ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
INGENIERÍA INFORMÁTICA

MEETNEAR. UNA APP MÓVIL PARA RELACIONARSE EN UN ENTORNO NUEVO

MEETNEAR. MOBILE APP TO SOCIALIZE IN A NEW ENVIRONMENT

Realizado por
Robin Soerries
Tutorizado por
Dr. D. Vicente M. Arévalo Espejo
Departamento
Ingeniería de Sistemas y Automática

UNIVERSIDAD DE MÁLAGA
MÁLAGA, Noviembre 2018

Fecha defensa:
El Secretario del Tribunal

Resumen: En la actualidad cada vez más personas por motivos de trabajo o personales se ven obligados a cambiarse de ciudad o incluso de país, esto conlleva la difícil tarea de hacer amistades en sus nuevas localidades. Tras una mudanza y los nuevos retos a los que hay que enfrentarse hace que el tiempo disponible para hacer amistades sea muy reducido.

Recién llegado a un lugar nuevo, se puede hacer uso de diferentes ofertas que suelen facilitar este asunto Tándem en diferentes locales (lo cual supone un coste), buscar en diferentes redes sociales (*Facebook*, *Tinder*, *Instagram*) ,etc. El uso de este tipo de redes sociales, aunque pueden ser una interesante alternativa, pueden resultar un auténtico problema dado que están orientadas a crear o fomentar relaciones virtuales y/o bien para encontrar pareja.

Ya que yo mismo me he encontrado con esta situación, y he visto la necesidad de interactuar con alguien fuera del entorno laboral, me ha hecho intentar solucionar este problema con esta aplicación. Debido a que la APP mostrará de forma directa los usuarios con los que podría interactuar de forma digital e incluso personal.

Palabras claves: aplicación, app, móvil, Android, Windows, iOS, web, Angular, geolocalización, personas, chat.

Abstract: Today more and more people for work or personal reasons are forced to change city or even country, this involves the difficult task of making friends in their new locations. After a move and the new challenges that have to be faced, the time available to make friends is very limited.

Newly arrived at a new place, you can make use of different offers that tend to facilitate this issue as Tandem in different locations (which involves a cost), search in different social networks (*Facebook*, *Tinder*, *Instagram*), etc. The use of this type of social networks, although they can be an interesting alternative, can be a real problem since they are oriented to create or encourage virtual relationships and / or to find a partner.

Since I have encountered this situation myself, and I have seen the need to interact with someone outside the work environment, it has made me try to solve this problem with this application. Because the APP will directly show the users with whom it could interact digitally and even personally.

Keywords: application, app, mobile, Android, Windows, iOS, web, Angular, geolocation, people, chat.

Índice

Capítulo 1. Introducción

1.1. Motivación

Nos encontramos en una era en la que la tecnología está creciendo, avanzando e incorporándose en nuestras vidas a una velocidad vertiginosa. Con la llegada del internet, nuestra forma de comunicarnos, socializar y vivir ha cambiado de forma drástica, cuando hace unos pocos años aún acudíamos a grandes superficies a realizar nuestras compras, hoy en día ni salimos de casa con un solo clic tenemos todo lo que deseamos, nos informamos a través de medios digitales, apenas se utiliza ya un periódico físico, y conocemos nuestras parejas por la red y no por pubs y discotecas.

Partiendo que en 1996 se calculó por primera vez el número total de usuarios de internet, y este arrojó una cifra de 40 millones, en 2018 estamos hablando ya de unos 4.000 millones de usuarios, un crecimiento del 1000% en 22 años.

Llegados a este punto, internet no es una simple herramienta de intercambio de información, sino una herramienta muy compleja y sofisticada para la comunicación, creación de contenido e incluso inmersión digital a través de la realidad aumentada. Y para todo esto ni siquiera es necesario poseer complejos equipos, con dispositivos tan diminutos como pueden ser nuestros teléfonos móviles, somos capaces de enviar y recibir información de una punta del mundo a la otra en menos de un segundo.

Este crecimiento también se ha visto afectado positivamente en los últimos años debido a un hito logrado en 2014, cuando por primera vez el uso del teléfono móvil superó al de equipos de sobremesa, y en parte es debido a las velocidades que podemos alcanzar con estos que en ocasiones gracias a las redes 3G/4G y futura 5G incluso superan las conexiones físicas en el domicilio.

A día de hoy tenemos toda la información, en la palma de nuestra mano, y es por esto por lo que la inmensa mayoría de redes sociales, chats, herramientas de comunicación apuestan por éstos mismos. Hay una gran cantidad de aplicaciones que ayudan a por en contacto a personas, ofreciendo servicios como perfiles de usuarios completos, perfiles profesionales para encontrar trabajo, o simplemente una red para auto promocionarse a través de fotos y/o videos.

Actualmente cada vez más profesionales se ven obligados a ejercer su profesión en otra provincia o incluso país, al igual que muchos otros por deseo propio deciden emprender un camino diferente y emigrar.

Yo mismo me he encontrado en esta situación, y al mudarme a las Islas Baleares, me encontré con muchísimos retos, entre ellos el nuevo trabajo, arreglar todos los papeles, buscar vivienda, etc. Entre tantas obligaciones es muy complicado encontrar tiempo para relacionarse y establecer un círculo social. Si además añadimos que en muchas empresas hay una diversificación por edades considerable, te puedes encontrar en una situación que ni en el trabajo consigues relacionarte de forma no profesional.

1.2. Objetivos

El objetivo fundamental de este proyecto consiste, por tanto, en el desarrollo de una aplicación multiplataforma que facilite la tarea de poner en contacto a personas cercanas para relacionarse y conocerse. Uno de los objetivos fundamentales que nos hemos propuesto en este proyecto es el desarrollo de una app multiplataforma ya que en la actualidad el acceso a este tipo de contenidos se puede realizar desde cualquier tipo de plataforma (*e.g. smartvs, wearables, smartphones, PCs, etc.*)

Este objetivo principal se puede descomponer a su vez en los siguientes objetivos particulares:

- Análisis del estado del arte: aplicaciones y/o redes similares disponibles en el mercado.

- Desarrollar e implementar una interfaz gráfico intuitiva y amigable que permita entre otras cosas:
  - Filtrar la distancia de búsqueda.
  - Filtrar la edad de los usuarios cercanos.
  - Iniciar un chat inmediato y anónimo con un usuario que cumpla los filtros.

- Validar y probar la aplicación desarrollada en varios dispositivos móviles en plataforma Android

En este proyecto se utiliza tanto los conocimientos adquiridos durante la carrera, como los adquiridos durante el desarrollo del presente proyecto.

1.3. Tecnologías y Materiales usados

-Hardware

- Portátil Macbook Pro Early 2015 con sistema operativo OSX High Sierra
- Teléfono móvil Xiaomi Mi5s con Android 7.0 Nougat
- Teléfono móvil Samsung J7 con Android 8.0 Oreo

-Software

- Editor de texto Sublime Text 3.0
- Navegador web Google Chrome
- Administrador de paquetes NPM
- Microsoft Word 2016

1.4. Contenido de la memoria

Este trabajo fin de grado queda dividido en diferentes capítulos intentando mostrar las partes más importantes de cada fase de desarrollo.

- Capítulo 1. Introducción. En este capítulo se realiza una breve introducción del proyecto que se desea realizar, describiendo el origen de la necesidad y los objetivos de este, el material empleado para lograrlo y un resumen del contenido de esta memoria.

- Capítulo 2. Conocimientos previos. En esta sección tratamos de informar al lector sobre las diferentes tecnologías utilizadas para el desarrollo del proyecto, definiendo cada una de ellas y describiendo algunas nociones básicas de las mismas.

- Capítulo 3. Especificación de requisitos. Este apartado tendrá la función de detallar las funciones que albergar la aplicación con un análisis de requisitos (funcionales y no funcionales).

- Capítulo 4. Análisis y diseño. En este capítulo se analizan los requisitos establecidos y se plantea una opción de diseño. Se comentan los casos de uso que surgen tras el análisis anteriormente mencionado, seguido de la definición de la aplicación y de la base de datos estableciendo la estructura de ésta.

- Capítulo 5. Desarrollo. Esta sección será el corazón de la memoria y con ello la más extensa, ya que explica todo el desarrollo de la APP MeetNear. Comienza con la implementación de la BBDD y la creación y configuración del proyecto, desde aquí comentaremos cada aspecto visual de la aplicación y

explicaremos el funcionamiento y finalidad detrás de cada uno de estos. Además, se detallará como se aplicó la metodología SCRUM al desarrollo.

- Capítulo 6. Pruebas y resultados. En esta sección hablaremos sobre las pruebas realizadas, los diferentes escenarios y los resultados obtenidos por la aplicación.

- Capítulo 7. Conclusiones y trabajo futuro. En última instancia, se comentará las conclusiones extraíbles de todo el proceso, comentaremos posibles propuestas futuras que se podrían implantar.

- Bibliografía. Material bibliográfico utilizado para la realización de la memoria y el desarrollo de la APP.

Capítulo 2. Conocimientos previos

En esta sección daremos una breve introducción a las tecnologías utilizadas para la realización de este Trabajo Fin de Grado ya que ninguna de ellas forma parte del plan de estudios.

2.1 Ionic 3

Ionic es un SDK de código abierto para el desarrollo de aplicaciones móviles híbridasFigura 1: Logotipo de IONIC [1].. Construido encima de Angular, Ionic ofrece herramientas y servicios para desarrollar aplicaciones móviles híbridas utilizando tecnologías Web como CSS, HTML5. Las aplicaciones pueden ser desarrolladas con estas tecnologías para luego ser distribuidas por las tiendas de aplicaciones de los dispositivos móviles (Play store, APP Store, …) tras ser adaptadas por Cordova. Este SDK fue creado por Max Lynch, Ben Sperry y Adam Bradley de Drifty Co. en 2013, actualmente se encuentra en su tercera versión.

Figura 1: Logotipo de IONIC [1].

2.2 Angular 2

AngularJS es un *web application framework* de código abierto basado en JavaScript principalmente mantenido por Google, utilizado para la creación de interfaces de usuario de aplicaciones web donde el objetivo es simplificar el desarrollo, pruebas, mantenimiento y el uso de las aplicaciones web [5].

AngularJS (también llamado Angular) sigue el patrón de MVC (Modelo Vista Controlador), permitiendo así separar el *Front-End* (lado del cliente) del *Back-End* (lado del servidor) lo cual permite el desarrollo en paralelo.

Los 2 conceptos más utilizados por Ionic de Angular en el desarrollo de una aplicación como la de esta memoria, son los siguientes:

- Controladores: Es la parte lógica del desarrollo, son archivos escritos en TypeScript que se encargan de inicializar y modificar la información de nuestra aplicación.

- Enlace de datos bidireccional: Esto es utilizado para tener sincronizado el modelo y la vista, por lo que cuando se produce un cambio en el modelo podemos notificar a la vista para que esta se adapte y viceversa. Así la vista es dinámica ya que variará según el modelo.

Figura 2: Logotipo de Angular [5]

2.3 TypeScript

Es un lenguaje de Programación de código abierto desarrollado y mantenido por Microsoft [12]. El mismo creador de C# Anders Hejlsberg ha trabajado en el desarrollo de TypeScript, que es un superconjunto de JavaScript añadiendo tipado estático y objetos basados en clases.

TypeScript es traducido a JavaScript por medio del compilador para ser entendido por los diferentes navegadores web.

El tipado en TypeScript no es obligatorio, pero si recomendado, por lo que, si creamos una variable del tipo Integer e intentamos asignarle un tipo array, el compilador nos devolverá un error, esto ayuda a controlar el tipado correcto dentro de nuestra aplicación.

Figura 3: Logotipo de TypeScript [12]

2.4 Firebase

Firebase fue creado en 2011 por James Tamplin y Andew Lee y adquirida por Google en 2014, es una plataforma de desarrollo móvil en la nube [11]. Se podría categorizar como BaaS (*Backend as a Service*) ya que proporciona una API la cual permite sincronizar y guardar datos en la nube en tiempo real para nuestras aplicaciones. Está compuesto de diversas funcionalidades y características entre las cuales destacan las siguientes:

- Implantación sencilla.
- Sincronización inmediata, la BBDD se encuentra sincronizada en tiempo real con la aplicación por lo que, si se modifica un dato en la APP, este dato será actualizado en la BBDD.
- Los datos son almacenados en forma de JSON, por lo que permite ser utilizado en todas las plataformas mediante API REST.
- El Hosting y el dominio personalizado son totalmente gratuitos.
- Ofrece servicios de Autenticación diversos por RRSS y Email
- Tiene funcionalidades de almacenamiento en la nube de forma gratuita.

Aparte ofrece una gran cantidad de funcionalidades como analíticas, notificaciones *push, testing*…

Figura 4: Logotipo de Firebase [11]

2.5 Apache Cordova

Es la versión de código abierto de PhoneGap originalmente creado por Nitobi y adquirido por Adobe Systems en 2011 [10].

Nos permite crear aplicaciones móviles utilizando CSS3, JavaScript y HTML5 sin la necesidad de utilizar APIs específicas de cada plataforma como Android, iOS o Windows Phone.

Nos permite encapsular todo el código CSS3, JavaScript y HTML y extender el funcionamiento del HTML y del JavaScript según la plataforma destino. Finalmente, con esto obtendremos una aplicación híbrida con apariencia Nativa sin serlo. Todas las vistas de la aplicación se realizarán siempre con "*web views*" pero a la vez podremos acceder a funciones nativas del dispositivo como puede ser la cámara, GPS, LED, etc.

Figura 5: Figura Apache Cordova [10]

2.6 npm (*Node Package Manager*)

Creado por Isaac Z. y escrito íntegramente en JavaScript es un manejador de paquetes para Node.js en un entorno de ejecución para JavaScript [6]. Surgió debido a la difícil tarea de administración de paquetes para Node.js hasta entonces y observar proyectos similares como PEAR para PHP y CPAN para Perl. A día de hoy, se instala de forma automática junto con Node.js y nos permite instalar paquetes de muchas tecnologías (entre ellas las mencionadas antes) mediante la línea de comando, y asegurándonos que estén actualizados y solucionar dependencias entre ellos.

Figura 6: Logotipo Node Package Manager [6]

Capítulo 3. Especificación de requisitos

La elaboración y enumeración de requisitos de un sistema consiste en una serie de normas en las cuales se describen los servicios que ofrecerá el sistema, y bajo las restricciones que trabajará el mismo. Normalmente estos requisitos suelen dividirse en dos grupos diferentes:

- **Requisitos funcionales:** Estos requisitos tiene como finalidad describir la funcionalidad o los servicios que se espera del sistema, como norma general la respuesta a una entrada.

- **Requisitos no funcionales:** Este tipo de requisito indica las restricciones que tendrá el sistema, ya sea de diseño o a la implementación, por lo tanto, son requisitos que afectarán al funcionamiento correcto del sistema.

A continuación, detallaremos cual es la finalidad y que restricciones deberá tener la herramienta para así poder extraer los requisitos de la herramienta y proceder a su desarrollo.

3.1 Análisis de la APP

Antes de comenzar con la ejecución de este Trabajo de Fin de Grado, se analizaron las necesidades que un servicio como este deberá ofrecer para cumplir con su finalidad. Como me encontraba en esta situación tras mi mudanza a las Islas Baleares me percaté rápidamente que una necesidad era no contactar con una persona solamente, sino poder interactuar con varias personas en un mismo tiempo, ya que la compatibilidad personal es algo delicada y así se aumenta las probabilidades de congeniar. Además, la distancia entre los usuarios debería ser razonable puesto que de lo contrario el tiempo invertido en el desplazamiento sería demasiado significativo, finalmente observé que las edades eran muy dispares, por lo que surgió la necesidad de establecer ciertos rangos más factibles y con el conjunto de todo acertar con los demás usuarios. Además, era importante no revelar datos personales ya que muchas personas no quieren hacer público ningún dato personal como email, edad, etc.

3.2 Aplicación MeetNear

En esta sección se detallan los requisitos funcionales y no funcionales de la aplicación a desarrollar:

3.2.1 Requisitos funcionales

- RF01 – Pantalla de inicio: La aplicación deberá mostrar una pantalla principal, en la cual el usuario podrá elegir si desea registrarse o si desea logarse.
- RF02 – Pantalla de registro: El usuario podrá registrarse en la aplicación tras llegar aquí desde la página de inicio.
    - RF02.1 – Email: El usuario se podrá registrar solo con una dirección de email válida en formato.
    - RF02.2 – Contraseña: La contraseña deberá de estar compuesta de mínimo 6 caracteres, diferenciando entre mayúsculas y minúsculas.
- RF03 – Login: El usuario podrá logarse siempre y cuando se encuentre registrado previamente, facilitando su email y su contraseña.
- RF04 – Filtros de búsqueda: El usuario tendrá la obligación de establecer una serie de filtros, para filtrar la búsqueda de otros usuarios.
    - RF04.1 -Filtro edad: El usuario tendrá la opción de filtrar diferentes rangos de edades ([18-23], [24-29], [30-35], [>36], [indiferente]).
    - RF04.2- Filtro distancia: El usuario tendrá la opción de filtrar diferentes distancias ([<2km], [<5km], [<10km], [<15km], [indiferente]).
- RF05 – Listado de chats: Se mostrará una lista con diferentes usuarios para chatear.
    - RF05.1 -Las variables de los usuarios mostrados deben encajar con los filtros establecidos por el usuario.
    - RF05.2 -El usuario podrá seleccionar cualquier usuario de la lista para abrir una sala de chat con él.
    - RF05.3 -El usuario tendrá la opción de volver para modificar los criterios de búsqueda.
- RF06 - Sala de Chat: Se mostrará una ventana en la cual es usuario podrá mandar mensajes al usuario deseado.

3.2.2 Requisitos no funcionales

- RFN01 – Conexión a Internet: La aplicación necesita una conexión a internet para comunicarse con la BBDD y los servicios de Firebase. Esta conexión es proporcionada por el propio dispositivo.
- RNF02 – Usabilidad: Debido a que la aplicación es multiplataforma y puede ser utilizada en diferentes dispositivos con diferentes resoluciones y tamaños de pantalla, la interfaz debe ser dinámica y adaptable a las pantallas para no influir en la usabilidad de la aplicación.

- RNF03 – Tecnologías web: Los leguajes de programación y las tecnologías utilizadas deben ser HTML, CSS y JavaScript, por lo tanto, propios de la web. Con esto la aplicación web podrá ser exportada a Android, iOS o Windows phone o incluso al propio navegador web.
- RNF04 – Conexión a GPS: El dispositivo deberá permitir a la aplicación acceder a la ubicación de este, para ubicar al usuario filtrando así de forma correcta a los demás usuarios.

Capítulo 4. Análisis y diseño

4.1 Casos de Uso

En este apartado observamos los casos de uso derivados de los requisitos funcionales especificados en el apartado anterior Figura 7.

Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas.

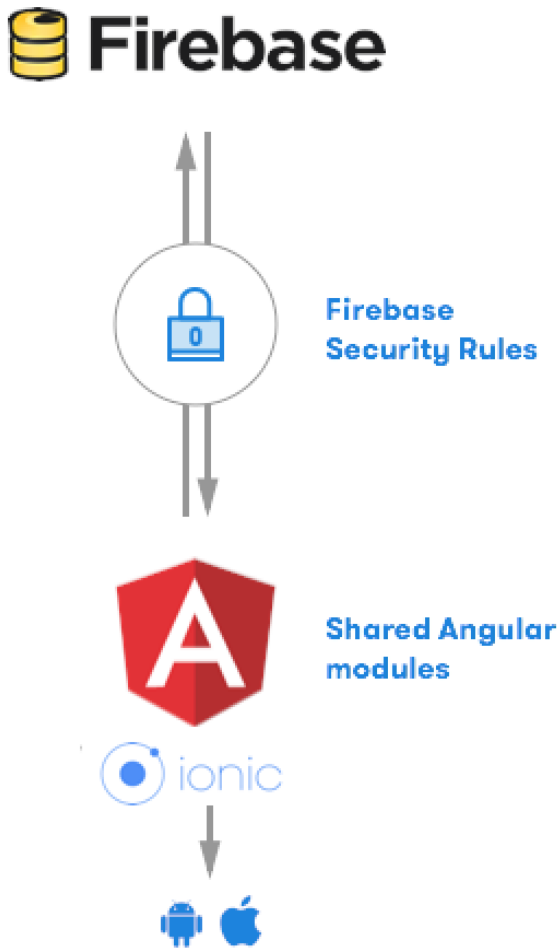

Figura 7: Esquema intervención aplicación MEETNEAR

4.1.1. Aplicación MeetNear

En la aplicación podemos observar el siguiente diagrama de Casos de Uso que sigue la aplicación Figura 8.

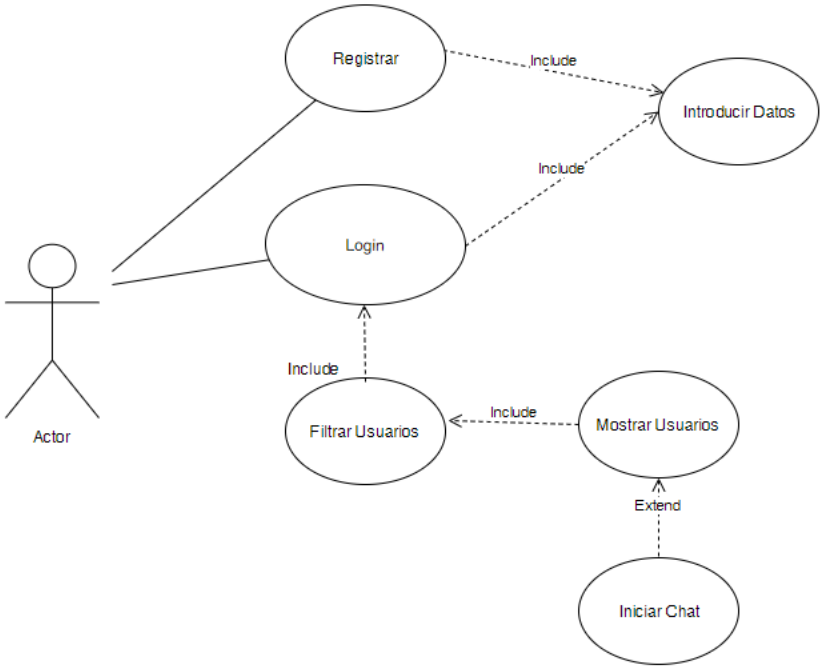

Figura 8: Diagrama de Casos de Uso aplicación MEETNEAR

- CU01 – Registrar:
  - *Descripción:* El usuario podrá registrarse en la aplicación para tener una cuenta en la misma.
  - *Actores:* Usuario
  - *Precondición:* El Usuario deberá rellenar su e-mail, nombre, contraseña (2 veces), edad y tener el GPS activado.
  - *Postcondición:* Será redirigido a una ventana para filtrar a los demás usuarios.
  - *Excepciones:*
    1. Si el formato del email no es válido será notificado.
    2. Si el email ya está en uso será notificado.
    3. Si la contraseña cumple los requisitos de seguridad será notificado.
    4. Si la contraseña no concuerda con la repetida será notificado.
  - *Secuencia:*
    1. Seleccionar Registrar.
    2. El usuario introduce su email.
    3. El usuario introduce un nombre.
    4. El usuario introduce la misma contraseña dos veces.
    5. El usuario introduce su edad.
    6. El usuario pulsa el botón “Registrar”.
    7. Se mostrará la ventana de filtros.

- CU02 – Login:
    - *Descripción:* El usuario se puede logar en la aplicación.
    - *Actores:* Usuario.
    - *Precondición:* El usuario debe de haberse registrado previamente y tener el GPS activado.
    - *Postcondición:* Será redirigido a una ventana para filtrar a los demás usuarios.
    - *Excepciones:*
        1. Si el email no existe será notificado.
        2. Si la contraseña es incorrecta será notificado.
    - *Secuencia:*
        1. Seleccionar Login.
        2. El usuario introduce su email.
        3. El usuario introduce su contraseña.
        4. Se mostrará la ventada de filtros.
- CU03 – Filtrar usuarios:
    - *Descripción:* El usuario debe aplicar los filtros de búsqueda.
    - *Actores:* Usuario
    - *Precondición:* El usuario debe de estar logado
    - *Postcondicion:* Al usuario se le mostrarán los usuarios filtrados.
    - *Excepciones:* Ninguna.
    - *Secuencia:*
        1. El usuario selecciona en un desplegable el rango de edad deseado.
        2. El usuario selecciona en un desplegable la distancia máxima deseada.
        3. El usuario pulsa el botón “BUSCAR”.
        4. Al usuario se le mostrará el listado de usuarios filtrados.
- CU04 – Iniciar chat:
    - *Descripción:* El usuario podrá chatear con otro usuario de la aplicación.
    - *Actores:* Usuario.
    - *Precondición:* El usuario tendrá que haber aplicado los filtros.
    - *Excepciones:* Ninguna.
    - *Secuencia:*
        1. El usuario seleccionará un usuario con el cual quiere chatear.
        2. Se abrirá una “sala de chat” para interactuar.
        3. El usuario podrá escribir en un campo inferior.
        4. El usuario podrá mandar el mensaje con el botón de enviar.
        5. El mensaje subirá a la conversación y el otro usuario lo recibirá.

4.2 Arquitectura

En este proyecto podemos distinguir entre dos tipos de arquitecturas: por un lado, nos encontramos con la organización de clases y funciones siguiendo el patrón establecido por Angular, MVC (Modelo Vista Controlador). Y por el otro lado la arquitectura cliente servidor, en la cual el cliente es la aplicación y el servidor será el mencionado en el apartado 2.4 Firebase.

4.2.1 Arquitectura Modelo Vista Controlador

Con la utilización de Angular, seguimos el ya conocido patrón Modelo Vista Controlador Figura 9. En este patrón la arquitectura de software define tres partes interconectadas consiguiendo separar el funcionamiento interno de la aplicación y la información a la que se accede a la base de datos y de lo que se muestra al usuario.

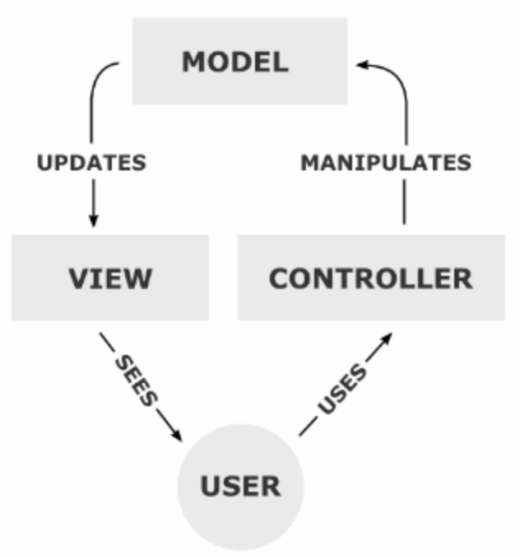

Figura 9: Modelo Vista Controlador

4.2.2 Arquitectura Cliente Servidor

En esta arquitectura software aislamos el cliente del servidor y los conectamos a través de Internet. En esta arquitectura la conexión solo se realiza cuando el cliente lo solicita, por lo que el servidor nunca suministrará información sin ser solicitada. Cuando el cliente la solicita datos, el servidor responde proporcionando los datos de la BBDD, al igual que cuando el cliente solicita almacenar nuevos datos, el servidor se ocupará de ello.

Visto esto, nuestro servidor Firebase nos proporciona un Servicio: proveernos de la información de la BBDD cuando los clientes lo soliciten respondiendo a todas las peticiones Figura 10.

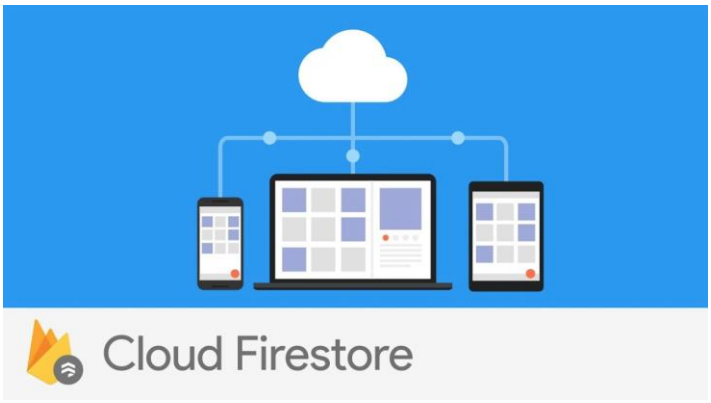

Figura 10: Ilustración Firestore

4.2.3 Modelo

Esta es la capa encargada de trabajar con los datos, dicho de otra forma, contiene los métodos y las funciones encargados de actualizar, insertar, obtener, etc. los datos de la BBDD lo cual es realizado por la API de Firebase. Igualmente, a lo largo de la memoria veremos métodos como *update()*, pero estos son métodos proporcionados por la librería de AngularFire (Librería encargada de conectar Angular con Firebase) y pertenecen al controlador.

4.2.4 Vista

La vista, contiene todo el código de nuestra aplicación referente a lo visual, ya que creará la visualización de las interfaces de usuario, básicamente es el conjunto de HTML y CSS. La vista recibe los datos del modelo, pero no accede directamente a ellos.

4.2.5 Controlador

El controlador contiene el código necesario para responder a las peticiones realizadas por la aplicación, como realizar una búsqueda, mostrar datos, etc.

Es la capa que enlaza la vista con el modelo, normalmente no es responsable de manipular los datos, pero en nuestro caso, al utilizar Firebase y no disponer de un servidor para manipular los datos, utilizamos el controlador para realizar los cálculos y las llamadas a la API para la obtención de datos.

4.3 Estructura y diseño

La interfaz de usuario, así como los estilos utilizados por Android, iOS y Windows Phone, se adaptan según la plataforma en la cual es exportada la aplicación gracias a Ionic. El framework se encarga de crear una plantilla limpia desde cero cumpliendo con las pautas nativas de iOS y Windows, y las de *Material Design* en Android. Gracias a esto, tenemos una base de estilos ya aplicado de antemano, pero con opción de ser modificado y personalizado por parte del desarrollador. En este proyecto se ha mantenido los estilos que trae Ionic intentando ofrecer una interfaz limpia y conocida a los usuarios intentando facilitarles el uso de la aplicación.

Al utilizar los estilos de la interfaz nativa de cada plataforma, el usuario acostumbrado a su plataforma sabrá directamente como interactuar con la aplicación y en que parte de la interfaz, por ejemplo, puede navegar hacia atrás Figura 11.

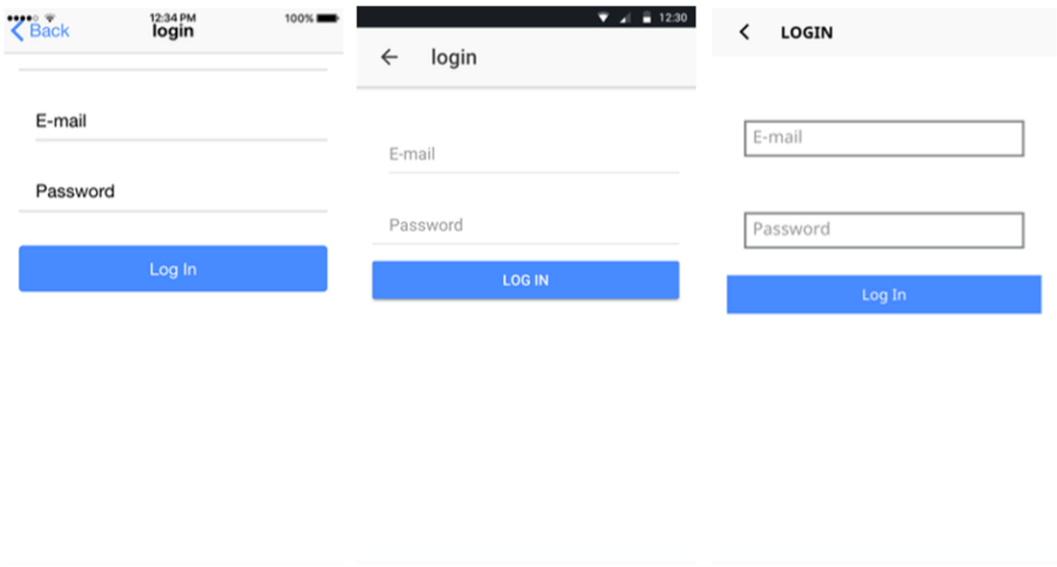

Figura 11: Vista Login según plataforma

Capítulo 5. Desarrollo

5.1 Base de datos

Para el funcionamiento de esta aplicación, no es necesario una estructura muy compleja, por lo que no hay que hacer sentencias SQL laboriosas o crear tablas relacionadas entre sí, realmente serán consultas bastante sencillas. En su mayor parte la información devuelta son listas de usuarios o de mensajes de los chats, pequeñas colecciones de datos para trabajar con los mismos. Por lo que las consultas que se harán en su mayoría son de leer/escribir usuarios y/o mensajes.

Por esta razón es suficiente con un modelado NoSQL, pudiendo escalar horizontalmente conforme se añadan más usuarios o mensajes.

Tal como ya mencionado en la introducción, Firebase se adapta perfectamente a esta necesidad ya que gracias a la API podemos acceder de forma sencilla a FireStore que almacena los datos en forma de colecciones y así poder leer solo ciertas colecciones como conversaciones entre dos individuos, sin estar obligados a recibir todas las demás.

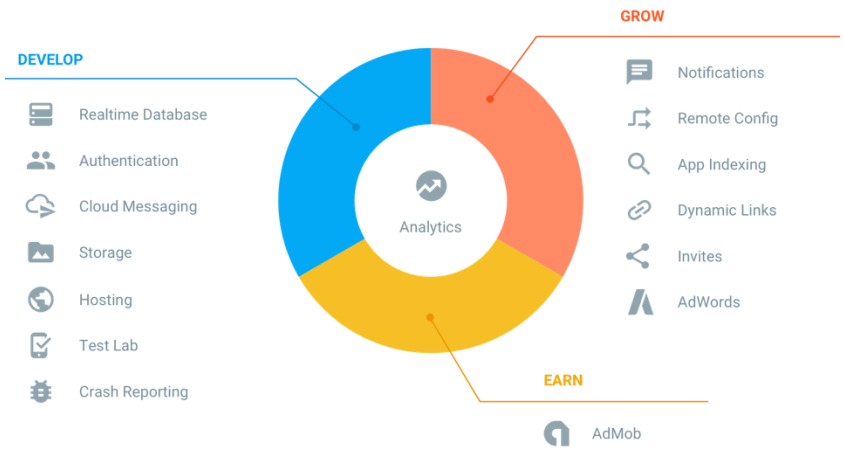

Figura 12: Gráfico de las funcionalidades de Firebase

En este caso la BBDD es dinámica y va creciendo según los usuarios que se registren, a comienzos solo tenemos definidos dos tipos de colecciones (Usuarios y Chats), cuando los documentos de los usuarios se identifican con el mail, y los chats con un id aleatorio Figura 13.

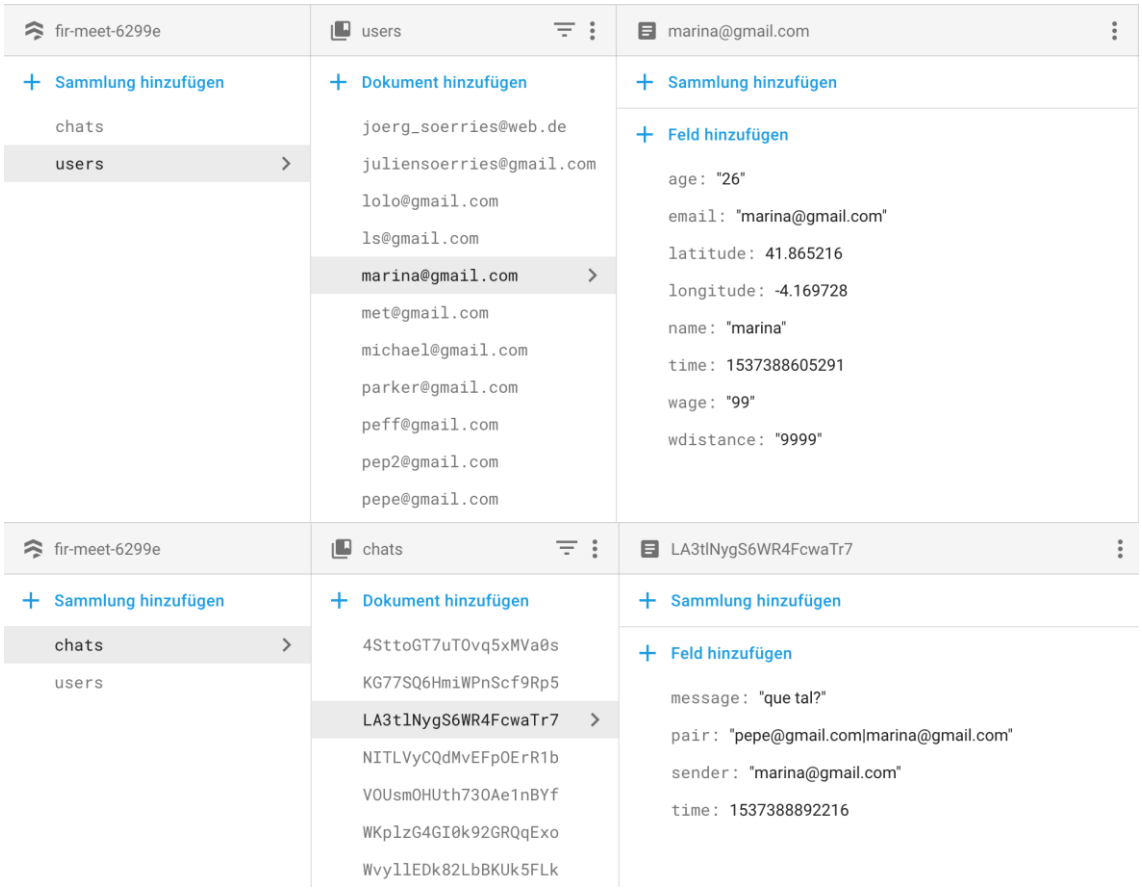

Figura 13: Contenido de ejemplo de Firestore

Como se puede observar, por cada mensaje emitido se genera un documento en la colección de "chats" que está compuesta por el mensaje, un *pair id* para identificar los participantes, el usuario que emite el mensaje y un sello de tiempo.

En el caso de los Usuarios, estos contienen todos los datos relevantes del mismo, como las coordenadas, nombre, edad, edad del filtro, distancia del filtro y el mail.

5.2 Creación y configuración del proyecto

La creación y configuración del proyecto puede separarse en tres pasos, dónde cada uno de ellos requiere ciertas configuraciones e instalaciones que deben realizarse.

- Ionic: inicialización y creación del proyecto en blanco.

- Firebase: Inicializar el entorno Firebase para la conexión con la APP.

- GitHub: creación y subida del contenido al repositorio privado para tener el control de versiones.

En primer lugar, debemos instalar Node.js, lo cual a su vez nos instalará NPM en nuestro equipo para administrar los paquetes de JavaScript, esto lo que nos permitirá es instalar y actualizar los paquetes de Ionic, Apache Cordova, Angular 2, Firebase y AngularFire2.

Una vez realizado esto, ya se procede a instalar Ionic 3 y Apache Cordova desde la terminal del equipo con los siguientes comandos:

```
npm install -g ionic cordova
```

Y ya con esto podemos iniciar el comando que nos inicializará el proyecto, para ello ejecutamos el siguiente comando:

```
ionic start meetnear blank
```

En este comando podemos ver como "meetnear" indica el nombre de nuestro proyecto y el "*blank*" indica un proyecto en blanco sin ningún tipo de interfaz predefinida.

Con esto ya tenemos el proyecto creado, ahora solo falta indicar las plataformas para las cuales estamos desarrollando, esto lo hacemos añadiéndolas al proyecto:

```
ionic platform add Android

ionic platform add ios

ionic platform add windows

ionic platform add browser
```

Dentro de la carpeta se ha creado la siguiente estructura Figura 14, la cual no se desvía demasiado de la de otros lenguajes de programación.

Para poder utilizar de forma correcta Firebase en la aplicación, se procederá a instalar AngularFire, la librería de conexión entre AngularJS y Firebase, además añadiremos el plugin de geolocalización para obtener las coordenadas del usuario, esto se realiza a través de npm y ionic con el siguiente comando:

```
npm install angularfire2 firebase –save

ionic cordova plugin add cordova-plugin-geolocation

npm install –save @ionic-native/geolocation
```

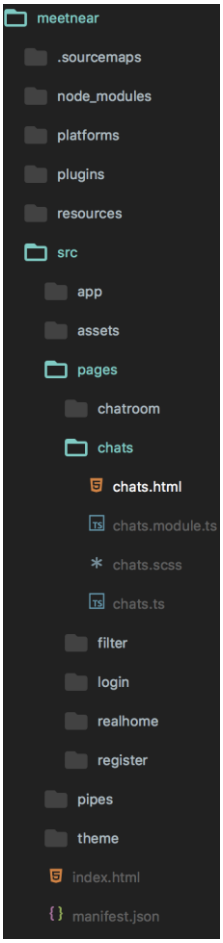

Figura 14: Estructura de proyecto en IONIC

Una vez realizado esto, para conectar la aplicación con Firebase debemos de facilitarle los datos de acceso y conexión al fichero app.module.ts de la app. En este caso he optado por crear un fichero con las variables constantes que podemos obtener en la web de Firebase Figura 15:

Figura 15: Variables de conexión con Firebase

Estos datos, son los necesarios para realizar las conexiones con Firebase, por lo que en el fichero app.module.ts importamos las librerías de AngularFire y la variable con los datos Figura 16.

Figura 16: Importaciones en app.module.ts

Finalmente, solo tenemos que pasarlo como parámetro el angularfiremodule para que la aplicación cuente con este módulo Figura 17.

```
38   imports: [
39     BrowserModule,
40     IonicModule.forRoot(MyApp),
41     IonicStorageModule.forRoot(),
42     AngularFireModule.initializeApp(appconfig.firebase),
43     AngularFirestoreModule,
44     AngularFireAuthModule,
45     PipesModule
46   ],
```

Figura 17: Parámetros del fichero app.module.ts

Con estos pasos ya tenemos la aplicación lista para comenzar a desarrollar, y con la conexión a Firebase realizada, ahora bien, para asegurarnos un correcto flujo de trabajo se ha creado un repositorio privado en GitHub para tener el código alojado en la web, y poder hacer uso del SCM (control de versiones) git Figura 18. A lo largo del proyecto se ha estado utilizando para evitar errores en cambios críticos y poder volver atrás en caso de fallos críticos.

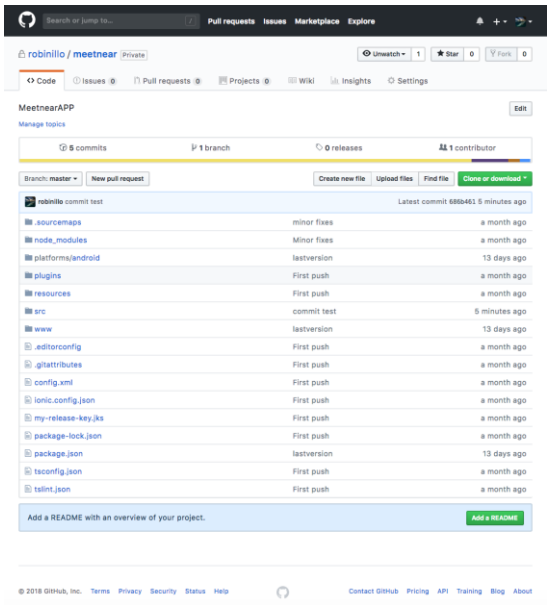

Figura 18: Vista del proyecto en GitHub

5.3 Aplicación MeetNear

En esta sección se describe las diferentes vistas de la aplicación además del funcionamiento y las restricciones de las mismas.

5.3.1 Register

Nada mas abrir la aplicación nos permite registrarnos en la misma, (la solapación del logo con la hora en iOS es un bug actual de ionic-lab utilizado para emular las plataformas) con el botón de "*login*" seremos redireccionados.

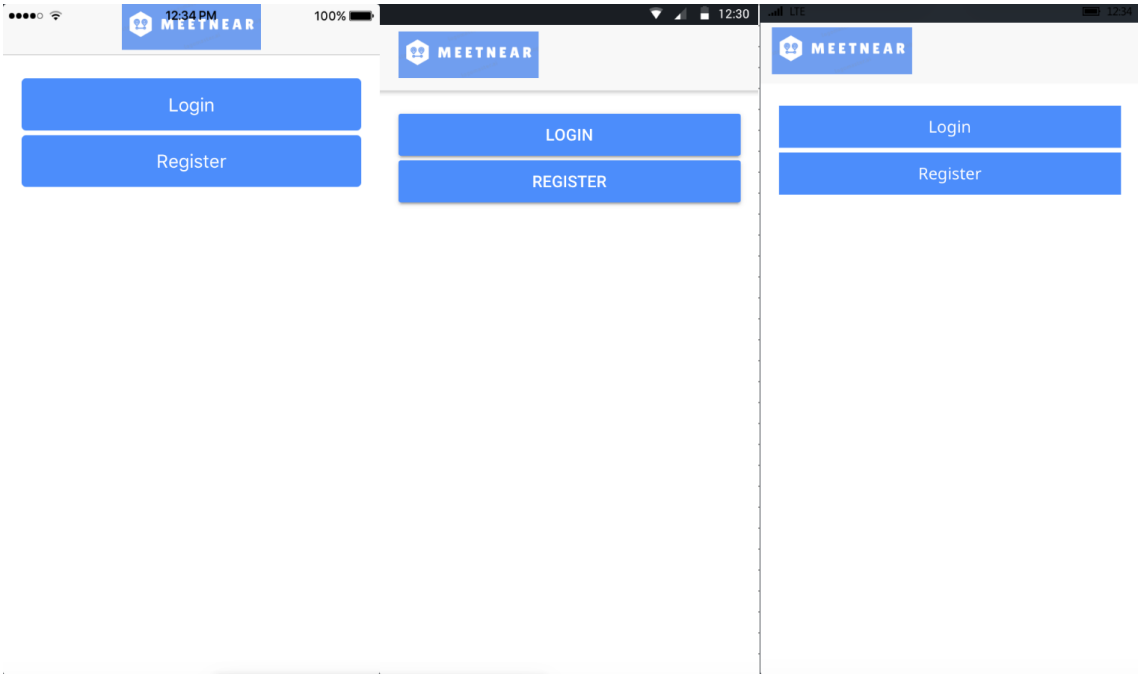

Figura 19: Pantalla inicial MEETNEAR según plataforma

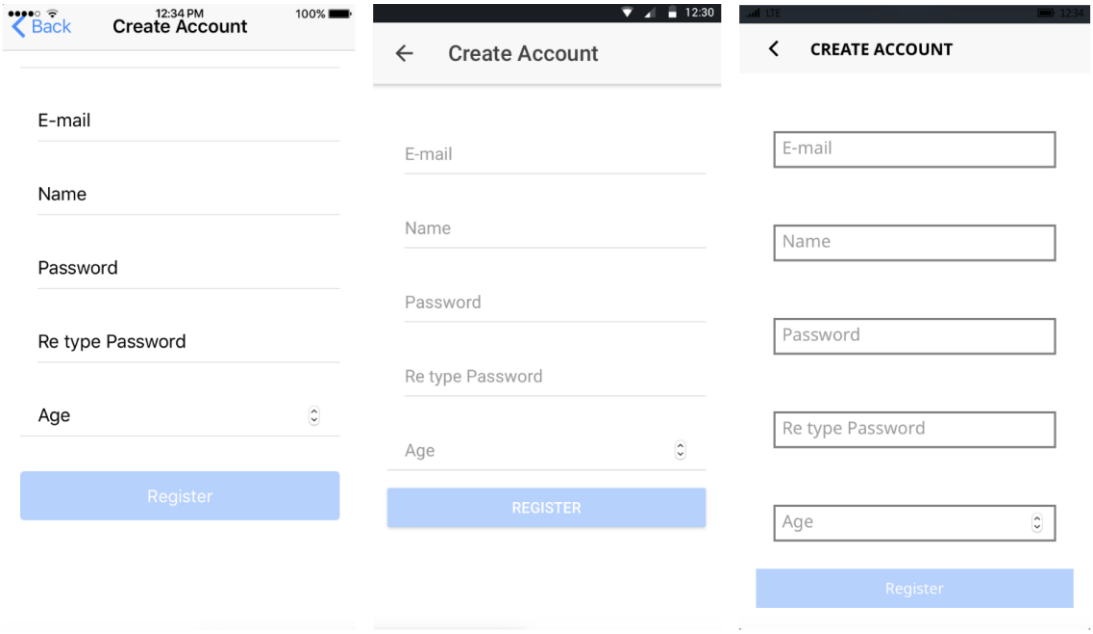

Figura 20: Página de registro de MEETNEAR

Por lo que, con un simple toque a la pantalla, se nos abrirá una nueva ventana con los campos requeridos para el registro Figura 20. Tal como comentado, gracias a Ionic se conserva el estilo de cada plataforma lo cual se puede ver bastante claro en la ilustración.

Para que las cuentas sean creadas de forma correcta, es obligatorio que los campos sean cumplimentados todos, ahora explicaremos los requisitos de cada uno de ellos:

- E-mail: Este campo es requerido para el registro con el servicio de autenticación de Firebase, por lo que debe de cumplir los requisitos de un email valido "*aaaaa@xxx.yyy*", en caso de no cumplirse, la aplicación mostrará un mensaje Figura 21.

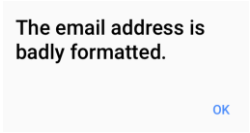

Figura 21: Error e-mail

Al igual que si el email ya existe, AngularfireAuth se encargará de comprobarlo y notificarlo al usuario Figura 22.

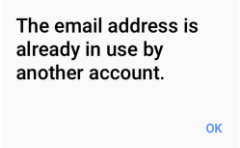

Figura 22: Error e-mail existente

- Name: Este campo es el nombre del usuario, no tiene restricciones se puede repetir, puede contener caracteres y números, por lo que se le deja la libre elección al usuario. Este campo no es utilizado para logar, se almacena junto con los datos del usuario para identificarlo en los chats y en las listas.

- Password: Este campo es de tipo contraseña, al igual que en email este campo es controlado por la autenticación de Firebase, por lo que, si la contraseña no cumple la longitud mínima, también se emitirá un mensaje al usuario para ponerlo en conocimiento de ello Figura 23.

- Re Password: Con el fin de evitar que el usuario introduzca la contraseña errónea, se le solicita nuevamente en este campo y si la contraseña no es idéntica a la primera salta un aviso Figura 24.

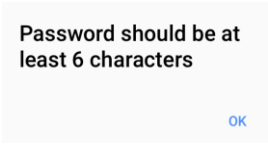

Figura 23: Error contraseña inválida

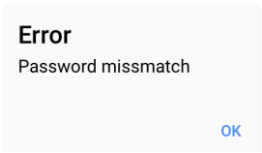

Figura 24: Error contraseña no coincide

- Age: Finalmente nos encontramos con el campo de edad, este es necesario para las filtraciones, ya a partir de este dato comprobamos si entra en los filtros de búsqueda o no. Simplemente hay que introducir un valor entre 16 y 99 para poder utilizar la aplicación Figura 25.

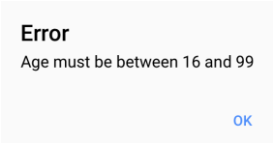

Figura 25: Error edad incorrecta

Solo cuando todos los datos están debidamente cumplimentados se puede pulsar el botón de registrar, pasando de un azul opaco a uno normal Figura 26.

Figura 26: Botones de registro

Las ventanas emergentes en caso de email y contraseña son gestionadas por AngularFire, las demás como la comprobación de que ambas contraseñas sean correctas son un simple alert Figura 27.

```
65  alert(message: string, message2: string){
66
67    this.alertCtrl.create({
68      title: message,
69      subTitle: message2,
70      buttons: ['OK']
71    }).present();
72
73  }
```

Figura 27: Función alert

El código HTML de esta función se encuentra en "*meetnear/src/app/pages/register/register.html*" y su correspondiente controlador, el cual maneja los datos y realiza los cálculos o acciones con dichos datos en "*meetnear/src/app/pages/register/register.ts*" Figura 28.

Al pulsar sobre el botón registrar, obtenemos los datos introducidos en el formulario HTML y el controlador se encarga de todas las gestiones.

A continuación, se puede observar el código encargado de las validaciones del formulario, la creación del alta del usuario y en caso satisfactorio el traslado a la siguiente ventana de filtros Figura 28.

```
82    registerUp(){
83      console.log(this.prof.value);
84
85      if (this.pwd.value != this.rpwd.value) {
86        this.alert('Error', 'Password missmatch');
87
88      }else if(this.profile.age <16 || this.profile.age > 99){
89
90        this.alert('Error', 'Age must be between 16 and 99');
91
92      }else{
93        let myLoader = this.loadingCtrl.create({
94          content: "Please wait..."
95        });
96
97        myLoader.present().then(() => {
98          this.profile.name = this.usrname.value;
99          this.profile.email = this.email.value;
100         this.fire.auth.createUserWithEmailAndPassword(this.email.value, this.pwd.value)
101         .then(data => {
102           this.profile.time = new Date().getTime(); //timestamp
103           console.log('usuario en registro:', this.profile);
104           this.chatservice.addUser(this.profile); // añadir datos usuario a db
105           this.chatservice.setMyUser(this.profile);
106           let toast = this.toastCtrl.create({
107                 message: "Login In Successful",
108                 duration: 3000,
109                 position: "top"
110           });
111           toast.present();
112           myLoader.dismiss();
113           this.alert('Registered!','');
114           this.navCtrl.setRoot(FilterPage);
115         })
116         .catch(error => {
117           console.log('got error2', error);
118           this.alert(error.message, '');
119           myLoader.dismiss();
120         })
121       })
122     }
```

Figura 28: Función de registro

5.3.2 Login

Una vez registrado, el usuario puede logarse a la aplicación ya sea desde el mismo dispositivo del que se ha creado la cuenta, o desde cualquier otro ya que accede a la autenticación de Firebase para buscar y asignar su usuario Figura 29.

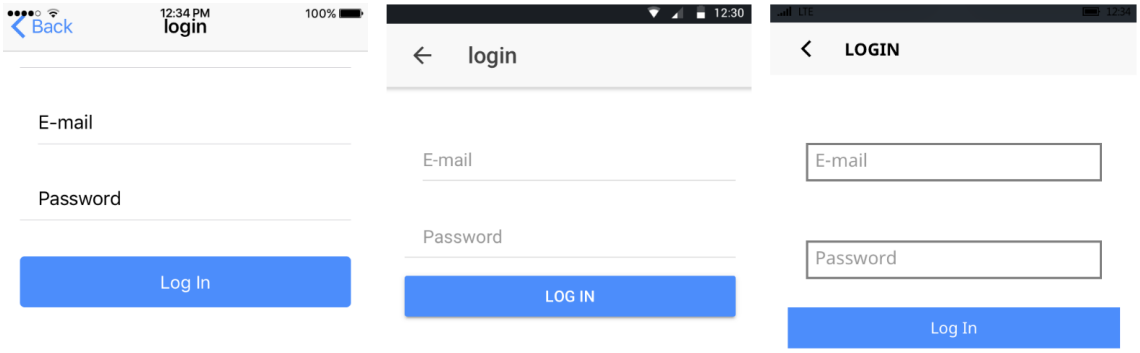

Figura 29: Página de login

Simplemente usando las credenciales del registro, la aplicación se encargará de cargar los datos del usuario en cuestión y abrir la sesión con su usuario. Al igual que en el punto de registrar, de esto se encarga Firebase, por lo que si el usuario no existe o la contraseña es incorrecta el usuario será notificado Figura 30.

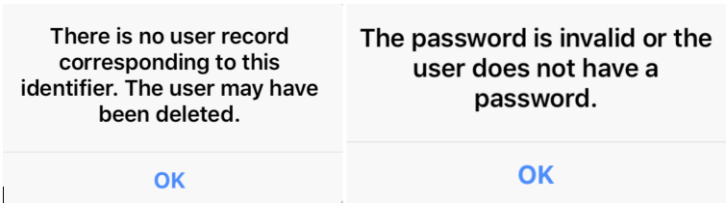

Figura 30: Error de login

El código HTML de esta función se encuentra en "*meetnear/src/app/pages/login/login.html*" y su correspondiente controlador, el cual maneja los datos y realiza los cálculos o acciones con dichos datos en "*meetnear/src/app/pages/loging/login.ts*".

Tanto en el registro como en el *login*, los datos del usuario quedan almacenados de forma local para diferenciar siempre el usuario que utiliza el dispositivo con los demás que se encuentran en la base de datos Figura 31.

```
78      let val = this.db
79          .collection<User>(appconfig.users_endpoint, ref => {
80            return ref.where("email", "==", this.usr.value);
81          })
82          .valueChanges()
83          .subscribe(users => {
84            console.log('usuario en login:', users[0]);
85            this.chatservice.setMyUser(users[0]);
86            myLoader.dismiss();
87            this.alert('Succesfull','you logged in correctly');
88            this.navCtrl.setRoot(FilterPage);
89            val.unsubscribe();
90          })
91
92        })
```

Figura 31: almacenaje variable local

5.3.3 Filtro

Nada más logarse o registrarse el usuario, se le mostrará la pantalla de filtro, la finalidad de esta es ajustar sus criterios de búsqueda con dos parámetros esenciales, la distancia y el rango de edad. Para ello se han establecido unos parámetros seleccionables, por lo que el usuario en una lista desplegable debe decidir la distancia máxima a la que quiera que se encuentren los demás usuarios, y el rango de edad que estos deben de tener Figura 32.

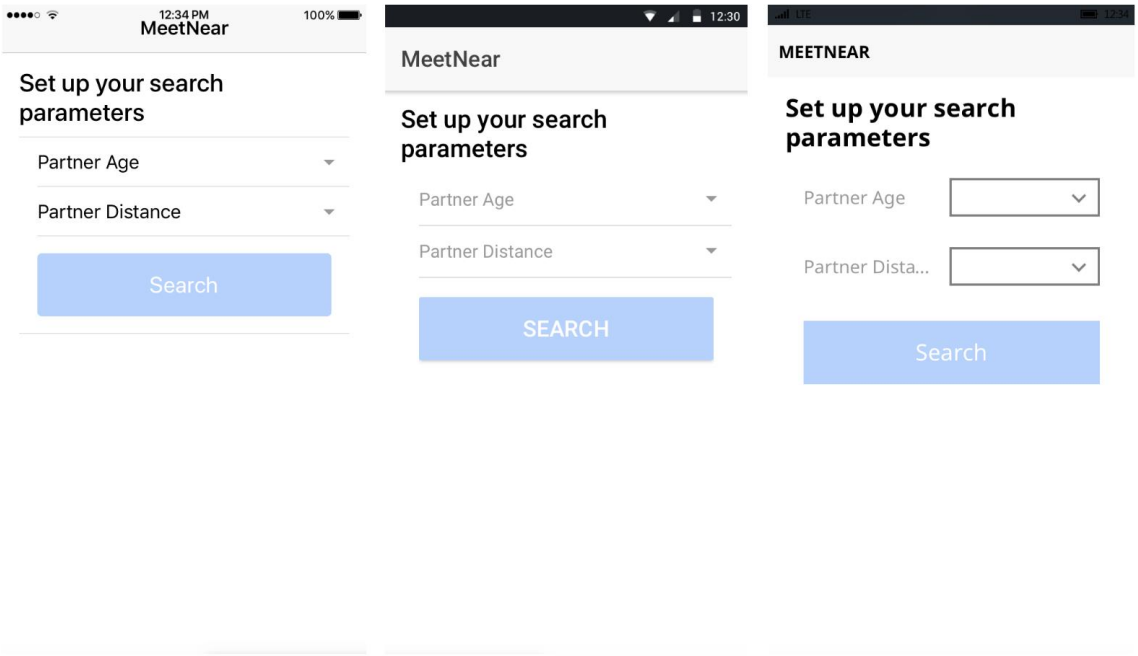

Figura 32: Página filtro de búsqueda

En todas las plataformas, el desplegable contiene lo mismo pero la interfaz es adaptada a la plataforma como todas las demás interfaces de la aplicación Figura 33.

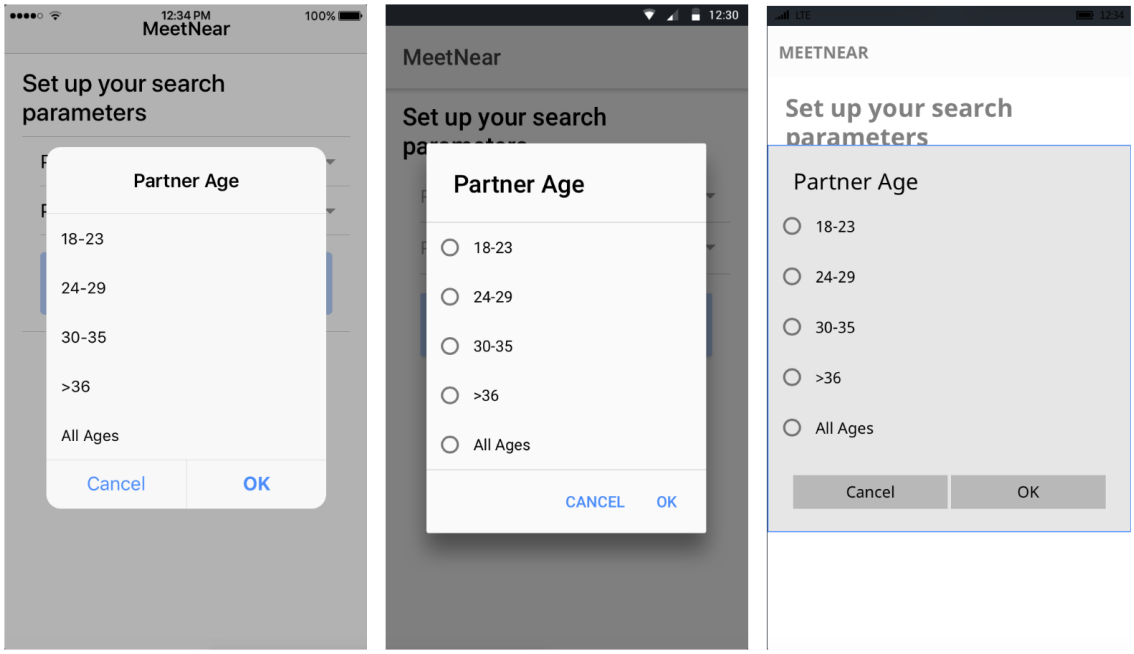

Figura 33: Ventana emergente con los parámetros de edad

Una vez seleccionados ambos filtros, el controlador se encargará de añadir estos parámetros a la BBDD para este usuario y le mostrará la siguiente vista.

El código HTML de esta función se encuentra en "*meetnear/src/app/pages/filter/filter.html*" y su correspondiente controlador, el cual maneja los datos y realiza los cálculos o acciones con dichos datos en "*meetnear/src/app/pages/filter/filter.ts*".

5.3.4 Chats

Esta es una de las vistas más complejas de la aplicación, ya que al cargar esta interfaz se realizan varios cálculos antes de mostrar la información. Para facilitar el desarrollo, y no tener que importar constantemente librerías y declarar las variables en el constructor, se ha creado una clase llamada *app.service.ts* ubicada en "*meetnear/src/app/app.service.ts*" para importar solo esta y crear varias funciones en la misma para agilizar el desarrollo, en esta clase podemos encontrar funciones como: "*setMyUser()*, o *getMyUser()*" para leer o escribirlo en la BBDD o "*checkComp()*" para comprobar la compatibilidad de dos usuarios al aplicar los filtros.

Esta última es muy importante ya que se comprueba que usuarios deben de ser mostrados en la lista y cuáles no, dicho de otra forma, con cuales puede chatear el usuario y con cuáles no.

Al abrirse esta ventana, lo primero que se realiza es añadir la ubicación actual del usuario a la BBDD, una vez realizado esto, el controlador comenzará a comprobar la compatibilidad de usuarios cargando cada uno de ellos y pasándolo por el comprobador de compatibilidad Figura 34.

```
61        this.db
62          .collection<User>(appconfig.users_endpoint)
63          .valueChanges()
64          .subscribe(users => {
65            //this.availableusers = users;
66            console.log(users);
67            this.availableusers = users.filter(user => {
68              console.log('usuario1', user, 'usuario2', this.chatuser);
69              if (user.email != this.chatuser.email ) {
70
71                console.log('comprobando compatibilidad entre:', user, this.chatuser);
72                if(this.chatService.checkComp(user,this.chatuser)){
73                  console.log('compatible');
74
75                  return user;
76
77                }else{
78
79                  console.log('no compatible');
80                }
81              }
82            });
```

Figura 34: comprobación de compatibilidad

Para comprobar esto se carga la función de *checkCom()* a la cual pasamos como parámetro los dos usuarios a comprobar. La función en primer lugar calculará la distancia a la que se encuentran, si la distancia cumple con los filtros de ambos usuarios seguirá un paso más adelante (*IF ELSE*), en el siguiente paso se comprueba la edad, si esta cumple también los requisitos de ambos usuarios, la función devolverá un valor booleano verdadero en caso de satisfacer los filtros de ambos usuarios y uno falso en caso contrario.

Si el usuario comprobado cumple estos requisitos es añadido en un *Array* de usuarios el cual finalmente es mostrado en forma de lista en la vista. Al pulsar sobre uno de estos, es llamado el método "*goToChat()*" el cual genera un id de pareja para identificar los mensajes concatenando ambas direcciones email. Finalmente se entra en la sala de chat Figura 35.

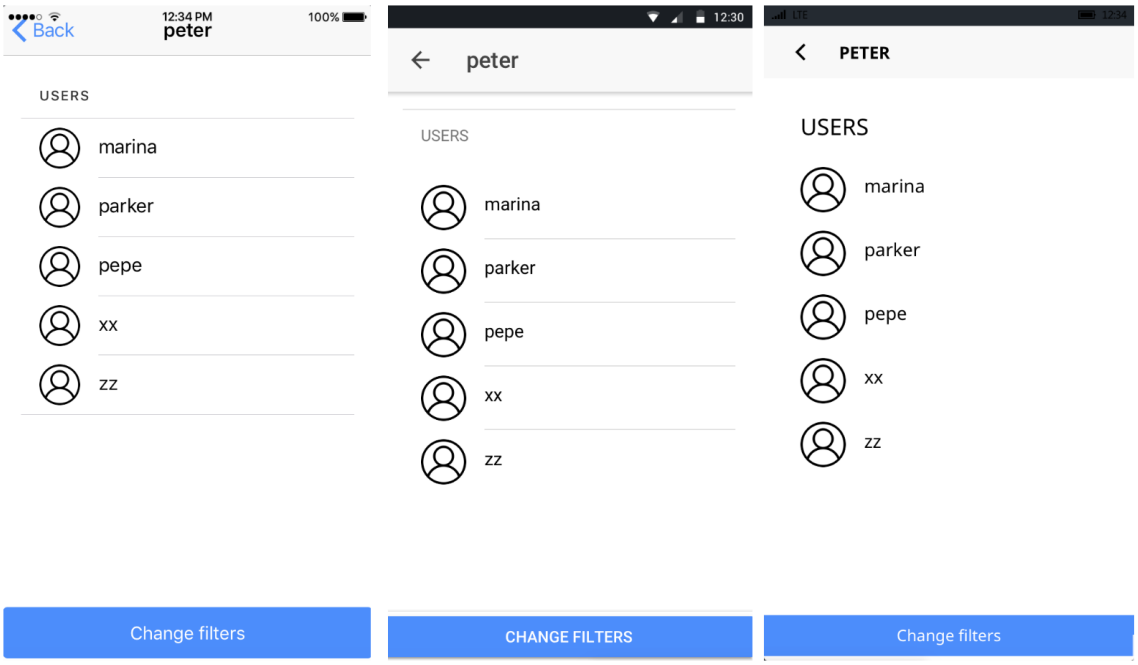

Figura 35: Lista de usuarios

5.3.5 ChatRoom

Finalmente tenemos las Salas, esta es la vista en la cual dos usuarios interactúan, para ello se muestra un campo inferior para escribir el mensaje, y un botón de envío para que el controlador envíe dicha información a la BBDD Figura 36.

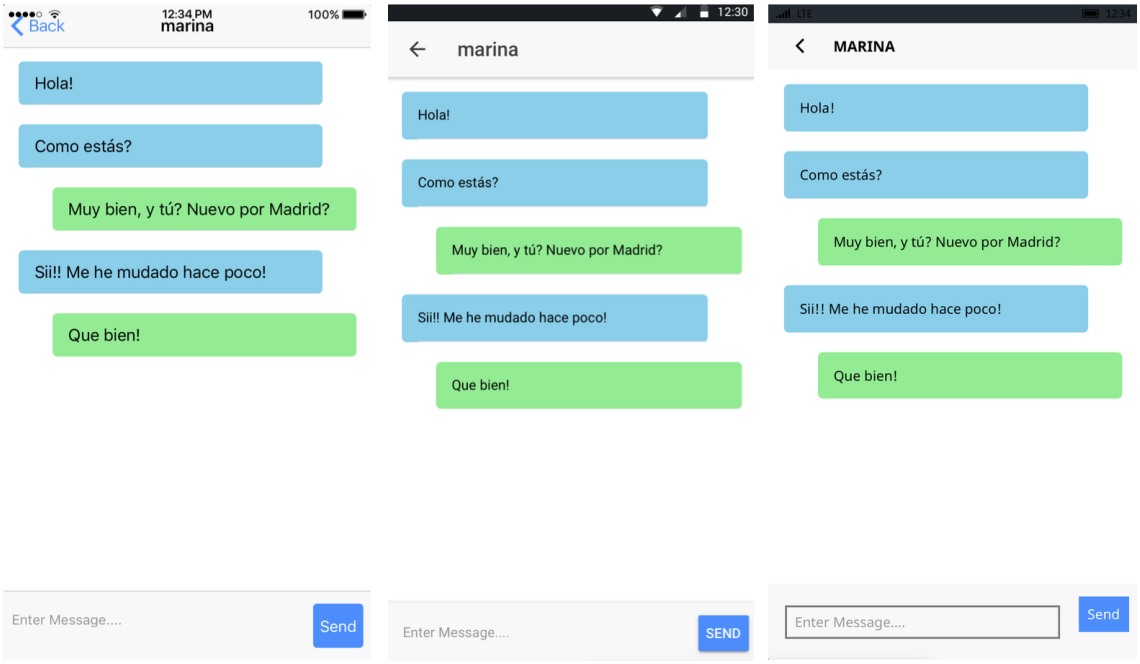

Figura 36: Ventana de chat

En este caso para obtener los chats de esta conversación, utilizando Firebase cargamos de la colección “Chats” los mensajes en los que coincida el “*pairID*” compuesto por ambos emails concatenados, además a la hora de mostrarlos en pantalla Angular se encarga de ordenarlos por el tiempo en el que fueron creados, por lo que los mensajes conservarán su orden original.

Para que los mensajes sean diferenciables entre los usuarios, se filtra según el emisor del mismo, por lo que se llama la función “*isChatPartner()*” para indicarnos si el mensaje cargado proviene del emisor o del receptor y así con el *CSS* poder adjudicarle diferentes estilos.

5.4 Metodología

Durante todo el desarrollo se ha seguido la metodología SCRUM, la cual es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto [13]. Siguiendo esta metodología se ha propuesto metas semanales y/o mensuales según la dificultad.

- Formación: Para afrontar la programación en Aplicaciones Híbridas (la cual no forma parte del plan de estudios de la carrera) fue necesario un importante esfuerzo de aprendizaje previo al desarrollo de la aplicación.

- Investigación: En esta fase se investigó principalmente la comunicación entre cliente y servidor, eliminando la idea inicial de desarrollar una API para la comunicación con un servidor que se montaría sobre un PC o Raspberry Pi, y se concluyó que lo más idónea es utilizar los servicios que ofrece Google con su herramienta de Firebase ya que se adaptaba perfectamente a la necesidad de la aplicación.

- Desarrollo de la Aplicación: Una vez adquirido estos conocimientos necesarios, se comenzó con el desarrollo de la aplicación.

- Validación y prueba: En esta fase se validó que la aplicación funciona correctamente en diferentes dispositivos (*smartphones y tablets*) y con distintos tipos de conexión de datos.

Cada una de estas fases estaba dividida en diferentes metas semanales para cumplir con los plazos fijados antes del comienzo del trabajo.

Capítulo 6. Pruebas y resultados

6.1 Pruebas

Para realizar las pruebas se ha generado el apk de la aplicación. Gracias a Ionic y cordova este es un proceso no demasiado complicado. En primer lugar, generamos un apk con el siguiente comando:

```
ionic cordova build –release android
```

Con esto ya se genera un paquete de la aplicación, pero al intentar instalarla en el dispositivo Android se generará un error ya que no está firmada. Para firmar el paquete tan solo debemos de generarnos una clave privada con *keytool* que está dentro de JDK de Android, una vez generada ya podemos proceder a firmar el paquete con nuestra clave privada usando *jarsigner* que también está incluido en el SDK de Android.

Una vez realizado esto, solo nos queda usar la herramienta *zipalign* del SDK de Android para optimizar el APK que generar un fichero listo para ser instalado en un dispositivo Android.

Tras realizar todos estos pasos, la aplicación ha sido distribuida a varios amigos y familiares para realizar pruebas y comentarme los errores, no ha sido publicada en el *Play store* ya que lleva un coste asociado. Una vez recibido el *feedback* de los amigos y familiares se ha procedido a subsanar los bugs que podíamos encontrar. En las primeras versiones surgieron varios, pero más adelante estos cada vez se reducían más hasta llegar a una versión estable.

Finalmente, se ha utilizado una herramienta que proporciona firebase llamada *TestLab* en la cual se simulan diferentes dispositivos indicando cualquier error en la aplicación, ya sea excepciones, congelaciones, etc. Una vez pasada la aplicación por dicha herramienta nos muestra el resultado Figura 37.

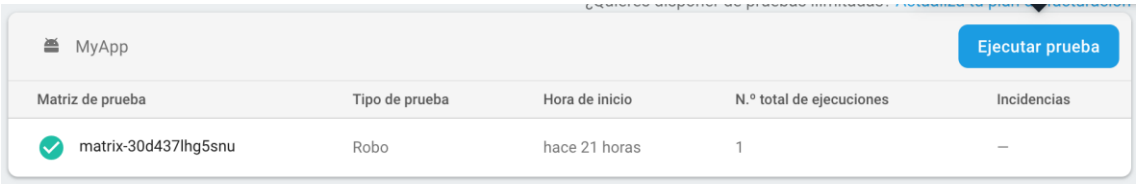

Figura 37: Resultado de TestLab analizando MEETNEAR

6.2 Resultados

Los resultados que hemos obtenido de la aplicación han sido satisfactorios tras subsanar varios *bugs*. Las primeras versiones de la aplicación estaban cargadas de errores como que el usuario no podía iniciar sesión al no conectar de forma correcta con la base de datos de Firebase, o que la aplicación no cargaba los usuarios compatibles por algunos errores en el código. Pero tras mucho depurar, y muchas horas en solucionar los errores, se ha observado que la aplicación funciona como se planteó y además en varios dispositivos móviles.

La aplicación funcionó perfectamente en los siguientes dispositivos entre otros:

- Xiaomi Mi 5s con Android 6.0
- Xiaomi Mi 8 con Android 9.0
- Samsung Galaxy Tab A con Android 8.0
- Navegador Google Chrome en OS

Finalmente un total de 22 usuarios en diferentes ubicaciones probaron la aplicación final con una media de 26 mensajes por usuario, en todos ellos la información se ha mostrado de forma correcta independientemente del dispositivo en el que lo ejecutaran.

Capítulo 7. Conclusiones y trabajo futuro

7.1 Conclusiones

Hay que hacer constar, lo importante que es internet para nosotros hoy en día, ya sea para uso personal o profesional, la posibilidad de disponer de toda la información en la palma de nuestra mano nos facilita enormemente una variedad de tareas, como en nuestro caso la de socializarse en un entorno completamente nuevo, sin conocer a personas con anterioridad.

En este trabajo fin de grado hemos explicado el procedimiento a seguir para desarrollar una aplicación móvil funcional con tecnologías web desde cero, utilizando herramientas y *frameworks opensource*, para facilitar el proceso de socialización al que se hace referencia arriba.

Gracias al uso de Ionic y Firebase hemos podido simplificar muchas tareas en el desarrollo que suelen ser bastante complejas y fastidiosas para un desarrollador. El hecho de desarrollar una nueva API desde cero, para que esta se conecte con el servidor y trate los datos, hubiera requerido mucho más tiempo de desarrollo a la vez que el aprendizaje de otra tecnología más para conseguirlo. Firebase ha facilitado mucho esta parte del proyecto, proporcionando una base de datos en la nube a la cual acceder desde el cliente, proporcionando las funciones necesarias para consultar, actualizar y subir datos a la base de datos. Además con Ionic y Apache Cordova, que tienen una gran comunidad detrás de ellos, ha sido más fácil aprender estas tecnologías, tras utilizar el primer comando observé de inmediato la potencia que hay detrás de ionic, ya que genero toda la estructura del proyecto con todas sus carpetas y archivos necesarios, implementó directamente todos los estilos nativos de Android, iOS y Windows Phone listo para usarse según el dispositivo en el que se ejecutaba y una página en blanco para comenzar a desarrollar e implementar los campos y funciones necesarias para la app.

Con este trabajo fin de grado no se ha desarrollado una aplicación, sino tres de forma paralela. Esto es posible gracias a los avances de las tecnologías móviles y web, y la posibilidad de crear estas aplicaciones híbridas visualizándolas con *web views*. Esto es algo que personalmente me parece muy interesante para pequeñas empresas que requieran que su producto sea accesible vía móvil, pero sin la necesidad de desembolsar el gasto que generarían tres aplicaciones desarrolladas de forma nativa.

No hay que decir que para conseguir un rendimiento alto en una aplicación que requiera muchos recursos del dispositivo, una aplicación desarrollada de forma nativa cumplirá mucho mejor con las exigencias de la misma. Pero para el desarrollo de este trabajo fin de grado bastaba con una aplicación sencilla, con una interfaz sencilla y que sea fácilmente accesible.

Gracias a este trabajo fin de grado, puedo decir que he aumentado de forma considerable mis conocimientos en muchos ámbitos del desarrollo de software, ya que en cada fase del desarrollo se ha tocado alguna rama del desarrollo de una aplicación completa: la base de datos, el diseño, la inicialización y el mantenimiento de los paquetes, el desarrollo con MVC, etc.

En resumen, un proyecto bastante completo en el cual se ha desarrollado una aplicación totalmente funcional utilizando tecnologías nuevas las cuales no había usado anteriormente en profundidad.

He sido capaz de ver un problema real y solucionarlo de la forma más eficaz posible desde mi punto de vista, y creando una aplicación que podría ser interesante para muchas personas que se encuentran en una situación como la que yo he pasado.

He disfrutado bastante aprendiendo las funcionalidades y el lenguaje de TypeScript y AngularJS para este proyecto, y me he dado cuenta de que con ganas y pasión y unos conocimientos básicos de programación es muy factible crear una aplicación híbrida basada en tecnologías web decente.

7.2 Trabajo Futuro

- Bloqueo de usuarios: Está claro que muchos usuarios utilizarán la aplicación para algunos fines alternativos a su idea, por lo que, si un usuario está descontento con la actitud de otro, sería una buena idea implementar un botón de “reportar” para que el comportamiento del mismo pueda ser comprobado y monitorizado y en caso de infringir el fin de la aplicación el usuario sea eliminado.

- Servidor y API Node.js: Debido al uso de firebase, se ha ahorrado mucho tiempo en el desarrollo de una API y la configuración de una base de datos. Aun así, sería interesante tener una API propia con la cual agilizar algunas llamadas, además sería interesante que muchos cálculos se hicieran desde el servidor para que la aplicación fuese más ágil y no requiera obtener tantos datos de la base de datos para luego desecharlos.

- Versión de escritorio: Ya que estamos utilizando el Ionic framework, es posible crear una aplicación web para acceder desde el navegador, pero debido a que el framework está más bien pensado para dispositivos móviles, en el navegador todo se vería una forma muy estirada y no adaptada a las pantallas de los ordenadores actuales, por lo que sería interesante hacer una versión web adaptada al escritorio utilizando tecnologías como HTML, CSS y JavaScript o incluso AngularJS y así tener una versión web adaptada plenamente al escritorio de un ordenador y al uso con ratón y teclado.

Capítulo 8. Bibliografía

[1] Documentación Ionic 3. https://ionicframework.com/docs/ . Agosto 2018

[2] Tutorial en YouTube de Ionic https://www.youtube.com/playlist?list=PLYxzS__5yYQng-XnJhB21Jc7NW1OIaqct . Septiembre 2018

[3] GitHub. https://github.com/ Septiembre 2018

[4] GitHub | AngularFire. https://github.com/angular/angularfire2 Septiembre 2018

[5] Angular. https://angular.io/ Septiembre 2018

[6] Documentación npm. https://docs.npmjs.com/ Septiembre 2018

[7] Documentación sobre CSS https://www.w3schools.com/cssref/ Octubre 2018

[8] Documentación sobre HTML https://devdocs.io/html/ Octubre 2018

[9] Building Ionic apps with Firestore https://blog.ionicframework.com/building-ionic-apps-with-firestore/ . Septiembre 2018

[10] Documentación Apache Cordova. https://cordova.apache.org/docs/es/3.1.0/guide/platforms/index.html Enero 2019.

[11] Documentación Firebase https://firebase.google.com/docs/?hl=es-419 Enero 2019

[12] Documentación TypeScript https://www.typescriptlang.org/docs/home.html Enero 2019

[13] Información sobre SCRUM https://metodologiascrum.readthedocs.io/en/latest/ Enero 2019