HTML de la aplicación

- bower_components y bower.json: son las dependencias Javascript/web instaladas por Bower.
- Gruntfile.js, package.json y node_modules: son ficheros de configuración y dependencias que necesita Grunt para las tareas automatizadas.
- test: carpeta para incluir las funcionalidades de tests de la aplicación.

Además del código fuente se entrega la documentación o memoria realizada para el proyecto, un documento de autoevaluación del alumno, una presentación en formato Powerpoint y un vídeo resumen explicativo del proyecto con una demo de la aplicación.

Anexo 2. Código fuente (extractos)

En esta sección se detallan las funcionalidades más relevantes con su correspondiente código, con el objetivo de conocer en mayor medida cómo se ha realizado la implementación de la aplicación. Se detallarán aspectos de almacenamiento en base de datos, funcionalidades concretas que sean relevantes y aspectos importantes de implementación con Angular. Se utilizan para esta sección funcionalidades sencillas que permitan explicar de manera clara y comprensible las decisiones tomadas y los recursos utilizados para el desarrollo.

Aspectos, ficheros e implicaciones importantes de AngularJS

En primer lugar es preciso entender cómo se estructura AngularJS. En el anexo anterior de Entregables del proyecto se ha hablado de la estructura a nivel de directorios para una aplicación web Angular, por lo que a la hora de trabajar e implementar código los ficheros imprescindibles de Angular deben estar bien codificados y conectados para un funcionamiento correcto del sistema. A continuación se detallan algunos aspectos importantes de estos ficheros para aclarar este punto.

Fichero index.html

En este fichero se define para nuestra aplicación el menú superior de navegación para cada usuario. En función del perfil del usuario se le mostrarán unos puntos de menú u otros, gracias a la directiva 'ng-hide' que oculta partes del menú según las variables de $rootScope (variables globales de la aplicación) coincidan o no.

No obstante, una de las implicaciones que tiene este fichero respecto a Angular es que es preciso incluir las dependencias de Bower en este fichero, así como todos los scripts javascript que se vayan a utilizar. Si no están definidos en este fichero no funcionarán correctamente.

Fichero bower.json

Como se ha indicado anteriormente, en el fichero 'index.html' se incluyen las dependencias de Bower pero es en el fichero 'bower.json' donde se definen las dependencias y las versiones de las mismas. Por ejemplo, y tal como se ha indicado a lo largo del documento, entre otros, se está utilizando la versión 1.4.8 de Angular y la versión 0.9.0 de AngularFire.

Fichero app.js

Otro de los ficheros imprescindibles para una aplicación Angular. En él se deben definir los módulos de los controladores. Además se configura la gestión de routing con '$routeProvider', es decir, la navegabilidad de la aplicación, de forma que cuando la aplicación tenga que redirigirse a un path concreto, se le indica en este fichero qué template HTML debe mostrar y qué controlador (controller) gestiona su uso. Por ejemplo para redirigir a la funcionalidad de consulta de citas, la codificación es la siguiente (no se incluye el código completo de la función).

```
tfmgestionCitasApp.config(['$routeProvider',
 function($routeProvider) {
  $routeProvider
   .when('/crearcita', {
    templateUrl: 'views/crearcita.html',
                    controller: 'MisCitas',
                    controllerAs:'miscitas'
   })
```

Por último se define el método 'run()' con el que se llama al service de login, que se detalla a continuación.

Services

Los services son pequeñas fábricas de funciones y objetos que se comunican con el servidor para enviar y obtener información que después será tratada por los controllers para mostrarla en las vistas.
Con el objetivo de probar y aprender el funcionamiento de los services se ha creado uno con el que gestionar las acciones de usuario: login, cambiar contraseña, crear usuario y salir de la aplicación. Para probar diferentes alternativas, finalmente, el método de login como tal se ha dejado en el fichero login.js pero al resto de métodos sí se les llama mediante el service. Por ejemplo para el método de cambio de contraseña se realiza la llamada de la siguiente manera:

```
 $scope.cambiarContrasenia = function() {
                    servicioLogin.cambiarContrasenia($scope.usuario.password, $scope.usuario.nuevaPassword,
$scope.usuario.nuevaPassword2);
          };
```

Como se indicaba los services no son más que pequeñas fábricas de funciones y objetos, por tanto en el fichero services.js se ha definido la factoría con las funciones correspondientes. Por ejemplo la definición de la factory del service y el método de cambiar contraseña se codifica como se ve a continuación (sólo se incluye la llamada a la función de cambiarContrasenia, no la codificación completa).

```
var tfmgestionCitasServicios = angular.module('tfmgestionCitasServicios', ['firebase']);

tfmgestionCitasServicios.factory('servicioLogin', ['$rootScope', '$firebase','$firebaseAuth','$location',
  function($rootScope, $firebase, $firebaseAuth, $location) {
    var loginObj = null;
    var refUsuarios = new Firebase ("https://tfmgestioncitas.firebaseio.com/Usuarios");
    var usuarios = $firebase(refUsuarios);
    $rootScope.usuario = null;
    $rootScope.loginError = null;
    $rootScope.change = null;
    return {
      init: function() {
        var databaseRef = new Firebase('https://tfmgestioncitas.firebaseio.com');
        return loginObj = $firebaseAuth(databaseRef);
      },

                              cambiarContrasenia: function(oldPass, nuevaPass, nuevaPass2){ …… }
```

Controllers

Los controllers contienen el código con la lógica que comunica el modelo con las vistas, es decir, es en cada uno de ellos en los que se ha codificado la implementación de la funcionalidad de la aplicación, así como la inicialización y modificación de la información que contienen los Scopes. Por tanto, para cada funcionalidad global se ha creado un nuevo fichero controller javascript. Naturalmente, es evidente que el núcleo importante y funcional de la aplicación se encuentra en estos ficheros. Como se indicaba anteriormente, una vez definido el módulo del controlador se ha de incluir en el fichero 'app.js' para su correcto funcionamiento.

Uso de la base de datos Firebase

La codificación de la implementación conlleva naturalmente el uso de base de datos, en este caso Firebase. A continuación se detallan los aspectos más importantes relacionados con la base de datos en cuanto a codificación se refiere.

Guardar un registro (hijo) en la base de datos Firebase.

En primer lugar cabe destacar la manera en la que se almacena la información en la base de datos Firebase. Al tener acceso vía web, la manipulación de datos es sencilla, mediante un árbol con nodos, siguiendo el concepto de base de datos NoSQL. Para guardar los atributos de un "registro" o hijo en uno de los nodos padre (por ejemplo en el nodo de Usuarios), Firebase nos ofrece diferentes alternativas (funciones) [20]. La elegida en este proyecto es la función 'push()'. Esta función almacena cada hijo con una clave única (ID cronológico como le llama Firebase) que se genera en el momento de realizar el guardado. Este ID o clave única que no es más que una cadena o serie de letras y números generados por un algoritmo en el que se

tiene en cuenta, para evitar conflictos y duplicidades, el instante justo en el que se realiza el guardado (horas, minutos y segundos).

Siguiendo el ejemplo del nodo Usuarios citado, para almacenar un usuario mediante la funcionalidad de 'CrearUsuario' el código es el siguiente.

```
crearUsuario: function(email, pass, nombre, apellido1, apellido2, docIdent, domicilio, perfil) {
                $rootScope.loginError = null;
                if (!apellido2)
                {
                        apellido2 = null;
                }
                var fb = new Firebase("https://tfmgestioncitas.firebaseio.com/Usuarios/");
                var usuario = $firebase(fb);
                usuario.$push({
                        email: email,
                        password: pass,
                        nombre: nombre,
                        apellido1: apellido1,
                        apellido2: apellido2,
                        docIdent: docIdent,
                        domicilio: domicilio,
                        estado: true,
                        perfil: perfil
                }).then(function(ref) {
                        console.log(ref.key()); // bar
                //$('#editModal').modal('hide');
                }, function(error) {
                        console.log("Error:", error);
                        console.error('Fallo de creación de usuario: ', error);
                        $rootScope.loginError = error;
                });
        }
```

Para almacenar datos en la base de datos es fundamental, en primer lugar, realizar la conexión con la misma mediante la línea:

```
var fb = new Firebase("https://tfmgestioncitas.firebaseio.com/Usuarios/");
```

En ella se indica la URL a la base de datos creada en Firebase, en este caso, con el nombre tfmgestioncitas. Esta base de datos se ha creado bajo una cuenta Google personal del alumno, que es lo que ha precisado para poder crear una base de datos en Firebase. Además de la URL a la base de datos, se añade el nodo sobre el que se va a almacenar un registro, es decir, sobre el nodo Usuarios (ya creado previamente en la base de datos).

En la variable 'fb' se guarda el objeto de la conexión al nodo Usuarios y con la línea

```
var usuario = $firebase(fb);
```

se tiene una variable 'usuario' sobre la que se hará el 'push()' para guardar el registro. Dentro del push, tal y como se observa en el código, se añaden todos los atributos que compondrán el salvado del usuario. Se guardan como si fuera una especie de lista en la que se indica el campo seguido de dos puntos y el valor. Por ejemplo 'email:email', donde el segundo email es el valor pasado por parámetro en la función.

Actualizar los datos de un registro (hijo) del nodo.

Para actualizar los datos de un hijo en un nodo de la base de datos el procedimiento es parecido de guardado normal, no obstante utilizamos la función 'update()'. En nuestra aplicación por ejemplo se pueden realizar modificaciones de Avisos ya creados. Una vez seleccionado el aviso, es decir, indirectamente se puede recuperar el id de ese hijo, se pulsa en el botón 'Modificar' y se abre una ventana modal con los datos de ese aviso para realizar las modificaciones que se deseen. Si se aceptan las modificaciones, el código del salvado de datos sería el que se muestra a continuación.

```
$scope.modificarAviso = function(id) {
  console.log(id);

  var firebaseObj = new Firebase("https://tfmgestioncitas.firebaseio.com/Avisos/" + id);

  var aviso = $firebase(firebaseObj);

  $scope.avisoAActualizar = aviso.$asObject();

  $('#editModal').modal();     // triggers the modal pop up
}

$scope.update = function() {
        console.log($scope.avisoAActualizar.$id);
        var fb = new Firebase("https://tfmgestioncitas.firebaseio.com/Avisos/" + $scope.avisoAActualizar.$id);
        var aviso = $firebase(fb);
        aviso.$update({
                titulo: $scope.avisoAActualizar.titulo,
                contenido: $scope.avisoAActualizar.contenido,
                usuAdmin: $rootScope.usuario,
                vigente: $scope.avisoAActualizar.vigente
        }).then(function(ref) {
                console.log(ref.key()); // bar
                $('#editModal').modal('hide');
        }, function(error) {
                console.log("Error:", error);
        });
`
```

En esta parte de código se observan aspectos interesantes que se pueden citar. En primer lugar se declara la función 'modificarAviso' mediante el objeto $scope. Este objeto es fundamental en Angular pues es donde se define la funcionalidad de la aplicación, los métodos en los controladores y las propiedades en las vistas. Es como

una variable global (similar a $rootScope, que se citó anteriormente) que se puede instanciar o declarar en cualquier punto de la aplicación y que se puede visualizar en cualquier vista.

De esta forma, según el código se tiene un objeto aviso que no es más que el aviso recuperado de la base de datos pasándole el id del aviso que se ha seleccionado para modificar por pantalla.

Para actualizar se tiene la función 'update' con el objeto $scope que a su vez lo que hará es guardar el aviso con los datos a modificar ($scope.avisoAActualizar.$id) en la base de datos mediante la función $update(), pasándole los atributos correspondientes, de manera análoga al guardado con $push(). Por ejemplo, si el usuario ha modificado el contenido del aviso, en el atributo contenido se guardará ese nuevo contenido, identificado como $scope.avisoAActualizar.contenido, que es como se define en el HTML. ¿Y cómo se da esa concordancia entre el HTML y el archivo javascript?
Para ello se usa la directiva ng-model en el HTML, que no es más que una directiva de Angular que asocia el campo (en este caso un textarea) para que se le pueda llamar desde javascript. De esta forma al campo textarea contenido tendrá la siguiente codificación:

```
<textarea class="form-control" id="message-text" ngmodel="avisoAActualizar.contenido"></textarea>
```

Eliminar un registro del árbol

Siguiendo con el ejemplo de los avisos, es posible que sea preciso eliminar un aviso (hijo) de los avisos. Una vez elegido el aviso a borrar y aceptado el borrado, en este caso se tendría el código siguiente:

```
$scope.borrarAviso = function() {
    var fb = new Firebase("https://tfmgestioncitas.firebaseio.com/Avisos/" + $scope.AvisoABorrar.$id);
    var aviso = $firebase(fb);
    aviso.$remove().then(function(ref) {
        $('#deleteModal').modal('hide');
    }, function(error) {
        console.log("Error:", error);
    });
}
```

En este caso el usuario que quiere borrar un aviso pulsa sobre el botón Borrar del aviso, le aparece una ventana modal con un mensaje de confirmación y si finalmente desea borrarlo pulsa el botón de aceptar el borrado. Como se tiene el aviso a borrar ($scope.AvisoABorrar.$id), para borrarlo definitivamente se usa la función $remove(). Automáticamente queda eliminado de la base de datos. Posteriormente cerramos la ventana modal ($('#deleteModal').modal('hide')).

Funcionalidades concretas

Para repasar las funcionalidades más importantes de la aplicación, a continuación se detalla alguna de ellas.

Funcionalidad de Login

Es la funcionalidad con la que se accede a la aplicación. Firebase tiene funciones en su API (por ejemplo mediante $firebaseAuth) que permiten realizar el login a la aplicación, entre otras formas, mediante email y contraseña, garantizando además seguridad en la autenticación.

No obstante, en el caso de esta aplicación, como los usuarios tienen un atributo de perfil que nos interesa conocer, para realizar el login se ha implementado un método propio en el que el sistema comprueba que el email y contraseña introducidos en el formulario coincidan con alguno de los usuarios de la base de datos, recuperando posteriormente el perfil del usuario en caso de que exista coincidencia. Este perfil se utiliza en las variables $rootScope con el objetivo de pintar un menú diferente en cada caso.

Para finalizar se redirige la navegación a la pantalla de avisos, que sería la primera que ve el usuario después de logarse. En caso de no haberse encontrado usuarios coincidentes se informa al usuario de este hecho con un alert.

Uno de los aspectos interesantes de esta funcionalidad es la del uso de la función 'snapshot' y en concreto del método ‘forEach’. Gracias a ello se consigue recorrer los registros de cada nodo y en la variable 'data' guardar todos sus valores. De esta forma se pueden comparar los valores introducidos por pantalla en el HTML con los almacenados en base de datos. En caso de coincidencia se guarda el email del usuario en una variable de $rootScope (sirve para mostrarlo en el menú superior) y se hace uso de ‘localStorage’ que es un método de HTML5 que permite guardar información en local (concretamente en forma de cookies) a través del navegador web, de manera que se guardará ahí también el email. Será una manera de tener "en sesión" el usuario. Todas estas variables de $rootScope y localStorage serán eliminadas o inicializadas al realizar el logout del sistema.

```
$scope.signIn = function(e){
 $scope.testEncontrado = false;
  var email = $scope.usuario.email;
  var password = $scope.usuario.password;
  var firebaseObj = new Firebase("https://tfmgestioncitas.firebaseio.com/Usuarios/");
        firebaseObj.once("value", function(snapshot){
                snapshot.forEach(function(childSnapshot){
                        var data = childSnapshot.val();
                        if ((email === data.email) && (password === data.password)){
                                $scope.testEncontrado = true;
                                $rootScope.usuario = email;
                                 localStorage.setItem("userEmail", email);
                                 if (data.perfil == 'Especialista'){
                                        $rootScope.especialista = true;
                                        $rootScope.ciudadano = false;
                                        $rootScope.administrador = false;
                                         $rootScope.nombreEspecialista = data.nombre + " " + data.apellido1;
                                }
                                else if (data.perfil == 'Ciudadano'){
                                        $rootScope.especialista = false;
                                        $rootScope.ciudadano = true;
                                        $rootScope.administrador = false;
                                }
                                else{
                                        $rootScope.especialista = false;
                                        $rootScope.ciudadano = false;
                                        $rootScope.administrador = true;
                                }
                                $location.path('/misavisos');
                                $scope.$apply();
                                console.log('Autenticación realizada con éxito');
                                return true;
                        }
                 });
                 if ($scope.testEncontrado === false) {
                        alert("Los datos introducidos no coinciden con ningún usuario");
                }
        });
}
```

Funcionalidad de creación de horario para cita

Para tener horarios disponibles para solicitar citas los administradores deben crear estos horarios. Esta funcionalidad tiene un comportamiento análogo a otras funcionalidades de creación, como la de creación de usuario, solicitud de cita, etc. En definitiva, se recogen unos datos de un formulario, se realizan validaciones en el HTML y posteriormente, si todo es correcto, se almacena la información en la base de datos.

```
'use strict';

 var controladorHorarios = angular.module('controladorHorarios', []);

controladorHorarios.controller('Horarios', [ '$scope', '$location', '$firebase', '$rootScope', function ($scope,
$location, $firebase, $rootScope) {
            if($rootScope.usuario==null){
      $location.path('/login');
   }
    var firebaseObj = new Firebase("https://tfmgestioncitas.firebaseio.com/Usuarios/");
             var sync = $firebase(firebaseObj);
            $scope.especialistas = sync.$asArray();

            $scope.crearHorario = function() {
            var duplicado = false;
             var especialista = $scope.especialistas
             var observaciones = $scope.horario.observaciones;
(.....)
             var fecha = $scope.horario.fecha;
             var hora = $scope.horario.hora;
             fecha.setHours(hora.getHours());
             fecha.setMinutes(hora.getMinutes());
              var fechaFormateada = fecha.getTime();
             var firebaseObj = new Firebase("https://tfmgestioncitas.firebaseio.com/Citas");
              var fb = $firebase(firebaseObj);
                        firebaseObj.once("value", function(snapshot){
                        snapshot.forEach(function(childSnapshot){
                                  var data = childSnapshot.val();
                                  if ((fechaFormateada == data.fecha) && (especialista.seleccionado == data.especialista)) {
                                            duplicado = true;
                                  }
                        })
             });
             if (duplicado == false){
                        fb.$push({
                                  fecha: fecha.getTime(),
                                  categoria: categoria,
                                  observaciones: observaciones,
                                  especialista: especialista.seleccionado,
                                  disponible: true
                        }).then(function(ref) {
                                  console.log(ref);
                                  $('#newModal').modal();
(.....)
             }
             else{
                        $('#editModal').modal();
             }
};
```

Cabe destacar que en la anterior imagen no se incluye el código completo del fichero pero como se puede observar en primer lugar se define el módulo del controlador. Seguidamente se define el controlador con los parámetros que se precisen y que se vayan a utilizar en esta funcionalidad. Antes de comenzar con la lógica concreta de la funcionalidad se verifica que si no hay usuario guardado en el $rootScope se redirija al usuario a la pantalla de login.

A continuación se comienza con la lógica de la funcionalidad en sí.

Antes de llamar a la función de crear el horario se guardan los usuarios en un array, para posteriormente obtener los perfiles especialistas de los usuarios que tengan dicho perfil. Finalmente, en la función de crear el horario, se guardan en distintas variables los datos que se recogen del formulario HTML y se llama a la función 'push()' para almacenar esa información en un nuevo registro, aunque previamente se habrá comprobado que dicho horario (con esa fecha y especialista introducidos) no exista ya, porque en caso de existir no pasará por el guardado de base de datos con 'push()' y abrirá una ventana modal informando de dicha duplicidad.

Aspectos importantes de los templates de HTML

Algunos de los aspectos importantes de las vistas HTML se han comentado en los ejemplos anteriores, tales como validaciones, directivas interesantes, etc.

En cuanto a las validaciones, en los formularios de las plantillas HTML se han utilizado validaciones que permiten evaluar si el campo es requerido o no, longitudes mínimas para los campos, mostrar elementos o deshabilitar botones en función de algún criterio, entre otros. También se han utilizado tablas 'responsive' con filtros de búsqueda y ordenación.

A continuación se incluye el código de cada uno de estos casos.

Validaciones

Angular contiene validaciones por defecto para algunos campos, como por ejemplo el campo de tipo email. Si queremos validar que un campo email esté bien formado la codificación sería la siguiente (este caso se puede encontrar en la plantilla crearusuario.html).

```
<div class="form-group" ng-class="{ 'has-error' : loginForm.email.$invalid }">
  <label>Email</label>
  <input type="email" name="email" class="form-control" ng-model="usuario.email">
  <p class="help-block" ng-show="loginForm.email.$invalid">Introduzca un email válido.</p>
</div>
```

Cabe destacar el usuario de la directiva 'ng-class' para definir si el email está bien formado, así como el bloque de ayuda con la directiva 'ng-show' en la que se indica un mensaje de ayuda si el email no está bien formado.

Otros validaciones interesantes son las de campos requeridos. En esta misma plantilla crearusuario.html se encuentra que el campo de primer apellido que precisa ser obligatorio. La codificación sería la siguiente.

```
 <div class="form-group" ng-class="{ 'has-error' : loginForm.apellido1.$error.required }">
    <label>Primer apellido</label>
    <input type="apellido1" placeholder="Primer Apellido" name="apellido1" class="form-control" ng-
          model="usuario.apellido1" required>
    <p class="help-block" ng-show="loginForm.apellido1.$error.required">El primer apellido es
obligatorio</p>
   </div>
```

Una validación similar es la que evalúa si el campo tiene la longitud mínima para darlo por válida. Un ejemplo de codificación con el campo contraseña (password) es la siguiente (plantilla crearusuario.html):

```
<div class="form-group" ng-class="{ 'has-error' : loginForm.password.$invalid }">
    <label>Password</label>
    <input type="password" name="password" class="form-control" ng-model="usuario.password" ng-
          minlength="6">
    <p class="help-block" ng-show="loginForm.password.$error.minlength">La longitud mínima de la
contraseña es 6 caracteres</p>
   </div>
```

La codificación es muy similar a los casos anteriores.

Otro recurso interesante es el de la directiva 'ng-disabled' con la que se puede, por ejemplo, deshabilitar un botón si no se cumplen ciertos criterios. Siguiendo con la plantilla de crearusuario.html, el botón de Crear se deshabilita si los campos obligatorios no están cumplimentados. Es una manera de orientar al usuario, evitando que pulse el botón antes de rellenar todos los datos, haciéndole ver que no está habilitado porque falta algún campo por cumplimentar.
La codificación sería la siguiente.

```
<button ng-disabled="!usuario.email || !usuario.password || !usuario.nombre || !usuario.apellido1"
type="button" ng-click="crearUsuario()" class="btn btn-lg btn-primary btn-block">Crear</button>
```

Tablas

Se ha utilizado una tabla 'responsive' para mostrar las citas de los usuarios. Para darle algo más de funcionalidad a la tabla se han utilizado filtros de búsqueda y de ordenación. Por ejemplo, los datos se pueden ordenar por fecha, de manera que con la directiva 'ng-click' se llama a una función de ordenación pasando el campo por el que queremos ordenar.

```
<a href="" ng-click="ordenarPor('fecha')">
```

Para que esta función funcione correctamente, en el fichero javascript (miscitas.js) hay que implementar la función para que ordene los datos

```
$scope.ordenarPor = function(orden) {
        $scope.ordenSeleccionado = orden;
};
```

Posteriormente, en la tabla se aplican todos los filtros necesarios. El 'body' de la tabla de las citas es el siguiente.

```
<tbody>
 <tr ng-repeat="cita in citas | orderBy:ordenSeleccionado | filter:cita.fecha" ng-show="cita.usuarioCita
    === usuario || cita.especialista === nombreEspecialista || administrador === true">
  <td>{{cita.fecha | date:'fullDate'}} a las {{cita.fecha | date:'HH:mm'}}h</td>
  <td>{{cita.usuarioCita}}</td>
  <td>{{cita.especialista | uppercase}}</td>
  <td>{{cita.observaciones}}</td>
  <td><a class="btn btn-danger" ng-click="confirmarCancelacion(cita.$id)" ng-hide="especialista===true"
    data-target="#deleteModal">Cancelar Cita</a></td>
 </tr>
</tbody>
```

Se utiliza la directiva 'ng-repeat' para recorrer todos los datos que se obtengan de las citas. Además se aplica el filtro de ordenación (orderBy:ordenSeleccionado) y un filtro de búsqueda por fecha que se ha definido anteriormente en la directiva 'ng-model'.
En función del perfil con el que el usuario esté logado se muestra la tabla con unos datos u otros. Igualmente el botón de cancelar cita, para el usuario especialista no se muestra.
Por último, destacar que en la fila de fecha, como ésta se guarda en Firebase como un número (función getTime()), es en esta plantilla donde se "formatea" para darle un valor visualmente entendible gracias a las funciones que nos aporta AngularJS.

Anexo 3. Guía de usuario

En este apartado se cuenta paso a paso como utilizar el servicio. Se acompaña de ilustraciones con las que guiar la información. Tiene la intención de servir como manual de usuario.

Acceso al sistema (para todos los perfiles)

En primer lugar se debe acceder a la URL del sistema. En el caso que nos ocupa, al tenerlo en local, y según la configuración establecida, la URL es http://localhost:9000/#/login

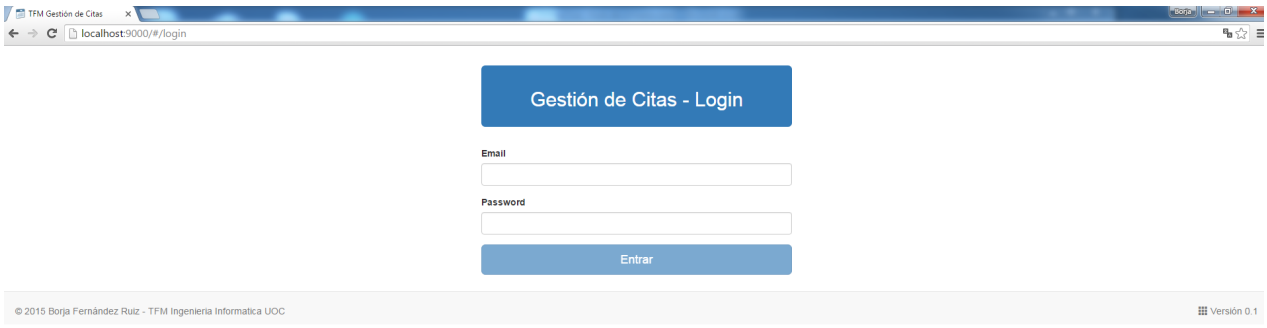


Ilustración 25. Pantalla de Login de la aplicación

En esta pantalla de inicio se deben introducir los datos de usuario con un formato correcto para acceder al sistema.

Ilustración 26. Pantalla de Login con errores

En caso de introducir un email con un formato no válido o una contraseña con una longitud menor de 6 caracteres o dígitos el botón de Entrar no se activará y se mostrarán mensajes de error debajo de campo. A su vez si los datos introducidos no corresponden con ningún usuario se mostrará una ventana de alerta avisando de este hecho.

Gestión de avisos de interés (sólo para perfil Administrador)

Si se accede al sistema con perfil Administrador se puede observar el menú superior con todas las tareas disponibles y en la zona central del contenido, los avisos de interés que estén vigentes, con la posibilidad de modificarlos o eliminarlos.

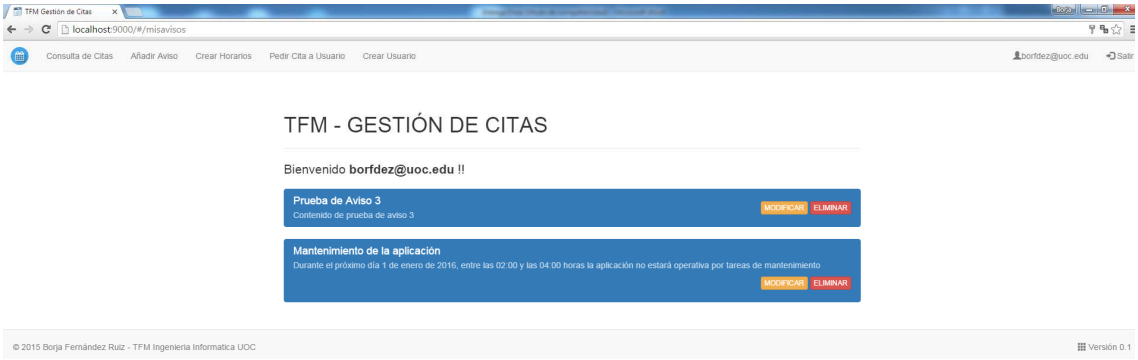


Ilustración 27. Consulta de avisos para perfil Administrador

Si se pulsa el botón de Modificar de un aviso de interés, se abrirá una ventana modal con los campos del aviso, para su modificación.

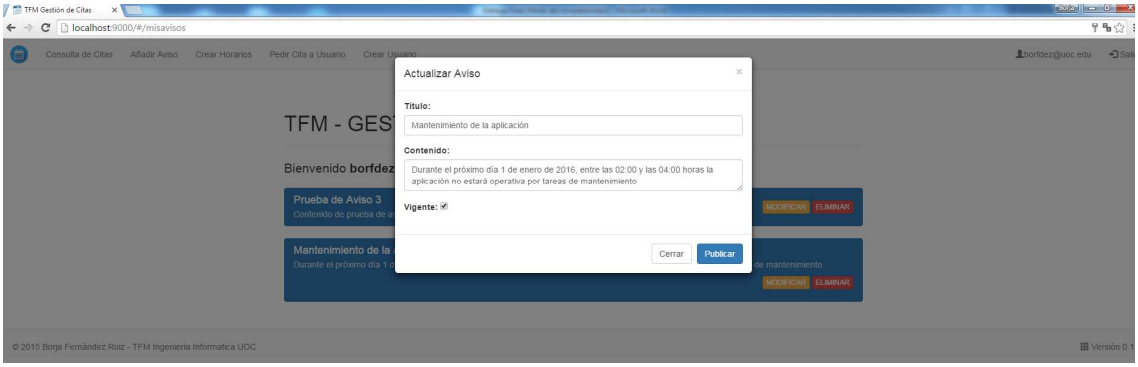


Ilustración 28. Modificación de avisos

Si se modifican datos y se pulsa en el botón Publicar, automáticamente se producirán los cambios y se actualizarán en la pantalla anterior. Igualmente, si se pulsa el botón Eliminar, aparecerá una ventana modal que informa de que si se acepta el borrado, el aviso se borrará automáticamente y permanentemente.

Para crear avisos de interés se pulsa en el menú superior en la opción de menú "Añadir Aviso". Cabe destacar como esta opción de menú queda sombreada, para que el usuario sepa en qué sección se encuentra.

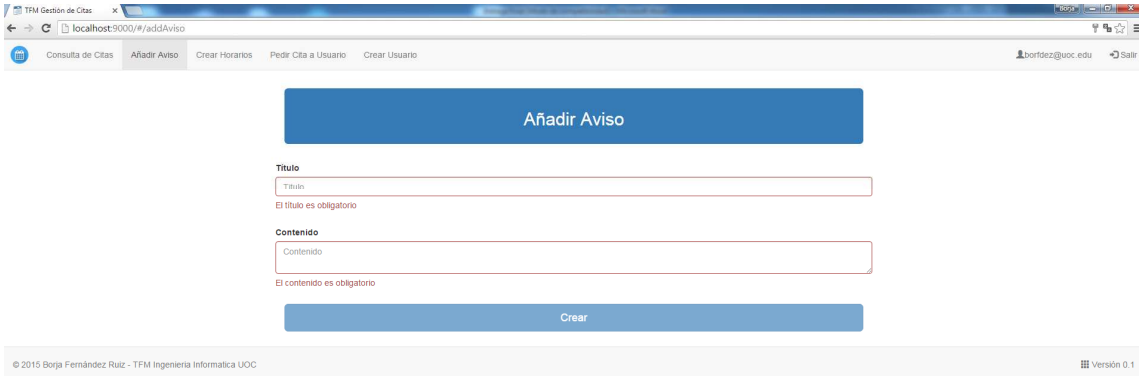


Ilustración 29. Añadir aviso

En el formulario se deberán cumplimentar obligatoriamente el Título y el Contenido del aviso, sino el formulario devolverá sendos mensajes de error. Si se cumplimentan ambos campos y se pulsa el botón Crear, el sistema nos redirige directamente a la página principal, en la que se observará un nuevo aviso, el que se ha creado segundos antes.

Modificar contraseña (para todos los perfiles)

En la parte superior derecha se observa el nombre del usuario conectado y el botón Salir. Si se pulsa sobre el usuario aparece un desplegable con la posibilidad de modificar la contraseña (esta funcionalidad es análoga a todos los perfiles de usuario). Si se accede, aparecerá un formulario para dicha modificación.

Ilustración 30. Modificar contraseña

Este formulario tiene las validaciones pertinentes por lo que no se activará el botón de aceptar si los valores de los campos no tienen la longitud mínima. A su vez se valida que la actual contraseña sea correcta y que la nueva contraseña introducida dos veces coincida. Si todos los datos introducidos son correctos, aparecerá una ventana modal que informa que la contraseña se ha modificado correctamente.

Crear horarios (sólo para perfil Administrador)

Con el objetivo de crear nuevos horarios para reserva de citas, si se pulsa en el menú en la opción "Crear Horario" aparece un formulario que se debe cumplimentar para crear el nuevo horario.

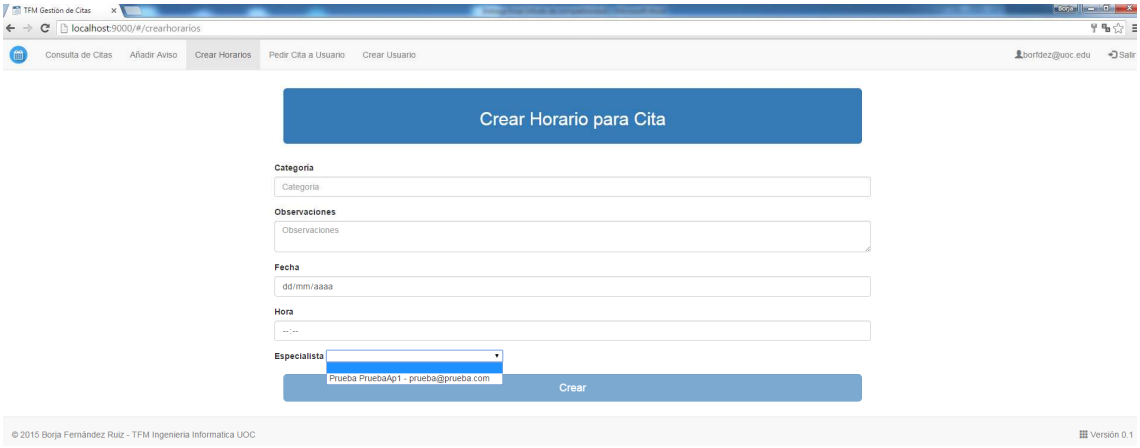


Ilustración 31. Crear horario para cita

Si se cumplimentan todos los datos correctamente (la categoría y las observaciones no son campos obligatorios) y se pulsa el botón crear el horario se crea, mostrando un mensaje indicando este hecho.

Si en los datos introducidos, la fecha completa y el especialista ya están en el sistema, es decir, que ya exista horario en esa fecha y para ese especialista, el sistema devuelve un mensaje de error informando de dicha posible duplicidad.

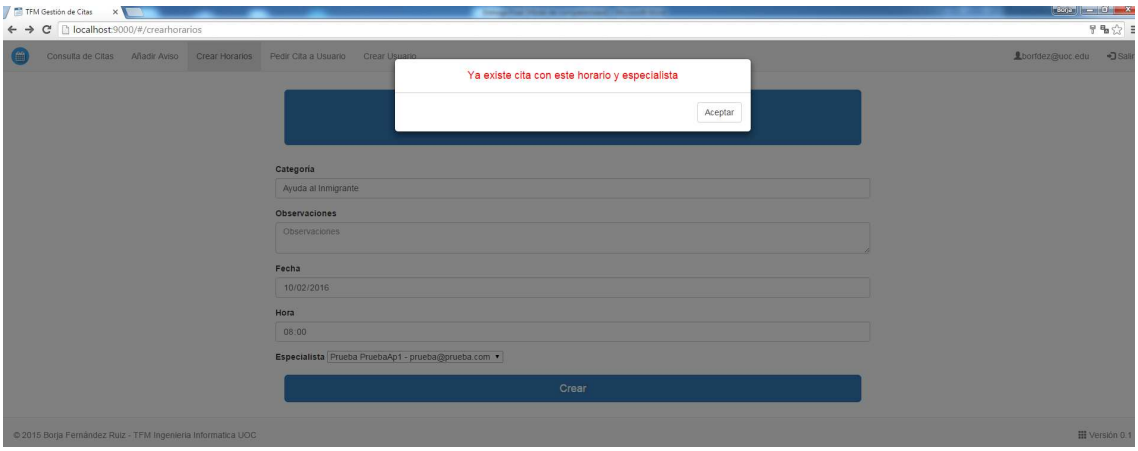


Ilustración 32. Mensaje de error para horarios de cita duplicado

Crear usuarios (sólo para perfil Administrador)

De una manera muy similar a la creación de horarios, si se selecciona la opción de menú para crear usuarios, aparece el formulario para tal funcionalidad.

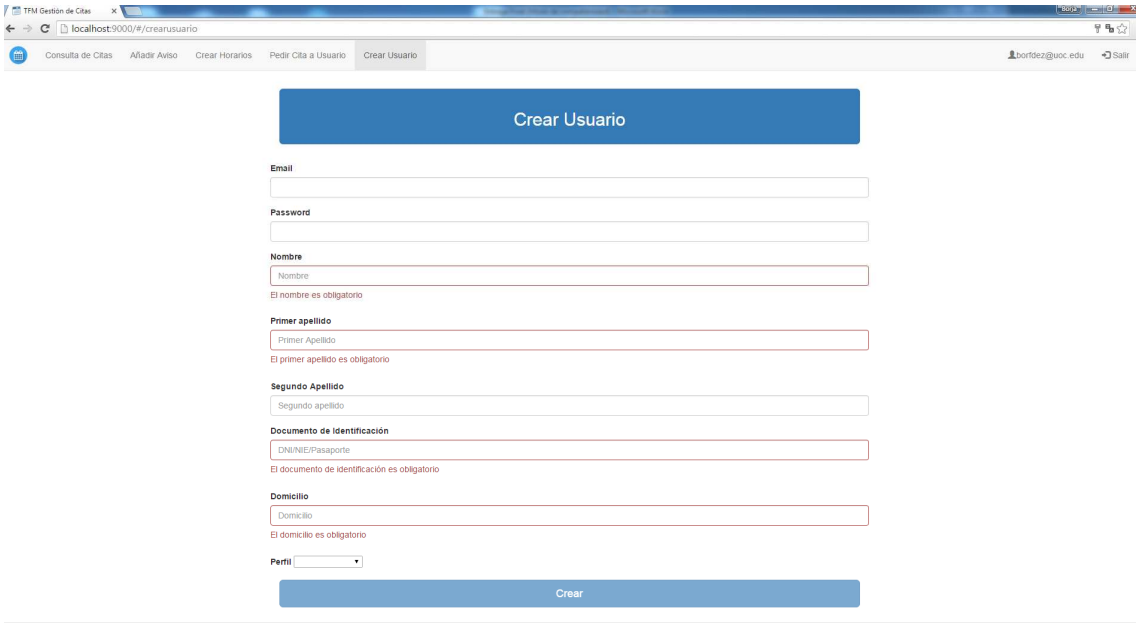


Ilustración 33. Pantalla de creación de usuario

Una vez se cumplimentan los datos y se pulsa el botón Crear, el sistema crea el nuevo usuario en la base de datos.

Consulta de citas (para todos los perfiles)

Los tres perfiles de usuario de la aplicación (Administrador, Especialista y Ciudadano) tienen una opción de menú 'Consulta de Citas' que permite consultar la agenda de citas.

Para el perfil Administrador si se accede a esta funcionalidad se observa una tabla con las distintas citas (ya reservadas por algún usuario o no). Si la cita ya está reservada, la columna de usuario tendrá el email del usuario que la reservó y además tendrá a su derecha el botón de 'Cancelar Cita'. Si las citas no tienen usuario no tendrán esta opción pues están disponibles para ser reservadas, tal y como se observa en la imagen de abajo.

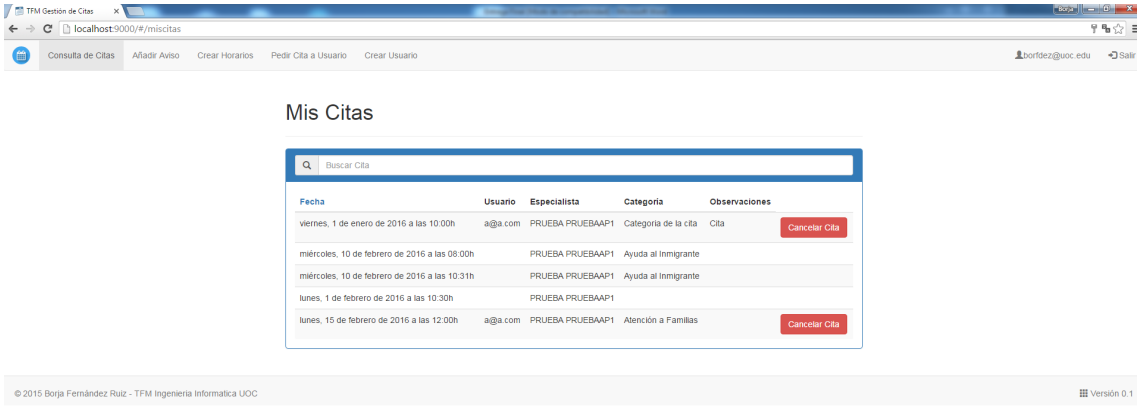


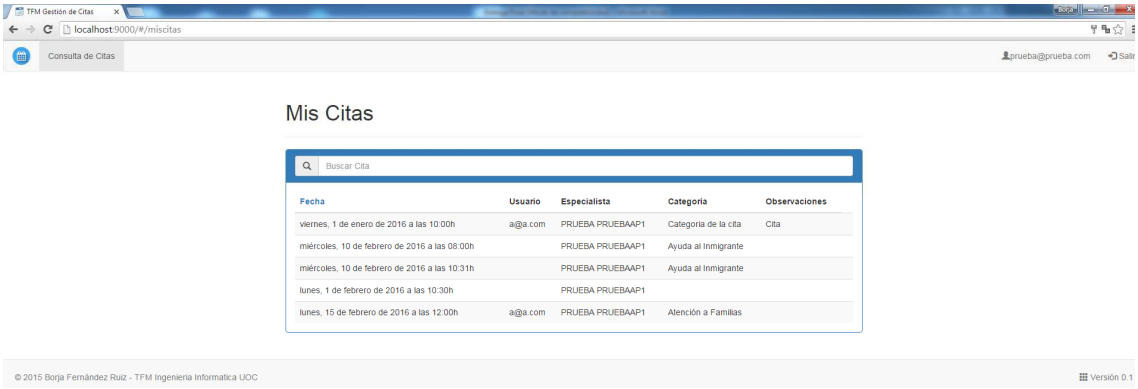
Ilustración 34. Consulta de citas para perfil Administrador

La columna Fecha se puede ordenar (de ahí que el título esté en color azul), de forma que si se pincha sobre él las citas se ordenarán por fecha (de más recientes a menos).

Para el perfil ciudadano esta consulta de citas es parecida, la única diferencia es que sólo le aparecerán sus citas, es decir, las citas que haya solicitado a su nombre (puede haberlas reservado él o un administrador).

Ilustración 35. Consulta de citas para perfil Ciudadano

Para el perfil Especialista esta opción es una mera funcionalidad de consulta. Este usuario de perfil especialista sólo verá sus citas, es decir, sólo verá las citas ya reservadas por algún ciudadano y que le hayan elegido a él como especialista. Como se puede observar, a diferencia de los otros dos perfiles, el usuario con perfil especialista no puede cancelar citas, no tiene esa opción. A continuación se detallará esta funcionalidad.



Solicitar y cancelar citas (para perfil Administrador y perfil Ciudadano)

Tanto los usuarios con perfil ciudadano como los administradores pueden solicitar y cancelar citas. La cancelación de citas se realiza, como se ha visto anteriormente, desde la opción de menú 'Consulta de Citas'. La cita que esté ya reservada para un usuario concreto tiene un botón 'Cancelar Cita'. Si se pulsa sobre él aparece una ventana modal con un mensaje informativo. Si en dicha ventana se pulsa el botón Eliminar la cita pasaría a estado Disponible, dejando por tanto de estar reservada para dicho usuario. En el caso de los usuarios ciudadanos esta cita desaparecería de la tabla de consulta de citas pues ya no es una cita reservada para él.

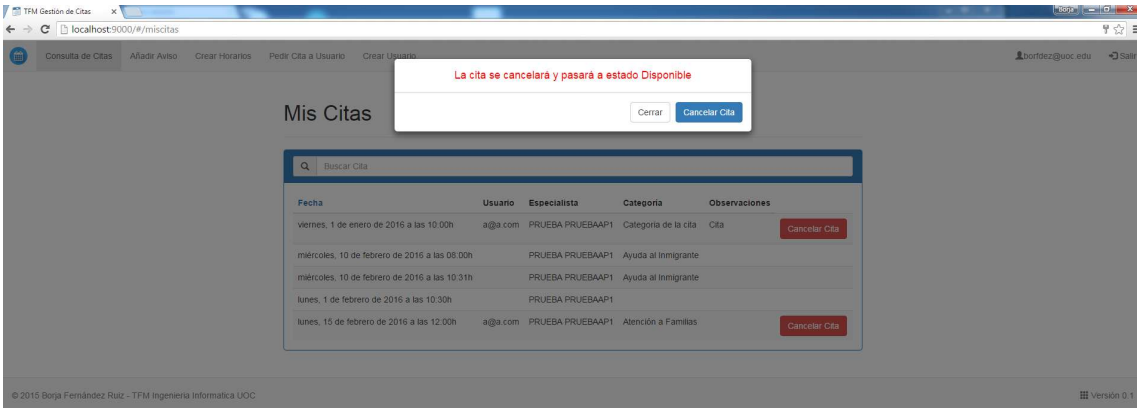


Ilustración 36. Cancelación de cita

En el caso de la solicitud de citas hay que diferenciar entre usuarios ciudadanos y usuarios administradores.

Los usuarios ciudadanos tienen una opción de menú 'Pedir Cita' para esta funcionalidad. Si acceden a este punto de la aplicación se muestra una tabla con los horarios disponibles para solicitar la cita. Si pulsan sobre el botón 'Solicitar Cita' la cita automáticamente quedará reservada para el usuario, desapareciendo de la tabla. Si se accede a la consulta de citas se puede observar como la nueva cita solicitada ya aparece en esta tabla.

Ilustración 37. Solicitud de cita del perfil Ciudadano

Para el usuario administrador existe otra opción de menú que se llama 'Pedir Cita a Usuario'. Al igual que en el caso anterior, al entrar en esta opción de menú aparece una tabla con los horarios disponibles para solicitar cita. A diferencia del caso anterior, si se pulsa sobre el botón Solicitar Cita aparece una ventana modal con los datos del horario elegido y con un desplegable para indicar el usuario sobre el que solicitar la cita. Si se pulsa sobre el botón 'Solicitar' la cita quedará reservada para ese usuario y dicha cita desaparecerá de la tabla de horarios disponibles, sino se puede pulsar sobre el botón 'Cerrar' para no realizar la operación.

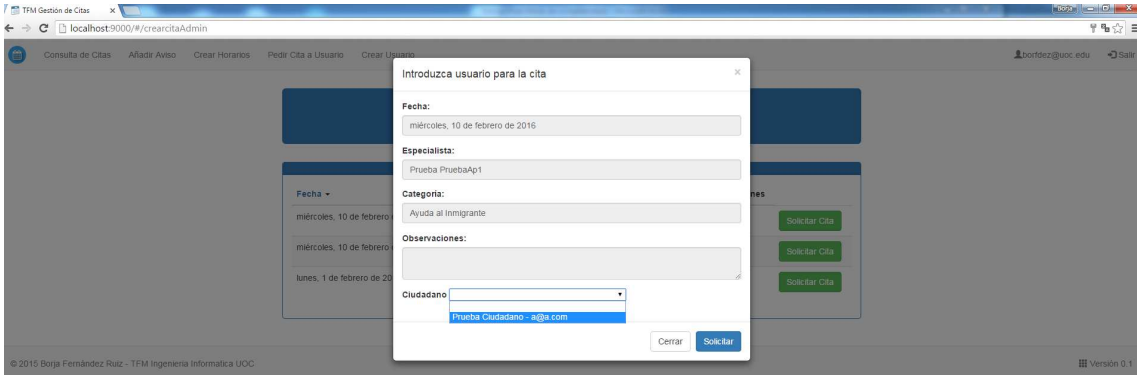


Ilustración 38. Solicitud de cita para un usuario del perfil Administrador

Salir del sistema (para todos los perfiles)

Se puede salir del sistema a través del botón 'Salir' que se encuentra en la parte superior del menú de todos los usuarios.

Anexo 4. Glosario/Índice analítico

En este apartado se detalla el glosario de términos y acrónimos utilizados en el documento.

- Framework: es un esquema (un esqueleto, un patrón) para el desarrollo y/o implementación de una aplicación.
- API: Application Programming Interface en inglés. Conjunto de subrutinas, funciones y procedimientos o métodos que ofrece una biblioteca concreta para ser utilizado por otro software como una capa de abstracción.
- Green Computing: se refiere al uso eficiente de los recursos computacionales minimizando el impacto ambiental, maximizando su viabilidad económica y asegurando deberes sociales.
- CU: caso de uso.
- Sketche: prototipo normalmente realizado a mano sobre papel.
- Responsive (o Responsive Web Design): Diseño web adaptable en castellano. Es una filosofía de diseño cuyo objetivo es adaptar el contenido de una página web o una aplicación web al dispositivo que se esté utilizando para visualizarlas.
- CVS: Concurrent Versioning System en inglés. Implementa un sistema de control de versiones.
- SVN: Subversion en inglés. Sistema de control de versiones open source.

Anexo 5. Bibliografía

En este apartado se detalla la bibliografía y referencias utilizadas a lo largo del documento.

[1] Enlace al BOE. "Ley 11/2007, de 22 de junio, de acceso electrónico de los ciudadanos a los Servicios Públicos". Último acceso: 01/10/2015.
https://www.boe.es/diario_boe/txt.php?id=BOE-A-2007-12352

[2] Enlace al sitio oficial de AngularFire, dentro de Firebase. Último acceso: 27/10/2015.
https://www.firebase.com/docs/web/libraries/angular/

[3] Enlace a github, aplicación todomvc. Último acceso: 28/10/2015.
https://github.com/tastejs/todomvc/tree/master/examples/firebase-angular

[4] Enlace al artículo "Three-Way Data Binding with Firebase and Angular" del blog de Firebase. Último acceso: 28/10/2015.
https://www.firebase.com/blog/2013-10-04-firebase-angular-data-binding.html

[5] Blog Koalite: "AngularJS, gracias por los servicios prestados". Último acceso: 26/12/2015.
http://blog.koalite.com/2014/06/angularjs-gracias-por-los-servicios-prestados/

[6] Blog gfi: "Súper héroes en la web con AngularJS". Último acceso: 26/12/2015.
http://blog.gfi.es/super-heroes-en-la-web-con-angularjs/

[7] Enlace al sitio oficial de MongoDB, contenido “NoSQL Databases Explained”. Último acceso: 24/12/2015.
https://www.mongodb.com/nosql-explained

[8] Enlace a Genbetadev.com, contenido "El concepto NoSQL, o cómo almacenar tus datos en una base de datos no relacional". Último acceso 25/12/2015.
http://www.genbetadev.com/bases-de-datos/el-concepto-nosql-o-como-almacenar-tus-datos-en-una-base-de-datos-no-relacional

[9] Sitio oficial de Yeoman. Último acceso: 26/12/2015.
http://yeoman.io/

[10] Sitio oficial de Bootstrap. Último acceso: 26/12/2015.
http://getbootstrap.com/

[11] Repositorio GitHub de Bootstrap. Último acceso: 26/12/2015.
https://github.com/twbs/bootstrap

[12] Fase Alpha de Bootstrap 4. Último acceso: 26/12/2015.
http://blog.getbootstrap.com/2015/08/19/bootstrap-4-alpha/

[13] Enlace a la página de la herramienta WireFrameSketcher. Último acceso: 27/10/2015.
http://wireframesketcher.com/

[14] Sitio oficial de Protractor end to end testing for AngularJS. Último acceso: 03/01/2016.
https://angular.github.io/protractor/#/

[15] Sitio oficial de Karma. Último acceso: 03/01/2016.
http://karma-runner.github.io/0.13/index.html

[16] Sitio oficial de Jasmine. Último acceso: 22/10/2015.
http://jasmine.github.io/2.2/introduction.html

[17] Tutorial de inicio de Yeoman. Último acceso: 15/12/2015
http://yeoman.io/learning/

[18] Sitio oficial de Firebase. "Migrating for AngularFire 0.9.x to 1.x.x". Último acceso: 26/12/2015.
https://www.firebase.com/docs/web/libraries/angular/migration-guides.html

[19] Guía rápida de Git. Último acceso: 15/12/2015
http://rogerdudler.github.io/git-guide/index.es.html

[20] Sitio oficial de Firebase. "Saving Data". Último acceso: 27/12/2015.
https://www.firebase.com/docs/web/guide/saving-data.html

Otras referencias:

- Canal de Youtube de Israel Guzmán - Tutoriales de AngularJS. Último acceso: 15/12/2015. https://www.youtube.com/user/angularjstutoriales
- Documentación de la API de AngularJS. Último acceso: 07/01/2016. https://docs.angularjs.org/api