



INGENIERÍA EN SISTEMAS DE TELECOMUNICACIÓN

Curso Académico 2017/2018

Trabajo Fin de Grado

PLANGO

Autor : Sara López Zambrano

Tutor : Pedro de las Heras

Trabajo Fin de Grado

planGO

Autor : Sara López Zambrano

Tutor : Dr. Pedro de las Heras

La defensa del presente Proyecto Fin de Carrera 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 / mi abuelo / mi abuela*

Agradecimientos

Aquí vienen los agradecimientos. . . Aunque está bien acordarse de la pareja, no hay que olvidarse de dar las gracias a tu madre, que aunque a veces no lo parezca disfrutará tanto de tus logros como tú. . . Además, la pareja quizás no sea para siempre, pero tu madre sí.

Resumen

Aquí viene un resumen del proyecto. Ha de constar de tres o cuatro párrafos, donde se presente de manera clara y concisa de qué va el proyecto. Han de quedar respondidas las siguientes preguntas:

- ¿De qué va este proyecto? ¿Cuál es su objetivo principal?
- ¿Cómo se ha realizado? ¿Qué tecnologías están involucradas?
- ¿En qué contexto se ha realizado el proyecto? ¿Es un proyecto dentro de un marco general?

Lo mejor es escribir el resumen al final.

Índice general

1. Introducción	1
1.1. Historia de la web	1
1.2. Trabajo relacionado	3
1.3. Tecnologías	4
1.3.1. HTML5	4
1.3.2. JavaScript	6
1.3.3. TypeScript	6
1.3.4. CSS3	7
1.3.5. Sass	8
1.3.6. Angular	8
1.3.7. Ionic 2	11
1.3.8. Node.js	13
1.3.9. Express	13
1.3.10. MongoDB	14
1.4. Estructura de la memoria	15
2. Objetivos	17
2.1. Objetivo general	17
2.2. Objetivos específicos	17
2.2.1. Frontend	17
2.2.2. Backend	18
2.3. Planificación temporal	18

3. Diseño e implementación	21
3.1. Frontend	21
3.1.1. Mapa de navegación	21
3.1.2. Funcional	23
3.1.3. Arquitectura	28
4. Resultados	33
5. Conclusiones	35
5.1. Consecución de objetivos	35
5.2. Aplicación de lo aprendido	35
5.3. Lecciones aprendidas	35
5.4. Trabajos futuros	36
A. Manual de usuario	37
Bibliografía	39

Índice de figuras

1.1. Estructura ionic	12
1.2. Estructura express	14
3.1. Mapa navegación	22
3.2. Pantalla de login	23
3.3. Header	24
3.4. Pantalla de Mis amigos	25
3.5. Nueva encuesta: paso 2	27
3.6. Pantalla de Votaciones pendientes	28
3.7. Carpeta help	29
3.8. Carpeta home	30
3.9. Carpeta list	30
3.10. Carpeta login	30
3.11. Carpeta my-friends	31
3.12. Carpeta newPoll	31
3.13. Carpeta profile	31
3.14. Carpeta vote	32

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 solo 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 e incluso realizar operaciones desde cualquier lugar. Existen aplicaciones para todos los gustos, necesidades e intereses.

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 habrá 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. Trabajo relacionado

Este proyecto está 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.

1.3. Tecnologías

1.3.1. 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 páginaWeb 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 navegación, 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 sitio web.
- 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.

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

1.3.3. 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 debuggear mas sencillo. Por último añadir que genera un código modular.

1.3.4. 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 aun continúan usando trucos diversos para conseguir efectos tan comunes o tan deseados como los bordes redondeados o el sombreado de elementos en la página.

1.3.5. 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 mixins permiten una solución limpia a las repeticiones de código, así como una forma fácil de alterar el mismo.

1.3.6. 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, asincrónamente 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 centrarlo mas en el uso real de la aplicaciopn: responder a eventos, tareas asincronas y flujos con esas tareas asincronas, 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 has 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 , TypeScript es más interesante que el uso de JavaScript ya que permite tipado estático, hacer genéricos y decorators.

Funcionalidades

- Modularización

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

- Componentes

Pieza de código que controla una parte de la vista. No es nada mas que una clase de ES6 añadiendole 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 esta 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 decorato llamado `Injstable`. 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.

1.3.7. 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 ve 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

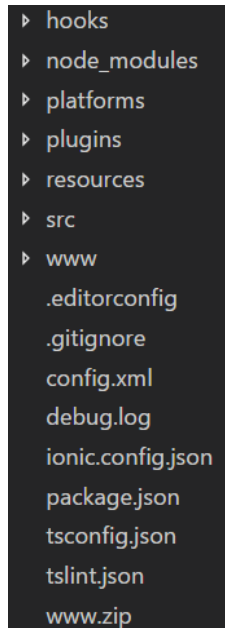
Apache Cordova

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:



```
▶ hooks
▶ node_modules
▶ platforms
▶ plugins
▶ resources
▶ src
▶ www
.editorconfig
.gitignore
config.xml
debug.log
ionic.config.json
package.json
tsconfig.json
tslint.json
www.zip
```

Figura 1.1: 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.

1.3.8. 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 pool de threads. La librería que construye esto es Libuv, una vez que el trabajo ha sido completado emite un evento recibido por Node.js.

1.3.9. 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 que 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 que método realizar para cada caso.

Estructura

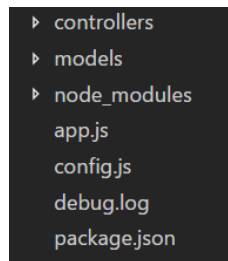


Figura 1.2: Estructura express

1.3.10. 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 compartan 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 necesario 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.

1.4. 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.

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 fácil usabilidad.

Por último se cuentan las conclusiones 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
- Trás 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.

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.

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 ha tiempo 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 dio 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

3.1. Frontend

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.

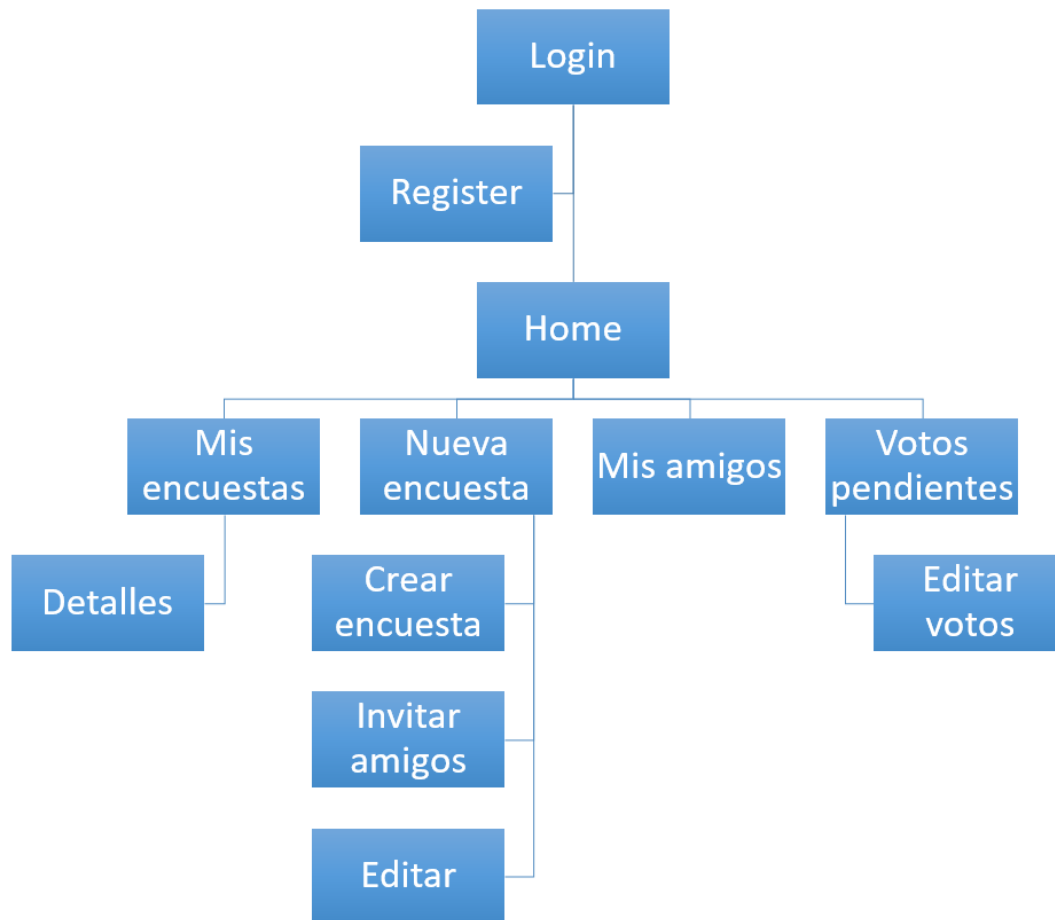


Figura 3.1: Mapa navegación

3.1.2. Funcional

Login



Figura 3.2: 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 válido.

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

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 'Regístrate'.

- Usuario: mediante una petición al servidor, se comprueba que no hay ningún usuario con este nombre.
- Email: se comprueba que sea un formato válido 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 más importante es identificar quien está votando y a quien va dirigida la encuesta.

Home

En esta pantalla se encuentran accesos directos a distintas partes de la aplicación. El usuario puede acceder a las encuestas que ha creado, 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 3.3: Header

El header cuenta con 4 botones:

- Menú lateral: nos permite acceder a cualquier parte de la aplicación. FOTO

- Icono planGO: este icono te dirige a la pantalla de Home.
- Icono ayuda: este icono te dirige a la pantalla de Ayuda. FOTO
- Icono flecha: este icono te permite acceder a la pantalla anterior.

Como vemos en la 3.3a y 3.3b 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

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

Mis amigos

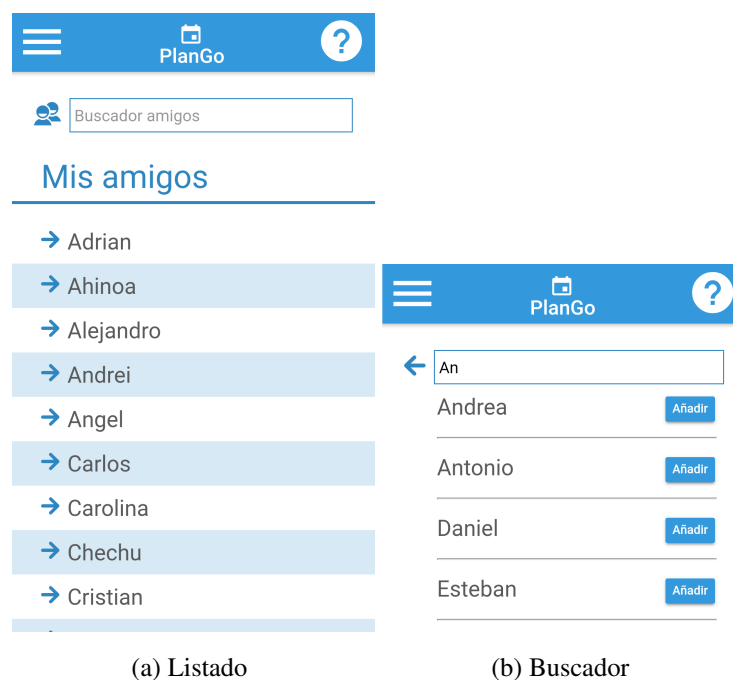


Figura 3.4: Pantalla de Mis amigos

Esta pantalla contiene un buscador en la parte superior y un listado, como se puede ver en la figura 3.4a.

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 que 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

Esta pantalla contiene dos desplegables, cada uno de los cuales está 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 está 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 realizarán entre textos que pueden ser de cualquier tipo. Como se ve en la figura, esta elección se hace pulsando en el botón 'Nueva 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 mostrará el error. Dentro de los campos opcionales



Figura 3.5: Nueva encuesta: paso 2

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.

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.

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 envía la encuesta al usuario seleccionado. En la figura 3.5b 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 3.5c.

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.

FOTO

Votaciones pendientes



Figura 3.6: 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 3.6a) 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 1.1 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.

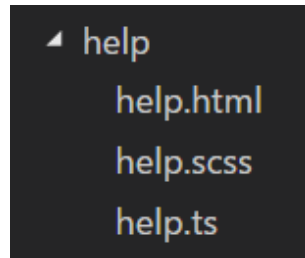


Figura 3.7: Carpeta help

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.

En la carpeta 'pages' es donde se encuentran todos los componentes creados. Como se ha visto en el punto 1.3.6, Angular está basado en componentes, que a su vez pueden contener otros componentes. De esta manera la arquitectura conseguida es más estructurada. Hay pantallas que solo contienen un componente y otras que están formadas por varios.

pages

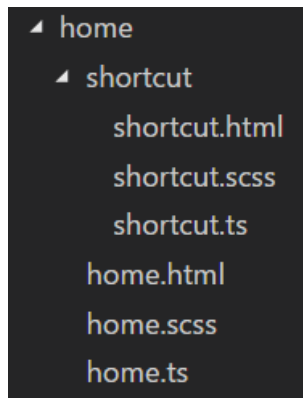


Figura 3.8: Carpeta home

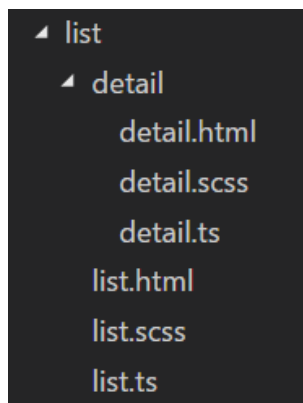


Figura 3.9: Carpeta list

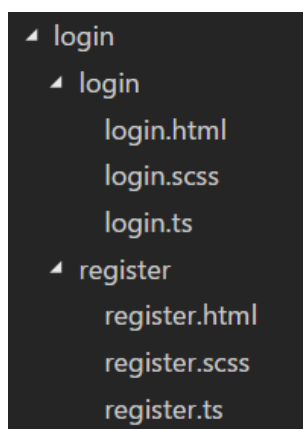
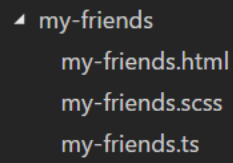
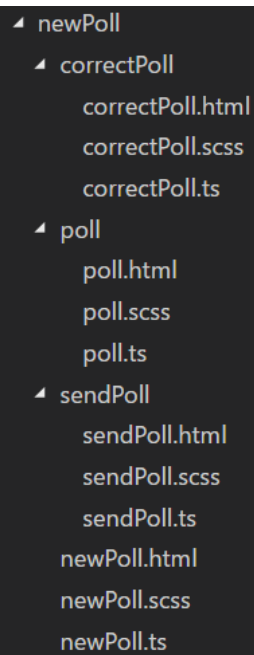


Figura 3.10: Carpeta login



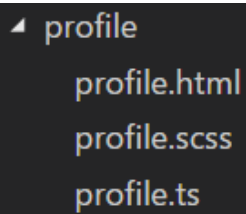
```
└─ my-friends
   ├── my-friends.html
   ├── my-friends.scss
   └── my-friends.ts
```

Figura 3.11: Carpeta my-friends



```
└─ newPoll
   ├── correctPoll
   │   ├── correctPoll.html
   │   ├── correctPoll.scss
   │   └── correctPoll.ts
   ├── poll
   │   ├── poll.html
   │   ├── poll.scss
   │   └── poll.ts
   ├── sendPoll
   │   ├── sendPoll.html
   │   ├── sendPoll.scss
   │   └── sendPoll.ts
   ├── newPoll.html
   ├── newPoll.scss
   └── newPoll.ts
```

Figura 3.12: Carpeta newPoll



```
└─ profile
   ├── profile.html
   ├── profile.scss
   └── profile.ts
```

Figura 3.13: Carpeta profile

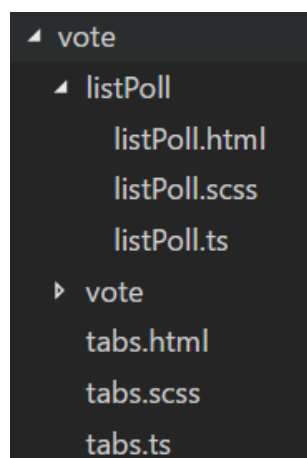


Figura 3.14: Carpeta vote

Capítulo 4

Resultados

En este capítulo se incluyen los resultados de tu trabajo fin de grado.

Si es una herramienta de análisis lo que has realizado, aquí puedes poner ejemplos de haberla utilizado para que se vea su utilidad.

Capítulo 5

Conclusiones

5.1. Consecución de objetivos

Esta sección es la sección espejo de las dos primeras del capítulo de objetivos, donde se planteaba el objetivo general y se elaboraban los específicos.

Es aquí donde hay que debatir si se ha conseguido y si no. Cuando algo no se ha conseguido, se ha de justificar, en términos de qué problemas se han encontrado y qué medidas se han tomado para mitigar esos problemas.

5.2. Aplicación de lo aprendido

Aquí viene lo que has aprendido durante el Grado/Máster y que has aplicado en el TFG/TFM. Una buena idea es poner las asignaturas relacionadas y comentar en un párrafo los conocimientos y habilidades puestos en práctica.

1. a
2. b

5.3. Lecciones aprendidas

Aquí viene lo que has aprendido en el Trabajo Fin de Grado/Máster.

1. a

2. b

5.4. Trabajos futuros

Ningún software se termina, así que aquí vienen ideas y funcionalidades que estarían bien tener implementadas en el futuro.

Es un apartado que sirve para dar ideas de cara a futuros TFGs/TFM.

Apéndice A

Manual de usuario

Bibliografía