Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Aplicación móvil para facilitar la búsqueda de responsables de compras y proveedores de empresas

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Alejandro Javier Vicente Macián

Tutor: David de Andrés Martínez

Cotutor: Juan Carlos Ruíz García

2019-2020

Resumen

Mediante el siguiente trabajo de final de grado, se explicará todo el proceso seguido para desarrollar una aplicación móvil que permita a los responsables de compras de una empresa gestionar sus clientes actuales y, al mismo tiempo, poder buscar nuevos. A su vez, las empresas se podrán registrar también en la aplicación para que esta les sirva como escaparate para darse a conocer a potenciales nuevos proveedores.

Con este fin en mente y para dar soporte a la aplicación que desarrollaremos, se introducirá toda la información de los usuarios dados de alta en nuestro servicio en una base de datos en la nube para que sea accesible desde la aplicación móvil.

Palabras clave: Ionic, Firebase, Android, aplicación móvil.

Abstract

Through the following final grade work, we will explain all the process followed to develop a mobile application that allows the purchasing managers of a company, arrange all their current customers and at the same time to be able to look for new ones. At the same time, the companies will be able to also register in the application, so it serves them as showcase to get to know potential new customers.

With this purpose in mind, and to support the application that we will develop; all the information gathered from the users registered in our application will be stored in a cloud database, so it will be accessible from the mobile application.

Keywords: Ionic, Firebase, Android, mobile application.

Tabla de contenidos

Tabla de figuras

Tablas

1 Introducción

Una de las cosas que más ha cambiado nuestra vida en los últimos años es la aparición de los teléfonos móviles inteligentes o smartphones. Gran parte de las cosas que antaño hacíamos con nuestros ordenadores, ya fueran portátiles o sobremesa, han ido poco a poco siendo realizadas por nuestros teléfonos. Desde cosas tan del día a día como hacer la compra, hasta buscar en un mapa el destino de nuestro viaje.

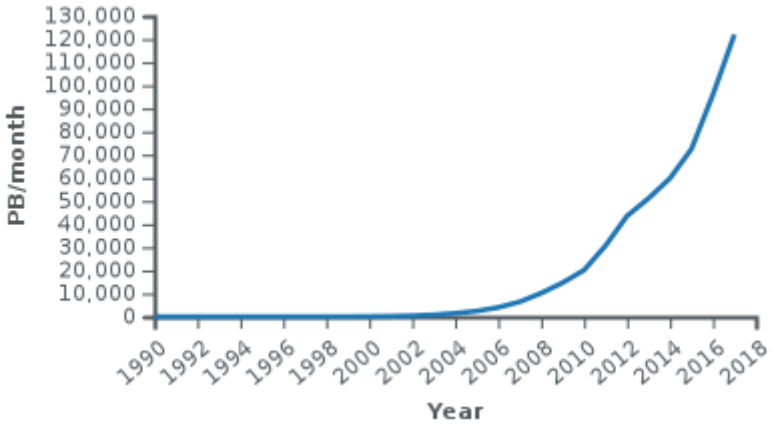

Figura 1. *Tráfico de internet [1]*

Como se puede observar en el gráfico de la figura 1, cada año hay más tráfico en internet, hecho que favorece que más y más sectores se apunten a cambiar sus modelos de negocio a uno basado en internet.

Cogiendo datos del año 2017, el sector textil ha crecido hasta ser el tercer sector que más porcentaje del total de sus ventas genera en internet con un 7%, solo por detrás de las agencias de viajes (15,6%) y el transporte aéreo (9,9%) [2]. Esto nos indica que si el sector textil, que es uno de los sectores más tradicionales en el sentido de que es costoso hacer cambiar al público de que compre en tiendas físicas, está consiguiendo esos números, el crecimiento de internet no va a parar

Por otro lado, nos damos cuenta de que el sector bancario cada vez apuesta más por los sistemas de la información para captar más clientes. La aparición de cuentas que se contratan por internet y se manejan exclusivamente desde ahí cada vez son más numerosas y están promoviendo cambios en el sector para actualizar sus plataformas.

Es por eso por lo que parece extraño que la manera de buscar nuevos clientes por parte de las empresas no haya cambiado demasiado en los últimos años, así como la manera que tienen las empresas de darse a conocer. Bajo este contexto nace la idea de crear una plataforma para facilitar dicha búsqueda, y qué mejor manera de hacerlo, que bajo una interfaz móvil.

Con esto en mente, la finalidad de este trabajo será la creación de una aplicación móvil que cumpla con este cometido y que sirva como punto de encuentro entre diferentes empresas.

1.1 Presentación/Motivación

La profesión de mi padre es la de representante de empresa y desde hace muchos años me he ido fijando en que parte de su trabajo consiste en buscar nuevos clientes para la empresa para la cual trabaja.

Esta búsqueda normalmente la hace por internet mediante un buscador web y básicamente contactando con los resultados que este le ofrece.

Tras observar esto, me parecía extraño que no hubiera ninguna aplicación que explotara esta necesidad y le facilitara dicho trabajo y me pareció un proyecto adecuado para llevar a cabo como proyecto de final de grado.

1.2 Objetivos

El objetivo principal de este trabajo es el de crear un portal de contacto para los responsables de compras de las empresas y modernizar la manera en la que se buscan nuevos clientes por parte de ellas. Todo esto lo conseguiremos gracias al desarrollo de una aplicación móvil que se conecte a una base de datos compartida por todos los usuarios de la aplicación. Esto permitirá que la búsqueda de nuevos clientes se pueda realizar desde un lugar destinado explícitamente a ello.

Por otro lado, al conseguir resolver este problema, las empresas también dispondrán de una plataforma en la que darse a conocer al resto del mundo, complementándose con los métodos de marketing más tradicionales que hay en el sector.

Por último, la aplicación deberá ser capaz también de guardar, a modo de agenda personal, los clientes de los usuarios que no estén registrados en la aplicación, disponiendo así de un lugar en el que guardarlos alejado de otros que se puedan extraviar, dañar, etc.

1.3 Estructura del documento

El documento que a continuación se presenta está dividido en los capítulos mencionados a continuación:

- Introducción: Aquí pondremos en contexto los objetivos, así como los motivos y necesidades que han propiciado realizar este trabajo
- Estado del arte: En este apartado realizaremos un estudio de cómo es la situación actual en productos similares al que vamos a desarrollar.
- Especificación: A lo largo de este capítulo, haremos el análisis teórico del proyecto.

- Tecnologías: Durante este apartado se introducirán las tecnologías que se van a usar para cumplir los objetivos, así como una breve introducción al funcionamiento de estas.
- Desarrollo: Aquí explicaremos los aspectos más importantes del desarrollo de la aplicación, así como problemas que hayamos tenido durante el mismo.
- Resultados: Durante este capítulo se compararán resultados de la aplicación reales, con aquellos que se introdujeron en la especificación.
- Conclusiones: En este capítulo analizaremos el trabajo y valoraremos el alcance de este, así como posibles mejoras a realizar de cara a un futuro.

2 Estado del arte

El hecho de disponer de un ordenador en nuestro bolsillo las 24 horas del día ha cambiado la forma en la que hacemos las cosas. Desde cosas tan simples como hacer la compra, han sido reemplazadas por sus homólogos en aplicaciones móviles, por lo que parece curioso que no haya una plataforma dedicada exclusivamente a que las empresas, independientemente del ámbito de los bienes que estas generen, puedan darse a conocer al resto del mundo.

Esta idea es la precursora de *ClientHub,* una plataforma que dichas empresas puedan utilizar como escaparate, así como una solución a los trabajadores que necesitan tener algún lugar donde guardar todos sus clientes y proveedores anotados.

Durante este capítulo intentaremos poner en situación al lector sobre el panorama actual en el que se encuentra tanto el desarrollo de aplicaciones para dispositivos móviles, así como una breve introducción a aplicaciones que ya existan y que propongan una solución similar a la que estamos planteando.

2.1 Fragmentación del mercado

Una de las primeras decisiones que debemos de tomar a la hora de empezar a desarrollar una aplicación móvil es elegir para qué sistema operativo la desarrollaremos.

El mercado tiene hoy en día dos claros exponentes que ocupan casi el 100% de los dispositivos móviles. Estos son *Android* y *iOS*.

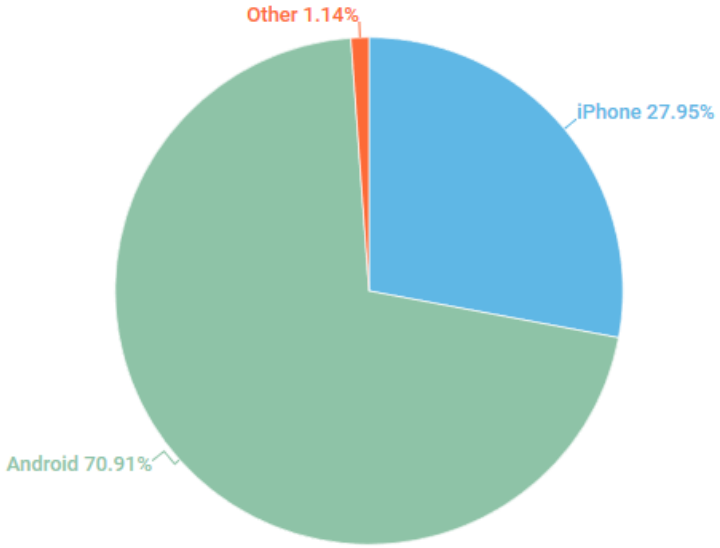

Figura 2. *Cuota de mercado Sistemas Operativos Europa [6]*

Como se puede ver en la estadística de la figura 2, Android es claramente el sistema operativo más utilizado por móviles en el mercado europeo. Pero, curiosamente, si miramos las ganancias de las aplicaciones en el mismo mercado, los datos están bastante más igualados.

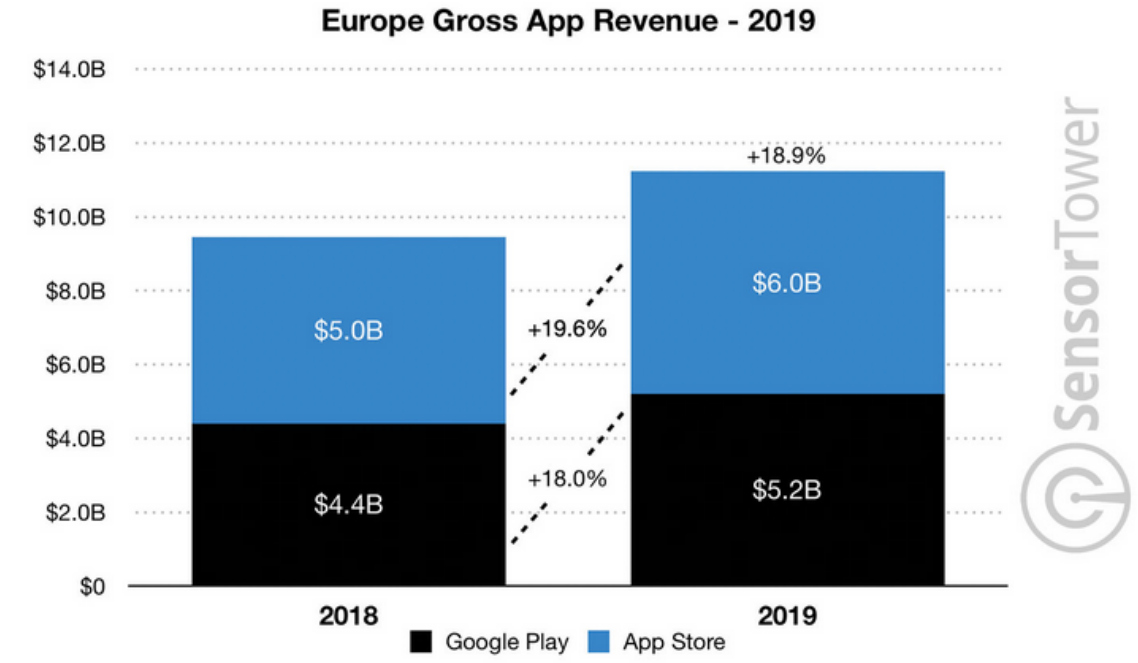

Figura 3. *Beneficio bruto de aplicaciones en Europa [5]*

La estadística de la figura 3 nos sugiere que en este caso prescindir de aproximadamente un 30% de los usuarios nos haría renunciar a algo más de la mitad de los beneficios de la aplicación que decidiésemos desarrollar.

Excluir alguno de los sistemas operativos existentes sería un error de cara a los beneficios que pudiera tener nuestra aplicación en un futuro. Es aquí donde se nos abren dos opciones para llegar a ambos sistemas operativos: desarrollar dos aplicaciones de manera nativa o utilizar ciertas tecnologías que nos permitan desarrollar una sola aplicación para los dos sistemas operativos.

Las ventajas de desarrollar una aplicación de manera nativa son bastante claras. Primeramente, dichas aplicaciones, al estar ejecutándose en el entorno para el que fueron diseñadas, funcionan con un rendimiento superior. Por otro lado, no dependemos de factores externos que puedan suponer problemas como una actualización de la plataforma que utilicemos para hacer estas aplicaciones híbridas.

Pero no todo son ventajas al desarrollar aplicaciones nativas ya que, para empezar, estaríamos enfrentándonos a hacer el doble de trabajo porque tendríamos que llevar dos desarrollos totalmente paralelos para conseguir tener ambos sistemas cubiertos. Otro de los problemas que se plantean es el coste de mantener dos aplicaciones. Por muy pequeño que sea este coste, se dobla al tener que necesitar el doble de personas para mantener dos aplicaciones.

Por otro lado, una solución híbrida nos ayuda a reducir este coste de mantenimiento reduciendo el número de personas que han de trabajar en él, así como también nos otorga muchas facilidades a la hora de lanzar la aplicación en menos tiempo debido a que solo tenemos que hacer un único desarrollo para ambas plataformas.

Los problemas de utilizar este último tipo de soluciones es la pérdida de rendimiento en la aplicación, así como tener una dependencia de una tercera parte que es la encargada de desarrollar la solución híbrida que usaremos, lo que podría ocasionar que, si esta tercera parte cesa el desarrollo de su solución, nos veamos afectados y tengamos que rehacer la aplicación desde cero.

Sopesando ventajas y desventajas de los dos tipos de métodos, se ha decidido utilizar para desarrollar la aplicación una solución híbrida. La ventaja más importante de utilizar este método es que no tenemos que hacer dos desarrollos diferentes y en este caso, tenemos un tiempo limitado y tan solo una persona trabajando en el desarrollo, por lo que hacer dos aplicaciones podría llegar a ser una carga de trabajo muy elevada. Por otro lado, una de las desventajas, que es la pérdida de rendimiento, no entrará en juego en la aplicación que tenemos en mente, ya que como se verá en futuros capítulos, la aplicación no tendrá ningún proceso crítico que requiera de demasiados recursos del teléfono móvil.

Es por estos motivos que, para llegar al mayor número de personas durante el trabajo, se implementará la aplicación de manera híbrida para que, de este modo, podamos llegar al mayor número potencial de usuarios y que gracias a esto, en un futuro tener también un mayor beneficio.

2.2 Aplicaciones similares

A continuación, en el siguiente apartado trataremos de examinar aplicaciones con funcionalidades similares a la que queremos desarrollar para tratar de discernir qué es lo que hace que una aplicación triunfe en el sector.

2.2.1 ClientiApp

***Figura 4**. Logo de ClientiApp*

ClientiApp [4] es una aplicación disponible para su descarga en *Google Play*. La aplicación tiene una valoración de 4,1 sobre 5 en dicho portal de descargas y más de 50,000 usuarios. Uno de los puntos más a favor de la aplicación es que tiene una interfaz muy moderna e intuitiva como podemos ver en la figura 5. La aplicación permite tomar nota de manera manual de los datos de un cliente y guardarlos de manera permanente en el dispositivo del usuario. Dichos datos van desde el nombre de la empresa hasta sus coordenadas para navegar por ella mediante cualquier dispositivo con GPS.

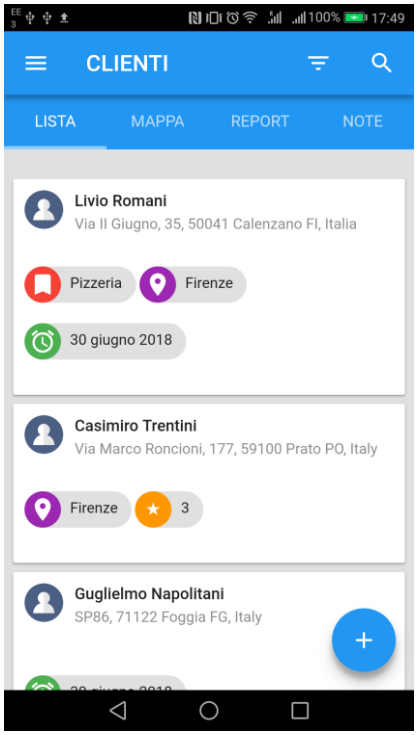

***Figura 5**. ClientiApp*

No obstante, la aplicación no cubre la necesidad clave que nosotros buscamos, que es dar soporte a las empresas que se registren para darse a conocer. A parte de esto, la aplicación tiene una serie de características, a priori muy interesantes, detrás de una versión de pago. Cosas como guardar los datos de nuestros clientes en una base de datos en la nube para que en el caso de que cambiemos de dispositivo móvil no perderlos, o la ya mencionada funcionalidad de GPS, están detrás de dicha versión de pago.

Por todo esto, la aplicación de ClientiApp nos parece una buena opción en la que fijarnos para hacer el diseño estético de nuestro proyecto, pero fuera de esto es relativamente limitada.

En resumen, podríamos decir que los puntos clave de la aplicación son:

- Disponible solo para Android
- Interfaz de usuario muy pulida y atractiva
- Posibilidad de guardar datos en la nube con la versión de pago
- Versión gratuita muy limitada
- Creación de empresas con mucha personalización
- No podemos buscar empresas registradas

2.2.2 LinkedIn

Figura 6. *Logo de LinkedIn*

LinkedIn [3] es una de las aplicaciones de referencia a la hora de buscar empleo. La aplicación está disponible en la tienda de aplicaciones de Android, al igual que en la de iOS. Tiene una valoración de 4,3 estrellas en el Google Play, así como más de 500 millones de descargas en este último.

La aplicación cuenta con una interfaz gráfica bastante atractiva pese a tener multitud de opciones por pantalla disponibles al mismo tiempo, como observamos en la figura 7, lo que puede dificultar el proceso de adaptación a los usuarios más inexpertos. También es interesante el que nos pida registrarnos antes de acceder a la misma, dado que de esta manera sabemos que al cambiar nosotros de dispositivo nuestros datos permanecerán guardados.

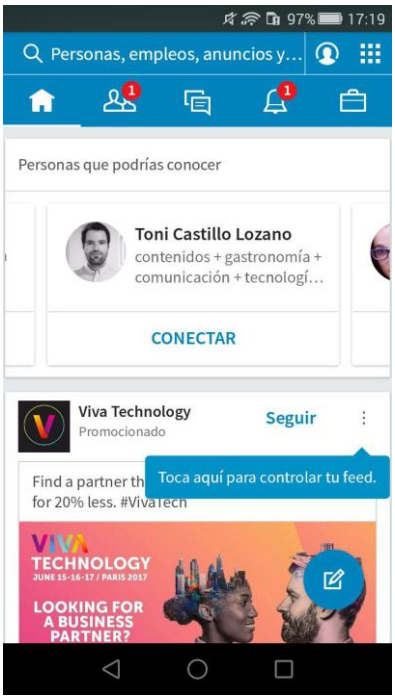

Figura 7. *LinkedIn*

Contamos con un apartado de búsqueda que nos ayuda a filtrar las empresas registradas con bastante facilidad (nombre, distancia, sector, etc.).

No obstante, LinkedIn no está enfocado al mismo público al que queremos atraer con nuestra aplicación, por lo que muchas de las funcionalidades que podríamos desear en nuestra aplicación no están presentes. Cosas como seguir a empresas o crear empresas que solo tú puedes ver para consultarlas en cualquier momento, no son funcionalidades que esta aplicación ofrece.

A modo de resumen sobre LinkedIn, podemos decir que:

- Disponible tanto en Android como en iOS
- Interfaz pulida, pero con demasiadas opciones simultáneas
- Datos del usuario guardados en la nube
- Aplicación gratuita
- No enfocada al mismo tipo de público que nuestra aplicación
- Imposibilidad de crear empresas que solo el usuario pueda ver
- Filtros de empresas muy completos

2.2.3 Comparativa de las aplicaciones

Tras un breve análisis de algunas de las aplicaciones similares que hay en el mercado, podemos decir que algunas de las claves a la hora de conseguir usuarios para cualquier aplicación son: la capacidad de llegar al mayor número de usuarios posibles pudiéndose

consumir desde iOS y Android, tener una interfaz atractiva e intuitiva y que la aplicación sea gratuita.

Respecto a nuestra aplicación, hemos visto que en situaciones similares deberíamos poder guardar nuestros datos en la nube para la comodidad del usuario, crear empresas de manera personal para ampliar la funcionalidad y poder buscar a las empresas registradas, que es el propósito principal de la aplicación que desarrollaremos.

Toda esta comparativa queda recogida en la tabla 1 presentada a continuación:

***Tabla 1**. Comparativa de aplicaciones*

	ClientiApp	LinkedIn	ClientHub
Disponible en Android + iOS	✗	✓	✓
Interfaz gráfica simple y limpia	✓	✗	✓
Guardar datos en la nube	✗	✓	✓
Aplicación gratuita	✗	✓	✓
Creación de empresas personales	✓	✗	✓
Búsqueda de empresas registradas	✗	✓	✓

3 Especificación

Durante el siguiente capítulo se va a explicar cómo se irán plantando poco a poco los cimientos de nuestra aplicación. Parte de esto pasa por señalar la arquitectura sobre la cual se va a desarrollar la aplicación, así como tomar los requisitos de esta.

Gracias al trabajo realizado en este capítulo, el periodo de desarrollo será mucho más claro en cuanto a las funcionalidades deseadas que ha de tener la aplicación.

3.1 Arquitectura a tres capas

La mayoría de aplicaciones hoy en día utilizan un modelo software a tres capas [9]. Esto es, dividir todas las funcionalidades de la aplicación en tres secciones bien diferenciadas:

- Persistencia: es la encargada de guardar, modificar y borrar los datos de los usuarios. Normalmente aquí se engloba todo lo que tiene que ver con los gestores de bases de datos

- Negocio: en esta capa es donde reside el verdadero funcionamiento de la aplicación. Esta capa se comunica tanto con la de presentación como con la de persistencia para gestionar todas las peticiones que hacen los usuarios.

- Presentación: aquí será donde los usuarios interactúen con la aplicación. Esta comunicará a la capa de negocio sobre las acciones de los usuarios para que esta última tome las acciones necesarias.

Pero, ahora bien ¿Por qué utilizamos este modelo a tres capas?

Son muchos los beneficios que obtenemos gracias a hacer un desarrollo bajo una arquitectura a tres capas, la cual en nuestro caso, ha sido elegida tanto por la familiaridad con ella, como por las siguientes características:

La primera ventaja que se obtiene es el desacoplamiento de las capas. El hecho de que las capas sean independientes entre sí nos otorga una gran flexibilidad a la hora de cambiar cosas en el desarrollo, tales como actualizar alguna de las capas a una versión más moderna. Esto es algo que, gracias a esta arquitectura, se hace una tarea más sencilla que si no estuviéramos utilizando este modelo.

Otra de las ventajas de utilizar este modelo es la facilidad de realizar pruebas sobre cualquier capa de manera independiente. Localizar y solucionar un problema con este tipo de arquitectura es más simple que si la aplicación no estuviera dividida, ya que siempre podemos saber en qué capa ha surgido el problema y atacarlo de la forma más adecuada posible.

Finalmente, gracias a esta arquitectura, podemos reutilizar el código de una manera más eficiente. Debido a que las capas no dependen las una de las otras, procesos similares pueden ser agrupados y así ahorrar tiempo a la hora de hacer el desarrollo.

3.2 Diagrama de casos de uso

Son muchas las maneras en que podemos plasmar de manera visual la especificación de nuestra aplicación y una de las más extendidas es la de realizar un diagrama de casos de uso [7]. En nuestro caso, nos basaremos en el estándar de UML para su realización.

Dicho diagrama plasma una serie de actores que son los distintos tipos de usuarios que utilizarán nuestra aplicación, una serie de casos de uso representados con círculos, los cuales indican funcionalidades que tendrá nuestro producto, así como una serie de relaciones entre actores y casos de uso representadas con líneas.

De entre estas relaciones hay que destacar las dirigidas. Estas relaciones dirigidas tratan de darle algo más de significado a dicha relación y van acompañadas de una palabra clave. Dos de las más importantes son:

- <<extends>>: Cuando una relación está marcada con el atributo extends nos indica que el caso del que sale la flecha es condicional al caso de uso al que se dirige la flecha y que, por lo tanto, solo pasará en algunas ocasiones

- <<include>>: El atributo include nos indica que el caso de uso al que se dirige la flecha es consecuencia directa de haber accedido al caso de uso del que parte la flecha.

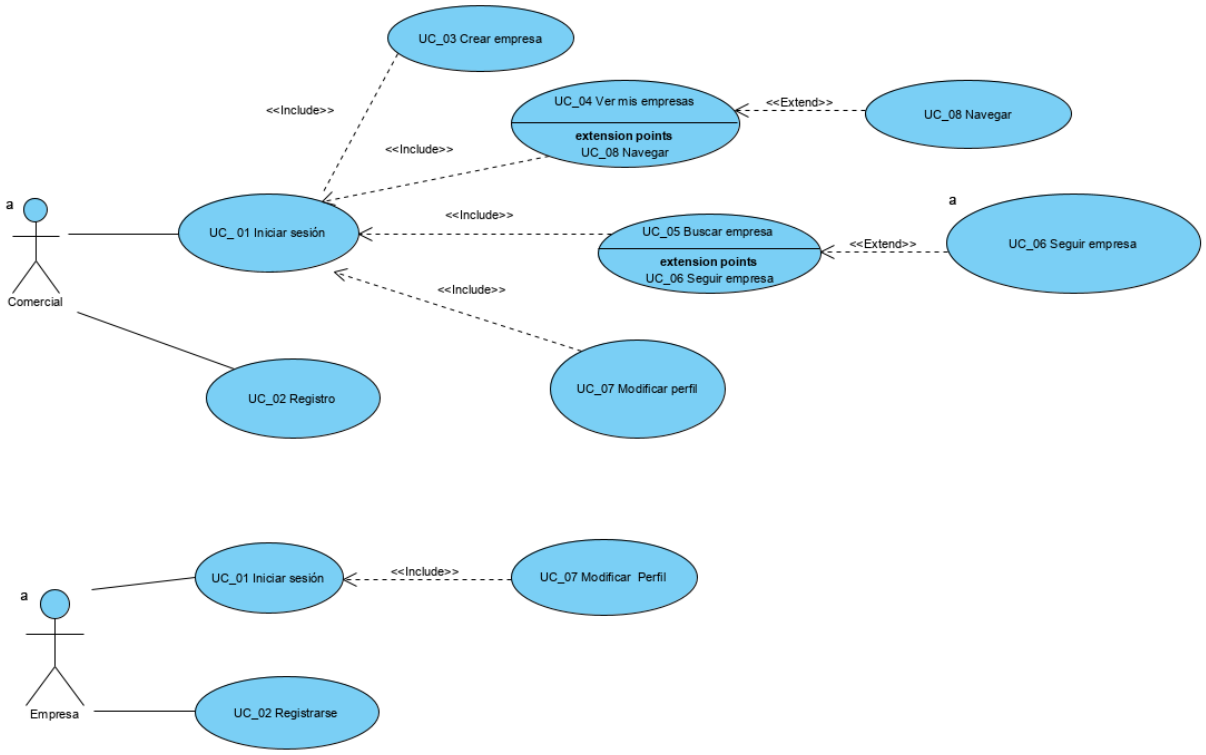

***Figura 8**. Diagrama de casos de uso*

3.3 Casos de uso

Para concretar cada tipo de caso de uso representado en la figura 8, a continuación, se procederá a dar una breve explicación de cada uno de ellos para que de esta manera se obtenga una visión más exacta de las funcionalidades que incorporará nuestra aplicación.

Tabla 2. *Caso de uso iniciar sesión*

Nombre del caso	UC_01 Iniciar sesión
Actores	Comercial y Empresa
Relaciones con otros casos	UC_03 UC_04 UC_05 UC_07
Explicación	Todo usuario debe poder iniciar sesión en la aplicación dando a conocer su correo electrónico, así como su contraseña.
Flujo básico	**1) Introducción de los datos** El sistema solicitará al usuario su email y su contraseña. **2) Inicio de sesión** El sistema comprobará si los datos son correctos y, en caso de serlo, procederá a redirigir al cliente a la pantalla de administración.
Flujo alternativo	**1) Problema con los datos** El sistema detecta que los datos no son correctos y mostrará por pantalla el error que se ha cometido
Precondiciones	• Debe de haber conexión a internet • El usuario debe estar registrado
Postcondiciones	• La aplicación quedará en la página de administración

Tabla 3*. Caso de uso registro*

Nombre del caso	UC_02 Registro
Actores	Comercial y Empresa
Relaciones con otros casos	———
Explicación	Los usuarios deben poder darse de alta en la aplicación dando su correo electrónico y su contraseña.
Flujo básico	**1) Introducción de los datos**

	El sistema solicitará al usuario el email con el que registrarse y la contraseña. **2) Registro en la base de datos** El sistema registrará al usuario en la base de datos si el email no está en uso. **3) Inicio de sesión** El sistema procederá a iniciar sesión por el usuario y redirigirle a la página de administración.
Flujo alternativo	**1) Problema con los datos** Si el sistema detecta que la contraseña es muy corta o el email está en uso, se informará de esto al usuario.
Precondiciones	• Debe de haber conexión a internet • El email no ha de estar en uso • La contraseña debe tener ocho o más caracteres
Postcondiciones	• Los datos del usuario quedarán grabados en base de datos • La aplicación quedará en la pantalla de administración

***Tabla 4**. Caso de uso crear empresa*

Nombre del caso	UC_03 Crear empresa
Actores	Comercial
Relaciones con otros casos	UC_01
Explicación	Un comercial deberá poder crear un registro de una empresa a la cual solo él tendrá acceso para su visualización.
Flujo básico	**1) El usuario pulsa sobre el botón añadir empresa** El sistema redirige al usuario a la pantalla de crear empresa **2) El usuario rellena el formulario** El sistema comprueba que todos los campos obligatorios están completados **3) El usuario crea la empresa** El sistema registra los datos del formulario en base de datos
Flujo alternativo	**1) El usuario no introduce algún campo obligatorio** El sistema informará al usuario de los campos obligatorios
Precondiciones	• Debe de haber conexión a internet

	• El usuario habrá iniciado sesión • Debe de haber señal GPS
Postcondiciones	• Los datos quedarán grabados en base de datos • Se redirigirá a la pantalla mis empresas

Tabla 5. *Caso de uso ver mis empresas*

Nombre del caso	UC_04 Ver mis empresas
Actores	Comercial
Relaciones con otros casos	UC_01 UC_08
Explicación	El usuario debe poder visualizar todas las empresas; tanto que ha creado él mismo, como las que sigue en una lista.
Flujo básico	**1) El usuario accederá a la pantalla mis empresas** El sistema mostrará por pantalla las empresas registradas en la aplicación
Flujo alternativo	———
Precondiciones	• Debe de haber conexión a internet • El usuario habrá iniciado sesión
Postcondiciones	———

Tabla 6*. Caso de uso buscar empresa*

Nombre del caso	UC_05 Buscar empresa
Actores	Comercial
Relaciones con otros casos	UC_01 UC_06
Explicación	El usuario deberá poder filtrar las empresas que haya registradas en la aplicación con una serie de criterios que se le ofrecerán por pantalla
Flujo básico	**1) El usuario pulsa sobre el botón filtrar** El sistema muestra en pantalla todos los filtros disponibles **2) El usuario selecciona los filtros a aplicar** El sistema selecciona todas las empresas que encajen con los filtros **3) El usuario confirma la aplicación de los filtros**

	El sistema muestra por pantalla los resultados filtrados
Flujo alternativo	———
Precondiciones	• Debe de haber conexión a internet • El usuario habrá iniciado sesión • Debe de haber alguna empresa con los filtros elegidos
Postcondiciones	• Se mostrarán por pantalla todas las empresas filtradas

***Tabla 7**. Caso de uso seguir empresa*

Nombre del caso	UC_06 Seguir empresa
Actores	Comercial y Empresa
Relaciones con otros casos	UC_01 UC_05
Explicación	De los resultados obtenidos al buscar empresas, el usuario deberá poder seleccionar alguno de ellos y seguirlos de manera que se mostrarán en la lista de UC_04
Flujo básico	**1) El usuario selecciona una empresa a seguir** Se abren los datos de la empresa, mostrando el botón de seguir **2) El usuario sigue a la empresa** El sistema guarda en la base de datos una referencia a la empresa seguida
Flujo alternativo	———
Precondiciones	• Debe de haber conexión a internet • El usuario habrá iniciado sesión
Postcondiciones	• Quedará una referencia en la base de datos de la empresa seguida

***Tabla 8**. Caso de uso modificar perfil*

Nombre del caso	UC_07 Modificar perfil
Actores	Comercial y Empresa
Relaciones con otros casos	UC_01
Explicación	El usuario tiene que poder modificar las credenciales con las que se registró en la aplicación.

Flujo básico	**1) El usuario modifica un campo de su perfil** El sistema actualiza los datos del formulario **2) El usuario acepta los cambios** Los cambios son trasladados a base de datos
Flujo alternativo	**1) El usuario hace un cambio no permitido** El sistema rechaza los cambios e informa al usuario del error
Precondiciones	• Debe de haber conexión a internet • El usuario habrá iniciado sesión
Postcondiciones	• Se modificarán los datos en base de datos

***Tabla 9**. Caso de uso navegar*

Nombre del caso	UC_08 Navegar
Actores	Comercial
Relaciones con otros casos	UC_04
Explicación	El usuario debe poder seleccionar una empresa y a continuación navegar a ella mediante su aplicación predefinida de navegación.
Flujo básico	**1) El usuario selecciona una empresa** Se abre una pantalla con los detalles de la empresa **2) El usuario pulsa sobre la localización de la empresa** El sistema informa con qué aplicación quiere completar la acción **3) El usuario elige con que aplicación navegar** El sistema abre dicha aplicación de navegación con los datos de la localización precargados
Flujo alternativo	**1) La empresa no dispone de localización** El sistema informará de la falta de localización **2) No hay conexión GPS** El sistema cederá a la aplicación de navegación qué toma de acción hacer
Precondiciones	• Debe de haber conexión a internet • El usuario habrá iniciado sesión

	• Debe de haber conexión GPS
Postcondiciones	• El sistema cederá el control a la aplicación de navegación

3.4 Modelo de datos

Conocidas las acciones principales que podrán hacer los usuarios de nuestra aplicación, así como los tipos de usuarios que esta va a tener, llega la hora de describir el modelo de datos que vamos a utilizar para poder realizar esas acciones.

Siguiendo con el estándar UML, vamos a utilizar un diagrama de clases [8] para especificar el modelo de datos que va a tener la aplicación. Un diagrama de clases es una representación visual de los objetos de los que constará nuestro sistema, los atributos que tendrán dichos objetos y las relaciones que estos mantendrán entre ellos.

Como se ha explicado antes en el diagrama de clases que utilizaremos para describir nuestro modelo de datos, tendremos los siguientes elementos:

- Clase: En un diagrama de clases, la clase representa una serie de objetos los cuales tendrán una seria de atributos comunes, así como un comportamiento similar. Las clases son representadas con un rectángulo, el cual está dividido en dos secciones, en la que la primera contiene el nombre de la clase y la segunda los atributos de esta.
- Atributos: Los atributos son los datos que contendrá la clase. Estos suelen tener un nombre que los identificará dentro de la clase y un tipo de datos que suelen tener (booleanos, numéricos, caracteres…).
- Relaciones: Una relación entre dos clases, representada con una línea entre ellas representa, como el propio nombre indica, un vínculo entre estas clases. En nuestro caso, estas relaciones irán acompañadas de unos números que indicarán la cantidad de objetos de una clase que se relacionarán con la otra.

3.5 Diagrama de clases

Tras el análisis de las necesidades que tendrá nuestra base de datos para poder representar toda la información que necesitaremos, obtenemos un diagrama de clases como el siguiente:

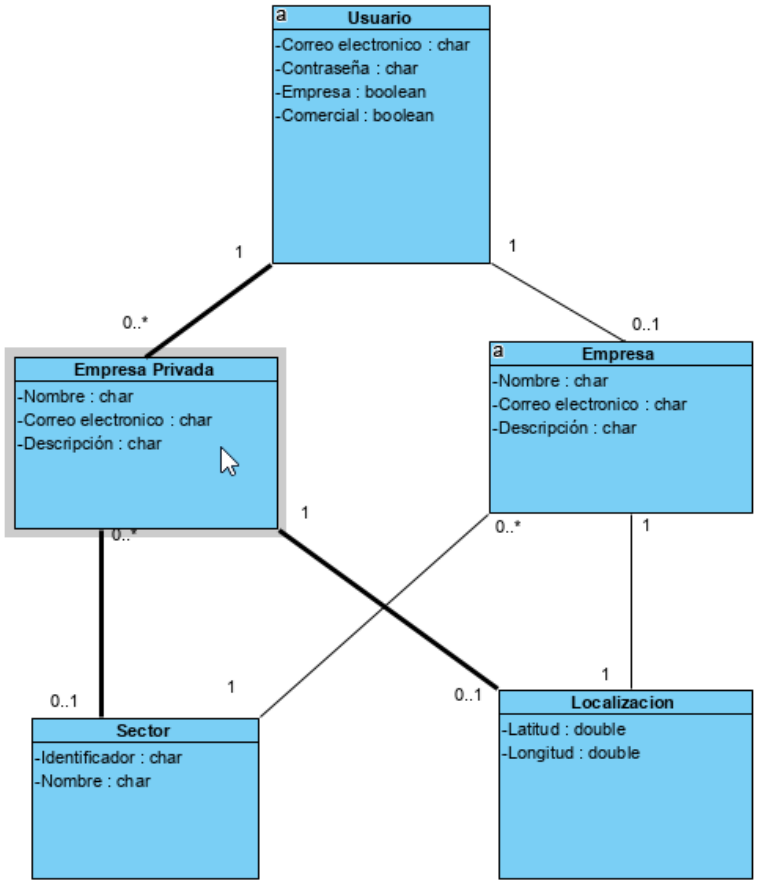

Figura 9. *Diagrama de clases*

Como podemos observar en la figura 9, nuestra aplicación tendrá 5 clases que estarán relacionadas entre ellas según las líneas que podemos observar.

Para empezar, la clase central de nuestro diagrama es la clase usuario. Dicha clase representará a cada usuario de la aplicación y, como podemos ver, contiene tanto su correo electrónico (que hará las veces de nombre de usuario), así como la contraseña. Por otro lado, los otros dos atributos nos permitirán diferenciar si el usuario es una empresa o es un comercial para saber qué funcionalidades tendrá disponibles.

Tanto la clase empresa como la clase empresa privada tienen los mismos atributos que son los que hemos considerado necesarios a la hora de dar de alta a una empresa. La diferencia y motivo por el que hemos optado separar estas dos clases en entidades diferentes es por la relación que tienen con la clase usuario dado que, únicamente cuando el campo empresa sea verdadero, habrá una relación entre la clase usuario y la clase empresa.

Por otra parte, la clase sector tendrá contenidos todos los posibles sectores que consideremos en los que puede estar una empresa y se vinculará con las clases empresa y empresa privada.

Por último, y con el fin de poder registrar la posición de las empresas para más tarde poder utilizar la aplicación de navegación de nuestro teléfono, se ha incluido una clase

que contendrá las latitudes y longitudes de las empresas que se registren en la aplicación.

3.6 Prototipos

Una vez obtenidos los requisitos de la aplicación, debemos realizar los prototipos que nos guiarán durante el desarrollo. Pero ¿por qué hacer estos prototipos?

Primeramente, es recomendable empezar una aplicación teniendo en mente unos prototipos sobre los que poder apoyarnos, ya que guiarán el camino que debemos seguir y evitarán que pongamos funcionalidades extra que no habían sido seleccionadas durante la obtención de requisitos.

Por otro lado, también sirve hacer prototipos para evitar hacer cambios en fases más avanzadas del desarrollo. Todo esto nos ayuda a evitar el tiempo extra que cuesta añadir una nueva funcionalidad que teníamos en mente, pero que no habíamos pensado incluir. Como se puede ver a continuación en la figura 10, el coste de realizar cambios crece conforme más avanzados estamos en el proceso de desarrollo de la aplicación.

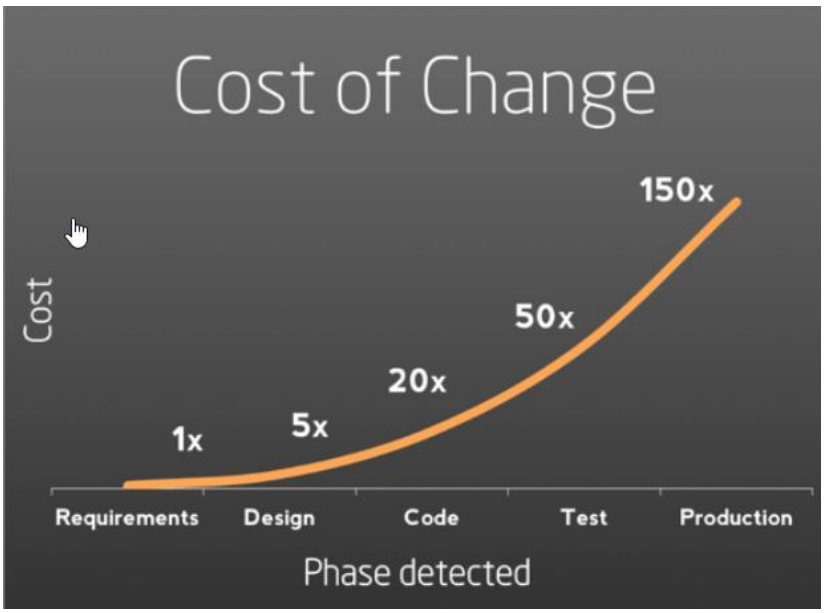

Figura 10. *Coste de cambio en el tiempo [10]*

3.6.1 Inicio de sesión y registro

Lo primero que debemos diseñar son las pantallas para que los usuarios se registren o se identifiquen en la aplicación. Estas pantallas simplemente deberán tener unos espacios para que los usuarios pongan sus credenciales y un botón para enviarlas al servidor de autenticación. Por otro lado, y para navegar entre ambas pantallas, habrá

otro botón que nos permitirá cambiar entre estas. La figura 11 nos muestra un diseño aproximado de cómo serán estas vistas una vez acabada la aplicación.

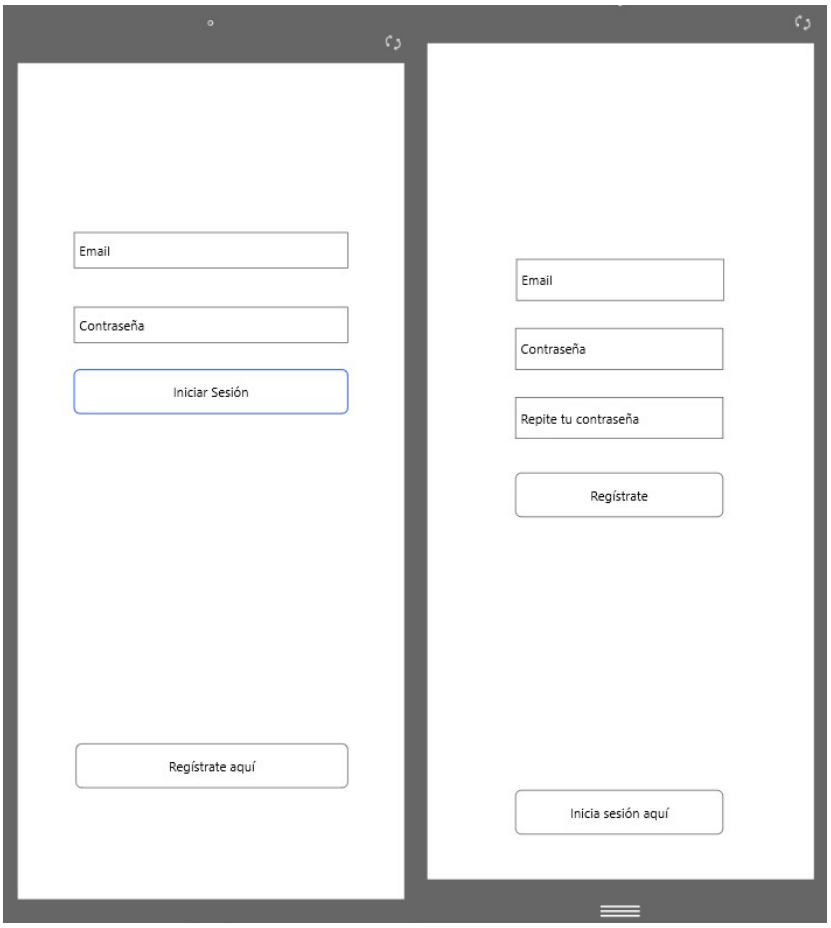

***Figura 11**. Prototipos de inicio de sesión y registro*

3.6.2 Pantallas de navegación

Una vez que hayamos iniciado sesión en la aplicación, se nos presentará en una barra de navegación las principales funcionalidades que podemos usar y de las cuales podemos ver su diseño en la figura 12: ver mis empresas, buscar empresas y mi perfil.

En la pantalla de ver mis empresas, al usuario se le mostrarán tanto las empresas de la base de datos que está siguiendo, como las empresas que él mismo ha creado. Estas empresas saldrán a modo de lista y estarán separas en dos grupos distintos (privadas y públicas) y se podrá cambiar entre ellas usando un botón.

Por otro lado, la pantalla de búsqueda de empresas tendrá una lista de todas las empresas que hay registradas en la aplicación y se podrá filtrar o bien rellenando un campo de texto para buscar por palabras clave o accediendo a una pantalla secundaria que mostrará más opciones de filtrado.

Por último, la pantalla mi perfil tendrá todos los datos que el usuario ha proporcionado a la aplicación. Dependiendo de si este usuario es o no una empresa, tendrá más o

menos campos pudiendo seleccionar, en el caso de ser una empresa, si esta va a ser visible para el resto de los usuarios.

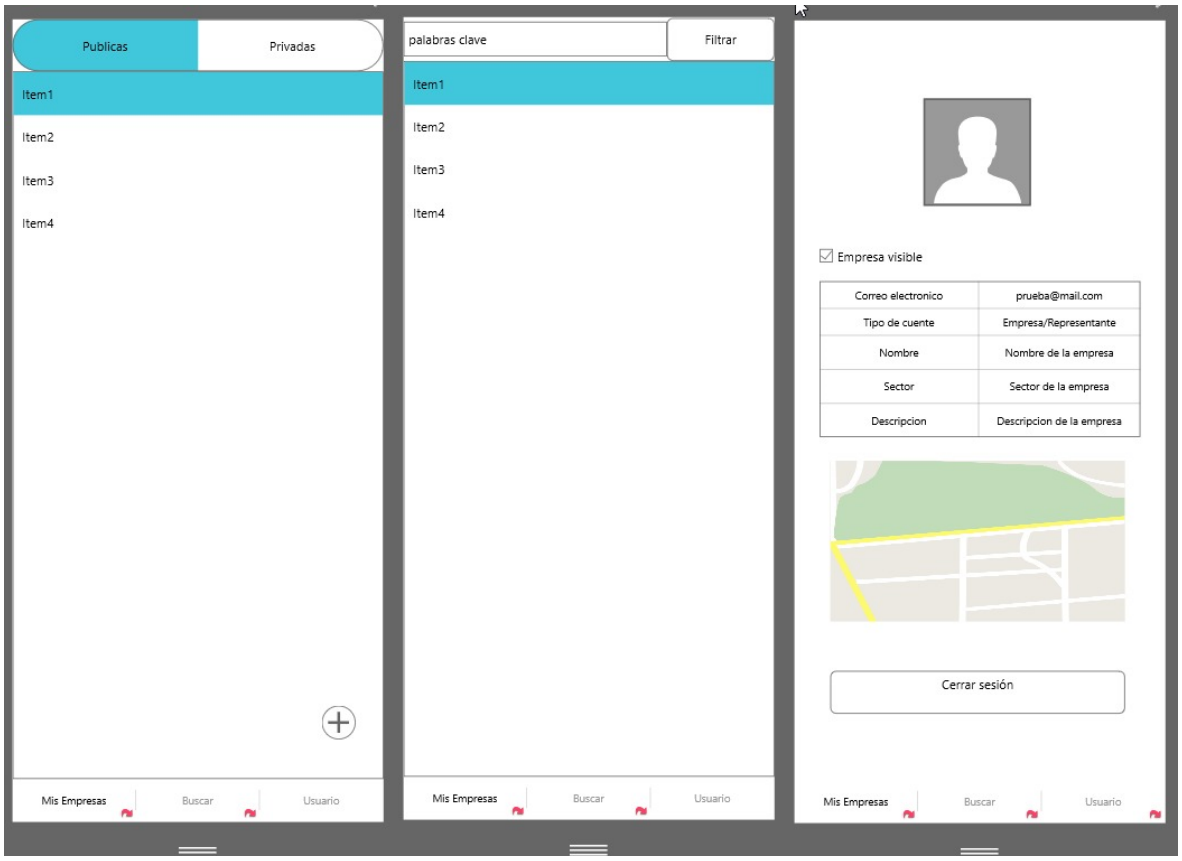

Figura 12. *Prototipos de mis empresas, buscar empresas y mi perfil*

3.6.3 Añadir, modificar o consultar empresa

Con el fin de registrar empresas, ya sea porque algún usuario se está registrando como tal o bien porque algún representante está creando alguna de manera privada, es necesario crear un formulario que le sirva al usuario como plantilla para realizar esta acción.

Figura 13. *Prototipo de añadir empresa*

Como podemos ver en la figura 13, tomaremos tres campos de texto para introducir los atributos que lo requieren, un campo predefinido para elegir el sector de todos los que habrá en la base de datos, un mapa para seleccionar la posición y un botón que nos permitirá tomar o elegir de la galería una foto para adjuntar a la empresa.

En el caso de querer editar una empresa, reutilizaremos el prototipo anterior y simplemente autocompletaremos los campos que ya tengamos en la base de datos.

Por último, para consultar las empresas que tengamos, reutilizaremos una vez más la vista anterior y cambiaremos los campos en los que el usuario puede introducir datos a campos estáticos que no se puedan modificar.

3.6.4 Pantalla de filtrado

Como además de filtrar por nombre las empresas registradas en la base de datos también queremos poder filtrarlas por otros campos, debemos crear una pantalla que agrupe todos estos filtros.

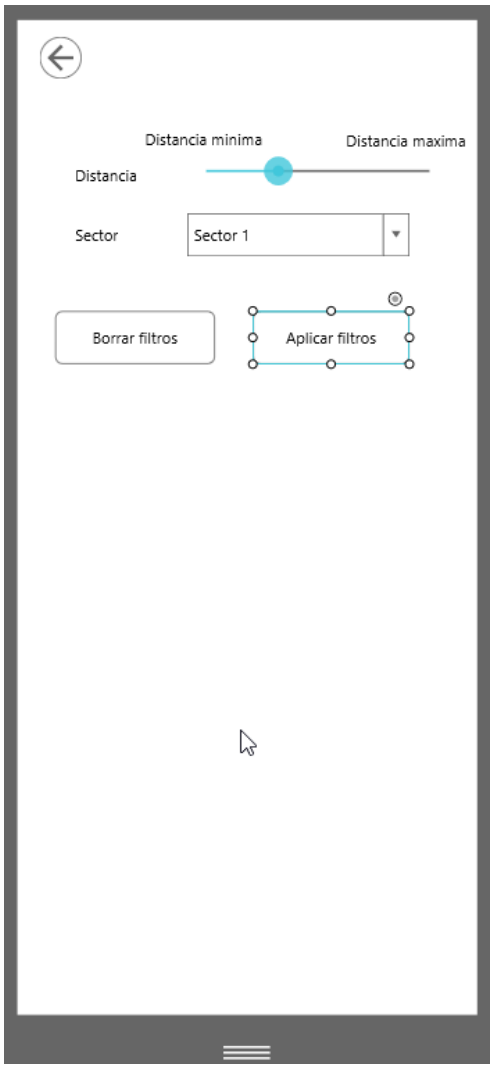

Figura 14. *Prototipo de filtrado*

En la figura 14 se observa cómo queremos que sea nuestra pantalla de búsqueda.

Por un lado, tenemos los campos por los que en un principio queremos que los usuarios puedan restringir las empresas que se les muestran, aunque no descartamos que durante la fase de implementación de la aplicación acaben siendo más campos de los mostrados en el prototipo, pero prevemos que esto no entorpecerá el desarrollo.

Por otro lado, tendríamos tanto un botón que confirmara tanto que se quieren aplicar los filtros seleccionados, como otro botón que borrara cualquier filtrado, tanto actual o ya realizado por el usuario.

3.6.5 Mapa de navegación

Para tener una visión global de cómo será la navegación en la aplicación, hemos dibujado sobre todos los prototipos presentados durante esta sección, una serie de flechas como se puede ver en la figura 15; que ayudan a ver la conexión que habrá entre las diferentes pantallas de la aplicación.

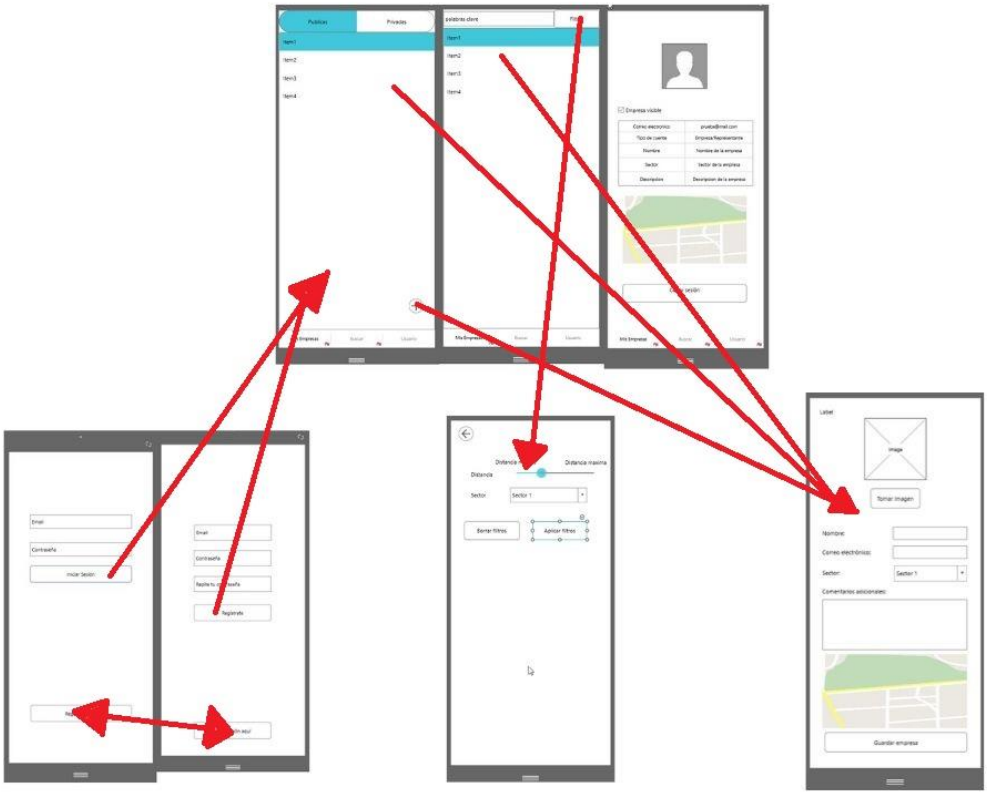

***Figura 15**. Mapa de navegación*

4 Tecnologías

Durante este apartado se dará una breve introducción de las tecnologías que se han utilizado para realizar la aplicación móvil, así como una pequeña introducción a las partes más importantes de estas y los motivos que nos han hecho seleccionarlas frente a otras alternativas presentes en el mercado.

4.1 Angular y Ionic

Una de las primeras cosas que se eligió es el entorno de trabajo que utilizaremos para hacer el *frontend*. Como ya se explicó en el capítulo 2 de este trabajo, queríamos utilizar alguna herramienta que nos permitiese hacer un único desarrollo y que, a la hora de publicar la aplicación, ese desarrollo nos sirviera para desplegarla tanto en Android como en iOS.

De entre todas las alternativas que había en el mercado, tres llamaron nuestra atención*: Ionic, React Native y jQuery Mobile*.

JQuery Mobile es un *framework* que incorpora muchos componentes ya hechos para móvil y nos proporciona una plataforma para desarrollar nuestras aplicaciones mediante HTML y CSS, algo en lo que ya tenemos experiencia trabajando. El problema fundamental que nos hizo descartar este entorno de trabajo es la poca escalabilidad que tiene. Hacer proyectos pequeños resulta tremendamente fácil e intuitivo, pero si quieres tener muchas pantallas, tener un buen control sobre los formularios y disponer de muchas funcionalidades en la misma aplicación, JQuery Mobile nos pareció, después de haberla considerado, una solución poco ideal para el proyecto en mente.

Por otro lado, *React Native* es uno de los entornos más utilizados para desarrollar aplicaciones móviles híbridas. Gran parte de este éxito viene dado por los componentes ya diseñados, así como la habilidad de poder ver, simplemente guardando el proyecto, los cambios que se han realizado. Tampoco nos podemos olvidar de que este *framework* comparte gran parte del flujo de desarrollo con *React*, uno de los entornos más utilizados hoy en día para desarrollar aplicaciones web. Por eso gran parte del éxito de *React Native* es debido también a lo rápido que está creciendo *React* en término de usuarios en el mercado actual como se puede apreciar en la figura 16.

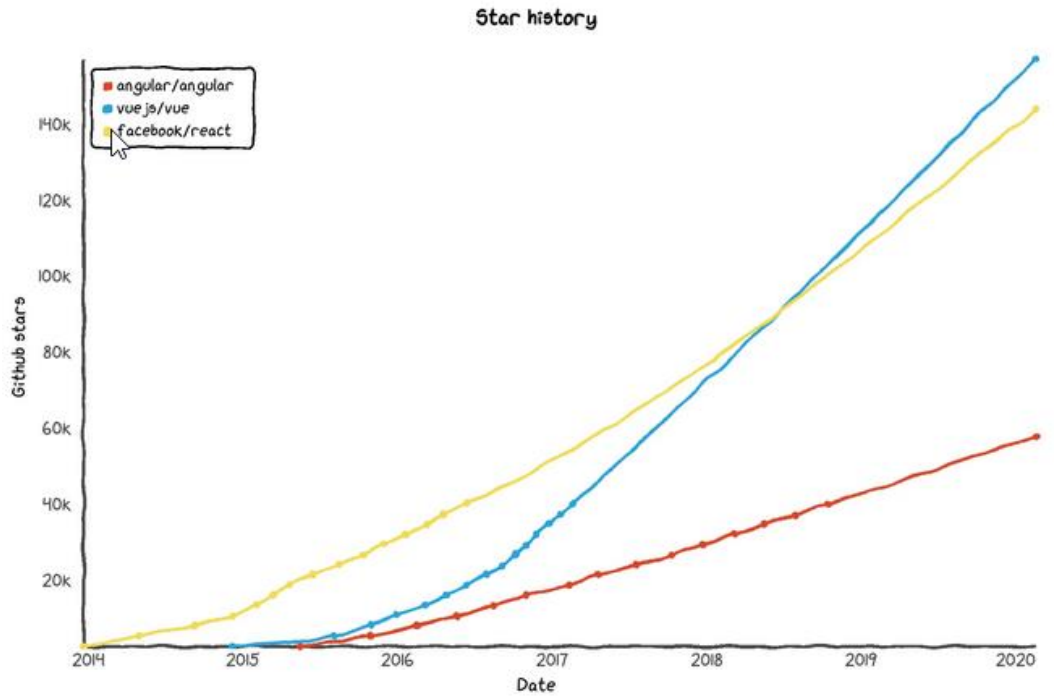

Figura 16. *Usuarios de Angular, Vue y React [11]*

El problema que nos hizo descartar React Native para el desarrollo de este proyecto es que es un *framework* complejo y no disponíamos de experiencia previa trabajando con él, por lo que los tiempos se habrían acortado mucho, teniendo que aprender el entorno durante la realización del trabajo.

El último *framework* sobre el que investigamos es *Ionic*. Este incorpora una gran cantidad de componentes e iconos por lo que no deberíamos preocuparnos demasiado por el aspecto de la aplicación, ya que estaría mantenida por los creadores de este entorno. Por otro lado, siendo este uno de los puntos por los que nos acabaríamos decidiendo por este, nos permite desarrollar eligiendo el *framework* que queramos de entre una lista. En nuestro caso, decidimos integrarlo con *Angular* dado que ya teníamos experiencia trabajando con él; aparte de que, al estar mantenido por una gran empresa como Google, nos aseguramos de que estará bien actualizado.

4.1.1 Estructura de un proyecto de Ionic

Como se ha dicho anteriormente, un proyecto de *Ionic* comparte una estructura casi igual que la de Angular, salvo que este último añade ciertas funcionalidades que hacen más fácil el desarrollo para dispositivos móviles.

Cuando iniciamos un proyecto en Ionic se nos generan una serie de carpetas y archivos a través de los cuales se construye nuestra aplicación, pero toda nuestra atención se centrará en la carpeta *app* ya que es ahí donde se aloja el código fuente de la aplicación.

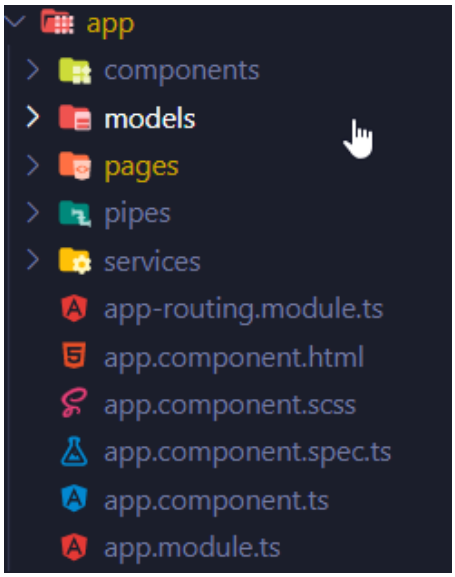

***Figura 17**. Carpeta app*

Como se observa en la figura 17, tenemos una serie de carpetas autogeneradas y archivos en los que estará contenido nuestro proyecto. De estos, las carpetas sirven únicamente a modo de organización y para que sea más intuitivo navegar dentro del proyecto y, en cuanto a los archivos, podríamos destacar los siguientes:

- App-routing. module.ts: En este archivo es donde se encuentran las rutas de nuestra aplicación. Cuando un usuario haga una petición para cambiar la vista en su dispositivo, se acudirá a este archivo para concretar que se ha de renderizar en pantalla dependiendo de lo que haya solicitado el usuario.
- App.component.html: Aquí es donde se encuentra la vista principal de nuestra aplicación, que en nuestro caso redirigirá al módulo de rutas para que renderice el componente adecuado.
- App.module.ts: En este archivo encontramos todas las dependencias de nuestro proyecto. Desde cosas como los conectores de las bases de datos hasta la petición para utilizar la cámara del dispositivo, está todo contenido en este archivo.

4.1.2 Páginas

Una de las partes fundamentales de *Ionic* son las páginas. Una página es un paquete completo que contiene todo lo necesario para representar una vista. Por cada página que creemos en el proyecto tendremos 6 archivos, cada uno de ellos dedicado a una función específica:

1. Archivo de rutas: Aquí tendremos un archivo que nos ayudará a establecer las rutas en la página. Al ya tener en una jerarquía superior otro archivo de rutas global, no haremos uso de este ya que tendremos una visión más global en el superior.
2. Archivo de módulo: En este archivo se inyectan las dependencias que tiene la vista de la que nos estamos ocupando desde el archivo de módulos superior.

3. Archivo HTML: Aquí es donde definiremos que verá el usuario en nuestra página. Al estar trabajando con Ionic tenemos más componentes disponibles de los que tendríamos si utilizáramos una versión normal de HTML
4. Archivo CSS: Este archivo se encarga de aplicar estilos específicos sobre la página.
5. Archivo spec: Gracias a este archivo podemos realizar pruebas unitarias sobre el componente si estamos buscando algún error sobre la página.
6. Archivo ts: Este archivo se encarga de manejar toda la lógica del controlador de la vista. Aquí haremos nuestras llamadas a los servicios, así como organizar la información que más tarde veremos en pantalla.

4.2 Firebase

A la hora de elegir el *backend* de la aplicación se nos presentaban muchas opciones para elegir: Por un lado, podíamos elegir una base de datos SQL más convencional y con la cual tenemos mucha experiencia, pero por otro lado podíamos optar por una base de datos NoSQL que están creciendo de manera muy rápida. En la siguiente figura se puede observar cómo bases de datos SQL (*Oracle, MySQL y Microsoft SQL Server*) parecen estar estancadas en número de proyectos, mientras que bases de datos NoSQL (*MongoDB, Redis y Cassandra*) están creciendo con el paso del tiempo de manera rápida, de acuerdo con la figura 18.

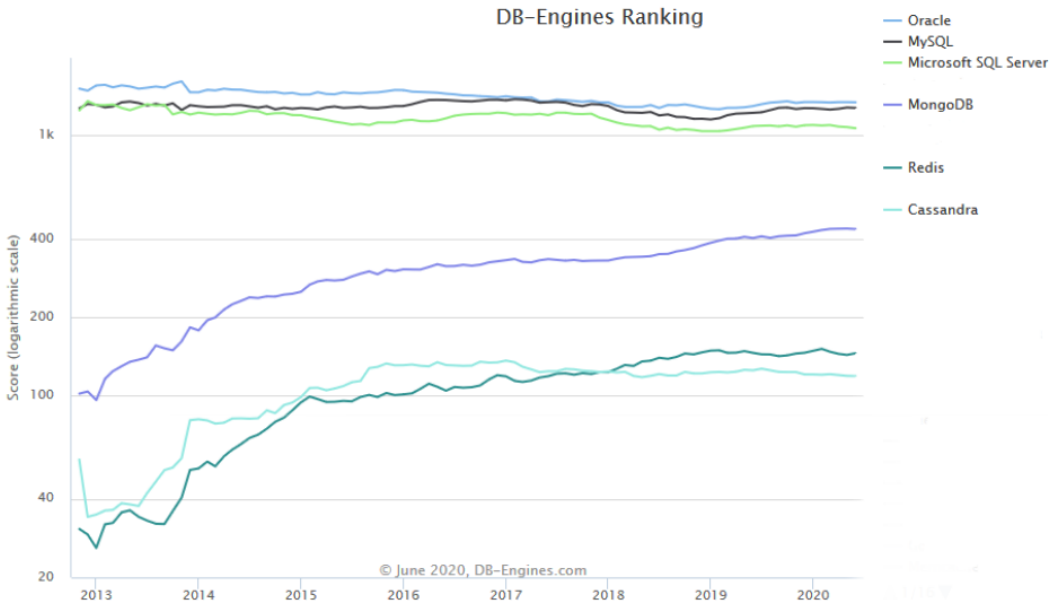

Figura 18. *Comparativa SQL y NoSQL*

Este interés del mercado por las bases de datos NoSQL viene dado por la gran cantidad de ventajas que nos aportan frente a las bases de datos convencionales:

- Esquemas de datos flexibles: Las bases de datos no relacionales tienen la característica de que no son tan rígidas como las relacionales y, por lo tanto, a la

hora de añadir algún dato al esquema o modificar alguno ya existente estas bases de datos nos presentan menos inconvenientes a la hora de hacer dichos cambios.

- Escalabilidad horizontal: Las bases de datos SQL escalan verticalmente, lo que significa que si necesitamos más capacidad, deberemos comprar una máquina más potente. En cambio, las bases de datos NoSQL escalan horizontalmente, lo que quiere decir que si necesitamos más capacidad solo hemos de comprar una nueva máquina y conectarla con la antigua, ahorrando así inversiones muy costosas.
- Mayor velocidad: Pese a que es un punto cuestionable, una consulta en una base de datos relacional suele acabar significando implicar muchas tablas para sacar los datos deseados. En una base de datos no relacional, se suelen tener todos los datos que se necesitan en un mismo documento, haciendo así que solo necesitemos de una consulta para recuperarlos.
- Facilidad de implementación: A diferencia de las bases de datos SQL, los sistemas NoSQL suelen facilitar a los desarrolladores el guardar los datos en la base de datos, ofreciendo sistemas que mapean los datos de tu leguaje de programación a la base de datos.

Por todo esto y, debido a que también nos parece un reto interesante aprender a usar una base de datos NoSQL, se decidió utilizar como *backend* Firebase un sistema de base de datos NoSQL desarrollado por Google e integrado en los servicios de la *Google Cloud Platform*. La decisión de elegir Firebase como sistema de gestión de bases de datos frente a otras alternativas del mercado más populares viene dada por la multitud de servicios que esta nos aporta y que vamos a usar: autenticación, base de datos, almacenamiento de archivos y una integración muy sencilla con *frameworks* como angular.

4.3 Github

Como sistema de control de versiones y repositorio online, decidimos usar *Github*. Aquí es donde se subirán las diferentes versiones del proyecto y donde se mantendrá un control de estas.

El motivo por el que utilizamos Github es por nuestra familiaridad con él, debido a que ya lo hemos utilizado en el pasado y por la posibilidad de crear repositorios privados entre varias personas, para así evitar que puedan acceder personas de poca confianza al código de nuestra aplicación.

4.4 Mockplus Classic

Mediante el uso de esta herramienta de prototipado, hemos diseñado todos los prototipos de las interfaces de la aplicación.

Hemos decidido usar esta herramienta en lugar de otras aplicaciones en el mercado que cumplen la misma función, ya que teníamos experiencia previa usándola, aparte de que nos proporciona todos los elementos que necesitamos para hacer los prototipos y no necesitamos acceder a paquetes externos a la aplicación.

4.5 Visual Paradigm

Visual Paradigm es una herramienta de creación de diagramas a través de muchos estándares diferentes.

Nosotros únicamente usaremos esta aplicación para hacer tanto los casos de uso como el diagrama de clases, pese a que consta de una gran cantidad de tipos de diagramas que podríamos realizar.

Dado que los diagramas estaban previamente hechos en papel y que tan solo los necesitábamos pasar a formato digital, usaremos la versión gratuita que consta de un mes completo con todas las funcionalidades.

4.6 Visual Studio Code

Como editor de código usaremos el procesador de texto de Microsoft, Visual Studio Code.

Los motivos por los que vamos a usar este, frente a soluciones de otras empresas, es por la familiaridad que tenemos con él, hablando de atajos de teclado y conocimiento de la interfaz, así como la gran cantidad de extensiones que se le pueden descargar para facilitar nuestro trabajo a la hora de escribir código.

Otro de los motivos por los que consideramos este editor para realizar el trabajo es la gran integración que tiene, gracias a contribuciones de terceros, con el *framework* de angular, pudiendo autocompletar partes del código que otros editores de texto no podrían hacer en determinadas ocasiones.

5 Desarrollo

La cantidad de código realizada durante el desarrollo de la aplicación ha sido muy extensa y es por eso por lo que no creemos que sería enriquecedor para esta memoria explicar la funcionalidad completa de todo lo que hemos programado. No obstante, sí hay ciertas partes que creemos importantes y que se salen de las funciones básicas que hemos deseado obtener y que, por lo tanto, sí que merecen algo más de atención por nuestra parte.

Durante este capítulo de la memoria queremos profundizar un poco en esas integraciones más importantes que hemos hecho durante este trabajo y tratar de arrojar un poco de luz sobre ellas.

5.1 Conexión con Firebase

Para que la aplicación se pueda comunicar con Firebase, lo primero que deberemos hacer es descargar el módulo de Node.js dedicado para ello. Dentro de la misma carpeta en la que tengamos el proyecto, deberemos introducir la siguiente línea:

```
npm install @angular/fire firebase -save
```

Esto nos instalará todas las funciones que necesitamos para poder interactuar con la base de datos a través de la aplicación.

Lo siguiente que debemos hacer es generar una clave que será mediante la cual accedamos a nuestro almacén de documentos y que, de esta manera, podamos controlar quién puede acceder a la base de datos.

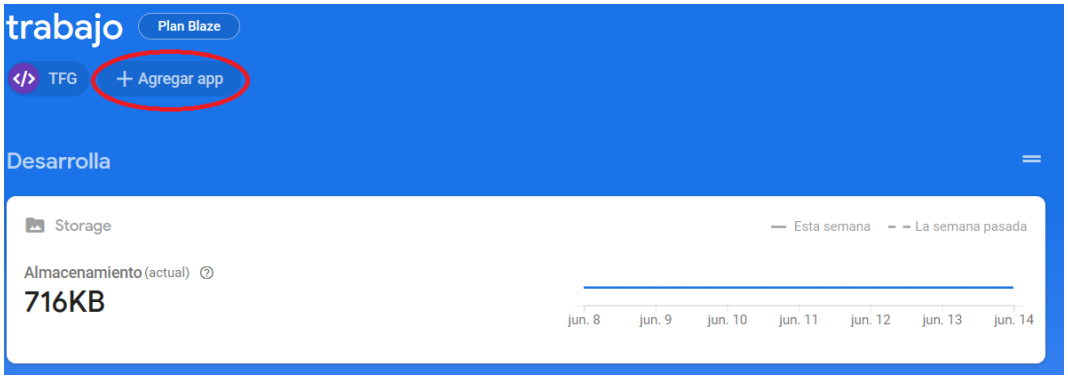

Figura 19. *Consola de Firebase*

Como se muestra en la figura 19, deberemos pulsar sobre el botón “Agregar app” y elegir el tipo de aplicación que queremos desarrollar; en nuestro caso, será una aplicación web, ya que estamos usando Angular como *framework*.

Una vez que hayamos realizado esto, se nos generarán una serie de claves y códigos, mediante los cuales la base de datos se asegurará de que nos estamos comunicando con ella a través de la aplicación. Cuando dispongamos de las claves, tenemos que copiarlas y adjuntarlas en el proyecto como variable del sistema.

Dichas variables las podemos encontrar en un archivo generado por angular llamado *environment.ts* en la carpeta del proyecto *environments*.

```
export const environment = {
  production: false,
  firebase: {
    apiKey: '[redacted]',
    authDomain: 'trabajo-413b7.firebaseapp.com',
    databaseURL: 'https://trabajo-413b7.firebaseio.com',
    projectId: 'trabajo-413b7',
    storageBucket: 'trabajo-413b7.appspot.com',
    messagingSenderId: '[redacted]',
    appId: '[redacted]',
    measurementId: '[redacted]'

  }
};
```

Figura 20. *Variables del sistema*

Nuestras credenciales para acceder a la base de datos tendrán un aspecto parecido al que se puede observar en la figura 20. Hemos omitido algunas de las claves por motivos de seguridad, ya que podrían ser utilizadas por terceros para acceder a todos los documentos. Luego, tenemos que vincular esta variable de entorno al contexto de Firebase. Para ello, en el archivo app.module.ts, tendremos que inicializar la base de datos con las credenciales que hemos introducido.

```
import { Injectable } from '@angular/core';
import { AngularFirestore } from '@angular/fire/firestore';

@Injectable({                                          1
  providedIn: 'root'
})
export class SectoresService {

  constructor(private db: AngularFirestore) { }  2

  getSectores() {
    return this.db.collection('/sectores').valueChanges();
  }
}  3
```

Figura 21. *Acceso a la base de datos*

Por último, y a modo de ejemplo, en la figura 21 podemos ver un ejemplo normal de cómo accedemos a alguna colección de documentos en la base de datos de Firebase.

Lo primero que tenemos que hacer es importar al servicio el módulo de Node.js que nos habíamos descargado (1), lo siguiente que debemos hacer es inyectar dicho módulo en el constructor del servicio para poder usarlo (2) y, finalmente, devolver una promesa de la colección de documentos que queramos consultar (3); en nuestro caso, la colección será sectores que es donde almacenamos los diferentes sectores disponibles.

Como estamos manteniendo un modelo a tres capas, esta promesa se procesará en el componente adecuado y así modularizar lo más posible el código y que un cambio de nombre en cualquier colección no repercuta en el *frontend* de la aplicación.

5.2 Autenticación con Firebase

En la sección anterior hemos comprobado las facilidades que nos da Firebase a la hora de crear una base de datos en la nube y dejarla en estado funcional en muy pocos pasos. Pero, otro de los motivos por el que escogimos Firebase y otra de las ventajas que nos aporta es la posibilidad de manejar la autenticación y registro de nuestros usuarios de una manera muy sencilla y similar a como manejamos la base de datos.

Los primeros pasos para utilizar la autenticación de Firebase son similares a los ya hechos a la hora de conectarnos con la base de datos. Lo primero sería descargarnos el módulo de Node.js correspondiente, el cual ya tenemos descargado, poner nuestras credenciales en las variables de entorno (figura 20) e inicializar la base de datos con dichas variables.

La diferencia entre la autenticación y el acceso a la base de datos radica en el módulo que vamos a usar, así como los métodos. Como podemos ver en la figura 22 vamos a utilizar esta vez Firebase como variable global para tener todos sus métodos.

```
import { Injectable } from '@angular/core';
import * as firebase from 'firebase/app';
```

Figura 22. *Variable global de Firebase*

Una vez importada esta variable global ya podemos usar todas las funciones que tiene Firebase disponibles, en nuestro caso las de autenticación. Como los métodos de registro e inicio de sesión son similares, explicaremos únicamente el de inicio de sesión ya que lo único que cambiaría sería la función de Firebase que utilizaríamos.

```
loginUser(usuario: string, pass: string) {
  firebase.auth().setPersistence(firebase.auth.Auth.Persistence.LOCAL);
  return new Promise<any>((resolve, reject) => {
    firebase.auth().signInWithEmailAndPassword(usuario, pass)
      .then(
        res => {
          this.guardarToken(res.user.uid);
          resolve(res);
        },
        err => reject(err));
  });
}
```

Figura 23. *Método loginUser*

A grandes rasgos, el método presentado en la figura 23 resuelve una promesa a la cual se le mandan, desde la página de Ionic correspondiente, tanto el correo electrónico de la persona que va a iniciar sesión como la contraseña. Si ambos son correctos, se guarda un identificador en el dispositivo del usuario para que no tenga que iniciar sesión cada vez que abre la aplicación y, si no son correctos mandará un mensaje de error. Una parte importante a la hora de hacer la experiencia fluida es el fragmento de código: "*firebase.auth.Auth.Persistence.LOCAL*" dado que gracias a él se establece que ese usuario será persistido en el dispositivo del usuario y, por lo tanto, se guardará la sesión.

Por último, y para completar el proceso de autenticación, hemos de proteger las partes de la aplicación que no queremos que sean accesibles si el usuario no se ha identificado; es decir, todas las pantallas de la aplicación que muestren contenido extraído de la base de datos.

Para la primera, Angular nos proporciona un servicio denominado *guard* que configuramos en nuestro caso resolviendo si el usuario está identificado o no, y que colocamos como indica la figura 24 en nuestro módulo de rutas, en aquellas vistas que requieran de una identificación previa.

```
path: 'nuevo-usuario',
canActivateChild: [AuthGuard],
loadChildren: () => import('./pages/nuevo-usuario/nuevo-usuario.module').then(m => m.NuevoUsuarioPageModule)
```

Figura 24. *Uso del guard*

Por otro lado, para proteger la base de datos de usuarios que no se hayan identificado, Firebase nos proporciona una opción en su consola mediante la cual dejar acceder a la base de datos solamente a aquellos usuarios que se hayan identificado. De esta manera, en el panel de control de Firebase, en el lugar donde se encuentra nuestra base de datos deberemos cambiar el código que hay al que se muestra en la figura 25 para que solo aquellos usuarios que estén identificados puedan leer o escribir en la base de datos.

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if request.auth.uid != null;
    }
  }
}
```

Figura 25. *Reglas de la base de datos*

5.3 Almacenamiento de Archivos en Firebase

Una de las ideas que surgió durante el desarrollo de este trabajo fue la posibilidad de agregar imágenes a las empresas que se registraran. Gracias a estar trabajando con Firebase tenemos una manera fácil de subir archivos y recuperarlos usando funciones similares a las que hemos visto cuando trabajábamos con la base de datos.

Lo primero que tenemos que hacer, y como ya hemos visto en casos anteriores cada vez que usamos Firebase, es tener correctamente instalado tanto el módulo como las credenciales que se nos han generado.

Una vez hecho esto podremos acceder a un objeto llamado *storage*, que contiene todas las funciones necesarias para subir o recuperar archivos.

```
guardarImagen(url: string) {
  const pictures = storage().ref(url);
  pictures.putString(this.img, 'data_url');
}
```

Figura 26. *Uso del storage de Firebase*

Como se puede apreciar en la figura 26, subir cualquier archivo, en nuestro caso una imagen, es tan fácil como pasar como argumento la localización en nuestro dispositivo del archivo a subir y subirlo como una cadena de caracteres.

Una de las cosas que pueden dar problemas es la duplicidad en los nombres subidos. Ya que como son imágenes tomadas por los usuarios no podemos controlar que alguna tenga el mismo nombre. La solución que se tomó para evitar estos problemas fue renombrar la imagen antes de subirla y ponerle un nombre de la hora en la que se ha subido. De esta manera, como estamos utilizando un número hasta los milisegundos, es muy improbable que se haya repetido. No obstante, y si la aplicación tuviera multitud de usuarios conectados al mismo tiempo, se podría dar que con este método se repitiera alguno de los nombres que recordamos han de ser únicos. La manera rápida de

solucionar esto es usar el propio Firebase para que nos genere claves únicas y así olvidarnos de posibles errores por culpa de este aspecto.

Aun así, y como la aplicación se va a usar en entornos controlados y no habrá demasiados usuarios simultáneos, se ha decidido utilizar el método de la fecha para asignar los nombres de las imágenes ya que así nos ahorramos una llamada a la base de datos innecesaria en nuestro caso.

5.4 Integración con Google Maps

Una de las cosas más importantes que tiene la aplicación es la posibilidad de almacenar la localización de las empresas que se registren. Para ello, después de analizar varias posibilidades para realizar esto, se decidió usar Google Maps por su gran presencia en todos los dispositivos Android, el hecho de que al estar trabajando con Firebase podríamos vincularlos, usar la misma cuenta simultáneamente para obtener estadísticas y por la facilidad de implementación y gran cantidad de recursos que Google da a los desarrolladores.

Lo primero que tenemos que hacer como en cualquier integración que hagamos desarrollando en Angular es incluir el módulo de Node.js correspondiente. En este caso:

```
npm install @angular/googlemaps
```

Una vez descargado esto, en la cabecera de nuestro proyecto deberemos incluir una URL que nos facilitara la consola de Google Maps para identificarnos con un número de identificación personal como se puede ver en la figura 27, en la cual, como ya hicimos con las de Firebase, hemos omitido parte de él en la imagen.

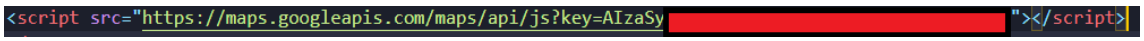

***Figura 27**. Credenciales de Google Maps*

Una vez ya estemos conectados con Google Maps tenemos que mostrar el mapa por pantalla. Para esto tenemos que ir al archivo HTML de la vista en la que queramos colocar el mapa y en un elemento *div* asignarle un identificador que usaremos en el controlador de la vista para cargar el mapa.

Con respecto al controlador de la vista, deberemos ejecutar una función al cargar la misma que se encargue de pasarle al mapa ciertos datos como las coordenadas, el nivel de enfoque o el lugar de la vista donde se tiene que colocar el mapa.

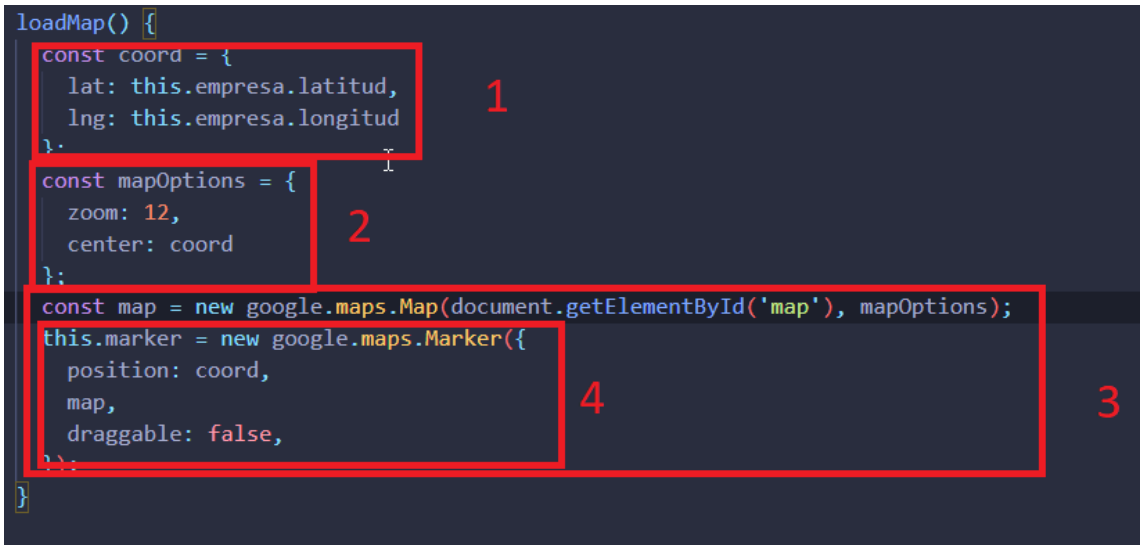

Figura 28. *Uso de Google Maps*

Como podemos observar en la figura 28, la función que carga el mapa consta de diferentes partes:

- La primera (1) es localizar dónde tendrá el centro el mapa. En nuestro caso será en las coordenadas que tengamos en la empresa de base de datos o, si fuera en el registro de la empresa, la posición que nos marque la antena GPS del dispositivo.
- La segunda (2) es establecer la altura a la que se situará el mapa, así como indicarle que las coordenadas introducidas en el punto uno son el centro del mapa.
- La tercera (3) es la de cargar el propio mapa. Como se ve en la figura, lo hacemos pasando a la función que lo crea tanto las opciones, como el identificador que tiene en la vista el elemento del mapa.
- Por último, como queremos que se visualice dónde está la empresa, debemos poner un marcador (4) con las coordenadas que hemos recuperado de la base de datos.

5.5 Integración con una API Rest

Uno de los campos que tienen que rellenar las empresas a la hora de registrarse es el del sector al que pertenecen. Este campo estará predefinido y se sacará de una colección de documentos de la base de datos. No obstante, y para que estos sectores estén actualizados, decidimos integrarnos con una API Rest que nos ofreciera los sectores de producción. Pese a que hay muchas API's Rest a elegir que nos ofrecen dicha función, decidimos integrarnos con la que pone a nuestra disposición el Ayuntamiento de Valencia [14]. Esta API nos proporcionará un fichero de tipo JSON con los sectores de producción con los cuales actualizaremos nuestra base de datos cuando sea necesario.

Para hacer la actualización deberíamos consultar periódicamente si los sectores han cambiado y, de haberlo hecho, añadir los nuevos que surjan. Firebase pone a nuestra disposición un servicio para establecer funciones programadas que sería lo ideal dadas nuestras necesidades; no obstante, tuvimos que buscar otra alternativa ya que este servicio solo es accesible mediante usuarios de pago y nosotros estamos usando la versión gratuita.

Por ello decidimos utilizar Node.js para crear un proceso que periódicamente consultara la API del Ayuntamiento de Valencia y actualizara los datos que hay en nuestra base de datos de manera acorde.

Lo primero que tendremos que hacer es utilizar la interfaz http de Node.js y descargar el documento JSON de la API del Ayuntamiento de Valencia. Como se puede observar en la figura 29, solo tenemos que conectarnos mediante la URL necesaria y, una vez tengamos los datos, llamar a la función que actualizará el contenido de la base de datos.

```
var URL = 'http://apigobiernoabiertocatalog.valencia.es/api/3/action/group_list';

function recibirDatos() {
    var req = http.get(URL, function(res) {
        var data = '',
            json_data;

        res.on('data', function(stream) {
            data += stream;
        });
        res.on('end', function() {
            json_data = JSON.parse(data);
            subirDatos(json_data.result);
        });
    });
    req.on('error', function(e) {
        console.log(e.message);
    });
}
```

***Figura 29**. Consultar API ayuntamiento de Valencia*

Como hemos dicho, una vez hemos descargado y consultado los datos, debemos actualizarlos no sin antes comprobar que no introducimos ninguno repetido. Para esto debemos utilizar el SDK que pone Firebase a nuestra disposición, consultar la colección correspondiente a los sectores y luego introducir los que no se repitan. Como podemos observar en la figura 30, el proceso para actualizar es muy similar al que utilizábamos cuando lo hacíamos desde Ionic.

```
function subirDatos(datos) {

    var serviceAccount = require("C:/Users/aVice/Downloads/trabajo-413b7-firebase-adminsdk-mr9de-8835ae6c47.json");
    if (!admin.apps.length) {
        admin.initializeApp({
            credential: admin.credential.cert(serviceAccount),
            databaseURL: "https://trabajo-413b7.firebaseio.com"
        });
    }

    var db = admin.firestore();
    db.collection('sectores').get().then(
        function(data) {
            var sectores = [];
            data.forEach(function(sector) {
                sectores.push(sector.data().nombre);
            });


            for (var dato of datos) {
                if (!sectores.includes(dato)) {
                    db.collection('sectores').add({ nombre: dato });
                    console.log(dato);
                }
            }

        }
    );
```

Figura 30. *Función subirDatos*

6 Resultados

Durante este capítulo se procederá a una explicación de los resultados obtenidos con la aplicación.

Primero compararemos algunas capturas hechas con la aplicación con los prototipos a partir de las cuales fueron diseñadas y así explicar ciertos cambios que se hicieron sobre los prototipos y los motivos que impulsaron estos cambios.

Luego, haremos un breve repaso sobre un formulario enviado a un grupo controlado de personas en el cual darán sus opiniones sobre algunas de las funciones que prevemos serán las más usadas por parte de los usuarios.

6.1 Aplicación frente a los prototipos

Durante el desarrollo de la aplicación, ciertas cosas que se establecieron al hacer los prototipos no se han podido mantener. Ya sea por motivos estéticos y funcionales o porque no se pensó en el momento la implicación que tendría sobre la función de la aplicación el diseño de ciertas vistas. A continuación, explicaremos el motivo de los cambios y lo compararemos con los prototipos previos.

6.1.1 Lista de empresas

En un principio, al diseñar la interfaz, se decidió presentar los listados de empresas como una lista de nombres de empresas y sobre las cuales se podría pulsar sobre los elementos y así acceder a un detalle más amplio de la empresa elegida. Al trabajar con Ionic nos dimos cuenta de que algunos de los elementos que nos ofrece ejercerían la misma función que esta lista y, a parte, podríamos mostrar más información y de esta manera aportar mucha más claridad al listado.

Otra cosa que no se planificó durante el prototipado fue la necesidad de establecer los botones tanto de borrar como de editar las empresas. Por suerte, Ionic nos ofrece la posibilidad de implementar botones ocultos que se muestran cuando deslizamos ciertos elementos a los lados. De esta manera, conseguimos tener los botones necesarios para realizar todas las acciones, sin necesidad de modificar el aspecto que tendría la pantalla que anteriormente planificamos.

Podemos observar los cambios de esta pantalla en la figura 31, como la lista de empresas que ahora se nos presenta a modo de tarjetas y que estas se pueden deslizar hacia los lados para revelar el botón, en este caso, de borrado.

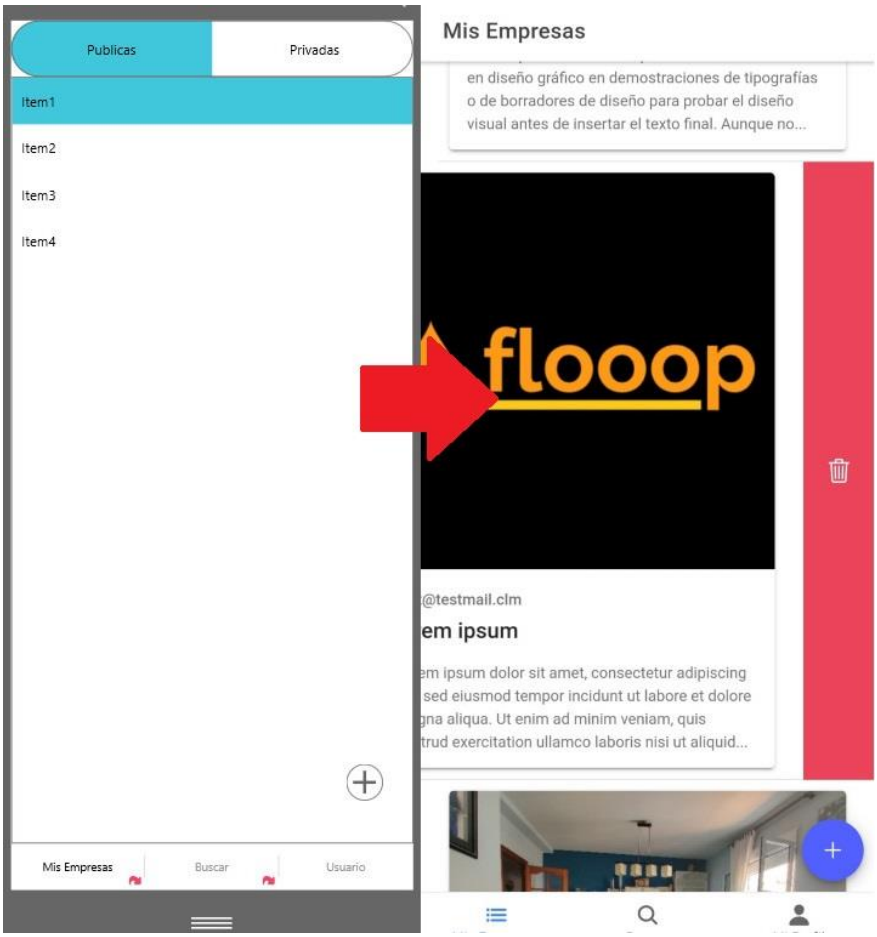

Figura 31. *Prototipo lista de empresas (izquierda) / Lista de empresas en la aplicación (derecha)*

6.1.2 Búsqueda de empresas

En el caso de la pantalla de búsqueda de empresas, el cambio realizado fue tanto para dar una funcionalidad extra, como para mostrar más claridad al usuario.

Dado que tenemos varios filtros que podemos aplicar, durante el prototipado no se pensó en la idea de poder quitar filtros, uno a uno, para dar resultados diferentes sin necesidad de acceder a la pantalla en la que se aplican.

En el caso de esta pantalla se añadió una pequeña sección que mostraría los filtros que se están aplicando a la búsqueda y la posibilidad de eliminarlos uno a uno desde esa misma pantalla, como podemos observar en la figura 32.

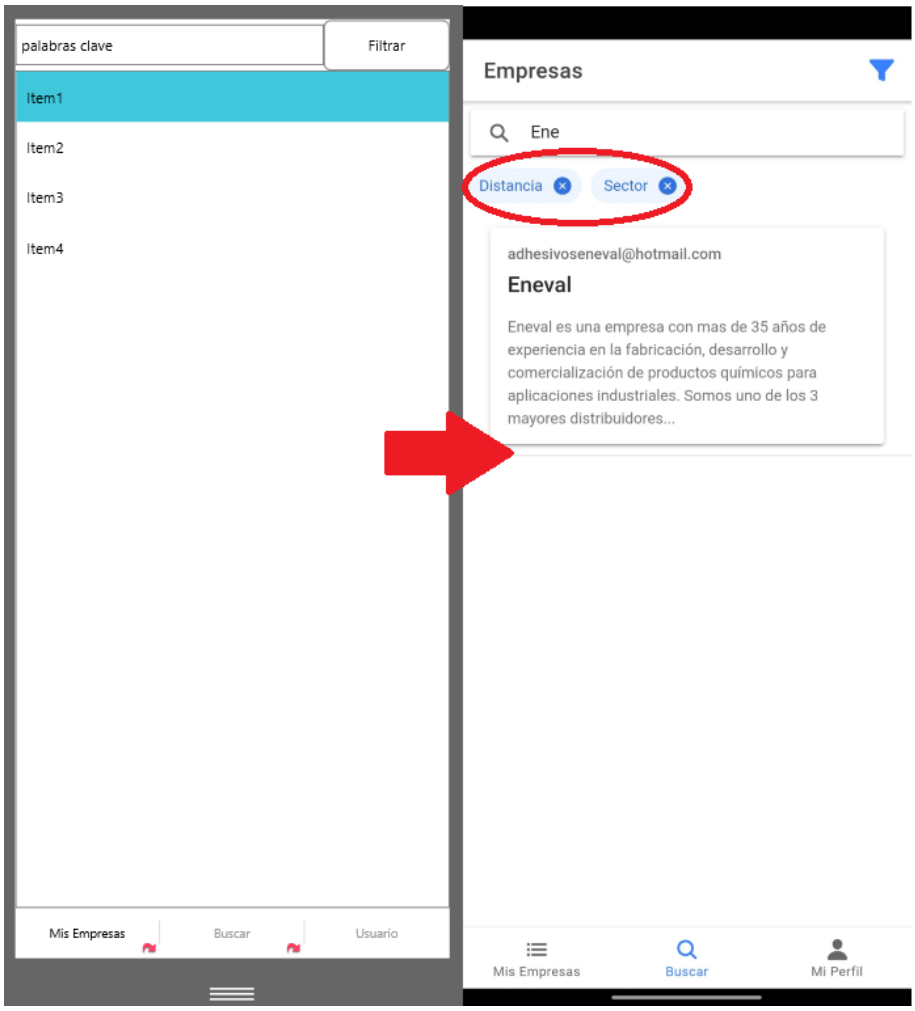

***Figura 32**. Prototipo buscar empresas (izquierda)/Aplicación buscar empresas (derecha)*

6.1.3 Pantalla de registro

En este caso, la pantalla original de registro no se modificó. Pero en el momento del análisis no se reparó en la problemática que supondría diferenciar a los dos tipos de usuarios base que tendríamos en la aplicación.

Es por ese motivo que, dada la necesidad de diferenciar a estos dos usuarios, necesitaríamos una pantalla que, después de registrarse el usuario en la aplicación, nos informara de cuál de los dos tipos de usuario se trataba: empresa o representante.

Para ello diseñamos una pantalla que preguntaba al usuario si era empresa o no. Y, a continuación, en caso de que la respuesta fuera afirmativa, debería de rellenar un formulario para ofrecer los datos de su empresa y que de esta manera empezara a salir junto con el resto de las empresas registradas.

Podemos ver en la captura de pantalla de la figura 33 la nueva vista resaltando la pregunta hecha a los usuarios que, en caso de ser negativa, haría desaparecer el formulario de la parte inferior.

***Figura 33**. Formulario de registro de empresas*

6.2 Resultados del formulario de aceptación

Para comprobar que los usuarios de la aplicación no tenían problemas en utilizar las funcionalidades que se consideraron fundamentales en el momento del análisis de requisitos, se decidió enviar la aplicación a un número reducido de personas junto con un formulario con ciertas tareas a realizar en la aplicación, para que valoraran la dificultad que habían tenido en cumplir dichas tareas.

Para esa valoración utilizamos una escala tipo Likert [15], no obstante, antes de cada apartado, haremos una pequeña explicación de las opciones que les presentaremos a los encuestados y lo que significarán en cuanto a los resultados.

6.2.1 Perfil de los usuarios encuestados

Primeramente, necesitamos saber el perfil de los usuarios que estaban utilizando la aplicación ya que no serviría de nada tener los resultados de un grupo que mayoritariamente cuenta con gente con mucha facilidad para utilizar aplicaciones móviles, al igual que tampoco nos serviría un grupo que tuviera las características inversas.

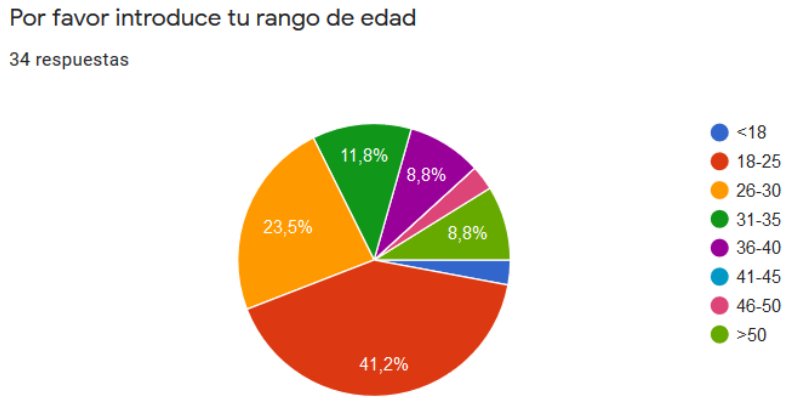

***Figura 34**. Rango de edades de los usuarios encuestados*

Como podemos ver en la figura 34, pese a que gran parte de los usuarios encuestados están por debajo de los 30 años, tenemos suficientes datos del resto de grupos de edad como para que no haya resultados demasiado anómalos.

No obstante, saber el rango de edad no nos dice si los usuarios tienen facilidad para usar nuevas tecnologías. Para eso realizamos otra entrada en la encuesta preguntando la opinión que tienen los usuarios de su destreza con las nuevas tecnologías y que lo valoraran del uno al cinco, siendo este último una opinión muy buena. Los resultados (figura 35) indican que, pese a que tenemos mayor cantidad de usuarios en el número cuatro, están bastante bien distribuidos entre el resto de los números y las conclusiones de la encuesta serán útiles.

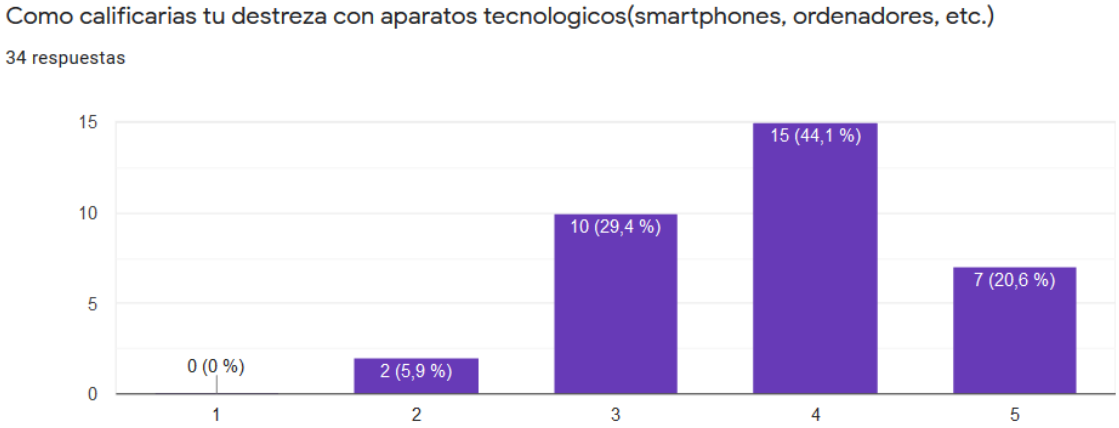

***Figura 35.** Destreza de los usuarios*

Por último, y dado que es la primera vez que los usuarios estaban utilizando la aplicación, también nos interesaba saber la velocidad con la que se suelen adaptar a una aplicación totalmente nueva, para así saber si algunos resultados podrían ser alterados con experiencia mediante el uso de esta.

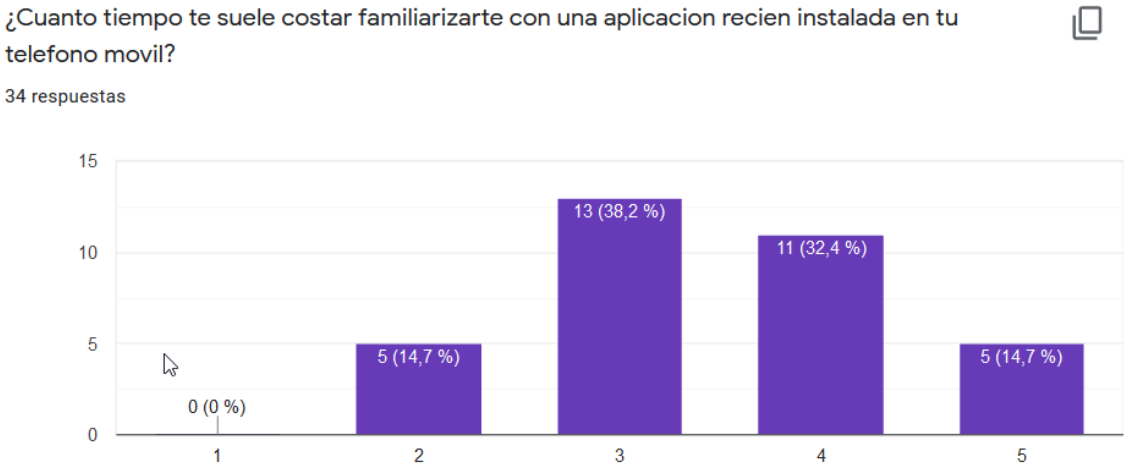

Figura 36. *Tiempo de familiarización de los usuarios*

En la figura 36 vemos cómo la mayor parte de los usuarios dieron una respuesta neutral, con una ligera inclinación hacia que no les cuesta familiarizarse con nuevas aplicaciones. Dado los rangos de edades que estamos manejando, esto no parece muy sorprendent, aunque se tendrá en cuenta al analizar los resultados de las tareas.

6.2.2 Resultados de las tareas

Durante esta sección se preguntó a los usuarios la dificultad de realizar ciertas acciones en la aplicación. Los usuarios encuestados pudieron valorar esta dificultad del uno al cinco, siendo uno muy difícil y cinco muy fácil.

Lo primero que se le pide a los encuestados es registrarse en la aplicación como una empresa. Los resultados fueron positivos ya que todos los usuarios consideraron que esta acción era o fácil o muy fácil (figura 37).

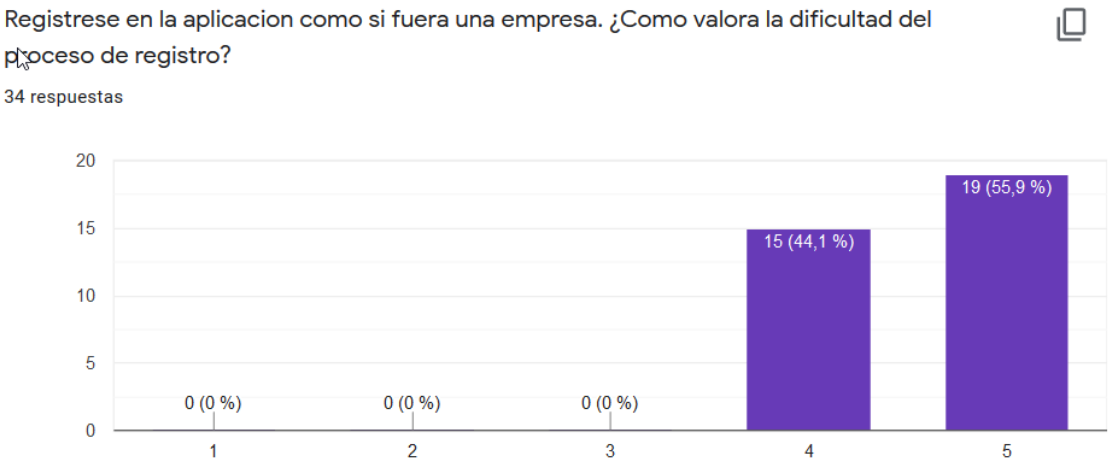

***Figura 37**. Resultados del registro*

Después, se les pidió a los usuarios que crearan una empresa privada en la aplicación. Los resultados (figura 38), al igual que con el registro, fueron muy positivos, situándose la mayor parte de las respuestas en el cuatro y sin ninguna respuesta en los números más bajos.

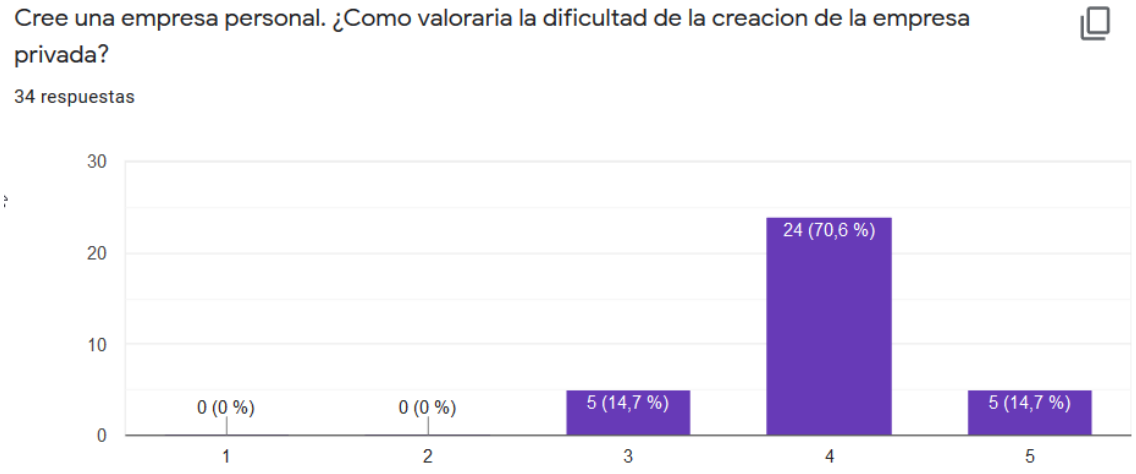

***Figura 38.** Resultados de crear una empresa personal*

En una de las acciones que peor resultados ha tenido, se les pidió a los usuarios que editaran la empresa que habían creado. La mayoría de los encuestados pusieron una nota de dos a esta acción, aunque los resultados son bastante dispersos. Estos resultados mostrados en la figura 39, pueden deberse a que el botón para realizar esa acción está oculto y solo se revela haciendo un gesto de arrastre en la pantalla, lo que puede confundir a los usuarios con menos experiencia.

***Figura 39**. Resultados de la edición de empresas*

La siguiente tarea que se les pidió a los usuarios es la de borrar la empresa que habían creado. Esta vez, y pese a que el gesto es el mismo para revelar el botón de borrar que para revelar el de editar, los usuarios tuvieron mucha más facilidad para realizar esta tarea como se puede ver en la figura 40.

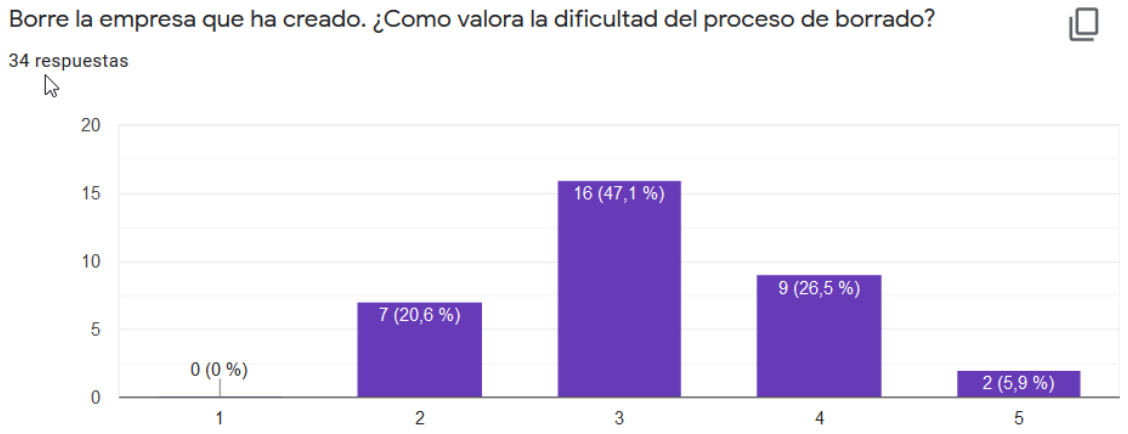

***Figura 40**. Resultados de borrar empresa*

A continuación, se les pidió a los usuarios que realizaran un filtrado sobre las empresas registradas. Los resultados (figura 41) aquí sí que fueron muy satisfactorios teniendo notas de entre cuatro y cinco, por lo que consideramos que los usuarios no tuvieron ninguna dificultad en realizar este paso.

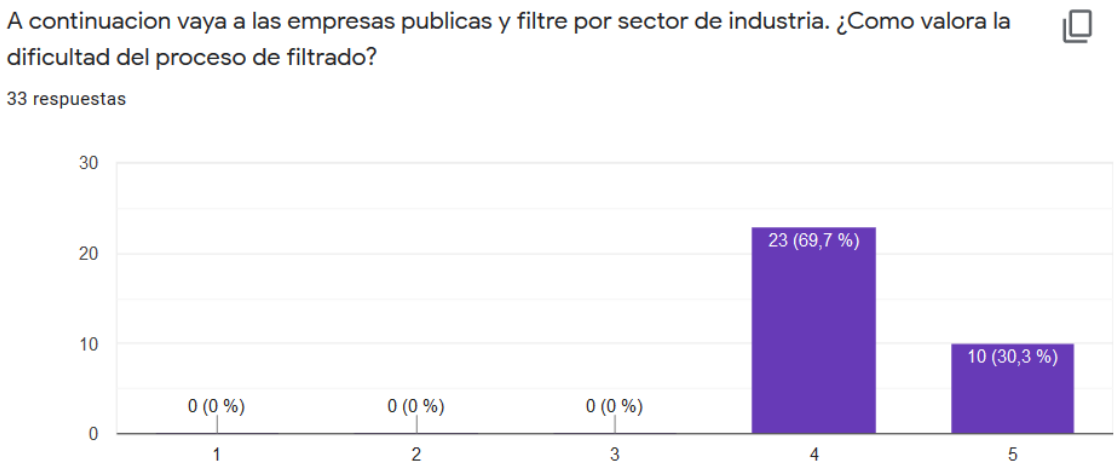

Figura 41. *Resultados del filtrado*

Seguidamente, los encuestados realizaron la tarea de seguir a una de las empresas que habían filtrado. Como el método para hacer aparecer el botón para completar la acción es el mismo que al editar o borrar empresas, pensábamos que los resultados serían similares. No obstante, la mayoría de encuestados dejaron una opinión neutra sobre esta acción como se puede observar en la figura 42. Esto puede indicar que, con el uso de la aplicación, los usuarios van aprendiendo cómo funcionan ciertas características.

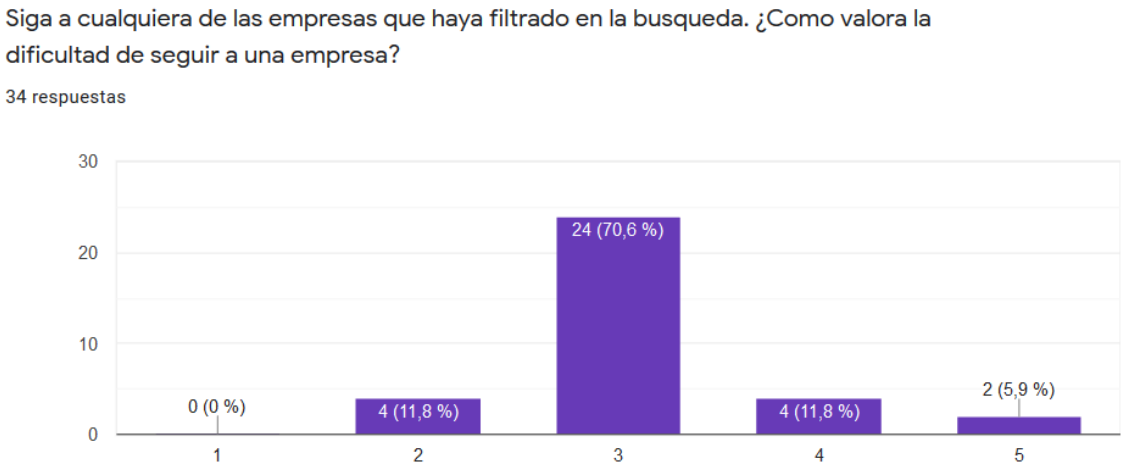

Figura 42*. Resultados de seguir empresa*

Por último, se les pidió a los usuarios que sacaran por pantalla los detalles de cualquier empresa. Aquí los resultados, como podemos apreciar en la figura 43, son buenos ya que hay una gran concentración de respuestas en los números más altos de la estadística.

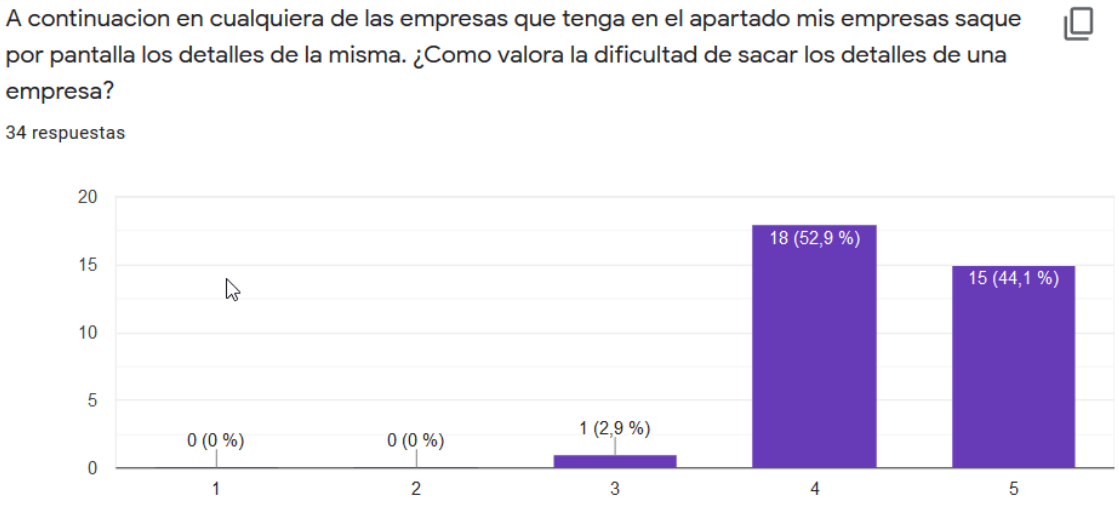

***Figura 43**. Resultados al sacar el detalle de una empresa*

6.2.3 Conclusiones de la encuesta

Como conclusiones de las encuestas realizadas podemos decir que la usabilidad de la aplicación es bastante buena.

Gran parte de las acciones que se les han pedido realizar a los usuarios han recibido notas altas por parte de los encuestados.

Pese a esto, hay que decir que los usuarios han tenido cierta dificultad para realizar una serie de acciones similares, pero que conforme se iban familiarizando con la aplicación, estas dificultades iban desapareciendo. Para solucionar esto se podría hacer un breve tutorial en el momento del registro para que los usuarios encuentren de manera rápida dónde están todos los controles de la aplicación sin necesidad de descubrirlo por ellos mismos.

Por último, y para terminar con las encuestas, se les pidió a los encuestados si recomendarán la aplicación a más personas y el 100% dejó una respuesta afirmativa como vemos en la figura 44.

Figura 44. *Recomendación de los encuestados*

7 Conclusiones

Terminando con la memoria de este trabajo de final de grado, queda exponer las conclusiones que hemos sacado durante el desarrollo de este y recordar los objetivos fundamentales que se plantearon durante el primer capítulo de esta memoria.

Comenzando con los retos que nos propusimos alcanzar en el capítulo introductorio, podemos decir que los hemos cumplido. Hemos conseguido desarrollar desde cero una aplicación para teléfonos móviles que funciona a modo de red social tanto para responsables de compras como para empresas. Las empresas pueden registrarse en nuestra aplicación y disponer de una plataforma que les sirve para darse a conocer, mientras que los comerciales pueden tanto seguir a estas empresas mencionadas como crear empresas privadas para que, de esta manera, puedan tener un lugar donde organizar sus clientes.

Para conseguir estas funcionalidades, se ha optado por usar una base de datos en tiempo real como es Firebase y, de esta manera, centralizar los datos de todos los usuarios en una sola plataforma, así como investigar las virtudes de estos servicios que se ponen a nuestra disposición.

Gran parte de los aspectos enriquecedores que ha tenido este trabajo vienen dados por la investigación que se ha tenido que hacer tanto del sector para el que estábamos desarrollando la aplicación, como por las tecnologías que hemos usado para el cumplimiento de este trabajo.

Hemos aprendido que hay ciertos sectores, como para el que hemos desarrollado este trabajo, que tienen mucho que avanzar en temas de modernización y esto abre las puertas a futuras aplicaciones para conseguir una mayor eficiencia dentro de estos.

Por otro lado, también hemos investigado sobre las tecnologías más usadas por los desarrolladores para la implementación de aplicaciones móviles, así como de sistemas de gestión de bases de datos que crecen en número de usuarios anualmente frente a aquellos más usados del mercado.

Cabe destacar también la perspectiva que nos ha aportado este trabajo de cara a hacer futuros desarrollos. Durante el grado, mayoritariamente los desarrollos que se hacían eran sobre un pequeño problema a resolver, que tenía una solución relativamente clara y lejos quedaban los problemas que pueden surgir en el día a día y que estén fuera de un entorno controlado. Al haber tenido que desarrollar una aplicación completamente desde cero, nos hemos encontrado con problemas que pueden surgir normalmente, y que no podríamos de otra manera habernos encontrado con ellos. Uno de estos problemas, que creemos que nos da una visión clara de a lo que nos estamos refiriendo, es el hecho de haber tenido que usar una aproximación diferente a la que nos habíamos planteado en un principio, ya que algunas de las funciones que íbamos a usar eran de pago y tuvimos que buscar soluciones diferentes que fueran gratuitas.

Como conclusión, podemos decir que, pese a las dificultades que han surgido durante los meses que hemos estado realizando el trabajo, este nos ha aportado conocimientos

que seguro podremos utilizar en el futuro y también nuevas perspectivas en lo que se refiere al sector en el que trabajaremos.

7.1 Mejoras y ampliaciones

Durante la realización del trabajo, fueron surgiendo distintas ideas que se podrían implementar en la aplicación y que, por temas de tiempo y recursos, no se pudieron llevar a cabo. A continuación, vamos a compartir algunas de estas ideas que puedan ampliar el trabajo realizado en un futuro:

- Chat para usuarios: Una de las ideas que surgieron investigando todas las posibilidades que tiene Firebase, es la posibilidad de integrar un chat en la aplicación. Sería muy interesante que, tanto los comerciales como las empresas, pudieran hablar entre ellos sin necesidad de utilizar otros medios como podría ser el correo electrónico.
- Modelo de pago: Actualmente, la aplicación no tiene una manera en la que nos pueda generar beneficios. Salvando la publicidad que pudiéramos poner en ella, se nos ocurrió implementar un sistema de pago para las empresas y que les permitiera salir en partes más altas de la lista si tenían esta opción activada. Creemos que es una buena manera tener este modelo de pago sin interferir en las funcionalidades principales de la aplicación.
- Modo de seguimiento: Nos gustaría haber implementado un modo de seguimiento en la aplicación mediante el cual, los comerciales podrían seguir un sector y en el momento en el que alguna empresa perteneciente a ese se registrarse, la aplicación informara al comercial de que hay una nueva empresa disponible en el sector.

8 Bibliografía

[1] Internet traffic, 19 de mayo de 2020, https://en.wikipedia.org/wiki/Internet_traffic , Fecha de consulta 20/05/2020

[2] M. Valero, Las ventas textiles se estancan con menos tiendas físicas frente al empuje de internet,20 de junio de 2018, https://www.elconfidencial.com/empresas/2018-06-20/textil-moda-ventas-ropa-e-commerce-centros-comerciales_1580915/, Fecha de consulta 29/05/2020

[3] LinkedIn, https://linkedin.com/ , Fecha de consulta 26/05/2020

[4] ClientiApp, https://www.clientiapp.com/es, Fecha de consulta 26/05/2020

[5] Craig Chapple, European Mobile App Consumer Spending Grew 19% in 2019 to More Than $11 Billion, 29 de enero de 2019, https://sensortower.com/blog/europe-app-revenue-and-downloads-2019, Fecha de consulta 22/05/2020

[6] Alfonso Casas, iPhone vs Android: cuota de mercado, 25 de febrero de 2019, https://www.pcworld.es/articulos/smartphones/iphone-vs-android-cuota-de-mercado-3692825/ , Fecha de consulta 22/05/2020

[7] Casos de uso, 24 de marzo de 2020, https://es.wikipedia.org/wiki/Caso_de_uso , Fecha de consulta 30/05/2020

[8] Tutorial de diagrama de clases UML, https://www.lucidchart.com/pages/es/tutorial-de-diagrama-de-clases-uml , Fecha de consulta 31/05/2020

[9] Programación por capas, 24 de mayo de 2020, https://es.wikipedia.org/wiki/Programaci%C3%B3n_por_capas , Fecha de consulta 30/05/2020

[10] What is software test and agile testing?, 13 de diciembre de 2017, https://rightpeoplegroup.com/what-is-software-test-and-agile-testing/ , Fecha de consulta 31/05/2020

[11] Shaumik Daityari, 8 de mayo de 2020, Angular vs React vs Vue: Which Framework to Choose in 2020, https://www.codeinwp.com/blog/angular-vs-vue-vs-react/#who-wins , Fecha de consulta 02/06/2020

[12] Ravindra Savaram, 28 de febrero de 2020, Top 20 Hybrid Mobile App Frameworks, https://mindmajix.com/top-20-hybrid-mobile-app-frameworks , Fecha de consulta 02/06/2020

[13] Junio de 2018, DB-Engines Ranking - Trend Popularity, https://db-engines.com/en/ranking_trend Fecha de consulta 03/06/2020

[14] Portal de transparencia y datos abiertos, http://gobiernoabierto.valencia.es/es/info-api/ , Fecha de consulta 20/05/2020

[15] Likert scale, 23 de junio de 2020, https://en.wikipedia.org/wiki/Likert_scale , Fecha de consulta 24/06/2020