



INGENIERÍA EN SISTEMAS DE TELECOMUNICACIÓN

Curso Académico 2017/2018

Trabajo Fin de Grado

Diseño e implementación de una aplicación móvil
híbrida para la planificación de eventos

Autor : Sara López Zambrano

Tutor : Pedro de las Heras Quirós

Trabajo Fin de Grado

Diseño e implementación de una aplicación móvil híbrida para la planificación
de eventos

Autor : Sara López Zambrano

Tutor : Pedro de las Heras Quirós

La defensa del presente Trabajo Fin de Grado se realizó el día de
de 2017, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2017

*Dedicado a
mi familia*

Agradecimientos

El mayor agradecimiento quiero dárselo a mis padres. Les agradezco enormemente todo lo que han hecho por mi tanto en esta etapa de mi vida como en todas las que he vivido. Gracias a ellos he conseguido llegar a este momento tan importante y he logrado alcanzar ese objetivo que me había marcado. Muchas gracias por apoyarme en todas mis decisiones, por ayudarme en los malos momentos y por disfrutar, igual o más que yo, de mis logros.

Tengo que agradecer a esta etapa de mi vida el haber conocido a una persona muy especial, mi pareja. Hemos compartido horas y horas de estudio, nervios, malos momentos, pero sobre todo mucha felicidad de vernos conseguir acabar lo que nos hemos propuesto.

También he conocido personas que me llevo conmigo para siempre, mis telequitas. Les agradezco estar ahí tanto en los buenos como en los malos momentos.

Resumen

Este proyecto se trata del desarrollo de una aplicación móvil cuya funcionalidad principal es el recuento de votos para una determinada encuesta. Tomar decisiones cuando el grupo de personas es amplio es difícil. Para ayudar en estas decisiones se puede utilizar esta aplicación, que permite la creación de encuestas y los votantes pueden elegir cuando o qué les viene mejor. El usuario que crea la encuesta, podrá consultar en cualquier momento cual es la opción que tiene mayor número de votos y a quien pertenecen. Por ejemplo, un equipo de trabajo debe tener una reunión para revisar el estado de su proyecto, pero hay que tener en cuenta la disponibilidad de todos. Una manera fácil de acordar una fecha y hora, en la que todos puedan, es utilizar esta aplicación para crear una encuesta con una serie de posibilidades que después podrán ser votadas por el equipo.

Este proyecto se ha llevado a cabo utilizando una serie de *frameworks* muy utilizados hoy en día como es Angular, para la parte de *frontend*. Basándose en las búsquedas de Google, Angular es uno de los *frameworks* que dominan el mercado. El aprendizaje de esta tecnología es el principal motivo de la realización de este proyecto, aparte de mi gran interés en el desarrollo web. La parte *backend* está desarrollada con el framework Express. Este está basado en Node.js que también es muy utilizado.

Para la realización del desarrollo, aparte de los *frameworks* mencionados, ha sido necesario el aprendizaje de JavaScript, TypeScript, Node.js, HTML5, CSS3, HTTP y MongoDB.

Una vez finalizada la implementación de la aplicación y realizado un despliegue utilizando el PC como servidor, se ha procedido con la realización de una prueba de validación basada en la utilización de esta aplicación por usuarios que han instalado el apk en sus dispositivos móviles. Tras usar la aplicación dichos usuarios se han sometido a una encuesta. La finalidad

de esta, como de cualquier aplicación, es conocer posibles mejoras de cara a futuras versiones y saber como de útil parece a los usuarios.

Índice general

1. Introducción	1
1.1. Historia de la web	2
1.2. Tecnologías para el desarrollo de aplicaciones web	4
1.2.1. HTML5	4
1.2.2. JavaScript	5
1.2.3. TypeScript	6
Ventajas	6
Compilación a JavaScript	7
1.2.4. CSS3	7
1.2.5. Sass	7
1.2.6. Angular	8
Funcionalidades	9
Arquitectura	11
1.2.7. Ionic 2	11
Componentes	11
Apache Cordova	12
Estructura	12

1.2.8. HTTP	13
1.2.9. Node.js	14
1.2.10. Express	14
Estructura	15
1.2.11. MongoDB	15
Colecciones	15
Velocidad	16
1.3. Trabajo relacionado	16
1.4. Estructura de la memoria	17
2. Objetivos	19
2.1. Objetivo general	19
2.2. Objetivos específicos	19
2.2.1. Frontend	19
2.2.2. Backend	20
2.3. Planificación temporal	21
3. Diseño e implementación	23
3.1. Frontend	23
3.1.1. Mapa de navegación	23
3.1.2. Funcional	26
Login	26
Registro	27
Home	28

Header	29
Mis encuestas	30
Mis amigos	31
Perfil	32
Nueva encuesta	33
Votaciones pendientes	36
3.1.3. Arquitectura	36
App	37
Pages	37
Shared	40
3.1.4. Capa de servicios	40
3.2. Backend	43
3.2.1. Servidor	43
3.2.2. Modelos	44
3.2.3. Controladores	45
3.2.4. Base de datos: MongoDB	47
3.2.5. Colecciones	48
Users	48
Polls	48
Friends	48
Votes	49
Pending	49
Send	49

3.2.6. Relación entre colecciones	50
3.2.7. Seguridad	51
4. Prueba	53
4.1. Objetivos de la prueba	53
4.2. Desarrollo	54
4.3. Conclusiones de la prueba	55
5. Conclusiones	57
5.1. Aplicación de lo aprendido	57
5.2. Lecciones aprendidas	57
5.3. Trabajos futuros	58
A. Manual de usuario	59
Bibliografía	61

Índice de figuras

1.1. Arquitectura Angular	11
1.2. Estructura ionic	12
1.3. Estructura express	15
3.1. Arquitectura de la aplicación	23
3.2. Mapa navegación	25
3.3. Pantalla de login	26
3.4. Pantalla de Registro	27
3.5. Errores pantalla de Registro	27
3.6. Pantalla de Home	28
3.7. Header	29
3.8. Pantalla de Menú y Ayuda	29
3.9. Pantalla de Mis encuestas	30
3.10. Pantalla de Mis amigos	31
3.11. Pantalla de Perfil	32
3.12. Tipo de encuesta	33
3.13. Pantalla de Nueva encuesta: paso 1	34

3.14. Pantalla de Nueva encuesta: paso 2	35
3.15. Pantalla de Votaciones pendientes	36
3.16. Estructura componentes	39
3.17. Peticiones HTTP	42
3.18. Ejemplo: ruta encuestas	44
3.19. Ejemplo: modelo encuesta	45
3.20. Ejemplo: función controlador	46
3.21. Arquitectura MongoDB	47
3.22. Ejemplo colección 'Users'	47
3.23. Relación entre colecciones	50
3.24. Encriptación	51
4.1. Resultado prueba	55
4.2. Media resultado prueba	56

Capítulo 1

Introducción

En poco tiempo se ha producido un gran avance en el mundo de las tecnologías y en concreto en los dispositivos móviles. Tanto ha sido así que estamos al alcance de la mayoría de las cosas solo disponiendo de un dispositivo móvil con conexión a Internet. Si navegamos en el tiempo, desde dispositivos que sólo nos ofrecían la capacidad de mantener una comunicación instantánea entre dos personas, hemos llegado a incluso poder realizar pagos. Pero es tan importante la evolución a nivel de dispositivo como a nivel de aplicaciones. Se han desarrollado aplicaciones muy potentes que nos facilitan la vida. Un claro ejemplo es el poder gestionar tus cuentas del banco y realizar operaciones desde cualquier lugar. Existen aplicaciones para todos los gustos, necesidades e intereses.

La oferta en cuanto a desarrollo de aplicaciones móviles ha ido aumentando conforme los hábitos de consumo se han ido dirigiendo hacia el incremento de la utilización de dispositivos móviles. Y es que el estilo de vida que impera en estos momentos, definido por la movilidad y la falta de tiempo, hace que aparatos como tablets o móviles sean mucho más importantes al poderse llevar sin problemas de un lado a otro.

Encontrar un día y una hora para quedar entre varias personas nunca ha sido fácil. Gracias a Internet y a las nuevas tecnologías ya no hay excusas. Existen en la red unas herramientas web llamadas planificadores online que nos permiten encontrar las mejores fechas y horas para organizar una cita, ya sea física o virtual.

1.1. Historia de la web

La historia de la web abarca ya más de 25 años, en los que se han alternado períodos de intenso desarrollo con otros de estancamiento. El primer servidor de páginas web de la historia se puso en marcha en diciembre de 1990. El inventor de la web, Tim Berners-Lee, pretendía crear un sistema que permitiera a los investigadores del CERN compartir fácilmente la información. La primera versión del lenguaje de marcas inventado por Berners-Lee nunca fue publicado como documento oficial, pero si lo hubiera sido se hubiera llamado HTML 1.0.

Los investigadores del CERN, diseminaron en sus universidades de origen el sistema creado por Berners-Lee, puesto que se trataba de un sistema abierto y libre. En aquella época ya existía Internet, pero su acceso estaba limitado principalmente a Universidades y centros de investigación.

En noviembre de 1993 se publicó la versión 1.0 de Mosaic, un navegador creado en la Universidad de Illinois por Marc Andreessen y que superaba a todos al permitir, por ejemplo, incluir imágenes en las páginas web.

En 1994 se permitió el acceso de particulares y empresas a Internet. La web se convirtió enseguida en el servicio más empleado para ofrecer información. La web empezó a verse como una gigantesca oportunidad de negocio y Marc Andreessen dejó la universidad para fundar Netscape, que publicaría la versión 1.0 de su navegador en diciembre de 1994.

En octubre de 1994 Berners-Lee fundó el World Wide Web Consortium (W3C), el W3C está organizado en grupos de trabajo. Los primeros grupos de trabajo que se crearon se dedicaron al HTML y a las CSS.

En 1995 Microsoft incluyó en Windows un navegador, Internet Explorer, que poco a poco comenzó a crecer en el mercado en detrimento de Netscape. Entre 1995 y 2000 Microsoft y Netscape publicaron nuevas versiones cada año. Para diferenciar sus productos, cada navegador fue incorporando nuevas etiquetas, lo que supuso un riesgo de fragmentación de la web.

En esos años, el W3C también publicó recomendaciones a ritmo frenético. Por un lado, para consensuar un HTML común para todos los navegadores. Pero por otro lado, proponiendo innovaciones muy importantes, como la separación entre contenido y presentación mediante

hojas de estilo (CSS).

En 1995 Brian Eitch creó para Netscape 2.0 el lenguaje de programación Javascript, cuyos programas se podían incluir directamente en las páginas web para ser ejecutados por el navegador. Microsoft creó su propia variante parcialmente incompatible. La normalización de Javascript la llevó a cabo la organización ECMA, que en 1997 empezó a publicar normas para unificar y desarrollar el lenguaje.

Ante la necesidad por incluir nuevos campos el W3C creó el XML. El problema era que el HTML no cumplía las nuevas reglas del XML y el W3C planteó reformular el HTML de acuerdo con ellas (ese nuevo lenguaje se llamaría XHTML).

En 1998 Netscape creó la organización Mozilla.

En el año 2000 la guerra de navegadores culminó con la victoria de Internet Explorer y la desaparición de Netscape. Microsoft decidió no seguir innovando y no habría nuevas versiones después de Internet Explorer 6.

Durante estos años, la organización Mozilla desarrolló un nuevo navegador, Mozilla, de uso muy minoritario pero que respetaba las recomendaciones del W3C, apareciendo como una alternativa a Internet Explorer.

Debido a la competencia, Microsoft retomó el desarrollo de Internet Explorer. En 2004 se creó también el WHATWG (grupo formado por Mozilla, Apple y Opera), para retomar el desarrollo del HTML que el W3C había abandonado en favor del XHTML, bajo el nombre de HTML 5.

En 2007 el W3C formó un grupo de trabajo sobre HTML, que trabajaría conjuntamente con el WHATWG para publicar la recomendación HTML 5.

En 2009 Google publicó su propio navegador: Google Chrome. En 2011 el W3C abandonó el desarrollo del XHTML y se concentró en el HTML 5.

En 2011 el WHATWG abandonó por su parte el nombre de HTML 5 y pasó a denominarlo simplemente HTML, abandonando la idea de versiones en favor de una norma "líquida", continuamente modificada y mejorada.

En 2013 Microsoft consiguió con Internet Explorer 11 cumplir de forma correcta las antiguas recomendaciones HTML 4 y CSS 2 y admitir lenguajes XML como SVG. Pero para sacar todo el partido a HTML 5, Microsoft creó un nuevo navegador (Edge), que ya no estará ligado a las nuevas versiones de Windows.

En estos años Google Chrome ha desbancado a Internet Explorer, entre otros motivos debido al gran uso de los teléfonos móviles y al hecho de que los usuarios de Windows 7 no pueden usar Edge.

1.2. Tecnologías para el desarrollo de aplicaciones web

1.2.1. HTML5

HTML, que significa Lenguaje de Marcado para Hipertextos, es el elemento de construcción más básico de una página web y se usa para crear y representar visualmente una página web. Determina el contenido de la página web, pero no su funcionalidad. Hiper Texto se refiere a enlaces que conectan una página web con otra, ya sea dentro de una página web o entre diferentes sitios web. Un lenguaje de marcado hace referencia a aquellos lenguajes que emplean etiquetas. Estas etiquetas ya están predefinidas dentro del lenguaje respectivo y contienen la información que ayudan a leer el texto. Su principal diferencia con los lenguajes de programación es que éstos últimos poseen funciones aritméticas o variables, mientras que los lenguajes de marcado no.

HTML5 es la quinta revisión del estándar que fue creado en 1990 y su versión definitiva se publicó en octubre de 2014. Con HTML5, los navegadores como Firefox, Chrome, Explorer, Safari y más pueden saber cómo mostrar una determinada página web, saber dónde están los elementos, dónde poner las imágenes, dónde ubicar el texto.

En HTML5, se han tomado en cuenta mejoras en la creación de la estructura del código web y en el manejo óptimo de las etiquetas web. De esta manera, se convierte en un estándar mucho más versátil, que permitirá realizar una interacción mucho más poderosa y simple, mejorando la experiencia de uso por parte del usuario y facilitando la depuración del código web.

Las ventajas principales de esta versión son:

- Nueva estructura de etiquetas: permite separar el encabezado, la barra de navegacion, secciones de la página, textos, diálogos y el pie de página.
- Introducción de etiquetas video y audio: por medio de las etiquetas `<video>` y `<audio>` de HTML5, ahora puedes añadir videos o audio sin necesidad de usar Adobe Flash o cualquier otro plugin de tercero. Toda la acción sucede desde el propio navegador, lo que puede ayudar a disminuir al tamaño del archivo final de tu página.
- Geolocalización: permite al sitio detectar la ubicación de cada usuario que ingresa al sitioweb.
- Aplicaciones web: desarrollar aplicaciones HTML5 tiene la ventaja de que el resultado final es completamente accesible, es decir, se puede acceder a esta aplicación desde un ordenador, tablet o móvil.
- Capacidad de realizar ejecuciones offline: esto permite realizar aplicaciones de escritorio.
- Canvas: nueva etiqueta de dibujo sobre la página web.

1.2.2. JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

Surgió por la necesidad de ampliar las posibilidades del HTML. En efecto, al poco tiempo de que las páginas web apareciesen, se hizo patente que se necesitaba algo más que las limitadas prestaciones del lenguaje básico, ya que el HTML solamente provee de elementos que actúan exclusivamente sobre el texto y su estilo, pero no permite, como ejemplo sencillo, ni siquiera abrir una nueva ventana o emitir un mensaje de aviso. La temprana aparición de este lenguaje,

es posiblemente la causa de que se haya convertido en un estándar soportado por todos los navegadores actuales.

Los documentos HTML permiten incrustar fragmentos de código JavaScript, bien dentro del propio archivo HTML o bien realizando una carga de ese código indicando el archivo donde se encuentra el código JavaScript. Dentro de un documento HTML puede haber ninguno, uno o varios scripts de JavaScript.

Además, tambien es utilizado del lado del servidor, ya que tiene la ventaja de poseer un excelente modelo de eventos, ideal para la programación asíncrona.

1.2.3. TypeScript

TypeScript es un lenguaje de programación de alto nivel que implementa muchos de los mecanismos más habituales de la programación orientada a objetos, pudiendo extraer grandes beneficios que serán especialmente deseables en aplicaciones grandes, capaces de escalar correctamente durante todo su tiempo de mantenimiento. Puede ser usado para desarrollar aplicaciones JavaScript que se ejecutarán en el lado del cliente o del servidor (Node.js).

TypeScript convierte su código en Javascript común. Es llamado también Superset de Javascript, lo que significa que si el navegador está basado en Javascript, este nunca llegará a saber que el código original fue realizado con TypeScript y ejecutará el Javascript como lenguaje original.

Ventajas

Las principales ventajas con respecto a JavaScript son:

- Tipado estático: permite mejorar el soporte y la detección de errores sin necesidad de arrancar el código. Los tipos básicos son: boolean, number, Any y void.
- Genéricos: muy útiles para hacer código mas reusable, especialmente en listas y Arrays.
- *decorators*: anotaciones que modifican comportamientos a nivel de clase, propiedad, método o parámetro.

Compilación a JavaScript

Para compilar a JavaScript se puede elegir el target al que queremos compilar, es decir, elegir entre ES5 o ES6.

TypeScript nos permite utilizar código JavaScript, lo que nos permite aprovechar un código ya existente, incluso tiparlo mediante ficheros *.ts para TypeScript.

Tiene la capacidad de concatenar varios ficheros en uno. También ofrece la posibilidad de generar sourcemaps lo cual nos mapea el código compilado a ficheros sin compilar para que haga el proceso de debuguear más sencillo. Por último añadir que genera un código modular.

1.2.4. CSS3

Es el lenguaje utilizado para describir la presentación de documentos HTML o XML. Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página y una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento.

El objetivo inicial de CSS, separar el contenido de la forma, se cumplió ya con las primeras especificaciones del lenguaje. Sin embargo, el objetivo de ofrecer un control total a los diseñadores sobre los elementos de la página ha sido más difícil de cubrir. Las especificaciones anteriores del lenguaje tenían muchas utilidades para aplicar estilos a las webs, pero los desarrolladores aún continúan usando trucos diversos para conseguir efectos tan comunes o tan deseados como los bordes redondeados o el sombreado de elementos en la página.

1.2.5. Sass

Sass es un metalenguaje de Hojas de Estilo en Cascada (CSS). Es un lenguaje de script que es traducido a CSS.

Extiende CSS proveyendo de varios mecanismos que están presentes en los lenguajes de programación tradicionales, particularmente lenguajes orientados a objetos, pero éste no está disponible para CSS3 como tal. Cuando SassScript se interpreta, éste crea bloques de reglas CSS

para varios selectores que están definidos en el fichero SASS. El intérprete de SASS traduce SassScript en CSS.

Sass permite la definición de variables de tipo number, string, colores y booleanos. Otra ventaja es que soporta mixins. Un mixin es una sección de código que contiene código Sass. Cada vez que se llama un mixin en el proceso de conversión el contenido del mismo es insertado en el lugar de la llamada. Los mixin permiten una solución limpia a las repeticiones de código, así como una forma fácil de alterar el mismo.

1.2.6. Angular

Angular es un framework de desarrollo para JavaScript creado por Google. La finalidad de Angular es facilitar el desarrollo de aplicaciones web SPA y además proporcionar herramientas para trabajar con los elementos de una web de una manera más sencilla y óptima. Una aplicación web SPA creada con Angular es una web de una sola página, en la cual la navegación entre secciones y páginas de la aplicación, así como la carga de datos, se realiza de manera dinámica, casi instantánea, asincrónicamente haciendo llamadas al servidor (*backend* con un API REST) y sobre todo sin refrescar la página en ningún momento. Es decir las aplicaciones web que podemos hacer con Angular son reactivas y no recargan el navegador, todo es muy dinámico y asíncrono con ajax.

Los fundamentos de Angular son:

- Web components:

Es una API estandar para crear componentes. Tiene cuatro partes principales: las templates son bloques de html optimizados para ser reusados, HTML *imports* permite la carga de dependencias de la forma mas optima posible, *Custom Elements* permite definir tabs propios para HTML5 con su propio ciclo de vida y su lógica y *Shadow DOM* que se centra en modularizar el DOM, esto da la ventaja de modularizar el CSS y evitar conflictos entre distintas reglas del CSS.

- Programación reactiva:

La programación reactiva es un nuevo paradigma orientado a programar basado en flujos de datos que son los encargados de transmitir los cambios a nuestra aplicación. Con este nuevo paradigma se intenta simplificar la arquitectura de las aplicaciones y centrarla más en el uso real de la aplicación: responder a eventos, tareas asíncronas y flujos con esas tareas asíncronas, etc.

En Angular esto se puede llevar a cabo con la librería RxJS. Dicha librería es una enorme cantidad de clases y métodos para poder hacer que toda la programación que se ha venido haciendo, se pueda manejar de manera reactiva basada en Observables de manera fácil.

- ES6:

Es el estándar que sigue JavaScript desde junio de 2015.

Contiene algunas novedades como: clases e interfaces, variables y constantes, arrow functions, promises, definición de módulos, iteradores, etc.

- TypeScript:

Como se ha explicado más detalladamente en el apartado 1.2.3, es más interesante que el uso de JavaScript ya que permite tipado estático, hacer genéricos y *decorators*.

Funcionalidades

- Modularización

La filosofía de Angular empuja hacia que todo sea modular. Se crea una clase, se exporta y ya está disponible para ser importada en otro componente o en otra parte de la aplicación y ser reutilizada. La propia librería de Angular está modularizada.

- Componentes

Pieza de código que controla una parte de la vista. No es nada más que una clase de ES6 añadiéndole un *decorator* de TypeScript llamado Component. Esto permite transformar esta clase en un web component con un nombre de selector y una template. La template puede crearse en un fichero aparte para que la organización del código sea buena.

Una aplicación así de Angular es un componente, que contiene subcomponentes.

Angular tiene un ciclo de vida para componentes y directivas, como son: ngOnChanges, ngOnInit, ngDoCheck, ngAfterContentInit, ngAfterContentChecked, ngAfterViewInit, etc.

- **Templates**

En esta parte se crean los bloques de la sintaxis de HTML. Acepta todas las tags de HTML5 menos script, no puede embeber código JavaScript dentro de la template.

Las etiquetas de HTML se pueden potenciar con las directivas de Angular.

Toda la lógica que se ejecute en el template está siendo ejecutado en el *scope* del componente al que pertenece.

- **Bindeo de datos**

El bindeo es de una sola dirección. Los distintos tipos de bindeo que existen son del componente hacia el DOM, para mostrar valores en la vista, para pasarselo a otras templates como parámetro, y del DOM al componentes, para enviarle los eventos.

- **Inyección**

Usa la inyección de dependencias para los servicios. Un servicio es una clase de ES6 que contiene un *decorator* llamado Injectable. De esta forma esta clase puede ser inyectada en cualquier parte de la aplicación. Además es necesario incluirlo como *provider* de forma que sepa de esta dependencia la aplicación.

Para inyectar, se hace a través del constructor.

- **Routing**

Angular tiene su propio componente de rutas y navegación entre vistas. Es un servicio opcional y no forma parte del *core* de Angular.

Interpreta las URL como una instrucción de navegación a un componente o vista. Opcionalmente puede pasar parámetros.

- **Detección de cambios**

Angular crea por cada componente un detector de cambios en tiempo de ejecución adaptado a la estructura concreta de cada uno de ellos. Esto permite una gran optimización de dichos detectores de cambios.

Arquitectura

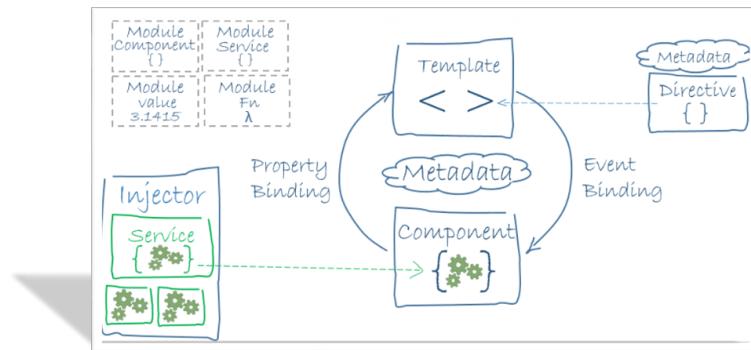


Figura 1.1: Arquitectura Angular

1.2.7. Ionic 2

Se trata de un framework destinado al desarrollo de aplicaciones híbridas, aunque también puede ser utilizado para implementar aplicaciones web. Una aplicación híbrida es aquella desarrollada por las tecnologías web: HTML, CSS Y JavaScript. Este tipo de aplicaciones tienen una serie de ventajas como ser compatibles para una gran cantidad de sistemas operativos con un tiempo de desarrollo menor, pero a cambio de esta gran ventaja el rendimiento es menor que en una aplicación nativa.

Su característica fundamental es que usa por debajo Angular, esto le da ventajas como tener una buena estructura de proyecto y contar con una buena gama de componentes y directivas.

Componentes

Los componentes se utilizan unos a otros para la obtención de objetivos globales de la aplicación. Están pensados para, de manera modular y encapsulada, resolver pequeños problemas. Ionic ofrece componentes fáciles de utilizar, pero para comportamientos más específicos de nuestro modelo de negocio, será necesario crear nuestros propios componentes.

Los componentes de Ionic 2 se adaptan al dispositivo estéticamente. Manteniendo el mismo código, en un dispositivo iOS tiene diferente vista que en un dispositivo Android, ya que se visión de nativa y además da al usuario una experiencia cercana a la que está acostumbrada en

su teléfono. Sin embargo, es decisión del desarrollador mantener esta visión en su aplicación o personalizar la estética a su gusto. adapta al sistema operativo en el que se compila. Esto permite que una aplicación híbrida de

Apache Cordova

Para el acceso a componentes nativos desde la aplicación de Ionic, como la cámara, acelerómetro, teclado, usa plugins que proporciona Apache Cordova. También nos permite compilar el desarrollo realizado con Ionic con tecnologías web en aplicaciones para móviles instalables mediante tiendas de aplicaciones.

Estructura

Un proyecto con Ionic contiene una lista de carpetas y archivos. Cada parte tiene su función:

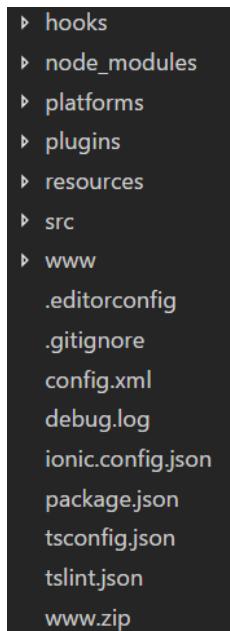


Figura 1.2: Estructura ionic

- SRC: carpeta que contiene los archivos fuente con el código desarrollado de la aplicación.
- WWW: contiene los archivos que se producen al realizar la transpilación del TypeScript y compilado de los archivos Sass, es decir, la transformación de todos los archivos de 'src',

de tal manera que el navegador sea capaz de entender.

- PLUGINS: contiene todos los plugins nativos que se utilizan para la aplicación.
- PLATFORM: archivos de cada plataforma a la que da soporte la aplicación. Suele ser Android e iOS.
- RESOURCES: contiene los iconos de la aplicación y el splash screen.
- HOOKS: scripts que se crean para ser ejecutados automáticamente después de algo específico.
- NODE MODULES: dependencias de npm que vienen definidas en el package.json e instaladas en local dentro de tu proyecto.

1.2.8. HTTP

Son las siglas de *Hypertext Transfer Protocol*. Es un protocolo de transferencia donde se utiliza un sistema mediante el cual se permite la transferencia de información entre diferentes servicios y los clientes que utilizan páginas web.

Este protocolo opera por petición y respuesta entre el cliente y el servidor. A menudo las peticiones tienen que ver con archivos, ejecución de un programa, consulta a una base de datos, traducción y otras funcionalidades. Toda la información que opera en la Web mediante este protocolo es identificada mediante el URL o dirección.

La típica transacción de protocolo HTTP se compone de un encabezado seguido por una línea en blanco y luego un dato. Este encabezado define la acción requerida por el servidor.

Una solicitud HTTP es un conjunto de líneas que el navegador envía al servidor. Contiene una línea de solicitud que especifica el tipo de documento solicitado, el método que se aplicará y la versión del protocolo utilizada. La línea está formada por tres elementos que deben estar separados por un espacio: el método, la dirección URL y la versión del protocolo utilizada por el cliente. También contiene los campos del encabezado de solicitud y el cuerpo de la solicitud.

1.2.9. Node.js

Node.js es un entorno de ejecución multiplataforma de código abierto para desarrollar aplicaciones web. Esta librería se ejecuta sobre JavaScript y está basado en el motor V8 de Javascript de Google. Este motor está diseñado para correr en un navegador y ejecutar el código de Javascript de una forma extremadamente rápida.

Se trata de un intérprete Javascript del lado del servidor, lo que permite utilizar el mismo lenguaje de programación tanto para cliente como para servidor. Node sirve para facilitar la creación de aplicaciones web escalables de manera sencilla y con gran estabilidad, pudiendo ser utilizado para desarrollar cualquier tipo de aplicación. Además, es importante volver a destacar su altísima velocidad y su flexibilidad, dos de sus cualidades más importantes.

Trabaja con un único hilo de ejecución que es el encargado de organizar todo el flujo de trabajo que se deba realizar. Gestiona sus tareas de manera asíncrona y para trabajar de manera óptima delega todo el trabajo en un poll de threads. La librería que construye esto es Libuv, una vez que el trabajo ha sido completado emite un evento recibido por Node.js.

1.2.10. Express

Express es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles. Contiene muchos métodos de programa de utilidad HTTP y *middleware*.

Node.js es una plataforma construida sobre el motor de JavaScript de Google Chrome (V8) que permite la ejecución de JavaScript en el lado del servidor. Permite montar un servidor HTTP utilizando el modulo http que viene incluido en el core de Node.

Para comenzar un proyecto en Express es necesario configurar en qué puerto e IP va a estar escuchando el servidor para atender a las peticiones. Además, es necesario añadir las urls a las que tiene que atender y qué método realizar para cada caso.

Estructura

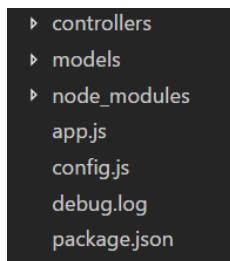


Figura 1.3: Estructura express

1.2.11. MongoDB

Se trata de una base de datos no relacional, es decir, NoSQL. Esto significa que los datos no son almacenados en tablas y no garantiza consistencia. Este tipo de bases de datos tienen ciertas desventajas, como: poca eficiencia en aplicaciones que necesiten usar los datos intensivamente, o si contiene gran número de indexaciones. A cambio de esta desventaja tiene la capacidad de manejar mucha cantidad de datos, no generan cuellos de botella y se ejecutan en clusters de máquinas baratas.

Este tipo de bases de datos surgieron por el *Big Data*, donde la información que se genera es muy grande, de manera rápida y constante, y que, además, en ocasiones la forma es no estructurada y cambiante. Las bases de datos relacionales tenían carencias para afrontar esto.

Colecciones

MongoDB es una base de datos orientada a documentos, es decir, los documentos son almacenados en BSON, que es una representación binaria de JSON. No siguen un esquema fijo, los documentos de una misma colección pueden tener esquemas diferentes.

En MongoDB los documentos se agrupan en colecciones. Aunque lo normal es que los documentos de una colección comparten estructura, puede ser flexible porque la estructura no se impone a ningún documento y dinámica porque la estructura puede cambiar. En el caso de que se decida variar la estructura de datos, no sería necesarios crear ni modificar las colecciones,

bastaría con almacenar los nuevos documentos, con una estructura distinta, en la misma colección o en otra.

No proporciona integridad referencial. Esto significa que, si en una colección hacemos referencia a un documento de otra colección, la base de datos no tiene la responsabilidad de comprobar que el documento referenciado existe.

Velocidad

MongoDB tiene una baja velocidad en generar o modificar información. Esto provoca que el acceso a la base de datos este más tiempo bloqueado. Una operación de escritura bloquea el acceso a toda la base de datos en la que se efectúa la operación. Para solucionar esto hay que evitar actualizaciones que provoquen movimientos, es decir, que el crecimiento de un documento sea tan grande que suponga un cambio de cajón. Cada documento se encuentra almacenado en un cajón el cual cuenta con un espacio extra para posibles crecimientos futuros.

A cambio de esta baja velocidad de escritura, proporciona una gran rapidez de lectura.

1.3. Trabajo relacionado

Este proyecto esta inspirado en una aplicación similar conocida con el nombre de Doodle. Es una herramienta automatizada muy sencilla que te evita hacer varias llamadas o enviar multitud de emails o mensajes para quedar con alguien. Por lo tanto, mejora la organización en todos los sentidos, tanto en eficiencia como en rapidez y concreción.

Pero esta no es la única aplicación que existe para el tema de la planificación, existen otras similares:

- Timebridge: dirigida a profesionales. Su gran aliciente es que dispone además de soluciones para teleconferencia y reuniones en línea, aunque no pueda competir con otras más especializadas en ello y sean de pago. Además no requiere instalación y permite compartir archivos, acciones de pizarra, tomar notas en tiempo real de forma colaborativa, el seguimiento posterior a la reunión de elementos de acción y notas.

- When is Good: es una de las más sencillas aunque no muy intuitiva y la más adecuada para programar planes informales.
- Calendly: se recomienda para establecer reuniones comerciales ya que funciona para encuentros cara a cara y al permitir acceder a la disponibilidad de la otra persona se pueden concertar citas de forma inmediata.
- TimePal: es una aplicación muy útil cuando los asistentes son de diferentes husos horarios. Funciona como un tablero en el que se introducen las distintas horas en la que se está disponible y muestra la hora que sería en las otras regiones. Además indica las horas del amanecer y atardecer en los distintos lugares así como si es horario laboral o no.
- ScheduleOnce: servicio orientado a empresas se conecta con los calendarios de Outlook, Office 365, Google, and iCloud y se puede integrar con Salesforce, Infusionsoft, GoToMeeting and WebEx. Accedes a la página web, indicas las franjas horarias que te interesan y la herramienta crea dos enlaces: el primero se enviará a los invitados, que indicaran el día y la hora que les conviene, y, el segundo, servirá para hacer el seguimiento de las respuestas.

1.4. Estructura de la memoria

Después de esta introducción, en la que hemos analizado diferentes plataformas que motivan el desarrollo de este proyecto y ofrecido una visión global del mundo web, explicando sus orígenes, como ha evolucionado su desarrollo en la actualidad con tecnologías modernas y comentando brevemente las tecnologías y herramientas utilizadas en este proyecto, continuaremos describiendo las necesidades que han llevado a la realización de este Trabajo Fin de Grado. Esto se presenta en el capítulo 2.

En el capítulo 3 se explica el diseño e implementación del proyecto. Este capítulo es el más extenso ya que contiene todo el desarrollo del proyecto. De la parte *frontend* se detalla el mapa de navegación que contiene la aplicación, el diseño de cada pantalla con su correspondiente funcionalidad y la estructura de proyecto. De la parte *backend* se describe la estructura de

colecciones de la base de datos, los controladores utilizados, las relaciones que existen entre las colecciones y la seguridad.

En el capítulo 4 se detalla como se ha llevado un experimento cuya finalidad es conocer la opinión de los usuarios sobre la aplicación, es importante que tenga facil usabilidad.

Por último están las conclusión obtenidas tras todo el aprendizaje y desarrollo del proyecto, así como trabajos futuros para mejorar ciertas partes de la aplicación.

Capítulo 2

Objetivos

2.1. Objetivo general

Mi trabajo fin de grado consiste en crear una aplicación que ayude a tomar decisiones a la hora de organizar eventos o reuniones, además de realizar encuestas de todo tipo.

2.2. Objetivos específicos

2.2.1. Frontend

- Controlar los formatos introducidos al registrarse, de tal manera que no pueda introducirse un email con formato no válido ni una contraseña con menos de 6 caracteres.
- Visión global del usuario en cuanto a todas las encuestas que ha creado y poder visualizar los votos de esta.
- Permitir elegir el tipo de encuesta que se quiere crear. Esta elección sera entre encuesta de tipo fecha o de tipo texto.
- Dar opción al usuario de añadir hora y fecha o solo una fecha, en el caso de encuesta de tipo fecha.

- Al crear una encuesta poder elegir si se permite votar multiples opciones o se restringe a una sola votación por persona.
- Validar los campos introducidos al crear una encuesta. Los campos obligatorios son el título y las opciones.
- Como datos opcionales, poder añadir a la encuesta la ubicación del evento o algún comentario.
- Un usuario sólo podrá enviar la encuesta creada a aquellos usuarios que estén en su lista de amigos
- Tras realizar la votación de una encuesta, es posible modificar el voto.
- Un usuario puede cambiar sus datos personales de la cuenta, que son el nombre de usuario y contraseña.
- Para facilitar la búsqueda de amigos, el usuario no deberá introducir el nombre concreto en el buscador, se hace una búsqueda de todos los nombres que contengan los caracteres introducidos.
- Un usuario que aún no tenga amigos en su lista, no puede crear una encuesta.

2.2.2. Backend

- Sólo puede haber un nombre de usuario y email registrado en la aplicación.
- Encriptar las contraseñas que se han registrado y así guardarlas en la base de datos.
- Permitir el cambio de nombre de usuario o contraseña, siempre que el usuario introduzca una contraseña válida.
- Permitir borrar una cuenta.
- Solo puede haber un voto por usuario y opción.

2.3. Planificación temporal

El tiempo empleado la finalización de este proyecto ha sido de cinco meses naturales, de los cuales la dedicación solo ha podido ser parcial. La media de horas empleadas al día entre semana ha podido ser, aproximadamente, de tres horas y los fines de semana una media de cinco horas.

El proyecto ha pasado por varias fases a lo largo del tiempo:

- Primer mes:

Mi dedicación durante este tiempo fue a la realización de un tutorial sobre Angular, el cual me dió una serie de conocimientos para poder empezar la parte visual de la aplicación. Tras este tutorial, empecé a crear las primeras pantallas de la aplicación.

- Segundo mes:

Este mes fue dedicado a la realización de un tutorial destinado a conectar Angular con Meteor para poder desarrollar la parte *backend* con Meteor. Durante este periodo, debido a una serie de problemas con la realización de este tutorial, decidí buscar otra alternativa y así poder avanzar en el proyecto. Finalmente la tecnología elegida fue Express. Esto me permitió poder realizar las primeras llamadas HTTP desde la parte *frontend* de la aplicación.

- Tercer y cuarto mes:

Una vez montada la base de la parte front y la parte back, desarrolle en paralelo ambas partes. Desarrollaba la pantalla y a la par, creaba los controladores necesarios en Express para que atendiera a las peticiones necesarias en dicha pantalla. Una vez terminado todo el desarrollo, fue importante la parte del testeo de la aplicación. Durante esta parte se cambiaron funcionalidades de la aplicación como: poder añadir hora como opción en una encuesta, ordenar alfabéticamente los listados, cambio en pantalla de visualización de votos, etc.

- Quinto mes:

Este último tramo ha sido dedicado a la elaboración de la memoria y la preparación de la presentación. Además de esto, durante esta etapa, se hizo el experimento de poner a un grupo de personas a usar la aplicación siguiendo una serie de hitos, para posteriormente contestar unas preguntas.

Capítulo 3

Diseño e implementación

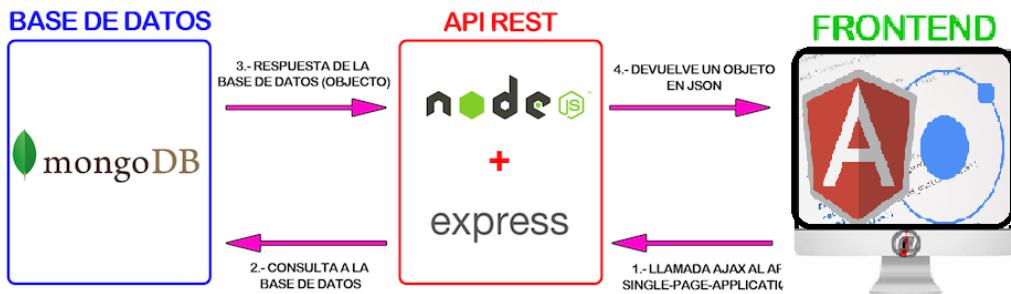


Figura 3.1: Arquitectura de la aplicación

3.1. Frontend

Es una especialidad para el desarrollo web, que trabaja la interfaz web y hace que el usuario pueda interactuar con nuestra web. Casi todo lo se ve en la pantalla cuando se accede a una web es desarrollo *frontend*, se centra en dar formato a contenidos, desarrollo del aspecto de la web y manipular resultados de datos obtenidos.

3.1.1. Mapa de navegación

El término navegación describe la acción de moverse entre las páginas y dentro de la página. Es el punto de partida de la experiencia del usuario. Es la forma en que los usuarios buscan el

contenido y las características que les interesan.

Existen dos tipos de estructura de navegación: jerárquica y plana. En una estructura jerárquica, las páginas se organizan en una estructura parecida a un árbol. Cada página secundaria tiene un único elemento primario, pero un elemento primario puede tener una o más páginas secundarias. Para llegar a una página secundaria, hay que moverse a través del elemento primario. En el caso de la estructura plana o lateral, las páginas existen en paralelo. Puedes ir de una página a otra en cualquier orden.

Esta aplicación sigue una estructura combinada entre jerárquica y plana. Se usan estructuras planas para las páginas de nivel superior que pueden verse en cualquier orden, y estructuras jerárquicas para las páginas que tienen relaciones más complejas.

En la figura 3.2 se puede ver el mapa de navegación de la aplicación. Empieza en la pantalla de login. Desde ahí se puede acceder al registro o a la pantalla de home si se han introducido bien los parámetros. Una vez en la pantalla de home, se puede acceder a cualquier parte de la aplicación.

Cada una de las demás pantallas tienen su propia navegación. En el caso de Mis encuestas, se puede acceder al detalle de esta. En el caso de nueva encuesta, se accede al primer paso, de este al segundo y de este al tercero. En el caso de Mis amigos, no contiene ninguna navegación aparte de la propia del Header y en el caso de Votos pendientes, de ella puedes navegar a editar votos.

La navegación del Header, también podemos verla en la figura 3.2 y esta puesta como un caso por separado, ya que es común para todas las pantallas que contienen estos header. Como se puede ver, existen dos tipos de header, que serán explicados con mas detalle en el apartado 3.1.2.

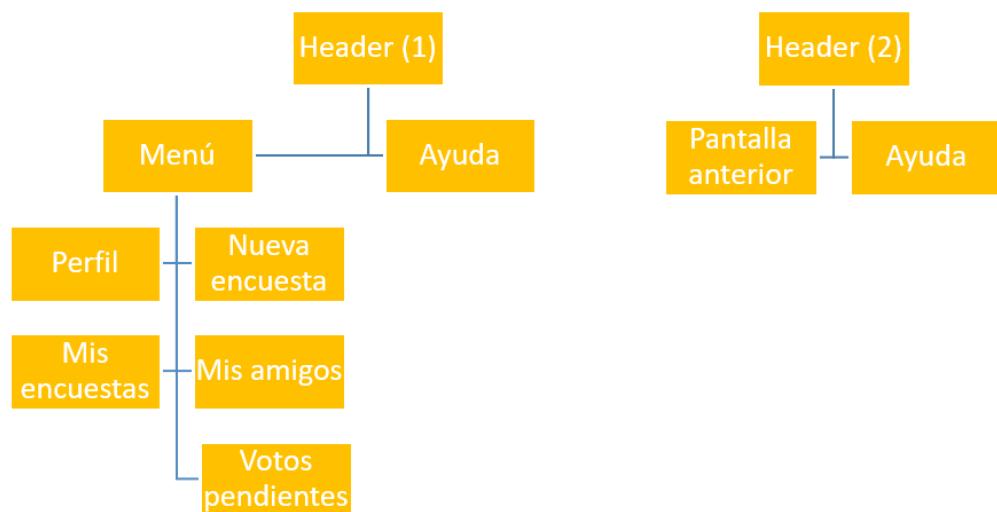
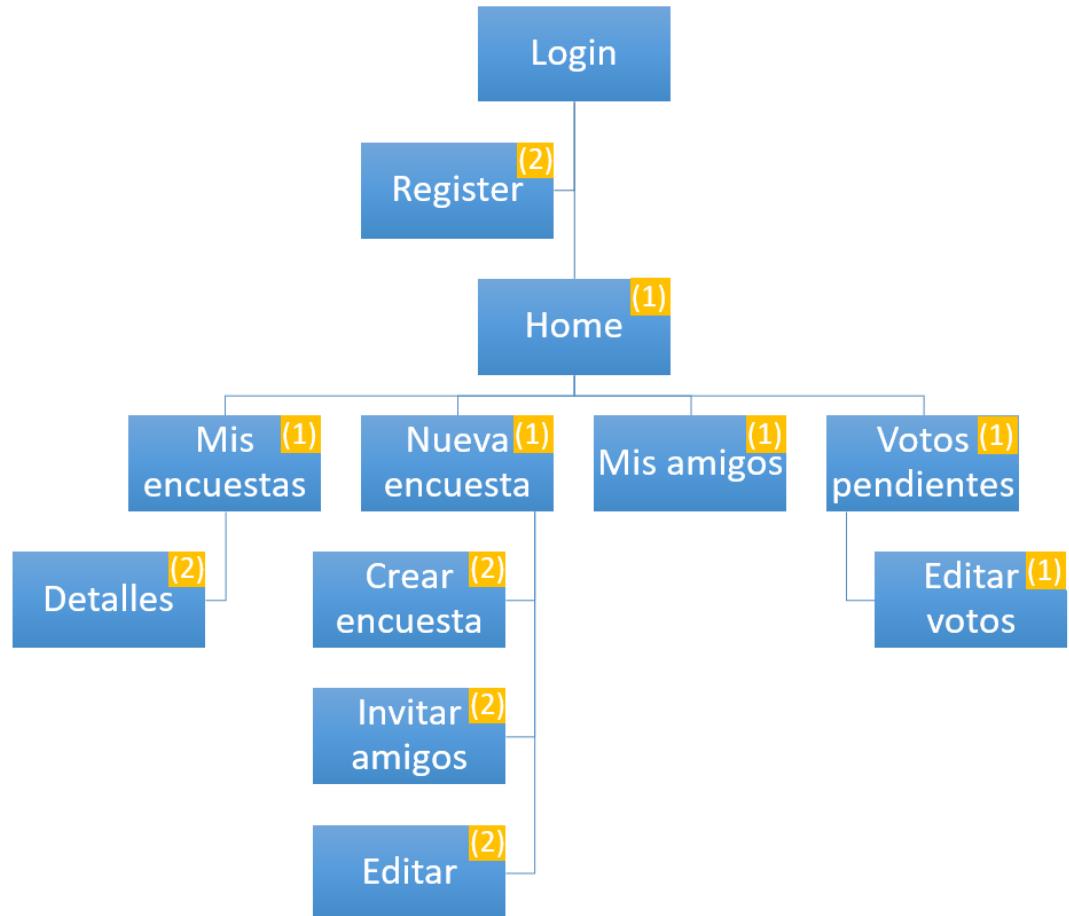


Figura 3.2: Mapa navegación

3.1.2. Funcional

Login

(a) Login



[Crear cuenta](#)

[Iniciar sesión](#)

(b) Error



[Crear cuenta](#)

[Iniciar sesión](#)

Figura 3.3: Pantalla de login

En la pantalla de login podemos encontrar dos inputs para poder introducir el correo y la contraseña. De esta forma se puede comprobar si el usuario tiene cuenta en la aplicación y además se identifica al usuario que está usando la aplicación.

Se trata de un formulario reactivo, esto permite que los valores no tienen que ser recuperados de un servidor, se pueden validar de forma inmediata. Ambos son campos requeridos por lo que se comprueba que han sido introducidos, y en el caso de la contraseña que contenga mínimo 6 caracteres. Para el caso del email, se comprueba que sea un formato valido.

También encontramos un enlace para que aquellos usuarios que aún no han utilizado esta aplicación puedan crearse una cuenta. Este enlace lleva a la página de registro.

Finalmente encontramos un botón para iniciar sesión. Al pulsar en este botón se realiza la validación del formato de email y se manda una petición al servidor.

Registro



Figura 3.4: Pantalla de Registro

Usuario: <input type="text" value="sara"/> <div style="color: red; font-size: small;">Ya existe una cuenta con este nombre</div>	Email: <input type="text" value="s@s.es"/> <div style="color: red; font-size: small;">Ya existe una cuenta con este email</div>
(a) Nombre	(b) Email
 Contraseña: <input type="password" value=".."/> <div style="color: red; font-size: small;">La contraseña debe contener al menos 6 caracteres</div>	
Repetir contraseña: <input type="password" value="....."/> <div style="color: red; font-size: small;">No coinciden</div>	
(c) Contraseña	(d) Contraseña no coincide

Figura 3.5: Errores pantalla de Registro

Contiene un formulario para los datos personales del usuario. Es un formulario reactivo con sus correspondientes validaciones. Estas no se comprueban hasta que no se pulsa el botón 'Registrate'.

- Usuario: mediante una petición al servidor, se comprueba que no hay ningún usuario con este nombre.

- Email: se comprueba que sea un formato valido y, además, mediante una petición al servidor, que no esté ya registrado.
- Contraseña: se comprueba que no contenga menos de 6 caracteres.

He considerado que para este tipo de aplicación no son necesarios más datos sobre el usuario, lo mas importante es identificar quien esta votando y a quien va dirigida la encuesta.

Home



Figura 3.6: Pantalla de Home

En esta pantalla se encuentran accesos directos a distintas partes de la aplicación. Desde esta se puede acceder a todas las partes de la aplicación, excepto al perfil que se accede mediante el menú lateral.

El usuario puede acceder a sus encuestas creadas, a crear una nueva encuesta, consultar y añadir a sus amigos y acceder a las votaciones.

Al pulsar alguno de los accesos directos, se produce la navegación a la pantalla que corresponde.

Header



Figura 3.7: Header

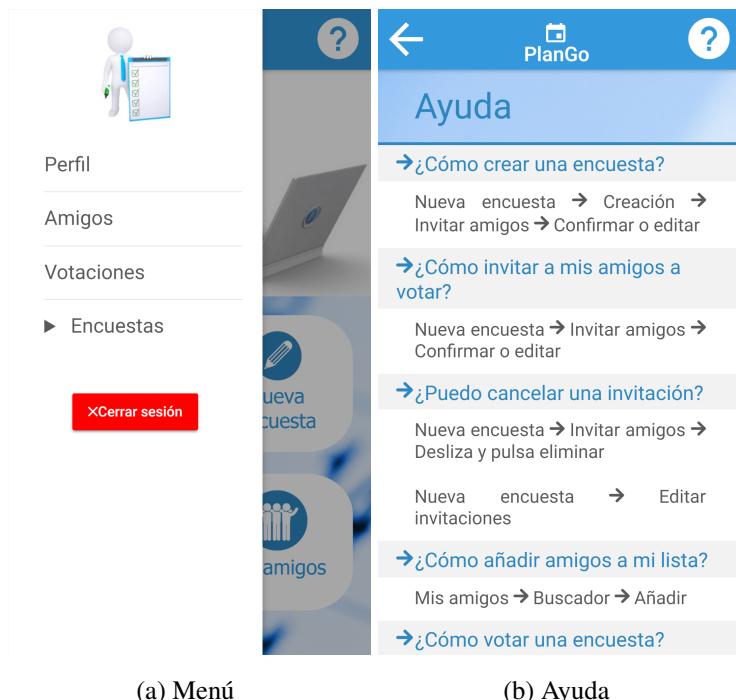


Figura 3.8: Pantalla de Menú y Ayuda

El header cuenta con 4 botones:

- Menú lateral: nos permite acceder a cualquier parte de la aplicación, figura 3.8a.
- Icono planGO: este ícono te dirige a la pantalla de Home.
- Icono ayuda: este ícono te dirige a la pantalla de Ayuda, figura 3.8b
- Icono flecha: este ícono te permite acceder a la pantalla anterior.

Como vemos en las figuras 3.7a y 3.7b el menu lateral y el icono flecha no se muestran a la vez. Esto va a depender de la pantalla en la que nos encontremos. Cuando se produce navegación jerárquica, aparece el icono atrás, para el resto de los casos aparece el icono de menú.

Mis encuestas



Figura 3.9: Pantalla de Mis encuestas

En esta pantalla el usuario tiene acceso a todas las encuestas que han sido creadas por él. Como vemos en la figura 3.9a, aparece un listado con todas las encuestas. Es posible hacer click en cualquiera de las encuestas para acceder a los votos que ha obtenido.

Si no hay votos para la encuesta seleccionada, se mostrara un mensaje de aviso. En caso de que haya votaciones, la pantalla mostrada varia en función si es de tipo texto o de tipo fecha.

Cuando la encuesta es de tipo texto, se muestra una tabla con las opciones que tiene disponibles la encuesta y un recuento de votos por usuario. Cuando la encuesta es de tipo fecha, aparece un calendario con los días que existen como opción remarcados. Estos días se pueden pulsar, para que aparezca el detalle de las horas a elegir, en caso de que las haya. Después hay una tabla

igual que la descrita anteriormente para tipo texto.

Debido a que la tabla crece horizontalmente según sea el número de opciones, esta tabla permite realizar scroll horizontal.

Mis amigos

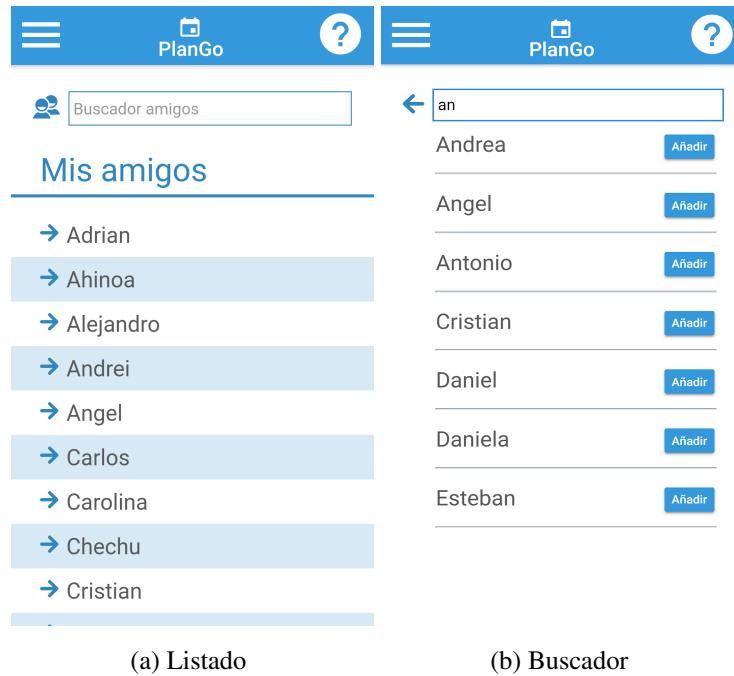


Figura 3.10: Pantalla de Mis amigos

Esta pantalla contiene un buscador en la parte superior y un listado, como se puede ver en la figura 3.10a.

El buscador permite encontrar usuarios, que tengan cuenta en la aplicación, introduciendo su nombre. Este filtro tiene autocompletar, de tal manera que el listado de búsqueda se va actualizando a la vez que el usuario va introduciendo caracteres. Esto es muy útil para el usuario, pues no tiene porque saber el nombre exacto del usuario que quiere buscar. Solo aparecen los usuarios que aún no tiene en su lista de amigos. En el caso de querer cerrar esta búsqueda, existe un botón con forma de flecha, para volver al estado anterior.

Para añadir un nuevo amigo, tras haber realizado su búsqueda, hay que pulsar en el botón

'añadir'.

En el listado se ven todos los usuarios que contiene en su lista de amigos.

Perfil

(a) Cambio nombre

(b) Cambio contraseña

Figura 3.11: Pantalla de Perfil

Esta pantalla contiene dos desplegables, cada uno de los cuales esta destinado a una función diferente. Es una pantalla bastante intuitiva como se ve en la fig.

En el desplegable 'Cambiar Username' el usuario podrá cambiar el nombre con el que esta identificado en la aplicación. En el caso del desplegable 'Cambiar contraseña' el usuario podrá cambiar su contraseña actual.

Para todos los cambios que se realicen en el perfil del usuario, por seguridad, es necesario que introduzca la contraseña actual. De esta manera solo el propio usuario conocedor de su contraseña podrá realizar cambios.

Nueva encuesta

Esta es la parte principal de la aplicación, la creación de la encuesta. Existen dos tipos de encuesta: por fecha o por texto. En una encuesta por fecha, se podrá votar sobre fechas y horas. En el caso de una encuesta por texto las votaciones se realizaran entre textos que pueden ser de cualquier tipo. Como se ve en la figura 3.12, esta elección se hace pulsando en el botón 'Nueva encuesta'.

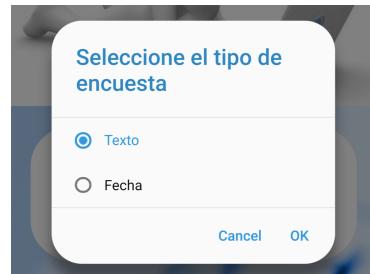


Figura 3.12: Tipo de encuesta

Una vez seleccionado el tipo de encuesta que se quiere crear, se accede a rellenar los datos necesarios para la encuesta como se ve en la figura. El formulario contiene varios campos, los cuales algunos son opcionales y otros no. Los campos requeridos son el título y las opciones, si esto no se cumple, como se ve en la figura, se mostrara el error. Dentro de los camposopcionales encontramos que se puede añadir una ubicación o un comentario. Además de esto, hay una opción que restringe la encuesta a un solo voto por votante.

Cena

Añadir opción

mi casa

restaurante

▶ Añadir más detalles

▶ Opciones de voto

Continuar

(a) Tipo texto

Reunión

Seleccionar fechas

03/01/2018

– Añadir hora

Hora

Día: 03-01-2018

Hora: 03:00 Hora: 00:21

▶ Añadir más detalles

▶ Opciones de voto

(b) Tipo fecha

Figura 3.13: Pantalla de Nueva encuesta: paso 1

Una vez completado el primer paso, pasamos al segundo. En este, el usuario tiene que elegir a que amigos enviar la encuesta, de esta manera, estos podrán realizar su voto.



Figura 3.14: Pantalla de Nueva encuesta: paso 2

En este paso encontramos un buscador y un listado. El buscador tiene la funcionalidad de facilitar al usuario la búsqueda de aquellos amigos a los que quiere enviar la encuesta. En el listado aparecen todos los amigos con un botón de 'Enviar'. Este botón envia la encuesta al usuario seleccionado. En la figura 3.14b se puede ver el resumen que se genera de todas las invitaciones realizadas. En el caso de querer cancelar una invitación es posible arrastrando sobre el nombre y pulsando en la papelera como se ve en la figura 3.14c.

Una vez completados estos pasos, en el último paso tenemos la posibilidad de modificar datos introducidos durante la creación de la encuesta o modificar las invitaciones a amigos.

Votaciones pendientes



Figura 3.15: Pantalla de Votaciones pendientes

Esta pantalla contiene dos tabs. Ambas contienen un listado de encuestas, pero están destinadas a distinta funcionalidad.

En la tab pendientes (figura 3.15a) se encuentran todas las encuestas que el usuario aún no ha votado. Para poder realizar el voto hay que pulsar encima del título de la encuesta. De esta manera se accede al detalle de la encuesta y sus opciones. Una vez decidido el voto, se pulsa en 'OSD'.

En la tab editar voto, aparecen las encuestas que ya han sido votadas. Esta pantalla permite modificar un voto ya enviado. Su funcionalidad es igual que en el caso de pendientes, al pulsar encima del título se accede a la votación.

3.1.3. Arquitectura

La arquitectura general es la de la figura 1.2 que ya ha sido explicada en el capítulo de introducción en la tecnología Ionic. Como ya hemos dicho, la parte en la que se lleva a cabo

todo el desarrollo es en la carpeta 'src', por esto es la parte en la que voy a entrar en detalle.

App

Existe una carpeta llamada 'app', en la cual lo más importante es:

- Una carpeta que contiene ficheros destinados a la declaración de variables para ser usadas en toda la aplicación. Esta serie de variables son utilizadas en los estilos de los componentes. Contienen los colores, tamaños de letra e iconos. Este es un ejemplo de buenas prácticas, si fuera necesario cambiar la apariencia de toda la aplicación, bastaría con modificar alguna de estas variables.
- Fichero app.component.ts: contiene la inicialización de la aplicación y configuración para elegir cuál será la pantalla de inicio.
- Fichero app.module.ts: tiene gran importancia ya que en él están declarados todos los componentes y servicios que han sido creados.
- Fichero app.css: contiene reglas de CSS que se aplican a todos los componentes.

Pages

En la carpeta 'pages' es donde se encuentran todos los componentes creados. Como se ha visto en el punto 1.2.6, Angular esta basado en componentes, que ha su vez pueden contener otros componentes. De esta manera la arquitectura conseguida es más estructurada.

Cada componente esta formado por tres ficheros: el typescript donde está la lógica, el html donde está la vista y el css que modifica la vista.

En este proyecto, hay pantallas que están formadas únicamente por un componente y otras que contienen más. La pantalla de login es la única que realmente contiene un único componente, en cambio Registro, Home y Perfil aparte de su propio componente, están formados por el componente común Header. En la figura 3.16, de forma esquemática, se puede ver qué componentes forman cada una de las pantallas.

La pantalla de Mis encuestas cuenta con dos diferentes, uno que muestra el listado de todas las encuestas creadas por el usuario logeado y otro para mostrar el detalle y sus votos. Este último contiene a su vez el componente de calendario, que sólo se muestra en el caso de ser una encuesta de tipo fecha.

La pantalla de Nueva encuesta está formada por cuatro componentes. Un componente principal que contiene las tabs y el control sobre estas, y los componentes de las pantallas del primer, segundo y tercer paso. El primer paso es la creación, el segundo la invitación y por ultimo la pantalla de editar.

La pantalla de Votaciones pendientes , tiene estructura similar a la de Nueva encuesta. Hay un componente principal que contiene las tabs y el control de estás, y los dos componentes que forman las pantallas de cada tab.

El componente Header, que es común para la mayoría de las pantallas, lo esta puesto en la figura aparte. Este contiene el menú lateral, que es un componente y la navegación a Ayuda, que también es un componente.

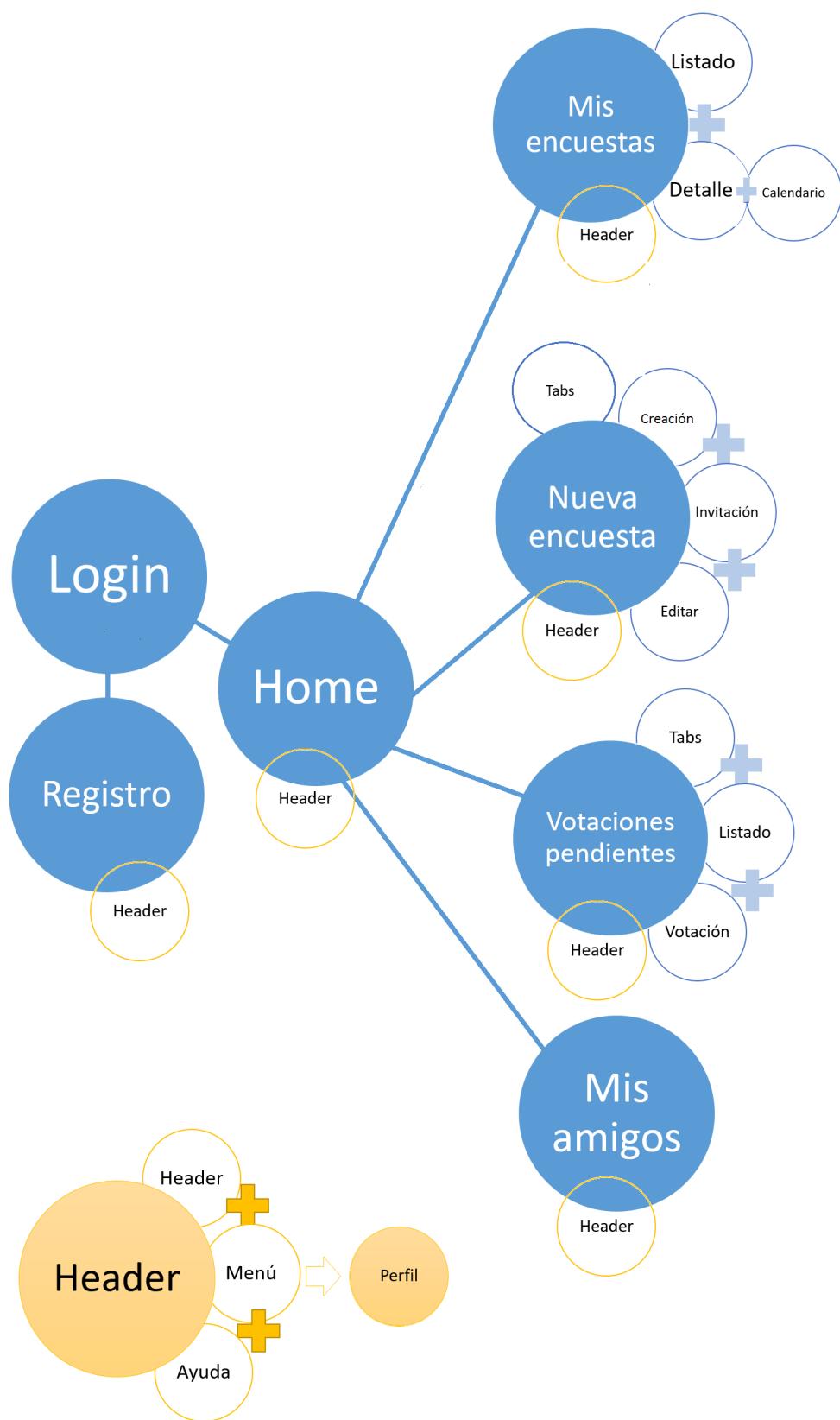


Figura 3.16: Estructura componentes

Shared

Esta carpeta ha sido creada para todos los servicios de la aplicación, componentes comunes y modelos de datos. Los servicios son los encargados de hacer la lógica de negocio, así como las peticiones al servidor. Los modelos son interfaces que definen qué atributos contiene un objeto, de esta forma se protege el código contra posibles fallos por mal tipado.

- Los componentes comunes y por lo tanto, que han sido reusados son el header, el menú y el calendario. En el caso del calendario, no es un componente común como tal, ya que solo es utilizado en una pantalla, pero ha considerado que debe estar en esta carpeta por estar totalmente preparado para ser rehusado.
- Los modelos son: friend, poll, user y vote. Es una forma de asegurarnos que las variables que creamos contienen exclusivamente lo que está declarado en el modelo. Esta es la gran ventaja que tiene TypeScript como ya se ha comentado.
- Existe un servicio para las peticiones al servidos de cada una de las pantallas. Además, hay un servicio que contiene toda la lógica de cómo realizar peticiones HTTP de tipo GET, PUT y POST. Esto facilita que en cada servicio solo sea necesario importar este servicio y después usarlo para las distintas peticiones que se necesitan.

3.1.4. Capa de servicios

En este apartado vamos a dejar de lado la parte de la vista de la aplicación para ir adentrandonos en la parte de servicios. Por un lado hay un proyecto para todo el desarrollo de la parte front y otro para la parte back, pero esto tiene que ser conectado de alguna manera. Una vez visto este paso intermedio, profundizaremos en la parte back en el apartado 3.2.

El cliente envía una petición HTTP, en este caso la aplicación. La petición es un mensaje de texto creado por un cliente en un formato especial. El cliente envía la petición a un servidor, y luego espera la respuesta. La primera línea de una petición HTTP es la más importante y contiene dos cosas: la URI y el método HTTP. La URI es la dirección o ubicación que identifica únicamente al recurso que solicita el cliente y el método HTTP define lo que quieres hacer

con el recurso.

En la figura 3.17 están representadas todas las peticiones que realiza la aplicación, separadas por pantallas. El método GET se utiliza para recuperar datos del servidor y transmite información a través de la URI agregandole parámetros. El método POST se usa para crear un recurso en el servidor y los datos se incluyen en el cuerpo de la petición.

- Login: solo es necesario consultar si el usuario y contraseña introducidos son correctos y existen en la base de datos. El método utilizado para ello es de tipo POST, ya que los detalles de email y contraseña se enviarán en el cuerpo de mensajes HTTP en lugar de en la URL.
- Registro: lo que se quiere es generar un recurso en el servidor, es decir, añadir en la base de datos un nuevo usuario. El método utilizado para ello es de tipo POST, en cuyo cuerpo lleva el nombre de usuario, email y contraseña.
- Mis encuestas: es necesario realizar una petición para recuperar todas las encuestas. Para ello se realiza una petición GET con el id del usuario como parámetro. Además de esta petición, a la hora de visualizar los votos que tiene determinada encuesta, se utiliza una petición GET con id de encuesta como parámetro.
- Nueva encuesta: a la hora de crear una encuesta, como ya ha sido explicado en el apartado 3.1.2, existen tres pasos diferentes, de los cuales, los dos primeros requieren peticiones HTTP. En el primer paso, en la creación de la encuesta, se usa una petición POST para que se añada la encuesta en la base de datos. En el cuerpo de esta petición, van el título, las opciones y todos los datos que definen la encuesta. Una vez creada la encuesta, en el segundo paso, hay que enviar la encuesta a los amigos. Para pintar el listado de amigos, es necesario traer estos datos del servidor mediante una petición GET con id de usuario como parámetro. Además se utilizan peticiones para añadir o borrar la encuesta al amigo seleccionado.
- Votaciones pendientes: para poder pintar el listado de votos pendientes o de editar votos, se realiza la petición GET correspondiente, que tendrá como parámetro el id del usuario. Para guardar cualquier voto, se hace mediante una petición POST con los votos en el

cuerpo. Si se realiza la votación de una encuesta que pertenece a los votos pendientes, desaparecerá de los votos pendientes y pasará a ser de la lista de editar votos. Todo esto se realiza con peticiones POST.

- Perfil: esta pantalla permite los cambios de usuario o de contraseña. Ambos se hacen con peticiones PUT, ya que solo se trata de actualizar los valores en la base de datos.
- Mis amigos: esta parte de la aplicación requiere de dos listados. Ambos son obtenidos mediante peticiones GET, cada una apuntando a la URL correspondiente.



Figura 3.17: Peticiones HTTP

3.2. Backend

Está enfocado sólo a lenguajes de programación, orientado a funcionamientos, trabajar con datos internos, crear aplicaciones que controlen datos de la base de datos de la web, para poder ser consultados por el usuario.

Toda la parte *backend* esta desarrollada con las tecnologías Express y MongoDB.

3.2.1. Servidor

Un servidor es un ordenador u otro tipo de equipo informático encargado de suministrar información a una serie de clientes, que pueden ser tanto personas como otros dispositivos conectados a él.

En este proyecto se han utilizado servicios REST, que significa 'Representational State Transfer', traducido 'transferencia de representación de estado'. La clave de REST es que un servicio REST no tiene estado lo que quiere decir que, entre dos llamadas cualesquiera, el servicio pierde todos sus datos.

El servidor esta desarrollado con un framwork llamado Express, cuyos detalles ya han sido explicados en el apartado 1.2.10.

Existe un fichero llamado 'app.js' para la creación del servidor y por lo tanto hacer llamadas HTTP. En este fichero se encuentran las dependencias necesarias. Tambien se encuentra que con HTTP se crea el servidor y se le indicamos en que IP y puerto estará escuchando. A esta dirección donde la parte del cliente enviará las peticiones. Todo esto es parte de la configuración necesaria para la creación de un servidor.

Una vez creado el servidor, se declaran las rutas con sus correspondientes métodos HTTP e instancias. Esto significa qué cuando el servidor reciba una petición con alguna de las rutas declaradas y con el correcto método, se lleva a cabo la instancia que está indicada para dicha ruta. En la figura 3.18 se puede ver un ejemplo de como se crean las rutas.

Aparte de esto, falta una parte importante, la conexión con la base de datos, en este caso con MongoDB.

En los siguientes apartados, se detallan los modelos y controladores, estos también son importados en el fichero 'app.js' para su uso.

```
//// POLLS ROUTE /////////////////////////////////
var polls = express.Router();

polls.route('/add')
    .post(pollCtrl.addPoll);

polls.route('')
    .get(pollCtrl.findAllPolls);

app.get('/poll/:id', pollCtrl.searchPolls)

app.use('/poll', polls);
```

Figura 3.18: Ejemplo: ruta encuestas

3.2.2. Modelos

Los modelos son creados usando Mongoose para poder guardar la información en la base de datos siguiendo el esquema. Como base de datos se ha utilizado MongoDB. Al ser una base de datos Open Source NoSQL orientada a documentos tipo JSON viene bien para entregar los datos en este formato en las llamadas a la API.

Los modelos que han sido creados son:

- User: contiene la información necesaria del usuario: el nombre, email y contraseña.
- Poll: contiene la información de una encuesta: título, ubicación, comentario, si solo es posible un voto, las opciones, el tipo de encuesta y el id del usuario que la ha creado. Ver figura 3.19
- Friend: contiene el id del usuario y los usuarios que son sus amigos.
- Vote: contiene el id de la encuesta y las opciones con sus respectivos votos.
- Pending-vote: contiene un id de usuario y las encuestas que tiene dicho usuario por votar.
- Send-vote: contiene un id de usuario y las encuestas dicho usuario ya ha votado.

Una vez creados los modelos, se puede implementar la conexión a la base de datos en el archivo antes explicado, 'app.js'.

```
var mongoose = require('mongoose'),
    Schema   = mongoose.Schema;

var pollSchema = new Schema({
  title:    { type: String },
  ubication:{ type: String },
  commentary: { type: String },
  oneVote:   { type: Boolean },
  possibilities : {type: Object},
  type: {type: String},
  idUser: {type: String}
});

module.exports = mongoose.model('poll', pollSchema);
```

Figura 3.19: Ejemplo: modelo encuesta

3.2.3. Controladores

La parte de controlador tiene mucho que ver con el enrutamiento y con lo que en Express llaman *middleware*: funciones que reciben una entrada (request) y una salida (response). Lo que sucede entre la entrada y la salida es el controlador.

Los controladores de las rutas de la API están creados en un archivo separado que se llama 'controllers'. Estos controladores son exportados para poder modularizarlos y que puedan ser llamados desde el archivo principal de la aplicación.

Existe un controlador por cada modelo de datos, en cada uno de los cuales existen funciones para llevar a cabo diferentes interacciones con la base de datos. Las interacciones suele ser que añadir datos, actualizarlos o eliminarlos.

En todos los controladores existen unas funciones que son comunes para todas pero con sus correspondientes peculiaridades:

- 'FindAll...': realiza una búsqueda por la colección de la base de datos que se le haya indicado. Devuelve todos los datos encontrados, sin ningún filtro.
- 'FindByID': como su propio nombre indica, realiza una búsqueda por la colección de la

base de datos que se le haya indicado y devuelve solo aquellos datos filtrados por un ID.

- 'Add...': función utilizada para añadir datos a la colección indicada. Esta función no coincide en todos los controladores. En el caso del controlador para usuarios, antes de añadir un nuevo usuario se comprueba que no haya ningún usuario con el mismo nombre ni con el mismo email. En los demás controladores, se comprueba si el dato a añadir ya existe en la base de datos, para actualizarlo o añadirlo.
- 'Search...': realiza una búsqueda por la base de datos. En cada uno de los controladores dicha búsqueda se realiza por el dato o datos que representan el modelo de datos. En el caso del controlador de usuarios se busca por email y contraseña y en el resto se busca por el id de usuario.
- 'Update...': como su nombre indica, sirve para actualizar datos en la base de datos sobre algo ya almacenado.
- 'Delete...': permite eliminar datos de la base de datos.

```
//GET - Return all users in the DB
exports.findAllUsers = function (req, res) {
    User.find(function (err, users) {
        if (err) res.send(500, err.message);

        console.log('GET /users')
        res.status(200).jsonp(users);
    });
};
```

Figura 3.20: Ejemplo: función controlador

Siempre que se quiera trabajar con los datos de una base de datos debe seguir una nomenclatura definida. Primeramente aparecerá el nombre de la colección sobre la que se quiere trabajar y posteriormente el método a usar. Los métodos usados para filtrar han sido findById para filtrar por id, findOne para buscar un dato concreto. Para introducir o eliminar datos se han usado otros métodos: 'save' y para borrar 'delete'.

3.2.4. Base de datos: MongoDB



Figura 3.21: Arquitectura MongoDB

La jerarquía principal de MongoDB viene dada por los elementos presentados en la figura 3.21. Utiliza documentos con distintas estructuras. Un conjunto de Campos formarán un Documento, que en caso de asociarse con otros formará una Colección. Las bases de datos estarán formada por Colecciones, y a su vez, cada servidor puede tener tantas bases de datos como el equipo lo permita.

En este caso solo hay una base de datos, que recibe el nombre de encuestasApp. Esta base de datos tiene seis colecciones que contienen una serie de campos diferentes, en función de las necesidades de dicha colección. Estas funcionalidades se verán detalladamente en el apartado 3.2.6.

En la figura 3.22, se puede ver en detalle como esta formada la base de datos. Aparecen todas las colecciones y en detalle la de 'Users'. Esta colección almacena Documentos con los campos de usuario, email y contraseña, además añade campo de id que identifica este documento en la base de datos y campos propios de la encriptación.



Figura 3.22: Ejemplo colección 'Users'

3.2.5. Colecciones

Una colección, como ya se ha explicado, es un contenedor de los documentos en MongoDB. No es necesaria su creación datos por primera vez, ya que Mongo comprueba previamente si existe la colección y si no, la crea automáticamente.

Todos los documentos tienen el campo de 'id', este campo es creado por Mongo para poder identificar cada documento y 'v' que hace referencia a la versión.

Users

Esta colección contiene documentos con los campos de nombre, email, contraseña. Además, también existen el campo 'encpassword', que nos permite saber si la contraseña está o no encriptada.

La principal función de esta colección es almacenar todas las cuentas de usuario que se han creado en la aplicación. Solo aquellos usuarios que tengan cuenta podrán acceder, introduciendo su email y contraseña. Además de esto, nos da un identificador que es único para cada usuario. Este id es muy importante a la hora de relacionar los datos de esta colección con otra.

Polls

Esta colección contiene documentos con los campos de título, ubicación, comentario, solo un voto, tipo, opciones e id de usuario.

Todas las encuestas que son creadas a través de la aplicación son almacenadas en esta colección. Para saber quien es el creador de la encuesta, cada documento contiene el campo 'idUser', este coincide con uno de los id de la colección de 'Users'. De esta manera tenemos relacionadas ambas colecciones.

Friends

Esta colección contiene documentos con los campos de id de usuario e id de amigos.

El id de usuario se corresponde a un id de la colección de 'Users' y nos permite conocer a qué usuario estamos haciendo referencia. El id de amigos se trata de un array de ids, todos ellos de la colección de 'Users'. De esta forma, cuando se necesita conocer qué amigos tiene un determinado usuario, solo es necesario filtrar esta colección por su id y, a partir de su array id de amigos y la colección 'Users', conocer los detalles de sus amigos.

Votes

Esta colección contiene documentos con los campos de id de encuesta y un array de opciones. Dentro de este array, hay otro array que contiene los voto.

El id de encuesta se corresponde a un id de la colección de 'Polls' y nos permite saber a qué encuesta estamos haciendo referencia. El array de opciones contiene todas las opciones de la encuesta. Por cada opción hay un array en el que se almacena quien la ha votado.

Pendings

Esta colección contiene documentos con los campos de id de usuario e id de encuesta.

La funcionalidad de esta colección es conocer que encuestas tiene el usuario pendientes de votar. Con el id de usuario se conoce el usuario que tiene que votar y con el id de encuesta la encuesta a votar. Accediendo a la colección 'Polls' con el id se pueden conocer los detalles de la encuesta.

Send

Esta colección contiene documentos con los campos de id de usuario e id de encuesta.

La funcionalidad de esta colección es conocer que encuestas ha votado ya el usuario. Con el id de usuario se conoce el usuario que ha votado y con el id de encuesta la encuesta. Accediendo a la colección 'Polls' con el id se pueden conocer los detalles de la encuesta.

3.2.6. Relación entre colecciones

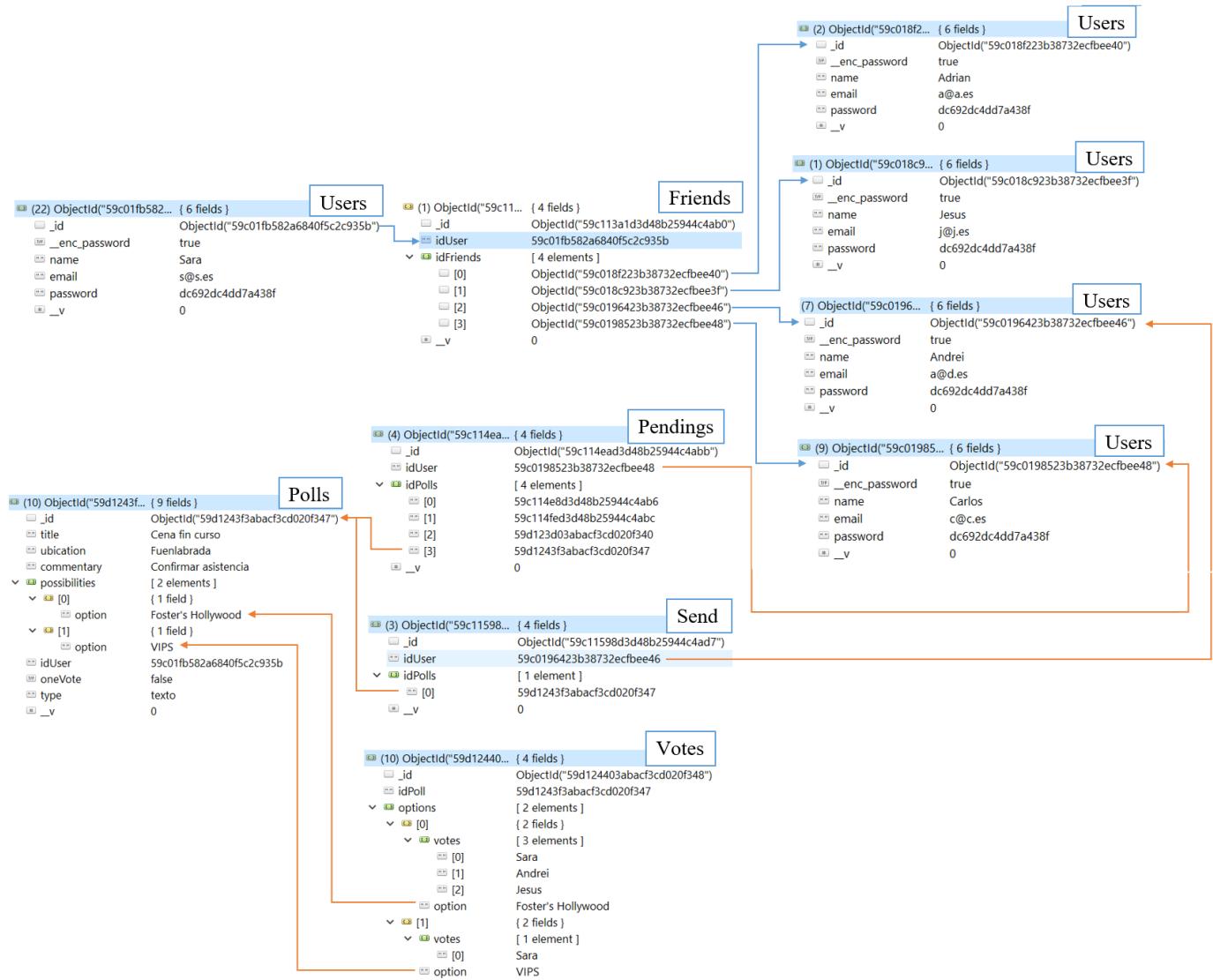


Figura 3.23: Relación entre colecciones

La mejor manera de entender las colecciones descritas en el apartado 3.2.6 es saber las relaciones que hay entre ellas. Algunas de las colecciones tienen la funcionalidad únicamente de relacionar dos colecciones. Es el caso de 'Friends', 'Pendigns' y 'Send'. Esto se consigue gracias a los identificadores únicos de cada documento.

Esta forma de relacionar datos tiene un inconveniente, la posible inconsistencia de datos. Una base de datos está inconsistente si dos datos que deberían ser iguales no lo son. Esto puede ocurrir si para una funcionalidad se necesita modificar dos colecciones distintas y, por algún

error, solo se realiza en una. Al tratarse de una base de datos no relacional, para solucionar esta inconsistencia, una colección debería contener todos los datos. Pero esto no sería eficiente, las operaciones con la base de datos llevarían más tiempo, incluso llegando a ser imposible su uso si el número de documentos es muy grande.

3.2.7. Seguridad

De los datos almacenados en la base de datos, la autenticación es lo más importante de cara a la seguridad. Por esto he considerado necesario guardar la contraseña encriptada. De esta manera, aunque alguien acceda a la base de datos, no podrá obtener la contraseña real.

La encriptación está hecha mediante un plugin de mongoose llamado `mongoose-field-encryption`. Ofrece un cifrado simétrico para campos individuales. Es necesario indicar una contraseña que permite en encriptado y desencriptado. El inconveniente de este método es que si alguien consiguiera esta contraseña tendría acceso a todas las contraseñas.

```
var mongoose = require('mongoose'),
    Schema   = mongoose.Schema;
mongooseEncryption = require('mongoose-field-encryption').fieldEncryption;

var userSchema = new Schema({
  name:     { type: String },
  email:   { type: String },
  password: { type: String },
});

userSchema.plugin(mongooseEncryption, {fields: ['password'], secret: '*****'});

module.exports = mongoose.model('user', userSchema);
```

Figura 3.24: Encriptación

Capítulo 4

Prueba

En esta última fase del proyecto, he realizado una prueba de usuarios con la última versión de la aplicación desarrollado. El propósito de esta es comprobar la experiencia de los usuarios, verificar los objetivos que han sido logrados y que no existen fallos en la aplicación.

El proceso de beta testing es el último paso necesario para disfrutar de una aplicación que realmente funciona dentro de cualquiera de las tiendas de aplicaciones de las grandes tecnológicas. En Android el sistema para la subida de una versión de prueba se llama alpha y betta, pero en el caso de iOS se llama TestFlight. Esto es lo que estamos realizando con esta prueba final del proyecto, sin utilizar las plataformas oficiales de Android y Apple.

4.1. Objetivos de la prueba

El objeto principal de la prueba es la obtención de resultados sobre la usabilidad y funcionalidad de los usuarios con la aplicación. Para que una aplicación sea de interés, es importante que cumpla dichos requisitos.

Para ello se determinan si se cumplen los siguientes objetivos:

- La aplicación debe cumplir la funcionalidad descrita en los requisitos establecidos inicialmente en la sección 2.

- El usuario debe intuir con facilidad como debe utilizar la aplicación, así como la navegación.

4.2. Desarrollo

Para poder desarrollar la prueba ha sido necesario que cada usuario contara con un dispositivo móvil. He instalado la aplicación en cada uno de los dispositivos, con la configuración correcta para que conecte con el servidor. Dado que la parte *backend* no está corriendo en ningún servidor real, he utilizado el propio ordenador como servidor.

Una vez acabado esto y con la aplicación preparada para funcionar, se ha empezado con la prueba. Esta está formada por varias partes. En una primera parte he considerado que sería interesante dejar a los usuarios navegar por la aplicación para así valorar como es de intuitiva. En una segunda fase, he marcado una serie de hitos a realizar:

1. Crear una cuenta.
2. Logearse.
3. Navegar por todos los accesos de la aplicación para comprobar que no contienen ningún dato.
4. Añadirse como amigos.
5. Crear una encuesta e invitar a los amigos.
6. Votar las encuestas pendientes
7. Cambiar datos del perfil.

En la tercera y última fase, han tenido que llenar una encuesta en la que debían poner una puntuación sobre diez a algunos puntos de la aplicación. Con los resultados de estas preguntas he conseguido llegar a unas conclusiones.

4.3. Conclusiones de la prueba

Una vez realizada la prueba, he considerado que los puntos más importantes sobre una aplicación son: el diseño (como de atractivo parece el aspecto visual), la usabilidad (si la aplicación funciona de forma fluida), la utilidad (si la usarían o no) y la intuitividad (manejo de la app sin ayuda).

Gracias a la primera fase de la prueba, los usuarios pueden reflexionar sobre como de intuitiva es la aplicación. Respecto a los demás puntos de interés, los usuarios lo han podido sacar sobre el desarrollo de toda la prueba.

En la figura 4.1 se pueden ver las puntuaciones que ha dado cada usuario. Dado que el número de usuarios que han realizado la prueba no es alto, los resultados obtenidos no son demasiado relevantes, pero si son suficientes para obtener una idea sobre la opinión que pueden tener los usuarios sobre la aplicación.

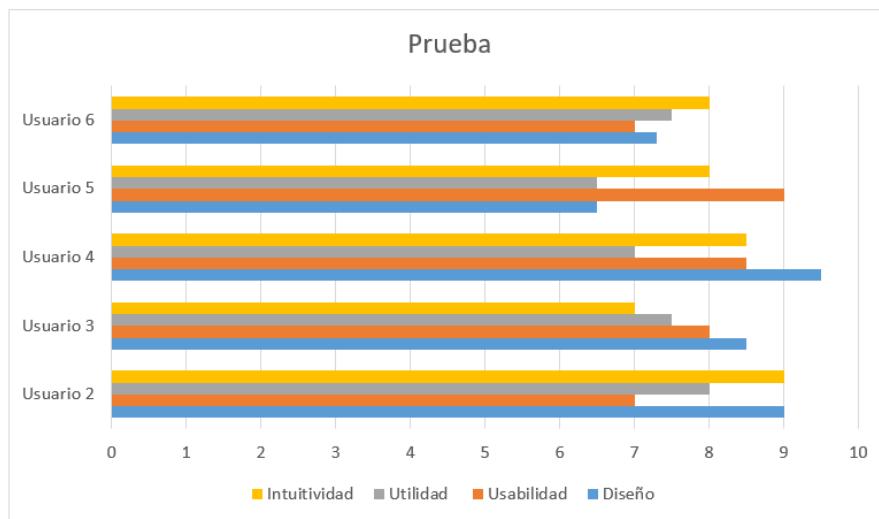


Figura 4.1: Resultado prueba

Como conclusión se obtiene que los puntos fuertes de la aplicación son el diseño y la intuitividad. El punto a mejorar sería la utilidad, aunque aún así tampoco tiene mal puntuación. Creo que al ser una aplicación con una funcionalidad ya conocida por los usuarios, no llama la atención especialmente por lo que ha podido repercutir en la nota sobre este punto. En cuanto a la usabilidad, creo que la nota obtenida es bastante correcta.

Tras este estudio, creo que sería interesante pensar una funcionalidad que sorprenda al usuario y que no este acostumbrado a ver.

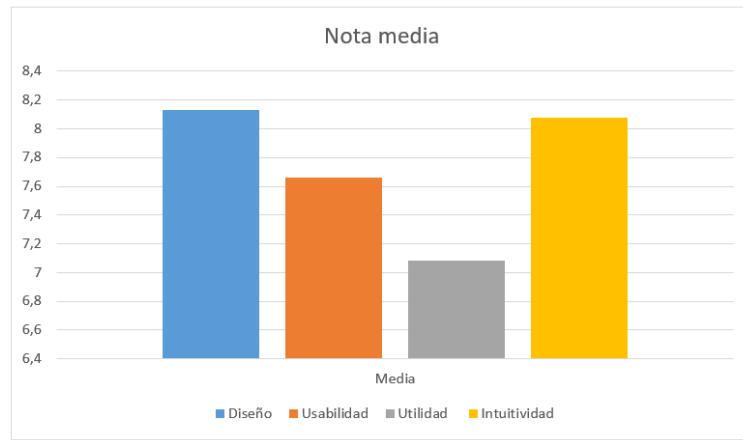


Figura 4.2: Media resultado prueba

Capítulo 5

Conclusiones

5.1. Aplicación de lo aprendido

Durante el grado de Ingeniería en Sistemas de Telecomunicación, solo he tenido una asignatura relacionada con este proyecto. Esta asignatura se llama Servicios y Aplicaciones de Ordenadores. Este proyecto ha sido inspirado en esta asignatura, ya que me pareció tan interesante que he querido profundizar en ello.

He aplicado la base de desarrollo web como HTML y CSS aprendidos en dicha asignatura.

5.2. Lecciones aprendidas

Durante el desarrollo de Trabajo Fin de Grado he aprendido:

1. Lenguaje programación JavaScript y TypeScript.
2. Ampliar los conocimientos de CSS.
3. Desarrollo en Angular e Ionic.
4. Creación de un servidor mediante el framework Express.
5. Utilización base de datos MongoDB.

6. Instalación de un apk en un dispositivo.

5.3. Trabajos futuros

Como cualquier aplicación, existen varias versiones de la misma. Esto se debe a que hay mejoras continuas y corrección de errores. En las stores de app es posible valorar cada aplicación y añadir quejar y sugerencias, es la mejor manera de que la aplicación evolucione.

A parte de mejoras en la usabilidad que usuarios puedan pedir, como posible trabajo futuro, sería interesante añadir más seguridad a la base de datos. Al tratarse de una aplicación cuyos datos almacenados no son de gran importancia, he considerado que era más importante desarrollar más el proyecto en otros ámbitos.

Sería interesante pensar algo que haga la aplicación única y capte la atención de los usuarios, como hemos visto en el apartado 4.3.

Apéndice A

Manual de usuario

Bibliografía