ULPGC
Universidad de Las Palmas de Gran Canaria

Escuela de Ingeniería Informática

COT (Clock-in On Time): Aplicación para control de horario de empleados que prestan servicios fuera del lugar de trabajo

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Belén Pulido Rivero

TUTORIZADO POR:

José Miguel Santos Espino

Fecha [09/2020]

AGRADECIMIENTOS

En primer lugar, quiero agradecer a todas aquellas personas que me han apoyado y animado en buenos y sobre todo en malos momentos. Por un lado, agradecer a mi familia, que siempre me ha apoyado en todo lo referente a mis estudios, lo cual ha sido un impulso y motivación para llegar hasta estas alturas.

Por otro lado, me gustaría agradecer a todos los profesores que me han impartido clases durante todos estos años y en especial, a mi tutor de TFT, José Miguel Santos Espino por haberme ayudado a lo largo del desarrollo de mi TFT y por darme muy buenos consejos para el mismo.

Finalmente, agradecer a los compañeros que me he tropezado a lo largo de estos años, los cuales me han ayudado en todo lo que he necesitado sin dudar y se han convertido en muy buenos amigos.

En definitiva, muchas gracias a todos.

Belén Pulido Rivero

RESUMEN

El Real Decreto-ley 8/2019 establece medidas urgentes de protección social y de lucha contra la precariedad laboral. Una de sus principales medidas es la obligación por parte de las empresas de mantener un registro de la jornada de trabajo de sus empleados (fichaje de entrada y salida). Este registro se vuelve complicado para los trabajadores que prestan sus servicios fuera del centro de trabajo, por ejemplo, transportistas o comerciales que trabajan en la calle. Este TFG consiste en el diseño de una aplicación, Clock-in On Time, orientada a facilitar el fichaje de este tipo de empleados desde un dispositivo móvil. El trabajo incluye la implementación de un prototipo funcional para la plataforma Android.

ABSTRACT

Royal Decree-Law 8/2019 establishes urgent measures for social protection and the fight against precarious employment. One of its main measures is the obligation for companies to keep a record of their employees' working hours (clocking-in and clocking-out). This record becomes complicated for workers who provide their services outside the work center, for example, transporters or commercials who work in the street. This TFG consists of the design of an application, Clock-in On Time, oriented to facilitate the clocking-in of this type of employees from a mobile device. The work includes the implementation of a functional prototype for the Android.

ÍNDICE DE CONTENIDO

1. Introducción 1

1.1. Antecedentes 1

1.2. La nueva regulación de la jornada de trabajo 1

1.3. Motivación y objetivos 2

1.4. Estructura de la memoria 2

2. Aspectos preliminares 4

2.1. Justificación de competencias 4

2.1.1. Competencia IS02 4

2.1.2. Competencia IS04 4

2.1.3. Competencia IS06 4

2.2. Aportaciones 4

3. Descripción del proyecto 5

3.1. Normativa y legislación 5

3.1.1. Reglamento general de protección de datos (RGPD) 5

3.1.2. Ley orgánica de protección de datos y garantía de derechos digitales (LOPDGDD) 5

3.1.3. Real Decreto-Ley 8/2019: Medidas urgentes de protección social y de lucha contra la precariedad laboral en la jornada de trabajo 5

3.2. Descripción del TFT 6

3.2.1. Descripción y características 6

3.3. Herramientas, servicios y tecnologías usadas 7

3.3.1. Herramientas 7

3.3.2. Servicios 9

3.3.3. Tecnologías 10

3.4. Metodología y planificación del proyecto 10

3.4.1. Scrum 11

3.4.2. Planificación del proyecto 12

4. Análisis 14

4.1. Análisis de mercado 14

4.1.1. Software para control horario 14

4.1.2. Estudio de mercado 14

4.1.3. Cuadro comparativo 17

4.1.4. Entrevistas a empresarios 18

4.1.5. Modelo de negocio 19

4.2. Análisis de requisitos 20

4.2.1. Stakeholders 20
4.2.2. Requisitos 21
5. Historias técnicas 24
5.1.1. Historias de usuario 25
5.1.2. Diagrama de casos de uso 35
5.1.3. Especificaciones de casos de uso 36
6. Diseño 41
6.1. Diseño de la interfaz de usuario 41
6.1.1. Boceto del proyecto 41
6.1.2. Elección de tipografía y gama de colores 43
6.2. Diseño del software 44
6.2.1. Arquitectura del proyecto 44
6.2.2. Diagrama de clases 45
6.3. Diseño de la base de datos 46
7. Estructura y elementos del proyecto 48
7.1. Estructura del proyecto Android 48
7.1.1. Archivos a nivel de proyecto 48
7.1.2. Todos los módulos del proyecto Android 48
7.2. Dependencias del proyecto 49
7.2.1. Dependencias inyectadas en el proyecto 49
7.3. Elementos importantes en Android Studio 50
8. Desarrollo del proyecto 51
8.1. Historias de usuario desarrolladas en el proyecto 51
8.2. Sprints realziados 52
8.2.1. Sprint 0 52
8.2.2. Sprint 1 54
8.2.3. Sprint 2 60
8.2.4. Sprint 3 68
8.3. Validación 74
9. Resultados, conclusiones y trabajo futuro 75
9.1. Resultados 75
9.2. Conclusiones 75
9.3. Trabajo futuro 76
10. Bibliografía 77

ÍNDICE DE TABLAS

Tabla 1. Planificación del proyecto *13*
Tabla 2. Tabla comparativa de aplicaciones *18*
Tabla 3. Pila de sprint 0 ..... *52*
Tabla 4. Pila de sprint 1 ..... *54*
Tabla 5. Pila de sprint 2 ..... *60*
Tabla 6. Pila de sprint 3 ..... *68*
Tabla 7. Pila de producto (Product Backlog). *83*

ÍNDICE DE ILUSTRACIONES

Ilustración 1. Logo del TFT 6
Ilustración 2. XML y Java para Android 10
Ilustración 3. Proceso de Scrum [20] 12
Ilustración 4. Aplicación "Control Laboral". 15
Ilustración 5. Aplicación "Control horario" (BIXPE) 15
Ilustración 6. Aplicación "Intratime" 16
Ilustración 7. Aplicación "Ficha.work" 16
Ilustración 8. Aplicación "Sesame Time" 17
Ilustración 9. Bussiness Model Canvas del proyecto. 20
Ilustración 10. Lean Canvas del proyecto 20
Ilustración 11. Diagrama de casos de uso (Visitante). 35
Ilustración 12. Diagrama de casos de uso (Empleado). 35
Ilustración 13. Diagrama de casos de uso (Administrador). 36
Ilustración 14. Boceto: Pantalla inicial 41
Ilustración 15. Boceto: Olvidé mi contraseña 41
Ilustración 16. Boceto: Pantalla inicial 41
Ilustración 17. Boceto: Listado de fichajes 42
Ilustración 18. Boceto: Perfil 42
Ilustración 19. Boceto: Justificación de falta 42
Ilustración 20. Boceto: Acerca de 42
Ilustración 21. Boceto: Modificar perfil 42
Ilustración 22. Boceto: Ayuda 42
Ilustración 23. Boceto: Listado de justificantes 42
Ilustración 24. Colores más elegidos por mujeres. [34] 43
Ilustración 25. Colores más elegidos por hombres [35] 43
Ilustración 26. Arquitectura del patrón MVC 44
Ilustración 27. Explicación del funcionamiento de MVC [31]. 44
Ilustración 28. Diagrama de comunicación de la aplicación y Firebase Authentication. 45
Ilustración 29. Reglas introducidas para dar segurdad a la base de datos Firebase. 45
Ilustración 30. Diagrama de clases del TFT 46
Ilustración 31. Estructura de la base de datos del proyecto. 46
Ilustración 32. TFT: Menú superior (xml) 55
Ilustración 33. TFT: Opciones del Menú superior (parte lógica). 55
Ilustración 34. TFT: Menú inferior (xml) 56
Ilustración 35. TFT: Menú inferior (parte lógica) 56
Ilustración 36. TFT: Diálogo de reconocimiento dactilar 57
Ilustración 37. TFT: Pantalla inicial 57
Ilustración 38. Diálogo para reconocimiento dactilar (Fichero HomeActivity.java) 57
Ilustración 39. Entidad TimeCard. 58
Ilustración 40. Clase Localización 58
Ilustración 41. TFT: Cuadro de diálogo recordatorio para obtener ubicación. 59
Ilustración 42. TFT: Justificar falta. 59
Ilustración 43. Cuadro de diálogo para una justificación realizada. 59
Ilustración 44. Cuadro de diálogo sobre la realización de la justificación 59
Ilustración 45. TFT: Código se la función selectFile 60
Ilustración 46. TFT: Código de la función uploadFile 60
Ilustración 47. TFT: Pantalla para reestablecer contraseña. 61
Ilustración 48. TFT: Cuadro de diálogo de cambio de contraseña 61
Ilustración 49. TFT: Correo para reestablecer contraseña 61

Ilustración 50. TFT: Distribución de pantalla de ayuda. 62
Ilustración 51. TFT: Pantalla de ayuda 62
Ilustración 52. TFT: row_recyler.xml (cardView para fichaje) 62
Ilustración 53. TFT: Listado de fichajes 63
Ilustración 54. TFT: Método que añade los fichajes al adaptador (TimeCardListActivity.java) 63
Ilustración 55. TFT: Adaptador para la lista de fichajes 64
Ilustración 56. Búsqueda de fichaje con falta justificada. 64
Ilustración 57. TFT: Calendario para buscar fecha de fichaje 64
Ilustración 58. Búsqueda de fichaje. 64
Ilustración 59. TFT: Método que muestra el calendario al usuario 65
Ilustración 60. TFT: Método que realiza filtra la lista de fichajes 65
Ilustración 61. TFT: row_recyler_justify.xml (cardView para justificantes) 66
Ilustración 62. TFT: Listado de justificantes 66
Ilustración 63. TFT: Adaptador para el listado de justificantes 67
Ilustración 64. TFT: Método que añade los justificantes al adaptador 67
Ilustración 65. TFT: Detalles de fichaje (Falta justificada) 68
Ilustración 66. TFT: Detalles de fichaje (Falta no justificada) 68
Ilustración 67. TFT: Ejemplo de uso del método putExtra() 69
Ilustración 68. TFT: Ejemplo de cómo se obtiene información introducida con putExtra() 69
Ilustración 69. TFT: Distribución de los componentes visuales para pantalla de perfil 70
Ilustración 70. TFT: Método que realiza una consulta a la BD para obtener la información del usuario ... 70
Ilustración 71. TFT: Editar perfil 71
Ilustración 72. TFT: Método que carga la información del usuario para modificarlos 71
Ilustración 73. TFT: Acciones de los botones de la pantalla de editar perfil 72
Ilustración 74. TFT: Apertura de un justificante PDF 72
Ilustración 75. TFT: Visualizar contraseña 73
Ilustración 76. TFT: Método para enseñar y ocultar la contraseña. 73
Ilustración 77. TFT: Ubicacion en Maps 73
Ilustración 78. TFT: Ver ubicación (Botones) 73
Ilustración 79. TFT: Visualización de las ubicaciones (código) 74

1. Introducción

1.1. Antecedentes

En los años anteriores, el control de horario laboral no era obligatorio en España y muchas de las empresas optaban por registrar el control de horario de sus empleados a través de plantillas Excel o incluso a lápiz y papel, sin embargo, algunas empresas que usaban estos mecanismos en su momento llegaron a perder el documento donde registraban estos datos o incluso, no realizaban copias de seguridad de los documentos Excel, lo cual dificultaba dicho control. Además, no existía un límite en materia de jornada y la inexistencia de seguridad jurídica tanto para las personas trabajadoras como para las empresas unido a ello la falta de control por parte de la inspección de trabajo y seguridad social.

Es por este motivo que el Real-Decreto Ley 8/2019 regula aspectos de protección social y de lucha contra la precariedad laboral en la jornada de trabajo, de manera que, hace obligatorio el registro de control de horario de los empleados con el fin de potenciar las políticas sociales.

1.2. La nueva regulación de la jornada de trabajo

El Real Decreto-Ley 8/2019 introduce importantes novedades sobre las medidas urgentes de protección social y de lucha contra la precariedad laboral en la jornada de trabajo, entre ellas la obligación de registro de horarios. Concretamente, el artículo 10 de esta ley, modifica el artículo 34 del Estatuto de los Trabajadores para incluir un nuevo apartado 9, que establece la obligatoriedad del registro diario de jornada (deberá incluir el horario concreto de inicio y finalización de la jornada de trabajo de cada persona trabajadora). Las sanciones previstas se clasifican en tres grupos, atendiendo a su gravedad [1]:

- **Infracciones leves**: 60€ - 625€ (Si se falta a la información de las condiciones laborales más importantes).
- **Infracción grave**: 625€ - 6.250€ (Si se detectan irregularidades en el registro de las horas).
- **Infracciones muy graves**: 6.250€ - 187.515€ (Si las irregularidades acumulan grandes diferencias).

Por otro lado, la empresa está obligada a conservar durante cuatro años los datos de la jornada laboral de sus empleados. Este punto es de obligatorio cumplimiento desde el 13 de mayo de 2019.

Además, existen empresas que cuentan con dos perfiles de empleados: Empleados que trabajan dentro del espacio de trabajo o empresa, y empleados que no trabajan dentro del espacio de trabajo, cuyo tiempo de trabajo cuenta desde que sale de su casa hasta que regresan. El control de este último tipo de empleado se puede realizar mediante dispositivos que cuenten con funcionalidades de geolocalización, respetando la ley de protección de datos. Es por ello qué surge la necesidad de aplicaciones que ayuden al empresario y al empleado llevar a cabo el registro de horario para cumplir esta nueva ley.

A su vez, hay excepciones a la aplicación del ámbito subjetivo de esta ley, como por ejemplo el personal de alta dirección, los trabajadores domésticos, los penados en instituciones penitenciarias, deportistas profesionales, artistas en espectáculos públicos, los trabajadores con discapacidad en centros especiales de empleo, etc.

Actualmente, debido a la situación generada por la pandemia de COVID-19, gran parte de la sociedad se ha visto obligada a adaptarse de forma rápida e inesperada al teletrabajo, por lo que, las tecnologías (dispositivos y aplicaciones) que ayudan a cumplir esta norma, juegan un papel bastante importante, ya que, en estos tiempos, resulta difícil y en algunos casos imposible, fichar de manera presencial.

1.3. Motivación y objetivos

El principal motivo para realizar este TFT fue aprender a realizar aplicaciones móviles desde cero. En el grado de ingeniería informática de la ULPGC existen asignaturas en las que se enseñan a realizar aplicaciones de escritorio básicas o aplicaciones webs, sin embargo, no existe ninguna asignatura dedicada al diseño e implementación de aplicaciones móviles y como me gustan los retos, decidí adentrarme en este mundo.

Hoy en día, el uso de los Smartphones representan muchos beneficios para la vida cotidiana. Estos dispositivos, además de permitir realizar las funciones principales de los teléfonos móviles (realizar llamadas, mandar mensajes de texto, etc), permiten hacer uso de aplicaciones móviles, como, por ejemplo, aplicaciones para realizar compras on-line, aplicaciones para reproducir películas y series o aplicaciones de mensajería para comunicarse con personas que estén en cualquier parte del mundo, lo cual resulta muy cómodo para sus usuarios.

En cuanto a los objetivos que se pretenden cubrir con la elaboración del trabajo se corresponden con la aplicación de todos los conocimientos adquiridos a lo largo del grado para obtener como resultado un prototipo funcional, el cual se mostrará en forma de aplicación móvil o tablet (sólo para aquellos dispositivos que posean un sistema operativo Android) cuyo requerimiento indispensable es que se traten de dispositivos que permitan el reconocimiento dactilar, ya que, el fichaje se realizará mediante este método con la intención de asemejar este sistema de fichaje al sistema común de fichajes que poseen la mayoría de las empresas (sistemas biométricos)

1.4. Estructura de la memoria

Este documento se ha estructurado de la siguiente manera:

2. Aspectos preliminares

En este capítulo se detallan la justificación de competencias y las aportaciones del proyecto.

3. Descripción del proyecto

Este capítulo consta de varios apartados: Normativa y legislación, descripción del TFT, herramientas, servicios y tecnologías utilizadas y metodología aplicada para su realización.

4. Análisis

Este capítulo está destinado a la parte de análisis, tanto análisis de mercado como análisis de requisitos.

5. Diseño

Este capítulo abarca toda la parte del diseño del proyecto, incluyendo el diseño de la interfaz de usuario, diseño del software y diseño de la base de datos.

6. Estructura y elementos del proyecto

En este capítulo se expone la estructura interna del proyecto, las dependencias inyectadas para su implementación y los elementos Android más importantes.

7. Desarrollo del proyecto

En este capítulo se describen todos los detalles del desarrollo del proyecto: Historias de usuario implementadas, sprints realizados y validación.

8. Resultados, conclusiones y trabajo futuro

En este capítulo, se detallan los resultados del proyecto, conclusiones y posibles extensiones que podría tener.

2. Aspectos preliminares

2.1. Justificación de competencias

2.1.1. Competencia IS02

Descripción: Capacidad para valorar las necesidades del cliente y especificar los requisitos software para satisfacer estas necesidades, reconciliando objetivos en conflicto mediante la búsqueda de compromisos aceptables dentro de las limitaciones derivadas del coste, del tiempo, de la existencia de sistemas ya desarrollados y de las propias organizaciones.

Justificación: Para una correcta elaboración del trabajo, se han realizado distintas entrevistas a empresarios cuyo perfil encaje en el ámbito del TFT (que posean empleados que desarrollen sus tareas de manera telemática o presten servicios fuera del lugar de trabajo), con el fin de captar necesidades y acercar el trabajo, lo más posible a la realidad.

2.1.2. Competencia IS04

Descripción: Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales.

Justificación: El primer paso que se dio para realizar el TFT, fue identificar el problema existente (Necesidad de controlar el horario de los empleados para cumplir con la normativa), y una vez descrito el problema, se procedió a diseñar y desarrollar una solución al mismo.

2.1.3. Competencia IS06

Descripción: Capacidad para diseñar soluciones apropiadas en uno o más dominios de aplicación utilizando métodos de la ingeniería del software que integren aspectos éticos, sociales, legales y económicos.

Justificación: Debido a la situación que estamos viviendo con la pandemia de COVID-19, muchas personas se han visto en la obligación de realizar sus tareas laborales en la modalidad de teletrabajo lo cual dificulta el control de horario de manera presencial y es por ello que surgen alternativas para ello, como la que se propone como TFT. Además, en cuanto a aspectos legales, se debe cumplir con la normativa implantada por la ley Real Decreto-Ley 8/2019.

2.2. Aportaciones

Con la realización de este proyecto, se pretende aportar un prototipo funcional cuyo objetivo consiste en ayudar a realizar de manera automática, fácil y rápida el registro de control de la jornada laboral de los empleados.

Clock-in On Time es un prototipo funcional que pretende satisfacer las necesidades del empresario de realizar el registro de la jornada laboral de sus empleados para cumplir con la nueva normativa y, además, los empleados podrán realizar el fichaje de su jornada laboral fácil y cómodamente.

3. Descripción del proyecto

3.1. Normativa y legislación

3.1.1. Reglamento general de protección de datos (RGPD)

El Reglamento General de Protección de Datos (RGPD) es el reglamento europeo relativo a la protección de las personas físicas en lo que respecta al tratamiento de sus datos personales y a la libre circulación de estos datos. Entró en vigor el 25 de mayo de 2016 y fue de aplicación el 25 de mayo de 2018, dos años durante los cuales, las empresas, las organizaciones, los organismos y las instituciones se fueron adaptando para su cumplimiento. Es una normativa a nivel de la Unión Europea, por lo que cualquier empresa perteneciente a la Unión, o aquellas empresas que tengan negocios en la Unión Europa, que manejen información personal de cualquier tipo, deberán acogerse a ella. Las multas por su no cumplimiento pueden llegar a los 20 millones de euros.

En España, la RGPD dejó obsoleta la Ley Orgánica de Protección de Datos de Carácter Personal (LOPD) de 1999, siendo sustituida el 6 de diciembre de 2018 por la Ley Orgánica de Protección de Datos Personales y garantía de derechos digitales acorde con la RGPD. [2]

3.1.2. Ley orgánica de protección de datos y garantía de derechos digitales (LOPDGDD)

La Ley Orgánica 3/2018, de 6 de diciembre, de Protección de Datos personales y garantía de los derechos digitales (LOPD-GDD) es una ley orgánica aprobada por las Cortes Generales de España que tiene por objeto adaptar el Derecho interno español al Reglamento General de Protección de Datos. Esta ley Orgánica deroga a la anterior Ley Orgánica 15/1999 de Protección de Datos de Carácter Personal (aunque se mantiene vigente para la regulación de ciertas actividades). [3]

Esta Ley entró en vigor el 7 de diciembre de 2018.

3.1.3. Real Decreto-Ley 8/2019: Medidas urgentes de protección social y de lucha contra la precariedad laboral en la jornada de trabajo

Es una ley, que se dispuso el 8 de marzo de 2019 y posteriormente, el 13 de marzo de 2019 entró en vigor, la cual establece la obligatoriedad de establecer un sistema de control horario en todas las empresas.

El motivo de la llegada de esta nueva normativa es tener un control y un registro de las horas trabajadas, en beneficio tanto de las empresas, como de los trabajadores. Para las empresas, será una manera de controlar los tiempos de trabajo y el absentismo laboral y para los trabajadores, será una medida de "defensa" frente a las horas extras, especialmente aquellas por las que no se reciben retribución. [4]

3.2. Descripción del TFT

3.2.1. Descripción y características

En este TFT se ha desarrollado un prototipo funcional en forma de aplicación móvil, cuyo objetivo es la resolución del problema inicial propuesto.

CARACTERÍSTICAS DESTACABLES DE UN PROTOTIPO:

- Permiten descubrir con rapidez si el usuario se encuentra satisfecho o no con los requisitos definidos.
- NO es el sistema final, ya que no tiene que abarcar la totalidad de la funcionalidad del sistema.
- Define sus requisitos indirectamente.
- El usuario se hace una idea de la estructura de la interfaz del sistema.

El nombre de este prototipo es ***Clock-in On Time*****.** Se trata de una aplicación móvil, destinada a empresas y empleados. Concretamente es una aplicación móvil, para dispositivos Android a través de la cual, los empleados pueden autenticarse con las credenciales que previamente su compañía le ha facilitado. Su funcionalidad principal consiste en permitir a los empleados de las empresas realizar el fichaje de entrada y salida de su jornada laboral diaria, de manera que, por ejemplo, empleados que presten servicios fuera de la empresa, puedan realizar dicho registro de horario sin necesidad de pasar por el lugar de trabajo.

Ilustración 1. Logo del TFT

Esta aplicación tiene la peculiaridad de que, al realizar el fichaje, el empleado debe introducir su huella dactilar, con el fin de que se parezca lo más posible a los sistemas más comunes para realizar fichajes que suelen utilizar las empresas hoy en día.

En pocas palabras, este proyecto representa a una aplicación destinada a dispositivos Android, que se comunica con el servidor Firebase y proporciona a los usuarios las siguientes funcionalidades:

- **Iniciar sesión**: Permite que los empleados de una empresa puedan iniciar sesión en la plataforma. Para ello el administrador (o encargado de esta tarea en la empresa) debe crear al empleado en su versión de escritorio y posteriormente facilitar estas credenciales al mismo para poder entrar. Si la empresa no da de alta al empleado en esta plataforma, este no puede hacer uso de la misma.
- **Realizar justificación**: Esta funcionalidad permite a los empleados realizar justificaciones de faltas al día de trabajo. Por ejemplo, un empleado que debe asistir a una cita médica puede realizar la justificación de falta al trabajo en ese día. (Para justificar la falta es necesario adjuntar el documento que acredite el motivo justificado).

- **Ver justificantes**: Con esta funcionalidad, el empleado podrá acceder al listado de justificantes adjuntados.
- **Ver justificante**: A través del listado de justificantes, el usuario podrá pinchar un justificante en concreto y visualizar su contenido.
- **Ver fichajes**: A través de esta funcionalidad, el empleado podrá acceder a una su lista de fichajes realizados.
- **Fichar entrada**: Aquellos usuarios que han sido dados de alta con éxito y pueden entrar a la aplicación, podrán realizar el fichaje de entrada. A través de este fichaje, se guarda en la base de datos correspondiente la hora en la que se realizó, el usuario que la realizó y la ubicación en la que se realizó.
- **Fichar salida:** Al igual que la funcionalidad de "Fichar salida" pero guardando los cambios correspondientes al horario de salida.
- **Ver fichajes**: Permite al empleado visualizar el lisado de los fichajes realizados.
- **Ver detalles del fichaje**: A partir del listado anterior, el empleado podrá seleccionar (haciendo *click*) en un fichaje en concreto y a continuación, le aparecerán los detalles del mismo.
- **Buscar fichaje por fecha**: A través de esta funcionalidad, el usuario podrá abrir el calendario y seleccionar una fecha en concreto para visualizar el fichaje correspondiente a la fecha.
- **Ver ayuda**: Funcionalidad que permite al usuario visualizar el panel de preguntas frecuentes.
- **Cerrar sesión**: Funcionalidad que permite al usuario dejar de estar identificado en la plataforma.

3.3. Herramientas, servicios y tecnologías usadas

Todos los logotipos y marcas gráficas de productos han sido obtenidos de sus correspondientes sitios webs.

3.3.1. Herramientas

Android studio

Android Studio es el entorno de desarrollo utilizado para la implementación del proyecto. Este entorno de desarrollo es el oficial para la plataforma Android. Fue anunciado el 16 de mayo de 2013 en la conferencia Google I/O, y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android. La primera versión estable fue publicada en diciembre de 2014.

Está basado en el software *IntelliJ IDEA* de **JetBrains** y ha sido publicado de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas Microsoft Windows, macOS y GNU/Linux. Ha sido diseñado específicamente para el desarrollo de Android. [5]

Esta herramienta permite desarrollar proyectos para Android utilizando el lenguaje **Kotlin** o **Java** para **Android**. Además, permite escoger la versión de Android en la que desarrollar los proyectos.

Asimismo, para la elaboración del proyecto he decidido escoger el Android SDK 23 (*Android 6.0 Marshmallow, API 23*), ya que permite la autenticación por huellas dactilares. La aplicación será compatible solo para dispositivos Android que posean esta versión o superior.

Este ha sido el entorno de desarrollo utilizado para implementar el proyecto, ya que es el entorno de desarrollo oficial para aplicaciones Android.

Star Uml

StarUML es una herramienta para el modelamiento de software basado en los estándares **UML** (*Unified Modeling Language*) y **MDA** (*Model Driven Arquitecture*), que en un principio era un producto comercial y que hace cerca de un año paso de ser un proyecto comercial (anteriormente llamado plastic) a uno de licencia abierta GNU/GPL. [6]

Google Chrome

Google Chrome es un navegador web de código cerrado[67] desarrollado por Google, aunque derivado de proyectos de código abierto (como el motor de renderizado Blink). Está disponible gratuitamente. El nombre del navegador deriva del término en inglés usado para el marco de la interfaz gráfica de usuario («chrome»). [7]

Esta herramienta ha sido utilizada para buscar información y tutoriales en la web.

Microsoft Word

Word un programa informático orientado al procesamiento de textos. Fue creado por la empresa Microsoft, y viene integrado de manera predeterminada en el paquete ofimático denominado Microsoft Office.

Con el uso de esta herramienta se ha elaborado la documentación de la memoria del TFT. [8]

Firebase

Firebase se trata de una plataforma creada por Google, cuya principal función es desarrollar y facilitar la creación de apps de elevada calidad de una forma rápida, con el fin de que se pueda aumentar la base de usuarios y ganar más dinero. La plataforma está subida en la nube y está disponible para diferentes plataformas como iOS, Android y web. Contiene diversas funciones para que cualquier desarrollador pueda combinar y adaptar la plataforma a medida de sus necesidades. [9]

Mockplus

Mockplus es una de las numerosas alternativas existentes en el mercado para diseñar y prototipar interfaces de usuario para diferentes plataformas, como por ejemplo Windows, Mac, aplicaciones móviles (iOS y Android) y Web. [10]

Trello

Trello es una herramienta, totalmente gratuita, de gestión de proyectos que permite organizar las tareas a realizar, que permite incluso, utilizarla en equipo. [11]

Esta herramienta se ha utilizado para la organización de las diferentes tareas a realizar en el proyecto.

GitHub

GitHub es un sistema de gestión de proyectos y control de versiones de código, así como una plataforma de red social diseñada para desarrolladores. Esta herramienta permite trabajar en colaboración con otras personas de todo el mundo, planificar proyectos y realizar un seguimiento de trabajo. [12]

GitHub es también, uno de los repositorios online más grandes de trabajo colaborativo en todo el mundo.

Esta herramienta se ha utilizado para la creación de un repositorio con el fin de almacenar los archivos del proyecto.

Además, al principio del proyecto se sincronizó el Android Studio con el repositorio GitHub del proyecto, lo cual, a través de este entorno de desarrollo, se fueron añadiendo los archivos directamente al repositorio GitHub donde se encuentra el proyecto.

3.3.2. Servicios

Logomaster.ai

Es una web que, a través con la ayuda de inteligencia artificial, permite la creación y diseño de logotipos para starstups, profesionales y pequeñas empresas. [13]

A través de esta herramienta se ha creado el logo para el proyecto del TFT.

Firebase Realtime Database

Firebase Realtime Database es una base de datos alojada en la nube. Los datos se almacenan en formato **JSON** y se sincronizan en tiempo real con cada cliente conectado. Cuando compilas apps multiplataforma con nuestros SDK de **iOS**, **Android** y **JavaScript**, todos tus clientes comparten una instancia de Realtime Database y reciben actualizaciones automáticamente con los datos más recientes. [14]

Firebase Authentication

Es una utilidad ofrecida por **Firebase**, la cual proporciona servicios de *backend*, SDK fáciles de usar y bibliotecas de IU ya elaboradas para autenticar a los usuarios en tu app. Admite la autenticación mediante contraseñas, números de teléfono, proveedores de identidad federada populares como Google, Facebook, Twitter y mucho más. [15]

Firebase Storage (Firebase Cloud Storage)

Es un servicio ofrecido por **Firebase** que permite cargar y descargar, de forma segura, archivos para sus aplicaciones independientemente de la calidad de la red. Se puede utilizar para almacenar imágenes, videos, audios, etc. Este servicio está respaldado por **Google Cloud Storage**, el cual es un servicio de almacenamiento de objetos potente, simple y rentable. **Firebase** Storage almacena sus archivos en un depósito de Google Cloud Storage compartido con la aplicación Google App Engine predeterminada, haciéndolos accesibles a través de las API de **Firebase** y **Google Cloud**, lo cual permite flexibilidad para cargar y descargar archivos de clientes móviles. [16]

3.3.3. Tecnologías

Java

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarán a menos que tenga Java instalado y cada día se crean más. Java es rápido, seguro y fiable. Desde portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet, Java está en todas partes. [17]

Lenguaje utilizado para la realización de la parte lógica del proyecto.

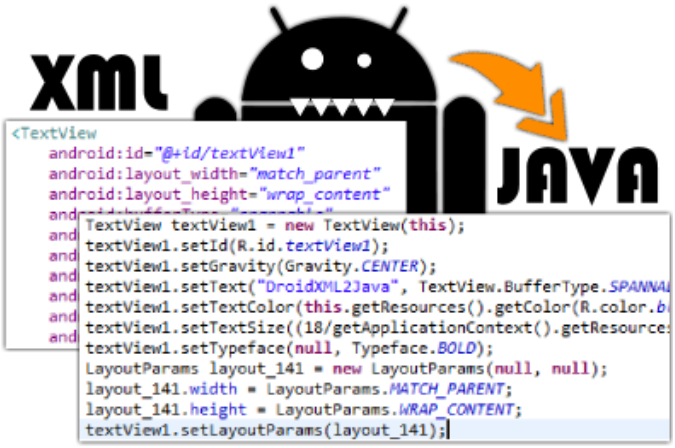

Ilustración 2. XML y Java para Android

XML (Extensible Markup Language)

El lenguaje de marcado es un conjunto de códigos que se pueden aplicar en el análisis de datos o la lectura de textos creados por computadoras o personas.

XML es un lenguaje de marcado que define un conjunto de reglas para la codificación de documentos. Existe vocabulario XML para Android, el cual, permite crear rápidamente diseños de IU y de los elementos de pantalla que contienen, de la misma manera que se hace cuando se crean páginas web en HTML**.** [18]

Este lenguaje de marcado fue utilizado para realizar la parte visual del proyecto (Vistas).

3.4. Metodología y planificación del proyecto

El proyecto se ha llevado a cabo mediante una metodología ágil (una metodología ágil es una metodología de gestión de proyectos que utiliza ciclos de desarrollo cortos llamados ***sprints*** para centrarse en la mejora continua del desarrollo de un producto o servicio, más que centrarse en la gestión de un propio proyecto). Existen algunas metodologías ágiles como por ejemplo Scrum, XP (***Extreme Programming***), Lean o Kanban.

La metodología que se ha seguido para la realización de este proyecto es **Scrum**.

3.4.1. Scrum

¿Qué es Scrum?

"Scrum es un marco de trabajo que permite el trabajo colaborativo entre equipos. Al igual que un equipo de rugby (de donde proviene su nombre) cuando entrena para el gran partido, scrum anima a los equipos a aprender mediante las experiencias, a organizarse de forma autónoma mientras se trabaja en un problema y a reflexionar sobre sus victorias y derrotas para mejorar continuamente.

Aunque son los equipos de desarrollo de software los que utilizan con mayor frecuencia el scrum del que estoy hablando, sus principios y lecciones se pueden aplicar a todo tipo de trabajo en equipo. Esta es una de las razones por las que es tan popular. Aunque se considera a menudo un marco de trabajo de gestión de proyectos ágiles, scrum hace referencia a un conjunto de reuniones, herramientas y funciones que trabajan de forma coordinada para ayudar a los equipos a estructurar y gestionar su trabajo." [19]

Roles en Scrum

En Scrum existen distintos roles para desempeñar las diferentes funciones existentes en esta metodología.

- **Product Owner (Propietario del producto)**
  Es el responsable de maximizar el valor del trabajo del equipo de desarrollo. La maximización del valor del trabajo viene de la mano de una buena gestión del **Product Backlog**, el cual explicaremos más adelante.

  Para finalizar, un **equipo Scrum** debe tener un solo **Product Owner** y este puede ser parte del equipo de desarrollo.

- **Scrum Master**
  Es el responsable de que las técnicas Scrum sean comprendidas y aplicadas en la organización. Es el manager de Scrum. Un líder que se encarga de eliminar impedimentos o inconvenientes que tenga el equipo dentro de un sprint, aplicando las mejores técnicas para fortalecer el equipo de marketing digital.

  Dentro de la organización, el Scrum Master tiene la labor de ayudar en la adopción de esta metodología en todos los equipos.

- **Scrum Team (Equipo de desarrollo)**
  Son los encargados de realizar las tareas priorizadas por el P**roduct Owner**. Es un equipo multifuncional y auto-organizado. Son los únicos que estiman las tareas del **Product Backlog** sin dejarse influenciar por nadie.

Artefactos Scrum

- La **pila de producto** o **Product Backlog** es el listado de funcionalidades (total del proyecto) en Scrum.
- La **pila de sprint** o **Sprint Backlog** es un listado que se corresponde con las funcionalidades de la pila de producto que se proponen para completar durante

un sprint, con el fin de obtener un incremento. Se descomponen las historias en tareas a resolver durante el sprint.

- El **incremento** se corresponde con el producto desarrollado al terminar un sprint

¿Cómo funciona Scrum?

Scrum es una metodología de desarrollo que desarrolla un marco de trabajo simple de forma iterativa e incremental.

En primer lugar, se desarrolla una ***pila de producto*** o ***Product Backlog***, que es un listado de las distintas funcionalidades que va a abarcar el proyecto entero. En esta pila de producto se escribirán las distintas historias de usuario con su prioridad. Una vez realizada la pila de producto, se planifican los **sprints**.

Posteriormente, se escogerán las diferentes historias de usuario a desarrollar en el sprint y se escribirán en la ***pila de sprint*** o ***Sprint Backlog***.

La duración de cada ***sprint*** (o *iteración*) puede ser de una a cuatro semanas y en él, se desarrollarán las distintas funcionalidades que se encuentren en la pila de sprint. Cada sprint o iteración debe suponer un incremento, el cual debe aportar valor al cliente.

Una vez finalizado el sprint, se realiza la revisión del sprint, en el cual se presenta dicho incremento y se evalúa el sprint (que ha ido bien, que ha ido mal y su motivo).

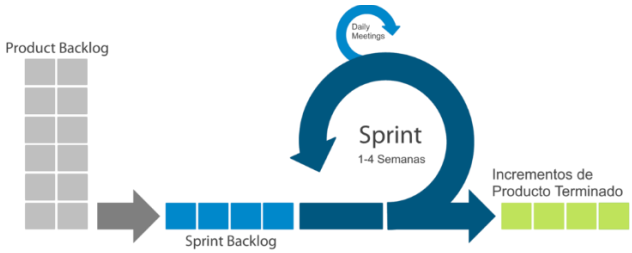

Ilustración 3. Proceso de Scrum [20]

3.4.2. Planificación del proyecto

FASES	DURACIÓN ESTIMADA	DURACIÓN REAL	TAREAS
Aprendizaje de herramientas de desarrollo	50 horas	85 horas	Tarea 1: Aprendizaje sobre las herramientas de desarrollo Android.
Captación de necesidades	10 horas	4 horas	Tarea 2: Consultar las necesidades de los usuarios de la aplicación.
Análisis	65 horas	40 horas	Tarea 2.1: Realización del análisis del mercado.
			Tarea 2.2: Realización del análisis de requisitos del Software.
Diseño y Desarrollo	125 horas	211 horas	Tarea 3.1: Realización de boceto del producto.

			Tarea 3.2: Implementación.
Validación	15 horas	8 horas	Tarea 4: Proporcionar la aplicación a clientes para que prueben su funcionamiento.
Documentación / Presentación	35 horas	50 horas	Tarea 5: Realización de la documentación sobre el funcionamiento y detalles del producto

Tabla 1. Planificación del proyecto

4. Análisis

4.1. Análisis de mercado

Todos los logotipos y marcas gráficas son recursos que han sido obtenidos de sus correspondientes sitios oficiales.

4.1.1. Software para control horario

Tradicionalmente, el control de horario se realizaba mediante plantillas Excel, pero, a día de hoy, se ha digitalizado el proceso con implementaciones de software que realizan dicha función.

El software para el control de horario de los trabajadores se ha convertido en una herramienta que sustituyen a las antiguas tarjetas de fichaje y documentos Excel.

Estas son algunas de las ventajas que posee la utilización de software específico para regular la normativa:

- Registro de la totalidad de la jornada de los trabajadores
- Simplificación de las tareas de Recursos Humanos
- Flexibilidad horaria.
- Gestión rápida de incidentes
- Automatización.

Por otro lado, es interesante saber cuáles son las características deseables que deben poseer este tipo de software:

- Fácil utilización y configuración intuitiva (Está pensado para empresarios de todo tipo y no sólo para aquellos que tengan conocimientos específicos).
- Posibilidad de visualizar el listado de fichajes (Permite visualizar un listado con todos los fichajes realizados).
- Fichajes geolocalizados (Pensado para empleados que desarrollan sus tareas en remoto).
- Posibilidad de programar recordatorios.

4.1.2. Estudio de mercado

Dentro del mercado ya existen aplicaciones que permiten el control horario de los empleados, pudiendo fichar la entrada y salida y hacer pausas. El acceso se realiza mediante usuario y contraseña y los fichajes se realizan haciendo click en el botón correspondiente. La mayoría de las aplicaciones ya existentes cuentan con su aplicación web y ofrecen servicios de geolocalización y algunas incluso, realizan una fotografía del empleado en cuestión, a través de la cámara del dispositivo móvil.

Entre todas las aplicaciones existentes que ofrecen los servicios y poseen las características mencionados anteriormente y, se encuentran las siguientes:

CONTROL LABORAL cuyo precio se encuentra desde 19€/mes dependiendo del número de empleados. Esta plataforma es utilizada por algunas empresas como por ejemplo, Iberdrola, Candelas o Inteserve. [21]

- **Valoración media de usuarios: 3,2 estrellas**
- Enlace al video de presentación Control laboral

Ilustración 4. Aplicación "Control Laboral".

CONTROL HORARIO (BIXPE) ofrece un servicio gratuito pero limitado en cuanto a funcionalidades y otro a partir de 2 euros por usuario al mes (que permite realizar fotografías de la persona que ficha y además ofrece historial de fichajes durante 4 años). [22]

- **Valoración media de usuarios: 4,4 estrellas**
- Enlace al vídeo de presentación Bixpe

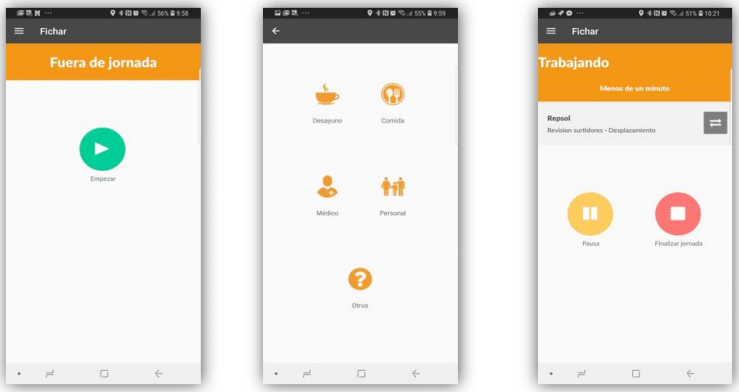

Ilustración 5. Aplicación "Control horario" (BIXPE)

INTRATIME es una aplicación que ofrece servicios a partir de 0,99 euros por usuario al mes. También permite realizar fichajes geolocalizados. [23]

- **Valoración media de usuarios: 3,5 estrellas**
- Enlace al vídeo de presentación de Intratime

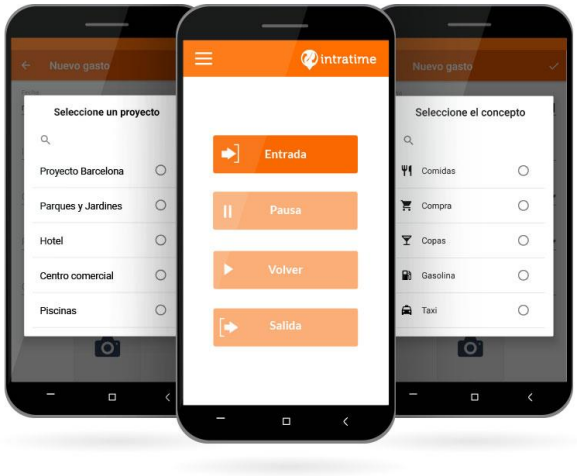

Ilustración 6. Aplicación "Intratime"

FICHA.WORK es una aplicación disponible para IOS, Android y web desde 1 euro al mes (IVA a parte) por trabajador. (ofrece el historial de fichajes durante 4 años). [24]

- **Valoración media de usuarios: 4,1 estrellas**

Ilustración 7. Aplicación "Ficha.work"

SESAME TIME cuyo servicio se presta a partir de los 29 euros por usuario al mes (Además permite programar vacaciones y ausencias). [25]

- **Valoración media de usuarios: 4,7 estrellas**
- Enlace al vídeo de presentación de Sesame time

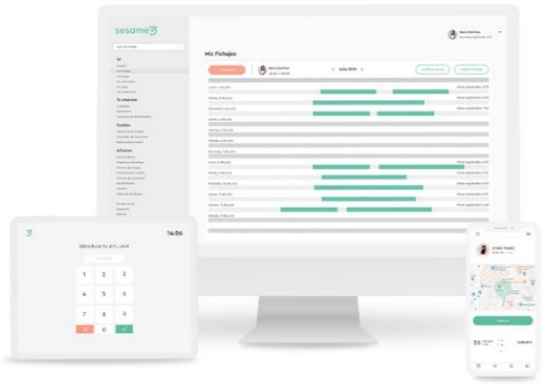

Ilustración 8. Aplicación "Sesame Time"

4.1.3. Cuadro comparativo

A continuación, se muestra un cuadro comparativo en el que se pueden observar las funcionalidades principales ofrecidas por cada una de las aplicaciones del listado anterior:

Características/ Aplicaciones	Control Laboral	Control Horario (Bixpe)	Ficha work	Intratime	Sesame Time
Dispone de aplicación móvil	Sí	Sí	Sí	Sí	Sí
Precio	19€/mes	2€/usuario al mes	1€/mes por trabajador	0,99€/mes por trabajador	29€/usuario al mes
Geolocalización /Ubicación	No	Sí	Sí	Sí	Sí
Historial de fichajes	Sí	Sí	-	-	-
Reconocimiento dactilar (desde la app móvil)	No	No	No	No	No
Reconocimiento dactilar (en la empresa)	-	-	-	-	Sí

Fotografía de la persona que ficha	No	Sí	-	-	-

Tabla 2. Tabla comparativa de aplicaciones

4.1.4. Entrevistas a empresarios

Con el fin de seguir obteniendo y captando información sobre el sector se han realizado entrevistas a empresarios que tienen trabajadores cuyos servicios se prestan fuera del lugar de trabajo.

- **PASO 1:** Identificar a los entrevistados: Como primer paso, procedí a observar a posibles empresas que, como requisito fundamental e indispensable, tuvieran empleados que prestaran servicios fuera del lugar de trabajo. Entre las empresas entrevistadas se encuentran

- **PASO 2:** Preparar la entrevista: Como segundo paso, realicé un listado de preguntas para la entrevista, las cuales se muestran a continuación:

> **1**. ¿Qué puestos de trabajos existen en tu empresa que presten servicios fuera de la empresa?
>
> **2.** ¿Ha habido alguna incidencia a la hora de controlar el horario de algún empleado?
>
> **3.** ¿Cómo controlan las horas que trabajan esos empleados? Si lo hace a través de una app, ¿Podría decirme el precio?
>
> **4.** ¿Cómo le gustaría controlar las horas trabajadas de sus empleados?

- **PASO 3:** Realizar la entrevista: En este caso, hice entrevistas tanto telemáticas como telefónicas. No pudieron ser de manera presencial por las consecuencias del virus COVID-19, y por este motivo no se pudieron entrevistar a tantas empresas como se tenía previsto. Entre las empresas entrevistadas se encuentran **Vitalife Canarias S.L**, **Alianza Alemán Blaker S.L** y **Edataconsulting**.

- **PASO 4:** Documentar los resultados: A continuación, se documentan los resultados obtenidos en el mismo orden del listado de las preguntas:
    1. Los empleados que poseen las empresas entrevistadas, en su gran mayoría son repartidores y comerciales. También existen desarrolladores software en remoto.
    2. En algunas empresas, han existido algunas incidencias con sus empleados, debido a que, o bien el empleado no se presenta en la cita con el cliente, o bien el empleado es impuntual en su jornada laboral, lo cual dificulta el registro de control horario. Al contrario, en otra de las empresas no se han registrado incidencias de este tipo.

3. En este punto, se registran dos tipos de controles. Por un lado, una de las empresas posee un sistema interno a través del cual se puede realizar el control horario y, por lo tanto, gratuito. Otra de las empresas solo posee el fichaje dentro del lugar de trabajo, por lo que, los repartidores y comerciales antes de acudir a la cita con el cliente, deben ir a la empresa y fichar, lo cual, es una pérdida de dinero en transporte, reflejaba una de las empresas. Finalmente, otra de las empresas, no se posee ningún tipo de sistema de control horario, haciéndose el control a través de firmas de documentos.
4. En este punto, las empresas han expresado la necesidad de poseer una aplicación software que les ayude a controlar a sus trabajadores de forma automática y de manera que, estos datos se guarden con el fin de no ser perdidos y poder, de esta manera, cumplir con la nueva ley impuesta.
5. Finalmente, este punto está totalmente claro. Las empresas expresan la necesidad de realizar un control más exhaustivo a sus empleados, de manera que, al realizar el fichaje, también se pueda registrar la ubicación en la que lo han realizado.

4.1.5. Modelo de negocio

El sistema completo debe constar de dos partes. Por un lado, una aplicación móvil, a través de la cual el empleado pueda realizar los fichajes de entrada y salida o justificar una falta (principalmente) y, por otro lado, una aplicación (móvil, web o de escritorio) a través de la cual, los administradores de las empresas (encargados de recursos humanos de las empresas) puedan gestionar a sus empleados y definir reglas de fichaje. Para este TFT, se ha realizado un prototipo funcional sobre la aplicación móvil para los empleados (faltando la aplicación para los administradores para conseguir el sistema completo).

- **Reglas de fichaje**: Las reglas de fichaje hacen referencia a aquellas instrucciones que deben ser cumplidas por el empleado.
    - Existen dos reglas de fichaje importantes, aunque pueden surgir más a lo largo del tiempo: Definir la jornada laboral de un empleado y establecer un límite entre dos horarios para su fichaje.
    - **EJEMPLO 1** (regla de fichaje de jornada laboral): Definir la jornada laboral de un empleado en 8 horas, sería una regla de fichaje definida, de manera que el empleado debe cumplir con ella diariamente y los periodos de actividad deben sumar la cifra establecida en la regla.
    - **EJEMPLO 2** (regla de fichaje de límite horario para fichar): Definir una regla de fichaje, que permita al usuario realizar el fichaje con un tiempo límite, es decir, permitirle fichar en un periodo de tiempo establecido, como por ejemplo, permitir al empleado que realice el fichaje entre las 8:00 y las 9:00.

A continuación, se muestra el *Lean Canvas* (para dar un enfoque más general) y *Bussiness Model Canvas* (para dar un enfoque más preciso del negocio) sobre el sistema completo (aplicación móvil + aplicación para administradores).

Bussiness Model Canvas

Bussiness Model Canvas es una plantilla de gestión estratégica para el desarrollo de nuevos modelos de negocio o documentar los ya existentes. Es un gráfico visual con elementos que describen propuestas de producto o de valor de la empresa, la infraestructura, los clientes y las finanzas. También ayuda a las empresas a alinear sus actividades mediante la ilustración de posibles compensaciones. [26]

<table>
<tr>
<td rowspan="2">Aliados Clave
- Proveedores de internet
- Tienda de aplicaciones</td>
<td>Actividades Clave
- Programación
- Publicidad
- Distribución</td>
<td rowspan="2">Propuesta de Valor
- Registro de la jornada laboral por parte de los empleados.
- Control y gestión de los empleados para las empresas.</td>
<td>Relación con el Cliente
- Redes sociales
- Web
- E-mail
- Teléfono.</td>
<td rowspan="2">Segmentos de Clientes
Empresas con empleados que presten servicios fuera y dentro del lugar de trabajo</td>
</tr>
<tr>
<td>Recursos Clave
- Aplicación Android
- Aplicación Web
- Internet
- Licencias</td>
<td>Canales
Comunicación: Web propia, redes sociales y anuncios

Entrega: Google Play y web</td>
</tr>
<tr>
<td colspan="3">Estructura de Costes
- Pago del personal.
- Pago de la publicidad y promoción.
- Pago de internet.</td>
<td colspan="2">Estructura de Ingresos
- Suscripción low: Suscripción para empresas de hasta 25 empleados.
- Suscripción medium: Suscripción para empresas de entre 26 y 60 empleados.
- Suscripción premium: Suscripción para empresas de más de 60 empleados.</td>
</tr>
</table>

Ilustración 9. Bussiness Model Canvas del proyecto

Lean Canvas

Lean Canvas es una herramienta estratégica empresarial, que permite analizar de manera visual nuestro modelo de negocio para aumentar sus probabilidades de éxito. [27]

<table>
<tr>
<td rowspan="2">Problema
- Difícil controlar y geolocalizar a un empleado que preste servicios fuera de la empresa.

- Difícil llevar un control sobre el horario de todos los empleados de una empresa.

- Las empresas deben cumplir con la norma de control de horario impuesta por el Real Decreto-Ley 8/2019.</td>
<td>Solución
- Aplicación móvil con la que los empleados pueden fichar y ser geolocalizados.

- Aplicación web o de escritorio a través de la cual se pueden gestionar a los empleados.</td>
<td rowspan="2">Propuesta de Valor única
- Registro de la jornada laboral por parte de los empleados.
- Control y gestión de los empleados para las empresas.</td>
<td>Ventaja especial
- Precios únicos.

- Distintos tipos de suscripciones (dependiendo del número de empleados).

- Autenticación por huella dactilar.</td>
<td rowspan="2">Segmentos de Clientes
Empresas con empleados que presten servicios fuera y dentro del lugar de trabajo</td>
</tr>
<tr>
<td>Métricas Clave
- Descargas app/mes.

- Suscripciones low/medium/premium.</td>
<td>Canales
Comunicación: Web propia, redes sociales y anuncios

Entrega: Google Play y web</td>
</tr>
<tr>
<td colspan="3">Estructura de Costes
- Pago del personal.
- Pago de la publicidad y promoción.
- Pago de internet.</td>
<td colspan="2">Estructura de Ingresos
- Suscripción low: Empresas de hasta 25 empleados.
- Suscripción medium: Empresas de entre 26 y 60 empleados.
- Suscripción premium: Empresas de más de 60 empleados.</td>
</tr>
</table>

Ilustración 10. Lean Canvas del proyecto

4.2. Análisis de requisitos

4.2.1. Stakeholders

Los ***Stakeholders*** son aquellas personas o partes interesadas en el sistema que se está desarrollando. Hace referencia a todos aquellos agentes, internos o externos que de alguna u otra forma están involucrados en la actividad de una empresa y que por ende resultan

afectados con el desempeño de ésta. Existen dos grupos de stakeholders, los **primarios** (que tienen influencia directa, como, por ejemplo, empleados, gerentes, propietarios, etc) y los **instrumentales** (que tienen influencia directa y están en el ámbito de la empresa. Pueden influenciar a los primarios, como por ejemplo pueden ser, proveedores, sociedad, gobierno, etc).

En el grupo de stackeholders podemos incluir a la ***empresa***, que será el principal afectado en el producto (y por lo tanto, muy interesados en el mismo), ***usuarios*** que consumirán el servicio (como empleados y administradores, también muy interesados), ***desarrolladores*** (grupo de personas que realizaran el proyecto, muy interesados), la ***competencia*** (que hace referencia a los propietarios de los productos que competirán con esta nueva aplicación) y finalmente la ***autoridad*** (ministerio de trabajo y seguridad social y administraciones públicas, muy interesados).

4.2.2. Requisitos

¿Qué son los requisitos funcionales?

Los requerimientos funcionales son declaraciones de los servicios que proveerá el sistema, de la manera en que éste reaccionará a entradas particulares. En algunos casos, los requerimientos funcionales de los sistemas también declaran explícitamente lo que el sistema no debe hacer. [28]

¿Qué son los requisitos no funcionales?

Son aquellos requerimientos que no se refieren directamente a las funciones específicas que entrega el sistema, sino a las propiedades emergentes de éste como la fiabilidad, la respuesta en el tiempo y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y la representación de datos que se utiliza en la interfaz del sistema. [28]

Requisitos de la aplicación:

Una vez definidos los stakeholders, se procedió a identificar los requisitos, tanto funcionales como los no funcionales. Los requisitos establecidos para el proyecto se detallan a continuación:

REQUISITOS FUNCIONALES PARA LA APLICACIÓN MÓVIL:

- El sistema debe permitir solicitar el cambio de contraseña.
- El sistema debe enviar un correo al usuario tras solicitar el cambio de contraseña para realizar dicho cambio.
- El sistema debe alertar al usuario de que los fichajes deben hacerse parado y no en movimiento.
- El sistema debe permitir realizar el fichaje de entrada.
- El sistema debe permitir realizar el fichaje de salida.
- El sistema debe mostrar al empleado el tiempo trabajado en la jornada laboral.
- El sistema debe avisar al usuario si intenta registrar el fichaje de salida a falta del fichaje de entrada.
- El sistema debe permitir visualizar el perfil
- El sistema debe permitir modificar email, número de teléfono y contraseña del empleado.

- El sistema debe permitir visualizar la contraseña al editar perfil.
- El sistema debe mostrar un apartado de ayuda al empleado.
- El sistema debe permitir justificar una falta en el día que se produce.
- El sistema debe permitir cancelar la justificación de una falta.
- El sistema debe avisar al usuario de que se ha realizado la justificación con éxito.
- El sistema debe inhabilitar las opciones de fichaje (de entrada y salida) si el empleado justifica la falta.
- El sistema debe inhabilitar la opción de fichaje de entrada si se ha realizado previamente.
- El sistema debe inhabilitar la opción de fichaje de salida si se ha realizado previamente.
- El sistema debe alertar al usuario de que debe activar el GPS del dispositivo.
- El sistema debe permitir visualizar el listado de fichajes realizados.
- El sistema debe permitir filtrar el listado de fichajes realizados, permitiendo realizar una búsqueda de un día concreto a través del calendario.
- El sistema debe permitir ver los detalles de un fichaje.
- El sistema debe permitir visualizar el listado de justificantes adjuntados.
- El sistema debe permitir visualizar el contenido de un justificante.
- El sistema debe permitir visualizar la ubicación de un fichaje junto con sus detalles.

REQUISITOS FUNCIONALES PARA LA APLICACIÓN DE ESCRITORIO:

- El sistema debe permitir visualizar el listado de empleados.
- El sistema debe permitir seleccionar un empleado para modificar sus datos.
- El sistema debe permitir seleccionar un empleado para modificar sus datos de fichaje.
- El sistema debe notificar la información de las jornadas laborales de los empleados.
- El sistema debe permitir añadir un empleado.
- El sistema debe mostrar un formulario para añadir a un nuevo empleado en la plataforma.
- El sistema debe permitir definir reglas de fichaje.
- El sistema debe permitir eliminar a un empleado.
- El sistema debe permitir cancelar la eliminación de un empleado.
- El sistema debe mostrar una ventana emergente para confirmar la eliminación de un empleado.

REQUISITOS NO FUNCIONALES PARA LA APLICACIÓN MÓVIL:

- El sistema debe ser desarrollado para dispositivos Android
- El sistema debe ser compatible para la versión 6.0 (API 23, Marshmallow) o versiones posteriores.
- Los nuevos datos deben ser insertados en la base de datos en menos de 10 segundos.
- El tiempo para iniciar o reiniciar el sistema no podrá ser mayor a 5 minutos.
- La aplicación debe estar instalada en el dispositivo Android para su posterior uso.
- El sistema necesita acceso a la ubicación para realizar la geolocalización del empleado
- El sistema necesita acceso a internet para realizar los fichajes o justificaciones.
- El sistema necesita acceso a la autenticación por reconocimiento dactilar.

- El sistema debe permanecer activo mientras se realiza el reconocimiento dactilar.
- El sistema debe ser intuitivo y fácil de usar.
- El tiempo de aprendizaje del sistema para cada empleado debe ser menor a 20 minutos.
- El sistema hará uso de servicios Firebase para comunicarse con el servicio de autenticación, el servicio de almacenamiento y base de datos.

REQUISITOS NO FUNCIONALES PARA LA APLICACIÓN WEB O DE ESCRITORIO:

- El sistema debe ser desarrollado como aplicación web.
- El sistema debe ser compatible con Google Chrome y con Firefox.
- El sistema debe ser compatible con Windows.
- El sistema necesita acceso a internet para visualizar la información.
- El sistema debe ser intuitivo y fácil de usar.
- Los nuevos datos deben ser insertados en la base de datos en menos de 10 segundos.
- Los datos modificados en la base de datos deben ser actualizados en menos de 10 segundos.
- El tiempo de aprendizaje del sistema para cada administrador debe ser inferior a 35 minutos.
- El sistema hará uso de servicios Firebase para comunicarse con el servicio de autenticación, el servicio de almacenamiento y base de datos.

5. Historias técnicas

Las historias técnicas definen un conjunto de tareas que deben realizarse, las cuales no se entregan ni están directamente relacionadas con ninguna historia específica y no son de valor inmediato para el dueño del producto, como por ejemplo puede ser, la configuración de una base de datos.

A continuación, se muestran las historias técnicas del proyecto:

Historia técnica 1	
ID	***HT-1***
Nombre	***Estudio de las tecnologías (Android y firebase)***
Prioridad	***Alta***
Descripción	***Yo*** *como desarrollador* ***Deseo*** *aprender a programar en Android* ***Para*** *implementar el proyecto*

Historia técnica 2	
ID	***HT-2***
Nombre	***Creación de tablón Trello***
Prioridad	***Media-baja***
Descripción	***Yo*** *como desarrollador* ***Deseo*** *crear un tablón en Trello* ***Para*** *organizar las tareas*

Historia técnica 3	
ID	***HT-3***
Nombre	***Creación de repositorio GitHub***
Prioridad	***Media***
Descripción	***Yo*** *como desarrollador* ***Deseo*** *crear un repositorio en GitHub* ***Para*** *almacenar el proyecto*

Historia técnica 4	
ID	***HT-4***
Nombre	***Instalación y configuración del entorno de trabajo***
Prioridad	***Alta***
Descripción	***Yo*** *como desarrollador* ***Deseo*** *instalar el Android Studio* ***Para*** *realizar el desarrollo del proyecto*

Historia técnica 5	
ID	***HT-5***
Nombre	***Configuraciones Firebase previas al proyecto***
Prioridad	***Alta***
Descripción	***Yo*** *como desarrollador* ***Deseo*** *realizar las configuraciones de Firebase* ***Para*** *poder utilizar esta herramienta en el proyecto*

5.1.1. Historias de usuario

Una historia de usuario es una representación de un requisito escrito en una o dos frases, utilizando el lenguaje común del usuario. Las historias de usuario se usan para capturar la esencia del valor de negocio de un sistema usando lenguaje cotidiano.

Estas historias de usuario deben ser **Independientes** (para que cualquier miembro del equipo de desarrollo pueda centrarse en la historia y no dependa de otras), **negociables** (que se pueda cuantificar), **valiosas** (que tenga importancia o relevancia), **estimables** (que puedan estimarse), **pequeñas** (para que se puedan abordar en poco tiempo) y **testeables** (**INVEST**).

*** Las historias de usuario no son CASOS DE USO.**

A continuación, se muestran todas las historias de usuario del proyecto, organizadas según el usuario:

Usuario empleado.

Historia de usuario: Fichar entrada	
ID	HU-01
Nombre	Fichar entrada
Prioridad	Alta
Descripción	**Yo** como empleado

	Deseo poder fichar la entrada **Para** guardar el registro
Criterios de validación	▪ Vista que permita al usuario introducir su huella. ▪ Los datos se añaden correctamente a la base de datos.

Historia de usuario: Fichar salida	
ID	HU-02
Nombre	Fichar salida
Prioridad	Alta
Descripción	**Yo** como empleado **Deseo** poder fichar la salida **Para** guardar el registro
Criterios de validación	▪ Vista que permita al usuario introducir su huella. ▪ Los datos se añaden correctamente a la base de datos.

Historia de usuario: Fichaje geolocalizado	
ID	HU-03
Nombre	Fichaje geolocalizado
Prioridad	Alta
Descripción	**Yo** como empleado **Deseo** poder fichar con geolocalización **Para** registrar mi ubicación
Criterios de validación	La ubicación se guarda de manera exitosa en la base de datos.

Historia de usuario: Ver fichajes	
ID	HU-04
Nombre	Ver fichajes
Prioridad	Media

Descripción	**Yo** como empleado **Deseo** poder visualizar mis fichajes **Para** realizar comprobaciones
Criterios de validación	▪ Verificación de datos correctos. ▪ Vista que permite visualizar una lista de tarjetas (tarjetas con día, horario de entrada y horario de salida).

Historia de usuario: Ver detalles de fichaje	
ID	HU-05
Nombre	Ver detalles de fichaje
Prioridad	Media
Descripción	**Yo** como empleado **Deseo** poder ver los detalles del fichaje **Para** visualizar los datos correspondientes
Criterios de validación	▪ Verificación de datos correctos. ▪ Vista que permite visualizar tres tarjetas. Una con los detalles del horario de entrada del trabajador, otra tarjeta con los detalles del horario de salida del trabajador y finalmente una última tarjeta que permita visualizar la justificación de la falta si la hubiera.

Historia de usuario: Buscar por fecha	
ID	HU-06
Nombre	Buscar por fecha
Prioridad	Media
Descripción	**Yo** como empleado **Deseo** poder buscar fichajes por fecha **Para** comprobar el fichaje
Criterios de validación	▪ Visualización de la tarjeta que contiene la fecha, horario de entrada y salida del fichaje realizado en el día seleccionado si existiera. ▪ Si no existe el fichaje, se muestra un mensaje que lo comunique.

Historia de usuario: Justificar falta	
ID	HU-07
Nombre	Justificar falta
Prioridad	Alta
Descripción	**Yo** como empleado **Deseo** poder justificar una falta **Para** comunicar el motivo
Criterios de validación	▪ Vista que permite visualizar un formulario a rellenar con el motivo y documento de justificación correspondiente. ▪ Ventana de alerta que comunica la correcta justificación.

Historia de usuario: Ver justificantes	
ID	HU-08
Nombre	Ver justificantes
Prioridad	Media
Descripción	**Yo** como empleado **Deseo** poder visualizar el listado de justificantes **Para** ver su contenido
Criterios de validación	▪ Verificación de datos correctos. ▪ Vista que permite visualizar una lista de documentos pdf (justificantes) realizados por el empleado.

Historia de usuario: Ver justificante	
ID	HU-09
Nombre	Ver justificante
Prioridad	Baja
Descripción	**Yo** como empleado **Deseo** poder visualizar el justificante seleccionado **Para** ver su contenido

Criterios de validación	El sistema permite visualizar el contenido del justificante seleccionado.

Historia de usuario: Ver ayuda	
ID	HU-10
Nombre	Ver ayuda
Prioridad	Alta
Descripción	**Yo** como empleado **Deseo** poder visualizar la ayuda de la aplicación **Para** solventar dudas acerca de su utilización
Criterios de validación	Existe una vista que permite visualizar un listado con preguntas frecuentes.

Historia de usuario: Solicitar cambio de contraseña	
ID	HU-11
Nombre	Solicitar cambio de contraseña
Prioridad	Alta
Descripción	**Yo** como empleado **Deseo** poder solicitar cambio de contraseña **Para** poder acceder a la plataforma
Criterios de validación	▪ Vista que permite visualizar un formulario a rellenar por el empleado. ▪ El sistema muestra una ventana emergente con la confirmación de la solicitud.

Historia de usuario: Ver mis datos de perfil	
ID	HU-12
Nombre	Ver mis datos del perfil
Prioridad	Media
Descripción	**Yo** como empleado **Deseo** poder ver mi perfil **Para** visualizar mis datos

Criterios de validación	El empleado dispone de una vista que muestra estos datos: Imagen del empleado (si la hubiera), nombre y apellidos, compañía en la que trabaja, población, número de teléfono y DNI.

Historia de usuario: Modificar mis datos de perfil	
ID	HU-13
Nombre	Modificar mis datos de perfil
Prioridad	Media
Descripción	**Yo** como empleado **Deseo** poder modificar perfil **Para** actualizar mis datos
Criterios de validación	▪ Vista que permite visualizar un formulario a rellenar por el empleado. ▪ El sistema muestra una ventana emergente con la confirmación de la solicitud.

Historia de usuario: Ver ubicación	
ID	HU-14
Nombre	Ver ubicación
Prioridad	Baja
Descripción	**Yo** como empleado **Deseo** poder ver ubicación de mi fichaje **Para** comprobaciones
Criterios de validación	Vista que permite visualizar la ubicación del fichaje

Usuario administrador:

Historia de usuario: Añadir empleado	
ID	HU-15
Nombre	Añadir empleado
Prioridad	Alta
Descripción	**Yo** como administrador

	Deseo poder añadir empleado **Para** que haga uso de la aplicación
Criterios de validación	▪ Vista que permite visualizar un formulario a rellenar con los datos del nuevo cliente. ▪ Correcta inserción del nuevo usuario a la base de datos.

Historia de usuario: Definir reglas de fichaje	
ID	HU-16
Nombre	Definir reglas de fichaje
Prioridad	Alta
Descripción	**Yo** como administrador **Deseo** poder definir reglas de fichaje **Para** que el empleado fiche acorde a ellas
Criterios de validación	▪ Vista que permite visualizar un formulario a rellenar con los datos del fichaje del empleado. ▪ Correcta inserción del nuevo usuario a la base de datos.

Historia de usuario: Eliminar empleado	
ID	HU-17
Nombre	Eliminar empleado
Prioridad	Alta
Descripción	**Yo** como administrador **Deseo** poder eliminar empleado **Para** darlo de baja
Criterios de validación	▪ Vista que permite visualizar una ventana de alerta para confirmar la acción. ▪ Correcta eliminación del empleado de la base de datos. ▪ El empleado no se encuentra en la lista de empleados.

Historia de usuario: Modificar datos personales del empleado	
ID	HU-18
Nombre	Modificar datos personales del empleado
Prioridad	Media
Descripción	**Yo** como administrador **Deseo** poder modificar los datos personales de un empleado **Para** actualizar los mismos
Criterios de validación	▪ Vista que permite visualizar un formulario con los datos personales del empleado y que se puede modificar. ▪ Correcta inserción de los cambios en la base de datos.

Historia de usuario: Modificar reglas de fichaje	
ID	HU-19
Nombre	Modificar reglas de fichaje
Prioridad	Media
Descripción	**Yo** como administrador **Deseo** poder modificar las reglas de fichaje **Para** actualizar las mismas
Criterios de validación	▪ Vista que permite visualizar un formulario con los datos de fichaje del empleado y que se puede modificar. ▪ Correcta inserción de los cambios en la base de datos.

Historia de usuario: Ver listado de empleados	
ID	HU-20
Nombre	**Ver listado de empleados**
Prioridad	Alta
Descripción	**Yo** como administrador **Deseo** poder ver el listado de empleados **Para** visualizar los empleados registrados

Criterios de validación	▪ Verificación de los datos correctos (empleados pertenecientes a la empresa en cuestión). ▪ Vista que permite visualizar la lista de empleados de la empresa.

Usuario empleado y usuario administrador:

Historia de usuario: Visualizar contraseña	
ID	HU-21
Nombre	Visualizar contraseña
Prioridad	Baja
Descripción	**Yo** como usuario **Deseo** poder visualizar mi contraseña **Para** poder verificarla.
Criterios de validación	El usuario puede ver u ocultar la contraseña.

Historia de usuario: Login	
ID	HU-22
Nombre	Login
Prioridad	Alta
Descripción	**Yo** como usuario **Deseo** poder iniciar sesión en Clock-in On Time **Para** poder realizar gestiones
Criterios de validación	El usuario puede visualizar la pantalla inicial de la plataforma.

Historia de usuario: Cerrar sesión	
ID	HU-23
Nombre	**Cerrar sesión**
Prioridad	Alta
Descripción	**Yo** como usuario **Deseo** poder cerrar sesión

	Para dejar de estar identificado
Criterios de validación	▪ Se cierra sesión correctamente. ▪ El usuario visualiza la pantalla de inicio de sesión.

Historia de usuario: Toolbar (Menú superior)	
ID	HU-24
Nombre	**Menú superior**
Prioridad	Alta
Descripción	**Yo** como usuario **Deseo** poder visualizar el menú superior **Para** acceder a la ayuda y a cerrar sesión
Criterios de validación	Vista en forma de Toolbar que permite al usuario visualizar la ayuda y cerrar sesión

Historia de usuario: Bottom Menú (Menú inferior)	
ID	HU-25
Nombre	**Menú inferior**
Prioridad	Alta
Descripción	**Yo** como usuario **Deseo** poder visualizar el menú inferior **Para** navegar por la aplicación
Criterios de validación	Vista en forma de Bottom Menu que permite al usuario navegar por las distintas opciones de la app

Usuario visitante:

Historia de usuario: Ver información	
ID	HU-26
Nombre	**Ver información**
Prioridad	Baja
Descripción	**Yo** como visitante **Deseo** poder visualizar la información de la aplicación

	Para conocer su objetivo
Criterios de validación	El sistema permite visualizar la información relevante de la aplicación.

5.1.2. Diagrama de casos de uso

El diagrama de casos de uso es un esquema que representa la forma en como un cliente (actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan.

Los diagramas de casos de uso permiten mostrar el contorno (actores) y alcance (requisitos funcionales expresados como casos de uso) de un sistema. Además, describen la secuencia de interacciones que se producen entre el sistema y los actores del mismo para realizar una determinada función.

- Los actores son elementos externos (personas, otros sistemas, etc). Un actor puede participar en varios casos de uso y un caso de uso puede interactuar con varios actores.

La aplicación móvil está destinada a usuarios con el rol de "**Empleados**" y "**Visitantes**", en su versión de escritorio, destinada a usuarios con el rol de "**Administrador**" que serán los encargados de gestionar a los usuarios empleados y los informes obligatorios sobre el control de horario de los empleados. A continuación, en la siguiente figura se muestran los diagramas de casos de uso correspondientes al sistema.

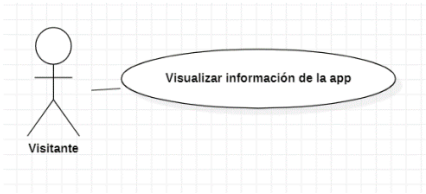

Ilustración 11. Diagrama de casos de uso (Visitante).

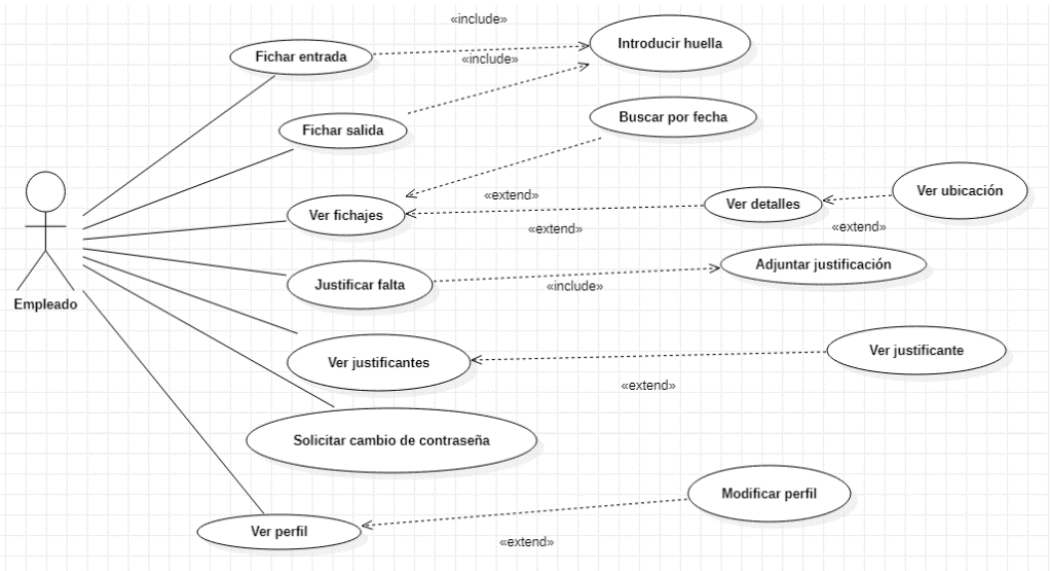

Ilustración 12. Diagrama de casos de uso (Empleado).

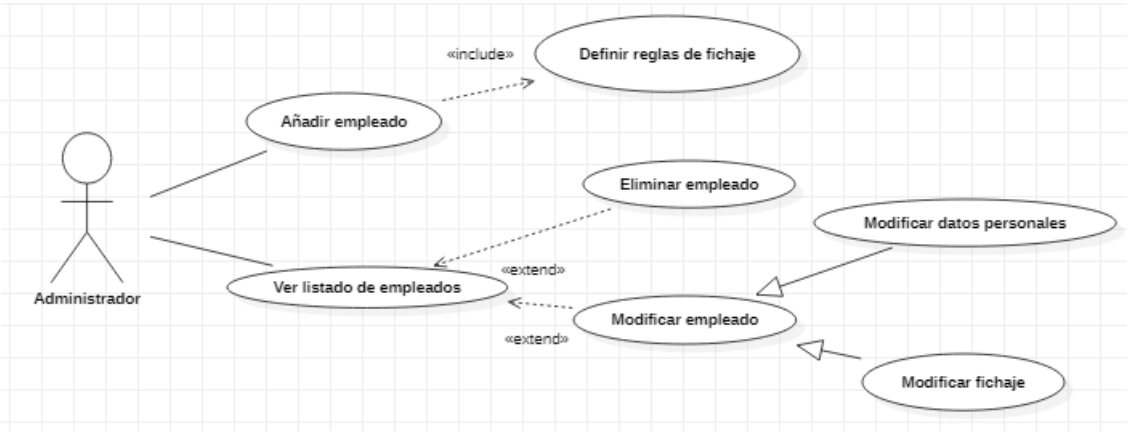

Ilustración 13. Diagrama de casos de uso (Administrador).

5.1.3. Especificaciones de casos de uso

¿Qué son las especificaciones de casos de uso y para qué sirven?

Una especificación de casos de uso es una descripción que proporciona detalles textuales de un caso de uso. A través de esta descripción, se reflejará el actor involucrado, las precondiciones, postcondiciones, posibles extensiones, variaciones y excepciones. También se introduce el flujo normal para el caso de uso dado.

A continuación, se muestran las especificaciones de casos de uso de los casos de uso identificados para la aplicación:

CASO DE USO	**1**	**Fichar entrada**		
Descripción		El empleado realiza el fichaje de entrada a la jornada laboral		
Actores		Empleado		
Precondiciones		El empleado ha iniciado sesión		
Flujo normal		**Paso**	**Acción**	
		1	El empleado pincha en “Fichar entrada”	
		2	El sistema le muestra el sistema de reconocimiento dactilar	
		3	El usuario introduce su huella	
		4	El sistema muestra la hora de fichaje en el cuadro correspondiente	
Postcondiciones		El fichaje se ha realizado con éxito		
Variaciones		**Paso**	**Acción**	
		3A	La huella introducida no es correcta	
Extensiones		**Paso**	**Condición**	**Caso de Uso**

CASO DE USO	2	Fichar salida		
Descripción		El empleado realiza el fichaje de salida a la jornada laboral		
Actores		Empleado		
Precondiciones		El empleado ha iniciado sesión		
Flujo normal		Paso	Acción	
		1	El empleado pincha en "Fichar salida"	
		2	El sistema le muestra el sistema de reconocimiento dactilar	
		3	El usuario introduce su huella	
		4	El sistema muestra la hora de fichaje en el cuadro correspondiente	
Postcondiciones		El sistema muestra el proyecto y el jefe asignado a dicho proyecto.		
Variaciones		Paso	Acción	
		3A	La huella introducida no es correcta	
Extensiones		Paso	Condición	Caso de Uso

CASO DE USO	3	Ver fichajes		
Descripción		El empleado visualiza el listado de fichajes		
Actores		Empleado		
Precondiciones		El empleado ha iniciado sesión		
Flujo normal		Paso	Acción	
		1	El empleado pulsa en "Fichajes"	
		2	El sistema le muestra el listado de fichajes	
Postcondiciones		Se muestra el listado de fichajes		
Variaciones		Paso	Acción	
Extensiones		Paso	Condición	Caso de Uso
		2A	El usuario quiere ver los datos concretos del fichaje	CU 4: Ver detalles

CASO DE USO	4	Ver detalles		
Descripción		El empleado visualiza los detalles de un fichaje		
Actores		Empleado		
Precondiciones		El empleado ha iniciado sesión		
Flujo normal		Paso	Acción	
		1	El empleado pincha en un fichaje concreto (A partir del listado de fichajes)	
		2	El sistema muestra la información del fichaje	
Postcondiciones		El empleado ha visualizado los datos del fichaje		
Variaciones		Paso	Acción	
Extensiones		Paso	Condición	Caso de Uso

CASO DE USO	5	Buscar por fecha		
Descripción		El empleado visualiza un fichaje concreto		
Actores		Empleado		
Precondiciones		El empleado ha iniciado sesión y ha introducido una fecha		
Flujo normal		Paso	Acción	
		1	El empleado pincha en "Calendario"	
		2	El empleado selecciona una fecha	
		3	El sistema muestra un mensaje con la fecha seleccionada	
		4	El empleado confirma el filtrado	
		5	El sistema muestra el fichaje para la fecha seleccionada	
Postcondiciones		Se visualiza el fichaje para la fecha seleccionada		
Variaciones		Paso	Acción	
Extensiones		Paso	Condición	Caso de Uso
		5A	El empleado quiere ver los detalles del fichaje filtrado.	CU 4: Ver detalles
Excepciones		5	No existe fichaje para la fecha seleccionada	

CASO DE USO	6	Justificar falta		
Descripción		El empleado justifica la falta al trabajo		
Actores		Empleado		
Precondiciones		El empleado ha iniciado sesión		
Flujo normal		Paso	Acción	
		1	El empleado pincha en "Justificar falta"	
		2	El sistema muestra el formulario a rellenar	
		3	El empleado rellena el formulario	
		4	El empleado adjunta la justificación	
		5	El empleado pulsa en "Justificar"	
		6	El sistema muestra un mensaje de éxito	
Postcondiciones		Se ha justificado la falta		
Variaciones		Paso	Acción	
		5A	El empleado pulsa en "cancelar"	
Extensiones		Paso	Condición	Caso de Uso
Excepciones		4	El empleado introduce un archivo en un formato no válido	
		5	El empleado no adjunta archivo PDF	

CASO DE USO	7	Ver justificantes		
Descripción		El empleado visualiza el listado de justificantes adjuntados		
Actores		Empleado		
Precondiciones		El empleado ha iniciado sesión		
Flujo normal		Paso	Acción	
		1	El empleado pincha en “Justificantes”	
		2	El sistema muestra el listado de justificantes	
Postcondiciones		Se muestra el listado de justificantes		
Variaciones		Paso	Acción	
Extensiones		Paso	Condición	Caso de Uso
		3	El empleado quiere ver el documento de justificación	CU 8: Ver justificante

CASO DE USO	8	Ver justificante		
Descripción		El empelado visualiza el contenido del fichero adjuntado como justificante		
Actores		Empleado		
Precondiciones		El empleado no ha iniciado sesión y ha pulsado un justificante del listado		
Flujo normal		Paso	Acción	
		1	El empleado pincha en un justificante a partir del listado de justificantes	
		2	El sistema le muestra el contenido del documento.	
Postcondiciones				
Variaciones		Paso	Acción	
Extensiones		Paso	Condición	Caso de Uso

CASO DE USO	9	Solicitar cambio de contraseña		
Descripción		El empleado ha olvidado su contraseña y solicita el cambio		
Actores		Empleado		
Precondiciones		El empleado no ha iniciado sesión		
Flujo normal		Paso	Acción	
		1	El sistema muestra la pantalla principal	
		2	El empleado pulsa en “Olvidé mi contraseña”	
		3	El sistema muestra el formulario a rellenar	
		4	El empleado rellena el formulario	
		5	El sistema muestra un mensaje de confirmación (Envío de instrucciones al correo correspondiente)	
Postcondiciones				
Variaciones		Paso	Acción	
Extensiones		Paso	Condición	Caso de Uso

CASO DE USO	10	Ver información	
Descripción	El visitante visualiza la información de la aplicación		
Actores	Visitante		
Precondiciones	No se ha iniciado sesión		
Flujo normal	Paso	Acción	
	1	El sistema muestra al visitante la pantalla principal	
	2	El visitante pulsa en "Información"	
	3	El sistema muestra la información de la aplicación	
Postcondiciones	Se ha creado el informe del proyecto con éxito		
Variaciones	Paso	Acción	
Extensiones	Paso	Condición	Caso de Uso

CASO DE USO	12	Modificar perfil	
Descripción	El empleado quiere modificar sus datos del perfil		
Actores	Empleado		
Precondiciones	El empleado ha iniciado sesión		
Flujo normal	Paso	Acción	
	1	El empleado pulsa en "Editar perfil"	
	2	El sistema muestra un formulario con sus datos (para ser modificado).	
	3	El empleado rellena el formulario	
	4	El empleado pulsa en "Modificar"	
	5	El sistema muestra una ventana emergente con la confirmación de la modificación	
	6	El sistema redirige a la vista del perfil	
Postcondiciones	El perfil ha sido editado		
Variaciones	Paso	Acción	
	4A	El empleado pulsa en "Cancelar"	
Extensiones	Paso	Condición	Caso de Uso

CASO DE USO	13	Ver ubicación	
Descripción	El empleado quiere ver la ubicación de su fichaje en el Google Maps		
Actores	Empleado		
Precondiciones	El empleado ha iniciado sesión y existe el fichaje para la fecha seleccionada		
Flujo normal	Paso	Acción	
	1	El empleado pulsa en "Ver fichajes"	
	2	El sistema muestra la lista de fichajes	
	3	El empleado pulsa en un fichaje concreto	
	4	El sistema muestra los detalles del fichaje	
	5	El usuario pulsa en "Ver ubicación"	
	6	El sistema muestra la ubicación en Google Maps	
Postcondiciones	El sistema ha mostrado la ubicación en Google Maps		
Variaciones	Paso	Acción	
Extensiones	Paso	Condición	Caso de Uso

6. Diseño

6.1. Diseño de la interfaz de usuario

¿Qué es una interfaz de usuario?

La interfaz de usuario es el medio con el que el usuario puede comunicarse con una máquina. Equipo, computadora o dispositivo, y comprende todos los puntos de contacto entre el usuario y el equipo. [29]

¿Qué son los Bocetos o Wireframes?

Los bocetos o wireframes son una representación visual de una interfaz (UI) en la que se utilizan formas simples como cajas, círculos, líneas y flechas para así mostrar una estructuración básica del contenido y la jerarquía de la información. En definitiva, son dibujos muy sencillos para que el cliente pueda ver una etapa temprana del producto sin que se haya desarrollado, con el fin de poder tomar decisiones básicas en cuanto a la organización del contenido. [30]

6.1.1. Boceto del proyecto

El *boceto* o *wireframe* es uno de los pasos más importantes en el diseño de una aplicación, ya que contiene una idea básica del diseño, centrándose en la organización de los elementos y usabilidad, es por ello que, decidí definir en primer lugar, las historias de usuario y casos de uso, y una vez completado este primer paso, comencé a realizar el boceto de la aplicación, con el que tener una idea aproximada de cómo podría organizarse la interfaz de usuario.

Para realizar esta segunda etapa del proyecto hice uso del programa "**Mockplus**", con el que realicé el boceto del proyecto, que se muestra a continuación:

Ilustración 16. Boceto: Pantalla inicial

:

Ilustración 15. Boceto: Olvidé mi contraseña

Ilustración 14. Boceto: Pantalla inicial

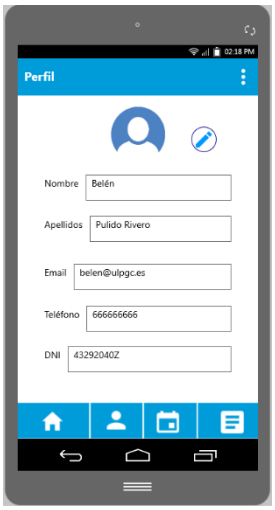

Ilustración 18. Boceto: Perfil

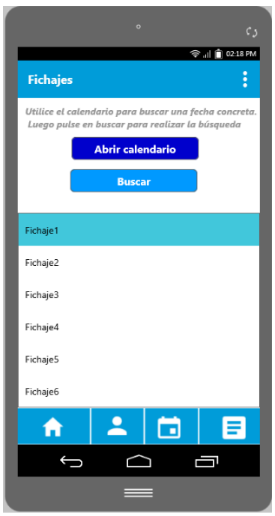

Ilustración 17. Boceto: Listado de fichajes

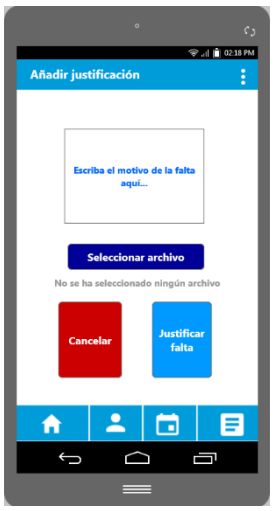

Ilustración 19. Boceto: Justificación de falta

Ilustración 22. Boceto: Ayuda

Ilustración 20. Boceto: Modificar perfil

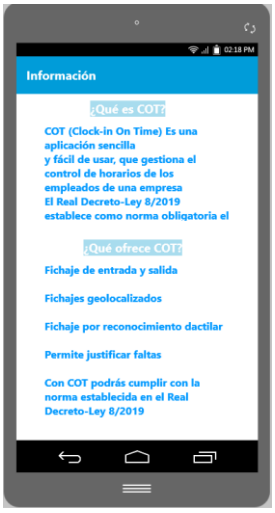

Ilustración 21. Boceto: Acerca de

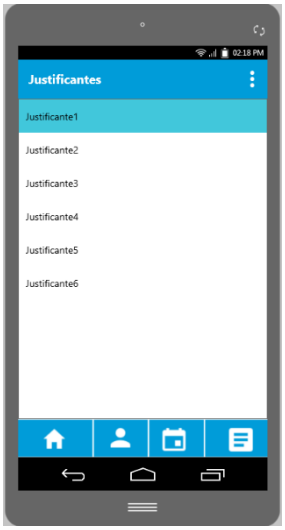

Ilustración 23. Boceto: Listado de justificantes

6.1.2. Elección de tipografía y gama de colores

Los colores y la tipografía para una aplicación son elementos que deben ser escogidos y estructurados detenidamente. Son elementos que facilitan la usabilidad y legibilidad de la aplicación.

Tipografía elegida

Para el proyecto, he elegido la fuente "*sans-serif*". Es una fuente que transmite seguridad, actualidad, profesionalidad, dinamismo y minimalismo y muchas de las aplicaciones más famosas a día de hoy, utilizan esta fuente.

Dentro del proyecto, esta fuente ha sido utilizada con diferentes formatos:

- El formato **negrita** ha sido utilizado para destacar mensajes o información relevante para el usuario, como puede ser por ejemplo, las horas a las que ha fichado.
- El formato *cursiva* ha sido utilizado para captar la atención del usuario, por su forma.

**** En algunas ocasiones se han combinado ambos formatos para captar aún más la atención del usuario.***

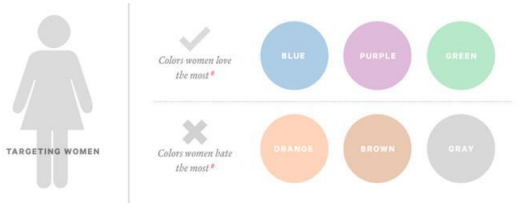

Ilustración 24. Colores más elegidos por mujeres. [34]

Ilustración 25. Colores más elegidos por hombres [35]

Colores elegidos

Para la elaboración de este TFT, he elegido el color blanco y una gama cromática derivada del azul.

Una vez analizadas todas las posibles competencias en el mercado, he observado que todas, o la mayoría utilizan para sus aplicaciones, colores muy llamativos, como por ejemplo el naranja (que trasmite coraje, simpatía y/o éxito) o amarillo (que transmite felicidad, brillo, calor y/o energía).

Para la elección de los colores, me he centrado en observar aplicaciones con éxito, como por ejemplo Whatsapp, aplicación minimalista, (en cuanto a la elección de su tema) que utiliza el color verde y el blanco.

El color azul tranquiliza nuestra mente y nos proporciona frescura y transparencia, así como madurez. Es el color del cielo y del mar, por lo que nos genera sensación de profundidad. Además, es un color que transmite seguridad, confianza, sanidad y/o lealtad.

El color blanco, es un color que se relaciona con la luz y amplitud y nos conduce a pensar en bondad, pureza y sinceridad. Además, transmite tranquilidad y seriedad y nos empuja a ideas positivas.

Es por ello, que además de lo que el color azul y el blanco transmiten, otros motivos que me llevaron a utilizarlos fueron, qué no encontré ninguna aplicación del sector que los utilizara y porque me gusta lo sencillo y minimalista.

6.2. Diseño del software

6.2.1. Arquitectura del proyecto

Patrón MVC

El patrón utilizado para el desarrollo del proyecto es el patrón **MVC** (Modelo-Vista-Controlador).

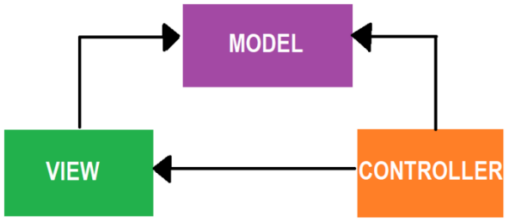

Ilustración 26. Arquitectura del patrón MVC

MVC es un patrón de arquitectura software que, utilizando tres componentes (vistas, modelos y controladores) separa la lógica de la aplicación de la lógica de la vista en una aplicación. Es una arquitectura importante puesto que se utiliza tanto en componentes gráficos básicos hasta en sistemas empresariales.

- **Vista**: Se corresponde con los ***layouts*** de Android. Estos ***layouts*** (ficheros XML) son un conjunto de contenedores en donde se pueden colocar elementos con diseño, como por ejemplo botones, imágenes, textos, etc.
- **Controlador**: Se corresponde con los ficheros ***Activity*** de Android. Se corresponden con los archivos ***.java*** que contienen la lógica y se comunican con los ***layouts***.
- **Modelo**: Son los ficheros ***.java*** que contienen la clase que representan a las entidades. Como por ejemplo la entidad ***Usuario***.

A continuación, se muestra un diagrama que muestra el funcionamiento del patrón Modelo-Vista-Controlador.

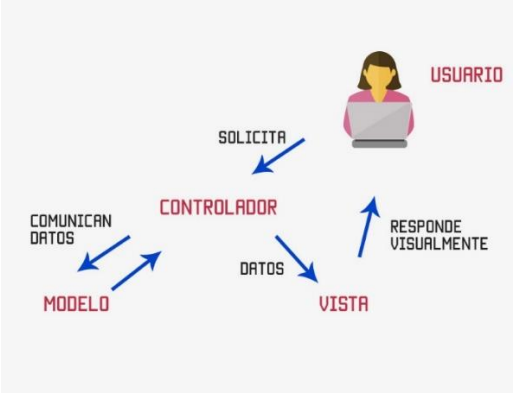

Ilustración 27. Explicación del funcionamiento de MVC [31]

Comunicación con el servidor

El servidor utilizado es **Firebase** (que se ha mencionado anteriormente). La aplicación trabaja en comunicación con dicho servidor. El servidor **Firebase** tiene muchos servicios. Entre ellos, se ha hecho uso de su servicio de autenticación (**Authentication**) y el servicio de la base de datos en tiempo real (**Realtime database**).

Autenticación

Cuando se inicia la aplicación y el usuario introduce sus credenciales, a través del servicio de **Firebase Authentication** (en este caso, una autenticación por email y contraseña), se omprueba si ese usuario existe.

Ilustración 28. Diagrama de comunicación de la aplicación y Firebase Authentication

Base de datos

COT hace uso de una base de datos en tiempo real. Estas bases de datos son bases de datos NoSQL alojadas en la nube. Los datos que existen en ella se almacenan en formato JSON y se sincronizan con todos los clientes en tiempo real, (si algún dato cambia, los dispositivos conectados recibe la actualización en cuestión de milisegundos), y se mantienen disponibles incluso si la app no tiene conexión.

La manera en la que se comunica la aplicación con esta base de datos es directa, es decir, el dispositivo se comunica directamente con la base de datos.

Además, la seguridad y la validación de los datos están disponibles a través de las reglas de seguridad de **Firebase Realtime Database**, reglas basadas en expresiones que se ejecutan cuando se leen o escriben datos. Por ello, para dar más seguridad, se introdujeron algunas reglas, que se muestran como resultado en la ilustración 14.

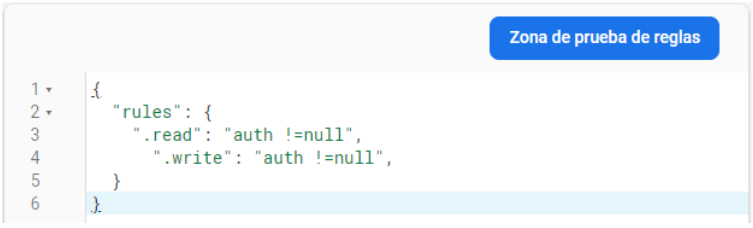

Ilustración 29. Reglas introducidas para dar segurdad a la base de datos Firebase.

6.2.2. Diagrama de clases

¿Qué es un diagrama de clases y para qué sirve?

Un diagrama de clases es un tipo de diagrama que sirve para visualizar las relaciones entre las clases que involucran al sistema. Estas clases pueden ser asociativas, de herencia, de uso y de agregación. [32]

A continuación, se muestra el diagrama de clases del proyecto:

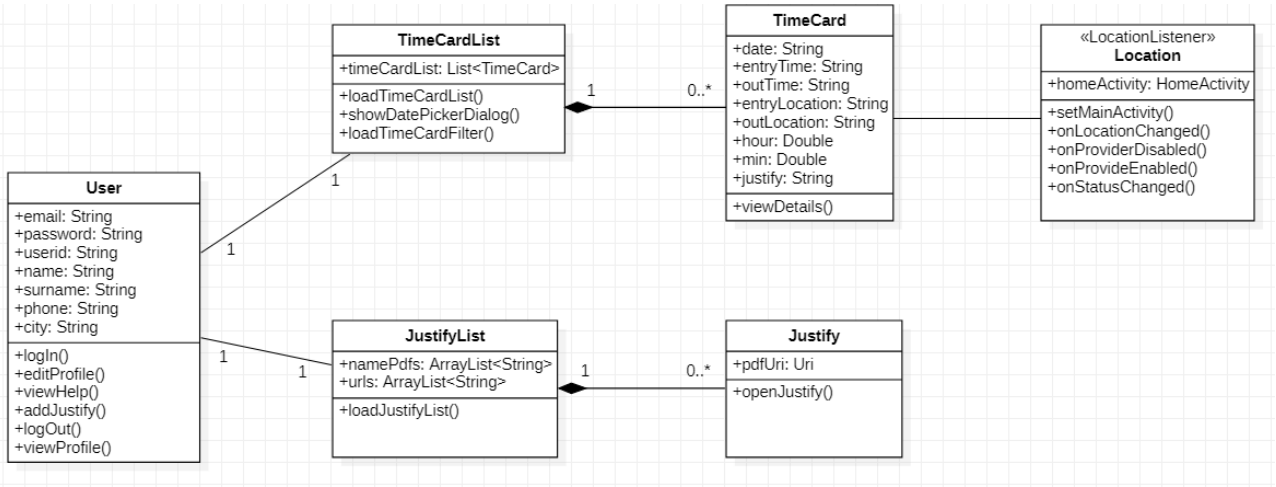

Ilustración 30. Diagrama de clases del TFT

Como se puede observar, en este diagrama de clases se muestra la entidad **Usuario** y sus acciones (*Iniciar sesión, editar perfil, visualizar la ayuda, añadir justificante, cerrar sesión y visualizar perfil*). Esta entidad tiene relación directa con el listado de fichajes y el listado de justificantes. El listado de fichajes se compone de una lista que almacena objetos del tipo **TimeCard** (*Fichaje*), que, a su vez, tienen relación directa con la entidad **Location**, que es la encargada de obtener la ubicación de los fichajes. Por otro lado, se encuentra el listado de justificantes que estará formado por archivos pdf (*justificantes*). Estos justificantes tienen como atributo una ***Uri*** para obtener la ubicación del archivo dentro del almacenamiento de **Firebase**.

6.3. Diseño de la base de datos

¿Qué es JSON?

Es un formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de JavaScript. Los tipos de datos disponibles en **JSON** son números, cadenas, booleanos, null, arrays y objetos.

En cuanto a la base de datos del proyecto, como bien se mencionaba en puntos anteriores, **Firebase** proporciona una base de datos en tiempo real, siendo una base de datos NOSQL, estructurada en objetos **JSON**, lo cual quiere decir que no existen ni tablas ni registros, de manera que, al agregar datos a la base de datos, estos se convierten

Ilustración 31. Estructura de la base de datos del proyecto.

en un nodo **JSON** existente con una clave asociada. Estos datos se pueden representar como determinados tipos nativos que se corresponden con los tipos de JSON disponibles, lo cual hace que escribamos un código más fácil de mantener.

En la ilustración 31 se puede observar la estructura de la base de datos del proyecto, viéndose reflejada la estructura **JSON**.

7. Estructura y elementos del proyecto

7.1. Estructura del proyecto Android

7.1.1. Archivos a nivel de proyecto

Dentro de un proyecto Android existen dos carpetas. Por un lado, se encuentra la carpeta **Grandle Script** y por otro lado se encuentra la carpeta **App**.

La carpeta **Grandle Script** contiene información necesaria para la compilación del proyecto, como, por ejemplo, la versión del SDK de Android utilizada para compilar, la mínima versión de Android que soportará la aplicación, referencias a librerías externas utilizadas, etc.

La carpeta App a su vez, se divide en cinco carpetas más: Manifest, Java, Res.

- **App/Manifest** contiene el archivo AndroidManifest.xml que describe información esencial de la aplicación para las herramientas de creación de Android, el sistema operativo Android y Google Play.
- **App/Java** contiene los archivos de código fuente Java separados por nombres de paquetes incluido el código de prueba JUnit.
- **App/build** contiene una serie de elementos de código que se generan automáticamente al compilar el proyecto. También contiene el fichero "R.java" donde se define la clase R. Esta clase R contendrá una serie de constantes con los identificadores de todos los recursos de la aplicación incluidos.
- **App/res** es una carpeta que a su vez se divide en cinco carpetas más: App/res/drawable, App/res/layout, App/res/menu, App/res/mipmap y App/res/values.
    - **App/res/drawable:** Para archivos de mapas de bits (.png, .jpj, etc) o archivos XML para recursos dibujables.
    - **App/res/layout:** Para archivos XML que definen el diseño de una interfaz de usuario.
    - **App/res/menu:** Para archivos XML que definen menús de aplicaciones, como por ejemplo menús de opciones, menús conceptuales o submenús.
    - **App/res/mipmap**: Archivos de elementos de diseño para diferentes densidades de los iconos de selectores.
    - **App/res/values**: Para archivos XML que contienen valores simples como strings, valores enteros y colores.
        - **Arrays.xml** para matrices de recursos.
        - **Colors.xml** para valores de color.
        - **Dimens.xml** para valores de dimensión
        - **Strings.xml** para valores de strings
        - **Styles.xml** para estilos.

7.1.2. Todos los módulos del proyecto Android

En la estructura real del proyecto se incluyen todos los archivos ocultos de la vista de Android. A continuación, se describen los módulos existentes:

- **Build/** contiene resultados de compilación

- **Libs/** contiene las librerías java externas (ficheros .jar) que utilice la aplicación. Normalmente se hacen referencia a dichas librerías a través del fichero build.gradle.
- **Src/** Contiene todos los archivos de código y recursos para el módulo en los siguientes directorios:
    - **androidTest/** contiene el código para las pruebas de instrumentación que se ejecutan en un dispositivo Android.
    - **Main/** contiene los archivos de conjunto de fuentes "principales": el código y los recursos Android compartidos por todas las variantes de compilación (los archivos para otras variantes residen en directorios del mismo nivel como */src/debug/* para el tipo de compilación de depuración).
    - **AndroidManifest.xml:** Describe la naturaleza de la aplicación y cada uno de sus componentes.
    - **Java/** contiene fuentes de código java.
    - **Jni/** contiene el código nativo en el cual se usa la interfaz nativa de java (JNI).
    - **Gen/** contiene los archivos Java que genera Android Studio como el archivo R.java y las interfaces creadas desde los archivos AIDL.
    - **Res**/ contiene recursos de aplicación como archivos de elementos de diseño, archivos de diseño y strings de IU.
    - **Assets/** contiene el archivo que se debe compilar en un archivo .apk tal como está.
- **Test/** contiene el código de pruebas locales que se ejcutan en el JVM de host.
- **Build.gradle** (a nivel de módulo) es un archivo que define las configuraciones de compilación específicas para el módulo.
- **Build.gradle** (a nivel de proyecto) es un archivo que define la configuración de compilación que se aplica a todos los módulos.

7.2. Dependencias del proyecto

¿Qué es la inyección de dependencias y para qué sirve?

Es un patrón de diseño orientado a objetos, en el que se suministran objetos a una clase en lugar de ser la propia clase quien cree dichos objetos. Esos objetos cumplen contratos que necesitan nuestras clases para poder funcionar (de ahí el concepto de dependencia). [33]

En otras palabras, se trata de un patrón de diseño que se encarga de extraer la responsabilidad de la creación de instancias de un componente para delegarla en otro.

7.2.1. Dependencias inyectadas en el proyecto

Para poder hacer uso de determinados objetos y servicios en el proyecto, además de las que el proyecto inyecta automáticamente en su creación, se tuvieron que inyectar una serie de dependencias más:

- **Firebase-core:17.4.4**: Para agregar el SDK de firebase al proyecto.
- **Firebase-auth:19.3.2**: Para hacer uso del servicio de autenticación de firebase.
- **Firebase-database:19.3.1**: Para hacer uso del servicio de base de datos de firebase.
- **Firebase-storage:17.0.0**: Para hacer uso del servicio de almacenamiento de firebase.
- **Cardview:1.0.0**: Para poder utilizar el objeto "tarjeta" de Android (CardView).
- **Recyclerview:1.1.0**: Para poder utilizar el objeto recyclerView que nos proporciona Android (Vista en forma de lista).

7.3. Elementos importantes en Android Studio

Existe una serie de elementos clave que resultan imprescindibles para desarrollar aplicaciones en Android:

- **Vista (View):** Las vistas son los elementos que componen la interfaz de usuario de una aplicación, por ejemplo, un botón o entrada de texto. Todas las vistas van a ser objetos descendientes de la clase View, y por tanto, pueden ser definidas utilizando código Java, aunque lo habitual es definirlas utilizando un fichero *XML.*
- **Layout**: Un layout es un conjunto de vistas agrupadas de una determinada forma.
- **Actividad (Activity):** Una aplicación en Android va a estar formada por un conjunto de elementos básicos de visualización (pantallas de aplicación). En Android cada uno de estos elementos o pantallas se conoce como actividad.
- **Fragmento (Fragment)**: Un fragment está formado por la unión de varias vistas para crear un bloque funcional de la interfaz de usuario. Estos fragmentos pueden combinarse dentro de una actividad.
- **Intención (Intent)**: Una intención representa la voluntad de realizar alguna acción. Se suele utilizar cada vez que se quiere lanzar una actividad, un servicio, un anuncio o comunicarse con un servicio.

8. Desarrollo del proyecto

8.1. Historias de usuario desarrolladas en el proyecto

Las aplicaciones del estilo de la escogida para este TFT necesitan una aplicación móvil que permita a los empleados realizar los fichajes diarios de la jornada laboral y otra aplicación que permita controlar todos los datos de los empleados, y a través de la cual, el administrador que se encargue de gestionar a los usuarios, en este caso, a los empleados.

En este TFT se va a desarrollar un prototipo funcional que representa una aplicación móvil a través de la cual, los empleados puedan realizar fichajes, es por tal motivo, que se han escogido las siguientes historias de usuario a desarrollar:

- **HU-01 (Fichar entrada)**: Funcionalidad que permite al empleado realizar el fichaje de entrada a través de su huella dactilar.
- **HU-02 (Fichar salida)**: Funcionalidad que permite al empleado realizar el fichaje de salida a través de su huella dactilar.
- **HU-03 (Fichaje geolocalizado)**: Funcionalidad que permite al empleado, al fichar, guardar la ubicación en la que realizó el fichaje.
- **HU-04 (Ver fichajes)**: Funcionalidad que permite al empleado ver el listado de sus fichajes.
- **HU-05 (Ver detalles de fichaje)**: Funcionalidad que, a través del listado de fichajes, permite al empleado pinchar en un fichaje concreto para visualizar sus detalles (Hora de entrada, hora de salida y ubicaciones)
- **HU-06 (Buscar por fecha)**: Funcionalidad que permite al empleado filtrar el listado de fichajes mediante una fecha concreta, con el fin de visualizar el fichaje existente en esa fecha.
- **HU-07 (Justificar falta)**: Funcionalidad que permite al empleado realizar una justificación de falta al trabajo.
- **HU-08: (Ver justificantes)**: Funcionalidad a través de la cual, el empleado puede acceder al listado de justificantes adjuntados.
- **HU-09 (Ver justificante)**: Funcionalidad que permite al empleado, visualizar el contenido de un justificante.
- **HU-10: (Ver ayuda)**: Funcionalidad que permite al empleado, visualizar un listado de preguntas frecuentes sobre la utilización de la aplicación.
- **HU-11: (Solicitar cambio de contraseña)**: Funcionalidad que permite al empleado, realizar una solicitud de cambio de contraseña.
- **HU-12: (Ver mis datos de perfil)**: Funcionalidad que permite al usuario visualizar sus datos del perfil.
- **HU-13 :(Modificar mis datos de perfil)**: Funcionalidad que permite al usuario actualizar sus datos del perfil.
- **HU-14 (Ver ubicación):** Funcionalidad que permite al usuario ver la ubicación de su fichaje.
- **HU-21: (Visualizar contraseña)**: Función que permite al usuario ver u ocultar su contraseña.
- **HU-22: (Login)**: Funcionalidad que permite al empleado iniciar sesión en la plataforma.

- **HU-23: (Cerrar sesión)**: Funcionalidad que permite al empleado dejar de estar identificado en la plataforma.
- **HU-24: (Menú superior)**: Menú superior (Toolbar) para la app.
- **HU-25: (Menú inferior)**: Menu inferior (Bottom Menu) para la app
- **HU-26: (Ver información)**: Esta funcionalidad permite a los visitantes de la app visualizar la información de la app (qué es y que proporciona).

8.2. Sprints realziados

A continuación, se describen los detalles de todos los sprints del proyecto.

En el Anexo nº 1 se puede observar la pila de producto, que contiene todas las historias técnicas e historias de usuario del proyecto. Estas historias aparecen con su prioridad, tiempo estimado y tiempo real de realización.

8.2.1. Sprint 0

Este sprint se corresponde con el sprint inicial del proyecto. Durante este sprint las tareas realizadas fueron tareas no funcionales, relacionadas con el estudio de las tecnologías a utilizar y configuraciones pertinentes para iniciar el proyecto (realización de las historias técnicas).

ID	Nombre	Estimación
HT-1	**Estudio de las tecnologías (Android y Firebase)**	**80 horas**
HT-2	**Creación de tablón Trello**	**35 minutos**
HT-3	**Creación de repositorio GitHub**	**5 minutos**
HT-4	**Instalación y configuración del entorno de trabajo**	**1 hora**
HT-5	**Configuraciones Firebase previas al proyecto**	**15 horas**
Tiempo estimado		**96 horas y 40 minutos.**

Tabla 3. Pila de sprint 0

A continuación, se detallan las tareas realizadas para cada historia técnica de este sprint inicial:

HT-1: Estudio de las tecnologías (Android y firebase)

Esta historia técnica consta de dos tareas:

- **Tarea 1: Realización de curso sobre Android Studio**: Al no tener ningún conocimiento previo sobre aplicaciones móviles, decidí realizar un curso desde cero sobre Android Studio, utilizando el lenguaje Java (Android Studio es la herramienta oficial para la creación de aplicaciones Android).
- **Tarea 2**: **Realización de curso sobre Firebase**: Firebase es una tecnología proporcionada por Google, la cual presta algunos servicios como autenticación,

almacenamiento o base de datos en tiempo real. Al no tener ningún conocimiento previo sobre como funciona esta herramienta, decidí realizar un curso sobre ello, con el fin de entenderla y poder aplicarla al proyecto propuesto.

HT-2: Creación de tablón Trello

Para la realización de esta tarea, accedí a www.trello.com y una vez dentro, comencé a diseñar un tablón con el fin de organizar las tareas a realizar a lo largo de todo el proyecto. Una vez creado el tablón, comencé a introducir tarjetas con las diferentes tareas a realizar y detallando su contenido.

HT-3: Creación de repositorio GitHub

Para la realización de esta tarea, accedí en primer lugar, a www.github.com. Una vez dentro, procedí a crear un nuevo repositorio donde almacenar los archivos y ficheros del proyecto.

HT-4: Instalación y configuración del entorno de trabajo

Esta historia técnica consta de 2 tareas, las cuales se desglosan a continuación:

- **Tarea 1: Descarga e instalación del entorno de trabajo**: En primer lugar, accedí al sitio oficial para descargar el Android Studio. Una vez descargado comencé a instalarlo.
- **Tarea 2: Configuración del entorno de trabajo**: Una vez instalado el Android Studio, comencé a realizar su configuración. En primer lugar, procedí a configurar los SDK correspondientes y finalmente configuré el proyecto para sincronizarlo con el repositorio GitHub.

HT-5: Configuraciones Firebase previas al proyecto

Esta historia técnica consta de 4 tareas detalladas a continuación:

- **Tarea 1: Visualización de tutoriales sobre la configuración de Firebase en un proyecto Android**: Para la realización de esta tarea, decidí visualizar unos tutoriales que me ayudaran a realizar las configuraciones correspondientes sobre el proyecto para poder utilizar esta herramienta.
- **Tarea 2: Configuración del servicio de autenticación de Firebase**: Firebase proporciona un servicio de autenticación a través del cual, permite iniciar sesión en la aplicación de diferentes maneras, como por ejemplo, iniciar sesión con la cuenta de Google, de Facebook o Instagram, pero en mi caso, a mi aplicación solo se podrá acceder mediante e-mail y contraseña, los cuales deben ser facilitados por la compañía en cuestión, es por ello, que se tuvo que configurar dicho servicio para que el método de acceso solo fuera este.
- **Tarea 3**: **Configuración de la base de datos Real Time Database de Firebase**: En primer lugar, cree la base de datos correspondiente y una vez creada, comencé a configurarla. Con el fin de ofrecer mayor seguridad sobre los datos almacenados en dicha base de datos, configuré las reglas de la base de datos para que solo los usuarios autenticados pudieran realizar lecturas o escrituras en dicha base de datos. Dichas reglas se muestran en la ilustración 29.

- **Tarea 4: Configuración del almacenamiento en Firebase**: Para ello, en primer lugar, se creó un depósito predeterminado de Storage. Luego procedí a agregar el SDK de Cloud Storage para la aplicación y finalmente comencé a realizar la configuración del Cloud Storage.

8.2.2. Sprint 1

El objetivo principal de este Sprint fue realizar el sistema de fichaje básico de la aplicación, es decir, realizar las historias de usuario correspondientes para que los usuarios pudieran realizar fichajes o justificaciones (principales funcionalidades), de manera que, he querido hacer un producto mínimo viable.

- *Producto mínimo viable:* Es un producto con suficientes características para satisfacer a los clientes iniciales y proporcionar retroalimentación para el desarrollo futuro. De esta manera, al final de cada sprint, se le entregará al cliente un producto de manera que pueda irlo utilizando mientras se siguen desarrollando las demás funcionalidades del mismo.

ID	Nombre	Estimación
HU-01	**Fichar entrada**	**8 horas**
HU-02	**Fichar salida**	**8 horas**
HU-03	**Fichaje geolocalizado**	**10 horas**
HU-07	**Justificar Falta**	**7 horas**
HU-22	**Login**	**3 horas**
HU-23	**Cerrar sesión**	**1 hora**
HU-24	**Menú superior**	**1 hora**
HU-25	**Menú inferior**	**3 horas**
Tiempo estimado		**41 horas**

Tabla 4. Pila de sprint 1

A continuación, se detallan las tareas realizadas para cada historia de usuario de este sprint inicial. Las historias de usuario se describen en el orden en el que fueron realizadas:

HU-24: Menú superior

Para realizar el menú superior, utilicé el objeto **Toolbar** que proporciona **Android Studio.** Una vez creado el menú, comencé a darle estilo y a crear sus opciones.

- El proyecto entero se ve beneficiado por la realización de esta historia de usuario, ya que, será reutilizada para toda la aplicación siendo la única diferencia el título, ya que este dependerá de la pantalla en la que se encuentre el usuario.

```
<androidx.appcompat.widget.Toolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="?attr/colorPrimary"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.0"
    app:popupTheme="@style/UserToolbarStyle" />
```

Ilustración 32. TFT: Menú superior (xml)

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();
    if(id == R.id.logoutAction){
        SharedPreferences preferences = getSharedPreferences( name: "credenciales", Context.MODE_PRIVATE);
        String emailValueRead=preferences.getString( key: "emailUser", defValue: "");
        String passValueRead=preferences.getString( key: "passUser", defValue: "");

        SharedPreferences.Editor editor=preferences.edit();
        editor.putString("emailUser","");
        editor.putString("passUser", "");

        editor.commit();

        firebaseAuth.getInstance().signOut();
        Intent intent = new Intent(getApplication(), LoginActivity.class);
        startActivity(intent);
    }
    if(id == R.id.help){
        Intent intent = new Intent(getApplication(), HelpActivity.class);
        startActivity(intent);
    }

    return super.onOptionsItemSelected(item);
}
```

Ilustración 33. TFT: Opciones del Menú superior (parte lógica).

HU-21: Login

- **Maquetación de la vista**: Para el desarrollo de la parte visual de esta funcionalidad, se desarrolló una pantalla, que contiene dos cuadros de texto, uno para introducir el e-mail y otro para introducir la contraseña. A su vez, también existen dos botones, cuyas acciones son iniciar sesión y solicitar nueva contraseña.

- **Parte lógica (Controlador)**: Para realizar la lógica de inicio de sesión, se creó un objeto **FirebaseAuth** el cual nos permite utilizar la función **signInWithEmailAndPassword** para realizar el inicio de sesión mediante e-mail y contraseña, pero, cuando la aplicación se cerraba o se ejecutaba en segundo plano, automáticamente el usuario aparecía con la sesión cerrada, es por ello, que este problema se solventó creando un objeto del tipo **SharedPreferences** con el fin de guardar pared clave-valor, en este caso el e-mail y la contraseña, de manera que, si estos pares clave-valor no están vacíos, quiere decir que el usuario no ha cerrado sesión y se le muestra su sesión iniciada.

HU-25: Menú inferior

Para la realización del menú inferior, utilicé el objeto **BottomNavigationView** que proporciona el Android Studio. Una vez creado, comencé a crear las opciones (creando también los iconos que tendrían). Finalmente le di estilo a la barra inferior.

```
<com.google.android.material.bottomnavigation.BottomNavigationView
    android:id="@+id/bottomNav"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="1.0"
    app:itemBackground="@color/colorPrimary"
    app:itemTextColor="@color/white"
    app:menu="@menu/bottom_menu">

</com.google.android.material.bottomnavigation.BottomNavigationView>
```

Ilustración 34. TFT: Menú inferior (xml)

```
bottomNavigationView = findViewById(R.id.bottomNav);

bottomNavigationView.setOnNavigationItemReselectedListener((item) → {
        Fragment fr = null;
        switch(item.getItemId()){
            case R.id.homeFr:
                Intent intent = new Intent(getApplicationContext(), HomeActivity.class);
                startActivity(intent);
                break;

            case R.id.justifiesFr:
                Intent intent1 = new Intent(getApplicationContext(), JustifyListActivity.class);
                startActivity(intent1);
                break;

            case R.id.profileFr:
                Intent intent2 = new Intent(getApplicationContext(), ProfileActivity.class);
                startActivity(intent2);
                break;

            case R.id.timecardFr:
                Intent intent3 = new Intent(getApplicationContext(), TimeCardListActivity.class);
                startActivity(intent3);
                break;

        }
});
```

Ilustración 35. TFT: Menú inferior (parte lógica)

- El proyecto entero se ve beneficiado por la realización de esta historia de usuario, ya que, será reutilizada para toda la aplicación.

HU-23: Cerrar sesión

- **Maquetación de la vista:** Para la parte visual de esta funcionalidad, solo se introdujo en la barra superior (**Toolbar**) una opción que fuera *cerrar sesión*.

- **Parte lógica (Controlador)**: Para la realización de esta funcionalidad, utilicé el método **signOut**() que ofrece el objeto **FirebaseAuth** y además, sustituí el e-mail y la contraseña guardadas en el objeto **SharedPreferences** por dos objetos tipo **Strings** vacíos.

HU-01: Fichar entrada

- **Maquetación de la vista**: Para el desarrollo de esta funcionalidad, en primer lugar, se creó un layout con una serie de componentes, entre ellos dos **text view** (componente que almacena texto) para mostrar la hora a la que fichó el empleado y por otro lado tres botones, dos de ellos permiten al usuario realizar el fichaje, de manera que, al pulsarlos, aparece el cuadro de diálogo para introducir la huella dactilar.

Ilustración 37. TFT: Pantalla inicial

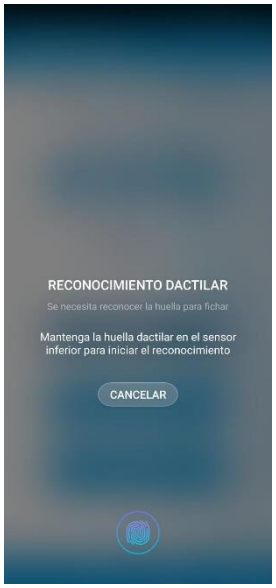

Ilustración 36. TFT: Diálogo de reconocimiento dactilar

- **Parte lógica (Controlador)**: Para el desarrollo de la parte lógica de la funcionalidad, en primer lugar, cree una variable tipo **LocalDate** para guardar la fecha del fichaje y una variable **LocalTime** para guardar la hora de entrada a la que ficha el empleado. Posteriormente cree dos variables de tipo **String**, una para guardar la hora a la que ficha el empleado convertida en **String**, y otra para guardar la ubicación en la que fichó el empleado.

  Además, también cree un objeto de tipo **DateFormatter** para cambiar el formato de la hora. Una vez realizados estos pasos iniciales, procedí a utilizar el dialogo de reconocimiento biométrico para realizar el fichaje, de manera que, una vez el empleado introduzca su huella, se guarda la hora a la que la introdujo como horario de entrada.

  Finalmente se introducen todos estos datos en la base de datos, dejando vacíos todos los datos relacionados con la justificación (ya que si el empleado ficha la entrada, significará que está trabajando y no tendrá efecto o el justificar la falta) y el horario de salida, (horario y ubicación) ya que estos dejarán de estar vacíos cuando se fiche la salida.

```
final BiometricPrompt biometricPrompt = new BiometricPrompt.Builder( context: this)
        .setTitle("RECONOCIMIENTO DACTILAR")
        .setSubtitle("Se necesita reconocer la huella para fichar")
        .setDescription("Mantenga la huella dactilar en el sensor inferior para iniciar el reconocimiento")
        .setNegativeButton( text: "Cancelar", executor, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

            }
        }).build();
```

Ilustración 38. Diálogo para reconocimiento dactilar (Fichero HomeActivity.java)

HU-02: Fichar salida

- **Maquetación de la vista** Para la realización de la parte visual de esta funcionalidad, se realizaron los mismos pasos que en la funcionalidad anterior, creación de un botón para fichar la salida y un **text view** que contenga la hora de salida.

Además, cree un nuevo **text view** para comunicar la duración de la jornada laboral (se puede observar la vista en la ilustración 37).

- **Parte lógica (Controlador)**: La parte lógica de esta funcionalidad es muy parecida a la anterior. La diferencia erradica en la introducción de todos los datos en la base de datos y no solo los datos del horario de entrada. Para ello, se ha creado una entidad denominada **TimeCard** la cual contiene toda la información del fichaje a guardar.
  En la ilustración 39, se muestra el constructor del objeto **TimeCard**. Para guardar un objeto de este tipo es necesario registrar la fecha, el horario de entrada y su localización, el horario de salida y su localización, el id del usuario, el motivo de la justificación de la falta (si lo hubiera), el número de horas trabajadas y el número de minutos trabajados (estos dos últimos destinados a mostrarlos en el cuadro de texto correspondiente a la duración de la jornada laboral).

```
public TimeCard(String day, String entryTime, String outTime, String userId, String entryLocation, String outLocation, String justify,
                Double hour, Double min) {
    this.date = day;
    this.entryLocation = entryLocation;
    this.outLocation = outLocation;
    this.entryTime = entryTime;
    this.outTime = outTime;
    this.userId = userId;
    this.justify = justify;
    this.hour=hour;
    this.min=min;
}
```

Ilustración 39. Entidad TimeCard.

HU-03: Fichaje geolocalizado

Esta funcionalidad consta de una única tarea en la parte lógica, la cual se corresponde con obtener la ubicación del empleado en el momento en el que realiza los fichajes de entrada y salida.

Para ello, en primer lugar, creé dos variables tipo **String** para guardar la localización de entrada y la localización de salida. Luego, construí una función mediante la cual se inicializa el servicio GPS del dispositivo, de manera que, en caso de que la funcionalidad GPS del dispositivo no esté activada, lanza una ventana emergente para activarlo y de estar activado sigue su flujo normal. Además, cree una clase denominada Localización que implementa la interfaz **LocationListener** mediante la cual se obtiene la latitud y longitud en la que se encuentra el usuario al momento de fichar (estos fichajes han de realizarse parados para hacer posible el reconocimiento de la ubicación).

```
public class Localizacion implements LocationListener {
    HomeActivity mainActivity;
    public HomeActivity getMainActivity() { return mainActivity; }
    public void setMainActivity(HomeActivity mainActivity) { this.mainActivity = mainActivity; }
    @Override
    public void onLocationChanged(Location loc) {
        // Este metodo se ejecuta cada vez que el GPS recibe nuevas coordenadas
        // debido a la deteccion de un cambio de ubicacion
        loc.getLatitude();
        loc.getLongitude();
        String sLatitud = String.valueOf(loc.getLatitude());
        String sLongitud = String.valueOf(loc.getLongitude());

        this.mainActivity.setLocation(loc);
    }
    @Override
    public void onProviderDisabled(String provider) {
        // Este metodo se ejecuta cuando el GPS es desactivado
        infoGPS.setText("GPS Desactivado");
    }
    @Override
    public void onProviderEnabled(String provider) {
        // Este metodo se ejecuta cuando el GPS es activado
        infoGPS.setText("GPS Activado");
    }
```

Ilustración 40. Clase Localización

Luego construí una función mediante la cual se obtiene la dirección a través de la latitud y longitud.

Finalmente cree un cuadro de diálogo, el cual, recordase al empleado que, para realizar los fichajes, es necesario que no se encuentre en movimiento, para hacer posible el reconocimiento de la ubicación. Este cuadro de diálogo aparece cuando el usuario muestra la actividad principal (HomeActivity.java).

Ilustración 41. TFT: Cuadro de diálogo recordatorio para obtener ubicación.

HU-07 Justificar falta

- **Maquetación de la vista:** Para la realización de la parte visual de esta funcionalidad se ha creado una nueva pantalla, a través de la cual se puede citar el motivo de la falta y un botón para seleccionar el justificante en formato PDF y también cuenta con otros dos botones, uno para realizar la justificación y otro para cancelarla. Además, al entrar en esta nueva actividad, se mostrará un cuadro de diálogo (ilustración 43) que informa al empleado que las justificaciones de las faltas solo se pueden realizar el día en el que se falta.

  Finalmente se mostrará un cuadro de diálogo que informe al empleado que su justificación se ha realizado con éxito (ilustración 43).

Ilustración 42. TFT: Justificar falta.

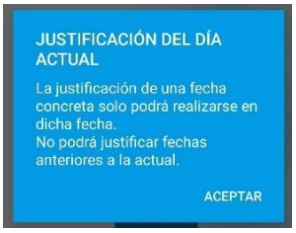

Ilustración 44. Cuadro de diálogo sobre la realización de la justificación

Ilustración 43. Cuadro de diálogo para una justificación realizada.

- **Parte lógica (Controlador):** Para la realización de la parte lógica de esta funcionalidad, en primer lugar, obtuve los elementos de la pantalla (botones y cuadros de texto) para poder trabajar con ellos y en el caso del cuadro de texto, obtener su contenido. Luego, cree un objeto **FirebaseAuth** para obtener el id del usuario en cuestión. Una vez realizado todos estos pasos previos, construí una función denominada ***selectFile***, a través de la cual, selecciono un fichero a subir y luego construí una función denominada ***UploadFile***, para subir el fichero *pdf* que adjunta el usuario. Todo este proceso lo hago haciendo uso del objeto **StorageReference** que proporciona **Firebase**, a través del cual podemos obtener la referencia del almacenamiento que previamente hemos configurado en **Firebase**.
  - Para guardar el archivo *pdf*, creo una **String** para el nombre del archivo, de manera que el nombre será la fecha concatenada con la hora a la que se subió

el archivo en **System.currentTimeMillis**, por ejemplo 2020-08-28_1598619124067.pdf

- Para más organización de los archivos *PDF* subidos al proyecto de **Firestore**, cree una carpeta en el **Storage Firebase** denominada ***/Uploads*** que contendrá todos los justificantes subidos.

```
private void selectFile(){
    Intent intent = new Intent();
    intent.setType("application/pdf");
    intent.setAction(Intent.ACTION_GET_CONTENT);
    startActivityForResult(intent, requestCode: 86);
}
```

Ilustración 45. TFT: Código se la función selectFile

```
private void uploadFile(Uri pdfUri) {
    progressDialog = new ProgressDialog( context: this);
    progressDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
    progressDialog.setTitle("Subiendo justificante...");
    progressDialog.setProgress(0);
    progressDialog.show();
    FirebaseUser userId = FirebaseAuth.getInstance().getCurrentUser();
    final String id = userId.getUid();
    final String fileName=""+today+"_"+System.currentTimeMillis()+".pdf";
    final String fileName1 = today+"_"+System.currentTimeMillis();
    StorageReference storageReference=storage.getReference();
    storageReference.child("Uploads").child(fileName).putFile(pdfUri).addOnSuccessListener(
            (OnSuccessListener) (taskSnapshot) -> {
                String url = taskSnapshot.getMetadata().getReference().getDownloadUrl().toString();
                usersDB.child("justificantes").child(id).child(fileName1).setValue(url).addOnCompleteListener((task) -> {
                    if(task.isSuccessful()){
                        Toast.makeText( context: AddJustifyActivity.this, text: "Justificación subida con éxito", Toast.LENGTH_SHORT).show();
                        Intent intent = new Intent(getApplication(), HomeActivity.class);
                        startActivity(intent);
                    }else{
                        Toast.makeText( context: AddJustifyActivity.this, text: "No se ha subido el archivo con éxtio:(", Toast.LENGTH_SHORT).show();
                    }
                });
            }
    ).addOnFailureListener((e) -> {
            Toast.makeText( context: AddJustifyActivity.this, text: "No se ha subido el archivo con éxtio:(", Toast.LENGTH_SHORT).show();
    }).addOnProgressListener((OnProgressListener) (taskSnapshot) -> {
            int currentProgress= (int) (100*taskSnapshot.getBytesTransferred()/taskSnapshot.getTotalByteCount());
            progressDialog.setProgress(currentProgress);
    });
}
```

Ilustración 46. TFT: Código de la función uploadFile

8.2.3. Sprint 2

ID	Nombre	Estimación
HU-04	**Ver fichajes**	**5 horas**
HU-06	**Buscar por fecha**	**5 horas**
HU-08	**Ver justificantes**	**5 horas**
HU-10	**Ver ayuda**	**4 horas**
HU-11	**Solicitar cambio de contraseña**	**4 horas**
Tiempo estimado		**23 horas**

Tabla 5. Pila de sprint 2

A continuación, se muestran detalladas las historias de usuario en el orden realizado:

HU-11: Solicitar cambio de contraseña

- **Maquetación de la vista:** Para esta funcionalidad, se desarrolló un nuevo layout el cual posee tres cajas de texto, para introducir el DNI, número de teléfono y e-mail, respectivamente y un botón para realizar la solicitud. Una vez el empleado ha solicitado el cambio de contraseña, aparece un cuadro de diálogo que le informa de ello.

  A continuación, se muestran imágenes sobre la vista de esta funcionalidad (incluyendo el cuadro de diálogo):

Ilustración 47. TFT: Pantalla para reestablecer contraseña.

Ilustración 48. TFT: Cuadro de diálogo de cambio de contraseña

Ilustración 49. TFT: Correo para reestablecer contraseña

- **Parte lógica (Controlador):** Para la parte lógica, cree un objeto **FirebaseAuth** mediante el cual, puede utilizarse la función **sendPasswordResetEmail** para realizar el envío del e-mail con el link para reestablecer la contraseña.

HU-10 Ver ayuda

Para la realización de esta funcionalidad en primer lugar, se introdujo en el Toolbar (o menú superior) una opción que llevase a esta funcionalidad. Luego, cree una nueva pantalla, la cual posee un objeto **ScrollView**, el cual nos permite disponer de una gran cantidad de componentes visuales que superan la cantidad de espacio del dispositivo, permitiendo al

usuario desplazar con el dedo la interfaz creada. Es por ello, que dentro del objeto anteriormente mencionado, se ha introducido un **LinearLayout** que es un layout que establece los componentes visuales uno junto al otro, ya sea horizontal o vertical. En este caso, decidí ponerlos de forma vertical, lo cual es lo apropiado en estos casos.

Dentro de este **LinearLayout** fui introduciendo **cardViews**, que son componentes que ofrece Android Studio, que permiten poner elementos en forma de tarjeta. Una vez creado estos **cardViews**, en su interior introduje cuadros de texto en los que poner las distintas opciones de ayuda.

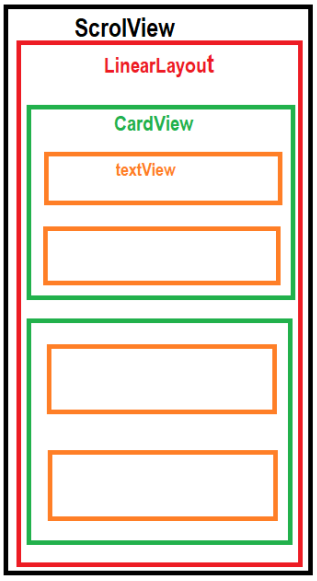

Ilustración 50. TFT: Distribución de pantalla de avuda.

Ilustración 51. TFT: Pantalla de ayuda

HU-04: Ver fichajes

- **Maquetación de la vista**: Para la parte visual de esta funcionalidad, cree un nuevo layout el cual contiene un cuadro de texto informativo, dos botones (para la siguiente funcionalidad) y un **RecyclerView** (que es una vista en forma de lista) en el que se mostrarán los **CardViews** con el contenido de los fichajes.
  El aspecto de los **CardViews** está definida en otro fichero denominado row_recycler.xml.

Ilustración 52. TFT: row_recyler.xml (cardView para fichaje)

En la ilustración 53 se puede observar cómo está distribuida la pantalla correspondiente a esta funcionalidad y también, como se han introducido los **CardViews** con los fichajes en cuestión.

Ilustración 53. TFT: Listado de fichajes

- **Parte lógica (Controlador)**: En primer lugar, cree un adaptador, mecanismo que hace de puente entre los datos y las vistas contenidas en un **ListView** o **RecyclerView**.
  El adaptador, es una clase encargada de obtener los componentes visuales del **CardView** creado anteriormente en el fichero row_recycler.xml y darle el valor correspondiente a los elementos visuales. Este adaptador tendrá un **ArrayList** que almacena objetos del tipo *TimeCard* (que será la lista de fichajes, el modelo), el cual se inicializará en su constructor.
  A su vez, en el fichero TimeCardListActivity.java (que es el controlador de la actividad), se crea un **ArrayList** que almacene *TimeCard* y un objeto adaptador (el mismo que el anteriormente mencionado), el cual será inicializado con el **ArrayList** anterior.
  Luego, se realiza una consulta a la base de datos, de manera que, vamos a obtener todos los fichajes que se correspondan con el usuario en cuestión (buscando por el id del usuario) y todas las coincidencias se irán almacenando en el **ArrayList**, de manera que, todos esos fichajes los tendrá almacenados el adaptador.

```
private void loadTimeCardList(){
    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
    String id = user.getUid();

    timeCardDB.getReference( path: "fichajes").child(id).addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            if(snapshot.exists()){
                list.clear();
                for (DataSnapshot data: snapshot.getChildren()){
                    TimeCard timeCard = data.getValue(TimeCard.class);
                    list.add(timeCard);
                    adapter.notifyDataSetChanged();
                }
                infoDate.setText("");
            }else{
                infoDate.setText("Aún no tienes fichajes registrados");
            }

        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });
}
```

Ilustración 54. TFT: Método que añade los fichajes al adaptador (TimeCardListActivity.java)

```
public class Adapter extends RecyclerView.Adapter<Adapter.TimeCardViewHolder>{
    List<TimeCard> list;

    public Adapter(List<TimeCard> list) { this.list = list; }

    @NonNull
    @Override
    public TimeCardViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.row_recycler,parent, attachToRoot: false);
        TimeCardViewHolder holder = new TimeCardViewHolder(v);
        return holder;
    }

    @Override
    public void onBindViewHolder(@NonNull TimeCardViewHolder holder, int position) {
        TimeCard timecard = list.get(position);
        if(timecard.getEntryTime().equals("") && timecard.getOutTime().equals("")){
            holder.justify.setText("Motivo: "+timecard.getJustify());
            holder.workhour.setText("Jornada laboral no existente");
            holder.date.setText(timecard.getDate());
            holder.entry.setText("Inicio: No registrado");
            holder.out.setText("Fin: No registrado");
            holder.entryLocation.setText("Sin ubicación");
            holder.outLocation.setText("Sin ubicación");
        }else{
            holder.workhour.setText("Jornada: "+timecard.getHour()+ " horas y "+ timecard.getMin()+" minutos.");
            holder.date.setText(timecard.getDate());
            holder.justify.setText("Sin justificación");
            holder.entry.setText("Inicio: "+timecard.getEntryTime());
            holder.out.setText("Fin: "+timecard.getOutTime());
            holder.entryLocation.setText(timecard.getEntryLocation());
            holder.outLocation.setText(timecard.getOutLocation());
        }
        holder.setOnClickListeners();

    }

    @Override
    public int getItemCount() { return list.size(); }
```

Ilustración 55. TFT: Adaptador para la lista de fichajes

HU-06: Buscar por fecha

- **Maquetación de la vista** Como bien se mencionaba en la historia de usuario anterior, se introdujeron dos cuadros de texto informativo para el usuario y dos botones para realizar la búsqueda por fecha, uno para abrir el calendario y seleccionar el día y otra para realizar la consulta.

Ilustración 58. Búsqueda de fichaje con falta justificada.

Ilustración 56. TFT: Calendario para buscar fecha de fichaje

Ilustración 57. Búsqueda de fichaje.

Se han adjuntado dos ilustraciones sobre el resultado de una búsqueda de un fichaje (ilustración 56 e ilustración 58). En la ilustración 56 se observa la tarjeta de un fichaje para el

cual se ha realizado una justificación, por ese motivo se muestra tanto en horario de entrada como de salida *No registrado.* En la ilustración 58 se observa la tarjeta de un fichaje que no tiene justificación, y por lo tanto tiene horario de entrada y de salida.

- **Parte lógica (Controlador)**: Para la parte lógica de esta funcionalidad, se añadieron los siguientes métodos al controlador de fichajes (*TimeCardListActivity.java*):
    - El método **showDatePickerDialog()** que muestra el calendario al usuario

```
private void showDatePickerDialog(){
    final Calendar cldr = Calendar.getInstance();
    int day = cldr.get(Calendar.DAY_OF_MONTH);
    int month = cldr.get(Calendar.MONTH);
    int year = cldr.get(Calendar.YEAR);
    String date="";
    // date picker dialog
    picker = new DatePickerDialog( context: TimeCardListActivity.this,R.style.DatePickerStyle,
            (view, year, monthOfYear, dayOfMonth) -> {
                    if(monthOfYear+1<10 && dayOfMonth<10){
                        infoDate.setText(year+"-"+"0"+(monthOfYear+1) +"-"+"0"+dayOfMonth);
                    }else if(monthOfYear+1>=10 && dayOfMonth<10){
                        infoDate.setText(year+"-"+(monthOfYear + 1) +"-"+"0"+dayOfMonth);
                    }else if(monthOfYear+1<10 && dayOfMonth>=10){
                        infoDate.setText(year+"-0"+(monthOfYear+1) +"-"+dayOfMonth);
                    }else{
                        infoDate.setText(year+"-"+ (monthOfYear + 1) +"-"+dayOfMonth);
                    }

            }, year, month, day);
    picker.show();
    dateToSearch=infoDate.getText().toString();
}
```

Ilustración 59. TFT: Método que muestra el calendario al usuario

    - El método **LoadTimeCardFilter()** al que se le pasa una fecha y realiza la búsqueda en la base de datos.

```
private void loadTimeCardFilter(final String date_search){
    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
    String id = user.getUid();

    timeCardDB.getReference( path: "fichajes").child(id).addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            if(snapshot.exists()){
                for (DataSnapshot data: snapshot.getChildren()){
                    TimeCard timeCard = data.getValue(TimeCard.class);
                    if(timeCard.getDate().equals(infoDate.getText())){
                        list.clear();
                        list.add(timeCard);
                        adapter.notifyDataSetChanged();
                        infoDate.setText("");
                    }
                }

                if(!snapshot.getChildren().toString().contains(infoDate.getText())){
                    infoDate.setText("No existen registros para la fecha seleccionada");
                    list.clear();
                    for (DataSnapshot data: snapshot.getChildren()){
                        TimeCard timeCard = data.getValue(TimeCard.class);
                        list.add(timeCard);
                        adapter.notifyDataSetChanged();
                    }
                }
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });
```

Ilustración 60. TFT: Método que realiza filtra la lista de fichajes

HU-08: Ver justificantes

- **Maquetación de la vista**: Al igual que en el listado de fichajes, en el listado de justificantes también se hace uso del objeto **RecyclerView**, el cual almacenará **CardViews**. El aspecto definido en *XML* de estos **CardViews** se encuentra en el fichero *row_recycler_justify.xml*.
  Como se puede observar en la siguiente ilustración, el **CardView** posee un **TextView** en el cual, se introducirá el nombre del fichero pdf que contiene el justificante.

Ilustración 61. TFT: row_recyler_justify.xml (cardView para justificantes)

En la ilustración 62 se puede observar la distribución de los elementos y la pantalla del listado de justificantes.

Ilustración 62. TFT: Listado de justificantes

- **Parte lógica (Controlador)**: Al igual que en la funcionalidad de ver fichajes, he tenido que crear un nuevo adaptador, mediante el cual rellenar los componentes visuales de los **CardViews** con su correspondiente información. El mecanismo es el mismo que para la funcionalidad anterior.

```
public class AdapterJustify extends RecyclerView.Adapter<AdapterJustify.ViewHolderJustify>{
    RecyclerView recyclerView;
    Context context;
    ArrayList<String> items;
    ArrayList<String> urls;

    public void update(String name, String url){
        items.add(name);
        urls.add(url);
        notifyDataSetChanged();
    }
    public AdapterJustify(RecyclerView recyclerView, Context context, ArrayList<String> items, ArrayList<String> urls) {
        this.recyclerView = recyclerView;
        this.context = context;
        this.items = items;
        this.urls = urls;
    }

    @NonNull
    @Override
    public ViewHolderJustify onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(context).inflate(R.layout.row_recycler_justify, parent,  attachToRoot: false);
        return new ViewHolderJustify(view);
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolderJustify holder, int position) {
        //Intialise the elements of individual items
        holder.nameOfFile.setText(items.get(position));
    }

    @Override
    public int getItemCount() { return items.size(); }
```

Ilustración 63. TFT: Adaptador para el listado de justificantes

```
    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
    String id = user.getUid();

    DatabaseReference justificantes = FirebaseDatabase.getInstance().getReference( path: "justificantes");
    justificantes.child(id).addChildEventListener(new ChildEventListener() {
        @Override
        public void onChildAdded(@NonNull DataSnapshot snapshot, @Nullable String previousChildName) {
            String fileName = snapshot.getKey();
            String url = snapshot.getValue(String.class);
            ((AdapterJustify)recyclerView.getAdapter()).update(fileName, url);
        }

        @Override
        public void onChildChanged(@NonNull DataSnapshot snapshot, @Nullable String previousChildName) {

        }

        @Override
        public void onChildRemoved(@NonNull DataSnapshot snapshot) {

        }

        @Override
        public void onChildMoved(@NonNull DataSnapshot snapshot, @Nullable String previousChildName) {

        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });
    recyclerView=findViewById(R.id.rvJustify);

    recyclerView.setLayoutManager(new LinearLayoutManager( context: JustifyListActivity.this));
    AdapterJustify adapter = new AdapterJustify(recyclerView,  context: JustifyListActivity.this,
            new ArrayList<String>(),new ArrayList<String>());
    recyclerView.setAdapter(adapter);
```

Ilustración 64. TFT: Método que añade los justificantes al adaptador

8.2.4. Sprint 3

ID	Nombre	Estimación
HU-05	**Ver detalles de fichajes**	**5 horas**
HU-09	**Ver justificante**	**4 horas**
HU-12	**Ver mis datos de perfil**	**4 horas**
HU-13	**Modificar mis datos de perfil**	**4 horas**
HU-14	**Ver ubicación**	**1 hora**
HU-21	**Visualizar contraseña**	**2 horas**
HU-26	**Ver información**	**1 hora**
Tiempo estimado		**21 horas**

Tabla 6. Pila de sprint 3

A continuación, se detallan las historias de usuario en el orden en el que fueron realizadas:

HU-05: Ver detalles de fichaje

- **Maquetación de la vista** Para la parte visual, se creó un nuevo layout, el cual contendrá dos **TextView**, para mostrar la fecha y el total de la jornada laboral y tres **CardViews** para mostrar los detalles de entrada, los detalles de salida y el motivo de la justificación si lo hubiera.

Ilustración 66. TFT: Detalles de fichaje (Falta justificada)

Ilustración 65. TFT: Detalles de fichaje (Falta no justificada)

A continuación, se muestran dos ilustraciones en las cuales se puede observar la distribución de los elementos visuales de esta funcionalidad:

En la ilustración 65 se puede observar cómo quedarían los componentes visuales si el empleado justificara la falta al trabajo, quedando ausentes los horarios de entrada y salida. En la ilustración 66 se puede observar un ejemplo de cómo se verían los componentes visuales en caso de que el empleado haya trabajado, apareciendo los detalles de entrada y salida.

- **Parte lógica (Controlador)**: Android permite que, al cambiar de actividad (funcionalidad o pantalla) se pueda enviar información (**Strings**, **enteros**, etc) entre actividades haciendo uso del método **putExtra()**, de esta manera, toda la información del fichaje que fue previamente seleccionado (pulsando sobre él) fue enviada a la nueva actividad haciendo uso de este método que ofrece Android.
  A continuación, en la ilustración 67, como puede observarse, el método **putExtra()** requiere, en primer lugar un nombre para que, posteriormente pueda identificarse la información y recuperarse como se muestra en la ilustración 68 (para recuperar la información se utiliza, en este caso, el método **getString()** que recupera el valor introduciendo él nombre que previamente le hemos dado) y en segundo lugar, se envía el dato, quedando de la siguiente manera:

  Intent.putExtra("nombre del atributo" , valor);

```
@Override
public void onClick(View v) {
    Intent intent = new Intent(context, TimeCardDetailActivity.class);

    intent.putExtra( name: "entryLocationDetail", entryLocation.getText());
    intent.putExtra( name: "justifyDetail", justify.getText());
    intent.putExtra( name: "outLocationDetail", outLocation.getText());
    intent.putExtra( name: "dateDetail",date.getText());
    intent.putExtra( name: "entryDetail",entry.getText());
    intent.putExtra( name: "outDetail",out.getText());
    intent.putExtra( name: "workHour",workhour.getText());
    context.startActivity(intent);
}
```

Ilustración 67. TFT: Ejemplo de uso del método putExtra()

```
workHourDetail=(TextView)findViewById(R.id.workHour);
dateDetail=(TextView)findViewById(R.id.dateDetail);
entryDetail=(TextView)findViewById(R.id.entryDetail);
justificationText=(TextView)findViewById(R.id.justificationText);
outDetail=(TextView)findViewById(R.id.outDetail);
entryLocationDetail=(TextView)findViewById(R.id.entryLocationDetail);
outLocationDetail=(TextView)findViewById(R.id.outLocationDetail);

Bundle extras = getIntent().getExtras();
String date="";
String out="";
String entry="";
String entryLocation="";
String outLocation="";
String justify="";
String workHour="";
if (extras!=null){
    workHour=extras.getString( key: "workHour");
    date = extras.getString( key: "dateDetail");
    out=extras.getString( key: "outDetail");
    entry=extras.getString( key: "entryDetail");
    entryLocation=extras.getString( key: "entryLocationDetail");
    outLocation=extras.getString( key: "outLocationDetail");
    justify=extras.getString( key: "justifyDetail");

}
workHourDetail.setText(workHour);
dateDetail.setText(date);
entryDetail.setText(entry);
justificationText.setText(justify);
entryLocationDetail.setText("Ubicación: "+entryLocation);
outDetail.setText(out);
outLocationDetail.setText("Ubicación: "+outLocation);
```

Ilustración 68. TFT: Ejemplo de cómo se obtiene información introducida con putExtra()

HU-12: Ver mis datos de perfil

- **Maquetación de la vista**: Se desarrolló una pantalla que reflejara los datos del empleado. Además, también se introdujo un botón para modificar el perfil (historia de usuario siguiente), con el fin de adelantar trabajo futuro.
  La distribución de los Layouts con la información que almacenarán se muestra a continuación:

Ilustración 69. TFT: Distribución de los componentes visuales para pantalla de perfil

- **Parte lógica (Controlador):** Para esta parte, en primer lugar, obtuve los componentes visuales de y luego cree un objeto **FirebaseAuth** con el cual obtener el id del usuario en cuestión y una vez averiguado su id, cree un método que realiza una consulta a la base de datos a través de ese id.
  Una vez realizada la consulta, con el método **setText()** que tienen los componentes visuales de Android, comencé a meter la información obtenida de la base de datos en sus correspondientes vistas.

```
private void loadProfile(){
    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
    String id = user.getUid();

    Query q = FirebaseDatabase.getInstance().getReference( path: "usuarios").orderByChild("userid").equalTo(id);
    q.addChildEventListener(new ChildEventListener() {
        @Override
        public void onChildAdded(@NonNull DataSnapshot snapshot, @Nullable String previousChildName) {
            String name = snapshot.child("nombre").getValue().toString();
            String surname = snapshot.child("apellidos").getValue().toString();
            String email = snapshot.child("email").getValue().toString();
            String dni = snapshot.child("dni").getValue().toString();
            String company = snapshot.child("empresa").getValue().toString();
            String city = snapshot.child("poblacion").getValue().toString();
            String phone = snapshot.child("telefono").getValue().toString();
            nameText.setText(name);
            nameTitle.setText(name+" "+surname);
            surnameText.setText(surname);
            emailText.setText(email);
            dniText.setText(dni);
            companyText.setText(company);
            cityText.setText(city);
            phoneText.setText(phone);
        }

        @Override
        public void onChildChanged(@NonNull DataSnapshot snapshot, @Nullable String previousChildName) {}

        @Override
        public void onChildRemoved(@NonNull DataSnapshot snapshot) {}

        @Override
        public void onChildMoved(@NonNull DataSnapshot snapshot, @Nullable String previousChildName) {}

        @Override
        public void onCancelled(@NonNull DatabaseError error) {}
    });
```

Ilustración 70. TFT: Método que realiza una consulta a la BD para obtener la información del usuario

HU-13: Modificar perfil

- **Maquetación de la vista:** Se desarrolló una pantalla que posee cuatro **EditText** para modificar e-mail, número de teléfono y contraseña (con confirmación) y dos botones uno para realizar el cambio y otro para cancelar.
  Además, se introdujo un icono a la derecha de los **EditTexts** de la contraseña para la posterior funcionalidad a realizar (Historia de usuario visualizar contraseña).

Ilustración 71. TFT: Editar perfil

- **Parte lógica (Controlador)**: En primer lugar, cree un método denominado **loadProfile()** que cargada los datos del usuario en los **EditText** para que el usuario, pudiera visualizar sus propios datos antes de cambiarlos.
  Luego, le di acciones, por un lado, al botón de editar, de manera que, en ese momento captura la información que el usuario cambio en los **EditTexts** para modificar el registro en la base de datos y por otro lado al botón de cancelar, el cual simplemente regresa a la pantalla anterior.

```
private void loadProfile(){
    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
    String id = user.getUid();

    Query q = FirebaseDatabase.getInstance().getReference( path: "usuarios").orderByChild("userid").equalTo(id);
    q.addChildEventListener(new ChildEventListener() {
        @Override
        public void onChildAdded(@NonNull DataSnapshot snapshot, @Nullable String previousChildName) {
            name = snapshot.child("nombre").getValue().toString();
            surname = snapshot.child("apellidos").getValue().toString();
            email = snapshot.child("email").getValue().toString();
            dni = snapshot.child("dni").getValue().toString();
            company = snapshot.child("empresa").getValue().toString();
            city = snapshot.child("poblacion").getValue().toString();
            phone = snapshot.child("telefono").getValue().toString();
           password = snapshot.child("password").getValue().toString();
           emailEdit.setText(email);
           telEdit.setText(phone);
           pass.setText(password);
           pass2.setText(password);
        }

        @Override
        public void onChildChanged(@NonNull DataSnapshot snapshot, @Nullable String previousChildName) {}

        @Override
        public void onChildRemoved(@NonNull DataSnapshot snapshot) {

        }

        @Override
        public void onChildMoved(@NonNull DataSnapshot snapshot, @Nullable String previousChildName) {}


        @Override
        public void onCancelled(@NonNull DatabaseError error) {}
    });
}
```

Ilustración 72. TFT: Método que carga la información del usuario para modificarlos

```
confirm.setOnClickListener(new View.OnClickListener() {
    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
    String id = user.getUid();
    @Override
    public void onClick(View v) {
        Map<String, Object> user = new HashMap<>();
        user.put("email", emailEdit.getText().toString());
        user.put("password", pass.getText().toString());
        user.put("telefono", telEdit.getText().toString());
        user.put("apellidos", surname);
        user.put("nombre", name);
        user.put("empresa", company);
        user.put("poblacion", city);
        user.put("dni", dni);
        user.put("userid", id);


        usersDB.setValue(user);
        Intent intent = new Intent(getApplication(), ProfileActivity.class);
        startActivity(intent);
    }
});
cancel.setOnClickListener((v) -> {
        Intent intent = new Intent(getApplicationContext(), ProfileActivity.class);
        startActivity(intent);
});
```

Ilustración 73. TFT: Acciones de los botones de la pantalla de editar perfil

HU-09: Ver justificante

Esta funcionalidad consiste en visualizar el pdf que el usuario ha pulsado. Para ello se utiliza el programa por defecto lector de pdf que tenga el usuario.

Para la realización de esta funcionalidad, se introdujo en el adaptador de justificantes, en primer lugar, que los elementos que se muestran en el **RecyclerView** puedan ser pulsados. Una vez pulsado, se obtiene el nombre del fichero y se accede a él a través del objeto **StorageReference**, de manera que, para acceder al mismo, se referencia la ubicación en la que se encuentra dentro del almacenamiento de Firebase.

A continuación, se muestra una ilustración, en la que se puede observar cómo se accede al fichero a través del objeto anteriormente mencionado:

- Los archivos se encuentran dentro de la carpeta ***/Uploads***

```
public class ViewHolderJustify extends RecyclerView.ViewHolder{
    TextView nameOfFile;
    public ViewHolderJustify(@NonNull final View itemView) {
        super(itemView);
        nameOfFile=(TextView)itemView.findViewById(R.id.nameOfFileJustify);
        itemView.setOnClickListener((v) -> {
                final int position = recyclerView.getChildLayoutPosition(v);
                StorageReference filepath = FirebaseStorage.getInstance().getReference().child("Uploads").child(items.get(position));
                StorageReference storageReference = FirebaseStorage.getInstance().getReference();
                String name="Uploads/"+items.get(position)+".pdf";
                StorageReference path = storageReference.child(name);
                path.getDownloadUrl().addOnSuccessListener((OnSuccessListener) (uri) -> {
                        Intent intent = new Intent(Intent.ACTION_VIEW, uri);
                        intent.setDataAndType(uri, type: "application/pdf");
                        context.startActivity(intent);
                });
        });
    }
}
```

Ilustración 74. TFT: Apertura de un justificante PDF

HU-21: Visualizar contraseña

- **Maquetación de la vista:** En la pantalla de editar perfil, se añadió un icono al lado de los EditTexts de las contraseñas con el fin de que el usuario pulse y visualice su contraseña y compruebe si ambas contraseñas coinciden.
-
- **Parte lógica (Controlador):** Añadí acciones a los iconos añadidos al lado de los **EditTexts** de las contraseñas, de manera que, al pulsarlos, ejecutan un método establecido en el controlador de la actividad. Al tener dos **EditText** para las contraseñas, tuve que realizar dos métodos idénticos (cuyos nombres son ***ShowPassword*** y ***ShowPassword2***), pero no reutilizables, ya que ambos operan sobre componentes visuales diferentes.

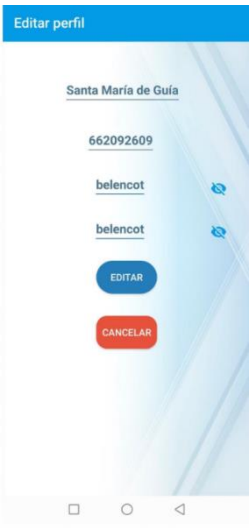

Ilustración 75. TFT: Visualizar contraseña

```
public void ShowPassword(View view) {
    if(view.getId()==R.id.iv1){

        if(pass.getTransformationMethod().equals(PasswordTransformationMethod.getInstance())){
            ((ImageView)(view)).setImageResource(R.drawable.hide_password);
            //Show Password
            pass.setTransformationMethod(HideReturnsTransformationMethod.getInstance());
        }
        else{
            ((ImageView)(view)).setImageResource(R.drawable.ic_visibility_black_24dp);
            //Hide Password
            pass.setTransformationMethod(PasswordTransformationMethod.getInstance());

        }
    }
}
```

Ilustración 76. TFT: Método para enseñar y ocultar la contraseña.

HU-14: Ver ubicación

- **Maquetación de la vista**: Para la realización de la parte visual de esta funcionalidad, simplemente se introdujeron dos botones en la actividad “Ver detalles de fichaje” con los que visualizar la ubicación del fichaje de salida y de entrada en la aplicación **Google Maps.**
- **Parte lógica (Controlador):** Para la realización de la parte lógica, se obtuvieron las direcciones en forma de **String** y se inició una nueva actividad del tipo ***ACTION_VIEW*** a la que se le pasa la dirección y la muestra en **Google Maps**.

Ilustración 77. TFT: Ubicacion en Maps

Ilustración 78. TFT: Ver ubicación (Botones)

```
entryGlobal=entryLocation;
outGlobal=outLocation;
if(entryGlobal.equals("Sin ubicación")){
    locationEntry.setEnabled(false);
}
if(outGlobal.equals("Sin ubicación")){
    locationOut.setEnabled(false);
}
locationEntry.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String map = "http://maps.google.com/maps?q=" +
                entryGlobal;

        Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse(map));
        startActivity(i);
    }
});
locationOut.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String map = "http://maps.google.com/maps?q=" +
                outGlobal;

        Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse(map));
        startActivity(i);
    }
});
```

Ilustración 79. TFT: Visualización de las ubicaciones (código)

8.3. Validación

Tenía previsto realizar las validaciones a gran escala, a través de usuarios relacionados con el ámbito del proyecto (empleados que trabajan fuera del centro de trabajo, como, por ejemplo, repartidores o comerciales y empresarios).

La idea era, contactar con empresas o empleados de estas características y concertar una cita con ellos. En la cita, el objetivo erradicaba en proporcionarles la aplicación y que ellos mismos la probaran y dieran su opinión al respecto, pero, debido a la situación y a las circunstancias que estamos viviendo con el COVID-19 y a las medidas impuestas por las autoridades sanitarias, se me ha hecho imposible realizar este tipo de validaciones. Es por ello, que he optado, por realizar validaciones a menor escala, proporcionarle la aplicación a personas cercanas y familiares para que la probaran y me dieran su opinión.

9. Resultados, conclusiones y trabajo futuro

9.1. Resultados

Tras finalizar el desarrollo del prototipo funcional, puedo afirmar que se han cumplido con todos los objetivos fijados al principio.

Esta aplicación permite realizar las acciones básicas para cumplimentar la nueva ley, de manera que, un empleado podrá:

- Iniciar sesión a través de e-mail y contraseña.
- Fichar la entrada y automáticamente se guardará la ubicación en la que se encuentra.
- Fichar la salida y al igual que en la funcionalidad anterior, se guardará automáticamente la ubicación en la que se encuentra.
- Justificar una falta, introduciendo el motivo y adjuntando la justificación en PDF.
- Ver sus datos de perfil.
- Editar sus datos de perfil (E-mail, teléfono y contraseña, ya que son los datos más relevantes sobre el usuario).
- Visualizar el listado de fichajes realizados.
- Ver los detalles de un fichaje en concreto, pulsando en él.
- Buscar el fichaje por fecha concreta en el calendario.
- Visualizar la ubicación del fichaje seleccionado.
- Visualizar el listado de justificantes.
- Ver el contenido de un justificante en concreto, pulsando en él.
- Visualizar la ayuda.
- Cerrar sesión.

Además, un usuario visitante o no autenticado podrá visualizar la información sobre la aplicación.

9.2. Conclusiones

Una vez terminado el proyecto, puedo decir que mi experiencia ha sido totalmente satisfactoria. En la realización de este proyecto he puesto en práctica los conocimientos adquiridos a lo largo del grado, además, ha sido todo un reto para mí, ya que es la primera vez que desarrollo una aplicación móvil y utilizo servicios de Firebase, como por ejemplo bases de datos *NOSQL*, haciéndolo sin ningún tipo de conocimiento previo y por ello, me ha llevado el doble de tiempo de realización.

También he de decir que, como en casi todo proyecto, durante el desarrollo han surgido algunos contratiempos al realizar algunas historias de usuario, (ya que han sido más complicadas de realizar de lo que esperaba), quedando el tiempo de realización por encima del estimado, pero eso no ha entorpecido la finalización de las funcionalidades, porque con muchas ganas y entusiasmo he conseguido solventar los problemas que han ido apareciendo, haciendo que me entrenara más y obtuviese más experiencia.

En definitiva, la realización de este TFT, me ha proporcionado conocimientos totalmente nuevos en desarrollo software y sobre todo en desarrollo de aplicaciones Android, que a día de hoy tienen una gran importancia en el mercado.

9.3. Trabajo futuro

En este TFT, se ha realizado un prototipo funcional, desarrollándose todos los requisitos y funcionalidades preestablecidos al principio del mismo. No obstante, si se pretendiera lanzar este proyecto al mercado, habría que estudiar posibles nuevas funcionalidades y/o mejoras para incorporar en un futuro antes de su lanzamiento, como por ejemplo puede ser, realizar una aplicación de escritorio o web a través de la cual, los administradores de las empresas que contraten el servicio puedan realizar acciones como generar informes y añadir, modificar, consultar y eliminar empleados. También se podría mejorar la aplicación para los empleados, pudiendo, a través de la aplicación, mostrar el estado de las reglas de fichaje (por ejemplo, que la aplicación muestre avisos al empleado si no ha fichado durante el límite propuesto o si se ha pasado del tiempo establecido como jornada laboral)

10. Bibliografía

[1] rrhhdigital, «Nuevo Decreto-Ley de control horario,» [En línea]. Disponible en: http://www.rrhhdigital.com/secciones/legal/136519/Nuevo-Decreto-Ley-de-Control-Horario-Que-dice-la-nueva-normativa-y-a-que-multas-se-enfrentan-las-empresas-que-la-incumplan?target=_self.

[2] Wikipedia, «Reglamento general de protección de datos,» [En línea]. Disponible en: https://es.wikipedia.org/wiki/Reglamento_General_de_Protecci%C3%B3n_de_Datos

[3] Wikipedia, «Ley orgánica de protección de datos personales y garantía de derechos digitales,» [En línea]. Disponible en: https://es.wikipedia.org/wiki/Ley_Org%C3%A1nica_de_Protecci%C3%B3n_de_Datos_Personales_y_garant%C3%ADa_de_los_derechos_digitales

[4] BOE, «Real Decreto-Ley 8/2019. Medidas urgentes de protección social y de lucha contra la precariedad laboral en la jornada de trabajo,» [En línea]. Disponible en: https://www.boe.es/eli/es/rdl/2019/03/08/8

[5] Wikipedia, «Android Studio,» [En línea]. Disponible en: https://www.google.es/url?sa=i&url=https%3A%2F%2Fwww.linuxadictos.com%2Fya-fue-liberada-la-beta-de-android-studio-3-5-y-estos-son-sus-cambios.html&psig=AOvVaw3MoXp56yEVLg0h_ygtUMs_&ust=1599736453763000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCPiKi-z42-sCFQA

[6] Wikipedia, «StarUML,» [En línea]. Disponible en: https://en.wikipedia.org/wiki/StarUML.

[7] Wikipedia, «Google Chrome,» [En línea]. Disponible en: es.wikipedia.org/wiki/Google_Chrome.

[8] Wikipedia, «Microsoft Word,» [En línea]. Disponible en: https://es.wikipedia.org/wiki/Microsoft_Word.

[9] digital55, «¿Qué es Firebase?,» [En línea]. Disponible en: https://www.digital55.com/desarrollo-tecnologia/que-es-firebase-funcionalidades-ventajas-conclusiones/.

[10] Cice, «Mockplus: diseño de prototipos interfaces,» [En línea]. Disponible en: https://www.cice.es/noticia/mockplus-diseno-prototipos-interfaces-ux-ui/.

[11] Trello, [En línea]. Disponible en: https://trello.com/.

[12] Wikipedia, «GitHub,» [En línea]. Disponible en: https://es.wikipedia.org/wiki/GitHub.

[13] Logomaster, [En línea]. Disponible en: https://logomaster.ai/.

[14] Firebase, «Realtime Database Firebase,» [En línea]. Disponible en: https://firebase.google.com/docs/database?hl=es.

[15] Firebase, «Firebase Authentication,» [En línea]. Disponible en: https://firebase.google.com/docs/auth?hl=es

[16] Firebase, «Firebase Storage,» [En línea]. Disponible en: https://firebase.google.com/docs/storage?hl=es.

[17] «What is Java?,» [En línea]. Disponible en: https://www.java.com/es/download/faq/whatis_java.xml.

[18] Rockcontent, «¿Qué es XML?,» [En línea]. Disponible en: https://rockcontent.com/es/blog/que-es-xml/.

[19] Apd, «¿Qué es Scrum?,» [En línea]. Disponible en: www.apd.es/metodologia-scrum-que-es/.

[20] RichardGracia, «Scrums para startups,» [En línea]. Disponible en: https://richardgracia.com/scrum-para-startups/.

[21] Control laboral, [En línea]. Disponible en: www.controllaboral.es/.

[22] Bixpe, [En línea]. Disponible en: www.bixpe.com/app-de-control-horario/.

[23] Intratime, [En línea]. Disponible en: www.intratime.es/.

[24] Ficha Work, [En línea]. Disponible en: www.ficha.work/.

[25] Sesame time, [En línea]. Disponible en: www.sesametime.com/control-horario/.

[26] Wikipedia, «Lienzo de modelo de negocio,» [En línea]. Disponible en: https://es.wikipedia.org/wiki/Lienzo_de_modelo_de_negocio#:~:text=Business%20Model%20Canvas%2C%20traducido%20como,o%20documentar%20los%20ya%20existentes.&text=El%20modelo%20de%20negocio%20del%20lienzo%20consta%20de%209%20piezas,representaci%C3%B3n%20gr%C3.

[27] cecarm, «¿Qué es Lean Canvas?,» [En línea]. Disponible en: https://www.cecarm.com/emprendedor/estrategia/consultas-y-faqs/que-es-el-lean-canvas-3801#:~:text=El%20Lean%20Canvas%20es%20una,Definici%C3%B3n%20de%20nuestros%20clientes%20objetivos&text=Definir%20las%20v%C3%ADas%20de%20ingresos%20que%20podr%C3%A1n%20uti.

[28] Google sites, «Técnicas para identificar requisitos funcionales y no funcionales,» [En línea]. Disponible en: https://sites.google.com/site/metodologiareq/capitulo-ii/tecnicas-para-identificar-requisitos-funcionales-y-no-funcionales.

[29] Wikipedia, «Interfaz de usuario,» [En línea]. Disponible en: https://es.wikipedia.org/wiki/Interfaz_de_usuario#Seg%C3%BAn_la_forma_de_interactuar_del_usuario.

[30] Neodoo, «Diseño ui-ux y prototipado de aplicaciones,» [En línea]. Disponible en: https://blog.neodoo.es/2020/01/28/diseno-ui-ux-y-prototipado-de-aplicaciones/#:~:text=1.-,Bocetos%20o%20Wireframes,la%20jerarqu%C3%ADa%20de%20la%20informaci%C3%B3n.

[31] Google , «explicación MVC,» [En línea]. Disponible en: https://www.google.com/url?sa=i&url=https%3A%2F%2Fikastaroak.ulhi.net%2Fedu%2Fes%2FDAW%2FDWES%2FDWES07%2Fes_DAW_DWES07_Contenidos%2Fwebsite_123_arquitectura_mvc.html&psig=AOvVaw2AziiViM05PbqRS-GoCiqf&ust=1600196820129000&source=images&cd=vfe&ved=0CAIQjRxq.

[32] EcuRed, «Diagrama de clases,» [En línea]. Disponible en: https://www.ecured.cu/Diagrama_de_Clase#Diagrama_de_Clases.

[33] Wikipedia, «Inyección de dependencias,» [En línea]. Disponible en: https://es.wikipedia.org/wiki/Inyecci%C3%B3n_de_dependencias.

[34] CmsPhoto, «Targeting women (colors of apps),» [En línea]. Disponible en: https://cmsphoto.ww-cdn.com/superstatic/81328/art/grande/6608783-9969508.jpg?v=1399640119.

[35] CmsPhoto, «Targetin Men (colors of apps),» [En línea]. Disponible en: https://cmsphoto.ww-cdn.com/superstatic/81328/art/grande/6608783-9969520.jpg?v=1399640200.

Anexos

Anexo nº 1. Pila de producto (Product Backlog)

- **Significado de la nomenclatura:**
  - HT: Historia técnica.
  - HU: Historia de usuario.

- **Significado de los colores:**
  - PRIORIDAD ALTA.
  - PRIORIDAD MEDIA.
  - PRIORIDAD BAJA.

ID	Nombre	Descripción	Prioridad	Estimación	Duración real
HT-1	Estudio de las tecnologías (Android y firebase)	Yo como desarrollador deseo aprender a programar en Android y firebase para implementar el proyecto	Alta	80 horas	85 horas
HT-2	Creación de tablón Trello	Yo como desarrollador deseo crear un tablón Trello para organizar las tareas	Baja	35 minutos	15 minutos
HT-3	Creación de repositorio GitHub	Yo como desarrollador deseo crear un repositorio GitHub para almacenar el proyecto	Alta	5 minutos	5 minutos
HT-4	Instalación y configuración del entorno de trabajo	Yo como desarrollador deseo instalar y configurar el programa Android Studio para realizar el desarrollo del proyecto	Alta	1 hora	2 horas
HT-5	Configuraciones Firebase previas al proyecto	Yo como desarrollador Deseo realizar las configuraciones de Firebase	Alta	15 horas	20 horas

		Para poder utilizar esta herramienta en el proyecto			
HU-01	Fichar entrada	Yo como empleado deseo poder fichar la entrada para guardar el registro	Alta	15 horas	16 horas
HU-02	Fichar salida	Yo como empleado deseo poder fichar la salida para guardar el registro	Alta	8 horas	8 horas
HU-03	Fichaje geolocalizado	Yo como empleado deseo fichar con geolocalización para registrar mi ubicación	Alta	10 horas	11 horas
HU-04	Ver fichajes	Yo como empleado deseo poder visualizar mis fichajes para realizar comprobaciones	Media	5 horas	5 horas
HU-05	Ver detalles de fichaje	Yo como empleado deseo poder ver los detalles del fichaje para visualizar los datos correspondientes	Media	5 horas	5 horas
HU-06	Buscar por fecha	Yo como empleado deseo poder buscar por fecha para comprobar el fichaje	Media	5 horas	5 horas y 30 minutos
HU-07	Justificar falta	Yo como empleado deseo poder justificar una falta para comunicar el motivo	Alta	7 horas	8 horas
HU-08	Ver justificantes	Yo como empleado deseo poder visualizar el listado de justificantes para ver su contenido	Media	5 horas	6 horas

HU-09	**Ver justificante**	**Yo como empleado deseo poder visualizar el justificante seleccionado para ver su contenido**	**Baja**	**4 horas**	**7 horas**
HU-10	**Ver ayuda**	**Yo como empleado deseo poder visualizar la ayuda de la aplicación para solventar dudas acerca de su utilización**	**Alta**	**4 horas**	**4 horas**
HU-11	**Solicitar cambio de contraseña**	**Yo como empleado deseo poder solicitar cambio de contraseña para poder acceder a la plataforma**	**Alta**	**4 horas**	**5 horas**
HU-12	**Ver mis datos de perfil**	**Yo como empleado deseo poder ver mi perfil para visualizar mis datos**	**Media**	**4 horas**	**5 horas**
HU-13	**Modificar mis datos de perfil**	**Yo como empleado deseo poder modificar mi perfil para actualizar mis datos**	**Media**	**4 horas**	**9 horas**
HU-14	**Ver ubicación**	**Yo como empleado deseo poder ver mi ubicación para comprobaciones**	**Baja**	**1 hora**	**1 hora**
HU-21	**Visualizar contraseña**	**Yo como usuario deseo poder visualizar mi contraseña para verificarla**	**Baja**	**2 horas**	**2 horas**
HU-22	**Login**	**Yo como usuario deseo poder iniciar en Clock-in On Time para poder realizar gestiones**	**Alta**	**3 horas**	**6 horas**
HU-23	**Cerrar sesión**	**Yo como usuario deseo poder cerrar sesión para dejar de**	**Alta**	**1 hora**	**1 hora**

		estar identificado			
HU-24	**Menú superior**	**Yo como usuario deseo poder visualizar el menú superior para acceder a las opciones de ver ayuda y cerrar sesión**	**Alta**	**1 horas**	**1 horas**
HU-25	**Menú inferior**	**Yo como usuario deseo poder visualizar el menú inferior para poder navegar por la aplicación**	**Alta**	**3 horas**	**3 horas**
HU-26	**Ver información**	**Yo como visitante deseo poder visualizar la información de la aplicación para conocer su objetivo**	**Baja**	**1 hora**	**1 hora y 30 minutos**
TOTAL DE HORAS				**181 horas y 40 minutos**	**217 horas y 20 minutos**

Tabla 7. Pila de producto (Product Backlog).