Sistema para el control de horario laboral y movimientos utilizando balizas de proximidad

Moisés Coda Cabeza de Vaca
Máster universitario de Desarrollo de aplicaciones para dispositivos móviles
Desarrollo de aplicaciones para dispositivos Android

Francesc D'Assís Giralt Queralt
Carles Garrigues Olivella

Enero 2020


Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-CompartirIgual 3.0 España de Creative Commons

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Sistema para el control de horario laboral y movimientos utilizando balizas de proximidad*
Nombre del autor:	*Moisés Coda Cabeza de Vaca*
Nombre del consultor/a:	*Francesc D'Assís Giralt Queralt*
Nombre del PRA:	*Carles Garrigues Olivella*
Fecha de entrega (mm/aaaa):	01/2020
Titulación:	*Máster universitario en Desarrollo de aplicaciones para dispositivos móviles*
Área del Trabajo Final:	*TFM*
Idioma del trabajo:	*Castellano*
Palabras clave	*Android, Control de presencia, Bluetooth*

Resumen del Trabajo:

Actualmente, con el cambio de legislación laboral, se ha hecho indispensable en las empresas el uso de aplicaciones que permitan registrar los accesos y el horario laboral de los trabajadores. Uno de los principales problemas que encuentran las empresas es la falta de precisión y en muchos casos la falta de colaboración de todas las partes implicadas.

Con este trabajo se presenta un sistema que permite registrar las entradas y salidas de trabajadores de forma precisa utilizando balizas colocadas estratégicamente. En este esquema, el trabajador simplemente tendría que confirmar las entradas o indicar el tipo de salida. De esta manera, el trabajador no tiene que tomar la iniciativa a la hora de llevar un registro.

Finalmente, con la información recopilada el sistema genera informes que sirven como documento oficial y además, permite generar estadísticas sobre como distribuyen los trabadores la jornada laboral.

Abstract:

Currently, with the last changes in labor legislation, has become necessary the use of applications that allow companies to register the workers access and working hours. One of the most common problems to face with this kind of applications is the lack of precission and in many cases, the lack of collaboration of the people involved.

This work presents a system that allows companies to register the time of entrance and exit of workers using strategically placed beacons. With this approach, the worker would simply have to confirm his entries or select the reason for an exit. Then, worker does not have to take the initiative for register

his movements.

Finally, with the registered information, system builds reports that serves as an official document and also can offer statistics on how people spent the working time.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Con los últimos cambios realizados en la legislación laboral, se ha hecho imprescindible dentro de las empresas el uso de aplicaciones que permitan registrar el horario de entrada y salida de los trabajadores.

En muchos casos, los responsables de recursos humanos se encuentran con el problema de la falta de rigurosidad de estos mecanismos o directamente, la falta de colaboración de los trabajadores.

Se hace necesario pues, plantear alternativas que permitan automatizar lo máximo posible este tipo de rutinas. Con la normalización de la tecnología móvil y, sobre todo, el auge en los últimos años del IoT (Internet of Things), se abren nuevas posibilidades a explorar a la hora de plantear soluciones a problemas cotidianos.

A continuación, se presentará un breve análisis de las alternativas comerciales existentes para gestionar el control horario en empresas y analizaremos sus defectos y virtudes.

1.1.1 Controllaboral [1]

Esta aplicación representa a un primer grupo de aplicaciones que permiten registrar las entradas y salidas de los trabajadores basadas únicamente en el dato aportado por el trabajador y un horario laboral preestablecido.

Este tipo de aplicaciones adolecen de falta de precisión y flexibilidad ante todo tipo de eventualidades: salidas de trabajo, horario flexible, etc.

1.1.2 Timenet [2] / Trackpeople [3]

Estas aplicaciones representan un conjunto de aplicaciones que utilizan cualquier dispositivo fijo o móvil (PCs, tablets o móviles) para que el trabajador registre la hora de entrada y salida.

El problema de este tipo de aplicaciones es que dependen de que el trabajador recuerde marcar las entradas y salidas en el momento de hacerlo. En caso contrario se lanzan alertas que pueden dar lugar a falsas impuntualidades o desajustes en los informes de horas.

Este tipo de aplicaciones también incluyen la posibilidad de utilizar el geoposicionamiento para determinar si el trabajador ha realizado el marcaje dentro de las instalaciones o incluso, para realizar un seguimiento de los movimientos del trabajador fuera de la empresa.

Este tipo de funcionalidades ha dado lugar en los últimos años a problemas legales en las empresas que obligan a los trabajadores a ser localizados en todo momento. De hecho, sentencias recientes ratifican el hecho de que un empleado no puede ser obligado a usar su dispositivo móvil particular para controlar su posición al entender que supone una violación de la privacidad.

1.1.3 Programa de control de presencia [4]

Esta aplicación comercial utiliza el mismo sistema de marcado que el grupo anterior. La razón por la que se ha tenido en cuenta es porque incluye un par de aspectos interesantes para nuestra aplicación: la posibilidad de establecer múltiples puestos de trabajo y la capacidad de crear diferentes códigos de parada de jornada: comida, descanso, etc.

1.1.4 Timepro [5]

Se trata de una de las aplicaciones más completas que se ha encontrado en el mercado. Utiliza balizas de proximidad y seguimiento de la ubicación para determinar las entradas y salidas de los trabajadores.

Por contra, no permite adaptar los diferentes tipos de salida en función de las necesidades de la empresa y, además, el hardware a instalar (balizas) va necesariamente incluido en el contrato de servicio de la aplicación.

1.1.5 Conclusiones

Tras estudiar las posibilidades que ofrece el mercado actual, se observa que existe la necesidad de desarrollar sistemas que cumplan los siguientes objetivos:

- Mayor automatización del proceso, eliminando todo lo posible la necesidad de que el trabajador tome la iniciativa.
- No resulte invasivo para los trabajadores (como el geoposicionamiento constante).
- Sea accesible a pequeñas y medianas empresas pudiendo seleccionar libremente el hardware necesario.
- Permitir flexibilidad en el registro ya que según la legislación actual “la empresa ha de garantizar el registro diario de jornada… sin perjuicio de la flexibilidad horaria que pueda existir” [7].

Para conseguir los objetivos anteriores se plantea el desarrollo de un sistema basado en sensores de proximidad que sirva como base software para crear aplicaciones de control de presencia en espacios cerrados.

1.2 Objetivos del Trabajo

Las principales funcionales que se pretenden conseguir con el sistema son las siguientes:

Gestión

- Permitir registrar y configurar distintas balizas y asociarlas a distintos centros de trabajo.
- Permitir configurar los distintos lugares y estados en los que puede encontrarse un trabajador (en su puesto, descansando, en desplazamiento laboral…)
- Admitir distintos perfiles de acceso a la aplicación: administrador, responsable y trabajador.
- Permitir registrar trabajadores con su información personal, jornada laboral, etc.

Control horario

- Registrar las entradas y salidas de los trabajadores mediante señales de proximidad generadas por balizas.
- Mostrar mensajes a los trabajadores que permitan especificar los tipos de movimientos que realizan.
- Almacenar la información registrada permitiendo modificar la información generada incorrectamente.

Análisis

- Visualizar estadísticas sobre la distribución de la jornada laboral de los distintos trabajadores.
- Generar informes mensuales con las horas trabajadas (incluyendo las horas complementarias y horas extras).

Además, se considera importante que el sistema sea capaz de cumplir los siguientes requisitos:

- Implementar medidas que minimicen los errores en los registros de entrada o salida.
- Implementar las soluciones utilizando librerías de código abierto que permitan ampliar el desarrollo y adaptarlo a futuras necesidades.
- Simplificar el uso de los dispositivos hardware (balizas) para que el sistema sea compatible con el mayor número de dispositivos.

Queda fuera del alcance de este trabajo el desarrollo de las siguientes capacidades:

- Funciones de configuración de las balizas: identificador, rango, potencia, etc. La configuración de los dispositivos se realizará utilizando aplicaciones externas.

1.3 Enfoque y método seguido

1.3.1 Solución hardware

Para cumplir con los objetivos propuestos se utilizarán balizas de proximidad como base hardware del sistema. Las balizas de proximidad utilizan un tipo de conexión de bajo consumo llamada Bluetooth Low Energy (BLE), introducida en el estándar 4.0. que emite una señal de muy bajo consumo. Las baterías pueden durar hasta dos años.

Este tipo de dispositivos son capaces de enviar una pequeña cantidad de información, como un identificador, que será recibida por los dispositivos móviles de los usuarios. Si estos identificadores se encuentran registrados y asociados a una localización o centro de trabajo, es posible controlar el paso de los usuarios a través de estas localizaciones.

La mayoría de los fabricantes de balizas trabajan con pedidos con un número mínimo de unidades. Aun así, se han encontrado dispositivos que se venden de manera individual.

Finalmente, se ha seleccionado una baliza de largo alcance del fabricante Avvel [8] para realizar las pruebas. Se trata de un modelo de largo alcance y configurable que funciona bajo el protocolo iBeacon.

La decisión se ha basado en la documentación y el software que se ofrece para su configuración.

1.3.2 Solución software

En cuanto al software, las principales opciones que se barajan a la hora de llevar a cabo los objetivos propuestos son: el desarrollo de una aplicación web, híbrida o nativa.

Para asegurar que la aplicación pueda acceder a las funciones del dispositivo necesarias para interactuar con las balizas bluetooth, se decide desarrollar una aplicación nativa para dispositivos móviles.

Otra de las razones es que el desarrollo nativo nos permite adaptarnos rápidamente a los cambios y mejoras que se produzcan en los protocolos de comunicaciones con los dispositivos bluetooth. Con aplicaciones web o

híbridas, los nuevos desarrollos necesarios para adaptarlas a los cambios que se produzcan dependerán de la existencia de librerías de terceros.

A lo largo de este trabajo se desarrollará una aplicación para dispositivos Android, siendo conscientes de que, para el pleno uso en un entorno real, sería necesario en un futuro desarrollar una aplicación nativa para IOs.

Para el desarrollo de las funcionalidades principales se utilizarán soluciones de código abierto[1]:

- Para la comunicación con las balizas se utilizará la librería Android Beacon Library.
- Para la elaboración de gráficas con la información recopilada se utilizará la MPAndroidChart.

1.3.3 Metodología de desarrollo

Dentro de las metodologías disponibles para el desarrollo de aplicaciones móviles descritas en [9] se escoge seguir un desarrollo en cascada. Los motivos principales son:

- Se trabaja con tiempos ajustados que no permiten realizar ciclos de iteración. Por lo tanto, se necesita una planificación estable desde el principio.
- El conjunto de requisitos va a ser estable desde el principio. No se esperan entradas o modificaciones en este punto del desarrollo.
- Cualquier requisito no implementado o errores no solventados entrarán dentro de la fase de mantenimiento.

El método de desarrollo rápido de aplicaciones nos permite tener un prototipo rápidamente. Esto es un beneficio cuando hay poco tiempo disponible, pero en este caso no es posible llevar a cabo iteraciones posteriores para implementar nuevos requisitos.

Las metodologías ágiles y mobile-D quedarían también descartadas por razones similares a la anterior. Se prefiere crear un conjunto de requisitos lo más estable posible y, a partir de ahí generar un diseño que se adecue a estos lo máximo posible. De este modo que no se estima modificar o añadir requisitos durante la duración del proyecto.

Aún así, no se descarta que sea necesario llevar a cabo modificaciones en el diseño provenientes de varias fuentes: comentarios recibidos, problemas encontrados durante la implementación, etc. Estos eventos pueden obligar a llevar a cabo un ciclo de corrección de diseño y adaptación de la implementación.

[1] El uso concreto de software de terceros puede variar durante el desarrollo del trabajo.

Finalmente, si la aplicación acabase en un producto con una finalidad comercial con un desarrollo a mayor plazo, se escogería por un desarrollo ágil que permitiese añadir funcionalidades en ciclos posteriores de desarrollo.

1.3.4 Descripción del funcionamiento

Dentro de cada centro de trabajo se colocará como mínimo una baliza de corto alcance (o cualquier baliza configurada para bajo alcance) en la entrada del centro. En el interior del edificio se pueden colocar distintas balizas de largo alcance en las distintas zonas de trabajo.

La baliza de corto alcance registra la entrada del trabajador. Si no existen más balizas asociadas al centro, se mostrará una notificación para confirmar la entrada. En caso contrario, se registrará la hora de entrada para posterior confirmación.

Si el edificio tiene más balizas asociadas, al llegar a una de las zonas de trabajo, se recibirá una notificación para confirmar la hora de entrada registrada anteriormente.

Al volver pasar por la entrada se registrar el paso para posterior confirmación. Al dejar de recibir señales. se pedirá confirmación de la salida.

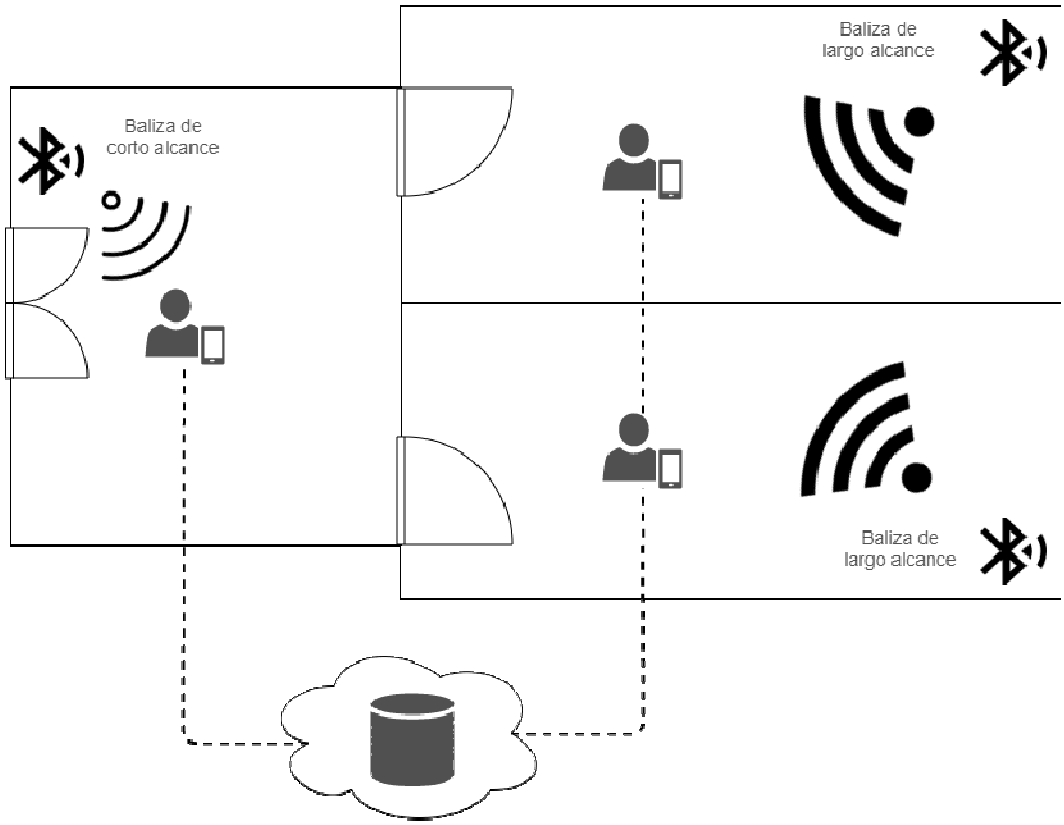

Figura 1. Descripción general

1.3.5 Riesgos

El principal riesgo dentro de este proyecto es la posibilidad de que la solución hardware no se adecúe a las necesidades y objetivos establecidos. Para paliar los posibles derivados de esto, se plantean las siguientes medidas:

- Valorar distintas modelos de balizas disponibles en el mercado y sustituir el escogido por otro si hubiese necesidad.
- Utilizar más de una zona de detección (balizas exteriores e interiores) para evitar falsas entradas, salidas y solapamientos.
- Valorar el uso de aplicaciones que simulan dispositivos BLE.

Por otra parte, desarrollar una interfaz para consultar estadísticas, registros de entrada y salida, filtrar por distintos parámetros, etc. puede suponer un coste elevado dentro del marco de este trabajo. Por este motivo, existe la posibilidad de no poder desarrollar un módulo de análisis que cumpla los objetivos establecidos.

Para evitar en la medida de lo posible el problema, se seguirá una estrategia que permita descartar funcionalidades llegado el caso. Por ejemplo, independizar los distintos tipos de consulta tanto en el diseño como en la implementación.

1.4 Planificación del Trabajo

Para llevar a cabo el trabajo se parte de la base de un trabajo diario de unas 3 horas en días laborales y 6 horas en días no laborables (sábados y domingos).

A continuación, se muestra un listado de las tareas a realizar y su organización en los distintos hitos de entrega:

Tarea	Duración	Inicio	Final
PEC1 – Plan de trabajo	45 horas	23/09/19	5/10/19
Estudio de mercado	9 horas	23/09/19	25/10/19
Selección de hardware	3 horas	26/09/19	27/09/19
Definición de alcance	6 horas	28/09/19	28/09/19
Estrategia de la solución	12 horas	29/09/19	01/10/19
Configuración del hardware	3 horas	02/10/19	02/10/19
Planificación	12 horas	03/10/19	05/10/19
PEC2 – Diseño	87 horas	06/10/19	27/10/19
Prueba de concepto con balizas	12 horas	06/10/19	08/10/19
Elaboración de perfiles de usuario	6 horas	09/10/19	10/10/19
Elaboración de escenarios	9 horas	11/10/19	12/10/19
Elaboración de prototipo	33 horas	13/10/19	20/10/19
Elaboración de casos de uso	9 horas	21/10/19	23/10/19
Esquema de bases de datos	3 horas	24/10/19	24/10/19
Diseño de arquitectura	3 horas	25/10/19	25/10/19
Diagramas UML	12 horas	26/10/19	27/10/19

PEC3 – Implementación	168 horas	28/10/19	10/12/19
Implementación de modelo de datos	9 horas	28/10/19	30/10/19
Implementación de módulo de configuración	27 horas	31/10/19	06/11/19
Implementación de servicio de comunicación con balizas	21 horas	07/11/19	11/11/19
Implementación de módulo de registro horario	45 horas	12/11/19	23/11/19
Implementación de módulo de informes	18 horas	24/11/19	28/11/19
Implementación de módulo de consultas	48 horas	29/11/19	10/12/19
PEC 4 – Entrega final	120 horas	11/12/19	03/01/20
Elaboración de conjunto de pruebas	21 horas	11/12/19	15/12/19
Corrección de errores	9 horas	16/12/19	18/12/19
Redactar memoria de implementación	12 horas	19/12/19	21/12/19
Manual de usuario	12 horas	22/12/19	23/12/19
Manual de compilación y ejecución	6 horas	24/12/19	25/12/19
Elaboración de la presentación	24 horas	26/12/19	29/12/19
Montaje del video	18 horas	30/12/19	02/01/20
Revisión de la memoria	6 horas	03/01/20	03/01/20

A continuación, presenta un diagrama de Gantt con la organización temporal de la planificación. Para la elaboración se ha tenido en cuenta el número de horas estimadas de trabajo.

También se ha establecido que, a partir del día 22 de diciembre se consideran todos los días con la misma carga de trabajo que en fin de semana. Teniendo en cuenta esto y que el tiempo de implementación dentro de la PEC3 podría ser ajustado, se ha decidido incluir la elaboración de procedimientos de prueba dentro de la PEC4.

Nombre	Fecha de inicio	Fecha de fin
Total	23/09/19	3/01/20
PEC1 - Plan de trabajo	23/09/19	5/10/19
Estudio de mercado	23/09/19	25/09/19
Selección de hardware	26/09/19	27/09/19
Definición de alcance	28/09/19	28/09/19
Estrategia de la solución	29/09/19	1/10/19
Configuración de hardware	2/10/19	2/10/19
Planificación	3/10/19	5/10/19
PEC 2 - Diseño	6/10/19	27/10/19
Prueba de concepto con balizas	6/10/19	8/10/19
Elaboración de perfiles de usuario	9/10/19	10/10/19
Elaboración de escenarios	11/10/19	12/10/19
Elaboración de prototipo	13/10/19	20/10/19
Elaboración de casos de uso	21/10/19	23/10/19
Diagrama de bases de datos	24/10/19	24/10/19
Diseño de arquitectura	25/10/19	25/10/19
Diagramas UML	26/10/19	27/10/19
PEC 3 - Implementación	28/10/19	10/12/19
Impl. de modelo de datos	28/10/19	30/10/19
Impl. de módulo de configuración	31/10/19	6/11/19
Impl. de servicio comunicación con balizas	7/11/19	11/11/19
Impl. de módulo de registro horario	12/11/19	23/11/19
Impl. de módulo de informes	24/11/19	28/11/19
Impl. de módulo de consultas	29/11/19	10/12/19
PEC 4 - Entrega final	11/12/19	3/01/20
Elaboración de pruebas	11/12/19	15/12/19
Corrección de fallos	16/12/19	18/12/19
Memoria de implementación	19/12/19	21/12/19
Manual de usuario	22/12/19	23/12/19
Manual de compilación y ejecución	24/12/19	24/12/19
Presentación	26/12/19	29/12/19
Montaje de vídeo	30/12/19	2/01/20
Revisión de memoria	3/01/20	3/01/20

Figura 2. Planificación

1.5 Breve sumario de productos obtenidos

Como resultado del trabajo realizado se obtendrán los siguientes productos entregables:

- Instalador de la aplicación Android en formato .apk.
- Código fuente de la aplicación Android comprimida en formato .rar.
- Memoria de trabajo en formato PDF.
- Manual de usuario de la aplicación como anexo en la memoria.
- Manual de compilación y ejecución como anexo en la memoria.
- Presentación en formato PPT (PowerPoint).
- Video presentación en formato TBD

1.6 Estructura del documento

A continuación, se describe la organización y contenido del documento.

Diseño centrado en el usuario (DCU): esta sección describe el enfoque del producto desde la perspectiva del diseño centrado en la experiencia de usuario. En este apartado se lleva a cabo un análisis de la competencia y entrevistas a potenciales usuarios. Con la información recabada se describen los perfiles de usuario y los escenarios de uso de la aplicación.

Finalmente, en esta sección se muestra un prototipo visual de la aplicación que será evaluado por los potenciales usuarios.

Especificación y diseño de la aplicación: en esta sección se describen los casos de uso del producto. Por otra parte, se establece la arquitectura del sistema, así como el diseño de clases (UML) de la aplicación.

Implementación: en esta sección se describen los principales elementos y decisiones de implementación.

Pruebas: esta sección incluye los distintos procedimientos de pruebas realizados para comprobar el funcionamiento de la aplicación y la adecuación de esta a los requisitos establecidos. Finalmente, incluye un listado de los errores registrados y resueltos.

Conclusiones: lecciones aprendidas a los largo del desarrollo del producto, así como futuras mejoras y modificaciones a ejecutar.

2. Diseño Centrado en el Usuario (DCU)

2.1 Usuarios y contexto de uso

Debido a que se trata de un producto para uso profesional, para llevar a cabo la fase de investigación y requisitos de usuario, nos interesa estudiar la manera en que sus potenciales usuarios realizan este tipo de tareas.

Por ello, dentro de los métodos sugeridos en [10] para la fase de conceptualización, se han llevado a cabo los siguientes:

- Análisis de la competencia
- Entrevista personal

2.1.1 Análisis de la competencia

En apartados anteriores se ha podido ver una selección de aplicaciones que se pueden considerar como competencia de nuestro producto. El problema principal a la hora de realizar el análisis es que todos los productos se usan bajo licencia, con lo cual no es posible, a priori ver el funcionamiento real de la aplicación.

Aun así, se ha recabado información que nos permite obtener ideas sobre como diseñar nuestro producto. A continuación, se pueden ver algunos aspectos que se van a tomar como referencia para nuestro diseño.

Línea de tiempo y calendario

En la aplicación Trackpeople [3] podemos ver como el usuario puede ver un calendario con los días trabajados. Al seleccionar el día se puede visualizar una línea de tiempo con los distintos eventos de la jornada laboral: inicio, fin, descansos, etc.

Este elemento de diseño aporta una visualización rápida y eficaz de la jornada laboral, así como permitir una navegación rápida entre los distintos días.

Figura 3. Benchmarking - Calendario y línea de tiempo

Consultas y análisis de datos

La aplicación Timepro [5] nos permite visualización de estadísticas sobre el tiempo dedicado por el trabajador a cada una de las posibles situaciones en las que se encuentra a lo largo del día.

La idea es trasladar este tipo de diseño a un módulo de nuestra aplicación que nos permita realizar consultas sobre cada uno de los trabajadores (en caso de ser responsable de recursos humanos).

Figura 4. Benchmarking - Consultas y análisis

Informes

Con la aplicación Controllaboral [1] es posible ver un ejemplo de informe mensual de la jornada laboral de un trabajador. Aunque la imagen corresponde al formato web, es posible hacerse una idea de la salida final que debe tener las aplicaciones que pretenden cumplir la legislación vigente.

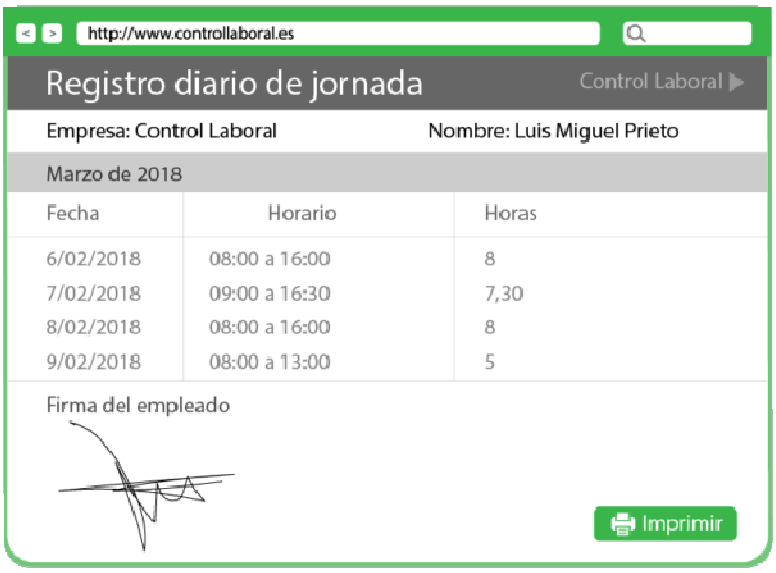

Figura 5. Benchmarking - Informes

2.1.2 Entrevistas

Como sujeto de la entrevista, se ha seleccionado a una persona que desempeña el cargo de responsable de R.R.H.H. (recursos humanos) para realizar una entrevista semiestructurada.

Para ello, se han preparado una serie de preguntas abiertas que nos permiten saber entre otras cosas: como se realiza la gestión de la jornada laboral en una empresa real, cuales son los problemas que se encuentra, así como información personal.

Con esta información es posible encontrar mejoras posibles a los productos existentes, plantear nuevos requisitos sobre el producto a desarrollar y generar un perfil del usuario principal de la aplicación.

A continuación, se incluye una transcripción fiel de la entrevista.

Datos personales

Nombre: Carolina
Apellidos: Sánchez Carmona
Edad:32
Sexo: mujer
Estudios:
Licenciatura en Administración y Dirección de Empresas.
Master en RRHH

¿Qué puesto ocupa dentro de la empresa?
Responsable de RRHH

¿Cuáles son sus funciones dentro de la empresa?
Me encargo de los procesos de selección e incorporación a la empresa, gestión del clima laboral, control horario y tramitación de permisos y vacaciones de los trabajadores, gestión de formación, control de prevención de riesgos laborales, resolución de conflictos laborales con los empleados...

¿Qué mecanismos se utilizan actualmente en la empresa para realizar el control de la jornada laboral de los trabajadores?
Utilizamos una APP que funciona desde el móvil personal del empleado o desde el PC

Explica qué acciones deben llevar a cabo los empleados para registrar su jornada laboral.
Deben indicar la entrada y la salida de la oficina, así como las pausas para descansar, comer, hacer gestiones fuera de la empresa, indicar pausa por cita médica...

También la utilizan para solicitar las vacaciones, permisos y disfrute de horas extraordinarias.

¿Qué acciones debe realizar como responsable de RRHH para comprobar que los registros son correctos?
Debe comprobar periódicamente que los registros se han realizado correctamente, y completar o corregir los errores u omisiones que se produzcan, para lo cual, debo citar al empleado y que me justifique el error y omisión del fichaje.

Así mismo, tengo que acceder periódicamente para revisar las solicitudes de vacaciones, permisos o disfrute de horas extra, para cuadrar los turnos y confirmar o denegar las solicitudes.

¿Suelen existir errores u omisiones por parte de los empleados en los registros horarios?
Es muy habitual, porque el empleado tiene que acordarse cada día de fichar su entrada cuando llega y la salida cuando se marcha, y se olvidan continuamente de fichar justo en el momento que se produce la entrada o la salida, produciéndose fichajes erróneos o inexactos.

Por ejemplo, suele ocurrir que algún empleado no se acuerda de fichar la entrada, y ficha 20 o 30 minutos más tarde después de haber entrado realmente en la oficina. Debido a esto, ese día le faltarán minutos en el cómputo diario de horas, a menos que me lo indique para que yo lo corrija manualmente.

También es muy común que se olviden fichar la salida, por lo que, a las 23:59h, el programa registra una salida automáticamente, generando una acumulación de horas errónea que luego tengo que corregir a mano con el empleado.

Por último, para las ausencias justificadas, el empleado no puede adjuntar un justificante. Esto ha de hacerlo personalmente, y no es posible dejar rastro en la APP de dicha justificación.

¿Qué mejoras propondrías al actual sistema?
Lo principal es que la aplicación detecte de algún modo que el empleado ha llegado a su puesto de trabajo y lo avise para que marque su entrada o salida en ese momento (para que no existas desfases de tiempo).

Sería perfecto que la aplicación deje huella del momento en el que detecta esa entrada o salida, por si el empleado no lo confirma, que luego sea fácil detectar el momento exacto en el que se produjo la incidencia y poder corregirlo a mano.

En cuanto a las ausencias justificadas, es importante que empleado pueda dejar "notas" y adjuntar el justificante de su ausencia.

¿Qué tipo de dispositivos suele utilizar durante su trabajo? ¿Para qué funciones los utiliza?

Utilizamos móvil y PC. El móvil es lo más cómodo y sencillo, porque siempre lo tenemos encima, y para realizar los fichajes es lo más sencillo. El PC lo utilizamos más para la solicitud de vacaciones y permisos, pera consultar los informes de horas mensuales...

¿Qué tipo de dispositivos suele utilizar en su tiempo libre? ¿Para qué funciones los utiliza?
Sobre todo, el móvil. Lo utilizo para todo (llamadas, mensajes, emails, apps de ocio, noticias, compras online...)

También suelo utilizar la Tablet, cuando estoy en casa relajada, más para actividades de ocio.

El PC sólo lo utilizo en el trabajo, o en casa cuando tengo que redactar algo o conectarme en remoto al PC del trabajo.

2.1.3 Conclusiones

Con la información recabada, es posible realizar un perfil de los usuarios principales de la aplicación, así como desarrollar una serie de escenarios que permitan establecer los requisitos y casos de uso que debe cumplir la aplicación.

Por otra parte, de la entrevista se pueden extraer ideas que se considerarán en el apartado de mejoras dentro de este documento. Algunas de éstas pueden ser:

- Permitir adjuntar documentos como justificantes de determinadas ausencias.
- Permitir registrar un calendario de días festivos.
- Permitir registrar las vacaciones de los empleados.

A partir de este punto, se comienza a hacer referencia a la aplicación utilizando el nombre comercial escogido: **Timestone**.

2.2 Diseño conceptual

Con la información recabada, es el momento de perfilar a los potenciales usuarios de la aplicación y los escenarios principales que debe contemplar.

2.2.1 Fichas de persona

Nombre: Carolina
Edad: 33 años
Nivel de estudios: estudios universitarios
Profesión: técnico de recursos humanos

Descripción de la persona

Carolina tiene 33 años, está casada y tiene una hija de dos años. Es licenciada en Administración y Gestión de Empresas y actualmente trabaja en una empresa como responsable de recursos humanos.

En la oficina realiza a diario tareas como: gestión de contratos, altas y bajas, así como controlar las vacaciones y ausencias de los trabajadores. Su trabajo también implica publicar ofertas de empleo y dirigir las entrevistas a los candidatos.

Una cuestión que se ha convertido en una prioridad es el control de la jornada laboral de los trabajadores de la empresa debido a las recientes modificaciones en la legislación.

Para llevar a cabo su trabajo, Carolina utiliza una variedad de programas y aplicaciones que se utilizan para llevar a cabo distintas tareas. Para el control de la jornada laboral, actualmente se utiliza una aplicación que implica que los trabajadores registren su horario de forma manual.

Algunas veces siente que sus compañeros no perciben la cantidad e importancia de las tareas que tiene que acometer en su día a día ya que, en muchos casos debe estar pendiente de que los trabajadores registren su jornada.

Nombre: Antonio
Edad: 26 años
Nivel de estudios: estudios universitarios
Profesión: analista software

Descripción de la persona

Antonio tiene 26 años, está soltero y trabaja como analista software en una pequeña empresa con unos 20 empleados.

Tanto en el trabajo como en su tiempo libre, Antonio trabaja con todo tipo de tecnología.

Dentro de sus funciones como analista, algunas veces tiene que realizar visitas a distintos clientes, viajar para llevar a cabo presentaciones o validar los proyectos en que participa.

los

A Antonio le encanta resolver problemas complejos y pasar horas delante del ordenador. Aun así, también es consciente de que su trabajo implica gran cantidad de formalismos con los cuales no se siente tan a gusto y que muchas veces deja en segundo plano.

Últimamente la empresa insiste mucho en que debe rellenar los formularios dedicados a reflejar la jornada laboral. La realidad es que muchas veces se le olvida y tiene que parar su trabajo para completarlos.

2.2.2 Fichas de escenario

Descripción de escenario – Configuración de aplicación

A Oscar le encargan configurar la nueva aplicación de registro de jornada laboral. Existen aspectos de la configuración que dependen de la organización de la empresa, así que decide reunirse con Carolina, la responsable de R.R.H.H.

Entre los dos registran los centros laborales existentes, así como el tipo de incidencias que contemplará la aplicación. Tras la reunión concretan que la aplicación gestionará las siguientes eventualidades: salidas de trabajo, descansos, paradas para comer y salidas justificadas (como una consulta médica).

Con toda la información recopilada, Oscar abre la aplicación como administrador e introduce los datos y ésta queda almacenada para su posterior uso.

Ahora Oscar tiene que configurar los dispositivos que se usarán en el centro de trabajo. En primer lugar, Oscar coloca los dispositivos bluetooth en los lugares

estratégicamente escogidos. Coloca uno en la entrada y luego uno en cada oficina.

Como administrador, al abrir la aplicación, ésta le muestra los dispositivos detectados. Oscar añade el primer dispositivo y lo asocia al nuevo centro de trabajo indicando que se encuentra en la entrada.

Finalmente, hace lo mismo con el resto de dispositivos indicando que se encuentran en el interior del edificio.

Descripción de escenario – Registro de entrada

Antonio llega por la mañana a la oficina, pasa por recepción y se dirige a su puesto de trabajo. Por el camino, se encuentra con un compañero y le recuerda que esa tarde tienen una reunión con un cliente en sus oficinas. Al final acaban ultimando los detalles de la reunión.

Cuando Antonio por fin llega a su sitio, recibe una notificación en el móvil de la nueva aplicación de control de jornada laboral: Timestone. La aplicación le pide confirmación para la hora de entrada en la oficina. La hora que indica se corresponde con el momento en que pasó por recepción. Antonio acepta y continua con su trabajo.

Descripción de escenario – Registro de salida temporal

Llega la hora de salir para la reunión con el cliente y Antonio sale de la oficina y se dirige al coche. Cuando está llegando, recibe otra notificación en el móvil. La aplicación le muestra la hora en que salió de la oficina y pide que indique el tipo de salida que está llevando a cabo. Antonio selecciona la opción correspondiente a una visita al cliente.

La reunión se alarga hasta el final de la jornada, así que Antonio regresa directamente a casa. Al día siguiente, de camino a la oficina, revisa la aplicación y comprueba que tiene una alerta indicando que el día anterior no notificó el fin de la jornada. Revisa la alerta e introduce la hora de salida del día anterior.

La alerta ha desaparecido y el registro de jornada laboral se mantiene correcto.

Descripción de escenario – Registro de fin de jornada

Al final del día, Antonio ha descubierto que la última versión del proyecto presenta un error que impediría el trabajo normal de los compañeros durante el día siguiente. Se lo comenta a su superior y llegan al acuerdo de seguir trabajando para corregir el error.

Finalmente consigue arreglar los problemas y sale de la oficina. De camino al coche recibe una notificación de la aplicación en el móvil para confirmar la salida. Como motivo de la salida, Antonio indica el fin de la jornada.

La aplicación computará el número total de horas desde la entrada hasta la salida.

Descripción de escenario – Consulta y generación de informes

Ha comenzado un nuevo mes y Carolina necesita generar los informes de jornada laboral del mes anterior. Para hacerlo, utiliza la nueva aplicación instalada en la empresa: Timestone.

Antes de generar los informes necesita revisar que no existan irregularidades en la jornada laboral de los trabajadores. Antes, tenía que comprobar manualmente los registros en busca de posibles errores, ahora simplemente revisa las alertas que le ofrece la aplicación.

Comprueba que existen varias alertas que indican que no se ha marcado la hora de fin de jornada. Carolina se pone en contacto con los empleados sobre los que existen este tipo de alertas y les insta a revisar su aplicación para resolver las incidencias.

Como último paso, revisa el número de horas realizadas por los trabajadores para confirmar que coinciden con lo estipulado en su contrato. Comprueba que en algunos trabajadores existen horas extra registradas y se pone en contacto con los responsables para confirmar que dichas horas fueron acordadas.

Finalmente, cuando todo está en orden, Carolina genera los informes mensuales que serán almacenados para futuras consultas o requerimientos.

2.3 Prototipado

En este punto, se desarrollo un prototipo horizontal de la aplicación que permitirá tomar decisiones funcionales y de diseño antes de comenzar la implementación.

Este prototipo funcionará también como herramienta de evaluación temprana para introducir cambios.

2.3.1 Árbol de navegación

El árbol de navegación permite establecer las relaciones y jerarquías entre las distintas pantallas que forma parte de la aplicación. El árbol de la aplicación se ha dividido entre pantallas de configuración y pantallas correspondientes a la operativa habitual.

Las pantallas de configuración comprenden la **configuración y el registro** de los siguientes conceptos:

- Usuarios
- Dispositivos BLE.
- Centros de trabajo.
- Tipos de eventos de parada y ausencia.

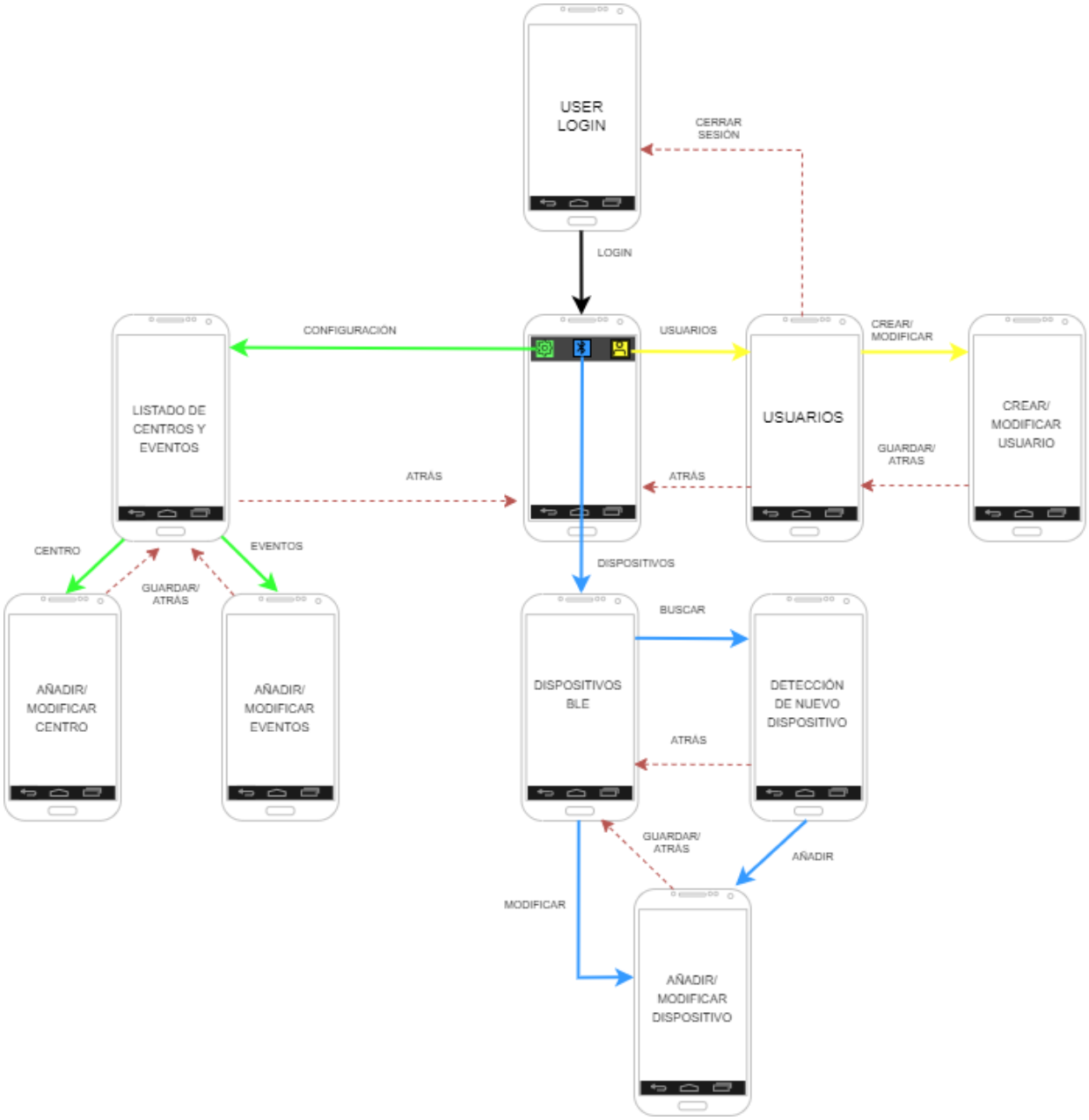

Figura 6. Árbol de navegación – Configuración

Para la operativa principal de la aplicación, se ha escogido un modelo de **navegación basado en pestañas**, las cuales contienen los principales bloques funcionales de la aplicación: notificaciones, registro de jornada, consultas y generación de informes.

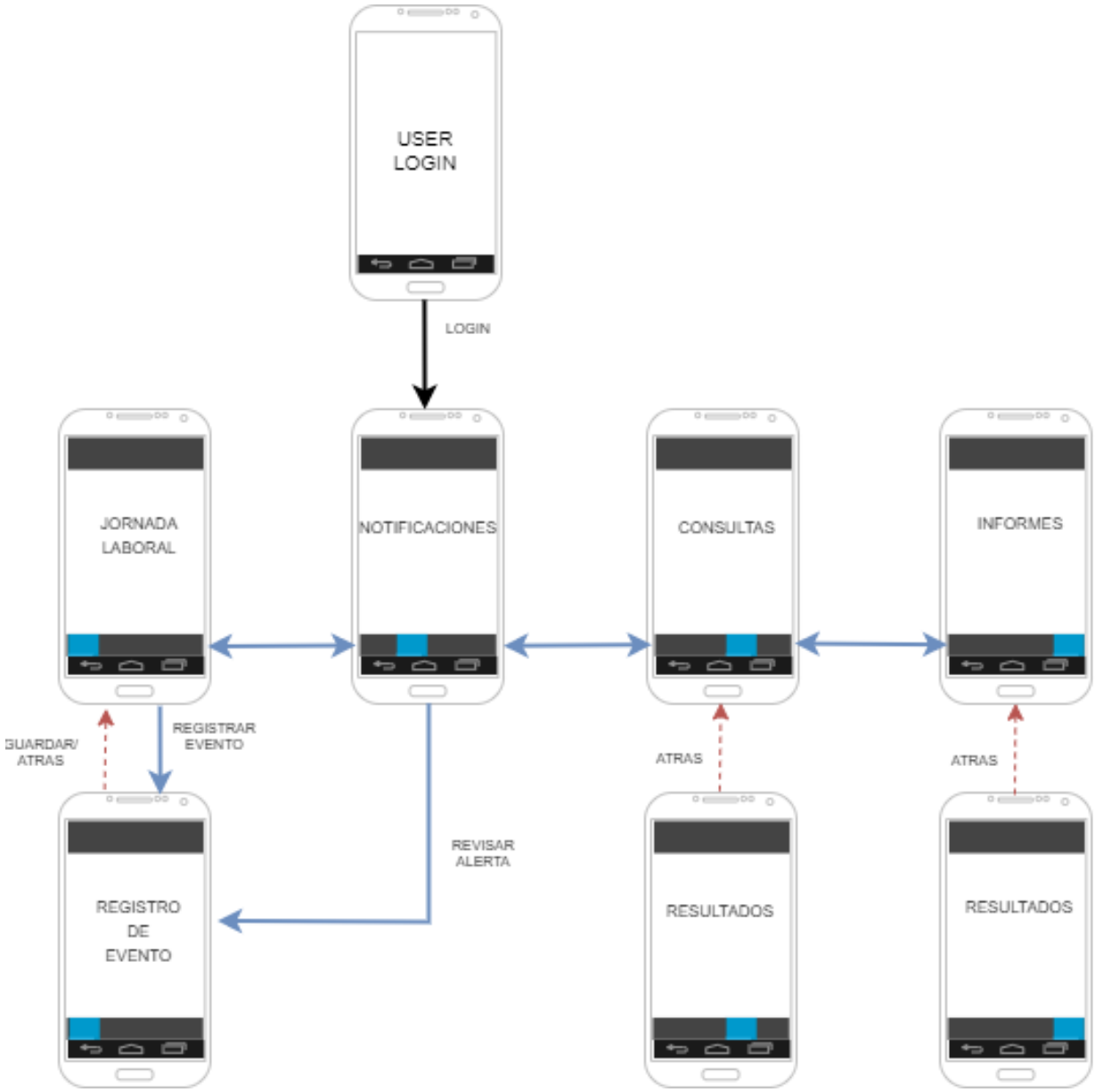

Figura 7. Árbol de navegación – Operación

2.3.2 Prototipo de alta fidelidad

Para esta sección se llevado a cabo el diseño de las principales pantallas de la aplicación. Como resultado, se ha obtenido también un prototipo interactivo que servirá como herramienta para la evaluación posterior.

Para esta tarea, se ha utilizado la herramienta **Adobe XD**.

Login

Esta es la primera pantalla de la aplicación. Para acceder es necesario introducir el correo y la contraseña registradas.

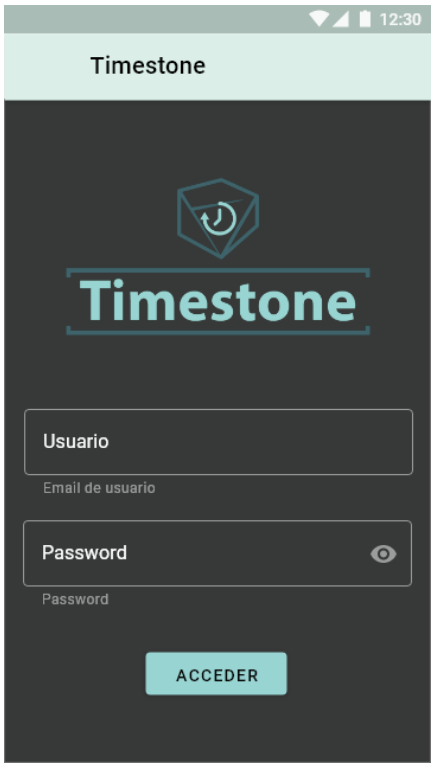

Figura 8. Pantallas - Login

Dispositivos BLE

Estas pantallas muestran el listado de dispositivos registrados. Así mismo, muestran los nuevos dispositivos detectados en el momento actual. En la pantalla de configuración pueden ser editados y asociados a un centro de trabajo y posición.

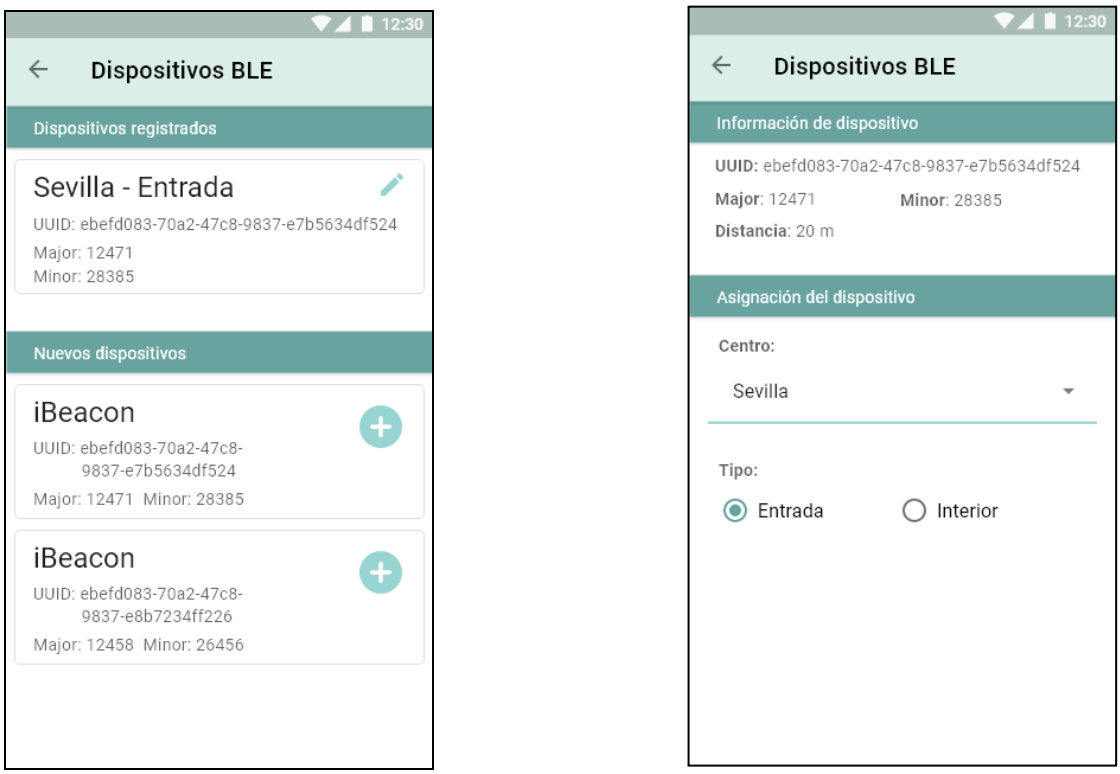

Figura 9. Prototipo – Configuración de dispositivos BLE

Centros y eventos

Estas pantallas muestran el listado de centros y eventos de ausencia registrados. Con ellas también es posible crear nuevas entradas y editar las ya existentes.

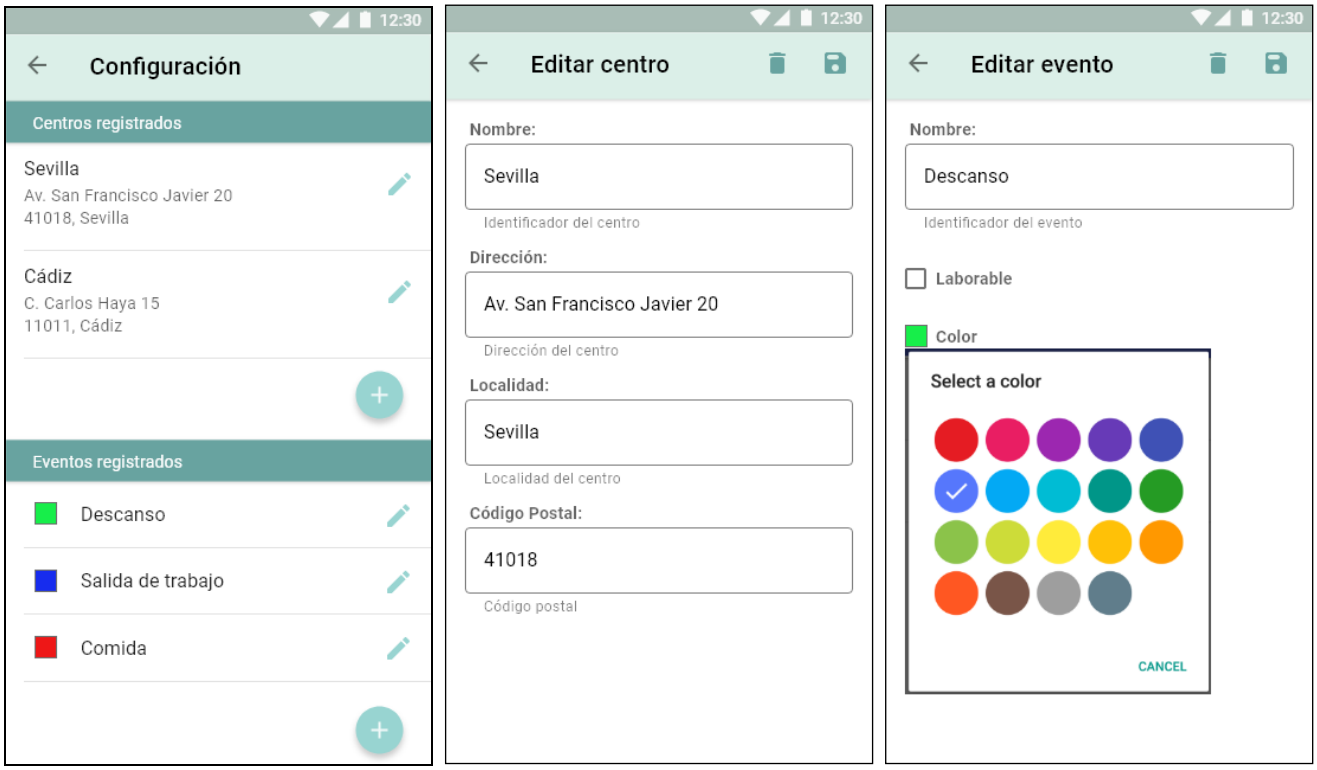

Figura 10. Prototipo – Configuración del sistema

Usuarios

Dentro de estas pantallas, la persona responsable de la gestión de la aplicación podrá registrar y editar la información de los trabajadores de la empresa. Un usuario normal también puede visualizar sus datos personales.

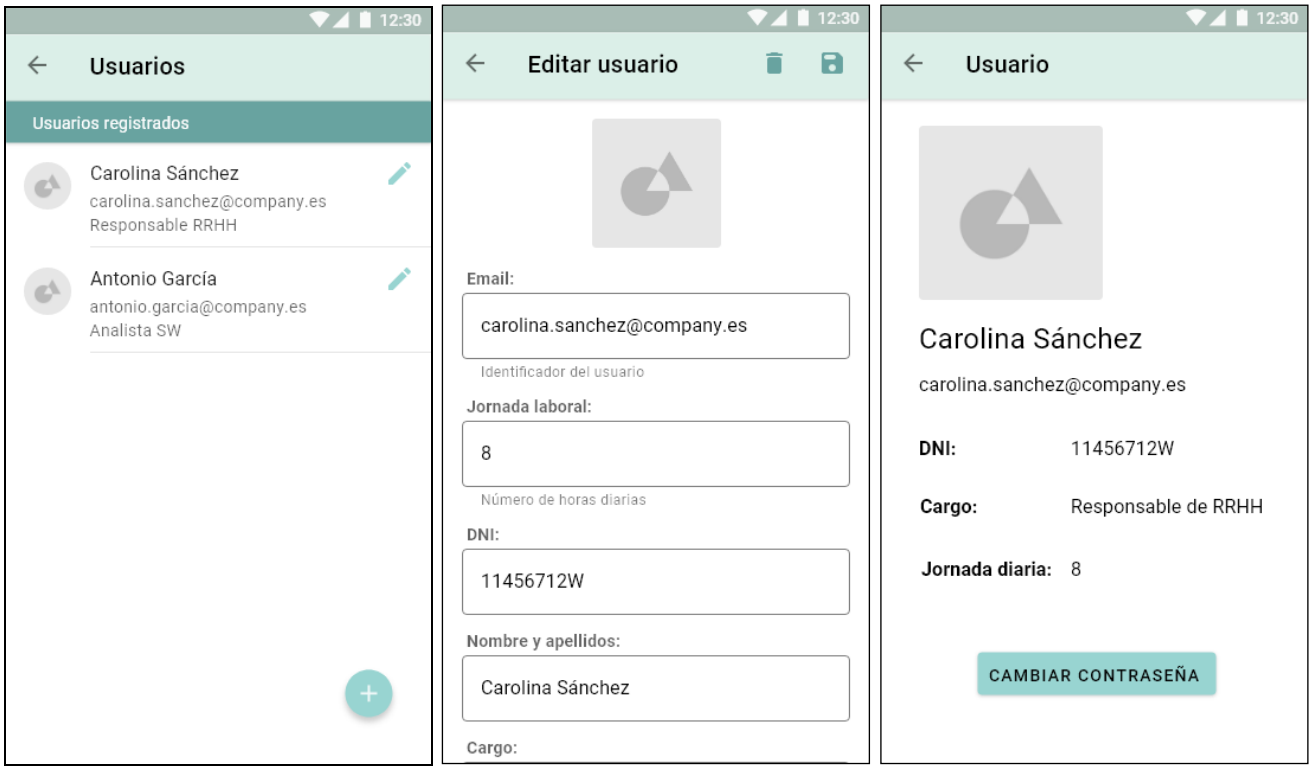

Figura 11. Prototipo - Listado usuarios

Alertas

Esta es la primera pantalla que se muestra tras el acceso. Contiene las alertas del sistema. Al revisarlas, se lleva al usuario a la pantalla donde puede resolverlas. Esta pantalla también se mostrará directamente cuando el usuario revise las notificaciones propias de la aplicación.

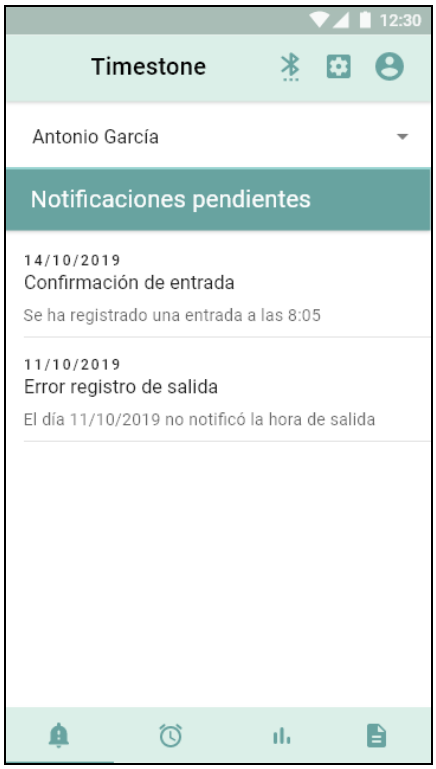

Figura 12. Prototipo - Notificaciones

Registro de jornada

El corazón de la aplicación y las pantallas que a diario van a ser utilizadas por todos los usuarios. Por un lado, muestra la jornada laboral del día seleccionado con los eventos registrados. También es posible registrar nuevos eventos de entrada y salida, así como editar los ya existentes.

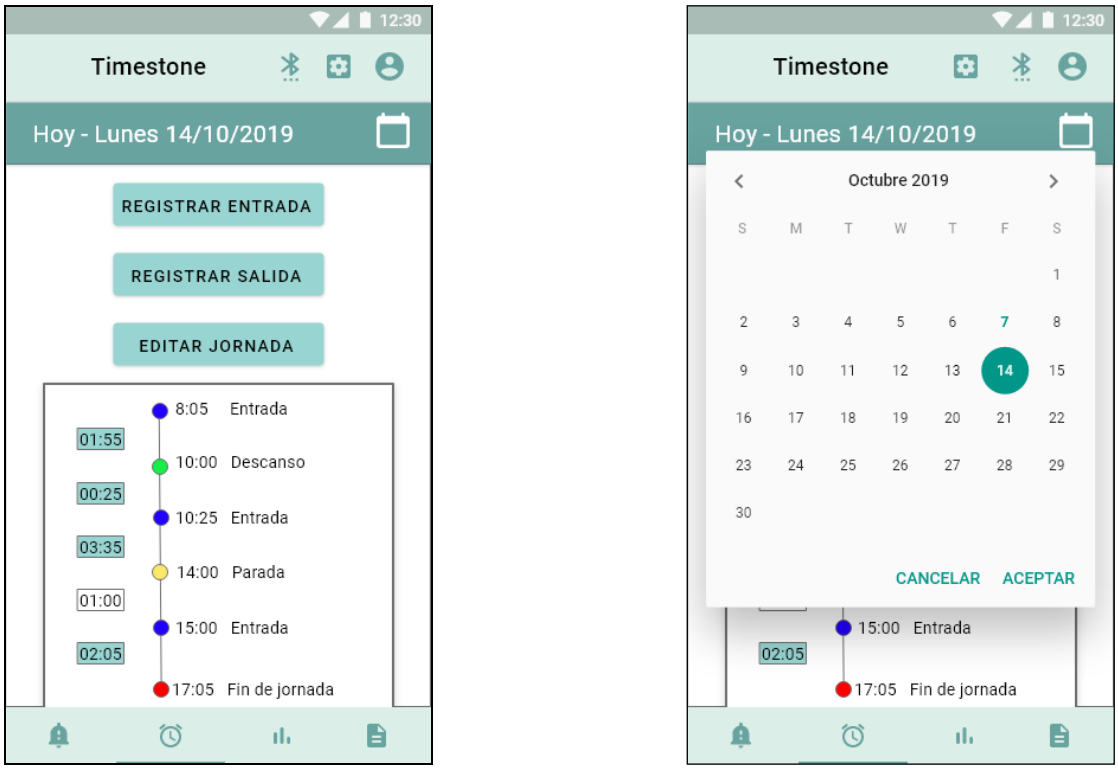

Figura 13. Prototipo - Jornada laboral

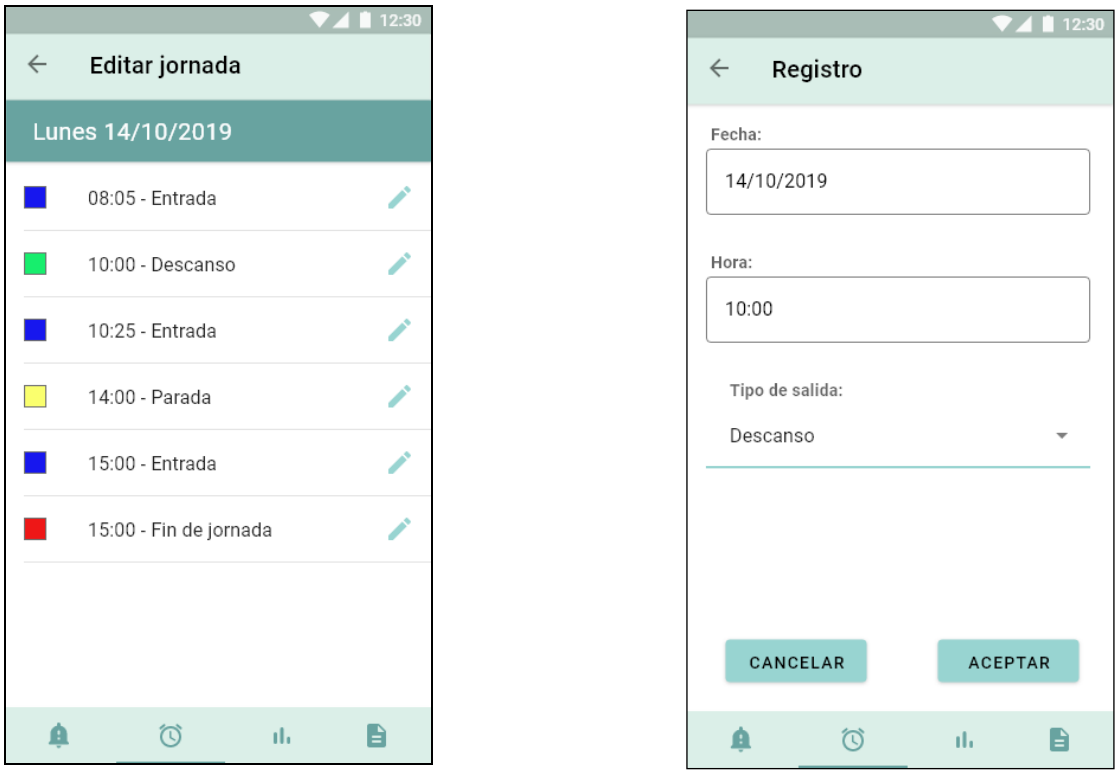

Figura 14. Prototipo - Editar jornada

Consultas

Esta funcionalidad permite generar estadísticas sobre el tiempo dedicado por los trabajadores al trabajo efectivo o en los distintos eventos registrados.

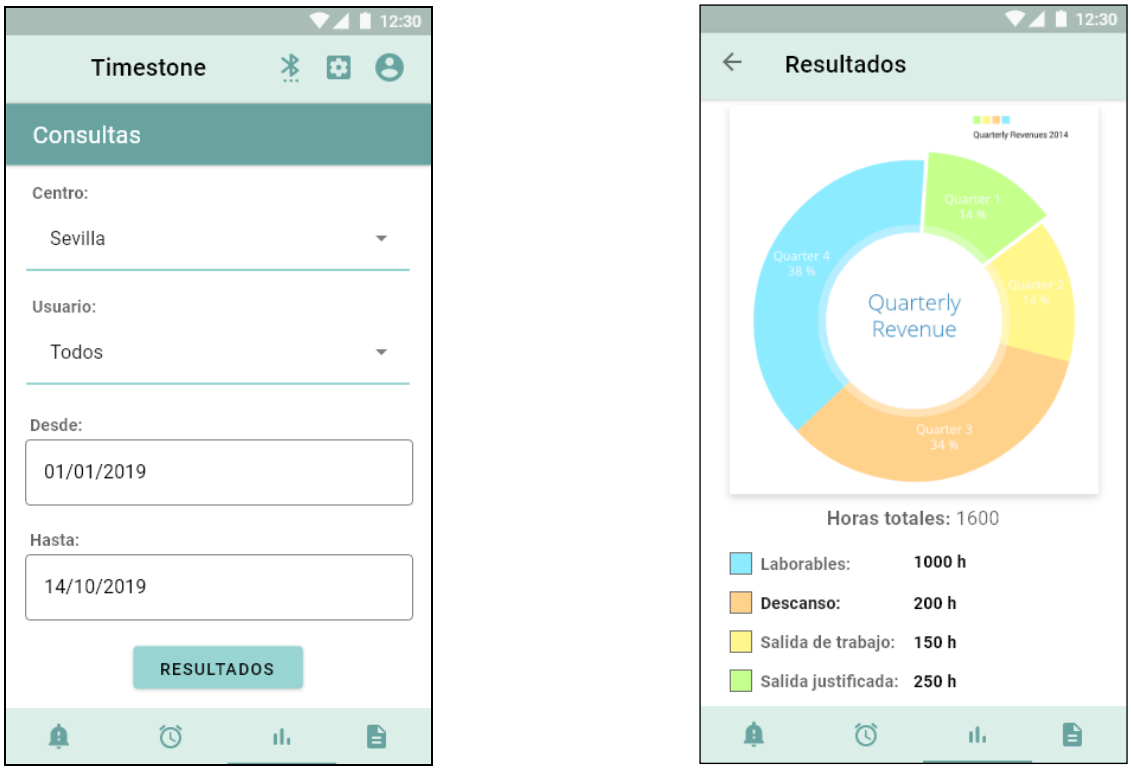

Figura 15. Prototipo – Consultas

Informes

Con la información recopilada es posible generar informes oficiales que reflejen la jornada laboral de un mes completo para el trabajador especificado.

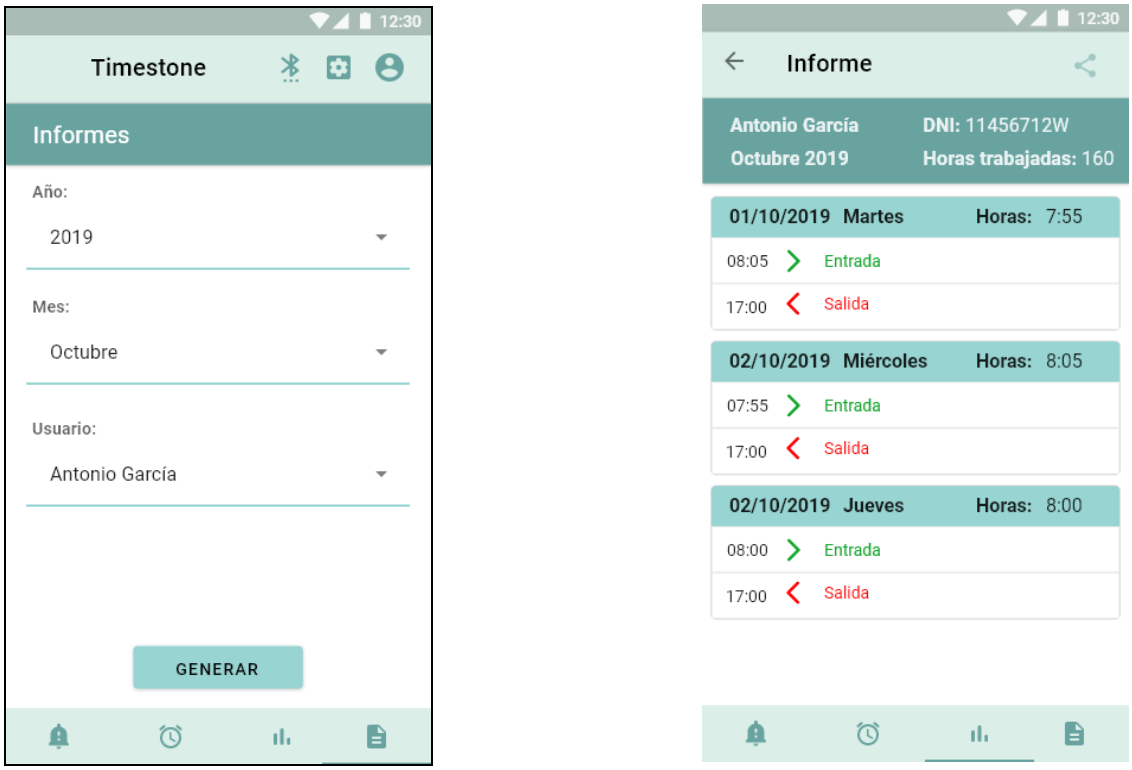

Figura 16. Prototipo – Informes

2.4 Evaluación

Una vez elaborado el prototipo, es necesario evaluarlo para obtener mejoras y corregir posibles ausencias. Para ello, se van a utilizar tres fuentes principales:

- Comentarios del sujeto utilizado durante las entrevistas. Para ello, se le presentará el prototipo en modo interactivo.
- Comentarios del director del proyecto. En este caso, el director del proyecto hace las veces de cliente.
- Limitaciones o mejoras que puedan surgir durante la fase de desarrollo. Disponer de librerías específicas puede hacer replantear el uso de ciertos elementos de diseño.

La evaluación del diseño se llevará a cabo tras recibir los comentarios del director. En este punto se añaden el resto de modificaciones obtenidas a través del resto de fuentes.

Finalmente, se tomará la decisión de qué cambios se van a incorporar. Se reflejarán las modificaciones en el diseño y la implementación se realizará en función de este diseño final.

2.4.1 Comentarios de usuario

A través de la evaluación del usuario se obtienen las siguientes modificaciones sobre las funcionalidades planteadas:

- El listado de notificaciones no permite descartar aquellas que no se desean confirmar.
- Para el usuario responsable de RRHH la pantalla de notificaciones muestra un filtro de usuario para ver las pertenecientes a cada trabajador. Esto no resulta útil ya que el responsable solo está interesado por los posibles errores.
- No es posible eliminar libremente registros en la pantalla de edición de la jornada laboral.
- En la ventana de gestión de jornada laboral, los botones de REGISTRAR ENTRADA, REGISTRAR SALIDA y EDITAR JORNADA ocupan demasiado espacio.

2.4.2 Limitaciones y mejoras del desarrollo

A lo largo del desarrollo, se han observado problemas o limitaciones que han obligado a realizar modificaciones. Estas son:

- Se observa que no es útil mostrar la distancia de la baliza en la pantalla de edición de éstas. La distancia serviría para identificar las balizas detectadas cuando aún no han sido registradas.

- No existe una manera de cambiar de usuario una vez registrado en la aplicación (se asume que el acceso queda registrado en el dispositivo).
- La ventana de creación de registros no permite establecer el centro de trabajo.

2.4.3 Acciones tomadas

Como respuesta a las valoraciones anteriores se han realizado las siguientes modificaciones sobre el diseño original:

- Añadido botón para eliminar notificaciones del listado.
- El usuario responsable de RRHH solo verá las notificaciones de error de los trabajadores. Se elimina el selector de usuario.
- Se añade un botón para eliminar libremente los registros en la ventana de edición de la jornada laboral.
- Se sustituyen los botones de REGISTRAR ENTRADA, REGISTRAR SALIDA y EDITAR JORNADA por botones icónicos (pendiente).
- La distancia de la baliza se muestra en el listado de balizas detectadas para poder determinar cuál se va a configurar.
- Se añade un botón que permite al usuario salir de la aplicación cambiando de usuario.
- Se ha añadido un selector de centro de trabajo en la ventana de creación de registros.

3. Especificación y diseño de aplicación

3.1 Casos de uso

A continuación, se presentan los principales casos de uso definidos para la aplicación. Antes, es necesario definir los distintos actores implicados en los distintos casos.

- **Administrador**: puede realizar tareas de configuración del sistema, registrar usuarios y dispositivos BLE.
- **Responsable de RRHH**: puede llevar a cabo tareas de configuración del sistema, registrar usuarios y es capaz de visualizar y editar la información del resto de trabajadores.
- **Trabajador**: puede introducir y modificar registros en su jornada laboral. No puede ver la información del resto de usuarios.
- **Aplicación**: la propia aplicación genera eventos automáticos causados por la detección de balizas o eventos temporales.

A la hora de seleccionar el actor para cada caso de uso, hay que tener en cuenta que el **Responsable de RRHH** también se considera como **Trabajador**.

3.1.1 Configuración del sistema

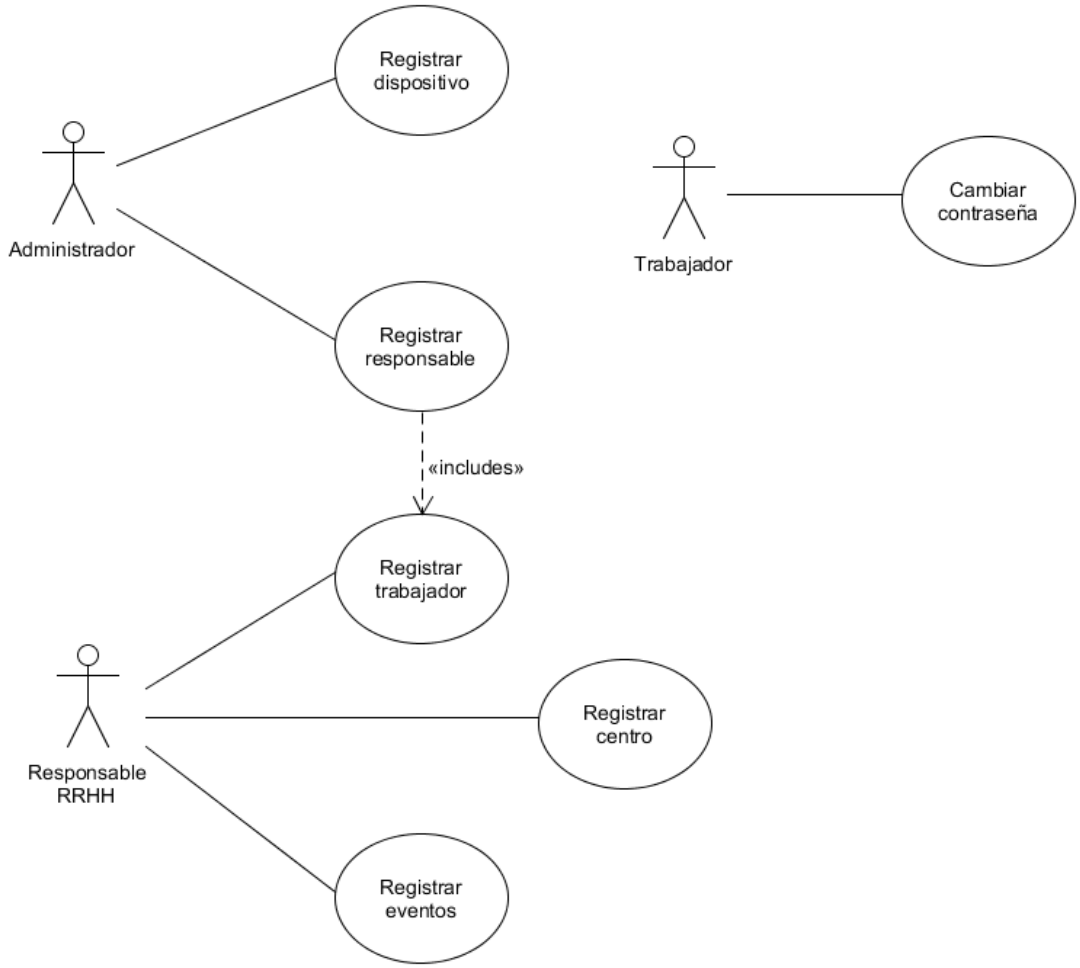

Figura 17. Casos de uso – Configuración del sistema

ID	CU-001
Nombre	Registrar usuario responsable
Flujo Normal	
Actores	Administrador
Precondiciones	Existe un usuario administrador en la base de datos Existe un perfil de usuario “Responsable de RRHH”
Descripción	1. El usuario accede a la aplicación como administrador con su contraseña. 2. Accede a la pantalla de gestión de usuarios y pulsa en *añadir*. 3. Introduce los datos del nuevo usuario y le asigna como cargo “Responsable de RRHH” 4. Guarda los cambios.
Postcondiciones	En el listado de usuarios aparece un nuevo miembro con el cargo “Responsable de RRHH” En la base de datos se ha registrado el nuevo usuario.

ID	CU-002
Nombre	Registrar nuevo dispositivo BLE
Flujo Normal	
Actores	Administrador
Precondiciones	Existe un usuario administrador en la base de datos Existe al menos un centro registrado en la base de datos. Existe al menos un dispositivo BLE detectable por la aplicación.
Descripción	1. El usuario accede a la aplicación como administrador con su contraseña. 2. Accede a la pantalla de gestión de dispositivos BLE. 3. Selecciona uno de las entradas en la lista de nuevos dispositivos y pulsa en *añadir*. 4. Selecciona un centro y la posición del dispositivo (entrada o interior). 5. Guarda los cambios.
Postcondiciones	En el listado de dispositivos registrados aparece el nuevo dispositivo con el nombre del centro asociado. En la base de datos se ha almacenado el nuevo dispositivo.

ID	CU-003
Nombre	Registrar nuevo trabajador
Flujo Normal	
Actores	Responsable de RRHH
Precondiciones	Existe un usuario *Responsable de RRHH* en la base de datos
Descripción	1. El usuario *Responsable de RRHH* accede a la aplicación con su id de usuario y su contraseña. 2. Accede a la pantalla de gestión de usuarios y pulsa en *añadir* nuevo usuario. 3. Introduce los datos del nuevo usuario. 4. Guarda los cambios,
Postcondiciones	En el listado de usuarios aparece el nuevo trabajador registrado. En la base de datos se ha almacenado el nuevo usuario.

ID	CU-004
Nombre	Registrar nuevo centro de trabajo
Flujo Normal	
Actores	Responsable de RRHH
Precondiciones	Existe un usuario *Responsable de RRHH* en la base de datos
Descripción	1. El usuario *Responsable de RRHH* accede a la aplicación con su id de usuario y su contraseña. 2. Accede a la pantalla de configuración y pulsa en *añadir* nuevo centro.

	3. Introduce los datos del nuevo centro. 4. Guarda los cambios,
Postcondiciones	En el listado de centros aparece el nuevo registrado. En la base de datos se ha almacenado el nuevo centro.

ID	CU-005
Nombre	Registrar nuevo evento
Flujo Normal	
Actores	Responsable de RRHH
Precondiciones	Existe un usuario *Responsable de RRHH* en la base de datos
Descripción	1. El usuario *Responsable de RRHH* accede a la aplicación con su id de usuario y su contraseña. 2. Accede a la pantalla de configuración y pulsa en *añadir* nuevo evento. 3. Introduce el nombre del evento, le asigna un color y determina si las horas dedicadas cuentan como laborables. 4. Guarda los cambios,
Postcondiciones	En el listado de eventos aparece el nuevo registrado. En la base de datos se ha almacenado el nuevo evento.

ID	CU-006
Nombre	Cambiar contraseña
Flujo Normal	
Actores	Trabajador
Precondiciones	El trabajador ha accedido a la aplicación con su id y contraseña.
Descripción	1. Accede a la pantalla de gestión de usuarios y pulsa en su perfil. 2. Pulsa en *cambiar contraseña*. 3. Introduce la nueva contraseña y la confirma. 4. Acepta los cambios,
Postcondiciones	En la base de datos se ha registrado la nueva contraseña para el usuario

3.1.2 Notificaciones de entrada/salida

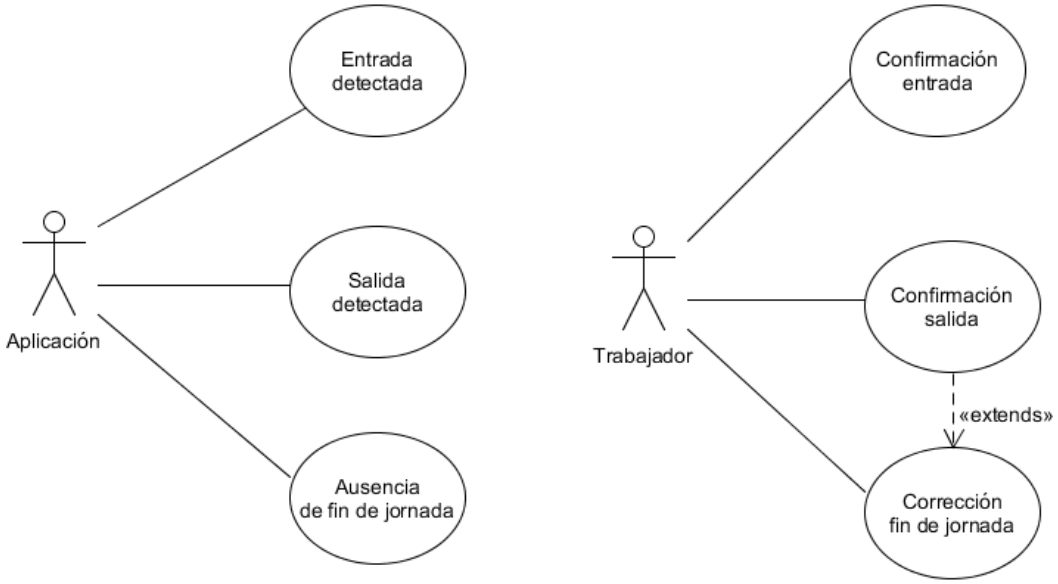

Figura 18. Casos de uso - Notificaciones

ID	CU-007
Nombre	Notificación de entrada detectada
Flujo Normal	
Actores	Aplicación
Precondiciones	El dispositivo móvil del trabajador tiene activado el sensor Bluetooth.
Descripción	1. El trabajador entra en un centro de trabajo y la aplicación detecta el paso por una baliza de entrada asociada a un centro de trabajo. 2. El trabajador accede a su puesto de trabajo y la aplicación detecta el paso por una baliza interior.
Postcondiciones	La aplicación muestra una notificación en el dispositivo móvil para confirmar la entrada. En la base de datos se ha registrado una alerta con la confirmación de la entrada

ID	CU-008
Nombre	Confirmar entrada detectada
Flujo Normal	
Actores	Trabajador
Precondiciones	El trabajador ha recibido en el móvil una notificación de confirmación de entrada.
Descripción	1. El trabajador revisa la notificación, accediendo a la aplicación. 2. En la pantalla de alertas existe un elemento indicando la hora de entrada en el centro y solicitando

	confirmación. 3. El trabajador pulsa en la alerta y accede a la pantalla de creación de nuevo registro de entrada. 4. Se muestra por defecto la hora en la que fue detectado por la baliza de la entrada. 5. El trabajador acepta los datos.
Postcondiciones	En la pantalla de jornada laboral aparece un registro de entrada para el día actual y la hora introducida. La alerta revisada ha desaparecido del listado.

ID	CU-009
Nombre	Notificación de salida detectada
Flujo Normal	
Actores	Aplicación
Precondiciones	El dispositivo móvil del trabajador tiene activado el sensor Bluetooth. Existe un registro de entrada previo para el trabajador en el mismo día.
Descripción	1. El trabajador sale de un centro de trabajo y la aplicación detecta el paso por una baliza de entrada asociada a un centro de trabajo. 2. La aplicación deja de detectar balizas asociadas algún centro.
Postcondiciones	La aplicación muestra una notificación en el dispositivo móvil para confirmar la salida. En la base de datos se ha registrado una alerta con la confirmación de la salida.

ID	CU-010
Nombre	Confirmación de salida detectada
Flujo Normal	
Actores	Trabajador
Precondiciones	El trabajador ha recibido en el móvil una notificación de confirmación de salida.
Descripción	1. El trabajador revisa la notificación, accediendo a la aplicación. 2. En la pantalla de alertas existe un elemento indicando la hora de salida del centro y solicitando confirmación. 3. El trabajador pulsa en la alerta y accede a la pantalla de creación de nuevo registro de salida. 4. Se muestra por defecto la hora en la que fue detectado por la baliza de la entrada. 5. El trabajador selecciona el motivo de la salida. 6. El trabajador acepta los datos.
Postcondiciones	En la pantalla de jornada laboral aparece un registro de salida

	para el día actual y la hora introducida, con el color establecido para el tipo de salida. La alerta revisada ha desaparecido del listado.

ID	CU-011
Nombre	Notificación de ausencia de fin de jornada
Flujo Normal	
Actores	Aplicación
Precondiciones	Existe un registro de entrada para el trabajador y no existe un registro de fin de jornada posterior para el mismo día.
Descripción	1. La aplicación detecta que la fecha ha cambiado.
Postcondiciones	La aplicación muestra una notificación en el dispositivo móvil indicando la ausencia de fin de jornada. En la base de datos se ha registrado una alerta con la ausencia de fin de jornada.

ID	CU-012
Nombre	Corrección de fin de jornada no registrada
Flujo Normal	
Actores	Trabajador
Precondiciones	El trabajador ha recibido en el móvil una notificación de ausencia de fin de jornada.
Descripción	1. El trabajador revisa la notificación, accediendo a la aplicación. 2. En la pantalla de alertas existe un elemento indicando la ausencia de registro de fin de jornada para una fecha concreta. 3. El trabajador pulsa en la alerta y accede a la pantalla de creación de nuevo registro fin de jornada. 4. El trabajador introduce la hora de fin de jornada. 5. El trabajador acepta los datos.
Postcondiciones	En la pantalla de jornada laboral aparece un registro de fin de jornada para el día corregido y la hora introducida. La alerta revisada ha desaparecido del listado.

3.1.3 Gestión manual de la jornada

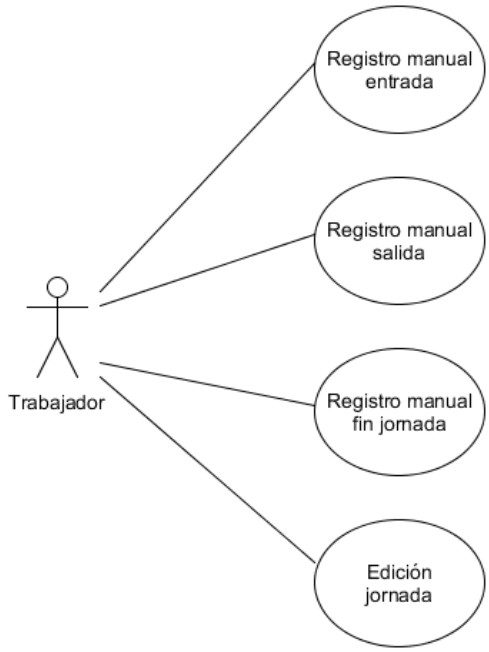

Figura 19. Casos de uso - Gestión manual de jornada

ID	CU-013
Nombre	Registro manual de entrada
Flujo Normal	
Actores	Trabajador
Precondiciones	El trabajador ha accedido a la aplicación con su id de usuario y contraseña. No existe una entrada registrada en el sistema para la misma fecha o el registro anterior se trata de una salida.
Descripción	1. El trabajador accede a la pantalla de registro de jornada laboral. 2. El trabajador selecciona la fecha de registro. 3. Pulsa en *registrar entrada*. 4. En la pantalla de edición, introduce la hora de entrada y acepta los cambios.
Postcondiciones	En la pantalla de jornada laboral aparece un registro entrada para el día seleccionado y la hora introducida. Las alertas de entrada asociadas a la fecha introducida, desaparecen del sistema.

ID	CU-014
Nombre	Registro manual de salida
Flujo Normal	
Actores	Trabajador
Precondiciones	El trabajador ha accedido a la aplicación con su id de usuario

	y contraseña. No existe una salida registrada en el sistema para la misma fecha o el registro anterior se trata de una entrada.
Descripción	1. El trabajador accede a la pantalla de registro de jornada laboral. 2. El trabajador selecciona la fecha de registro. 3. Pulsa en *registrar salida*. 4. En la pantalla de edición, introduce la hora de salida y una razón distinta a *Fin de jornada*. 5. El trabajador acepta los cambios.
Postcondiciones	En la pantalla de jornada laboral aparece un registro salida para el día seleccionado, la hora introducida y con el color correspondiente al motivo introducido. Las alertas de salida asociadas a la fecha introducida, desaparecen del sistema.

ID	CU-015
Nombre	Registro manual de fin de jornada
Flujo Normal	
Actores	Trabajador
Precondiciones	El trabajador ha accedido a la aplicación con su id de usuario y contraseña. No existe una salida de tipo fin de jornada registrada en el sistema para la misma fecha.
Descripción	1. El trabajador accede a la pantalla de registro de jornada laboral. 2. El trabajador selecciona la fecha de registro. 3. Pulsa en *registrar salida*. 4. En la pantalla de edición, introduce la hora de salida y una razón distinta a *Fin de jornada*. 5. El trabajador acepta los cambios.
Postcondiciones	En la pantalla de jornada laboral aparece un registro salida para el día seleccionado, la hora introducida y con el color correspondiente al motivo introducido. Las alertas de salida y ausencia de fin de jornada asociadas a la fecha introducida, desaparecen del sistema.

ID	CU-016
Nombre	Edición de jornada laboral
Flujo Normal	
Actores	Trabajador
Precondiciones	El trabajador ha accedido a la aplicación con su id de usuario y contraseña. Existen registros de entrada y salida para la fecha seleccionada.
Descripción	1. El trabajador accede a la pantalla de registro de jornada

	laboral. 2. El trabajador selecciona la fecha de registro. 3. Pulsa en *editar jornada*. 4. En el listado de eventos, selecciona aquel que desea modificar y pulsa en *editar*. 5. En la pantalla de edición modifica la información y guarda los cambios.
Postcondiciones	En la pantalla de jornada laboral aparece el registro con los datos modificados.

ID	CU-017
Nombre	Eliminar registro de jornada
Flujo Normal	
Actores	Trabajador
Precondiciones	El trabajador ha accedido a la aplicación con su id de usuario y contraseña. Existen registros de entrada y salida para la fecha seleccionada.
Descripción	1. El trabajador accede a la pantalla de registro de jornada laboral. 2. El trabajador selecciona la fecha de registro. 3. Pulsa en *editar jornada*. 4. En el listado de eventos, selecciona aquel que desea modificar y pulsa en *eliminar*.
Postcondiciones	En la pantalla de jornada laboral no aparece el registro eliminado.

3.1.4 Consultas e informes

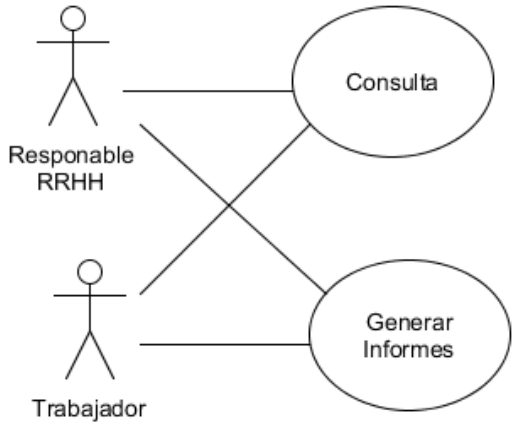

Figura 20. Casos de uso - Consultas e informes

ID	CU-018
Nombre	Consulta de estadísticas
Flujo Normal	
Actores	Trabajador
Precondiciones	El trabajador ha accedido a la aplicación con su id de usuario y contraseña. Existen jornadas laborales completas registradas para el trabajador.
Descripción	1. El trabajador accede a la pantalla consultas. 2. El trabajador selecciona la fecha de inicio y de fin. 3. El trabajador selecciona el centro y el usuario (solo Responsable de RRHH). 4. El usuario pulsa en *resultados*.
Postcondiciones	Se muestra una gráfica con las horas dedicadas en las distintas situaciones para el periodo y usuarios especificados. Aparecen los datos de horas totales trabajadas y horas dedicadas en cada situación.

ID	CU-019
Nombre	Generación de informes
Flujo Normal	
Actores	Trabajador
Precondiciones	El trabajador ha accedido a la aplicación con su id de usuario y contraseña. Existen jornadas laborales completas registradas para el trabajador.
Descripción	1. El trabajador accede a la pantalla informes. 2. El trabajador selecciona el año y mes del informe.

	3. El trabajador selecciona el usuario (solo Responsable de RRHH). 4. El usuario pulsa en *generar*.
Postcondiciones	Se un informe con las horas trabajadas y las entradas y salidas registradas en cada día del mes.

3.2 Diseño de arquitectura

3.2.1 Diseño de base de datos

En este apartado se describen las entidades principales que definen la información almacenada en la base de datos:

- **Company**: contiene la información básica sobre la empresa.
- **Center**: define un centro de trabajo. Cada centro de trabajo tiene asociado una serie de dispositivos BLE.
- **Device**: contiene la información de los dispositivos BLE registrados. Cada dispositivo está asociado a centro. El tipo de dispositivo puede ser de entrada o interior.
- **User**: incluye la información de los usuarios. Cada usuario tiene asociado un listado de alertas y de registros de jornada.
- **Alert**: esta entidad representa las alertas generadas por la aplicación. Contiene un identificador único. Cada alerta se asocia a un centro y un empleado.
- **Registry**: representa cada registro de entrada o salida que realiza el usuario de la aplicación. Contienen un identificador único. Cada registro se asocia a un centro y aun empleado. Además, se asocian con un tipo de evento.
- **Event**: esta entidad define los tipos de eventos registrados en la aplicación. Pueden ser de tipo entrada o salida.

A continuación, se presenta un diagrama entidad/relación que define el esquema de la base de datos y las relaciones existentes entre las entidades.

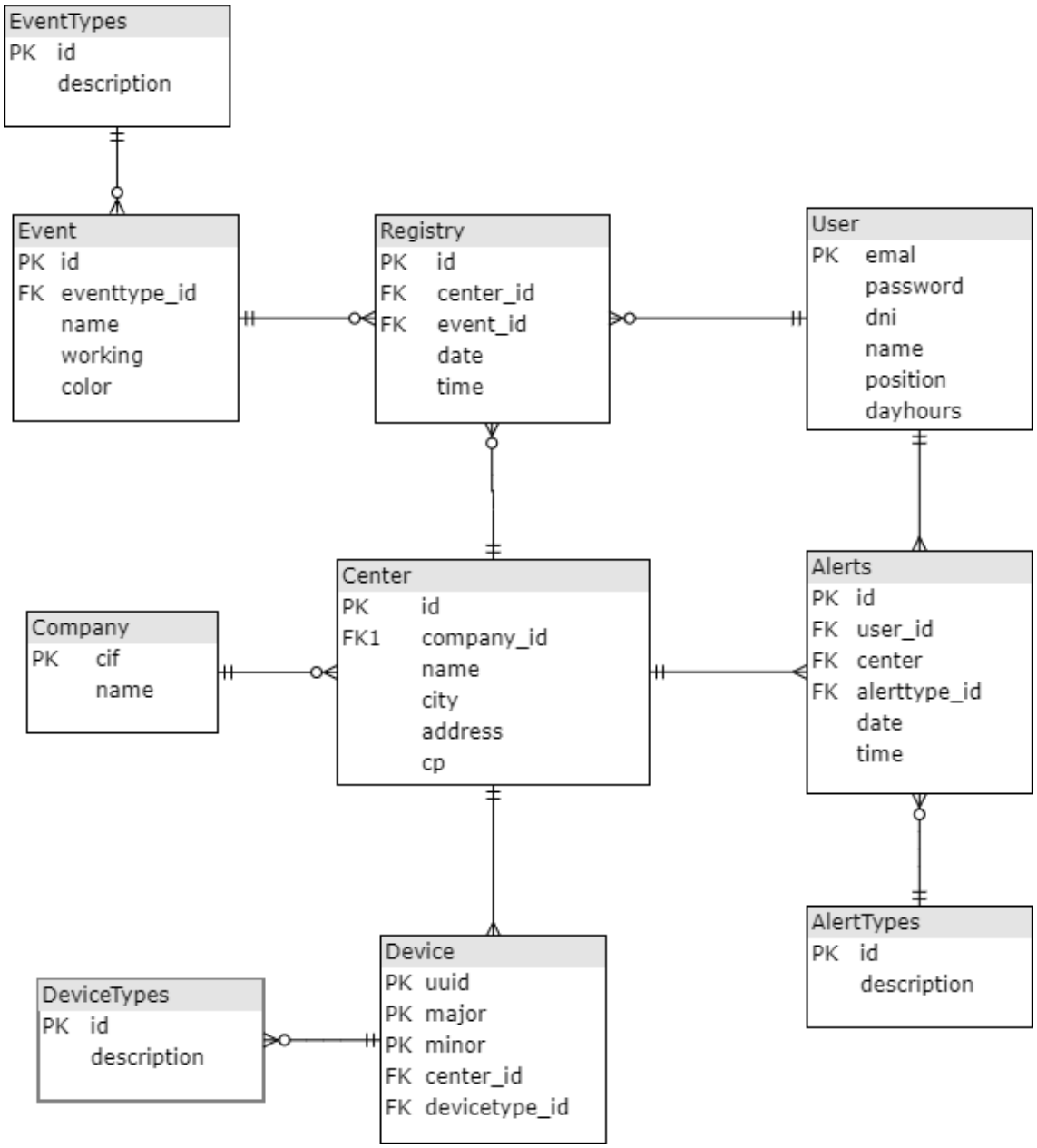

Figura 21. Diagrama base de datos

3.2.2 Diseño orientado a objetos

A partir de aquí, se definen las clases principales que implementan las funcionalidades del producto, sus principales atributos y funciones, así como las relaciones existentes.

Modelo de datos

La capa del modelo representa la información almacenada en la base.

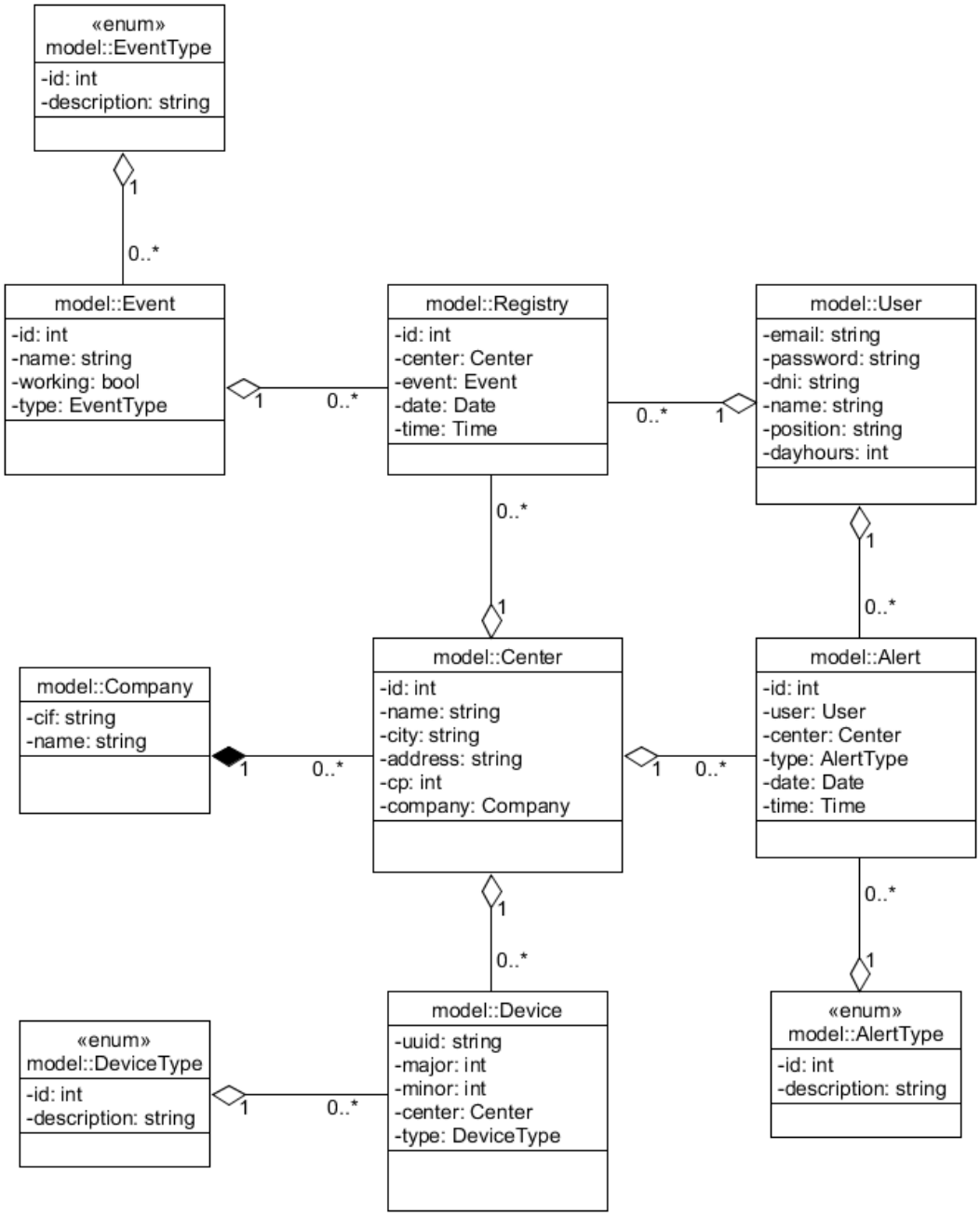

Figura 22. Diagrama de clases – Model

Gestión de datos

El resto de clases del sistema accede a la información del modelo de datos a través de una capa que permite acceder a la información, realizar consultas, añadir elementos y eliminar los ya existentes.

En el siguiente diagrama se muestran algunos ejemplos de clases y operaciones que formarán parte de esta capa.

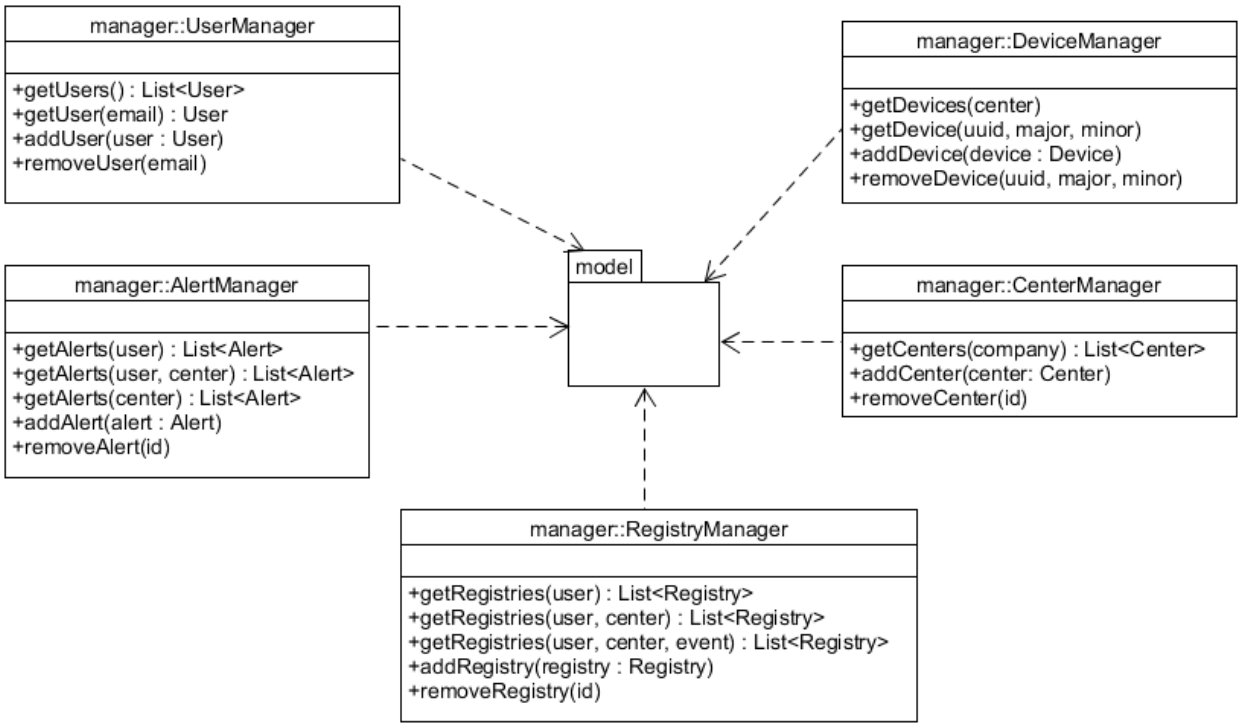

Figura 23. Diagrama de clases - Managers

Servicios

Determinadas funciones son llevadas a cabo por servicios que corren en segundo plano y no dependen de las acciones del usuario.

Como ejemplo, tenemos el servicio encargado de obtener señales procedentes de los dispositivos BLE, procesarlos y responder en consecuencia: registrando una alerta, notificando al usuario o actualizando la pantalla de gestión de dispositivos.

Por otra parte, también existirá un servicio que controla el paso del tiempo, generando alertas cuando se produzcan incoherencias como la ausencia de un fin de jornada.

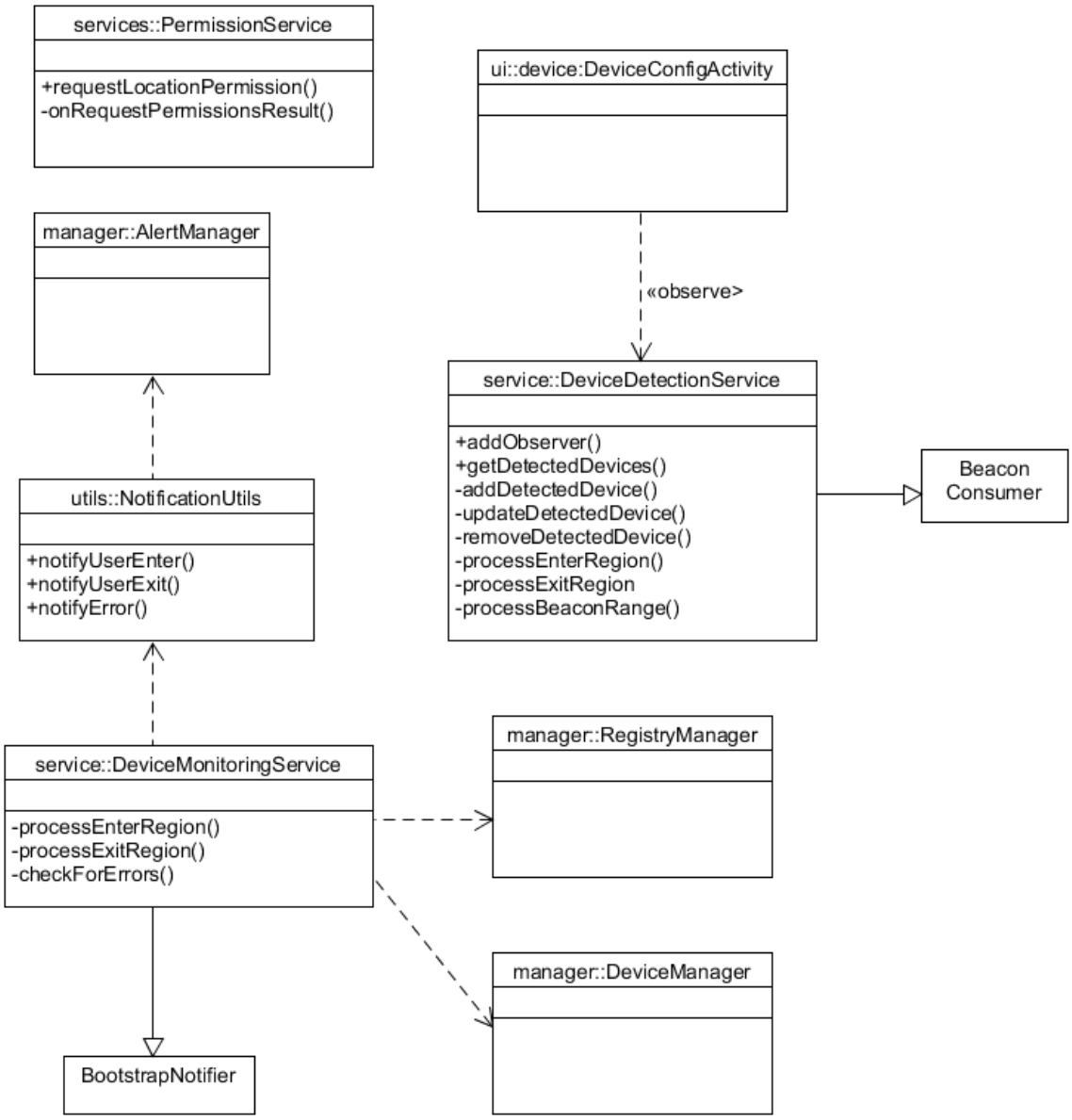

Figura 24. Diagrama de clases – Services

Adaptadores

Para mostrar los datos en la aplicación se utilizarán una serie de clases *adapters* que permiten abstraer el formato de los datos recibidos de su representación en pantalla.

Los tipos de adaptadores específicos que se utilizarán en la aplicación van a depender de los elementos gráficos que se utilicen (layouts). En este punto, se puede presentar un esquema básico de como interactúan con las distintas actividades:

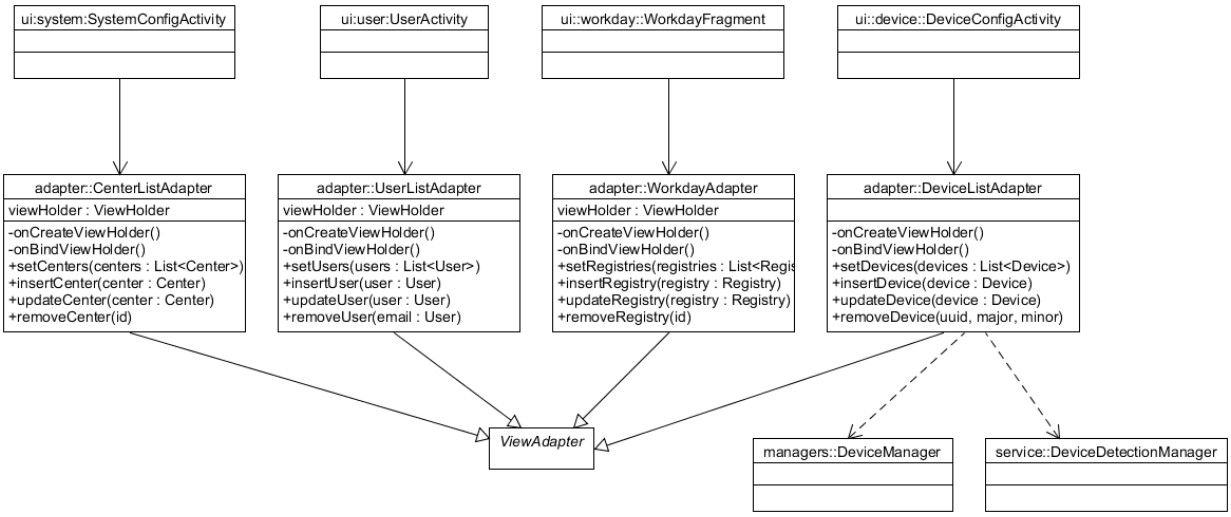

Figura 25. Diagrama de clases - Adapters

Visualización

En esta capa se incluyen todas las clases de tipo *Fragment* y *Activity*. Son responsables de la representación visual de la información y gestionar las acciones del usuario.

En principio, existirá una clase *Activity* por cada pantalla. Éstas, a su vez pueden contener varios fragmentos.

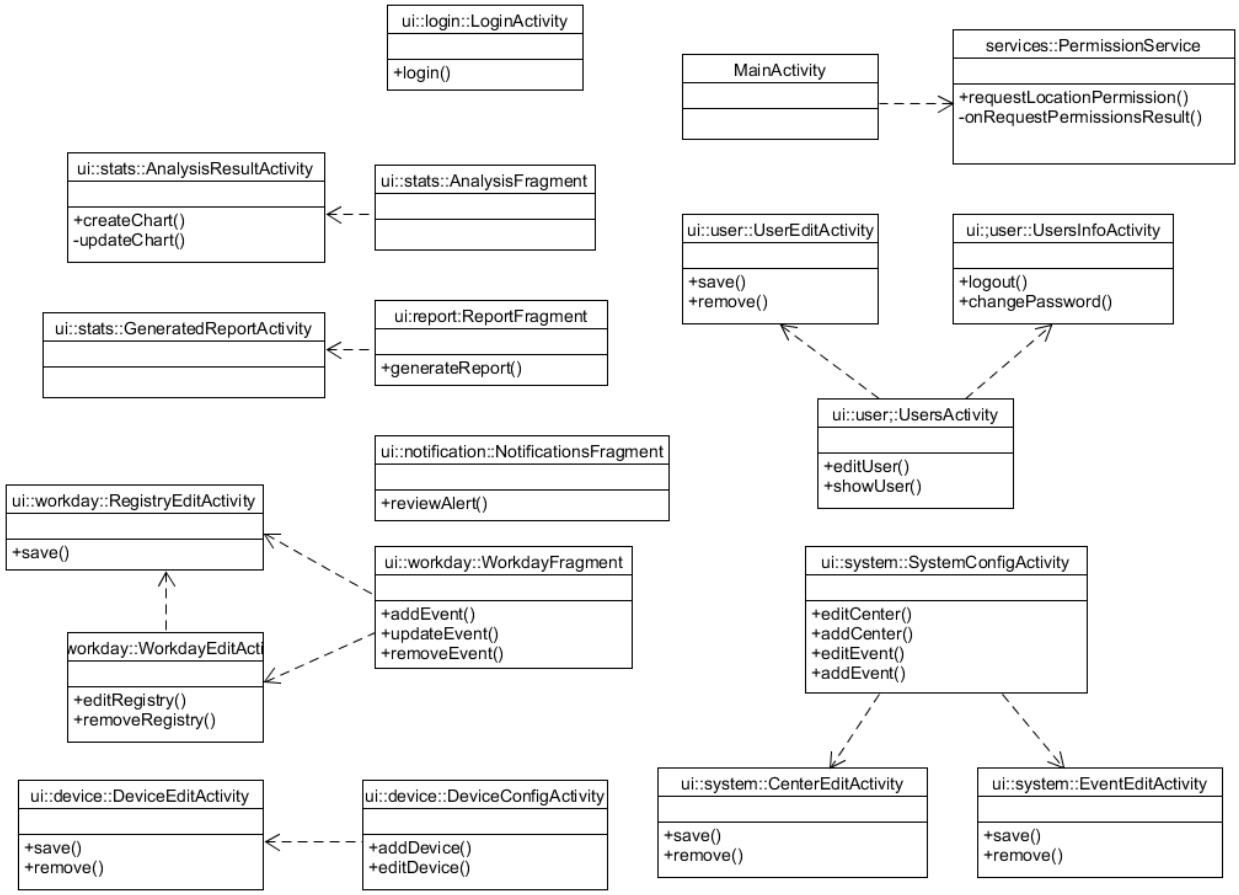

Figura 26. Diagrama de clases - Activities/Fragments

3.2.3 Arquitectura

Como se ha explicado anteriormente, para el diseño de la aplicación se ha seguido una arquitectura MVC (modelo vista controlador). Se ha escogido este tipo de arquitectura por los siguientes motivos:

- El tamaño limitado de la aplicación no invita a utilizar una arquitectura orientada al servicio (SOA).
- El modelo vista-controlador permite un nivel de abstracción suficiente para aceptar cambios y extensiones futuras.
- Se trata de un tipo de arquitectura bien conocida y existen múltiples patrones para llevarla a cabo.

4. Implementación

En este apartado se comentarán los aspectos principales de la implementación de la aplicación. El código se encuentra alojado en un repositorio público en **GitLab**, accesible desde la dirección descrita en [15].

4.1 Estructura del proyecto

Siguiendo el diseño planteado anteriormente, el proyecto se divide en los siguientes paquetes:

model: incluye las clases que representan cada una de las entidades almacenadas en la base de datos.
manager: contiene las clases que permiten manejar los datos representados en el modelo de datos.
adapter: contiene un conjunto de clases que actúan como enlace entre un conjunto de datos y una vista.
ui: contiene las distintas vistas y actividades que definen la interfaz gráfica de la aplicación.
service: contiene las clases que ejecutan acciones de larga duración en segundo plano.

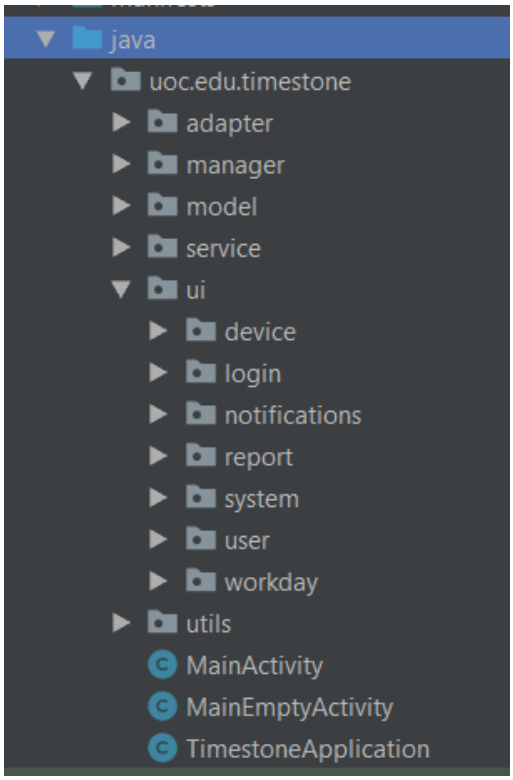

Figura 27. Estructura del proyecto

4.2 Gestión del modelo de datos

Para implementar el modelo de datos descrito anteriormente, se ha utilizado como gestor de base de datos **Firebase Realtime Database** [11].

Cada entidad se representa como una clase POJO (Plain Old Java Object). Por simplicidad, las entidades que definen tipos como DeviceType, EventType y similares han sido implementadas como enumerados dentro de dichas clases.

Para gestionar el almacenamiento, actualización borrado y consultas de cada una de estas entidades, se han creado una serie de manejadores (*Managers*).

Debido a que se trabaja con una base de datos NoSQL y para evitar las lecturas asíncronas, cada gestor mantiene en memoria una copia de los objetos que maneja.

Por ejemplo, la clase **DeviceManager** contiene una referencia a la tabla que la contiene y un listado de los objetos obtenidos.

```
public class DeviceManager implements ChildEventListener {
    private DatabaseReference mRef;
    private List<Device> mDevices = new ArrayList<>();
    static final String kDatabasereference = "devices";
```

En el constructor del manejador se obtiene la referencia a la base de datos. Además, cada manejador implementa un *listener* para los eventos de cambio en la tabla referenciada. Ante cada cambio, se actualizan los objetos locales:

```
public DeviceManager(FirebaseDatabase database) {
    mRef = database.getReference(kDatabasereference);
    mRef.addChildEventListener(this);
}

…

@Override
public void onChildAdded(DataSnapshot dataSnapshot, String s) {
    // Se añade el nuevo elemento a la lista
    mDevices.add(dataSnapshot.getValue(Device.class));
}

@Override
public void onChildChanged(DataSnapshot dataSnapshot, String s) {
   // Actualizar elemento en el listado
   …
}

@Override
public void onChildRemoved(@NonNull DataSnapshot dataSnapshot) {
   // Eliminar elemento en la lista
   …
}
```

Además, es posible añadir una clase que implemente la interfaz *ValueEventListener*, esto permite que los servicios y adaptadores que lo requieran sean notificados ante los cambios que se produzcan en los datos.

```
public void addValueEventListener(ValueEventListener listener) {
    mRef.addValueEventListener(listener);
}
```

Por ejemplo, la clase **DeviceListAdapter** puede notificar los cambios en el listado de dispositivos con las siguientes líneas:

```
public class DeviceListAdapter implements ValueEventListener {

…

  public DeviceListAdapter (Context context, ModeFlag mode) {
    …
    FirebaseDatabase reference = FirebaseDatabase.getInstance();
    mDeviceManager = new DeviceManager(reference);
    …
    mDeviceManager.addValueEventListener(this);
    …
  }

  @Override
  public void onDataChange(DataSnapshot dataSnapshot) {
      this.notifyDataSetChanged();
  }
}
```

Por otra parte, uno de los principales problemas encontrados a la hora de trabajar con una base de datos **NoSql** es la imposibilidad de realizar consultas complejas. Para solventar este problema se ha utilizado la **API Stream** de Java.

Por ejemplo, para obtener los dispositivos de un determinado tipo y que pertenecen a un centro específico utilizamos el siguiente método:

```
public List<Device> getDevicesFor(final String centerId, final
Device.DeviceType type) {

    List<Device> resultList = new ArrayList<>();
    resultList = mDevices.stream()
        .filter(new Predicate<Device>() {
            @Override
            public boolean test(Device d) {
                return d.getCenterId().equals(centerId);
            }
            })
            .filter(new Predicate<Device>() {
                @Override
                public boolean test(Device d) {
                    return d.getType().equals(type);
                }
            })
            .collect(Collectors.<Device>toList());

    return resultList;
}
```

4.3 Detección de dispositivos BLE

Para la detección de dispositivos BLE se utiliza la librería **Android Beacon Library** [6]. En la aplicación existen dos servicios que llevan a cabo la detección de dispositivos BLE.

4.3.1 Monitorización

El primero de ellos, **DeviceMonitoringService**, se encarga de monitorizar las entradas y salidas del trabajador. Para llevar a cabo esta función, este servicio debe estar en funcionamiento todo el tiempo.

Es necesario lanzar el servicio como **Foreground Service**, esto permite que el servicio no sea reciclado por el sistema operativo. Para ello, en primer lugar, es necesario ajustar los permisos en **AndroidManifest.xml**

```
<uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
```

Por otro lado, el sistema se lanzará como una aplicación sin necesidad de lanzar actividad alguna, solo el servicio de monitorización.

```
public class TimestoneApplication extends Application {
…

    @Override
    public void onCreate() {
        super.onCreate();

        Intent intent = new Intent(this, DeviceMonitoringService.class);
        startService(intent);
    }
}
```

El servicio de monitorización implementa la interfaz **BootstrapNotifier**, la cual nos permite monitorizar los dispositivos registrados en segundo plano y ejecutar nuestro código cada vez que se produzca una entrada o salida de una *región*.

```
public class DeviceMonitoringService extends Service implements
BootstrapNotifier, ValueEventListener {
```

Con el siguiente código se inicializa el servicio de detección para balizas de tipo iBeacon.

```
// Initialize beacon monitoring service
mBeaconManager = BeaconManager.getInstanceForApplication(this);
mBeaconManager.getBeaconParsers().add(new BeaconParser().
        setBeaconLayout("m:2-3=0215,i:4-19,i:20-21,i:22-23,p:24-24"));
```

A continuación, se crea un canal de notificaciones necesario para incluir la notificación del servicio permanente. Este es un requisito necesario para lanzar un servicio en primer plano.

```
// Initialize the notification channels for foreground service
Notification.Builder builder = new Notification.Builder(this);
builder.setSmallIcon(R.drawable.ic_launcher_foreground);
builder.setContentTitle(getString(R.string.app_notificaction_title));
Intent toGoIntent = new Intent(this, MainActivity.class);
PendingIntent pendingIntent = PendingIntent.getActivity(
        this, 0, toGoIntent, PendingIntent.FLAG_UPDATE_CURRENT
);

builder.setContentIntent(pendingIntent);

…
  mNotificationChannel = new NotificationChannel(
              TimestoneApplication.NOTIFICATION_ID, "Foreground Notification",
              NotificationManager.IMPORTANCE_DEFAULT);

  mNotificationChannel.setDescription("…");
  NotificationManager notificationManager = (NotificationManager)
            getSystemService(Context.NOTIFICATION_SERVICE);

  notificationManager.createNotificationChannel(mNotificationChannel);
  builder.setChannelId(mNotificationChannel.getId());
}
```

Finalmente, se lanza el servicio de detección en primer plano pasando como parámetro la notificación creada anteriormente:

```
mBeaconManager.enableForegroundServiceScanning(builder.build(), 456);
```

Añadiendo las siguientes líneas podemos ajustar la frecuencia de muestreo en la detección:

```
mBeaconManager.setEnableScheduledScanJobs(false);
mBeaconManager.setBackgroundBetweenScanPeriod(0);
mBeaconManager.setBackgroundScanPeriod(1100);
…
    mBeaconManager.updateScanPeriods();
…
```

Además, con la siguiente línea, es posible ahorrar hasta un 60% de batería consumida en la monitorización:

```
mBackgroundPowerSaver = new BackgroundPowerSaver(this);
```

En este punto tenemos un servicio que permite monitorizar balizas, pero no se le ha especificado ante cuáles de ellas debe reaccionar. Para ello, este servicio se suscribe a los cambios en la base de datos de dispositivos.

Como hemos visto anteriormente, es posible conocer cuando se ha leído la tabla mediante el método *onDataChange*.

```
@Override
public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
    Log.d(TAG, "Device data changed");
    disableForegroundMonitoring();
    enableForegroundMonitoring();
}
```

En este punto, se registran todas las regiones que se desean monitorizar, que se corresponden con los dispositivos registrados:

```
public void enableForegroundMonitoring() {

    // wake up the app when any beacon is seen (you can specify specific id
    // filers in the parameters below)

    int i = 1;
    for (Device device : mDeviceManager.getDevices()) {
        mRegionList.add(new Region("imestone.backgroundRegion_device" + i,
                Identifier.parse(device.getId()),
                Identifier.parse(String.valueOf(device.getMajor())),
                Identifier.parse(String.valueOf(device.getMinor()))));
        i++;
    }

    mRegionBootstrap = new RegionBootstrap(this, mRegionList);
}
```

Una vez hecho esto, los siguientes métodos se ejecutarán cada vez que el usuario entre o salga de una región:

```
@Override
public void didEnterRegion(org.altbeacon.beacon.Region region) {
    Log.d(TAG, "Got a didEnterRegion call for " + region.getId1());
    processEnterRegion(region);
}

…

@Override
public void didExitRegion(org.altbeacon.beacon.Region region) {
    Log.d(TAG, "Got a didExitRegion call for " + region.getId1());
    processExitRegion(region);
}
```

Para procesar las entradas y salidas se incluye una mecánica que diferencia entre dispositivos de entrada e interiores. Esta mecánica se ha añadido pensando en centros de trabajo grandes, en los que una sola baliza no puede cubrir todas las zonas y el usuario estaría recibiendo notificaciones de entrada y salida cada vez que se aleje o acerque a la entrada.

Así, solapando las zonas detección, mientras que el usuario se encuentre dentro de una de ellas, no se enviará una notificación de salida. Por ejemplo, para procesar un evento de salida, se tiene en cuenta si el dispositivo está en la entrada o el interior del edificio y si el trabajador se encuentra en el interior o no.

```
public void processExitRegion(Region region) {
    if (null != region.getId1()) {
        Device device = mDeviceManager.getDevice(region.getId1().toString());
```

```
        if (null != device) {
            Log.d(TAG, "processExitRegion for: " + device.getId());

            if (device.getType().equals(

                Device.DeviceType.DEVICE_TYPE_ENTRANCE)) {

                // Si se detecta una salida por un dispositivo externo y el
                // usuario no está en el interior, entonces se notifica la
                // salida
                if (!mDetectionMark.getInternal()) {

                    // Primero se almacena la notificación en la base
                    // de datos. Ésta se mostrará en la aplicación
                    NotificationUtils.notifyUserExit(
                          MainEmptyActivity.mCurrentUser, device);

                    // También se envía una notificación al dispositivo
                    // que alerte al usuario
                    sendDeviceNotfication(
                        EXIT_NOTIFICATION_ID,  Salida detectada,
                       Calendar.getInstance());
                }
            } else {

                // Si se ha salido de una región interior, se desmarca
                mDetectionMark.setInternal(false);
            }
        }
    }
```

4.3.2 Detección

El servicio de detección se utiliza durante la configuración del sistema. Es necesario detectar cualquier baliza próxima para poder añadirla a nuestro sistema.

El servicio encargado de esta tarea es **DeviceDetectionService**. Su funcionamiento es similar al de monitorización, pero la detección se lanza de forma activa y se deshabilita al finalizar el servicio.

En este caso basta con usar la clase **BeaconManager** e implementar la interfaz **BeaconConsumer**.

```
public class DeviceDetectionService extends Service implements BeaconConsumer
{

@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    …
    // Initialize beacon detection service
    mBeaconManager = BeaconManager.getInstanceForApplication(this);
    mBeaconManager.getBeaconParsers().add(new BeaconParser().
            setBeaconLayout("m:2-3=0215,i:4-19,i:20-21,i:22-23,p:24-24"));

    mBeaconManager.bind(this);

    return super.onStartCommand(intent, flags, startId);
 }
}
```

En este caso, no solo se lanza la monitorización de entradas y salidas, también se registra la distancia. El procesamiento de estos eventos se incluye en el siguiente método:

```
@Override
public void onBeaconServiceConnect() {

  // Monitor de regiones
  mBeaconManager.addMonitorNotifier(new MonitorNotifier() {

    // Método llamado al entrar en una región
    @Override
    public void didEnterRegion(Region region) {
      Log.d(TAG, "Got a didEnterRegion call");
      processEnterRegion(region);
    }

    // Método llamado al salir de una región
    @Override
    public void didExitRegion(Region region) {
        Log.d(TAG, "Got a didDetermineStateRegion call");
        processEnterRegion(region);
    }

    …

  });

  // Monitor de distancia
  mBeaconManager.addRangeNotifier(new RangeNotifier() {

     // Método llamado al calcular la distancia.
     // Se reciben todas las distancias a la vez
     @Override
     public void didRangeBeaconsInRegion(
               Collection<Beacon> collection, Region region) {

            Log.d(TAG, "didRangeBeaconsInRegion with size " +
                 collection.size());

            if (collection.size() > 0) {
                for (Beacon beacon : collection) {
                    processBeaconRange(beacon);
                }
            }
        }
    });

…
    mBeaconManager.startMonitoringBeaconsInRegion(region);
    mBeaconManager.startRangingBeaconsInRegion(region);
…
}
```

A continuación, se muestra cómo se procesa un evento de distancia para una baliza concreta.

```
public void processBeaconRange(Beacon beacon) {
  if (null != beacon.getId1()) {

    Device device = mDeviceManager.getDevice(beacon.getId1().toString());

    // Si el dispositivo no está registrado se actualiza en la lista local
    if (null == device) {
```

```
        device = getDetectedDevice(beacon.getId1().toString());

        // Si el dispositivo ya se encuentra en la lista local,
        // de dispositivos no registrados, se actualiza
        if (null != device) {
            mDeviceContainer.updateDetectedDevice(
                    device.getId(), beacon.getDistance());
        } else { // Otherwise, beacon will be added

            // Si no está en la lista local, se añade
            mDeviceContainer.addDetectedDevice(beacon.getId1().toString(),
                beacon.getId2().toInt(), beacon.getId3().toInt());
        }
    }

    // Si el dispositivo ya está registrado en la base de datos, se elimina
    // de la lista local con los dispositivos no registrados
    else {
        if (null != getDetectedDevice(device.getId())) {
            mDeviceContainer.removeDetectedDevice(device.getId());
        }
    }
}
```

La lista local de dispositivos no registrados puede ser observada y se notificarán los cambios. De esta manera, es posible crear un adaptador sobre esta información:

```
private static DeviceDetectionService.DeviceDetectionContainer
mDeviceContainer =
        new DeviceDetectionService.DeviceDetectionContainer();

…
  static class DeviceDetectionContainer extends Observable {

    public void addDetectedDevice(String uuid, int major, int minor) {
      Device device = new Device(uuid, major, minor,
                              Device.DeviceType.DEVICE_TYPE_ENTRANCE);

      mDetectedDevices.add(device);

      // Se notifica a los observadores
      setChanged();
      notifyObservers();
  }

}
```

4.4 Implementación de vistas

4.4.1 Listados

La mayoría de las vistas de la aplicación han sido implementadas utilizando adaptadores. Como ejemplo, se mostrará el desarrollo de la lista de notificaciones, que sirve de pantalla principal de la aplicación.

En esta ventana se muestran un listado con las notificaciones dirigidas al usuario. La actividad solo contiene la inicialización de la lista, que se encuentra representada con un **RecyclerView** y la creación del adaptador (**NotificationListAdapter**).

```
public class NotificationsFragment extends Fragment {
    private RecyclerView recyclerView;
    private NotificationListAdapter adapter;
    private View root;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {

        root = inflater.inflate(R.layout.fragment_notifications,
                  container, false);

        // Se carga la vista que contendrá la lista
        recyclerView = root.findViewById(
                   R.id.notification_list_recycler_view);

        // Se especifica un adaptador
        adapter = new NotificationListAdapter(root.getContext());
        recyclerView.setAdapter(adapter);
        recyclerView.setLayoutManager(
                  new LinearLayoutManager(root.getContext()));

        return root;
    }
}
```

La clase adaptadora incluye una clase interna de tipo **ViewHolder**, el cual implementa la lógica de visualización de cada item de la lista.

```
public class NotificationListAdapter extends
     RecyclerView.Adapter<NotificationListAdapter.NotificationListViewHolder>
     implements ValueEventListener {

    public static class NotificationListViewHolder
           extends RecyclerView.ViewHolder {
          …
          TextView mFirstLine;
          TextView mSecondLine;
          TextView mThirdLine;
          ImageButton mRemoveButton;

        public NotificationListViewHolder(@NonNull View itemView) {
            super(itemView);
            …
            mFirstLine = itemView.findViewById(R.id.firstLine);
            mSecondLine = itemView.findViewById(R.id.secondLine);
            mThirdLine = itemView.findViewById(R.id.thirdLine);
            mRemoveButton =
                itemView.findViewById(R.id.removeNotificationButton);
    }
}
```

Finalmente, con el método **onBindViewHolder** se carga la información de cada elemento de la vista.

```
@Override
public void onBindViewHolder(
     Final NotificationListAdapter.NotificationListViewHolder holder,
```

```
     int position) {

    // Para el usuario responsable de recursos humanos, solo se visualizarán
    // las alertas correspondientes a errores.
    if (MainEmptyActivity.mCurrentUser.getPosition().equals("RRHH")) {
        final Alert alert = mAlertManager.getAlertsForType(
                Alert.AlertType.ALERT_TYPE_ERROR).get(position);

        if (alert.getType().equals(Alert.AlertType.ALERT_TYPE_ERROR)) {
            holder.mSecondLine.setText("ERROR");
            holder.mThirdLine.setText("Descripción del error");
        }

    } else {

     // Para el resto de usuarios, se mostrarán todas las notificaciones
     // dirigidas al mismo
     final Alert alert = mAlertManager.getAlertsForUser(
            MainEmptyActivity.mCurrentUser.getId()).get(position);

         // En la primera línea se muestra la fecha
         holder.mFirstLine.setText(alert.getDate());


        // En las siguientes líneas se muestra un texto u otro dependiendo
        // de si es una notificación de entrada o de salida
        if (alert.getType().equals(Alert.AlertType.ALERT_TYPE_ENTER)) {
             holder.mSecondLine.setText(
                 mContext.getString(R.string.notification_enter_title));

             holder.mThirdLine.setText(
                 mContext.getString(R.string.notification_enter_description)
                      + " " + alert.getTime());

        } else if (alert.getType().equals(Alert.AlertType.ALERT_TYPE_EXIT)) {

          …

        } else if (alert.getType().equals(Alert.AlertType.ALERT_TYPE_ERROR)) {
          …
        }

    // Se establece el manejador para el botón de borrado
    holder.mRemoveButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            mAlertManager.removeAlert(alert);
        }
    });

    // Se establece las acciones necesarias al pulsar el ítem
    // En este caso, se lanza una actividad para crear un nuevo registro
    // con los datos obtenidos de la alerta revisada.
    holder.mView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent activityIntent =
                 new Intent(mContext, RegistryEditActivity.class);

            // Se establece el tipo de edición (confirmación de alerta)
            activityIntent.putExtra(
                     RegistryEditActivity.EDIT_TYPE,
                     RegistryEditActivity.EditType.CONFIRM_EVENT);

            // Se establece el tipo de registro (entrada o salida)
            if(alert.getType().equals(Alert.AlertType.ALERT_TYPE_ENTER)) {

                activityIntent.putExtra(
```

```
                    RegistryEditActivity.EVENT_TYPE,
                    Event.EventType.EVENT_TYPE_ENTER);

            } else {

                activityIntent.putExtra(
                    RegistryEditActivity.EVENT_TYPE,
                       Event.EventType.EVENT_TYPE_EXIT);
            }

            // Finalmente, se serializa la notificación seleccionada
            activityIntent.putExtra(
                   RegistryEditActivity.NOTIFICATION_EXTRA, alert);

            // Y se lanza la actividad
            mContext.startActivity(activityIntent);
        }
    });
}
```

El número de elementos a representar en la lista se obtiene implementando el siguiente método:

```
@Override
public int getItemCount() {
    if (MainEmptyActivity.mCurrentUser.getPosition().equals("RRHH")) {

        // Si el usuario es el responsable de recursos humanos, solo
        // se tienen en cuenta las notificaciones de error.
        return mAlertManager.getAlertsForType(
               Alert.AlertType.ALERT_TYPE_ERROR).size();

    } else {

        // En caso contrario se tendrán en cuenta todos los registros
        // existentes para el usuario
        return mAlertManager.getAlertsForUser(
            MainEmptyActivity.mCurrentUser.getId()).size();
    }
}
```

4.4.2 Timeline

En la ventana de gestión de la jornada laboral se ha introducido una línea de tiempo que representa las entradas y salidas del trabajador, marcándolas con el color asociado al evento.

Para implementarlo se ha utilizado la librería **TimelineView** [12]. Para crearla, se añade una vista de tipo **ListView** al layout y se genera el contenido a través de un adaptador.

Para rellenar los datos del adaptador se utiliza el siguiente método en la clase **WorkdayFragment**:

```
public void createTimeline(String dateString, String userId) {

    // Se crea un listado que contendrá las filas creadas
    ArrayList<TimelineRow> timelineRowsList = new ArrayList<>();
```

```
    int i = 0;
    Date date = mCurrentDate.getTime();

    // Se crea una nueva fila por cada registro existente para el usuario
    // actual y la fecha establecida
    for (Registry registry :
            mRegistryManager.getRegistriesFor(userId, dateString)) {

        Event event = mEventmanager.getEvent(registry.getEventId());

        if (null != event) {

            // Se crea la nueva fila
            TimelineRow myRow = new TimelineRow(i);

            // Se establece la fecha como título
            myRow.setTitle(registry.getTime());

            // Se establece el nombre del evento como descripción
            myRow.setDescription(event.getName());

            // Se establece el color del evento
            myRow.setBackgroundColor(Color.parseColor(event.getColor()));

            // Se establece el tamaño del fondo
            myRow.setBackgroundSize(40);

            // Se establece el color de las líneas que unen las filas
            myRow.setBellowLineColor(Color.argb(255, 0, 0, 0));

            // Se añade la fila a la lista
            timelineRowsList.add(myRow);
        }
    }

    // Se crea el adaptador usando la lista de filas creada
    ArrayAdapter<TimelineRow> myAdapter = new TimelineViewAdapter(
            root.getContext(), 0, timelineRowsList, false);

    // Se obtiene la vista y se establece el adaptador
    ListView myListView = root.findViewById(R.id.timeline_listView);
    myListView.setAdapter(myAdapter);

    // Se notifica a todos lo elementos que los datos han cambiado
    myAdapter.notifyDataSetChanged();
}
```

4.4.3 Cálculo de horas

En la generación del informe, es necesario calcular las horas trabajadas para cada día. Para ello, el adaptador de informe (**ReportListAdapter**) utiliza la clase **WorkingHoursUtils** para obtener el número de horas trabajadas utilizando un listado de registros.

```
public long getTimeInRegistryList(List<Registry> list) {

    long result = 0;

    // Se obtiene la hora del primer elemento de la lista
    Date lastDate = Calendar.getInstance().getTime();
    if (!list.isEmpty()) {
```

```
        try {
            lastDate = DateTimeUtils.toTimestamp(
                    list.get(0).getDate(), list.get(0).getTime());
        } catch (ParseException e) {
            return result;
        }
    }

    // Entonces, se realiza la diferencia entre la hora de cada nuevo evento
    // con la del evento anterior y el resultado se añade al total calculado
    for (int i = 1; i < list.size(); i++) {

        try {
            Event event = mEventManager.getEvent(
                    list.get(i - 1).getEventId());

            Date newDate = DateTimeUtils.toTimestamp(
                    list.get(i).getDate(), list.get(i).getTime());

            // Si el evento es de tipo no laborable, el tramo no será
            // considerado en la suma total
            if (null != event && event.isWorking()) {
                result += Math.abs(newDate.getTime() - lastDate.getTime());
            }
            lastDate = newDate;

        } catch (ParseException e) {
            return result;
        }
    }

    // Finalmente, se devuelve el total en milisegundos
    return result;
}
```

El listado que se pasa como parámetro al método, se obtiene al filtrar por usuario y fecha. Este proceso se repite para cada día del mes.

```
List<Registry> registryList =
         mRegistryManager.getRegistriesFor(mUserId, dateString);
```

4.4.4 Estadísticas de tiempo

La funcionalidad de consultas permite mostrar un gráfico con la distribución del tiempo de los usuarios entre los distintos tipos de eventos registrados. Para la generación de gráficos se utiliza la librería **MPAndroidChart** [13].

La generación y actualización de los datos del gráfico se lleva a cabo en la clase **PieChartAdapter**. Almacena los datos en una estructura que contiene el identificador del evento, el color definido y el número de horas dedicado al mismo. Además, se almacena el número total de horas desempeñadas en el rango de tiempo introducido por el usuario.

```
public class EntryData extends Observable {
    private HashMap<String, Pair<Long, Integer>> mEventMap = new HashMap<>();
    private long mTotalTime;

    public HashMap<String, Pair<Long, Integer>> getEventMap() {
        return mEventMap;
```

```
    }

    public long getTotalTime() {
        return mTotalTime;
    }

    public void setTotalTime(long totalTime) {
        this.mTotalTime = totalTime;
    }

    public void setUpdated() {
        setChanged();
        notifyObservers();
    }
}
```

La clase **AnalysisResultActivity** se suscribe como observador a los cambios en los datos anteriores:

```
mChartAdapter = new PieChartAdapter(mCenter.getId(), mUser.getId(),
        mFromDate, mUntilDate);

mChartAdapter.addObserver(this);

…

@Override
public void update(Observable o, Object arg) {
    updateChart(mChartAdapter.getEntryData().getEventMap(),
            mChartAdapter.getEntryData().getTotalTime());
}
```

La actividad crea un nuevo gráfico al inicializar:

```
// Creación del gráfico
chart = findViewById(R.id.worktimePieChart);
createChart();

…

public void createChart() {
    chart.setUsePercentValues(true);
    chart.getDescription().setEnabled(false);
    chart.setDrawHoleEnabled(false);
    chart.setExtraOffsets(5, 10, 5, 5);
    chart.setDragDecelerationFrictionCoef(0.95f);
    chart.setRotationAngle(0);
    chart.setRotationEnabled(true);
    chart.setHighlightPerTapEnabled(true);
    chart.animateY(1400, Easing.EaseInOutQuad);
    chart.setEntryLabelColor(Color.WHITE);
    chart.setEntryLabelTextSize(12f);
}
```

Finalmente, se actualizan los datos del gráfico con la nueva información recibida:

```
private void updateChart(HashMap<String, Pair<Long, Integer>> timeMap,
             long total) {

    ArrayList<PieEntry> entries = new ArrayList<>();
    ArrayList<Integer> colors = new ArrayList<>();
    ArrayList<LegendEntry> legendEntries = new ArrayList<>();
```

```
    // Se recorren todos los eventos
    Iterator<String> itr = timeMap.keySet().iterator();
    while (itr.hasNext()) {

        // Se obtiene el evento y el número de horas correspondiente
        String key = itr.next();
        Long value = timeMap.get(key).first;
        Event event = mEventManager.getEvent(key);

        if (null != event) {

            // Con la información se crea una nueva entrada
            if (total != 0) {
                entries.add(new PieEntry((float) ((value * 100) / total)));
            } else {
                entries.add(new PieEntry((float) 0));
            }

            // Se establece el color de cada evento
            colors.add(Color.parseColor(event.getColor()));

            // Se establece el valor para la leyenda para cada evento
            int seconds = (int) (value / 1000) % 60;
            int minutes = (int) ((value / (1000 * 60)) % 60);
            int hours = (int) ((value / (1000 * 60 * 60)) % 24);

            …

            Legend.LegendForm.SQUARE, Float.NaN, Float.NaN,
                        null, Color.parseColor(event.getColor())));
        }
    }

    // Se establece el nuevo conjunto de datos
    PieDataSet dataSet = new PieDataSet(entries, "Horas dedicadas");
    dataSet.setDrawIcons(false);
    dataSet.setSliceSpace(3f);
    dataSet.setIconsOffset(new MPPointF(0, 40));
    dataSet.setSelectionShift(5f);
    dataSet.setColors(colors);

    PieData data = new PieData(dataSet);
    data.setValueFormatter(new PercentFormatter(chart));
    data.setValueTextSize(11f);
    data.setValueTextColor(Color.WHITE);
    chart.setData(data);

    // Se establecen los valores para la leyenda
    Legend l = chart.getLegend();
    l.setVerticalAlignment(Legend.LegendVerticalAlignment.TOP);
    l.setHorizontalAlignment(Legend.LegendHorizontalAlignment.LEFT);
    l.setOrientation(Legend.LegendOrientation.VERTICAL);
    l.setDrawInside(false);
    l.setXOffset(5f);
    l.setXEntrySpace(10f);
    l.setYEntrySpace(5f);
    l.setYOffset(5f);
    l.setEntries(legendEntries);
    l.setTextSize(14f);

    …
}
```

4.5 Generación de informe

Una vez calculadas las horas trabajadas y la entrada y salida para un periodo determinado, es posible volcar dicha información en un documento PDF y compartirla usando cualquier aplicación disponible en el móvil.

Para ello, se ha incluido un botón para compartir dentro de la pantalla que muestra los informes generados (**GeneratedReportActivity**). Esta función utiliza métodos de la clase estática **ReportUtils**.

Para generar el documento en PDF se ha utilizado la librería **iTextpdf** [14].

```
public static String generateReport(User user, String year, String month,
            long worktime, Long[] daysWorktime, String[] entranceTime,
            String[] exitTime) throws IOException, DocumentException {

  // Se crea el documento PDF en una determinada ruta
  String path = android.os.Environment.
                  getExternalStorageDirectory().toString();

  Document document = new Document();

  // Se utiliza el año, el mes y el nombre de usuario para nombrar el fichero
  PdfWriter.getInstance(document,
            new FileOutputStream(path + "/" + user.getName() +
            user.getSurname1() + user.getSurname2() + "_" + year +
            "_" + month + ".pdf"));

  document.open();

  // Se establece el autor, la fecha de creación y el título del documento
  document.addAuthor("Timestone");
  document.addCreationDate();
  document.addTitle("Informe " + user.getName() + " " + user.getSurname1() +
             " " + user.getSurname2() + " " + year + " " + month);

  // A continuación se crea la cabecera del documento
  document.add(createHeader(user, year, month, worktime));
```

La cabecera del documento incluye los datos del usuario, el periodo temporal que abarca y el número total de horas trabajadas.

```
private static PdfPTable createHeader(User user, String year,
                  String month, long worktime) {

  …

  // Se crea una tabla con dos columnas con el usuario, el mes y año, año
  // como las horas trabajadas.

  PdfPTable table = new PdfPTable(2);
  table.addCell(user.getName() + " " + user.getSurname1() + " " +
               user.getSurname2());
  table.addCell("DNI:" + user.getId());
  table.addCell(month + " " + year);
  table.addCell("Horas: " + hours + ":" + minutesString + ":" +
               secondsString);
  table.setSpacingAfter(10);

    return table;
}
```

Para cada día del mes se genera una tabla con las horas trabajadas en el día y la hora de entrada y salida del trabajador.

```
private static PdfPTable createDayEntry(int year, int month, int day,
                  Long workhours,  String entrance, String exit) {

  // Se crea una tabla con dos columnas
  PdfPTable table = new PdfPTable(2);

  …

  // Se añade la fecha y las horas trabajadas a la nueva entrada
  table.addCell(dateString);
  table.addCell("Horas:" + hours + ":" + minutesString + ":" + secondsString);

  // Se añade una fila con la fecha de entrada
  PdfPCell cell1 = new PdfPCell(new Phrase("Entrada:" + entrance));
  table.addCell(cell1);

  // Se añade una fila con la fecha de salida
  PdfPCell cell2 = new PdfPCell(new Phrase("Salida:" + exit));
  table.addCell(cell2);

  table.setSpacingAfter(10);

  return table;
}
```

Finalmente, el documento generado se comparte utilizando un selector de aplicaciones.

```
case R.id.action_share:
  try {
    // Se genera el documento PDF con los parámetros calculados
    String path = ReportUtils.generateReport(mUser, mYear, mMonth, mWorktime,
                adapter.getDaysWorktime(), adapter.getEntranceTime(),
                adapter.getExitTime());

    // Se encía el documento utilizando un Intent selector
    Intent shareIntent = new Intent();
    File file = new File(path);

    if (file.exists()) {
        shareIntent.setType("application/pdf");
        shareIntent.putExtra(Intent.EXTRA_STREAM,
                             Uri.parse("file://" + path));
        shareIntent.putExtra(Intent.EXTRA_SUBJECT, "Sharing File...");
        startActivity(Intent.createChooser(shareIntent, "Share File"));
  }
```

5. Pruebas

Para probar el sistema se han utilizado dos enfoques distintos. Por un lado, se han elaborado un conjunto de pruebas automáticas para probar el funcionamiento de la base de datos y los cálculos y operaciones complejas llevadas a cabo con dichos datos. Por otra parte, se ha elaborado un procedimiento de pruebas manual para certificar la adecuación de la aplicación a los requisitos previamente definidos.

5.1 Pruebas automáticas

Para probar el funcionamiento del modelo de datos, así como las clases que gestionan el almacenamiento y consulta de información se han creado una serie de pruebas instrumentadas.

Como preparación de la prueba, se crean todos los *Managers* necesarios añadiendo la base *test* a la referencia de la base de datos. Esto se hace así para que las pruebas no contaminen los datos de producción.

```
public class DatabaseInstrumentedTest {

  @Before
  public void createManagers() {
    // Con esta línea se limpian los datos de prueba
    mDatabase.getReference("test").removeValue();

    mUserManager = new UserManager(mDatabase, "test");
    mCompanyManager = new CompanyManager(mDatabase, "test");
    mCenterManager = new CenterManager(mDatabase, "test");
 }
```

El primer test almacena un nuevo usuario y posteriormente comprueba que el usuario existe en la base de datos.

```
@Test
public void test1() {

    mUserManager.addUser("555555E", "user.name@company.es",
            "password", "John", "Doe", "Foo",
            "Analista", 8);

    sleep(2000);

    User user1 = mUserManager.getUser("555555E");
    if (user1 != null) {
        Assert.assertEquals(user1.getId(), "555555E");
    } else {
        Assert.fail("User was null");
    }
}
```

También se han creado pruebas automáticas que permiten detectar errores en los cálculos realizados por la aplicación referentes a estadísticas e informes.

Antes de ejecutar los test, se generan los registros correspondientes a un año de trabajo para un usuario determinado.

```
public void generateData() {

  // En primer lugar, se añaden datos base: usuarios, centro, tipos de eventos
  mUserManager.addUser("444444E", "jane_doe@company.es", "password",
                       "Jane", "Doe", "", "RRHH", 8);
  mUserManager.addUser("555555E", "john_doe@company.es", "password",
                      "John", "Doe", "", "Worker", 8);
  …

  // Para un año determinado, se generan registros para cada mes y día del año
  int year = 2020;
  for (int i = 0; i < 12; i++) {
      Calendar myCal = Calendar.getInstance();
      myCal.set(year, i, 1, 0, 0, 0);
      int daysOfMonth = myCal.getActualMaximum(Calendar.DAY_OF_MONTH);

      for (int j = 0; j < daysOfMonth; j++) {

          myCal.set(year, i, j + 1, 0, 0, 0);
          String dateString = DateTimeUtils.toDateString(myCal.getTime());

          mRegistryManager.addRegistry(mCenterManager.
                    getCenterByName("Central"),
                    mEventManager.getEvent(EventManager.ENTER_EVENT_ID),
                    mUserManager.getUser("555555E"), dateString, "09:00:00",
                    true);
  …
```

Con los datos generados, es posible comprobar, por ejemplo, el cálculo de horas totales para un mes determinado.

```
  @Test
  public void calculateMonthHours() {
    WorkingHourUtils workingHourUtils =
                     new WorkingHourUtils(mRegistryManager, mEventManager);

    long total = workingHourUtils.calculateWorkTime("2020",
                              "Enero", "555555E");
    long hours = (((total / 1000) / 60) / 60);
    Assert.assertEquals((8 * 31), hours);
  }
```

5.2 Pruebas de sistema

El procedimiento de pruebas definido en el **Anexo A**, permite comprobar el cumplimiento de requisitos definidos para el sistema.

5.3 Incidencias

Como resultado de la ejecución de las pruebas definidas en los apartados anteriores, se ha obtenido el conjunto de errores listados en el **Anexo B**.

6. Conclusiones

Tras el trabajo realizado, consideramos que se han logrado la mayoría de los objetivos que se plateaban inicialmente. Se ha conseguido crear una aplicación que permite automatizar, hasta cierto punto, el control de la jornada laboral de los trabajadores dentro de una empresa.

Por otra parte, otro de los objetivos era explorar el concepto de control de presencia basado en dispositivos Bluetooth de baja intensidad. En este sentido, se ha conseguido crear una base de trabajo utilizando un conjunto de herramientas software y hardware que permiten adaptar el sistema a distintas necesidades futuras.

Aun así, hemos detectado aspectos que deberían mejorar para considerar la aplicación como un producto completo que cumpla las necesidades de una empresa u organización.

Por el contexto del trabajo, nos hemos centrado en el desarrollo de una aplicación para dispositivos Android. Debido a ello, existen partes del desarrollo que, en un producto final, entendemos, deberían recaer en el lado del servidor.

Una posible mejora sería la introducción de una serie de servicios REST que ejecuten las consultas complejas y los cálculos necesarios para elaborar informes y estadísticas, recibiendo la aplicación los resultados.

Este punto anterior ha sido tenido en cuenta de tal manera que la arquitectura de la aplicación se ha diseñado pensando en este tipo de modificaciones. La gestión de datos queda encapsulada en la capa de control. La sustitución de la base de datos actual resultaría relativamente sencilla de ejecutar.

Las entrevistas con los usuarios potenciales y la evaluación del diseño han sido también una fuente de ideas y nos ha permitido detectar una serie de mejoras que convertirían el trabajo actual en un producto completamente viable para su explotación en empresas.

Entre estas mejoras se encuentran: la posibilidad de importar un calendario laboral con los días laborables del año, permitir a los trabajadores solicitar y registrar las vacaciones o permitir adjuntar documentos como justificante de ausencias y salidas.

Con el conocimiento adquirido sobre localización en interiores se pueden plantear evoluciones de la aplicación como la posibilidad de utilizar los elementos disponibles para monitorizar la ubicación de los trabajadores en cada momento. Con una adecuada disposición de los dispositivos BLE, y con la debida parametrización de estos, sería posible aprovechar la información de distancia para establecer la ubicación por zonas dentro de instalaciones de gran tamaño.

Esto puede resultar útil a en trabajos donde resulte necesario tener ubicados a los trabajadores para localizarlos en circunstancias determinadas como, por ejemplo, médicos dentro de un hospital o vigilantes de seguridad.

7. Glosario

IOT:	Internet of Things
BLE:	Bluetooth Low Energy
DCU:	Diseño Centrado en el Usuario
R.R.H.H	Recursos Humanos
MVC	Modelo Vista Controlador
SOA	Service Oriented Arquitecture
POJO	Plain Old Java Object

8. Bibliografía

[1] Control Laboral. “Sistema de control de horas laborales y jornadas de empleados”. controllaboral.es [artículo en línea]. [Fecha de consulta: 21/09/2019].
<https://www.controllaboral.es/>

[2] Timenet. “Sistema de control de horario, registro de la jornada laboral y gestión de proyectos”. registrojornadalaboral.es [artículo en línea]. [Fecha de consulta: 21/09/2019].
<https://www.registrojornadalaboral.es/es>

[3] Trackpeople. “La aplicación para el registro horario adaptada a la nueva ley”. trackpeople.es [artículo en línea]. [Fecha de consulta: 21/09/2019].
<https://trackpeople.es/>

[4] Programa de control de presencia. “Un software fácil y completo para el registro de jornada de empleados de todo tipo de empresas, oficinas y comercios”. programadecontroldepresencia.es [artículo en línea]. [Fecha de consulta: 21/09/2019].
<https://programadecontroldepresencia.es/superior/>

[5] Timepro. “Control de presencia laboral. Su app para el control horario”. timepro.es [artículo en línea]. [Fecha de consulta: 21/09/2019].
<https://www.timepro.es/>

[6] Radius Networks. “Android Beacon Library. An Android library providing APIs to interact with Beacons”. github.com [artículo en línea]. [Fecha de consulta: 21/09/2019].
<https://altbeacon.github.io/android-beacon-library/documentation.html>

[7] Iberley (2019, 5 de abril)."Obligatoriedad de registro de la jornada de los trabajadores". Iberley.es [artículo en línea]. [Fecha de consulta: 25/09/2019].
<https://www.iberley.es/temas/obligatoriedad-registro-jornada-trabajadores-61342>

[8] Avvel. “Avvel X, USB, Short and Long Range Guides”. Avvel.co.uk [artículo en línea]. [Fecha de consulta: 02/10/2019].
<https://www.avvel.co.uk/beacon-guides-1>

[9] Ramírez Vique, Robert. Boltà Torrell, Helena. “Métodos aplicados al desarrollo de aplicaciones móviles”. Tecnología y desarrollo en dispositivos móviles. Métodos para el desarrollo de aplicaciones móviles. FUOC.
PID_00246016

[10] Flamarich Zampalo, Jordi. “Fase de conceptualización”. Diseño de productos interactivos multidispositivo. Conceptualización. FUOC.
PID_00245395

[11] Google, “Firebase Realtime Database”. Documentación – Guías. Google LLC. [Fecha de consulta: 28/10/2019].
<https://firebase.google.com/docs/database>

[12] Github, “Customizable Timeline View for Android”. Library repository.
[Fecha de consulta: 6/12/2019].
<https://github.com/qapqap/TimelineView>

[13] Github, “MPAndroidChart - A powerful & easy to use chart library for Android”, Library repository. [Fecha de consulta: 11/12/2019]
<https://github.com/PhilJay/MPAndroidChart>

[14] Github, “ITextPdf – A powerful PDF Toolkit for PDF generation”, Library repository. [Fecha de consulta: 15/12/2019]
<https://github.com/itext/itextpdf>

[15] GitLab, “Timestone”, Project repository. [Fecha de consulta: 03/01/2020]
<https://gitlab.com/mcodacuoc/timestone>

9. Anexos

Anexo A: Procedimiento de pruebas

ID		TP-001
Requisitos		CU-001, CU-003, CU-006
Preparación		
Con el dispositivo móvil conectado, ejecutar la aplicación DataGenerationTest.		
Ningún usuario ha accedido previamente o el anterior ha salido de la aplicación		
Procedimiento		
#	**Paso**	**Resultado esperado**
1	Abrir la aplicación	Se muestra la pantalla de Login.
2	Rellenar los campos con los siguientes valores: Email: error_user@company.es Password: error Pulsar botón ACCEDER	La aplicación muestra un mensaje de error indicando que el usuario o la contraseña no son correctos.
3	Rellenar los campos con los siguientes valores: Email: jane_doe@company.es Password: password Pulsar botón ACCEDER	El usuario se registra correctamente y se muestra la pantalla principal de la aplicación. En la parte superior derecha se muestran botones para configurar el sistema y los usuarios.
4	Pulsar el botón de configuración de usuarios	Se muestra la lista de usuarios registrados en el sistema.
5	Pulsar sobre el botón de edición del usuario John Doe	Se muestra una ventana para editar la información del usuario John Doe. Los campos muestran la información almacenada actualmente para el usuario.
6	Ir a la pantalla anterior y pulsar el botón para añadir un usuario	Se muestra una ventana para introducir la información del nuevo usuario. Los campos están en blanco.
7	Pulsar el botón de guardado	Se muestra un mensaje de error indicando que es necesario rellenar todos los campos.
8	Rellenar los campos con la siguiente información: Email: moises.coda@company.es DNI: 777777H Nombre: Moisés Apellido1: Coda Apellido2: Cabeza de Vaca Puesto: Desarrollador Horas: 8 Pulsar el botón de guardado	Se muestra el listado de usuario con un nuevo elemento correspondiente a la información introducida.
9	En el listado de usuarios, pulsar en el elemento correspondiente a Jane Doe	Se muestra la información del usuario. Se muestran los siguientes botones: CAMBIAR CONTRASEÑA SALIR
10	Pulsar el botón CAMBIAR CONTRASEÑA	Se muestra un diálogo que permite cambiar la contraseña.
11	Introducir los siguientes valores y pulsar	Se muestra un mensaje de error indicando que

	ACEPTAR Contraseña actual: error Nueva contraseña: pass Confirmar contraseña: pass	la contraseña actual o la confirmación no son correctas
12	Pulsar el botón CAMBIAR CONTRASEÑA	Se muestra un diálogo que permite cambiar la contraseña.
13	Introducir los siguientes valores y pulsar ACEPTAR Contraseña actual: password Nueva contraseña: pass Confirmar contraseña: pass	Se muestra la pantalla de Login
14	Rellenar los campos con los siguientes valores: Email: jane_doe@company.es Password: pass Pulsar botón ACCEDER	El usuario se registra correctamente y se muestra la pantalla principal de la aplicación. En la parte superior derecha se muestran botones para configurar el sistema y los usuarios.
15	Pulsar el botón de configuración de usuarios	Se muestra la lista de usuarios registrados en el sistema.
16	Pulsar en el elemento de la lista correspondiente a Jane Doe	Se muestra la información del usuario.
17	Pulsar el botón SALIR	Se muestra la pantalla de Login
18	Rellenar los campos con los siguientes valores: Email: john_doe@company.es Password: password Pulsar botón ACCEDER	El usuario se registra correctamente y se muestra la pantalla principal de la aplicación. En la parte superior derecha se muestra solamente un botón para la configuración de usuarios.
19	Pulsar el botón de configuración de usuarios	Se muestra la información del usuario registrado. Se muestran los siguientes botones: CAMBIAR CONTRASEÑA SALIR

ID		TP-002
Requisitos		CU-002
Preparación		
Con el dispositivo móvil conectado, ejecutar la aplicación DataGenerationTest.		
Ningún usuario ha accedido previamente o el anterior ha salido de la aplicación.		
Utilizando la aplicación Beacon Simulator, simular un dispositivo de tipo iBeacon.		
El dispositivo móvil de prueba tiene el sensor Bluetooth activo		
Procedimiento		
#	**Paso**	**Resultado esperado**
1	Abrir la aplicación	Se muestra la pantalla de Login.
2	Rellenar los campos con los siguientes valores: Email: admin@company.es Password: admin Pulsar botón ACCEDER	El usuario se registra correctamente y se muestra la pantalla principal de la aplicación. En la parte superior derecha se muestran botones para configurar los usuarios y los dispositivos BLE.
4	Pulsar el botón de configuración de dispositivos	Se muestra un listado con los dispositivos registrados y otro con los dispositivos detectados no registrados.

5	(cont)	El listado de dispositivos registrados contiene un elemento.
6	(cont)	El listado de dispositivos detectados contiene al menos un elemento con el dispositivo simulado.
7	Pulsar en el botón añadir correspondiente al dispositivo simulado detectado.	Se muestra la ventana de edición de dispositivos. Se muestra un selector con los centros disponibles.
8	Seleccionar un centro disponible e Interior como tipo. Pulsar en el botón de guardado.	Se muestra el listado de dispositivos. El dispositivo detectado se ha añadido al listado de dispositivos registrados y finalmente, desaparece del listado de dispositivos detectados.
9	Pulsar el botón de edición sobre el nuevo dispositivo registrado. En la pantalla de edición, pulsar el botón para eliminar.	Se muestra el listado de dispositivos. El dispositivo eliminado vuelve a aparecer solo en el listado de dispositivos detectados.

ID		TP-003
Requisitos		CU-004, CU-005
Preparación		
Con el dispositivo móvil conectado, ejecutar la aplicación DataGenerationTest.		
Ningún usuario ha accedido previamente o el anterior ha salido de la aplicación.		
Procedimiento		
#	**Paso**	**Resultado esperado**
1	Abrir la aplicación	Se muestra la pantalla de Login.
2	Rellenar los campos con los siguientes valores: Email: jane_doe@company.es Password: password Pulsar botón ACCEDER	El usuario se registra correctamente y se muestra la pantalla principal de la aplicación. En la parte superior derecha se muestran botones para configurar el sistema y los usuarios.
4	Pulsar el botón de configuración del sistema	Se muestra un listado con los centros registrados y otro con los eventos registrados.
5	Pulsar el botón de edición de uno de los centros	Se muestra una ventana para la edición del centro. Los campos contienen los valores establecidos previamente. Los eventos de inicio y fin de jornada no son editables.
6	Modificar el código postal y pulsar el botón de guardado.	En el listado se muestra la información modificada.
7	Pulsar el botón para añadir un centro	Se muestra una ventana para la edición del centro. Los campos están en blanco.
8	Completar los campos con valores válidos. Pulsar el botón de guardado	El nuevo centro se ha añadido a la lista.
9	Pulsar el botón de edición para el nuevo centro y en la ventana de edición, pulsar el botón de borrado.	El nuevo centro desaparece del listado.
10	Pulsar el botón de edición de uno de los eventos.	Se muestra una ventana para la edición del evento. Los campos contienen los valores establecidos previamente.
11	Modificar el color del evento y pulsar en el botón de guardado.	En el listado se muestra la información modificada.
12	Pulsar el botón para añadir un evento	Se muestra una ventana para la edición del evento. Los campos están en blanco.
13	Completar los campos con valores válidos. Pulsar el botón de guardado	El nuevo evento se ha añadido a la lista.
14	Pulsar el botón de edición para el nuevo	El nuevo evento desaparece del listado.

	evento y en la ventana de edición, pulsar el botón de borrado.	

ID		TP-004
Requisitos		CU-007, CU-008, CU-009, CU-010
Preparación		
Con el dispositivo móvil conectado, ejecutar la aplicación DataGenerationTest.		
Ningún usuario ha accedido previamente o el anterior ha salido de la aplicación.		
Utilizando la aplicación Beacon Simulator, simular un dispositivo de tipo iBeacon.		
El dispositivo móvil de prueba tiene el sensor Bluetooth activo		
El dispositivo simulado se encuentra registrado como dispositivo de entrada		
Procedimiento		
#	**Paso**	**Resultado esperado**
1	Abrir la aplicación	Se muestra la pantalla de Login.
2	Rellenar los campos con los siguientes valores: Email: john_doe@company.es Password: password Pulsar botón ACCEDER	El usuario se registra correctamente y se muestra la pantalla principal de la aplicación. En la parte superior derecha se muestran botones para configurar el sistema y los usuarios.
3	Activar el dispositivo de entrada simulado	En el área de notificaciones del dispositivo, aparece una alerta indicando una posible entrada con la hora actual.
4	Revisar la notificación recibida.	Se abre la aplicación en la ventana de notificaciones mostrando en el listado un elemento correspondiente a la entrada detectada.
5	Pulsar en el elemento en la lista	Se abre la ventana de edición de registro. Los campos contienen la fecha, la hora y el centro de la entrada detectada.
6	Pulsar en ACEPTAR	La notificación de entrada ha sido eliminada del listado.
7	Desactivar el dispositivo de entrada simulado	En el área de notificaciones del dispositivo, aparece una alerta indicando una posible salida con la hora actual.
8	Revisar la notificación recibida.	Se abre la aplicación en la ventana de notificaciones mostrando en el listado un elemento correspondiente a la salida detectada.
9	Pulsar en el elemento en la lista	Se abre la ventana de edición de registro. Los campos contienen la fecha, la hora y el centro de la salida detectada.
10	Seleccionar *Descanso* como tipo de salida y pulsar en ACEPTAR	La notificación de salida ha sido eliminada del listado.
11	Acceder a la ventana de Jornada laboral	En el *timeline* aparece un registro de *Entrada* y un registro de *Descanso* con las horas introducidas.

ID	TP-005
Requisitos	CU-013, CU-014, CU-015, CU-016, CU-017
Preparación	
Con el dispositivo móvil conectado, ejecutar la aplicación DataGenerationTest.	
Ningún usuario ha accedido previamente o el anterior ha salido de la aplicación.	

Procedimiento		
#	**Paso**	**Resultado esperado**
1	Abrir la aplicación	Se muestra la pantalla de Login.
2	Rellenar los campos con los siguientes valores: Email: john_doe@company.es Password: password Pulsar botón ACCEDER	El usuario se registra correctamente y se muestra la pantalla principal de la aplicación. En la parte superior derecha se muestran botones para configurar el sistema y los usuarios.
3	Acceder a la sección de Jornada laboral	Se muestran una ventana con los botones REGISTRAR ENTRADA, REGISTRAR SALIDA y EDITAR JORNADA
4	Pulsar el botón REGISTRAR ENTRADA	Se muestra una ventana para registrar una entrada. Los campos contienen la fecha y la hora actual.
5	Pulsar ACEPTAR	Se muestra la ventana de jornada laboral y la línea de tiempo muestra una nueva entrada con la hora introducida.
6	Pulsar el botón REGISTRAR SALIDA	Se muestra una ventana para registrar una salida. Los campos contienen la fecha y la hora actual
7	Seleccionar *Comida* como tipo de salida y pulsar ACEPTAR	Se muestra la ventana de jornada laboral y la línea de tiempo muestra una nueva salida de tipo *Comida* con la hora introducida.
8	Pulsar el botón REGISTRAR SALIDA	Se muestra una ventana para registrar una salida. Los campos contienen la fecha y la hora actual
9	Seleccionar *Fin de jornada* como tipo de salida y pulsar ACEPTAR	Se muestra la ventana de jornada laboral y la línea de tiempo muestra una nueva salida de tipo *Fin de jornada* con la hora introducida.
10	Pulsar el botón EDITAR JORNADA	Se muestra un listado con todos los registros del día
11	Pulsar el botón de edición de la entrada de tipo *Comida*	Se muestra la ventana de edición de registro con los valores actuales.
12	Modificar el tipo a *Salida laboral* y pulsar el botón de guardado	Se muestra el listado de registros. El valor de la entrada ahora es *Salida laboral*
13	Pulsar el botón de borrado de la entrada *Salida laboral*	La entrada desaparece del listado.

ID	TP-006
Requisitos	CU-018
Preparación	
Con el dispositivo móvil conectado, ejecutar la aplicación DataGenerationTest.	
Ningún usuario ha accedido previamente o el anterior ha salido de la aplicación.	
Procedimiento	
#	**Paso**	**Resultado esperado**
1	Abrir la aplicación	Se muestra la pantalla de Login.
2	Rellenar los campos con los siguientes valores: Email: john_doe@company.es Password: password Pulsar botón ACCEDER	El usuario se registra correctamente y se muestra la pantalla principal de la aplicación. En la parte superior derecha se muestran botones para configurar el sistema y los usuarios.
3	Ir a la sección de consultas	Se muestra una ventana para seleccionar el centro, la fecha de inicio y la fecha de fin.

4	Introducir los siguientes valores y pulsar en RESULTADOS: Centro: Todos Desde: 01/01/2020 Hasta: 31/12/2020	Se muestra una gráfica con la distribución del tiempo del empleado para el periodo definido. El valor Entrada ocupa el 88,2 % El valor Comida ocupa el 5,9 % El valor Descanso ocupa el 5,9 %

ID		TP-007
Requisitos		CU-019
Preparación		
Con el dispositivo móvil conectado, ejecutar la aplicación DataGenerationTest.		
Ningún usuario ha accedido previamente o el anterior ha salido de la aplicación.		
Procedimiento		
#	**Paso**	**Resultado esperado**
1	Abrir la aplicación	Se muestra la pantalla de Login.
2	Rellenar los campos con los siguientes valores: Email: john_doe@company.es Password: password Pulsar botón ACCEDER	El usuario se registra correctamente y se muestra la pantalla principal de la aplicación. En la parte superior derecha se muestran botones para configurar el sistema y los usuarios.
3	Ir a la sección de informes	Se muestra una ventana para seleccionar el año y mes del informe.
4	Introducir los siguientes valores y pulsar en GENERAR: Año: 2020 Mes: Enero	Se muestra una ventana con el informe mensual. La cabecera muestra el nombre, apellidos y DNI del usuario actual, así como el año y mes seleccionados.
5	(cont)	En la cabecera se muestra un número de horas totales de 248.
6	(cont)	El informe muestra una entrada por cada día del mes con la fecha y el número de horas trabajadas (8h).
7	(cont)	Cada entrada muestra la hora de entrada y la hora de fin de jornada.
8	Pulsar el botón para compartir informe	Se muestra un selector de aplicaciones con la que compartir el informe.
9	Seleccionar la aplicación de correo	Se genera un correo con un fichero PDF adjunto con el nombre JohnDoe_2020_Enero.pdf

Anexo B: Listado de errores

#1 Error en el cálculo de horas mensuales — **ERROR**
El número de horas calculadas para un usuario determinado no es correcto. Este número se utiliza para los informes generados. — **RESUELTO**

#2 Porcentajes incorrectos en los gráficos — **ERROR**
Los valores de porcentajes que se muestran en el gráfico son incorrectos. Los valores de cada entrada deben ser acumulativos. — **RESUELTO**

#3 El diálogo de cambio de contraseña no comprueba la validez de los datos — **ERROR**
El diálogo de cambio de contraseña no comprueba la validez de los datos introducidos. Es necesario comprobar que la contraseña actual se corresponde con la registrada. Además, es necesario comprobar que la confirmación de contraseña es correcta. — **ABIERTO**

#4 La pantalla de edición de dispositivos muestra el botón de borrado con dispositivos nuevos. — **ERROR**
La pantalla de edición de dispositivos muestra el botón de borrado cuando se trata de un nuevo dispositivo. Al pulsar el botón en este caso se produce una caída de la aplicación — **ABIERTO**

#5 Caída de la aplicación al añadir un nuevo dispositivo interior — **ERROR**
Al añadir un nuevo dispositivo de tipo Interior, se produce una caída. — **RESUELTO**

#6 El botón de retroceso no funciona en ninguna actividad — **ERROR**
El botón de retroceso no funciona en ninguna de las actividades que lo contienen — **RESUELTO**

#7 Es necesario comprobar errores cada vez que se genera el informe — **MEJORA**
El proceso de comprobación de errores se lleva a cabo cada 12 horas y solo se comprueba el día anterior. Sería de ayuda que se compruebe los registros del mes entero cada vez que se genera un informe. — **RESUELTO**

#8 Las ventanas de informe y análisis muestran siempre la etiqueta Usuario — **ERROR**
Cuando el usuario no es el responsable de recursos humanos, estas ventanas no deben mostrar el selector de usuarios y tampoco la etiqueta de ese campo. — **RESUELTO**

#9 Gestión de solapamiento de diferentes dispositivos de regiones internas — **MEJORA**
Resulta útil mantener poder definir diferentes regiones internas solapadas dentro de un mismo centro. El sistema notificará la salida del trabajador únicamente no sea detectado dentro de ninguna de las regiones internas. — **RESUELTO**

#10 Error en el número de horas totales mostrado en la leyenda del gráfico — **ERROR**
El gráfico de horas muestra el número total de horas para cada evento. El número de horas siempre es mayor que el que debería ser. — **RESUELTO**

#11 El gráfico no muestra las etiquetas en la leyenda adecuadamente — **ERROR**
El gráfico muestra las horas de trabajo como “Entrada”. Debería mostrar “trabajo” — **RESUELTO**