

UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA INFORMÁTICA EN
TECNOLOGÍAS DE LA INFORMACIÓN



"DESARROLLO DE UNA APLICACIÓN
MÓVIL MULTIPLATAFORMA PARA
BÚSQUEDA DE EMPLEO UTILIZANDO
REACT NATIVE, REDUX Y FIREBASE"

TRABAJO FIN DE GRADO

Septiembre –2020

AUTOR: Alberto García Guilló

DIRECTORA: Yolanda Marhuenda García

RESUMEN

En el presente Trabajo de Fin de Grado (TFG) se realiza el estudio del uso de nuevas herramientas para el desarrollo de aplicaciones móviles. En concreto se ha utilizado la tecnología React Native que permite realizar aplicaciones móviles tanto para dispositivos Android como para dispositivos iOS, utilizando Javascript como único lenguaje y todo ello en un solo proyecto. Junto a React Native se ha utilizado la herramienta Redux para la gestión de estado en apps Javascript, así como la plataforma Firebase de Google que ofrece gran variedad de características para el desarrollo de aplicaciones y en concreto en este TFG se han considerado las correspondientes al desarrollo de base de datos en tiempo real, sistema de autenticación, almacenamiento de archivos y disparador de funciones.

Para mostrar el funcionamiento de estas herramientas se ha desarrollado una aplicación que tiene como finalidad poner en contacto a empresas con usuarios que se encuentran en el proceso de búsqueda de un empleo. La aplicación muestra para los usuarios un listado de ofertas de empleo, a las que pueden inscribirse adjuntando un currículum. Los usuarios podrán filtrar la búsqueda de ofertas gracias a un amplio filtro. También dispone de un apartado de chat donde los usuarios y las empresas podrán ponerse en contacto. Las empresas podrán publicar ofertas de empleo y valorar las inscripciones que realizan los usuarios, pudiendo rechazarlas o aceptarlas y concretar una entrevista de trabajo. Los usuarios y las empresas tienen un apartado con una agenda dónde pueden visualizar las fechas de sus entrevistas. Las empresas tienen un buscador a través del cual pueden buscar usuarios que cumplan con las características que deseen. Además, hay un apartado para los administradores de la aplicación, dónde tienen acceso a todos los usuarios y empresas y pueden dar de alta nuevos usuarios o empresas, modificarlos o eliminarlos de la aplicación.

AGRADECIMIENTOS

En primer lugar, quiero dar las gracias a mi directora del Trabajo de Fin de Grado por toda la ayuda y consejos que me ha dado para poder llevarlo a cabo.

También quiero dar la gracias a todos los profesores que he tenido durante la carrera, de los que he aprendido mucho, lo cual me ha servido para poder llegar hoy aquí y terminar el largo camino que comenzó hace unos años.

Acordarme también de mis compañeros de clase con los que he pasado largas mañanas y tardes en la biblioteca, estudiando y realizando las prácticas de las diferentes asignaturas.

Agradezco también la oportunidad que me dio el Servicio de Comunicación de la UMH dónde realicé prácticas durante un año en sus instalaciones, gracias a la bolsa de prácticas que oferta el Observatorio ocupacional de la universidad.

Y como no, dar las gracias a mis padres por el apoyo durante todo este tiempo y haberme transmitido su confianza para poder llegar hasta aquí.

ÍNDICE DE CAPÍTULOS

CAPÍTULO 1 - INTRODUCCIÓN	1
1.1 Introducción	1
1.2 Justificación de la importancia del trabajo.....	2
1.3 Qué se pretende conseguir.....	3
1.4 Qué no se pretende conseguir	3
CAPÍTULO 2 - ANTECEDENTES Y ESTADO DE LA CUESTIÓN	5
2.1 Uso de aplicaciones móviles en la actualidad.....	5
2.2 Tipos de aplicaciones móviles	6
2.2.1 Aplicaciones nativas	6
2.2.2 Aplicaciones web.....	7
2.2.3 Aplicaciones híbridas.....	8
2.3 Plataformas de desarrollo de aplicaciones multiplataforma	9
2.3.1 Ionic.....	9
2.3.2 React Native	9
2.3.3 Flutter	9
2.4 Valoración.....	10
CAPÍTULO 3 - OBJETIVOS DEL PROYECTO	11
3.1 Objetivos principales.....	11
3.2 Objetivos secundarios.....	11
CAPÍTULO 4 - HIPÓTESIS DEL TRABAJO	13
4.1 React Native	13
4.1.1 Componentes	15
4.1.2 Estado y propiedades	16
4.2 Redux.....	18

4.2.1	Actions	19
4.2.2	Reducers	20
4.2.3	Integración con React Native	21
4.3	Firestore	22
4.3.1	Autenticación	24
4.3.2	Base de datos en tiempo real	25
4.3.3	Cloud functions	26
4.3.4	Implementación.....	27
4.4	Recursos y componentes React Native	29
4.4.1	React Navigation.....	29
4.4.2	React Native Elements	33
4.4.3	React Native Gifted Chat	33
4.4.4	React Native Calendars	34
4.4.5	React Native DatePicker	35
4.4.6	React Native DocumentPicker	35
4.4.7	React Native DropdownAlert.....	36
4.4.8	React Native ImagePicker	36
4.4.9	React Native Indicators.....	37
4.4.10	React Native PDF	37
4.4.11	React Native Vector Icons	38
4.4.12	React Native HTML to PDF	38
4.4.13	XLSX	39
4.5	Librerías JavaScript	39
4.5.1	Lodash	39
4.5.2	Moment JS.....	40
4.5.3	Nodemailer.....	41

4.6	Herramientas software.....	41
4.6.1	Sublime Text.....	41
4.6.2	GitHub	42
4.6.3	Microsoft Word.....	43
4.6.4	Balsamiq Mockups.....	43
4.6.5	GanttProject.....	44
4.6.6	Draw.io	44
4.7	Lenguajes de programación.....	45
4.7.1	JavaScript.....	45
4.7.2	JSX.....	46
4.8	Herramientas hardware.....	46
4.8.1	Acer Aspire 5750G	46
4.8.2	Xiaomi Redmi Note 8 Pro	47
CAPÍTULO 5 - METODOLOGÍA Y RESULTADOS.....		49
5.1	Planificación del trabajo	49
5.1.1	Ciclo de vida	49
5.1.2	Diagrama de Gantt	53
5.2	Análisis de requisitos	54
5.2.1	Captura de requisitos y prototipo	54
5.2.2	Casos de uso.....	77
5.3	Diseño.....	89
5.3.1	Base de datos.....	89
5.3.2	Disparadores	92
5.3.3	Almacenamiento	92
5.3.4	Navegación.....	93
5.4	Implementación.....	95

5.5	Validación.....	98
5.6	Implantación.....	99
5.6.1	Generar APK (Android - Google Play)	99
5.6.2	Generar IPA (iOS - App Store).....	101
CAPÍTULO 6 - CONCLUSIONES Y PROPUESTAS		103
6.1	Conclusiones.....	103
6.2	Posibles desarrollos futuros	104
CAPÍTULO 7 - BIBLIOGRAFÍA		105



ÍNDICE DE TABLAS

Tabla 4.1: Momento de ejecución de los métodos de un componente React Native.	16
Tabla 5.2: Casos de uso del actor Administrador.....	79
Tabla 5.3: Casos de uso del actor Usuario.....	83
Tabla 5.4: Casos de uso del actor Empresa.....	87



ÍNDICE DE FIGURAS

Figura 2.1: Usuarios de Internet frente a usuarios de móvil.	5
Figura 2.2: Dispositivos para acceder a Internet en España.	5
Figura 2.3: Actividades realizadas desde el móvil en España.	6
Figura 2.4: Arquitectura de las aplicaciones nativas.	7
Figura 2.5: Arquitectura de las aplicaciones web.	8
Figura 2.6: Arquitectura de las aplicaciones híbridas.	8
Figura 4.1: Métodos del ciclo de vida de un componente React Native.	15
Figura 4.2: Términos y funcionamiento de Redux.	18
Figura 4.3: Ejemplo del funcionamiento de Redux en JavaScript.	19
Figura 4.5: Panel con el listado de usuarios en Firebase.	24
Figura 4.6: Plantillas de correo electrónico en Firebase.	25
Figura 4.7: Estructura de la base de datos de Firebase.	26
Figura 4.8: Reglas de seguridad sobre los datos de Firebase.	26
Figura 4.9: Listado de Cloud Functions del proyecto.	27
Figura 4.10: Archivo de configuración de Firebase.	27
Figura 4.11: Ejemplo del menú y la navegación con React Navigation.	32
Figura 4.12: Componentes de React Native Elements: Input, Button, ListItem. ...	33
Figura 4.13: Chat de la aplicación.	34
Figura 4.14: Agenda de la aplicación.	34
Figura 4.15: Input de tipo Calendario.	35
Figura 4.16: Seleccionador de PDF.	35
Figura 4.17: Mensaje de éxito.	36
Figura 4.18: Seleccionador de imágenes.	36
Figura 4.19: Ejemplo de uso de React Native Indicators.	37

Figura 4.20: Visor de archivos PDF.....	37
Figura 4.21: Algunos iconos de la aplicación.....	38
Figura 4.22: Listado exportado en formato PDF.....	38
Figura 4.23: Listados exportados en formatos Excel y CSV.....	39
Figura 4.24: Interfaz de Sublime Text 3.....	42
Figura 4.25: Actualización del proyecto almacenado en GitHub.	43
Figura 4.26: Interfaz de Balsamiq Mockups.....	44
Figura 4.27: Interfaz de GanttProject.....	44
Figura 4.28: Interfaz de Draw.io.	45
Figura 5.1: Diagrama de Gantt.	53
Figura 5.2: Duración y fechas de las tareas.	53
Figura 5.3: Prototipo Inicio sesión y Registro.	55
Figura 5.4: Prototipo listado ofertas de empleo y filtro.....	57
Figura 5.5: Prototipo detalles oferta, compartir, inscribirse y cancelar inscripción.	57
Figura 5.6: Prototipo listado de inscripciones, detalles y cancelar inscripción.....	58
Figura 5.7: Prototipo listado chats, chat y eliminar chat.	59
Figura 5.8: Prototipo perfil, ajustes, editar perfil y cambiar contraseña.	60
Figura 5.9: Prototipo listado currículums, detalles, crear nuevo y editar.	61
Figura 5.10: Prototipo listado entrevistas, preferencias y cerrar sesión.	62
Figura 5.11: Prototipo listado ofertas publicadas, exportar, publicar oferta, detalles, resolver inscripción, cerrar oferta, eliminar oferta y editar oferta.....	63
Figura 5.12: Prototipo inscritos en la oferta, exportar, detalles inscripción y resolver inscripción.....	64
Figura 5.13: Prototipo inscritos en todas las ofertas, exportar, detalles y resolver.	65

Figura 5.14: Prototipo listado de usuarios, exportar y búsqueda/filtrado usuarios.	66
Figura 5.15: Prototipo listado chats, chat y eliminar chat.	66
Figura 5.16: Prototipo perfil, ajustes y editar perfil.	67
Figura 5.17: Prototipo cambiar contraseña, ver entrevistas, editar entrevistas y cerrar sesión.....	68
Figura 5.18: Prototipo listado administradores, exportar y crear administrador....	69
Figura 5.19: Prototipo detalles del administrador, eliminar y editar administrador.	69
Figura 5.20: Prototipo listado usuarios, exportar y crear usuario.	70
Figura 5.21: Prototipo detalles del usuario, eliminar y editar usuario.	71
Figura 5.22: Prototipo Listado currículums del usuario, exportar y crear currículum.	71
Figura 5.23: Prototipo detalles currículum, eliminar y editar currículum.	72
Figura 5.24: Prototipo listado inscripciones del usuario, exportar, detalles y resolver inscripción.	73
Figura 5.25: Prototipo listado empresas, exportar y crear empresa.	73
Figura 5.26: Prototipo detalles empresa, eliminar empresa y editar empresa.....	74
Figura 5.27: Prototipo listado ofertas publicadas por la empresa, exportar y publicar nueva oferta.	75
Figura 5.28: Prototipo detalles oferta, resolver inscripción, cerrar oferta, eliminar oferta y editar oferta.	75
Figura 5.29: Prototipo listado inscritos, exportar, detalles y resolver inscripción..	76
Figura 5.30: Prototipo perfil, ajustes, editar perfil, cambiar contraseña y cerrar sesión.....	76
Figura 5.31: Diagrama C.U. Administrador: Autenticación.	79
Figura 5.32: Diagrama C.U. Administrador: Perfil y Ajustes.	79
Figura 5.33: Diagrama C.U. Administrador: Gestionar Administradores.	80

Figura 5.34: Diagrama C.U. Administrador: Gestionar Usuarios.	80
Figura 5.35: Diagrama C.U. Administrador: Gestionar currículums usuario.	80
Figura 5.36: Diagrama C.U. Administrador: Gestionar inscripciones usuario.	81
Figura 5.37: Diagrama C.U. Administrador: Gestionar empresas.....	81
Figura 5.38: Diagrama C.U. Administrador: Gestionar ofertas de empleo de la empresa.....	81
Figura 5.39: Diagrama C.U. Administrador: Gestionar inscripciones en las ofertas de empleo de la empresa.	82
Figura 5.40: Diagrama C.U. Usuario: Autenticación.	83
Figura 5.41: Diagrama C.U. Usuario: Perfil y Ajustes.....	84
Figura 5.42: Diagrama C.U. Usuario: Gestionar currículums.....	84
Figura 5.43: Diagrama C.U. Usuario: Ofertas de empleo.	84
Figura 5.44: Diagrama C.U. Usuario: Inscripciones.....	85
Figura 5.45: Diagrama C.U. Usuario: Chats.	85
Figura 5.46: Diagrama casos de uso C.U. Empresa: Autenticación.	87
Figura 5.47: Diagrama C.U. Empresa: Perfil y Ajustes.....	87
Figura 5.48: Diagrama C.U. Empresa: Ofertas.....	87
Figura 5.49: Diagrama C.U. Empresa: Inscripciones.....	88
Figura 5.50: Diagrama C.U. Empresa: Buscar usuarios.	88
Figura 5.51: Diagrama C.U. Empresa: Chats.	88
Figura 5.52: Diseño de la base de datos.	90
Figura 5.53: Disparadores de la aplicación.....	92
Figura 5.54: Mapa de pantallas y navegación de la aplicación.....	94
Figura 5.55: Bienvenida, inicio de sesión y registro con correo de bienvenida.....	95
Figura 5.56: Ofertas de empleo, filtro, detalles, compartir e inscripción del usuario.	95
Figura 5.57: Inscripciones y chats del usuario.....	96

Figura 5.58: Currículums, entrevistas y preferencias del usuario.....	96
Figura 5.59: Ofertas, exportar, publicar e inscripciones de la empresa.....	96
Figura 5.60: Búsqueda de usuarios con filtro, entrevistas y cerrar sesión de la empresa.	97
Figura 5.61: Gestión de administradores del administrador.	97
Figura 5.62: Gestión de usuarios y empresas del administrador.....	97
Figura 5.63: Modo apaisado de la aplicación.	98
Figura 5.64: Mensajes éxito, error y espera de la aplicación.	98
Figura 5.65: Generar clave firmada mediante <i>keytool</i>	99



CAPÍTULO 1 - INTRODUCCIÓN

1.1 Introducción

En la actualidad el uso de dispositivos móviles ha crecido tanto gracias a la globalización, lo que permite que prácticamente todo el mundo que disponga de medios pueda estar conectado a Internet. La forma de acceder a la información también ha ido evolucionando, de modo que el acceso a través de aplicaciones móviles ha ido ganando terreno frente al acceso a través de la web.

A la hora de desarrollar una aplicación móvil surgen dudas en torno a qué tecnologías emplear y cuál puede ser la más conveniente para cada caso. Podemos diferenciar tres tipos de aplicaciones para dispositivos móviles: aplicaciones web, aplicaciones nativas y aplicaciones híbridas.

En el capítulo 2 se profundiza en las características de cada tipo de aplicación, llegando a la conclusión que las aplicaciones que mejor rendimiento ofrecen son las nativas.

En el capítulo 4 se introduce React Native, una herramienta que nos permite desarrollar aplicaciones nativas para Android e iOS mediante un único proyecto programado en JavaScript. También se comenta el papel que juega Redux junto con React Native, y finalmente se describen las características de Firebase, que es una plataforma que ofrece servicios de autenticación, almacenamiento y base de datos en tiempo real, entre otras.

Para mostrar un ejemplo de uso de las tres tecnologías, se ha desarrollado una aplicación móvil de búsqueda de empleo. En el capítulo 5 se repasa la planificación del trabajo, el análisis de requisitos, el diseño, su implementación y la implantación.

1.2 Justificación de la importancia del trabajo

Para realizar la programación de aplicaciones nativas no es necesario utilizar sus lenguajes operativos, sino que existen diferentes alternativas. Esto es, se puede programar una aplicación nativa para Android e iOS, sin tener que programar en Java y Objective-C, respectivamente. La tecnología React Native permite desarrollar en un único proyecto una aplicación nativa para Android y otra para iOS utilizando únicamente JavaScript.

En este TFG se han elegido algunas tecnologías que no se han visto durante el Grado, para de algún modo darles visibilidad y mostrar que son óptimas para el desarrollo de aplicaciones móviles nativas.

Además, también se han utilizado conocimientos adquiridos durante el Grado. Un alto peso del trabajo recae en la Ingeniería del software, donde se expone la planificación, el análisis de requisitos y el diseño. Estos conocimientos han sido adquiridos en asignaturas como Ingeniería del software (2º), Interfaces de usuario (2º), Gestión de proyectos de ingeniería del software (2º) y Diseño de sistemas interactivos (4º). En la asignatura Desarrollo de aplicaciones para dispositivos móviles (4º) se vio la comparativa entre las aplicaciones móviles web, nativas e híbridas, lo que ha servido de punto de partida a la hora de plantear el trabajo.

En cuanto a bases de datos, se explica más adelante en detalle, pero en el caso de Firebase, no trabaja de forma relacional, sino que se hace a través de objetos JavaScript.

1.3 Qué se pretende conseguir

El objetivo del TFG es dar a conocer React Native y demostrar que es una buena opción a la hora de desarrollar una aplicación móvil nativa. Se explica en qué consiste el framework, cómo integrarlo con Redux y las características que ofrece Firebase.

Para plasmar la teoría de cómo funcionan estas herramientas se ponen en práctica los conocimientos adquiridos y se explica el diseño y desarrollo de una aplicación de búsqueda de empleo, en la que las empresas publican ofertas de empleo y los usuarios pueden inscribirse a ellas.

1.4 Qué no se pretende conseguir

No se busca realizar una aplicación móvil que resulte innovadora, sólo se trata de un ejemplo con el que mostrar que con el uso de React Native, Redux y Firebase se puede desarrollar una aplicación móvil multiplataforma sin la necesidad de programar en diferentes lenguajes dependiendo del sistema operativo en el que se trabaje.

Al final del trabajo se enumeran una serie de mejoras futuras que se podrían incluir en la aplicación para hacerla más completa.

CAPÍTULO 2 - ANTECEDENTES Y ESTADO DE LA CUESTIÓN

2.1 Uso de aplicaciones móviles en la actualidad

Estudios recientes señalan que, en el año 2017, por primera vez en la historia, el número de dispositivos móviles existentes superó al número de habitantes del planeta. En cuanto al número de usuarios de usuarios a nivel mundial con acceso a Internet se sitúa en el 50% de la población, mientras que el 66% de la población mundial dispone de un teléfono móvil. [1]



Figura 2.1: Usuarios de Internet frente a usuarios de móvil.

En España, el dispositivo más utilizado para acceder a Internet es el teléfono móvil, con casi el 95% de la población. La tablet es otro dispositivo bastante utilizado, en torno al 57%, pero en los últimos años su uso se ha estancado. El sistema operativo preferido de los usuarios de móviles y tablets es Android con un 83%, mientras que iOS se queda en torno al 11%.

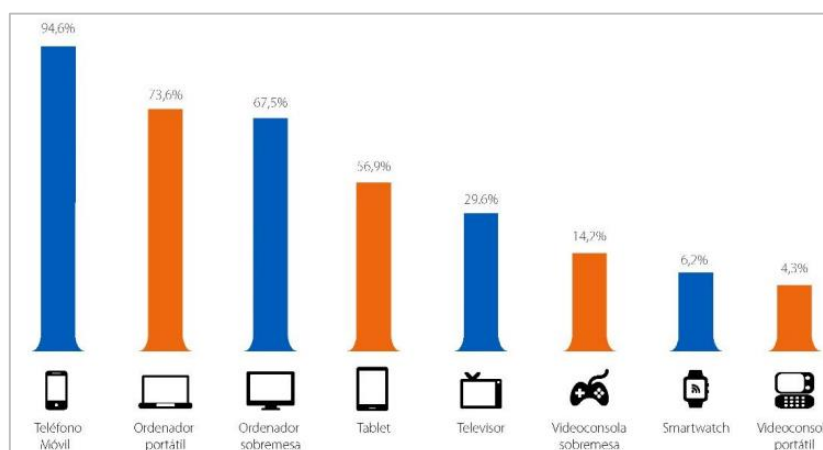


Figura 2.2: Dispositivos para acceder a Internet en España.

En cuanto al uso que se le da al móvil en España, la actividad que más realizan los usuarios es el acceso al correo electrónico con un 85%, seguido del uso de aplicaciones de mensajería instantánea con un 82% y la navegación por Internet con un 77%.

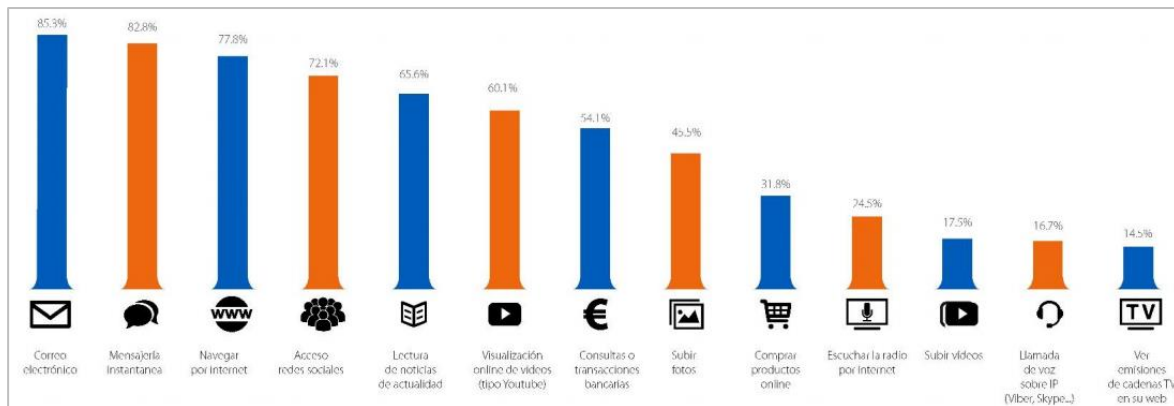


Figura 2.3: Actividades realizadas desde el móvil en España.

Visto el crecimiento en los últimos tiempos del uso de los dispositivos móviles y teniendo en cuenta el dato de que prácticamente el 95% de los españoles utiliza el móvil, y que es el medio más común de acceso a Internet, no cabe duda de que es un sector con un amplio mercado y el cual merece la pena explotar.

2.2 Tipos de aplicaciones móviles

Se pueden distinguir entre tres tipos de aplicaciones para dispositivos móviles según la tecnología que se emplee para su desarrollo: las aplicaciones nativas, las aplicaciones web y las aplicaciones híbridas. [2] En los siguientes apartados se incluyen cuáles son las características de cada una de ellas.

2.2.1 Aplicaciones nativas

Las aplicaciones nativas son las desarrolladas expresamente para un determinado sistema operativo, de forma que utilizan el SDK de dicha plataforma. En Android, las aplicaciones son programadas en Java, mientras que en iOS el lenguaje con el que son programadas es Objective-C, o Swift más recientemente.

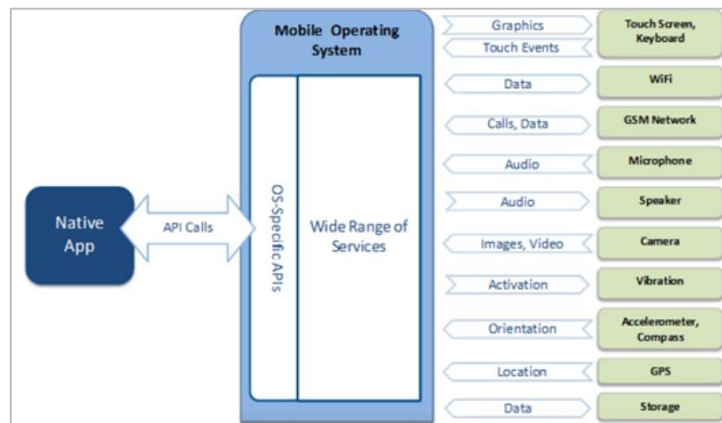


Figura 2.4: Arquitectura de las aplicaciones nativas.

Este tipo de aplicaciones tiene acceso a todos los servicios del sistema operativo del dispositivo, lo que aumenta la experiencia de usuario. Es, por tanto, la mejor opción cuando se trata de aplicaciones que precisen de un gran rendimiento.

Su distribución se realiza mediante las tiendas oficiales de aplicaciones, por tanto, debe ser aprobada para poder distribuirse por esta vía. El hecho de que se distribuya a través de una tienda oficial aumenta su visibilidad.

Respecto estas aplicaciones, hay que tener en cuenta que desarrollar una aplicación nativa para una única plataforma dejaría fuera gran parte del mercado y que, en caso de desarrollar aplicaciones nativas para diferentes plataformas, precisa desarrolladores con conocimientos en todas las tecnologías, lo que aumentaría los costes de desarrollo y el mantenimiento de la aplicación.

2.2.2 Aplicaciones web

Las aplicaciones web se desarrollan con tecnologías como HTML, CSS y JavaScript, y no dejan de ser versiones de páginas web adaptadas a cualquier dispositivo. El acceso se realiza a través de los navegadores, por lo que se trata de aplicaciones multiplataforma que no precisan de la necesidad de ser aprobada por ningún fabricante.

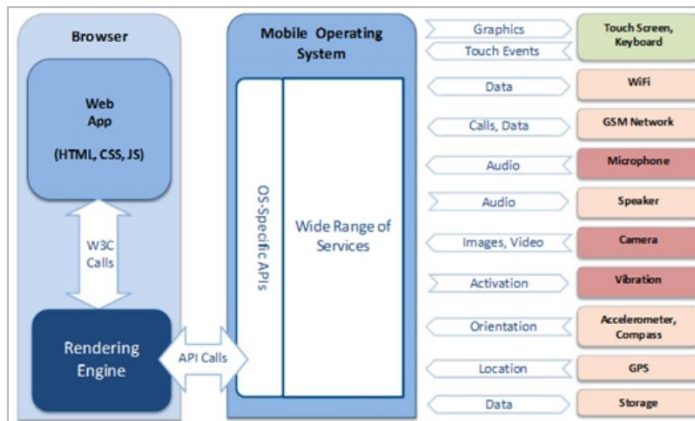


Figura 2.5: Arquitectura de las aplicaciones web.

Las aplicaciones web disponen de un acceso muy limitado a los servicios de los dispositivos móviles, por lo que sus funciones se ven limitadas. El acceso a ellas, al ser a través de navegadores, requiere de una conexión a Internet.

2.2.3 Aplicaciones híbridas

Las aplicaciones híbridas mezclan características de las aplicaciones nativas y web. Su desarrollo se realiza mediante el uso de tecnologías como HTML, CSS o JavaScript. El código se compila por separado para cada plataforma.

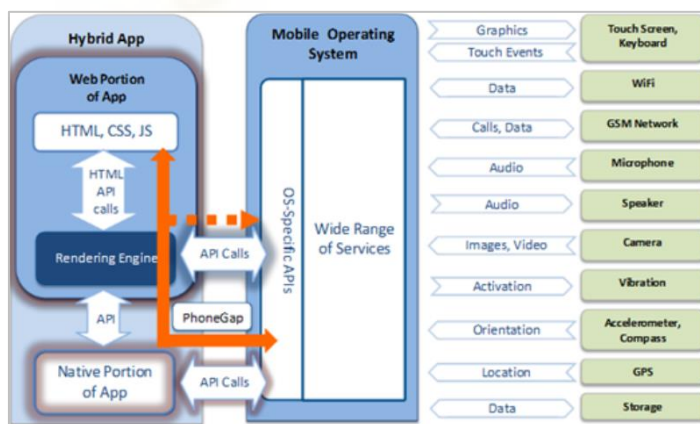


Figura 2.6: Arquitectura de las aplicaciones híbridas.

En este caso sí tienen acceso a los servicios del dispositivo, sin embargo, al tratarse de código HTML, son renderizadas por el navegador de la plataforma para la que se desarrolla, lo que provoca un rendimiento negativo. Para conseguir una buena experiencia de usuario es necesario dedicarle tiempo.

2.3 Plataformas de desarrollo de aplicaciones multiplataforma

A continuación, se describen algunas de las tecnologías más utilizadas para el desarrollo de aplicación multiplataforma. [3]

2.3.1 Ionic

Ionic es un framework destinado a la creación de aplicaciones híbridas. Fue lanzado en 2013 y combina AngularJS y Apache Cordova. Se programa mediante el uso de HTML, CSS y JavaScript. Las aplicaciones se embeben en un webview, por lo que su rendimiento es algo menor respecto a otras alternativas. El punto más fuerte es que prácticamente el 100% de su código es reutilizable. Su uso y popularidad ha sido elevado durante años, aunque actualmente ha perdido algo de mercado. No se han encontrado aplicaciones famosas que utilicen este framework.

2.3.2 React Native

React Native permite crear aplicaciones Android e iOS mediante el uso de JavaScript. Se lanzó a finales de 2015 y es propiedad de Facebook. Las aplicaciones resultantes son nativas, por lo que el rendimiento es elevado. Utiliza componentes nativos. Su forma de programar está enfocada en componentes lo que implica que prácticamente todo el código sea reutilizable. A nivel de popularidad se podría decir que es el más conocido, y además cuenta con una numerosísima comunidad de desarrolladores. Multitud de aplicaciones móviles han sido desarrolladas con este framework, como por ejemplo Facebook o Instagram.

2.3.3 Flutter

Flutter es un software de desarrollo de aplicaciones Android, iOS y web. Fue creado por Google y se lanzó en 2017. El lenguaje en el que se programan las aplicaciones es Dart, creado también por Google. Su rendimiento es elevado. Utiliza componentes nativos, pero creados por esta tecnología. El código es altamente reutilizable. Está ganando popularidad poco a poco. Dos de las aplicaciones más famosas desarrolladas con esta tecnología son Google Ads y Alibaba.

2.4 Valoración

Observando los datos de estudios recientes sobre el uso de dispositivos móviles y visto que el acceso a Internet a través de aplicaciones móviles se encuentra en cifras de récord, queda demostrado que el desarrollo de aplicaciones móviles es un sector en alza. Por ello, las empresas no quieren quedarse atrás, y cada vez más, introducen sus marcas y productos a través de aplicaciones móviles.

A la hora de desarrollar una aplicación móvil, el equipo de desarrollo debe enfrentarse a tomar una decisión en torno a qué tipo de aplicación va a desarrollar. Todo dependerá de las exigencias de la aplicación, los recursos económicos y los recursos humanos de los que disponga.

Como se ha explicado, las aplicaciones nativas son las que mejor rendimiento ofrecen, y esto es un punto muy importante para tener en cuenta. Sin embargo, el desarrollo de aplicaciones nativas mediante el uso de lenguajes nativos genera altos costes de desarrollo y mantenimiento, por lo que habría que encontrar la forma de desarrollar aplicaciones nativas, pero sin la necesidad de desarrollar en varios lenguajes de programación. En el análisis de herramientas que ofrecen la posibilidad de desarrollar aplicaciones nativas, se ha escogido React Native para su estudio en profundidad, ya que permite la creación de aplicaciones nativas, pero con la ventaja de que son programadas en un solo lenguaje y proyecto.

CAPÍTULO 3 - OBJETIVOS DEL PROYECTO

En este trabajo se analiza React Native como framework de desarrollo de aplicaciones nativas. Se muestra su uso mediante el diseño y desarrollo de una aplicación móvil de búsqueda de empleo.

3.1 Objetivos principales

En este trabajo se analiza React Native como herramienta para la creación de aplicaciones móviles nativas. Además, se estudia cómo realizar la integración con Redux para manejar el estado de las aplicaciones; y cómo sacar partido a las características que ofrece Firebase, una herramienta para el desarrollo de aplicaciones y bases de datos. También se describen las tecnologías y se comentan sus principales características.

Se quiere demostrar los conocimientos adquiridos durante estos años en el Grado, así como mostrar que hay nuevas tecnologías que no son vistas en las asignaturas, pero que en la actualidad son empleadas en multitud de desarrollos.

3.2 Objetivos secundarios

A partir del análisis de las tecnologías escogidas, se documenta el diseño y desarrollo de una aplicación de búsqueda de empleo. La aplicación pone en contacto a usuarios con las empresas a través de inscripciones a ofertas de empleo.

Las empresas publican ofertas de empleo y pueden evaluar las inscripciones que realicen los usuarios. Pueden concretar entrevistas de trabajo, citando a los usuarios para una fecha determinada. Por su parte, los usuarios tienen un listado de ofertas de empleo a las que pueden inscribirse. Pueden filtrar las ofertas gracias a un completo buscador. Deben adjuntar un currículum para completar su inscripción. Hay un chat para que se comuniquen los usuarios y las empresas. Las empresas también pueden localizar a usuarios a través de un filtro de búsqueda.

CAPÍTULO 4 - HIPÓTESIS DEL TRABAJO

4.1 React Native

React Native es un framework de desarrollo que permite crear aplicaciones móviles nativas de Android e iOS mediante el uso de JavaScript y JSX. Fue lanzada a finales de 2015 y es propiedad de Facebook. [4]

Su metodología de desarrollo permite crear aplicaciones móviles nativas con un único lenguaje de programación, su sintaxis es sencilla, basándose en la de React, y permite la reutilización del código gracias a la programación de componentes.

El desarrollo desde un equipo macOS permite probar la aplicación en emuladores iOS y Android, en cambio, si se desarrolla desde un equipo Windows o Linux, la aplicación únicamente se podrá probar en emuladores Android. Esto no impide el desarrollo de la aplicación, solo que limita al desarrollador a la hora de ir chequeando los cambios que va haciendo en ella.

Para empezar a desarrollar un proyecto en React Native, primero hay que instalar las dependencias, que varían según el sistema operativo desde el que se trabaje:

- macOS: Node, Watchman, la interfaz de línea de comandos de React Native, una versión reciente de JDK (Java SE Development Kit) y Xcode/Android Studio.
- Windows: Node, la interfaz de línea de comandos de React Native, Python 2, una versión reciente de JDK y Android Studio.
- Linux: Node, la interfaz de línea de comandos de React Native, una versión reciente de JDK y Android Studio.

Para crear una nueva aplicación, desde la consola de comandos se deberá introducir el comando `react-native init nombreproyecto`. Con ello, se creará un nuevo proyecto de React Native con el nombre que se le ha indicado y en el

directorio desde el que se ejecuta el comando. Dicho comando creará la estructura de la aplicación con un “Hola mundo”, lo que permitirá iniciar el desarrollo con gran parte del trabajo de configuración hecho.

La estructura del proyecto está organizada de la siguiente manera:

- *android*: carpeta con la configuración para Android.
- *ios*: carpeta con la configuración para iOS.
- *node_modules*: carpeta con los módulos de Node que utiliza la aplicación.
- *app.json*: archivo con el nombre y la versión de la aplicación.
- *index.js*: archivo principal de la aplicación.
- *package.json*: archivo que incluye el nombre y la versión de los paquetes que Node utiliza en la aplicación.
- *.gitignore*: archivo para indicar que carpetas y ficheros no sincronizar con GitHub.
- Otros archivos de configuración.

Para organizar mejor el proyecto, es común crear una carpeta llamada *src* donde ir añadiendo los componentes, multimedia o cualquier fichero que se vaya creando.

Para emular la aplicación, se ha optado por un dispositivo físico conectado mediante USB al equipo donde se desarrolla la aplicación. Para habilitar este método, es necesario activar la depuración USB. Además, será necesario ejecutar el comando `adb reverse tcp:8081 tcp:8081`, para establecer la conexión entre el equipo de desarrollo y el dispositivo donde se va a ejecutar la aplicación.

Para ejecutar la aplicación, independientemente de si es un dispositivo físico o virtual, el comando que se debe introducir es `react-native run-android` desde el directorio de la aplicación.

En los siguientes apartados se explica la forma de programar a través de componentes y qué son el estado y las propiedades de la aplicación.

4.1.1 Componentes

Los componentes permiten dividir la interfaz de usuario en elementos independientes y reutilizables. Los desarrolladores pueden crear componentes personalizados combinando los componentes por defecto que ofrece React Native. Algunos componentes utilizados comúnmente son `<View>`, `<Text>`, `<Button>`, `<TextInput>`, `<FlatList>` o `<Image>`.

Los componentes se definen como clases, y lo que se defina en el interior del método `render()` será lo que se muestre por pantalla. El método `render()` retorna un conjunto de componentes de React Native que dan forma a un componente personalizado.

Los componentes disponen de varios métodos para sobrescribir su ciclo de vida, dentro de cada uno de ellos se puede definir aquello que se desee que realice la aplicación desde el momento en que se inicializa el componente, durante su montaje, cuando se actualiza o cuando se desmonta.

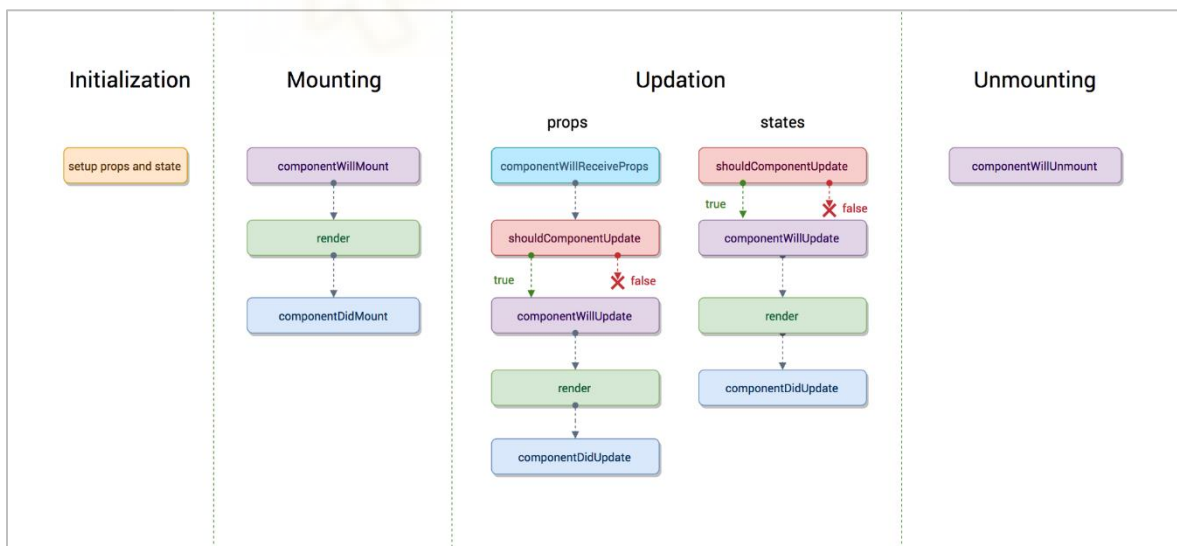


Figura 4.1: Métodos del ciclo de vida de un componente React Native.

Los métodos se utilizan según el momento en que se desea que se ejecute un determinado trozo del código. En la siguiente tabla se indican los métodos disponibles y cuándo se ejecutan:

Método	Momento de ejecución
<code>render()</code>	Único método obligatorio en un componente.
<code>componentWillMount()</code>	Antes de que el componente sea montado.
<code>componentDidMount()</code>	Cuando el componente ha sido montado.
<code>componentWillReceiveProps():</code>	Cuando se reciben nuevas propiedades de un componente padre.
<code>componentWillUpdate()</code>	Antes del <code>render()</code> (en la primera llamada al <code>render()</code> no se ejecuta) justo después de recibir el estado y las propiedades.
<code>componentDidUpdate():</code>	Después de que se haya actualizado el componente.
<code>componentWillUnmount()</code>	Antes de que se destruya el componente.

Tabla 4.1: Momento de ejecución de los métodos de un componente React Native.

4.1.2 Estado y propiedades

Los componentes pueden tener estado (*state*) y propiedades (*props*).

Las propiedades no pueden ser modificadas dentro de un componente. Son pasadas de un componente padre a un hijo, siempre en ese sentido, y nunca de un componente hijo a uno padre. Esto permite diseñar componentes reutilizables y que en diferentes partes de la aplicación reciben unas propiedades diferentes. La forma de acceder a las propiedades de un componente hijo es a través de `this.props`.

En el siguiente código se indica la estructura básica de dos componentes. Se definen como clase y dentro del método `render()` se retorna la estructura que se desea. El primero es un componente dónde se mostrará un el texto que se reciba dentro de la propiedad *message*, mientras que el segundo importa al primer componente y lo introduce en el interior de una vista. De esta forma se muestra como es la reutilización del código a través de los componentes.

```

export default class Success extends Component {
  render () {
    return (
      <Text>{this.props.message}</Text>
    )
  }
}

export default class SuccessView extends Component {
  render () {
    return (
      <View>
        <Success message={'¡Guardado correctamente!'}/>
      </View>
    )
  }
}

```

Por su parte, el estado es algo propio de cada componente y puede cambiar. Para definir un estado, dentro del constructor se define mediante `this.state = { input: ''}`. La forma de actualizar el estado es mediante `this.setState({ input: text })`, y para acceder al estado, mediante `this.state.input`.

Sin embargo, en este trabajo en lugar de utilizar la opción anterior para tratar el estado de la aplicación desarrollada, se ha optado por hacerlo mediante el uso de Redux, que será descrito en el siguiente apartado.

Cabe destacar que la mayor dificultad de todo el trabajo ha sido comprender la forma de programar con esta tecnología. Han sido varias semanas leyendo y viendo tutoriales por la web hasta alcanzar un nivel medio con el que poder llevar a cabo la aplicación desarrollada.

4.2 Redux

Redux es una librería que ofrece una solución sólida y estable para gestionar el estado de las aplicaciones desarrolladas en React Native. Siguiendo unos patrones, permite transformar el complejo modo de controlar los estados de la aplicación en una forma organizada y fácil de entender mediante el uso de JavaScript. Es una evolución de Flux, que es una arquitectura para el manejo de datos, y simplificada gracias al lenguaje Elm. [5]

Para comprender cómo funciona Redux, hay que explicar cuatro términos:

- *Store*: objeto que contiene los datos de la aplicación.
- *Action*: objeto que indica al *reducer* cómo cambiar sus datos. Tiene dos propiedades: *type* y *payload*.
- *Reducer*: función que retorna datos.
- *State*: datos que utiliza la aplicación.

El estado (*state*) de la aplicación se almacena en un único *store*. El estado es de solo lectura, de modo que para modificarlo hay que lanzar una acción (*action*). Los cambios se producen mediante el uso de los *reducers*, que son funciones que toman el estado anterior de la aplicación junto con una acción, y retornan un nuevo estado. En la siguiente figura se muestra de forma simplificada la conexión entre estos términos y su funcionamiento.

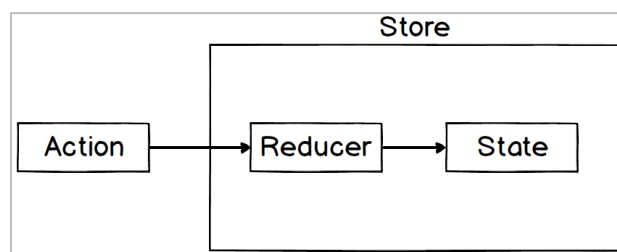


Figura 4.2: Términos y funcionamiento de Redux.

Para entender mejor el funcionamiento, se muestra un ejemplo con código JavaScript:

<pre> const reducer = () => []; const store = Redux.createStore(reducer); store.getState(); // [] const action = { type: 'split_string', payload: 'alberto' }; const reducer = (state = [], action) => { if(action.type === 'split_string'){ return action.payload.split(""); } return state; }; store.dispatch(action); store.getState(); // ['a', 'l', 'b', 'e', 'r', 't', 'o'] </pre>	<p>En primer lugar creamos la 'store', un objeto que almacenará el estado ('state') de la aplicación y los 'reducers'. Para crear la 'store', será necesario pasarle un 'reducer', que es una función que retorna algún valor. En este caso el 'reducer' es una 'arrow function' que devuelve un array.</p> <p>Por defecto, cuando creamos un Redux 'store' y le pasamos un 'reducer', los datos que retorna el 'reducer' se convertirán en el 'state' de la aplicación. De forma que al solicitar el 'state' de la aplicación, devolverá lo mismo que retorna el 'reducer'. En este caso retorna un array vacío.</p> <p>Para cambiar el 'state' de la aplicación, habrá que crear una 'action' para posteriormente pasársela al 'reducer' y que este modifique el estado de la aplicación. Una 'action', es un objeto JS que tiene dos propiedades: 'type' y 'payload'.</p> <p>'type': siempre será un string. Será la encargada de comunicar al 'reducer' que operación específica debe realizar, algo así como un comando o una instrucción.</p> <p>'payload': puede ser cualquier tipo de data. Serán los datos sobre los que queremos que el 'reducer' realice alguna acción.</p> <p>El 'reducer' puede recibir dos parámetros: 'previousState' y 'action'. Retorna un nuevo 'state'. Si la propiedad 'type' de la 'action' que recibe el 'reducer' es igual una de las opciones que hay en el interior del 'reducer', se devolverá el valor que se desee. En este otro 'reducer' se devuelve un array de caracteres formado por el 'payload' de la 'action' que recibe el 'reducer'. Por defecto habrá que devolver el 'state'.</p> <p>Pasamos la 'action' a la 'store'.</p> <p>Obtenemos el nuevo 'state' de la aplicación. En este caso será un array de caracteres.</p>
---	--

Figura 4.3: Ejemplo del funcionamiento de Redux en JavaScript.

4.2.1 Actions

Como se ha comentado, las *action* son objetos que contienen información y se utilizan para enviar los datos de la aplicación al *store*. Son objetos JavaScript que contienen una propiedad llamada *type* y otra propiedad opcional que suele llamarse *payload*. La propiedad *type* indica qué tipo de acción se debe realizar. Para una mejor gestión, suelen definirse como constantes. Las constantes se pueden definir en un archivo aparte. La propiedad *payload* tiene como función almacenar la información que se pudiera querer pasar a través de la *action*. En el siguiente código se muestra cómo es una *action*, un objeto JavaScript con las propiedades *type* y *payload*.

```

{
  type: INPUT_CHANGED,
  payload: { prop, value }
}

```

Otro término que debe explicarse es el de los creadores de acciones (*action creators*). Se trata de funciones que retornan acciones. Son sencillas funciones que retornan un objeto JavaScript. En el código siguiente se muestra la definición de un *action creator*, que retorna una *action* dónde se pasa una propiedad y un valor a través del *payload*.

```
export const inputChanged = ({ prop, value }) => {
  return {
    type: INPUT_CHANGED,
    payload: { prop, value }
  }
};
```

Para llamar al *action creator*, existe la función `dispatch()`, la cual provoca que se retorne la acción. El acceso a esta función solo puede hacerse desde el *store*, por lo que para una mejor organización del código es habitual utilizar la función `connect()` de la librería *react-redux*. [6]

4.2.2 Reducers

Los *reducers* son funciones JavaScript que retornan datos y se encargan de decir al *store* cómo ha cambiado el estado de la aplicación después de recibir el estado anterior y una *action*. La forma de comunicarse con el *store* es mediante el envío de una *action*. La función *reducer* siempre tiene que recibir por defecto el estado inicial la primera vez que es llamada.

Cada vez que se ejecuta una acción, el *store* la reenvía al *reducer*, y éste comprueba si el tipo de la acción está definido en la propia función, y en caso de que así sea, retornar un nuevo estado al *store*. Suele hacerse mediante el uso de condicionales *if* o *switch*. En el siguiente código se muestra la definición de un *reducer* que almacena en su estado el email y la contraseña. Cuando recibe una nueva acción, comprueba mediante un condicional *switch* si la propiedad *type* coincide con uno de sus casos. En este caso el *reducer* retorna un nuevo estado en el que actualiza una propiedad y le asigna un valor, todo ello recibido en la propiedad *payload* de la acción.

```

const INITIAL_STATE = {
  email: '',
  password: ''
}

export default (state = INITIAL_STATE, action) => {
  switch(action.type) {
    case INPUT_CHANGED:
      return {
        ...state,
        [action.payload.prop]: action.payload.value
      };
  }
};

```

A la hora de estructurar un proyecto, es habitual crear cada *reducer* en un archivo independiente. Cada *reducer* retornará un objeto con el nuevo estado, y es mediante el uso de `combineReducers()` de Redux la forma en la que se consigue combinar los resultados en un único objeto. Si se diera el caso de que ningún *reducer* modificara el estado anterior de la aplicación, no se retornaría ningún objeto. El código siguiente muestra como es un archivo que combina varios *reducer*.

```

import AuthReducer from ...;
import UserProfileReducer from ...;

export default combineReducers({
  auth: AuthReducer,
  profile: UserProfileReducer
});

```

4.2.3 Integración con React Native

La forma de integrar Redux con React Native es mediante la función `createStore()`. Esta función de Redux recibe como parámetro los reducers de la aplicación. En el ejemplo que se muestra a continuación, recibe otro parámetro `applyMiddleware(ReduxThunk)`, que recibiendo a su vez `ReduxThunk` de la librería *redux-thunk*, sirve para el uso de funciones asíncronas, y que en el caso de la aplicación desarrollada, al estar esperando respuestas de la base de datos de Firebase, se hace necesario. [7]

```

class App extends Component{
  const store = createStore(reducers, {}, applyMiddleware(ReduxThunk))
  render() {
    return(<Provider store={store}>
      <AppContainer />
    </Provider>);
  }
}

```

El encargado de hacer que el *store* sea accesible al resto de componentes que se encuentran anidados en su interior, es el componente `Provider` de *react-redux*, siempre que incluyan la función `connect()`, de la que se ha hablado en el apartado de las acciones. En el siguiente código se muestra la forma en la que se exporta el componente `ChangePassword` que hace uso de la función `connect()`.

```

export default connect(mapStateToProps, { inputChanged })(ChangePassword);

```

Aquellos componentes de la aplicación que tienen que acceder al estado almacenado en el *store*, lo hacen a través de la función `connect()`, que recibirá el *mapStateToProps* y los *actions creators* que precise. El *mapStateToProps* es una función que se encarga de transformar el estado almacenado en el *store*, en las *props* que utilizará el componente.

```

const mapStateToProps = ({ auth }) => {
  const { ... } = auth;
  return { ... };
};

```

Esta tecnología también ha supuesto una gran dedicación de tiempo para llegar a comprenderla. Dado que se integra con *React-Native*, ambas se han ido aprendiendo de forma simultánea.

4.3 Firebase

Firebase es una plataforma que ofrece diferentes servicios para el desarrollo de aplicaciones móviles y web. Es heredera de la antigua startup *Envolve*, la cual proporcionaba a los desarrolladores una API que permitía integrar un chat en sus desarrollos web. La gente empezó a usar *Envolve* para intercambiar datos, en lugar

de simples mensajes, lo que llevó a sus desarrolladores a crear Firebase en el año 2012, de modo que separaba el servicio de chat dejándolo para Envolv, y enfocó Firebase como una plataforma Backend-as-a-Service que ofrecería funcionalidad en tiempo real. Debido a su popularidad, en 2014 Google adquirió Firebase. [8]

Firebase proporciona a los desarrolladores un servicio de Autenticación que permite identificar a los usuarios mediante diferentes métodos: correo electrónico y contraseña, teléfono o redes sociales. [9]

Su servicio más destacado es el de base de datos en tiempo real. Permite tener sincronizados en tiempo real a los usuarios con la base de datos. La base de datos se estructura como un objeto JSON.

Otra de sus características es la del almacenamiento, de forma que permite procesar y almacenar archivos multimedia o documentos.

Cloud Functions es otra de las opciones que ofrece, y se trata de unos disparadores (triggers) que permiten ejecutar código de servidor como respuesta a los eventos lanzados por las funciones de Firebase.

Otras características de Firebase más recientes y que no se ha trabajado con ellas en el desarrollo de este TFG son: el servicio de Hosting, aunque está enfocada para aplicaciones web; Cloud Messaging, un servicio para enviar mensajes y notificaciones a los usuarios, de forma conjunta o personalizada para cada uno de ellos; y Kit AA (Machine Learning), para añadir aprendizaje automático a la aplicación.

Firebase está a disposición de cualquier programador que desee utilizarlo en sus proyectos, sin necesidad de realizar ningún desembolso económico, ya que ofrece un plan totalmente gratuito y, aunque algo limitado, con acceso a la mayoría de sus funcionalidades. También dispone de otro plan en el que se paga según el uso que se les dé a sus productos.

Algunos ejemplos de aplicaciones conocidas que usan Firebase son The New York Times, Shazam, Trivago o The economist.

En los siguientes apartados se va a analizar el sistema de autenticación, la base de datos en tiempo real, los disparadores y la forma de implementarlos.

4.3.1 Autenticación

Firebase Authentication permite crear sistemas de autenticación seguros, sencillos de configurar y con múltiples métodos de identificación. Permite identificar a los usuarios mediante correo electrónico y contraseña, número de teléfono, o mediante cuenta de Google, Play Juegos, Facebook, Twitter, GitHub, Yahoo, Microsoft, Apple o de forma anónima. La seguridad durante el proceso de identificación de los usuarios está garantizada por Google.

Desde el panel de administración de Firebase se pueden habilitar y deshabilitar los métodos de inicio de sesión. También hay un apartado donde aparece el listado de usuarios registrados en la aplicación, mostrando su id (correo electrónico en el caso de la aplicación que se va a desarrollar), el modo de inicio de sesión, la fecha de la creación de la cuenta, su última conexión y su identificador único en la base de datos de cuentas de Firebase.



Identificador	Proveedores	Fecha de creación	Inicio de sesión	UID de usuario ↑
edgmyramello@gmail.com		31 may. 2020	3 jun. 2020	XXXXXXXXXXXXXXXXXXXX >

Figura 4.5: Panel con el listado de usuarios en Firebase.

En otro apartado del panel de administración, se permite a los desarrolladores habilitar y editar plantillas, las cuales la plataforma enviaría a los usuarios en caso de tener habilitadas las opciones de verificación de correo electrónico, cambio de contraseña, cambio de correo electrónico, si se tuviera un servidor propio SMTP o para verificaciones vía SMS.

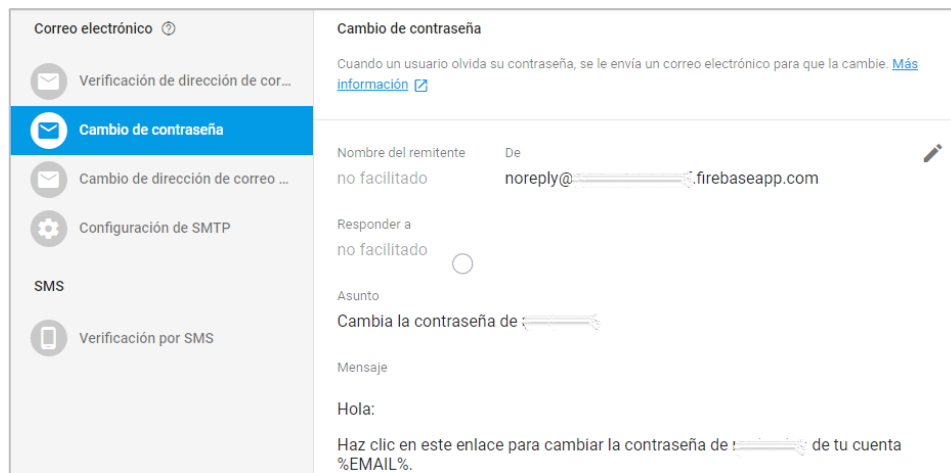


Figura 4.6: Plantillas de correo electrónico en Firebase.

En la aplicación que se ha desarrollado, se permite la opción de cambiar la contraseña, así que se ha hecho uso de la plantilla de cambio de contraseña. También se ha utilizado el servicio de autenticación mediante un correo electrónico y una contraseña para identificar a los usuarios. Para ello, se ha habilitado dicha opción en el panel de control web de Firebase.

4.3.2 Base de datos en tiempo real

La base de datos en tiempo real de Firebase permite almacenar y sincronizar los datos con los dispositivos a través de una base de datos NoSQL. La sincronización de la base de datos con los dispositivos se realiza en apenas milisegundos, y en caso de que la aplicación perdiera la conexión a Internet, Firebase garantiza que los datos siguen estando disponibles, sincronizando los cambios que hubiera cuando la aplicación recuperase la conexión a Internet. Permite crear aplicaciones sin la necesidad de utilizar servidores, ya que Firebase se encarga de ello.

La forma de estructurar los datos es en forma de objeto, el cual se va creando en forma de árbol. La aplicación recibe los datos en forma de un objeto JSON.



Figura 4.7: Estructura de la base de datos de Firebase.

Firebase dispone de unas reglas mediante las que se puede restringir el acceso a los datos de la aplicación y que pueden ser editadas. En la siguiente figura se muestra la forma en la que se definen las reglas para que solo pudieran leer y escribir en la base de datos aquellos usuarios que se han identificado mediante algún método de autenticación de Firebase.

```
1 {
2   "rules": {
3     ".read": "auth != null",
4     ".write": "auth != null",
```

Figura 4.8: Reglas de seguridad sobre los datos de Firebase.

4.3.3 Cloud functions

Las Cloud Functions de Firebase permiten procesar código de servidor sin la necesidad de disponer de un servidor propio, ya que se ejecutan en los servidores de Google.

Las funciones se programan en JavaScript y se almacenan en la nube mediante la consola de comandos ejecutando el comando `firebase deploy`.

En la aplicación desarrollada en este TFG, las Cloud Functions se han utilizado para desencadenar el envío de un correo electrónico a los usuarios, empresas o administradores, cuando se dan de alta o de baja. También para eliminar del servicio de autenticación de Firebase a aquellos usuarios que sean dados de baja de la aplicación.

Función	Activador	Región	Tiempo de ejecución	Memoria	Tiempo de espera
deleteAdminFromAu...	ref.update admins/{uid}/profile/deleted	us-central1	Node.js 10	256 MB	60s
deleteCompanyFrom...	ref.update companies/{uid}/profile/deleted	us-central1	Node.js 10	256 MB	60s
deleteUserFromAuth	ref.update users/{uid}/profile/deleted	us-central1	Node.js 10	256 MB	60s
sendByeMailAdmin	ref.update admins/{uid}/profile/deleted	us-central1	Node.js 10	256 MB	60s
sendByeMailCompany	ref.update companies/{uid}/profile/deleted	us-central1	Node.js 10	256 MB	60s
sendByeMailUser	ref.update users/{uid}/profile/deleted	us-central1	Node.js 10	256 MB	60s
sendWelcomeMailAd...	ref.create admins/{uid}	us-central1	Node.js 10	256 MB	60s
sendWelcomeMailCo...	ref.create companies/{uid}	us-central1	Node.js 10	256 MB	60s
sendWelcomeMailUs...	ref.create users/{uid}	us-central1	Node.js 10	256 MB	60s

Figura 4.9: Listado de Cloud Functions del proyecto.

4.3.4 Implementación

En cuanto a la implementación, el primer paso es crear un proyecto en Firebase. Para ello, desde la consola de Firebase (<https://console.firebase.google.com>) se añade un nuevo proyecto, indicando el nombre y la ubicación. Una vez creado el proyecto, el siguiente paso consiste en añadir Firebase a la aplicación mediante un archivo de configuración que contendrá las claves necesarias para establecer la conexión.

```
var config = {
  apiKey: "...",
  authDomain: "..." + ".firebaseapp.com",
  databaseURL: "https://" + ... + ".firebaseio.com",
  projectId: "...",
  storageBucket: "",
  messagingSenderId: "..."
};
firebase.initializeApp(config);
```

Figura 4.10: Archivo de configuración de Firebase.

Firebase ofrece una serie de métodos que, a partir de una instancia al archivo de configuración de Firebase, permiten realizar operaciones de autenticación, interacciones con la base de datos en tiempo real o gestión de archivos.

Para autenticación, el constructor al que se llama es `auth()`, el cual ofrece una serie de métodos. En la aplicación desarrollada se han utilizado:

- `createUserWithEmailAndPassword(email, password)`: para crear una cuenta asociada al correo electrónico y contraseña indicados.
- `currentUser`: información del usuario, entre ella su identificador.
- `reauthenticateWithCredential(credential)`: para verificar el correo y la contraseña actuales.
- `sendPasswordResetEmail(email, actionCodeSettings)`: para enviar un correo con las instrucciones para cambiar la contraseña.
- `signInWithEmailAndPassword(email, password)`: para iniciar sesión a través de correo electrónico y contraseña.
- `signOut()`: para cerrar sesión.
- `updatePassword(password)`: para actualizar la contraseña.

Para interacciones con la base de datos en tiempo real, el constructor al que se llama es `database()`. En la aplicación se han utilizado los siguientes métodos:

- `on(eventType, callback)`: para consultas en las que se quiera tener constancia de los cambios y en función del tipo de evento que se especifique, realizar una acción u otra. Este método es el que posibilita la sincronización entre la base de datos y la aplicación.
- `push(value)`: para añadir un nuevo elemento a la base de datos al que se le asigna un identificador único aleatorio.
- `set(value)`: para escribir datos.
- `ref(location)`: para obtener referencia a la localización que se indique.
- `update(values)`: para actualizar datos.

Para almacenar ficheros o imágenes, Firebase proporciona el constructor `storage()`. Se ha utilizado para subir imágenes y PDF desde la aplicación.

En cuanto a las Cloud Functions, se crea una instancia de `functions` y en la aplicación se ha combinado con el método `database` junto con `onCreate()` u `onUpdate()`, para desencadenar un disparador cuando se lleva a cabo una determinada acción en la base de datos. Por ejemplo, un caso de uso sería cuando

se crea un nuevo usuario en la base de datos, lo que desencadena una función que envía un correo electrónico.

```
sendWelcomeMailUser=functions.database.ref('/users/{uid}').onCreate()
```

Todos los métodos descritos retornan códigos de error en caso de fallo. En la aplicación desarrollada en este TFG se tienen en cuenta y se muestran mensajes personalizados de aviso en caso de que sucedan.

Hay que destacar que Firebase ofrece una amplísima documentación, lo que ayuda a que el tiempo de aprendizaje sea más reducido. No ha hecho falta emplear mucho tiempo en su aprendizaje, ya que en apenas minutos se puede obtener de su documentación las respuestas a las dudas, sobre cada uno de sus productos, que van surgiendo durante el desarrollo.

4.4 Recursos y componentes React Native

Todas las librerías que se van a describir a continuación cuentan con su propia documentación, gracias a la cual resulta muy sencillo integrarlas en el proyecto. La dificultad y el tiempo han sido mínimos, salvo un par de ellas que se detalla en sus respectivos apartados.

4.4.1 React Navigation

Nació por parte de la comunidad de desarrolladores que utilizan React Native y que se veían en la necesidad de encontrar una solución sencilla a la complejidad de crear la navegación de las aplicaciones móviles. Esta librería ofrece las herramientas para crear la navegación a través de JavaScript. Se ha utilizado su versión 4, aunque en las últimas semanas han lanzado su versión 5. [10]

La navegación se construye en torno al componente `createStackNavigator`, el cual provee la manera de realizar las transiciones entre las diferentes pantallas de la aplicación. El componente encola y desencola la ruta de pantallas por las que navega el usuario y proporciona animaciones entre el cambio de pantallas.

A continuación, se muestra como es la configuración y su estructura en forma de objeto. A cada ruta se le asigna un nombre y se indica la pantalla a la que hace referencia. Opcionalmente se podrá indicar por ejemplo el título que aparecerá en el encabezado, el color de fondo, el color de las letras o indicar si se omitirá el encabezado.

```
const LoginStack = createStackNavigator({
  Welcome: {
    screen: WelcomeScreen,
    navigationOptions: {}
  },
  LoginUser: {
    screen: LoginUserScreen,
    navigationOptions: {}
  },
})
```

Otro componente que se ha utilizado ha sido `createBottomTabNavigator`. Sirve para añadir un menú navegacional en la parte inferior de la pantalla. La forma de implementarlo es también en forma de objeto JavaScript. Se puede combinar con `createStackNavigator` para conseguir con `createBottomTabNavigator` un menú, y que en cada apartado haya un `createStackNavigator` con las diferentes pantallas de cada apartado.

En el siguiente código se muestra la forma en la que crea el menú inferior de la aplicación. Se indican las rutas que forman parte de la navegación de cada pestaña y también se establecen una serie de opciones para personalizar la apariencia del menú.

```

const UserTabs = createBottomTabNavigator({
  UserJobOffers: { screen: UserJOStack },
  UserRegisteredJobOffers: { screen: UserRegisteredJOStack },
  UserChatsList: { screen: UserChatsListStack }
},{
  tabBarOptions: {
    activeBackgroundColor: '#FFF',
    activeTintColor: '#3973A1',
    inactiveBackgroundColor: '#3973A1',
    inactiveTintColor: '#FFF'
  }
});

```

La forma de navegar entre las diferentes pantallas se consigue utilizando el método `navigate(stack)`. Este método se encuentra dentro de la propiedad `navigation`, la cual es pasada en cada pantalla por el propio `createStackNavigator`.

La forma de pasar parámetros se realiza a través del método `navigate(stack, { params })`, pasándolos como segundo parámetro y en forma de objeto. Para leer los parámetros en la siguiente pantalla, se utiliza la instrucción `this.props.navigation.state.params`.

Otro método utilizado para la navegación es `goBack()`, el cual retorna la navegación a la pantalla anterior.

Su integración con React Native se consigue creando un archivo `Router.js` donde se definen todos los `createStackNavigator` y el `createBottomTabNavigator`, e importándolo en el archivo principal `App.js` de la aplicación.

```

import RouterComponent from './Router';

export default class App extends Component{
  render() {
    return(<RouterComponent />);
  }
}

```

El resultado es, por un lado, el habitual menú de navegación con el nombre de las diferentes pantallas gracias al componente `createBottomTabNavigator`, y por otro, en cada una de ellas el componente `createStackNavigator` con el que se profundiza en cada uno de los apartados de la aplicación.

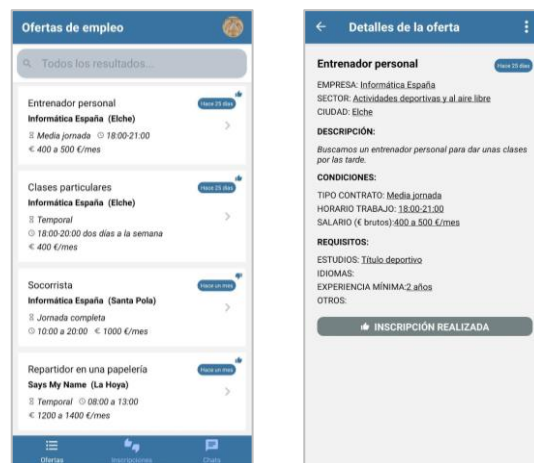


Figura 4.11: Ejemplo del menú y la navegación con React Navigation.

Además, para el flujo de autenticación se ha utilizado el componente `createSwitchNavigator`, el cual permite diferenciar rutas independientes para el sistema de registro e inicio de sesión y cada uno de los roles de la aplicación.

```
const AppNavigator = createSwitchNavigator({
  Login: LoginStack,
  User: UserTabs,
  Company: CompanyTabs,
  Admin: AdminTabs,
}, {
  initialRouteName: 'Login'
});
```

Al cambiar de ruta se reinicia la navegación, de modo que, si el usuario inicia sesión y cambia de la ruta de *Login* a la de *User*, al utilizar el botón de atrás físico o de la aplicación, nunca podría cambiar de ruta. En todo caso saldría de la aplicación.

La dificultad y tiempo empleado en su aprendizaje es medio. Cabe destacar que una vez se entiende su funcionamiento, su implementación es muy sencilla. Hay que resaltar que inicialmente se utilizó otra versión con bastantes errores, y que posteriormente fue corregida por los desarrolladores de la librería, y en la actualidad es bastante estable.

4.4.2 React Native Elements

Esta librería proporciona un kit de interfaces de usuario con componentes ya diseñados y listos para integrar en proyectos de React Native. En su completa documentación informa de las propiedades de cada componente. [11]

En la aplicación desarrollada se han utilizado los componentes `Button`, `ButtonGroup`, `Icon`, `Input`, `Listitem`, `SearchBar` y `Tile` de esta librería.

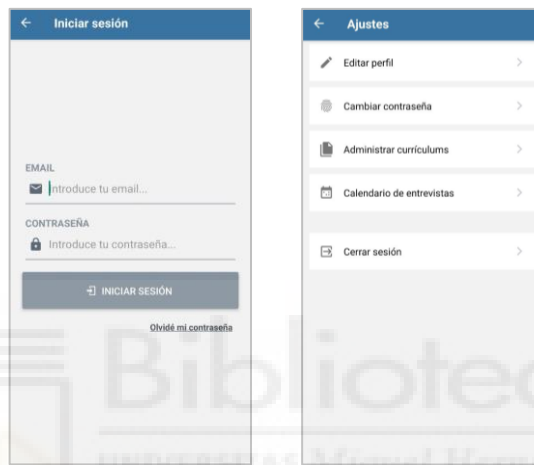


Figura 4.12: Componentes de React Native Elements: Input, Button, ListItem.

4.4.3 React Native Gifted Chat

Ofrece un interfaz de usuario para chats [12]. Al igual que el resto de las librerías y repositorios, su instalación se realiza mediante npm [13], un gestor de paquetes para Node JS.

Los mensajes se procesan a través de objetos. Cada mensaje debe tener un identificador único, un texto, la fecha de creación y un objeto con los datos del usuario que envía el mensaje.

```
{
  _id: -LCdfw7GbnU_Tt11K9wU
  createdAt: "2018-07-26T10:20:00Z"
  text: "Hola"
  user: { _id: "QhjZ2xZv8jhqNB9WRqjqjiDVztx2" }
}
```


En la aplicación desarrollada se ha utilizado para el chat entre los usuarios y las empresas.



Figura 4.13: Chat de la aplicación.

4.4.4 React Native Calendars

Esta librería ofrece una serie de componentes visuales para mostrar fechas en formato de calendario o agenda. Es altamente personalizable y permite marcar fechas o periodos asignándoles eventos. [14]

Su integración en la aplicación ha sido en el apartado del Calendario de entrevistas, dónde los usuarios y las empresas pueden ver de forma ordenada las fechas en las que tienen planificadas las entrevistas. Todo gracias al componente Agenda.

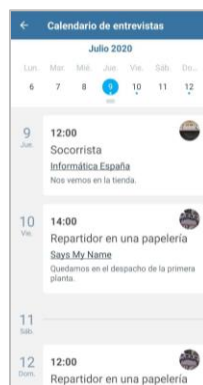


Figura 4.14: Agenda de la aplicación.

4.4.5 React Native DatePicker

Se trata de un componente con una interfaz que muestra un calendario para que el usuario seleccione una fecha. Se ha utilizado en los formularios en los que hay que indicar una fecha.



Figura 4.15: Input de tipo Calendario.

4.4.6 React Native DocumentPicker

Esta librería permite especificar qué tipo de documento seleccionar. Se ha utilizado en el apartado de los currículums de los usuarios, para seleccionar un documento de tipo PDF desde los archivos del dispositivo.



Figura 4.16: Seleccionador de PDF.

4.4.7 React Native DropdownAlert

Este componente permite mostrar mensajes que se despliegan en la parte superior de la pantalla por unos segundos. En la aplicación desarrollada se muestra siempre como mensaje de éxito cuando se crea, edita o elimina cualquier elemento.



Figura 4.17: Mensaje de éxito.

4.4.8 React Native ImagePicker

Esta librería ofrece una interfaz para seleccionar imágenes desde el dispositivo. También permite establecer un peso máximo de la imagen, de modo que impida cargar imágenes muy pesadas. Se ha utilizado para seleccionar las imágenes de los perfiles.

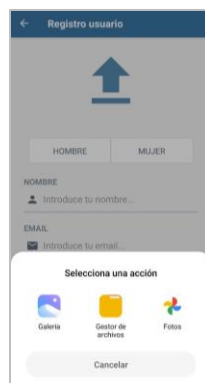


Figura 4.18: Seleccionador de imágenes.

4.4.9 React Native Indicators

Ofrece una serie de *spinners* ya preestablecidos. En la aplicación se ha utilizado el `UIActivityIndicator`, que se muestra siempre que se está cargando contenido o cuando se está esperando que se procese alguna acción.

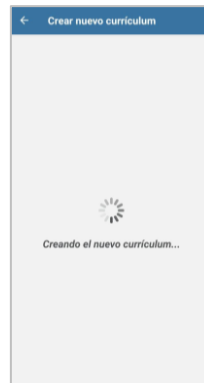


Figura 4.19: Ejemplo de uso de React Native Indicators.

4.4.10 React Native PDF

Es un componente que sirve para mostrar archivos PDF. Permite hacer zoom. Se ha utilizado para mostrar los currículums de los usuarios.

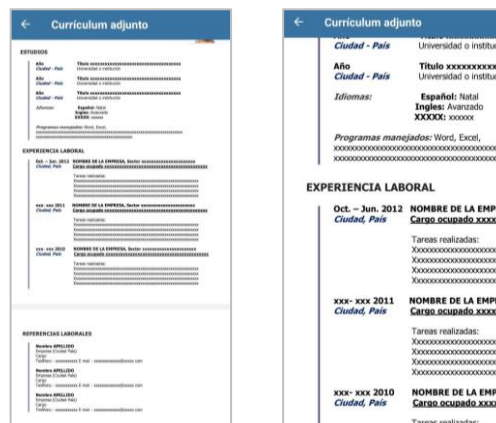


Figura 4.20: Visor de archivos PDF.

4.4.11 React Native Vector Icons

Esta librería contiene una gran cantidad de iconos que se han utilizado para darle mayor personalidad a la interfaz de la aplicación. Se muestran en botones, listas, menús y formularios.

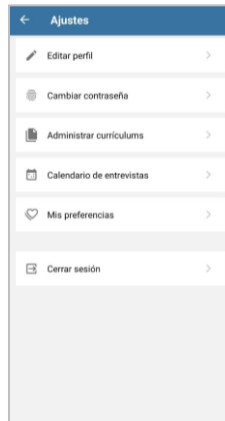


Figura 4.21: Algunos iconos de la aplicación.

4.4.12 React Native HTML to PDF

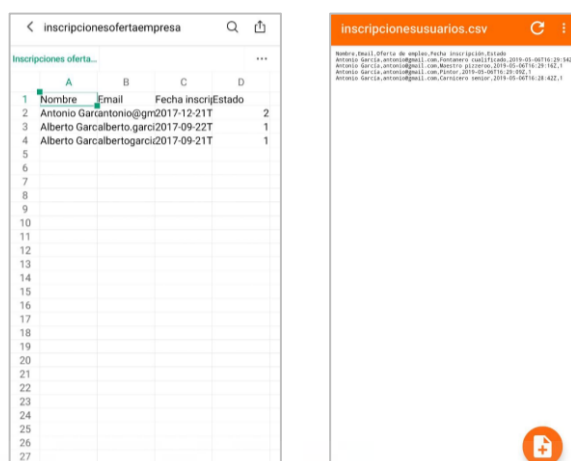
Permite crear documentos PDF mediante el uso de HTML. Se ha utilizado para exportar listados en formato PDF.



Figura 4.22: Listado exportado en formato PDF.

4.4.13 XLSX

Permite crear documentos en diferentes formatos de archivo. En la aplicación se ha utilizado para crear documentos en formatos XLS y CSV. Al igual que la anterior librería, también se ha empleado para exportar listados, en este caso en formato Excel y CSV.



	A	B	C	D	
1	Nombre	Email	Fecha inscri	Estado	
2	Antonio Garca	antonio@gn	2017-12-21T		2
3	Alberto Garca	alberto.garc	2017-09-22T		1
4	Alberto Garca	albertogarci	2017-09-21T		1
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					

```
inscripcionesusuarios.csv
Nombre,Email,Fecha inscri,Estado
Antonio Garca,antonio@gn,2017-12-21T,2
Alberto Garca,alberto.garc,2017-09-22T,1
Alberto Garca,albertogarci,2017-09-21T,1
```

Figura 4.23: Listados exportados en formatos Excel y CSV.

4.5 Librerías JavaScript

A continuación, se indican tres librerías JavaScript que han servido de ayuda para tratar con arrays, fechas y para enviar correos electrónicos.

4.5.1 Lodash

Ofrece multitud de funciones para el manejo de arrays y objetos JavaScript. Se caracteriza por la forma en la que se llama a sus métodos mediante `_.` [15] A lo largo del desarrollo de la aplicación se han empleado los siguientes métodos:

- `_.filter(array, función)`: recorre el array y retorna los valores que cumplen con la condición que se indica en la función.
- `_.forEach(objeto, función)`: para cada elemento del array permite realizar la acción que se indica en la función.
- `_.has(objeto, prop)`: comprueba si el objeto contiene una propiedad.

- `_.head(array)`: retorna el primer elemento del array.
- `_.includes(array, valor)`: comprueba si el array contiene un valor.
- `_.intersection(arrays)`: crea un array formado por los valores coincidentes en todos los arrays que recibe como parámetro.
- `_.isEmpty(array)`: comprueba si el array está vacío.
- `_.isNil(valor)`: comprueba si un valor es nulo.
- `_.last(array)`: retorna el último elemento del array.
- `_.map(array, función)`: crea un array retornando una serie de valores, que se indican en la función, para cada elemento del array que recibe.
- `_.orderBy(array, prop, orden)`: retorna un array ordenado según la propiedad indicada y en orden ascendente o descendente.
- `_.values(objeto)`: crea un array a partir del objeto que recibe.
- `_.words(texto)`: crea un array con las palabras que recibe como texto.

4.5.2 Moment JS

Es una librería que se ha empleado para el tratamiento de fechas. Los métodos que se han utilizado en la aplicación han sido: [16]

- `add(n, unidad)`: añade n unidades a la fecha original.
- `a.diff(b, unidad)`: retorna la diferencia entre la fecha a y b en la unidad de medida indicada (segundos, minutos, horas, días, meses, años).
- `format(formato)`: personaliza el formato de la fecha.
- `fromNow(fecha)`: devuelve el tiempo transcurrido desde la fecha indicada hasta el día de hoy.
- `a.isBefore(b)`: comprueba si la fecha a es anterior a la fecha b .
- `a.isBetween(b, c)`: comprueba si la fecha a se encuentra entre b y c .
- `moment()`: obtiene la fecha y hora actual.
- `month()`: devuelve el mes de una fecha.
- `subtract(n, unidad)`: resta n unidades a la fecha original.
- `year()`: devuelve el año de una fecha.

4.5.3 Nodemailer

Es un módulo de Node JS que permite enviar correos electrónicos. Su implementación es muy sencilla. Primero se necesita un objeto transportador el cual será el encargado de enviar el correo, y se consigue mediante el método `createTransport(config)`, el cual recibe un objeto con la configuración del host y la forma de autenticación. Por último con el método `sendMail(mensaje, callback)` se envía el correo, que recibe como parámetros el mensaje a enviar y opcionalmente una función callback que retorna éxito o error según se haya completado o no el envío.

```
var transporter = nodemailer.createTransport({
  host: 'smtp.gmail.com',
  auth: {
    type: 'login',
    user: mail,
    pass: password
  }
});
transporter.sendMail(mensaje, callback);
```

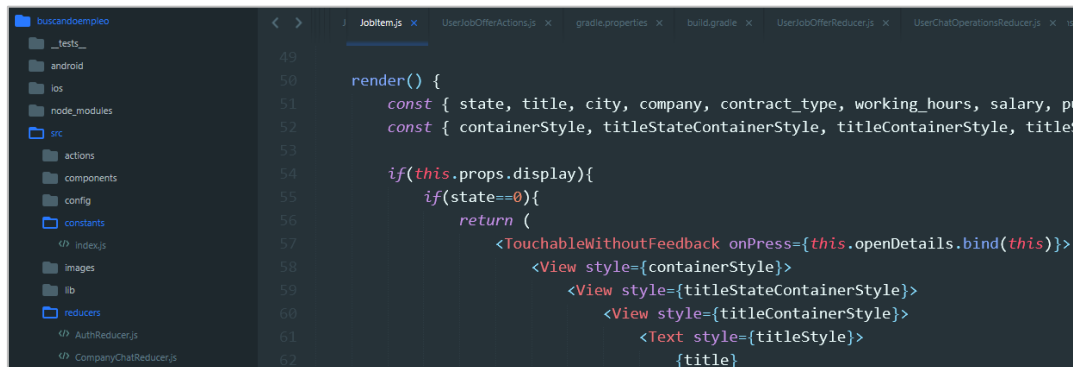
Su uso en la aplicación ha sido para enviar correos electrónicos de bienvenida y despedida a los usuarios que se registran o cuya cuenta es eliminada.

4.6 Herramientas software

4.6.1 Sublime Text

Se trata de un editor de código multiplataforma, que soporta gran cantidad de lenguajes de programación, permite tener múltiples ventanas abiertas a la vez y dispone de un mini mapa donde ver una previsualización de todo el fichero. Otra característica destacable es la multi selección, de modo que se puede editar diferentes partes del código simultáneamente. También dispone de un buscador de sintaxis para localizar trozos de código o para reemplazarlos. [17]

Se ha utilizado para desarrollar todo el código de la aplicación. Todo ello con la versión de prueba gratuita, que dispone de una funcionalidad completa y por un periodo de duración de por vida.



```
render() {
  const { state, title, city, company, contract_type, working hours, salary, pu
  const { containerStyle, titleStateContainerStyle, titleContainerStyle, titles

  if(this.props.display){
    if(state==0){
      return (
        <TouchableWithoutFeedback onPress={this.openDetails.bind(this)}>
        <View style={containerStyle}>
        <View style={titleStateContainerStyle}>
        <View style={titleContainerStyle}>
        <Text style={titleStyle}>
        {title}
      )
    }
  }
}
```

Figura 4.24: Interfaz de Sublime Text 3.

Es altamente personalizable, tanto estéticamente como a nivel de paquetes. Algunos de los paquetes que se pueden instalar, permiten cambiar el marcado del código fuente. Otros permiten actualizar y sincronizar de local a remoto desde el propio editor.

4.6.2 GitHub

GitHub es un controlador de versiones. Ofrece un servicio gratuito de repositorios públicos para registrar los cambios que van surgiendo en los archivos con los que se trabaja. Controla tanto los cambios en las diferentes versiones del desarrollo, como la fecha en la que tienen lugar y su descripción. [18]

Es importante hacer uso de un sistema de control de versiones cuando se va a desarrollar un proyecto, ya que es una forma registrar los cambios que se producen, lo que permite en un futuro detectar rápidamente cambios anteriores y, en caso de desearse, poder recuperar el estado anterior del proyecto fácilmente.

Otra de sus características es que permite trabajar de forma colaborativa, ya que varios desarrolladores pueden estar trabajando en su versión de la aplicación, y gracias a GitHub, pueden combinar ambas versiones y administrar los cambios.

The screenshot shows a GitHub commit interface. At the top, it says 'usuarios: eliminar chats' and 'master'. Below that, it shows the commit was made by 'agarciaguillo' on '12 Jul'. The commit message is 'usuarios: eliminar chats'. The diff shows changes to the file 'src/actions/UserChatActions.js'. The diff is in unified diff format, showing line-by-line changes. Line 1 has a '+' sign and 'import _ from 'lodash';'. Line 2 has a '+' sign and 'import firebase from '../config/firebase;'. Line 3 has a '+' sign and 'import {' followed by 'CHATS_USER_INPUT_CHANGED,' on line 4. Line 5 has a '+' sign and 'RESET_USER_CHAT_STATE,'.

Figura 4.25: Actualización del proyecto almacenado en GitHub.

Durante el desarrollo de la aplicación, se ha empleado para ir creando copias de seguridad del código y documentar los cambios en el desarrollo. En cada versión del código, GitHub registra las inserciones y borrados de cada archivo de la aplicación, y qué ficheros se añaden y cuáles se eliminan.

4.6.3 Microsoft Word

Forma parte del paquete Office de Windows. Es una herramienta que permite dar formato a texto y también incluir tablas e imágenes. Venía instalado en el equipo donde se ha desarrollado el proyecto. Se ha empleado para la redacción de la presente memoria del TFG. [19]

4.6.4 Balsamiq Mockups

Se trata de una herramienta que proporciona una interfaz gráfica con la que se ha creado el prototipo de la aplicación. Se ha utilizado la versión de prueba gratuita. [20]

Ofrece la posibilidad de diseñar un prototipo bastante completo en el que mostrar un aspecto similar al que se quiere posteriormente desarrollar. Dispone de multitud de componentes tales como pantallas, menús, botones, imágenes, texto, checkbox, diferentes formas con las que simular componentes y también permite enlazar entre diferentes pantallas para mostrar cómo será la navegación de la aplicación final.

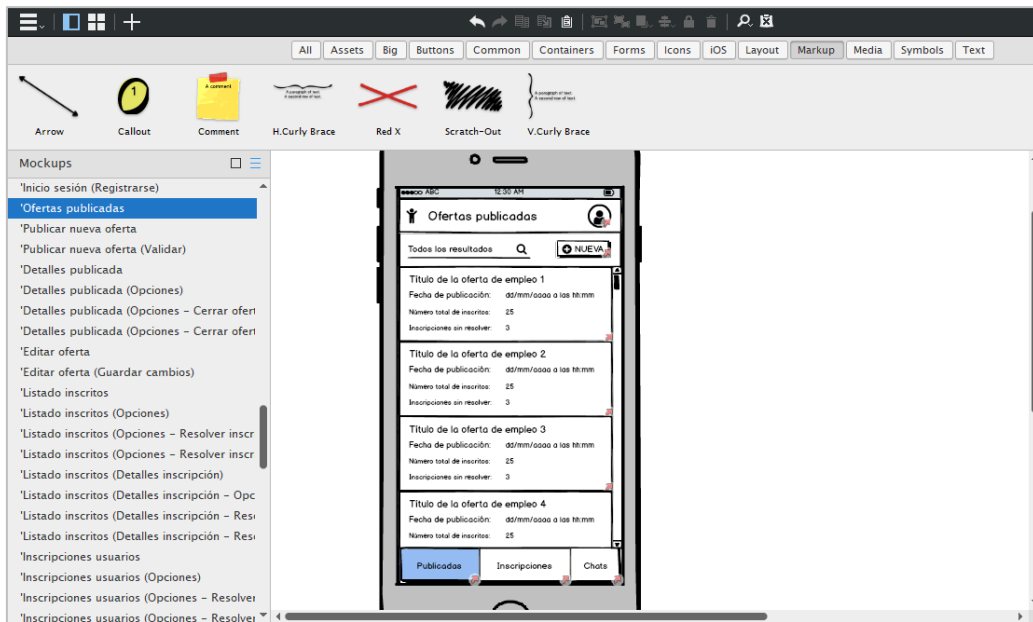


Figura 4.26: Interfaz de Balsamiq Mockups.

4.6.5 GanttProject

Es una aplicación de escritorio gratuita con la que realizar diagramas de Gantt. Se ha utilizado para la planificación del proyecto. [21]

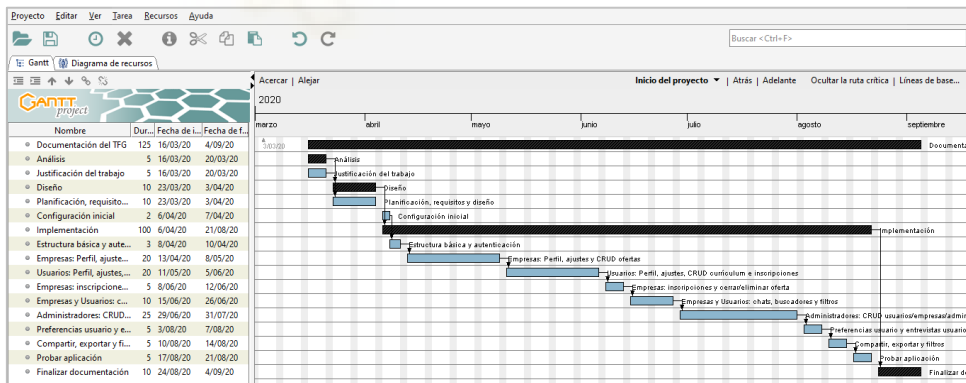


Figura 4.27: Interfaz de GanttProject.

4.6.6 Draw.io

Se trata de una herramienta que sirve para dibujar diagramas. Se ha utilizado su versión web gratuita. Permite el autoguardado en Google Drive. [22]

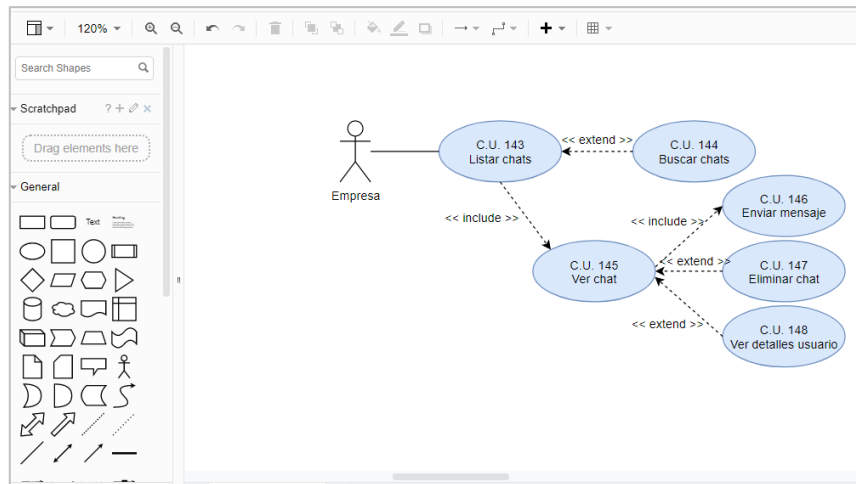


Figura 4.28: Interfaz de Draw.io.

4.7 Lenguajes de programación

4.7.1 JavaScript

Es el lenguaje de programación con el que se construyen las aplicaciones de React Native. Utiliza ES2015, también conocido ECMAScript 6, la cual introduce una serie de mejoras en JavaScript.

En el siguiente código se muestran algunas de las novedades de ES2015 que se emplean en React Native, como son la forma de importar otros ficheros, las definiciones de clases, la forma de definir funciones o nuevos tipos de variables.

```
import React, { Component } from 'react';

export default class Item extends Component {
  let i = 0;
}
```

4.7.2 JSX

Es una sintaxis para embeber código XML dentro de código JavaScript. A diferencia de otros frameworks, React Native introduce el lenguaje de marcado dentro del código. JSX produce componentes de React que proporcionan al desarrollador un lenguaje con el que crear sus propios componentes.

Existen multitud de componentes. Por ejemplo, el componente `View` permite crear un contenedor, mientras que el componente `Text` muestra solo texto.

```
<View>
  <Text>¡Te has inscrito correctamente!</Text>
</View>
```

4.8 Herramientas hardware

4.8.1 Acer Aspire 5750G

Para el análisis y desarrollo de la aplicación se ha utilizado un portátil personal de la marca Acer. Con él se han realizado todas las fases de las que se compone un proyecto de ingeniería del software, además del desarrollo de la aplicación móvil. Las características del equipo empleado son:

- Marca: Acer
- Modelo: Aspire 5750G
- Procesador: Intel Core i5 2450M CPU 2.50GHz
- Tarjeta gráfica: NVIDIA GeForce GT 630M
- Memoria RAM: 4GB
- Disco duro: 500GB
- Sistema operativo: Windows 10 Home
- Pantalla: 15,6" HD 1366x768 px
- Precio 500€ (Año 2012)

4.8.2 Xiaomi Redmi Note 8 Pro

Para ejecutar la aplicación e ir viendo los avances, en lugar de hacer uso de un emulador, se ha optado por un dispositivo físico. En él, y mediante una conexión USB, se ha ido ejecutando y comprobando los cambios que se han ido introduciendo a medida que iba avanzando el desarrollo de la aplicación. Las características del móvil son:

- Marca: Xiaomi
- Modelo: Redmi Note 8 Pro Global Version
- Procesador: MediaTek Helio G90T 2x2.05GHz ARM Cortex•A76 + 6x1.95 GHz ARM Cortex•A55
- Tarjeta gráfica: Arm Mali-G76 3EEMC4 800MHz
- RAM/ROM: 6GB/128GB
- Sistema operativo: MIUI V12 (Android 10 Q)
- Pantalla: 6.53" LTPS LCD 1080x2340px - 396ppi
- Precio 180€ (Año 2020)



CAPÍTULO 5 - METODOLOGÍA Y RESULTADOS

5.1 Planificación del trabajo

5.1.1 Ciclo de vida

En los siguientes apartados se va a documentar todo el desarrollo de la aplicación de búsqueda de empleo. Como cualquier proyecto de ingeniería del software, la forma de planificarlo y estructurarlo es siguiendo las fases de algún modelo de ciclo de vida del software, el cual explique el proceso de qué, cuándo, quién y cómo se debe hacer. Algunos tipos de ciclos de vida del software son: [25] [26]

- Lineal. Cada etapa del proyecto se realiza de manera lineal y una sola vez, cuando se termina una, se pasa a la siguiente.
- En cascada. Después de cada etapa se realiza una revisión para verificar si se pasa a la siguiente.
- Iterativo incremental. El software se va desarrollando por módulos y tras cada etapa, se hacen pruebas y el producto va adquiriendo nueva funcionalidad. Útil cuando se prevén nuevos requisitos a mitad del desarrollo.
- Construcción de prototipos. Se parte de especificaciones incompletas, y a medida que se avanza el desarrollo, se va ampliando el prototipo. Suele utilizarse cuando se va a utilizar tecnologías poco conocidas. Altamente costoso.

Para el desarrollo de la aplicación se ha decidido seguir el ciclo de vida iterativo incremental, por lo que se ha ido haciendo de forma escalonada y siguiendo una serie de iteraciones en las que se han establecido unos requerimientos para cada una de ellas. Podría decirse que también cuenta con características del modelo de prototipos, ya que desde un primer momento se ha ido creando, poco a poco en cada iteración, un prototipo más completo, con alta funcionalidad y navegación entre pantallas. En total se han definido doce iteraciones que se detallan a continuación:

ITERACIÓN 1 (Justificación del trabajo):

- Justificar importancia.
- Objetivos.
- Estudiar tecnologías para desarrollar aplicaciones móviles multiplataforma.

ITERACIÓN 2 (Planificación, requisitos y diseño):

- Planificar desarrollo de la aplicación.
- Capturar requisitos.
- Diseñar la estructura de la aplicación mediante prototipo.
- Diseñar la base de datos.

ITERACIÓN 3 (Configuración inicial):

- Configurar Firebase.
- Crear nuevo proyecto de React Native.

ITERACIÓN 4 (Estructura básica y autenticación):

- Crear estructura básica de pantallas y navegación de la aplicación.
- Configurar registro e inicio de sesión en la aplicación mediante Firebase.

ITERACIÓN 5 (Empresas: Perfil, ajustes y CRUD* ofertas):

- Ver perfil.
- Editar perfil.
- Cambiar contraseña.
- Cerrar sesión.
- Publicar oferta de empleo.
- Listar ofertas publicadas.
- Ver detalles oferta.
- Editar oferta.

*CRUD: siglas en inglés de Crear, Leer, Actualizar y Eliminar (Create, Read, Update y Delete).

ITERACIÓN 6 (Usuarios: Perfil, ajustes, CRUD currículum e inscripciones):

- Ver perfil.
- Editar perfil.
- Cambiar contraseña.
- Crear currículum.
- Listar currículums.
- Ver detalles currículum.
- Editar currículum.
- Eliminar currículum.
- Cerrar sesión.
- Listar ofertas de empleo disponibles.
- Ver detalles oferta.
- Inscribir en oferta.
- Listar inscripciones.
- Ver detalles inscripción.
- Cancelar inscripción.

ITERACIÓN 7 (Empresas: inscripciones y cerrar/eliminar oferta):

- Listar inscritos en oferta (en detalles oferta publicada y en apartado de todas las inscripciones).
- Ver detalles inscripciones.
- Resolver inscripciones.
- Cerrar oferta.
- Eliminar oferta.

ITERACIÓN 8 (Empresas y Usuarios: chats, buscadores y filtros):

- Iniciar chat.
- Listar chats.
- Enviar mensajes.
- Eliminar chat.
- Ver detalles usuario/empresa.
- Buscadores y filtros desplegados en listados.

ITERACIÓN 9 (Administradores: CRUD usuarios/empresas/administradores):

- Ver perfil.
- Editar perfil.
- Cambiar contraseña.
- Cerrar sesión.
- CRUD administradores.
- CRUD usuarios.
- CRUD currículums del usuario.
- Listar, ver detalles y resolver inscripciones del usuario.
- CRUD empresas.
- CRUD ofertas de la empresa.
- Listar, ver detalles y resolver inscripciones en ofertas de la empresa.
- Buscadores y filtros desplegados en listados.

ITERACIÓN 10 (Preferencias usuario y entrevistas usuarios/empresas):

- Editar preferencias (usuarios).
- Calendario de entrevistas (usuarios y empresas).

ITERACIÓN 11 (Compartir, exportar y filtros):

- Compartir ofertas por redes sociales (usuarios).
- Exportar listados en CSV/Excel/PDF (administradores y empresas).
- Añadir filtro para búsqueda de ofertas (usuarios).
- Mejora del filtro de búsqueda de usuarios (empresas).
- Editar entrevistas (empresas).

ITERACIÓN 12 (Probar aplicación):

- Probar a fondo la aplicación.
- Solucionar posibles errores.
- Generar APK final.

5.1.2 Diagrama de Gantt

La forma de representar el tiempo que se ha destinado a cada fase en el desarrollo del trabajo ha sido mediante un diagrama de Gantt. El periodo total de realización del TFG ha sido de alrededor de 6 meses, dedicando de media unas 4 horas diarias de lunes a viernes.

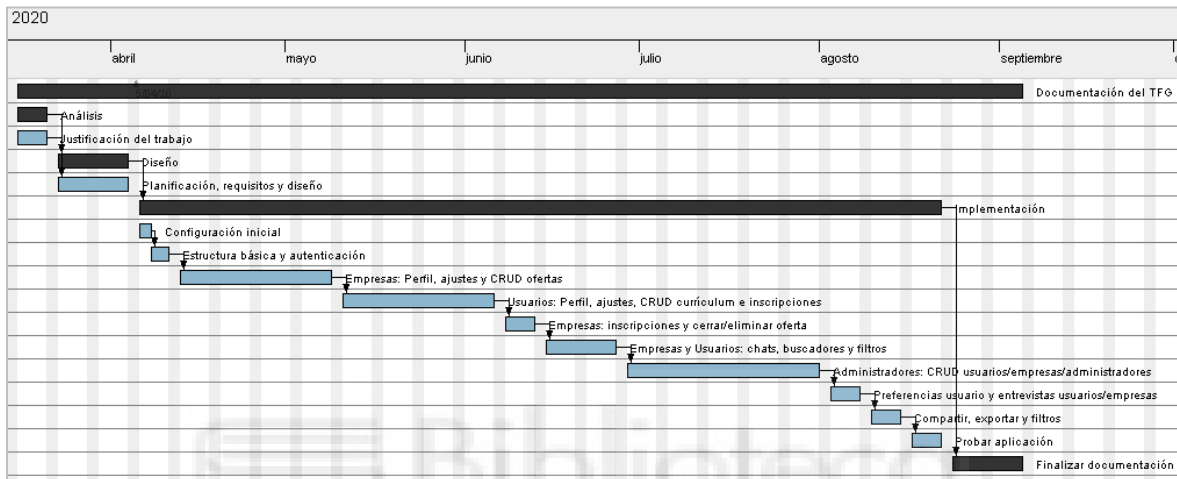


Figura 5.1: Diagrama de Gantt.

Nombre	Duración	Fecha de inicio	Fecha de fin
• Documentación del TFG	125	16/03/20	4/09/20
• Análisis	5	16/03/20	20/03/20
• Justificación del trabajo	5	16/03/20	20/03/20
• Diseño	10	23/03/20	3/04/20
• Planificación, requisitos y diseño	10	23/03/20	3/04/20
• Implementación	100	6/04/20	21/08/20
• Configuración inicial	2	6/04/20	7/04/20
• Estructura básica y autenticación	3	8/04/20	10/04/20
• Empresas: Perfil, ajustes y CRUD ofertas	20	13/04/20	8/05/20
• Usuarios: Perfil, ajustes, CRUD currículum e inscripciones	20	11/05/20	5/06/20
• Empresas: inscripciones y cerrar/eliminar oferta	5	8/06/20	12/06/20
• Empresas y Usuarios: chats, buscadores y filtros	10	15/06/20	26/06/20
• Administradores: CRUD usuarios/empresas/administradores	25	29/06/20	31/07/20
• Preferencias usuario y entrevistas usuarios/empresas	5	3/08/20	7/08/20
• Compartir, exportar y filtros	5	10/08/20	14/08/20
• Probar aplicación	5	17/08/20	21/08/20
• Finalizar documentación	10	24/08/20	4/09/20

Figura 5.2: Duración y fechas de las tareas.

5.2 Análisis de requisitos

5.2.1 Captura de requisitos y prototipo

En todo proyecto de ingeniería del software es importante definir los requisitos y también diseñar un prototipo de la aplicación que se va a desarrollar. Junto con la captura de requisitos, se ha optado por realizar un prototipo de alta fidelidad mediante el software Balsamiq Mockups. Se ha invertido mayor tiempo en su elaboración que si se hubiese optado por realizar uno de baja fidelidad, pero se ha tomado la decisión para ahorrar tiempo durante el desarrollo y tener plasmado desde el principio gran cantidad de detalles de la aplicación en cada una de las pantallas.

La aplicación desarrollada está dividida en cuatro apartados: Inicio de sesión/registro, Usuarios, Empresas y Administradores. Todos los actores comparten el apartado de Inicio de sesión/registro, y según su rol, son dirigidos a su correspondiente apartado una vez se autentiquen correctamente. El resto de los apartados de los otros actores son inaccesibles para ellos.

A continuación, se indican cuáles son los requisitos, junto con capturas del prototipo, para cada uno de los apartados de la aplicación.

INICIO SESIÓN/REGISTRO:

La primera vez que se abre la aplicación, el cliente accede al apartado de Inicio de sesión y Registro. En esa pantalla debe elegir si desea acceder como usuario, empresa o administrador.

Los usuarios y las empresas pueden iniciar sesión o registrarse. Los administradores solo pueden iniciar sesión, ya que el registro de los nuevos

administradores corresponde a la figura del superadministrador, que tiene las mismas funcionalidades que los administradores, pero además puede crear y editar administradores. Todos ellos, para iniciar sesión tienen que rellenar un formulario con su correo electrónico y su contraseña. Hay un apartado para recuperar su contraseña, en el cual pueden introducir su correo electrónico y les llega un mensaje con las instrucciones necesarias para recuperarla.

El formulario de registro de los usuarios consta de los campos imagen, sexo, nombre, email, fecha de nacimiento, teléfono, contraseña y repetir contraseña. Todos los campos son obligatorios salvo la imagen.

Por su parte, el formulario de registro de las empresas está formado por imagen, nombre, descripción, email, teléfono, dirección, ciudad, contraseña y repetir contraseña. Salvo la imagen, el resto son obligatorios.



Figura 5.3: Prototipo Inicio sesión y Registro.

En todos los formularios de inicio de sesión, registro y recuperación de contraseña hay códigos de error en caso de que se introduzcan mal los datos o se deje sin rellenar algún campo obligatorio.

Cuando los usuarios o las empresas se registran en la aplicación, o cuando algún administrador les da de alta, reciben un correo electrónico de bienvenida a la aplicación.

Si inician sesión o completan con éxito el registro, acceden a su apartado correspondiente. Si cierran la aplicación, al volver a abrirla, no deben volver a iniciar sesión, sino que acceden directamente a su apartado.

La estructura de las pantallas para los tres actores es la misma. En la parte superior está el título de la pantalla junto con el acceso al perfil en la esquina derecha, y en la parte inferior hay un menú para moverse entre los diferentes subapartados.

USUARIOS:

El apartado de los usuarios está formado por cuatro subapartados: perfil, listado de ofertas de empleo, inscripciones realizadas y chats con empresas. Al perfil, como se ha comentado en el párrafo anterior, se accede desde el encabezado, mientras que los otros tres subapartados forman parte del menú de la parte inferior.

La pantalla con las ofertas de empleo disponibles se muestra por defecto cuando abren la aplicación o cuando se autentican correctamente. Se muestra un listado con ofertas de empleo ordenadas por fecha de publicación reciente. De cada una de ellas se muestra el título, el nombre de la empresa, la ciudad, el tipo de contrato, el horario y el salario. También se muestra un icono que indica el estado de su inscripción en caso de que el usuario se haya registrado previamente. En la parte superior hay un buscador y un filtro, para que pueda filtrar según el sector, ciudad, tipo de contrato, salario y experiencia requerida en cada oferta de empleo. Pinchando sobre algún elemento de la lista se abre una nueva pantalla con los detalles de la oferta de empleo.



Figura 5.4: Prototipo listado ofertas de empleo y filtro.

En la pantalla con los detalles de la oferta de empleo se muestra el título, nombre de la empresa, sector, ciudad, descripción, tipo de contrato, horario de trabajo, salario, estudios, idiomas, experiencia mínima y otros. Hay un botón para que los usuarios se puedan inscribir. Para completar la inscripción, deben seleccionar uno de sus currículums previamente creados (se explicará más adelante). Si el usuario ya está inscrito, habrá un mensaje que le alertará de ello. Pueden compartir la oferta de empleo a través de redes sociales tocando en el icono del encabezado. También pueden acceder a los detalles de la empresa o iniciar un chat con ella (ambos se detallarán más adelante).



Figura 5.5: Prototipo detalles oferta, compartir, inscribirse y cancelar inscripción.

La pantalla con las inscripciones del usuario muestra un listado ordenado por fecha reciente. En cada inscripción se ve su estado, título de la oferta, empresa, ciudad y fecha de inscripción. En la parte superior hay un buscador y un desplegable para poder filtrar las ofertas y mostrar todas, solo las aceptadas, solo las rechazadas o únicamente las que están a la espera de resolverse.

Se puede acceder a los detalles de la inscripción, donde se muestra la fecha de inscripción, currículum adjunto, estado de la inscripción, y todos los detalles de la oferta de empleo (título, empresa, sector, ...). El usuario puede cancelar su inscripción siempre y cuando no haya sido todavía resuelta. En caso de que el usuario cancele su inscripción, desaparecería de su listado de inscripciones realizadas, y podría volver a inscribirse de nuevo si lo desea, desde el listado de ofertas de empleo. También puede ver los detalles de la empresa e iniciar un chat.



Figura 5.6: Prototipo listado de inscripciones, detalles y cancelar inscripción.

La tercera y última pantalla a la que puede acceder el usuario desde el menú navegacional de la parte inferior muestra un listado con los chats del usuario con las empresas. En cada chat se ve la imagen de la empresa, su nombre, el último mensaje de la conversación y un icono que avisa si hay algún mensaje sin leer en la conversación. Se ordenan de más recientes a más antiguos, según el último mensaje de la conversación. En la parte superior hay un buscador.

Dentro del chat se ordenan los mensajes de la conversación de abajo hacia arriba según su fecha de envío más reciente. En la parte inferior está la zona de escritura. Se muestra la fecha y hora de cada mensaje. El usuario puede eliminar el chat, y todos los mensajes y la conversación serían borrados de su listado de chats (sólo se borrarían para él, la empresa seguiría viéndolos en su listado). También puede acceder a los detalles de la empresa pinchando sobre el encabezado donde aparece la imagen y el nombre de la empresa.



Figura 5.7: Prototipo listado chats, chat y eliminar chat.

El perfil del usuario muestra sus datos personales: imagen, sexo, nombre, email, fecha de nacimiento, teléfono, ciudad; y sus preferencias: habilidades técnicas, sector, tipo de contrato, idiomas, autónomo, carnet de conducir y vehículo propio. También se muestran la fecha en la que se dio de alta en la aplicación y la fecha de la última edición del perfil. Hay un botón para acceder a los ajustes.

En la pantalla de ajustes el usuario puede acceder a editar su perfil, cambiar su contraseña, administrar sus currículums, ver el calendario de entrevistas, editar sus preferencias y cerrar sesión.

Editar perfil muestra un formulario donde se cargan por defecto los datos personales del usuario. Los campos son: imagen, sexo, nombre, email, fecha de nacimiento, teléfono y ciudad. El email no se puede cambiar.

Cambiar contraseña está compuesto por un formulario con tres campos donde el usuario debe introducir su actual contraseña, la nueva contraseña y repetir de nuevo la nueva contraseña. Es idéntico para las empresas y los administradores, por lo que no se explicará de nuevo su funcionamiento cuando se nombre más adelante.

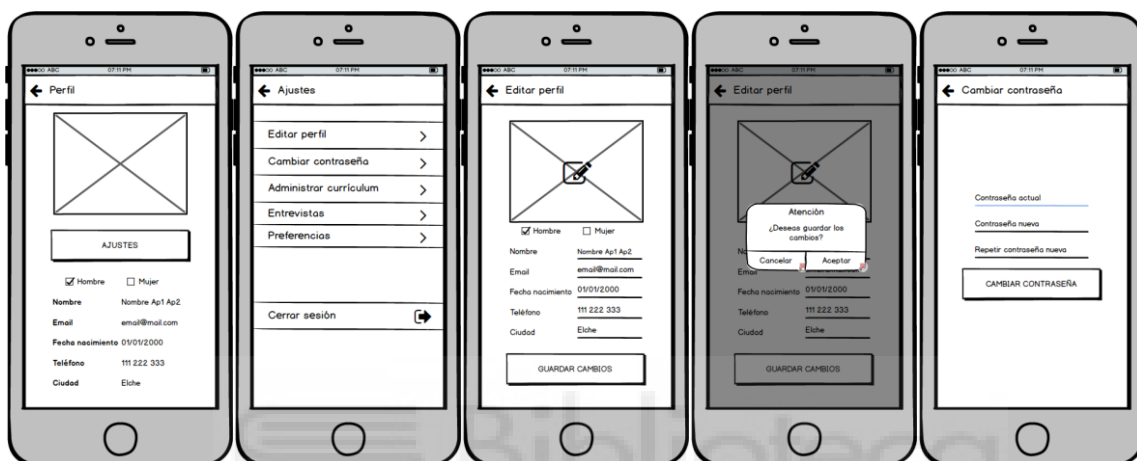


Figura 5.8: Prototipo perfil, ajustes, editar perfil y cambiar contraseña.

El listado de currículums muestra ordenados por fecha reciente todos los currículums del usuario. Se muestra para cada uno de ellos su título y el nombre del documento. En la parte superior hay un buscador y un botón para crear un nuevo currículum. La pantalla para crear uno nuevo está formada por un botón para cargar un documento PDF de la memoria del dispositivo y por dos campos para asignarle un título y una descripción. La descripción es opcional. Los detalles de cada currículum muestran el título, nombre del documento, descripción, fechas de creación y última edición, y un botón para acceder a la visualización del documento PDF, donde se puede hacer zoom. El usuario también puede editar y eliminar sus currículums. La pantalla para editar el currículum carga los datos actuales y tiene los mismos campos que la pantalla de creación. Si el usuario elimina un currículum, desaparecería de la lista de sus currículums y no podría adjuntarlo a ninguna nueva inscripción que realice. Sin embargo, queda presente en todas aquellas inscripciones en las que se hubiera adjuntado previamente, de modo que las empresas podrían seguir viéndolo en sus inscripciones anteriores.

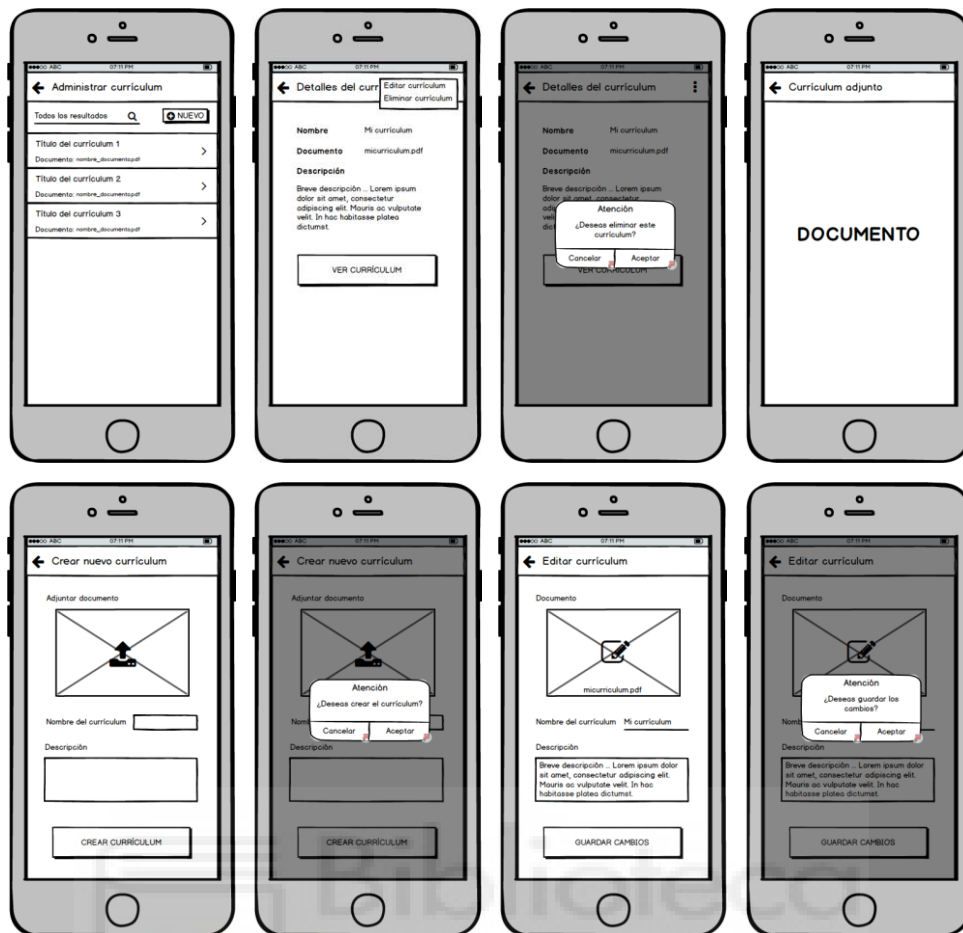


Figura 5.9: Prototipo listado currículums, detalles, crear nuevo y editar.

El calendario de entrevistas muestra una agenda que los usuarios pueden deslizar para comprobar las fechas en las que tienen programadas entrevistas de empleo referentes a las inscripciones que las empresas han resuelto citándoles para una determinada fecha. Los días en los que tengan una entrevista aparecen señalados y se puede ver la hora, título de la oferta de empleo, nombre de la empresa y las observaciones que deje la empresa.

Las preferencias del usuario son un complemento a sus datos personales, y sirven para permitir que las empresas puedan localizarles mediante un buscador específico que éstas disponen y que se detallará más adelante. El formulario consta de los campos: sector, habilidades técnicas, tipo de contrato, idiomas, autónomo, carnet de conducir, vehículo propio y una casilla donde indiquen si desean ser localizados por las empresas mediante el citado buscador.

Por último, los usuarios pueden cerrar sesión. Les aparece un mensaje de confirmación previo a cerrar sesión. Una vez cerrada la sesión, son redirigidos al apartado de Inicio de sesión y Registro. El funcionamiento es el mismo también para las empresas y los administradores, por tanto, se pasará por alto su explicación cuando se nombre en sus correspondientes apartados.



Figura 5.10: Prototipo listado entrevistas, preferencias y cerrar sesión.

EMPRESAS:

El apartado de las empresas está formado por cinco subapartados: perfil, listado de ofertas de empleo publicadas, inscripciones de usuarios, búsqueda de usuarios y chats. Al perfil se accede desde el encabezado, mientras que el resto forman parte del menú navegacional inferior.

La pantalla con las ofertas de empleo publicadas se muestra por defecto cuando abren la aplicación o cuando inician sesión. El listado está ordenado por fecha de publicación reciente. Para cada una se muestra el título, la fecha de publicación, el número de inscritos y el número de inscripciones sin resolver. También hay un icono que indica si la oferta de empleo está abierta o cerrada. En la parte superior hay un buscador, un desplegable para filtrar ofertas abiertas o cerradas, y un botón para publicar una nueva oferta de empleo. Adicionalmente, hay un botón para poder exportar el listado de ofertas publicadas en formato CSV, Excel o PDF.

Se puede acceder a los detalles de cada oferta de empleo publicada, donde aparece un recuadro con la fecha de publicación, la fecha de la última edición, el número de inscritos, el número de inscripciones sin resolver y el estado de la oferta (Abierta/Cerrada). También se muestra el título, sector, ciudad, descripción, tipo de contrato, horario de trabajo, salario, estudios, idiomas, experiencia mínima y otros. Hay un botón para acceder al listado de los usuarios inscritos. Además, se puede editar, cerrar o eliminar la propia oferta. Los campos que aparecen para editar la oferta son los mismos que en la pantalla de publicar una nueva oferta. Si la empresa cierra o elimina una oferta de empleo, todas las inscripciones de usuarios que no estuviesen resueltas serían rechazadas. Las ofertas cerradas no aparecen en el listado de ofertas que ven los usuarios, mientras que las eliminadas además son borradas del listado de ofertas publicadas por la empresa.



Figura 5.11: Prototipo listado ofertas publicadas, exportar, publicar oferta, detalles, resolver inscripción, cerrar oferta, eliminar oferta y editar oferta.

La pantalla con las inscripciones de los usuarios a la oferta de empleo seleccionada muestra un listado de los usuarios inscritos, ordenado por fecha reciente. Para cada inscripción se ve la imagen del usuario, estado de la inscripción, nombre del usuario y fecha de inscripción. En la parte superior hay un buscador y un desplegable para filtrar las inscripciones y mostrar las aceptadas, las rechazadas, las que están a la espera de resolverse o todas. Hay un botón para poder exportar el listado de usuarios inscritos en formato CSV, Excel o PDF.

En los detalles de la inscripción del usuario se muestra la fecha de inscripción y estado de la inscripción. Si ha sido resuelta, se muestran las observaciones dejadas al usuario. En caso de que haya sido aceptada, también se muestra la fecha de la entrevista. Además, aparecen los detalles de la oferta de empleo (título, empresa, sector, ...) y el nombre y currículum del usuario. También se pueden ver los detalles del usuario, visualizar el documento del currículum adjunto e iniciar un chat con el usuario. En la propia pantalla de los detalles de la inscripción del usuario es donde la empresa dispone de un pop-up para resolver la inscripción. Puede aceptarla o rechazarla. En caso de aceptarla, debe fijar una fecha y hora para la entrevista, y puede dejar al usuario un mensaje con las observaciones si lo desea. Si se rechaza, únicamente debe rellenar el campo observaciones para expresar el motivo.



Figura 5.12: Prototipo inscritos en la oferta, exportar, detalles inscripción y resolver inscripción.

La pantalla con las inscripciones de los usuarios muestra un listado de todas las inscripciones a todas las ofertas de empleo publicadas por la empresa, ordenado por fecha reciente. En cada inscripción se ve la imagen del usuario, estado de la inscripción, nombre del usuario y fecha de inscripción. En la parte superior hay un buscador y un desplegable para filtrar las inscripciones entre las aceptadas, las rechazadas, las que están sin resolverse o todas. Hay un botón para exportar el listado de inscritos en formato CSV, Excel o PDF. La pantalla de los detalles de la inscripción del usuario es idéntica a la descrita en el párrafo anterior.



Figura 5.13: Prototipo inscritos en todas las ofertas, exportar, detalles y resolver.

En tercer lugar, se encuentra el buscador de usuarios. En la parte superior hay un buscador y un filtro para que las empresas puedan localizar a usuarios que tengan activada la opción de ser localizados en sus preferencias. Se permite filtrar por: habilidades técnicas, sector, tipo de contrato, ciudad, idiomas, autónomo, carnet de conducir, vehículo propio, género y edad. Los usuarios que cumplan con los filtros son ordenados alfabéticamente por su nombre. Se muestra su imagen y su correo electrónico. Las empresas pueden acceder al perfil del usuario y ver sus datos personales y preferencias. Además, pueden iniciar un chat con ellos. Hay un botón para exportar un listado de usuarios en formato CSV, Excel o PDF.



Figura 5.14: Prototipo listado de usuarios, exportar y búsqueda/filtrado usuarios.

La cuarta pantalla a la que pueden acceder las empresas desde el menú navegacional de la parte inferior muestra un listado con los chats con usuarios. Se muestra para cada chat la imagen del usuario, su nombre y el último mensaje de la conversación. Hay un icono que avisa si hay algún mensaje sin leer en alguna conversación. Se ordenan según la fecha reciente del último mensaje. En la parte superior hay un buscador. El funcionamiento de la pantalla de chat es el mismo que el descrito en el apartado de los chats de los usuarios.



Figura 5.15: Prototipo listado chats, chat y eliminar chat.

El perfil de la empresa muestra su imagen, nombre, descripción, email, teléfono, dirección y ciudad. Se muestra la fecha en la que se dio de alta en la aplicación y la fecha en la que se haya editado por última vez su perfil. Hay un botón para acceder a los ajustes.

La pantalla de ajustes permite a la empresa acceder a editar su perfil, cambiar su contraseña, ver el calendario de entrevistas y cerrar sesión.

Editar perfil muestra un formulario sus datos actuales y permite editar su imagen, nombre, descripción, teléfono, dirección y ciudad. El email no se puede cambiar.

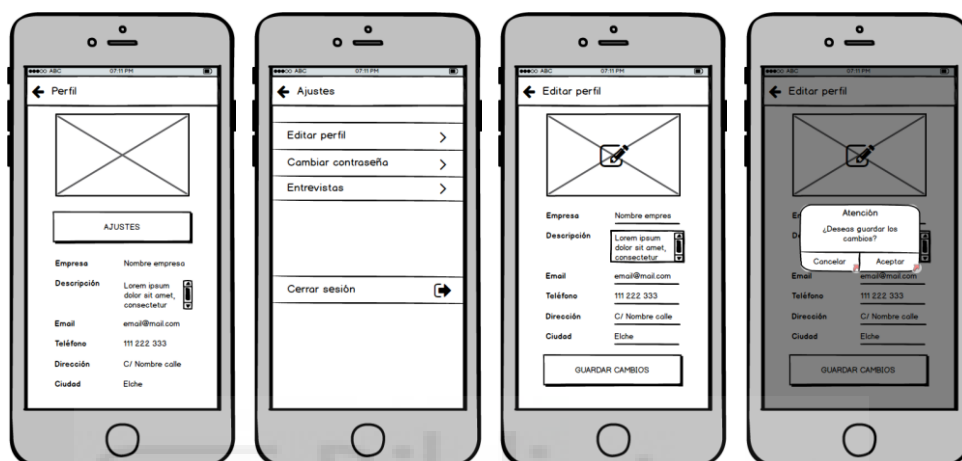


Figura 5.16: Prototipo perfil, ajustes y editar perfil.

Cambiar contraseña tiene el mismo aspecto y funcionamiento que para los usuarios.

El calendario de entrevistas muestra una agenda donde pueden comprobar las fechas en las que tienen programadas entrevistas de empleo referentes a las inscripciones de usuarios que han sido aceptadas y por tanto se haya fijado una fecha para la entrevista. Los días dónde hay alguna entrevista, aparecen señalados y se muestra un recuadro con la hora, título de la oferta de empleo, nombre del usuario y las observaciones que se le dejaron al usuario cuando se resolvió su inscripción. Además, pueden editar la fecha, hora y observaciones de cualquier entrevista. La forma de hacerlo es en la propia agenda, manteniendo pulsado sobre el recuadro con los datos de la entrevista hasta que se muestre un pop-up con el formulario para editarla. Si el usuario realiza una pulsación simple sobre alguna entrevista de la agenda, la aplicación le muestra un aviso con vibración y un pequeño texto advirtiéndole que, si desea editarla, debe mantener pulsado.

Por último, las empresas pueden cerrar sesión. El funcionamiento es el mismo que para los usuarios.



Figura 5.17: Prototipo cambiar contraseña, ver entrevistas, editar entrevistas y cerrar sesión.

ADMINISTRADORES Y SUPER:

El apartado de los administradores está formado por cuatro subapartados: perfil, listado de administradores, listado de usuarios y listado de empresas. Al perfil pueden acceder desde el encabezado. Los otros forman parte del menú navegacional inferior.

La primera pantalla del menú lista todos los administradores mostrando su imagen, nombre y correo electrónico. Además, aquellos que sean super, se les resalta con un icono. Hay un buscador en la parte superior, y para los super también hay un botón para añadir un nuevo administrador. Hay un botón para exportar un listado de los administradores en formato CSV, Excel o PDF.

El super es designado por el gestor de la aplicación y la base de datos, por lo que es un proceso externo a la aplicación y que por tanto no se puede realizar desde ella.

La pantalla para crear un nuevo administrador muestra un formulario para indicar el nombre y correo electrónico.



Figura 5.18: Prototipo listado administradores, exportar y crear administrador.

Los detalles de cada administrador muestran su imagen, nombre, email, fecha en que fueron dados de alta, y en caso de ser super, se muestra un icono junto a su nombre. Únicamente los super tienen acceso a editar y eliminar administradores. Cuando se elimine a un administrador, desaparece del listado de administradores. La pantalla para editar a un administrador carga su imagen, nombre y correo electrónico, pero solo se permite editar la imagen y el nombre.



Figura 5.19: Prototipo detalles del administrador, eliminar y editar administrador.

La segunda pantalla es la de los usuarios, donde se les lista mostrando su imagen, nombre y correo electrónico. En la parte superior hay un buscador y un botón para

añadir usuarios. También hay un botón para exportar un listado de los usuarios en formato CSV, Excel o PDF.

La pantalla para crear un nuevo usuario muestra un formulario para indicar el nombre, correo electrónico, sexo, fecha de nacimiento, teléfono, ciudad y opcionalmente se puede añadir una imagen.



Figura 5.20: Prototipo listado usuarios, exportar y crear usuario.

Los detalles de cada usuario muestran sus datos personales: imagen, sexo, nombre, email, fecha de nacimiento, teléfono, ciudad; sus preferencias: habilidades técnicas, sector, tipo de contrato, idiomas, autónomo, carnet de conducir y vehículo propio; también la fecha en la que se dio de alta en la aplicación y dos botones para acceder a sus currículums e inscripciones. Pueden de editar y eliminar al usuario. Cuando se elimine a un usuario, desaparece del listado de usuarios y todas sus inscripciones a ofertas de empleo que no hubieran sido resueltas, son canceladas. La pantalla para editar a un usuario carga su imagen, sexo, nombre, correo electrónico, fecha de nacimiento, teléfono y ciudad, pero el correo electrónico no se permite cambiar. Las preferencias del usuario no pueden ser modificadas por los administradores.



Figura 5.21: Prototipo detalles del usuario, eliminar y editar usuario.

El listado de currículums muestra ordenados por fecha reciente todos los currículums del usuario, incluidos los eliminados. Se muestra para cada uno de ellos su título, el nombre del documento y el estado del currículum (en uso o eliminado). En la parte superior hay un buscador, un filtro para mostrar Todos/En uso/Eliminados y un botón para crear un nuevo currículum. Hay también un botón para exportar un listado de los currículums en formato CSV, Excel o PDF. La pantalla para crear uno nuevo está formada por un botón para cargar un documento PDF desde el dispositivo y por dos campos para asignar un título y una descripción. La descripción es opcional.



Figura 5.22: Prototipo Listado currículums del usuario, exportar y crear currículum.

Los detalles de cada currículum muestran el título, nombre del documento, descripción, fechas de creación y última edición, y un botón para visualizar el documento PDF. También pueden editar y eliminar los currículums del usuario. Las pantallas de editar y la acción de eliminar son idénticas a las que se han explicado anteriormente en el apartado de los usuarios. Cuando se eliminan, siguen estando presentes en todas aquellas inscripciones en las que se hubieran adjuntado previamente, de modo que las empresas pueden seguir viéndolos en las inscripciones de usuarios anteriores.



Figura 5.23: Prototipo detalles currículum, eliminar y editar currículum.

Las inscripciones del usuario se muestran ordenadas por fecha reciente. Para cada una se ve su estado, título de la oferta, empresa, ciudad y fecha de inscripción. En la parte superior hay un buscador y un desplegable para poder filtrar las ofertas según su estado (Aceptadas/Rechazadas/Canceladas/Sin Resolver). Hay además un botón para exportar el listado de inscripciones en formato CSV, Excel o PDF.

En los detalles de la inscripción se muestra la fecha de inscripción, el estado de la inscripción, todos los detalles de la oferta de empleo (título, empresa, sector, ...), y el nombre y email del usuario. Es prácticamente igual que la vista de las inscripciones de usuarios que disponen las empresas. El administrador también puede resolver inscripciones de usuarios.



Figura 5.24: Prototipo listado inscripciones del usuario, exportar, detalles y resolver inscripción.

La tercera y última pantalla del menú está compuesta por las empresas, donde son listadas mostrando su imagen, nombre y correo electrónico. En la parte superior hay un buscador y un botón para añadir nuevas empresas. Hay también un botón para exportar el listado en formato CSV, Excel o PDF.

La pantalla para crear una nueva empresa muestra un formulario donde se debe indicar el nombre, descripción, correo electrónico, teléfono, dirección, ciudad y se puede añadir una imagen.



Figura 5.25: Prototipo listado empresas, exportar y crear empresa.

Los detalles de cada empresa muestran su imagen, nombre, descripción, email, teléfono, dirección, ciudad, la fecha en la que se dieron de alta en la aplicación y un botón para acceder sus ofertas de empleo publicadas. Los administradores pueden editar y eliminar a la empresa. Cuando se elimine a una empresa, desaparece del listado de empresas. Además, cuando se eliminen, todas sus ofertas de empleo son cerradas y las inscripciones de usuarios a sus ofertas de empleo que no hayan sido resueltas, son canceladas. La pantalla para editar a una empresa carga su imagen, nombre, descripción, correo electrónico, teléfono, dirección y ciudad. El correo electrónico no se puede cambiar.



Figura 5.26: Prototipo detalles empresa, eliminar empresa y editar empresa.

La pantalla con las ofertas publicadas por empresa es igual que la del apartado de las empresas. La única diferencia es que también se listan las ofertas eliminadas. Al igual que para las empresas, hay un buscador, un filtro, un botón para publicar una nueva oferta de empleo y otro botón para exportar el listado en formato CSV, Excel o PDF. La pantalla para crear una nueva oferta de empleo es idéntica a la descrita en el apartado de las empresas.



Figura 5.27: Prototipo listado ofertas publicadas por la empresa, exportar y publicar nueva oferta.

La pantalla con los detalles de cada oferta de empleo es igual que para las empresas. La única diferencia es que, dado que las ofertas eliminadas son visibles para los administradores, al acceder a los detalles, no les aparece la opción de editar o eliminar.



Figura 5.28: Prototipo detalles oferta, resolver inscripción, cerrar oferta, eliminar oferta y editar oferta.

El listado de los usuarios inscritos es también igual que el que ven las empresas. Solo se añade la opción de filtrar y mostrar también las inscripciones canceladas por los usuarios. Cuenta también con un botón para exportar el listado de inscritos en formato CSV, Excel o PDF.

La pantalla con los detalles de la inscripción del usuario también es prácticamente idéntica a la del apartado de las empresas. La única diferencia es que no se puede abrir un chat con el usuario. El resto de información mostrada, accesos a los detalles del usuario y currículum adjunto, además del formulario de resolución de la inscripción, son iguales.



Figura 5.29: Prototipo listado inscritos, exportar, detalles y resolver inscripción.

El perfil de la empresa muestra su imagen, nombre, email, la fecha en la que se dio de alta en la aplicación y la fecha en la que se haya editado por última vez su perfil. Hay un botón para acceder a los ajustes.

La pantalla de ajustes permite al administrador acceder a editar su perfil, cambiar su contraseña y cerrar sesión.

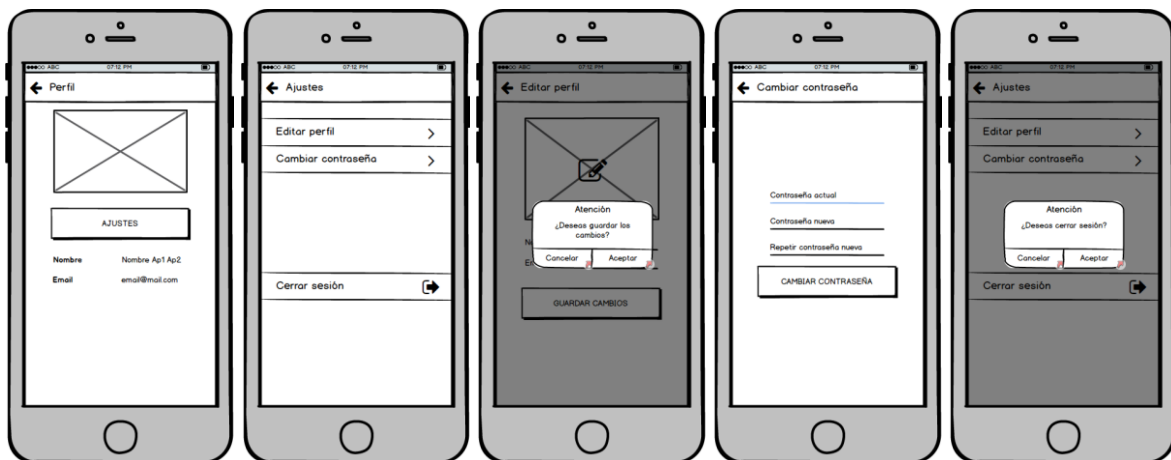


Figura 5.30: Prototipo perfil, ajustes, editar perfil, cambiar contraseña y cerrar sesión.

Los administradores no tienen acceso a los chats de los usuarios ni de las empresas.

Cuando se elimine a un usuario, empresa o administrador, se les envía un correo electrónico informándoles de su baja en la aplicación. También se les da de baja de la aplicación y ya no tienen acceso a su cuenta.

En todas las pantallas de la aplicación donde se cree, actualice o elimine algo, se muestra un mensaje de éxito en la parte superior de la pantalla para confirmar que la acción se ha completado correctamente. Además, todos los formularios muestran mensajes de error personalizados.

5.2.2 Casos de uso

Como se ha expuesto, la aplicación cuenta con tres actores diferentes. A continuación, se definen los casos de uso de cada uno de ellos.

Actor	Administrador
Descripción	Puede ver, añadir, editar y eliminar usuarios y empresas. También puede resolver inscripciones y ver, añadir, editar y eliminar currículums de los usuarios. No tiene acceso a los chats ni tampoco a las entrevistas.
Casos de uso	C.U. 1: Iniciar sesión. C.U. 2: Recuperar contraseña. C.U. 3: Ver perfil. C.U. 4: Listar ajustes. C.U. 5: Editar perfil. C.U. 6: Cambiar contraseña. C.U. 7: Cerrar sesión.

C.U. 8: Listar administradores.
C.U. 9: Crear administrador.
C.U. 10: Buscar administradores.
C.U. 11: Exportar administradores.
C.U. 12: Ver detalles administrador.
C.U. 13: Editar administrador.
C.U. 14: Eliminar administrador.
C.U. 15: Listar usuarios.
C.U. 16: Crear usuario.
C.U. 17: Buscar usuarios.
C.U. 18: Exportar usuarios.
C.U. 19: Ver detalles usuario.
C.U. 20: Editar usuario.
C.U. 21: Eliminar usuario.
C.U. 22: Listar currículums del usuario.
C.U. 23: Crear currículum.
C.U. 24: Buscar currículums.
C.U. 25: Filtrar currículums.
C.U. 26: Exportar currículums.
C.U. 27: Ver detalles currículum.
C.U. 28: Editar currículum.
C.U. 29: Eliminar currículum.
C.U. 30: Listar inscripciones del usuario.
C.U. 31: Buscar inscripciones.
C.U. 32: Filtrar inscripciones.
C.U. 33: Exportar inscripciones.
C.U. 34: Ver detalles inscripción.
C.U. 35: Mostrar/ocultar detalles oferta.
C.U. 36: Ver detalles currículum.
C.U. 37: Resolver inscripción.
C.U. 38: Listar empresas.
C.U. 39: Crear empresa.
C.U. 40: Buscar empresas.
C.U. 41: Exportar empresas.
C.U. 42: Ver detalles empresa.
C.U. 43: Editar empresa.
C.U. 44: Eliminar empresa.

	<p>C.U. 45: Listar ofertas de empleo de la empresa.</p> <p>C.U. 46: Crear oferta.</p> <p>C.U. 47: Buscar ofertas.</p> <p>C.U. 48: Filtrar ofertas.</p> <p>C.U. 49: Exportar ofertas.</p> <p>C.U. 50: Ver detalles oferta.</p> <p>C.U. 51: Editar oferta.</p> <p>C.U. 52: Cerrar oferta.</p> <p>C.U. 53: Eliminar oferta.</p> <p>C.U. 54: Listar inscripciones de usuarios.</p> <p>C.U. 55: Buscar inscripciones.</p> <p>C.U. 56: Filtrar inscripciones.</p> <p>C.U. 57: Exportar inscripciones.</p> <p>C.U. 58: Ver detalles inscripción.</p> <p>C.U. 59: Mostrar/ocultar detalles oferta</p> <p>C.U. 60: Ver detalles usuario.</p> <p>C.U. 61: Ver detalles currículum.</p> <p>C.U. 62: Resolver inscripción.</p>
--	--

Tabla 5.2: Casos de uso del actor Administrador.

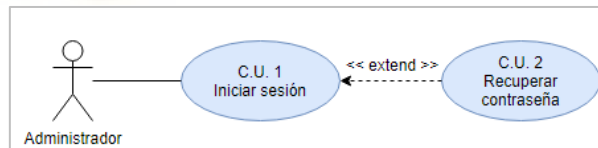


Figura 5.31: Diagrama C.U. Administrador: Autenticación.

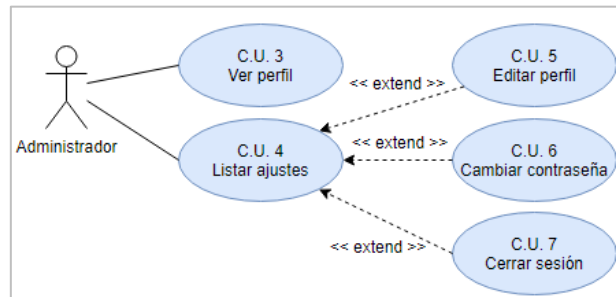


Figura 5.32: Diagrama C.U. Administrador: Perfil y Ajustes.

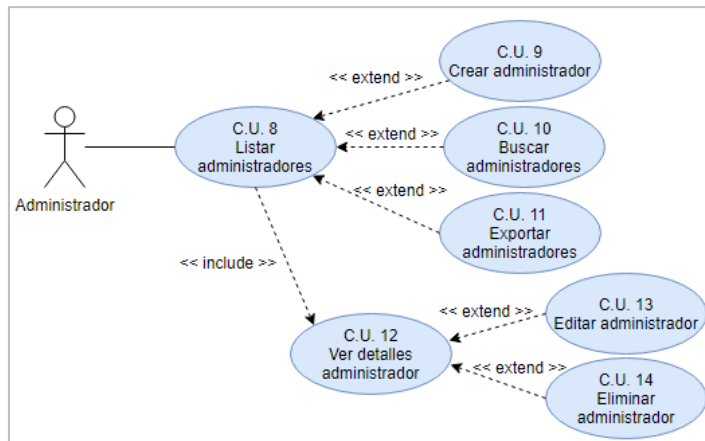


Figura 5.33: Diagrama C.U. Administrador: Gestionar Administradores.

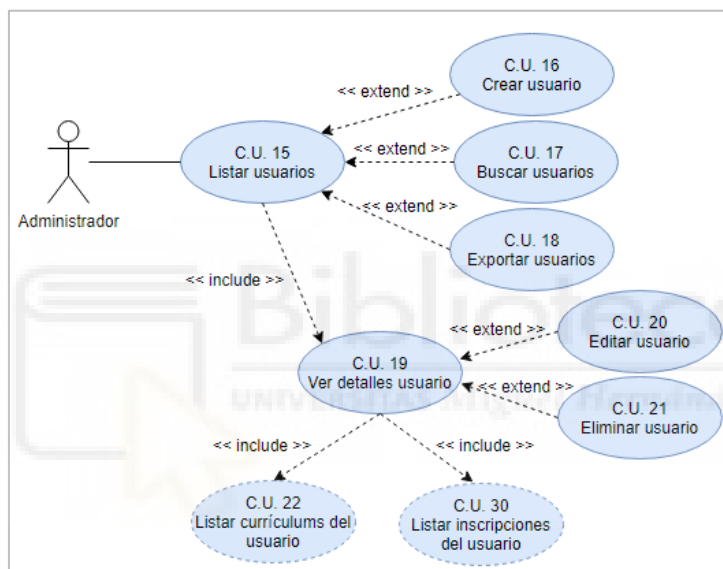


Figura 5.34: Diagrama C.U. Administrador: Gestionar Usuarios.

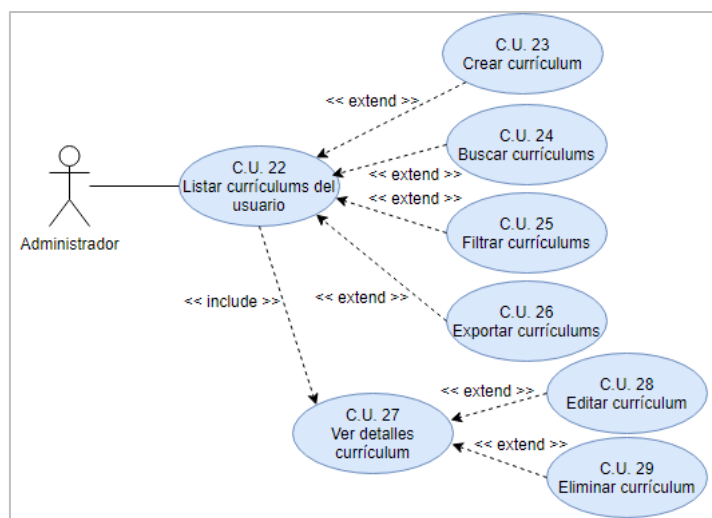


Figura 5.35: Diagrama C.U. Administrador: Gestionar currículums usuario.

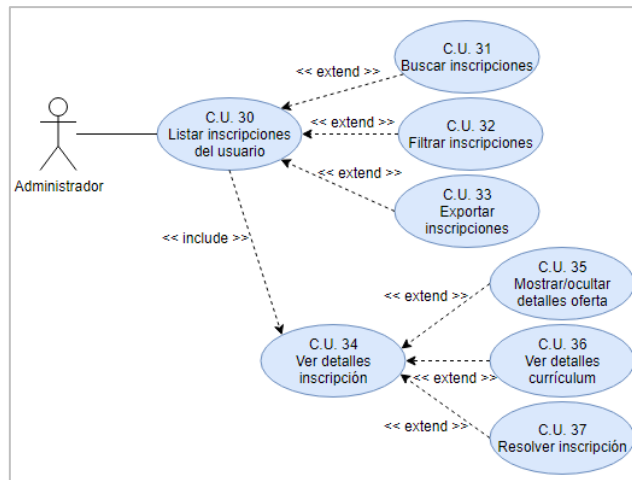


Figura 5.36: Diagrama C.U. Administrador: Gestionar inscripciones usuario.

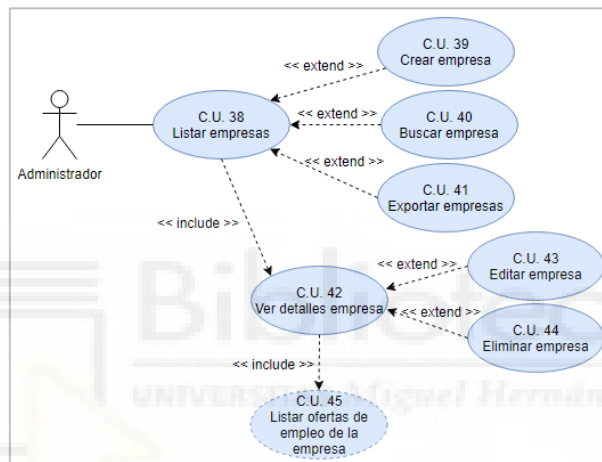


Figura 5.37: Diagrama C.U. Administrador: Gestionar empresas.

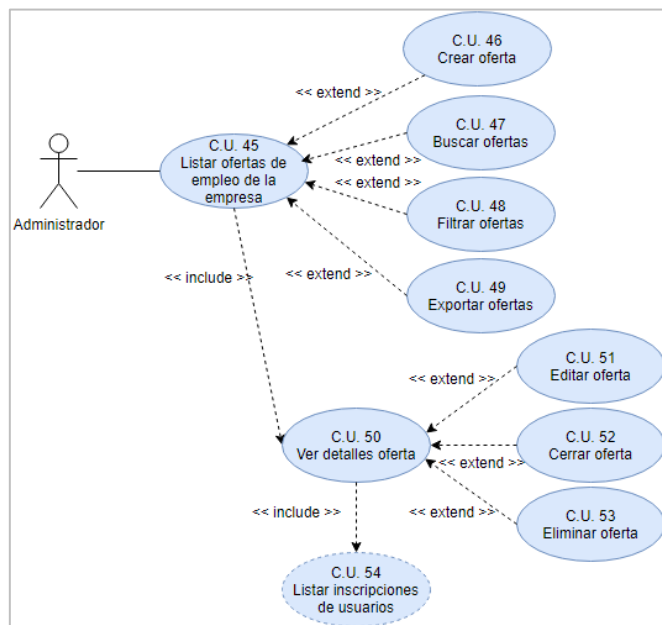


Figura 5.38: Diagrama C.U. Administrador: Gestionar ofertas de empleo de la empresa.

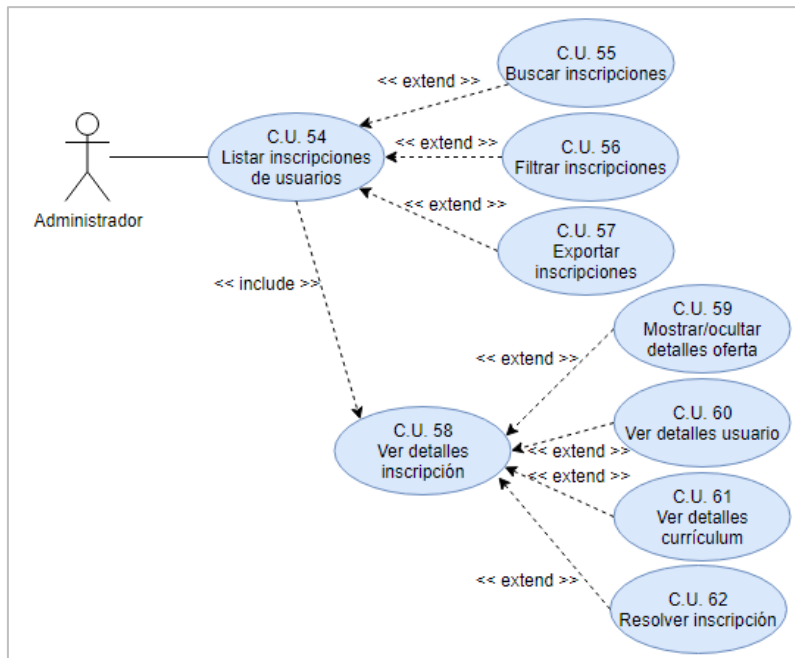


Figura 5.39: Diagrama C.U. Administrador: Gestionar inscripciones en las ofertas de empleo de la empresa.

Actor	Usuario
	Tendrá acceso a un listado de ofertas de empleo a las que podrá inscribirse adjuntando un currículum. Tendrá disponible un chat para hablar con empresas.
Casos de uso	<p>C.U. 63: Registrar usuario. C.U. 64: Iniciar sesión. C.U. 65: Recuperar contraseña. C.U. 66: Ver perfil. C.U. 67: Listar ajustes. C.U. 68: Editar perfil. C.U. 69: Cambiar contraseña. C.U. 70: Listar currículums. C.U. 71: Crear currículum. C.U. 72: Buscar currículums. C.U. 73: Ver detalles currículum. C.U. 74: Ver documento PDF. C.U. 75: Editar currículum.</p>

	<p>C.U. 76: Eliminar currículum.</p> <p>C.U. 77: Listar entrevistas.</p> <p>C.U. 78: Buscar entrevistas.</p> <p>C.U. 79: Editar preferencias.</p> <p>C.U. 80: Cerrar sesión.</p> <p>C.U. 81: Listar ofertas de empleo disponibles.</p> <p>C.U. 82: Buscar ofertas.</p> <p>C.U. 83: Filtrar ofertas.</p> <p>C.U. 84: Ver detalles oferta.</p> <p>C.U. 85: Realizar inscripción.</p> <p>C.U. 86: Compartir oferta.</p> <p>C.U. 87: Ver detalles empresa.</p> <p>C.U. 88: Iniciar chat.</p> <p>C.U. 89: Listar inscripciones.</p> <p>C.U. 90: Buscar inscripciones.</p> <p>C.U. 91: Filtrar inscripciones.</p> <p>C.U. 92: Ver detalles inscripción.</p> <p>C.U. 93: Cancelar inscripción.</p> <p>C.U. 94: Listar chats.</p> <p>C.U. 95: Buscar chats.</p> <p>C.U. 96: Ver chat.</p> <p>C.U. 97: Enviar mensaje.</p> <p>C.U. 98: Eliminar chat.</p> <p>C.U. 99: Ver detalles empresa.</p>
--	--

Tabla 5.3: Casos de uso del actor Usuario.

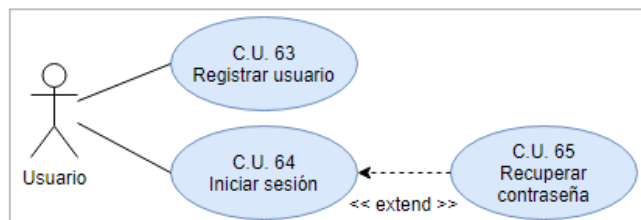


Figura 5.40: Diagrama C.U. Usuario: Autenticación.

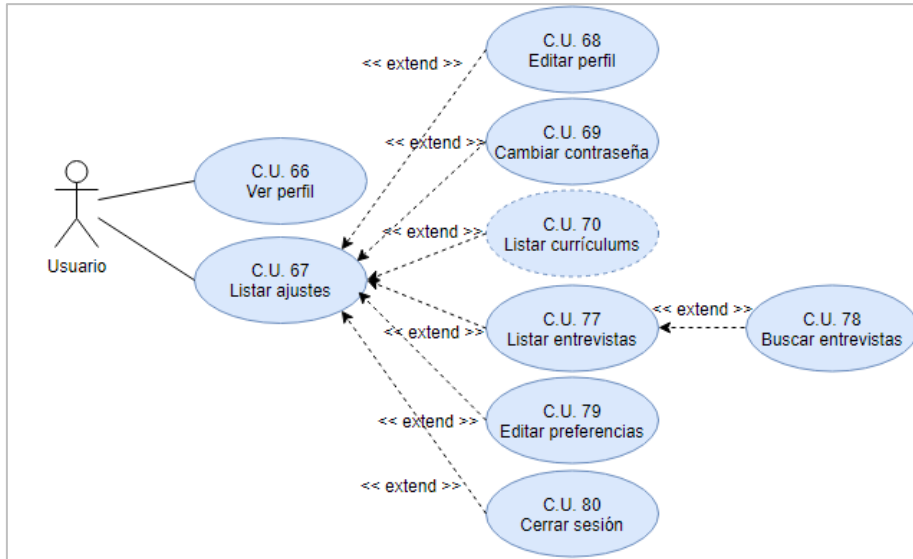


Figura 5.41: Diagrama C.U. Usuario: Perfil y Ajustes.

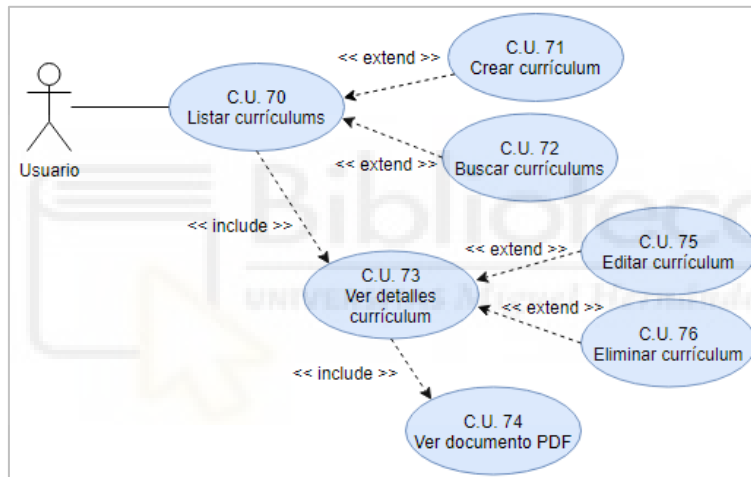


Figura 5.42: Diagrama C.U. Usuario: Gestionar currículums.

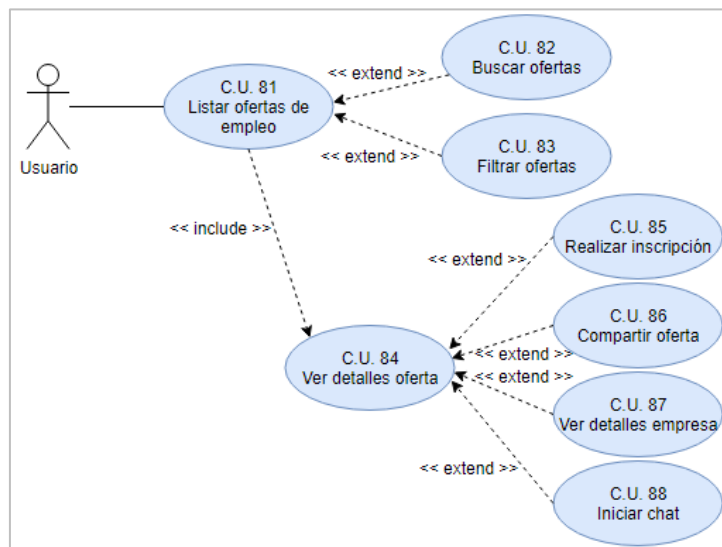


Figura 5.43: Diagrama C.U. Usuario: Ofertas de empleo.

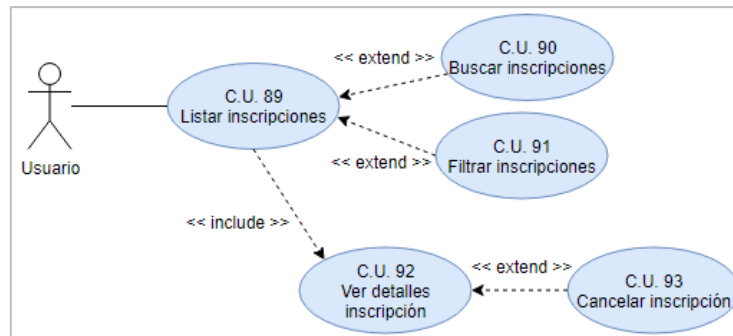


Figura 5.44: Diagrama C.U. Usuario: Inscripciones.

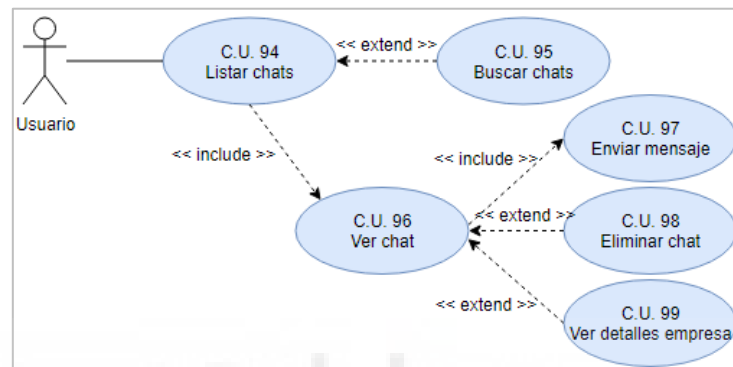


Figura 5.45: Diagrama C.U. Usuario: Chats.

Actor	Empresa
	<p>Publicará ofertas de empleo en las que los usuarios se podrán inscribir. Podrá concretar entrevistas con usuarios. Tendrá disponible un chat para hablar con usuarios.</p>
Casos de uso	<p>C.U. 100: Registrar empresa. C.U. 101: Iniciar sesión. C.U. 102: Recuperar contraseña. C.U. 103: Ver perfil. C.U. 104: Listar ajustes. C.U. 105: Editar perfil. C.U. 106: Cambiar contraseña. C.U. 107: Listar entrevistas. C.U. 108: Buscar entrevistas. C.U. 109: Editar entrevista. C.U. 110: Cerrar sesión.</p>

C.U. 111: Listar ofertas de empleo publicadas.
C.U. 112: Crear oferta.
C.U. 113: Buscar ofertas.
C.U. 114: Filtrar ofertas.
C.U. 115: Exportar ofertas.
C.U. 116: Ver detalles oferta.
C.U. 117: Editar oferta.
C.U. 118: Cerrar oferta.
C.U. 119: Eliminar oferta.
C.U. 120: Listar inscripciones en la oferta.
C.U. 121: Buscar inscripciones.
C.U. 122: Filtrar inscripciones.
C.U. 123: Exportar inscripciones.
C.U. 124: Ver detalles inscripción.
C.U. 125: Mostrar/ocultar detalles oferta
C.U. 126: Ver detalles usuario.
C.U. 127: Ver detalles currículum.
C.U. 128: Resolver inscripción.
C.U. 129: Listar inscripciones de usuarios en todas las ofertas.
C.U. 130: Buscar inscripciones.
C.U. 131: Filtrar inscripciones.
C.U. 132: Exportar inscripciones.
C.U. 133: Ver detalles inscripción.
C.U. 134: Mostrar/ocultar detalles oferta
C.U. 135: Ver detalles usuario.
C.U. 136: Ver detalles currículum.
C.U. 137: Resolver inscripción.
C.U. 138: Listar usuarios.
C.U. 139: Buscar usuarios.
C.U. 140: Filtrar usuarios.
C.U. 141: Exportar usuarios.
C.U. 142: Ver detalles usuario.
C.U. 143: Listar chats.
C.U. 144: Buscar chats.
C.U. 145: Ver chat.
C.U. 146: Enviar mensaje.

	<p>C.U. 147: Eliminar chat.</p> <p>C.U. 148: Ver detalles usuario.</p>
--	--

Tabla 5.4: Casos de uso del actor Empresa.

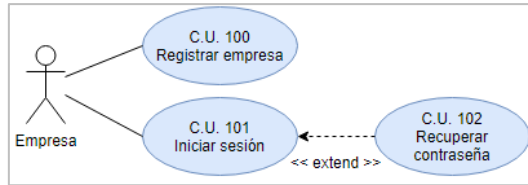


Figura 5.46: Diagrama casos de uso C.U. Empresa: Autenticación.

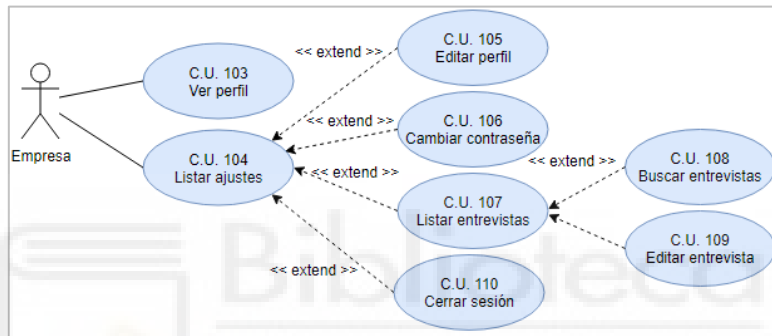


Figura 5.47: Diagrama C.U. Empresa: Perfil y Ajustes.

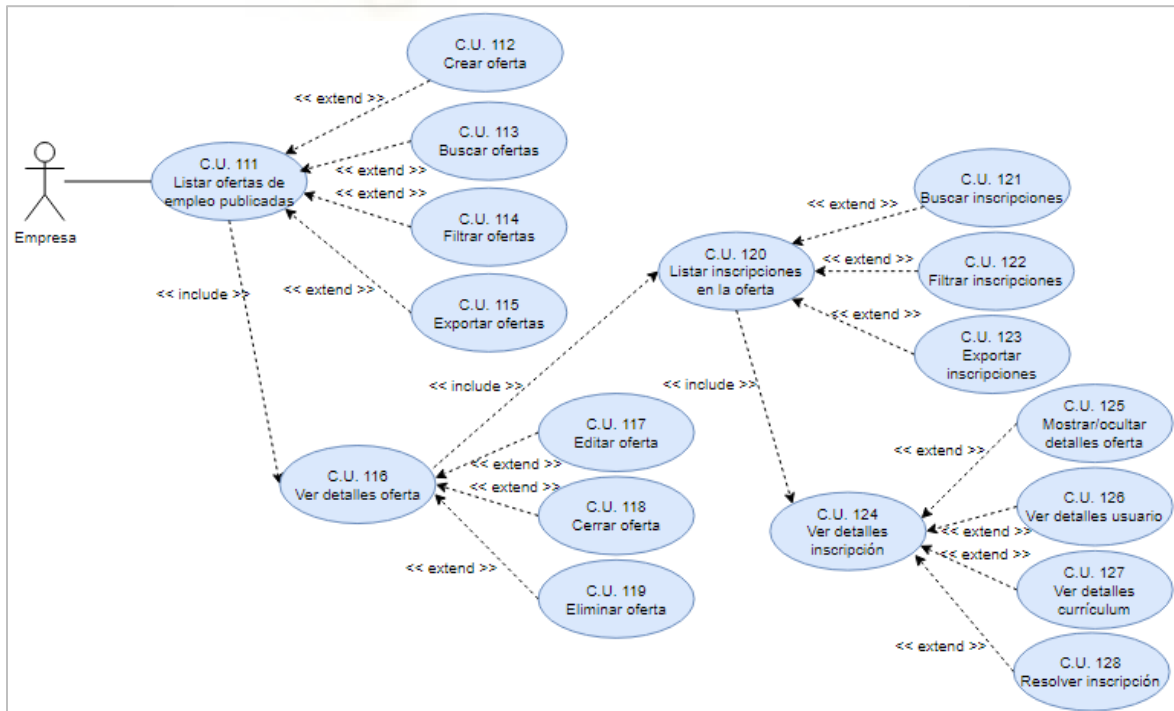


Figura 5.48: Diagrama C.U. Empresa: Ofertas.

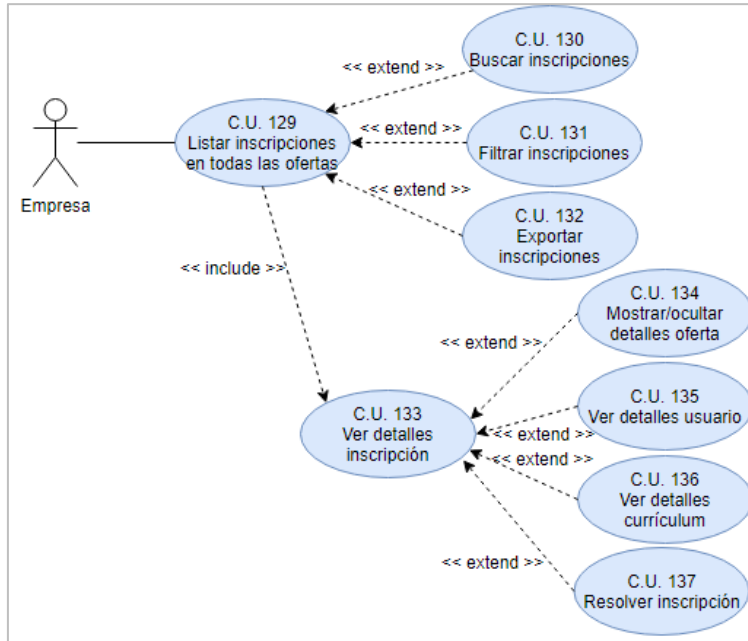


Figura 5.49: Diagrama C.U. Empresa: Incripciones.

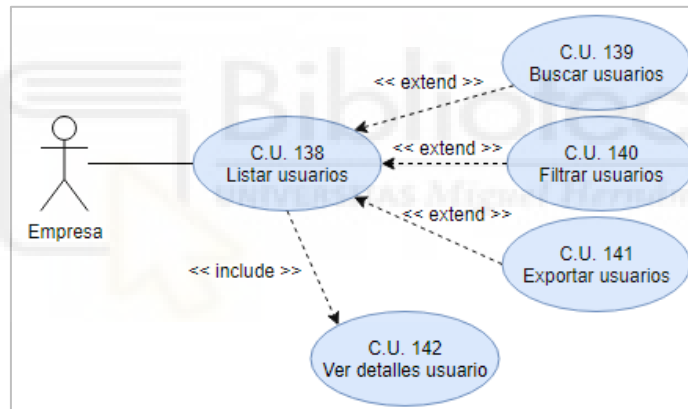


Figura 5.50: Diagrama C.U. Empresa: Buscar usuarios.

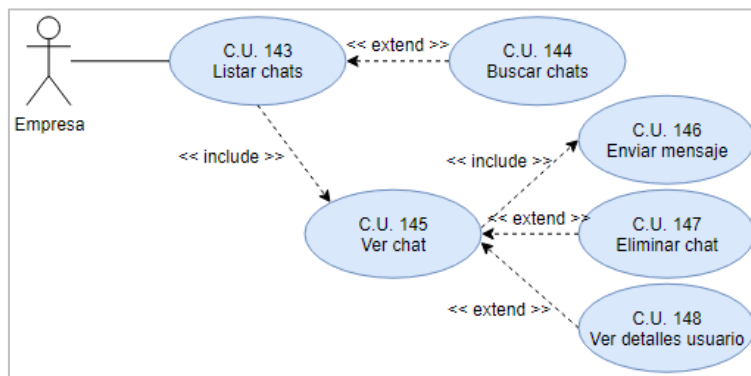


Figura 5.51: Diagrama C.U. Empresa: Chats.

5.3 Diseño

5.3.1 Base de datos

Toda la información de la aplicación se almacena en Firebase. La base de datos en tiempo real organiza la información mediante un objeto JavaScript en forma de árbol. Por tanto, no se trata de una base de datos relacional. La estructura en la que se ha organizado la base de datos es la mostrada en la Figura 5.52.

En el primer nivel de la base de datos se encuentran 5 objetos cada uno de los cuales contiene información referente a los administradores, las conversaciones de chats, todo lo referente a las empresas, las ofertas de empleo y la información de los usuarios, respectivamente. A continuación, se detalla cada uno de ellos.

- En *admins* se añade un nuevo subobjeto con su identificador único cada vez que se da de alta un nuevo administrador. En su interior está el objeto *profile*, que alberga todos los datos del perfil del administrador.
- En *chats* se crea un nuevo objeto con su propio identificador la primera vez que se inicia un chat entre un usuario y una empresa. Contiene el objeto *messages* donde se almacena cada conversación. Cada mensaje que se envía se añade a *messages* con su identificador y contiene la información relativa a cada mensaje. Cada usuario y empresa almacena en su propio objeto *chats* la información de la conversación, pero todos los mensajes están almacenados únicamente en este objeto independiente.

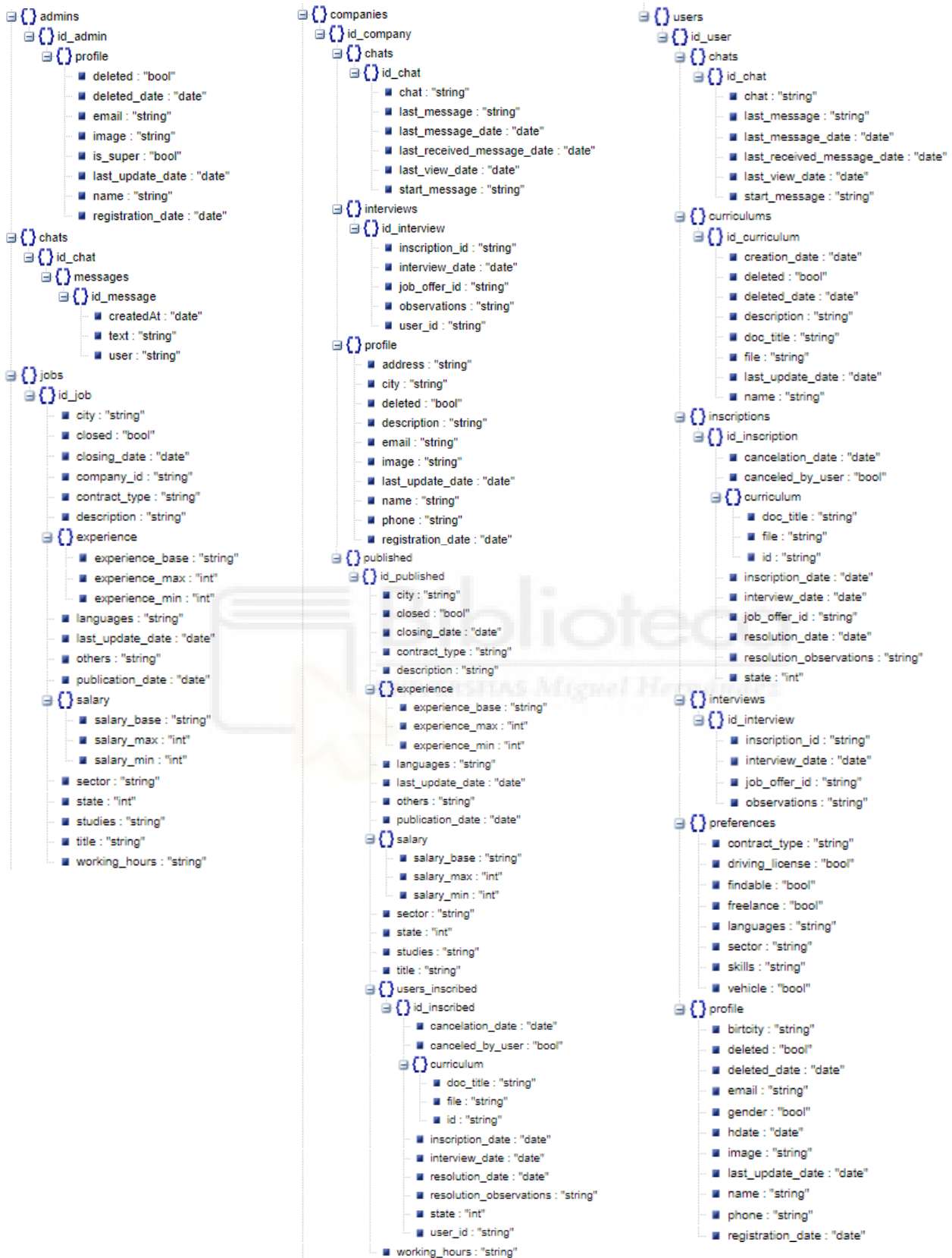


Figura 5.52: Diseño de la base de datos.

- En *companies* se crean nuevos objetos con un identificador único cada vez que se registra una empresa o es dada de alta por un administrador. Cada empresa almacena su información distribuida en cuatro objetos: *chats*, *interviews*, *profile* y *published*. Cada vez que la empresa inicia por primera vez una conversación con un usuario, o viceversa, se añade un nuevo objeto en *chats* con un identificador único. Este objeto contiene toda la información relacionada a la conversación. Cuando la empresa acepta una inscripción de un usuario de alguna de sus ofertas publicadas, se crea un nuevo objeto en *interviews*, donde se guarda toda la información relacionada con la entrevista. Los datos del perfil de la empresa se almacenan en el objeto *profile*. Cada vez que la empresa publica una nueva oferta de empleo, toda la información relacionada con la oferta y también la información de los usuarios que se vayan inscribiendo, se almacenada en *published*.
- En *jobs* se almacenan todas las ofertas de empleo de todas las empresas de la aplicación con su propio identificador. La información que se almacena es la misma que la del objeto *published* de las empresas, salvo que aquí no se almacena la información referente a las inscripciones de los usuarios. De este objeto se obtienen todas las ofertas que se muestran a los usuarios en el listado de ofertas de empleo disponibles.
- En *users* se añade un nuevo objeto con su identificador único cada vez que un usuario se registra o cuando un administrador le da de alta. Cada vez que el usuario inicia por primera vez una conversación con una empresa, o viceversa, se añade un nuevo objeto en *chats* con su identificador único, donde se almacena toda la información relacionada con la conversación. En *curriculums* se crear un nuevo objeto con su propio identificador cada vez que el usuario crea un nuevo currículum, y en su interior almacena la información referente a cada currículum. Cada vez que el usuario se inscribe en una oferta de empleo, se crea un nuevo objeto con su identificador en *inscriptions*, donde se guarda toda la información sobre la inscripción. Cuando las empresas aceptan una inscripción del usuario, se crea un nuevo

objeto con su identificador propio en *interviews*, donde está toda la información sobre la entrevista. Las preferencias del usuario se almacenan en el objeto *preferences* y los datos personales del usuario se almacenan en el objeto *profile*.

5.3.2 Disparadores

Las Cloud Functions de Firebase permiten programar funciones que se ejecutan cuando sucede una determinada acción, entre otros, en la base de datos.

sendWelcomeMailAd...	ref.create admins/(uid)	sendByeMailAdmin	ref.update admins/(uid)/profile/deleted	deleteAdminFromAu...	ref.update admins/(uid)/profile/deleted
sendWelcomeMailCo...	ref.create companies/(uid)	sendByeMailCompany	ref.update companies/(uid)/profile/deleted	deleteCompanyFrom...	ref.update companies/(uid)/profile/deleted
sendWelcomeMailUs...	ref.create users/(uid)	sendByeMailUser	ref.update users/(uid)/profile/deleted	deleteUserFromAuth	ref.update users/(uid)/profile/deleted

Figura 5.53: Disparadores de la aplicación.

Hay tres disparadores para cada tipo de actor de la aplicación:

- El primero se ejecuta cuando se crea un nuevo usuario, empresa o administrador y se añade en la base de datos. Firebase le envía un correo electrónico de bienvenida.
- El segundo se dispara cuando un usuario, empresa o administrador es eliminado de la aplicación. Se le envía un correo electrónico de despedida.
- El tercero también se activa cuando se elimina a un usuario, empresa o administrador. Su función es eliminarlos del sistema de autenticación para evitar que puedan iniciar sesión.

5.3.3 Almacenamiento

Todos los archivos de la aplicación se almacenan en el sistema de almacenamiento de Firebase.

Los currículums de los usuarios en formato PDF se almacenan en el directorio *docs/*, mientras que todas las imágenes de los perfiles de los usuarios, empresas y administradores se almacenan en *images/*.

La forma de acceder la aplicación a los archivos es mediante su url, que es la que se almacena en la base de datos para posteriormente tener acceso a ellos.

5.3.4 Navegación

De acuerdo con todo lo descrito hasta ahora, en el mapa de la Figura 5.54. se muestran las pantallas de la aplicación ordenadas por actores y donde se puede observar la navegación entre ellas.

Nada más abrir la aplicación se muestra brevemente la pantalla de bienvenida. Si no se está autenticado previamente, se inicia el proceso de inicio de sesión o registro. Seguidamente se accede al apartado de los administradores, de las empresas o de los usuarios, según el rol.

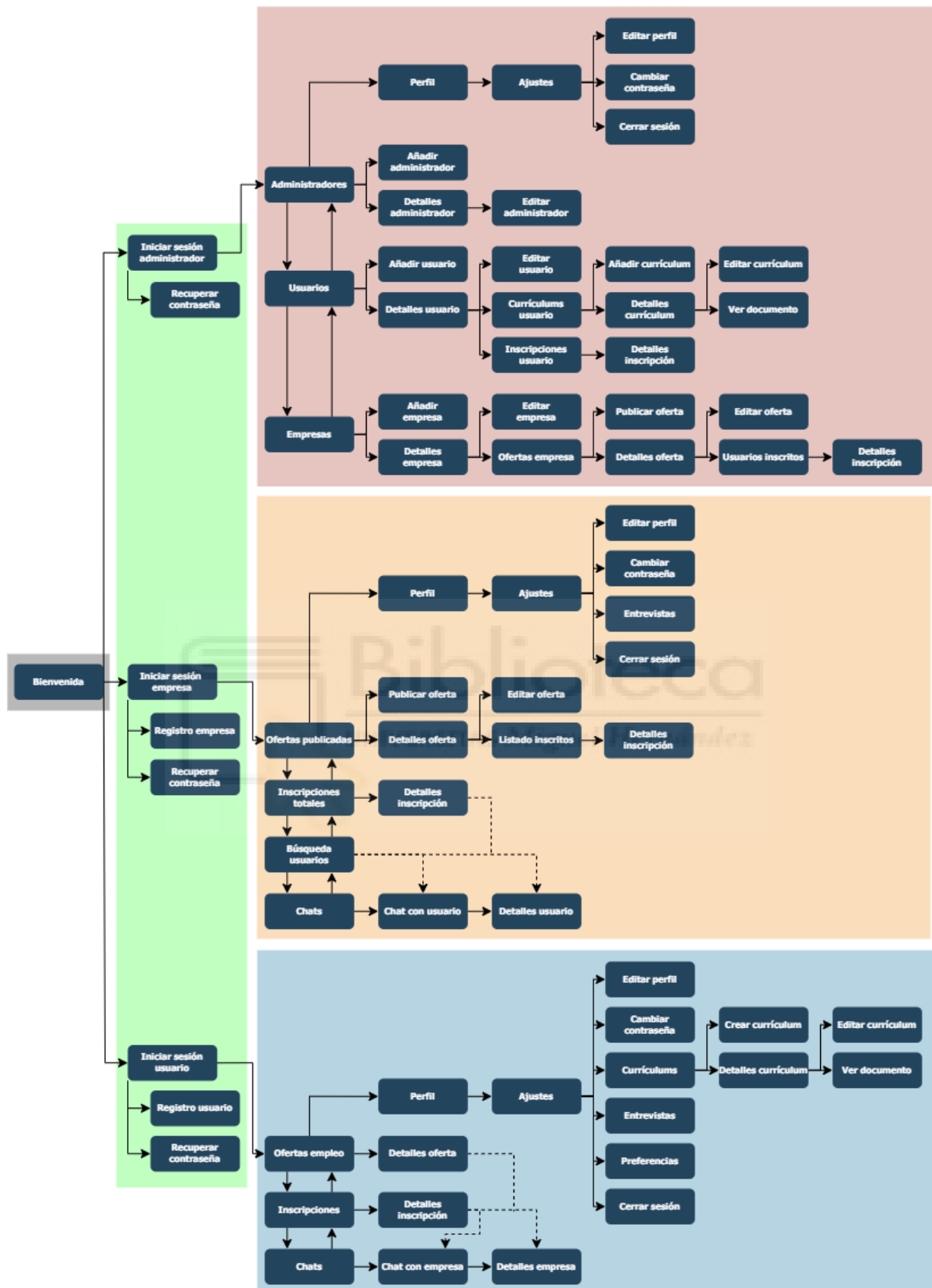


Figura 5.54: Mapa de pantallas y navegación de la aplicación.

5.4 Implementación

En esta etapa del proceso del desarrollo del proyecto, se ha llevado a cabo la implementación del código de la aplicación. Se han seguido las pautas recogidas en la captura de requisitos y se ha plasmado la interfaz de la aplicación según el diseño realizado en el prototipo.

Como resultado de la implementación, se muestran a continuación una serie de capturas de la aplicación ya terminada.



Figura 5.55: Bienvenida, inicio de sesión y registro con correo de bienvenida.

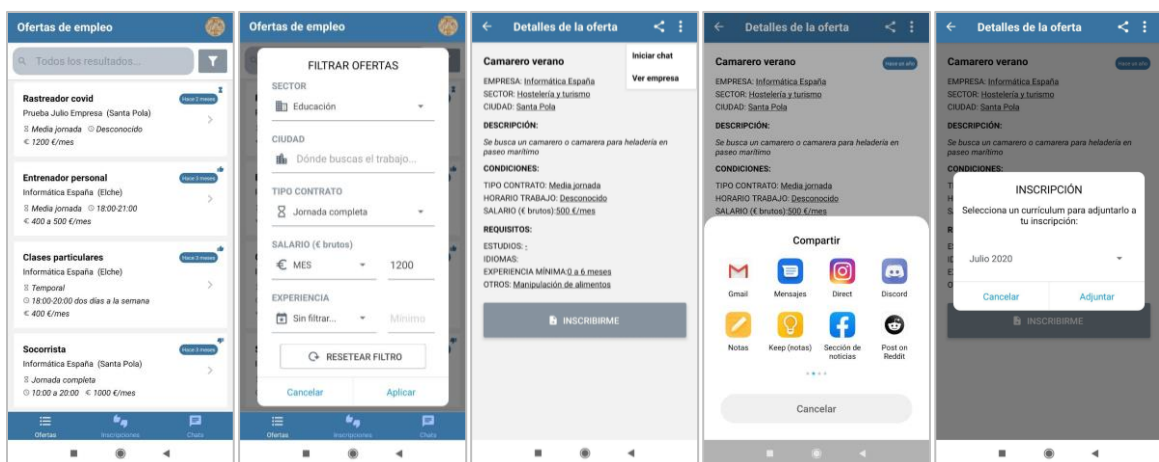


Figura 5.56: Ofertas de empleo, filtro, detalles, compartir e inscripción del usuario.

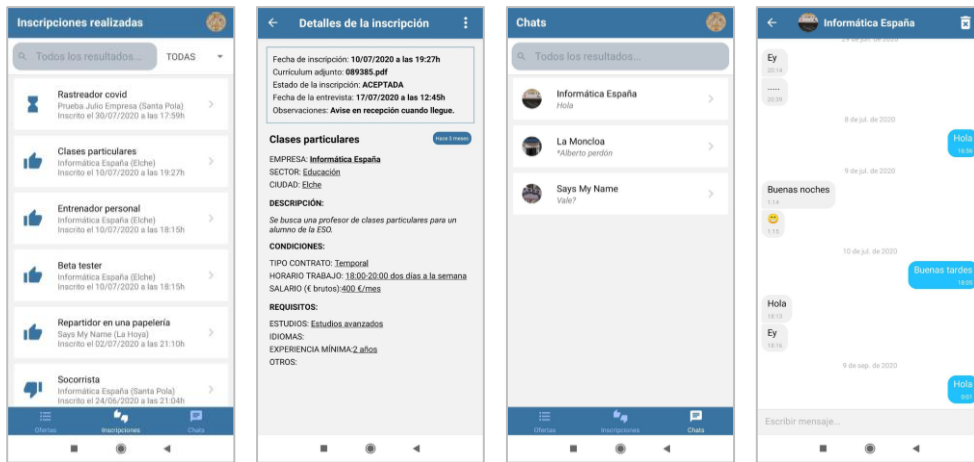


Figura 5.57: Inscripciones y chats del usuario.

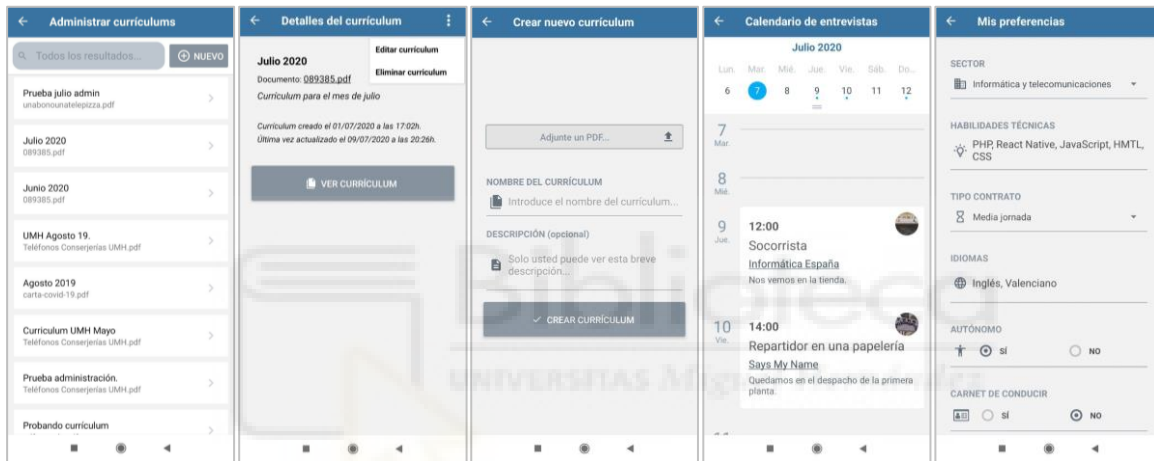


Figura 5.58: Currículums, entrevistas y preferencias del usuario.

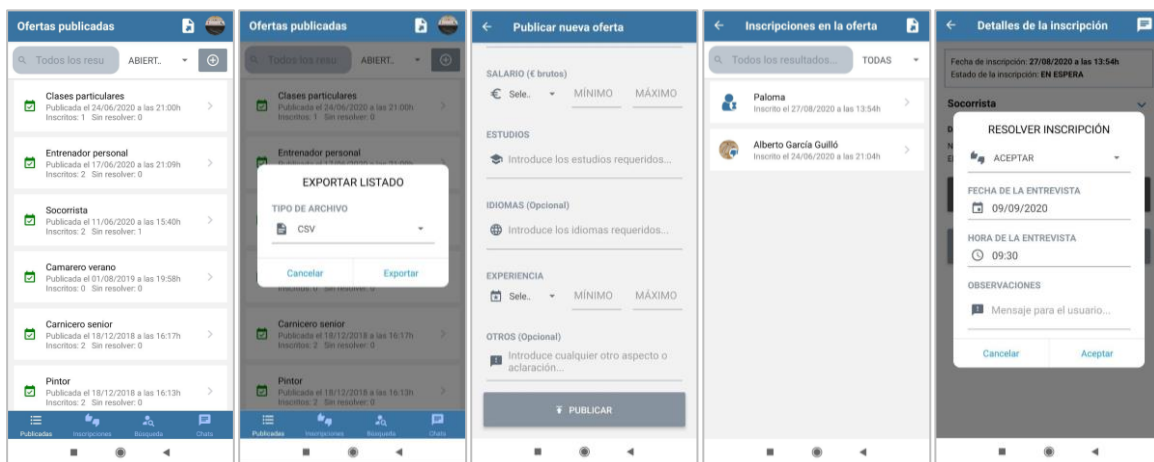


Figura 5.59: Ofertas, exportar, publicar e inscripciones de la empresa.

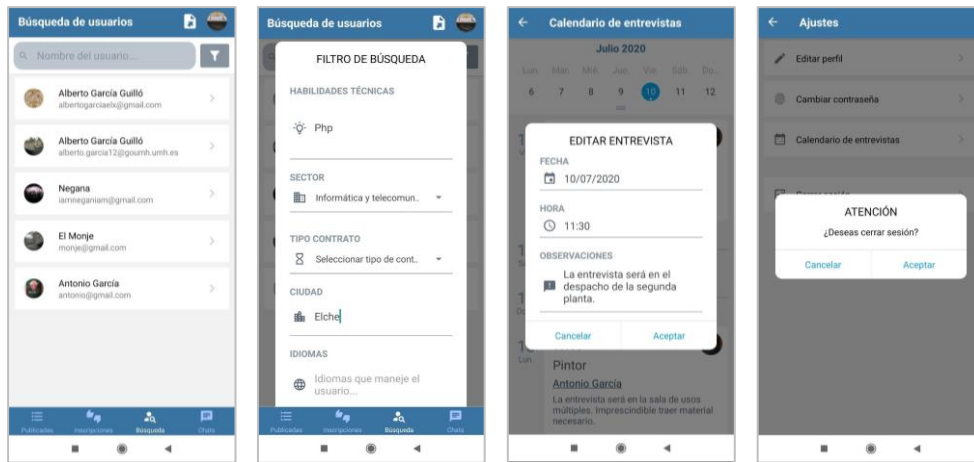


Figura 5.60: Búsqueda de usuarios con filtro, entrevistas y cerrar sesión de la empresa.

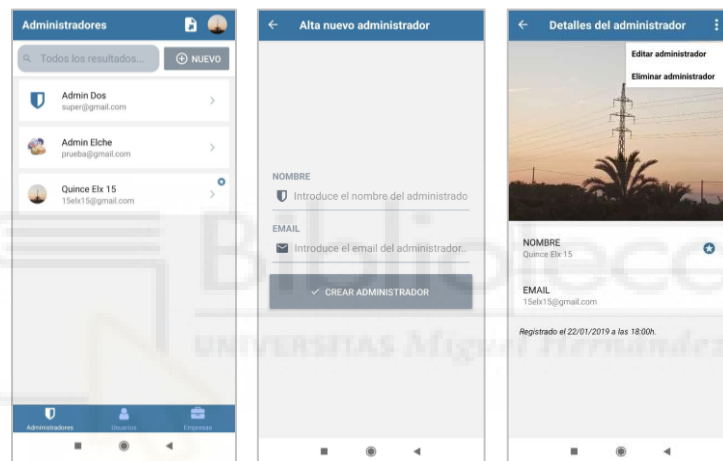


Figura 5.61: Gestión de administradores del administrador.

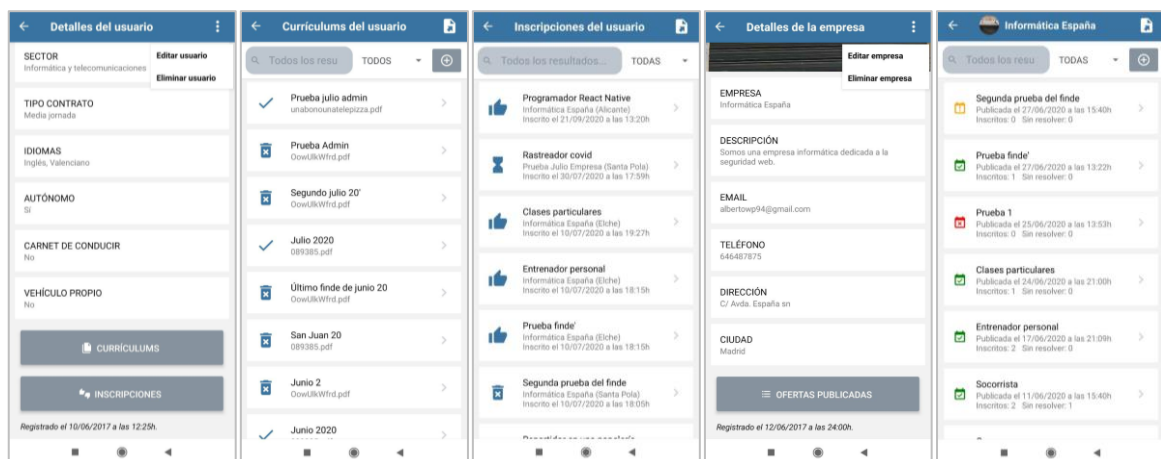


Figura 5.62: Gestión de usuarios y empresas del administrador.

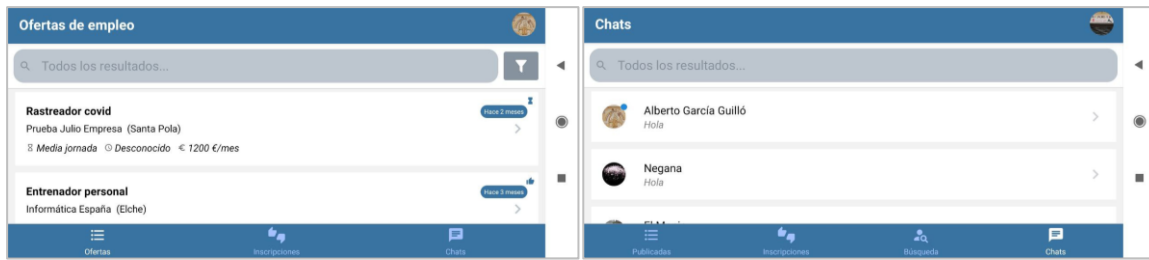


Figura 5.63: Modo apaisado de la aplicación.

Hay que destacar también que la interfaz se adapta perfectamente a diferentes tamaños y cambios de orientación del dispositivo.

5.5 Validación

A medida que se han ido terminando las diferentes iteraciones, se ha ido probando la aplicación. Se ha comprobado el correcto funcionamiento de la autenticación de los actores. También se ha verificado que la creación, listado, edición y eliminación de los distintos componentes es correcta.

La aplicación muestra mensajes de éxito cuando se completa satisfactoriamente una acción y mensajes de error en los formularios cuando sucede algún fallo. También se muestra un spinner cuando la aplicación está cargando o procesando un contenido.

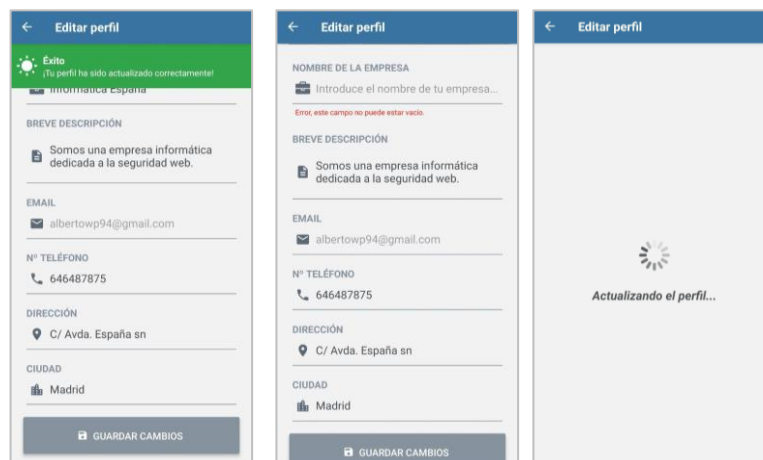


Figura 5.64: Mensajes éxito, error y espera de la aplicación.

5.6 Implantación

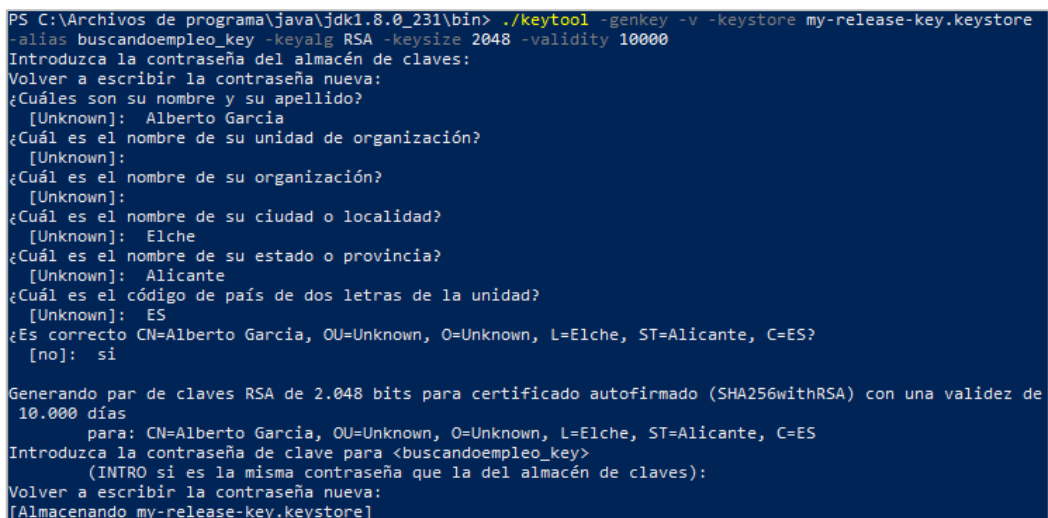
Después del análisis de requisitos, el diseño, la implementación y la validación, el último paso para completar el proceso, es la publicación de la aplicación en Google Play de Android o la App Store de iOS. No se ha publicado en ninguna de las tiendas oficiales porque no es uno de los objetivos del este trabajo. Pero sí se explica el proceso que hay que seguir para obtener los archivos que se necesitan en caso de querer publicar la aplicación.

5.6.1 Generar APK (Android - Google Play)

En el caso de Android, para distribuir la aplicación a través de Google Play, se requiere que todas las aplicaciones estén firmadas con un certificado digital. Por tanto, hay que generar una versión firmada del APK de la aplicación. [27]

Para generar una clave firmada, se utiliza la herramienta *keytool* proporcionada por Java Run Environment (JRE). El comando que se debe ejecutar es:

```
keytool -genkey -v -keystore my-release-key.keystore -alias my-key-alias -keyalg RSA -keysize 2048 -validity 10000
```



```
PS C:\Archivos de programa\java\jdk1.8.0_231\bin> ./keytool -genkey -v -keystore my-release-key.keystore
-alias buscandoempleo_key -keyalg RSA -keysize 2048 -validity 10000
Introduzca la contraseña del almacén de claves:
Volver a escribir la contraseña nueva:
¿Cuáles son su nombre y su apellido?
 [Unknown]: Alberto García
¿Cuál es el nombre de su unidad de organización?
 [Unknown]:
¿Cuál es el nombre de su organización?
 [Unknown]:
¿Cuál es el nombre de su ciudad o localidad?
 [Unknown]: Elche
¿Cuál es el nombre de su estado o provincia?
 [Unknown]: Alicante
¿Cuál es el código de país de dos letras de la unidad?
 [Unknown]: ES
¿Es correcto CN=Alberto García, OU=Unknown, O=Unknown, L=Elche, ST=Alicante, C=ES?
 [no]: si
Generando par de claves RSA de 2.048 bits para certificado autofirmado (SHA256withRSA) con una validez de
10.000 días
 para: CN=Alberto García, OU=Unknown, O=Unknown, L=Elche, ST=Alicante, C=ES
Introduzca la contraseña de clave para <buscandoempleo_key>
 (INTRO sí es la misma contraseña que la del almacén de claves):
Volver a escribir la contraseña nueva:
[Almacenando my-release-key.keystore]
```

Figura 5.65: Generar clave firmada mediante *keytool*.

Hay que indicar el nombre del fichero donde almacenarla, un alias, algoritmo RSA, un tamaño de clave de 2048 bits y una validez de 10000 días. Con ello se consigue crear una clave firmada digitalmente.

El siguiente paso es incluir la clave generada en la aplicación. Para ello, hay que copiar el archivo `my-release-key.keystore` dentro del directorio `android/app` del proyecto. A continuación, hay que editar el archivo `android/gradle.properties` e indicar el archivo con la clave, el alias y ambas contraseñas.

```
MYAPP_RELEASE_STORE_FILE=my-release-key.keystore
MYAPP_RELEASE_KEY_ALIAS=my-key-alias
MYAPP_RELEASE_STORE_PASSWORD=*****
MYAPP_RELEASE_KEY_PASSWORD=*****
```

Seguidamente, hay que añadir la configuración de la firma en el código de la aplicación, editando el archivo `android/app/build.gradle`.

```
android {
    ...
    defaultConfig { ... }
    signingConfigs {
        release {
            if (project.hasProperty('MYAPP_RELEASE_STORE_FILE')) {
                storeFile file(MYAPP_RELEASE_STORE_FILE)
                storePassword MYAPP_RELEASE_STORE_PASSWORD
                keyAlias MYAPP_RELEASE_KEY_ALIAS
                keyPassword MYAPP_RELEASE_KEY_PASSWORD
            }
        }
    }
    buildTypes {
        release {
            ...
            signingConfig signingConfigs.release
        }
    }
}
```

El último paso es generar el APK definitivo ejecutando el comando `gradlew assembleRelease` dentro del directorio Android del proyecto.

Seguidos todos los pasos, se dispone del APK para poder distribuirlo a través de Google Play.

5.6.2 Generar IPA (iOS - App Store)

Por su parte, el proceso de publicar la aplicación en la App Store es algo más complejo. Una vez se tiene el proyecto de React Native terminado, el siguiente paso es construir el archivo .ipa (iOS Application File), el cual es el que se suba finalmente a la App Store. Dado que no se dispone de un dispositivo MAC, requisito indispensable para poder construir el archivo .ipa, no se pueden aportar imágenes del proceso, pero se describe a continuación como sería. [28]

Es necesario disponer de una cuenta de Desarrollador de Apple y otra cuenta de iTunes Connect. Primero hay que crear una aplicación en iTunes Connect y después crear un certificado para la propia cuenta. Seguidamente se crea un perfil de distribución (Distribution Provisioning Profile) para la aplicación. A continuación, se crea, firma y empaqueta la aplicación desde Xcode. Como resultado se obtiene el archivo .ipa. Hechos todos los pasos, solo quedaría subir el archivo .ipa a la App Store.

CAPÍTULO 6 - CONCLUSIONES Y PROPUESTAS

6.1 Conclusiones

El desarrollo de aplicaciones móviles es un mercado en auge debido a que el uso de los dispositivos móviles se ha disparado hasta cifras de récord en los últimos años. Para el desarrollo de aplicaciones móviles un aspecto clave consiste en que exista un equilibrio entre rendimiento y costes de desarrollo.

React Native permite crear aplicaciones móviles multiplataforma mediante el desarrollo de un único proyecto programado en JavaScript. El uso de Redux aporta simplicidad a la hora de tratar el estado de la aplicación. Y Firebase es una herramienta que ofrece multitud de productos para el desarrollo de aplicaciones móviles, permite tener una base de datos y además hace que se pueda prescindir de un servidor.

Como ejemplo de uso de estas tecnologías, se ha desarrollado una aplicación de búsqueda de empleo. Se han seguido las fases que componen el ciclo de vida del software: planificación del trabajo, análisis de requisitos, diseño, implementación, validación e implantación.

Como experiencia personal me ha servido para conocer el proceso que compone todo el desarrollo de un proyecto de ingeniería del software, en este caso para el desarrollo de una aplicación móvil. He comprobado que es importante dedicar tiempo al principio a identificar bien los requisitos y diseñar de forma correcta la estructura de la aplicación. Un buen prototipo ayuda visualmente a entender al cliente (en este caso era principalmente yo, pero también mi tutora ha aportado ideas a la hora de incluir nuevas características a la aplicación) y al desarrollador el aspecto visual y la funcionalidad que tendrá finalmente la aplicación.

Ha merecido la pena el esfuerzo de estos meses para completar con éxito el desarrollo de una aplicación móvil multiplataforma mediante el uso de tecnologías que eran desconocidas para mí hasta el inicio del TFG. He podido lograrlo sobre todo gracias a los conocimientos y habilidades adquiridos estos años en el Grado.

6.2 Posibles desarrollos futuros

De cara a futuras mejoras de la aplicación, se podría ampliar su funcionalidad mediante la implementación de:

- Registro mediante Google/Redes sociales, aprovechando que Firebase lo ofrece como método de autenticación.
- Añadir notificaciones push para avisar a empresas o usuarios de nuevas inscripciones, resoluciones o mensajes de chats recibidos.
- Apartado donde se muestren gráficas para que las empresas vean sus ofertas de empleo más populares.
- Posibilidad de visualizar las ofertas de empleo en un mapa.
- Realizar una evaluación de la usabilidad de la aplicación.
- Incluir alguna característica que haga la aplicación accesible. Como por ejemplo ayudas auditivas para que al pulsar sobre algún elemento de la interfaz de la aplicación, salte un asistente de voz que lea el contenido.
- Adaptar la aplicación a futuros cambios en las reglas de interfaces de Android/iOS.
- Investigar nuevas formas de programar aplicaciones con React Native, por ejemplo, mediante Expo.

Y en cuanto a las tecnologías empleadas, seguir de cerca la evolución de React Native, Redux y Firebase, para intentar mejorar la aplicación, o para futuras aplicaciones desarrolladas con estas tecnologías.

CAPÍTULO 7 - BIBLIOGRAFÍA

[1] Informe Ditrendia Mobile en España y en el Mundo 2017

https://www.amic.media/media/files/file_352_1289.pdf

Marzo 2020

[2] 1.6 Aplicaciones web, nativas e híbridas

Autor: Antonio Peñalver

Asignatura: Desarrollo de aplicaciones para dispositivos móviles (3º GIITI)

Año: 2015

[3] Comparativa entre React Native y diferentes frameworks de programación

<https://openwebinars.net/blog/comparativa-react-native-y-diferentes-frameworks/>

Marzo 2020

[4] React Native

<https://facebook.github.io/react-native/>

Marzo 2020

[5] Redux

<https://redux.js.org/>

Marzo 2020

[6] React-Redux

<https://react-redux.js.org/>

Marzo 2020

[7] Redux-Thunk

<https://github.com/reduxjs/redux-thunk>

Marzo 2020

[8] Introduction to Firebase

<https://hackernoon.com/introduction-to-firebase-218a23186cd7>

Marzo 2020

[9] Firebase

<https://firebase.google.com/>

Marzo 2020

[10] React Navigation

<https://reactnavigation.org/>

Abril 2020

[11] React Native Elements

<https://react-native-training.github.io/react-native-elements/>

Abril 2020

[12] React Native Gifted Chat

<https://github.com/FaridSafi/react-native-gifted-chat>

Abril 2020

[13] Node Package Manager

<https://www.npmjs.com/>

Marzo 2020

[14] React Native Calendars

<https://github.com/wix/react-native-calendars>

Abril 2020

[15] Lodash

<https://lodash.com/docs>

Mayo 2020

[16] Moment JS

<https://momentjs.com/docs>

Marzo 2020

[17] Sublime Text 3

<https://www.sublimetext.com/3>

Marzo 2020

[18] GitHub

<https://github.com>

Abril 2020



[19] Microsoft Word

<https://products.office.com/es-es/word>

Marzo 2020

[20] Balsamiq Mockups 3 for Desktop

<https://balsamiq.com/products/>

Abril 2020

[21] GanttProject

<https://www.ganttproject.biz/>

Marzo 2020

[22] Draw.io

<https://app.diagrams.net/>

Marzo 2020

[23] Acer Aspire 5750G (Foto)

https://xtremmedia.com/ACER_ASPIRE_5750G_2454G50MN.html

Julio 2020

[24] Xiaomi Redmi Note 8 Pro (Foto)

<https://www.mi.com/es/redmi-note-8-pro/>

Julio 2020

[25] 01 - Introducción a la I.S (II)

Autor: Jesús Javier Rodríguez Sala

Asignatura: Ingeniería del Software (2º GIITI)

Año: 2013

[26] Implementación y debugging. Capítulo 1. Ciclo de vida del software

<https://ingsw.pbworks.com/f/Ciclo+de+Vida+del+Software.pdf>

Agosto 2020

[27] Generating Signed APK (Documentación oficial React Native)

<https://facebook.github.io/react-native/docs/signed-apk-android.html>

Agosto 2020

[28] Aguirre G. (2017), Cómo subir tu aplicación de React Native al App Store

<https://medium.com/underscopeio/c%C3%B3mo-subir-tu-aplicaci%C3%B3n-de-react-native-al-app-store-b4df02093eba>

Agosto 2020