Aplicación móvil para la planificación personal basada en la disponibilidad laboral

Nombre Estudiante: Daniel Díaz De La Iglesia
Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles

Nombre Consultor/a: Eduard Martin Lineros
Profesor/a responsable de la asignatura: Carles Garrigues Olivella

Junio de 2017

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Aplicación móvil para la planificación personal basada en la disponibilidad laboral.*
Nombre del autor:	*Daniel Díaz De La Iglesia*
Nombre del consultor/a:	*Eduard Martín Lineros*
Nombre del PRA:	*Carles Garrigues Olivella*
Fecha de entrega (mm/aaaa):	06/2017
Titulación::	Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles
Idioma del trabajo:	*castellano*
Palabras clave	*Planificación, disponibilidad, conciliación, gamificación*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

La finalidad de este TFM es la de crear una aplicación móvil, que, apoyada en la ludificación o gamificación, convierta la tarea de registrar las entradas y salidas al puesto de trabajo en un reto atractivo que merezca la pena.

Con la aplicación se podrá avisar de forma automática a los contactos deseados (familiares, compañeros de trabajo, amigos) de los eventos de salida o entrada pudiendo enviarles mensajes personalizados.

La comunicación de estos eventos con amigos o familiares, permitirá además ayudar a conciliar la vida laboral con la personal.

Abstract (in English, 250 words or less):

The purpose of this project is to develop a mobile application based on gamification that makes recording the entry and exit time from the office a worthwhile attractive challenge.

The application will make possible to automatically notify the desired contacts (family, co-workers, friends) the exit or entry events and send personalized messages to them.

Communicating these events to friends or family will also help to reconcile personal and working life

Índice

1 . Introducción

1.1 CONTEXTO Y JUSTIFICACIÓN DEL TRABAJO

La utilización teléfonos inteligentes, es amplia y ya no sólo se usa como un mero dispositivo desde el que hacer llamadas o enviar mensajes, si no que hace las funciones de un ordenador que el usuario lleva consigo y que puede disponer de el y de la información que proporciona internet de forma inmediata en cualquier lugar y momento. El usuario ya no tiene que sentarse delante de su ordenador para obtener información de internet, si no que en cualquier ubicación, por la calle, en el trabajo, viajando en transporte público, puede aprovechar cualquier momento para acceder de forma inmediata, y lo que es más importante, interactuar y tomar decisiones en base a la información recibida.

La proliferación del uso de teléfonos inteligentes ha ido de la mano con la evolución de los sistemas operativos específicos para móviles, pues está evolución permitió, mediante los correspondientes SDK's dar la posibilidad a desarrolladores a crear aplicaciones y ofrecerlas en un mercado global. Se puede considerar los primeros Nokia, que disponían de funciones avanzadas, tales como GPS, correo electrónico, etc como una generación inicial de lo que hoy son los smartphones, con una posibilidad muy limitada de desarrollo de aplicaciones y sin un mercado global de distribución de aplicaciones. Una vez que compañías como Google o Apple ofrecieron al mercado sus soluciones de sistema operativo móvil y una plataforma de distribución global, como por ejemplo Google Play, propiciaron la creación de aplicaciones, que generaron una necesidad y que fueron accesibles a la población mediante las plataformas propias de distribución de aplicaciones. Acciones como la realizada en 2010 por Blackberry en la que regalaban una tablet a cualquier desarrollador que simplemente publicase una aplicación en su plataforma, demuestran la importancia de tener una masa crítica de aplicaciones para poder ser apetecible de cara al cliente final.

El usuario final lo que demanda es disponer de una serie de servicios que puedan dar solución a sus necesidades particulares, por muy sectoriales o concretas que estas sean. Un ejemplo de este tipo de aplicaciones es “Gasolineras de España”, aplicación que mediante la geolocalización, permite a los usuarios de España tener acceso a las gasolineras más baratas cercanas a él.

En el ámbito laboral, y de una forma más intensa en los que ejercen su profesión por cuenta ajena, los trabajadores deben realizar un número concreto de horas semanales. El trabajador necesita conocer las horas de trabajo que ha realizado, para en función de la hora de entrada, salir más tarde o más temprano en el propio día, o recuperar horas durante la semana si en días previos ha realizado menos horas de las habituales.

Es por ello que una de las necesidades a cubrir es tener información de las horas realizadas y las horas pendientes de realizar. Normalmente para ello en las empresas hay aplicaciones de control de horario, pero bien por desconfianza, o bien porque en la empresa en la que trabaja no existe, el usuario suele llevar su propio control en engorrosas hojas de cálculo.

Además, en función de esa información adquirida, se suelen realizar una serie de acciones manuales y repetitivas que se pretende automatizar:

- En función de la hora en la que se entra, ya se conoce la hora de salida, con lo que se puede informar a familiares, amigos, etc de la hora prevista de salida para poder organizar actividades, y por lo tanto ayudar a conciliar mejor la vida personal con la laboral.
- Al llegar a la oficina, avisar a compañeros bien para tener una reunión previa de trabajo o para hablar de temas personales justo antes de empezar a trabajar.

El propósito de este trabajo es facilitar y automatizar acciones repetitivas diarias, y con la información obtenida poder ayudar a conciliar mejor la vida personal y laboral tanto dentro del trabajo como fuera.

1.2 OBJETIVOS DEL TRABAJO

Los pilares fundamentales de la aplicación a realizar son:

- Simplificar el proceso de llevar la contabilidad de las horas realizadas y que no sea algo tedioso sino un proceso ameno.
- Ayudar a conciliar la vida personal con la familiar, indicando tanto disponibilidad en el trabajo, como para poder organizar eventos fuera de el.

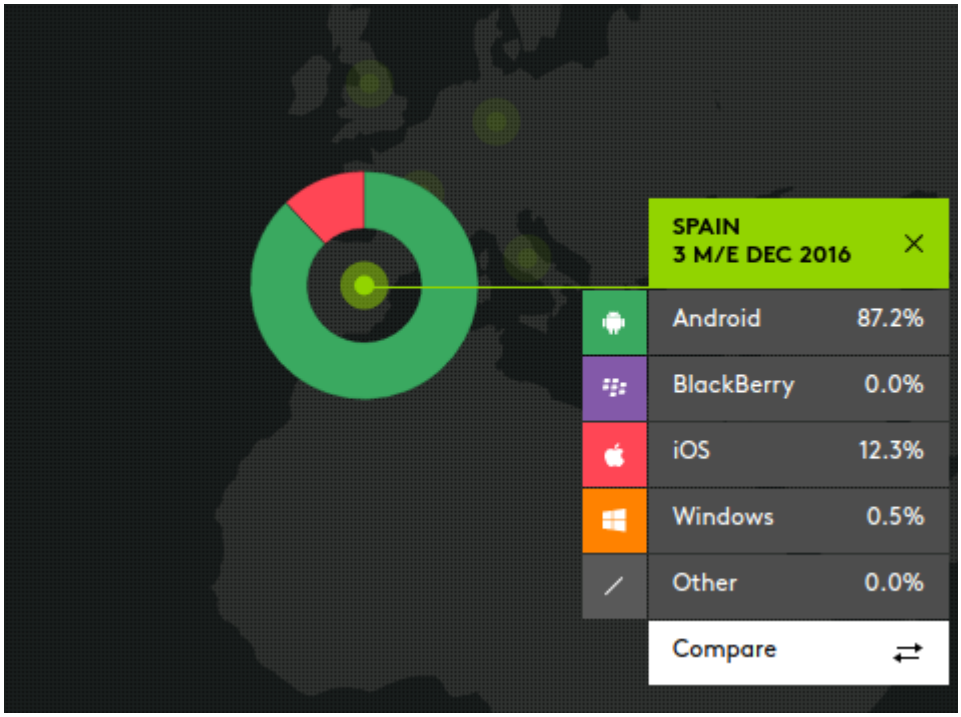
Para ello, la aplicación a desarrollar debería implementar las siguientes funciones:

- Registro sencillo de entrada y salida en el trabajo.
- Registro de las horas pendientes de realizar y en base a ello dar información concreta:
  - A qué hora se puede salir.
  - Si ya se han realizado las horas previstas para el día, avisar de que ya se puede ir del lugar de trabajo.
- Creación de acciones automáticas con mensajes personalizados:
  - Avisar a grupos concretos de cuando se entra a trabajar.
  - Notificar cuándo se pretende salir para organizar de forma fácil eventos.
- Funcionalidades avanzadas:
  - Mediante tecnología NFC, marcar automáticamente entrada y salida al tocar en una etiqueta NFC configurada.

1.3 ENFOQUE Y MÉTODO SEGUIDO

En base a las necesidades indicadas anteriormente, se detecta la necesidad de crear una aplicación nueva que venga a resolver, mediante la ayuda de la tecnología, todas aquellas tareas repetitivas que se realizan y a contribuir a mejorar la conciliación laboral.

Se escoge crear una aplicación basada en Android, sin denostar las aplicaciones para dispositivos iOS. El motivo de la elección es cubrir la mayoría de los dispositivos móviles. Android no está restringido a ningún fabricante y el lenguaje de programación sobre el que se sustenta, Java, es ampliamente conocido y está formado por una amplia comunidad.



Fuente 1: https://www.kantarworldpanel.com/global/smartphone-os-market-share/

Como se puede ver en el panel de Kantar [4], en España, en el mes de diciembre de 2016, el uso de Android alcanzaba la cifra de un 87’2%, frente al 12’3 de iOS. En el mercado internacional la cifra es algo menor, pero se garantiza crear un producto destinado a un grupo amplio de usuarios. No es objeto decidir qué tecnología o sistema operativo es el mejor, sino presentar a la mayoría del mercado el producto con un coste de creación comedido. En función del éxito del producto, si convence a la mayoría del mercado que está en Android, podría crearse versiones concretas para el resto de sistemas operativos para dispositivos móviles.

1.4 PLANIFICACIÓN DEL TRABAJO

Para la realización de este TFM, se empleará como metodología de desarrollo SCRUM [5], que está enfocada en el desarrollo ágil de productos y que está recomendanda para entornos en los que:

- El mercado es competitivo, además de los conceptos de básicos de calidad, coste y diferenciación exigen rapidez y flexibilidad.
- Se exige ciclos de desarrollo más cortos para entregar nuevas funcionalidades en el menor tiempo posible.

Scrum adapta los principios básicos de los sistemas Lean:

- Maximizar el valor
- Eliminar el gasto

Así como los valores de la corriente Agile:

- Colaboración con el cliente
- Adaptación a cambios
- Entrega periódica de productos funcionales

Y las mejores prácticas de la industria:

- Ciclos cortos de desarrollo
- Autogestión de equipos
- Visibilidad del proceso

Todos estos puntos fuertes de Scrum, hacen de esta metodología un marco de trabajo que proporciona una gran productividad, así como una motivación importante para el equipo de trabajo, al sentirse parte importante del proceso. La filosofía de Scrum no se caracteriza por la necesidad de seguir un plan, sino en una continua adaptación a los cambios en el proyecto.

Es por ello, que para este proyecto concreto, se ve adecuado el uso de esta metodología, y se seguirá a lo largo de todo el proceso, mediante las iteraciones, llamadas “Sprint” en Scrum.

En cada iteración, de 1 semanas de duración, se proporcionará una parte del producto final completamente funcional y lista para usar (la funcionalidad desarrollada), y de forma iterativa, se le irán añadiendo funcionalidades hasta llegar al producto final.

A continuación se indica la pila de producto. Indicar que la pila de producto es algo vivo y se irá adaptando ciclo a ciclo.

Categoría	Historia	Prioridad	Dedicación (días)	Fecha entrega
Diseño Centrado en Usuario	Usuarios y contexto de uso [Análisis] Diseño conceptual [Diseño] Prototipado [Diseño] Evaluación [Evaluación]	1000	4	05/04/17
Diseño técnico de la aplicación	Definición de los casos de uso	960	5	05/04/17
Diseño técnico de la aplicación	Diseño de la arquitectura de la aplicación	920	2	05/04/17
Documentación	Incluir los trabajos realizados en la memoria	880	2	05/04/17
Desarrollo	Autenticación de usuario	840	1	17/05/17
Desarrollo	Registro de entrada y salida	800	1	17/05/17
Desarrollo	Contabilidad de las horas pendientes de realizar	760	1	17/05/17
Desarrollo	Proporcionar información: - A que hora se puede salir - Si ya se han realizado las horas, avisar de que puede salir.	720	2	17/05/17
Desarrollo	Creación de acciones automáticas con mensajes personalizados: - Avisar a grupos concretos de cuando se entra a trabajar. - Notificar cuándo se pretende salir para organizar de forma fácil eventos.	680	8	17/05/17
Desarrollo	Uso de tecnología NFC para marcar entrada o salida.	640	6	17/05/17
Documentación	Incluir los trabajos realizados en la memoria	600	5	17/05/17
Entrega	Preparación de material a entregar: C	560	1	07/06/17
Entrega	Finalización de la memoria.	520	9	07/06/17
Entrega	Presentación vídeo	480	4	07/06/17

En esta pila de producto aparecen 3 hitos, que son los correspondientes a cada entrega marcada por las 3 siguientes PECs.

1.5 BREVE SUMARIO DE PRODUCTOS OBTENIDOS

El producto obtenido consiste en una aplicación funcional que permita la planificación personal en base a la disponibilidad laboral.

1.6 BREVE DESCRIPCIÓN DE LOS OTROS CAPÍTULOS DE LA MEMORIA

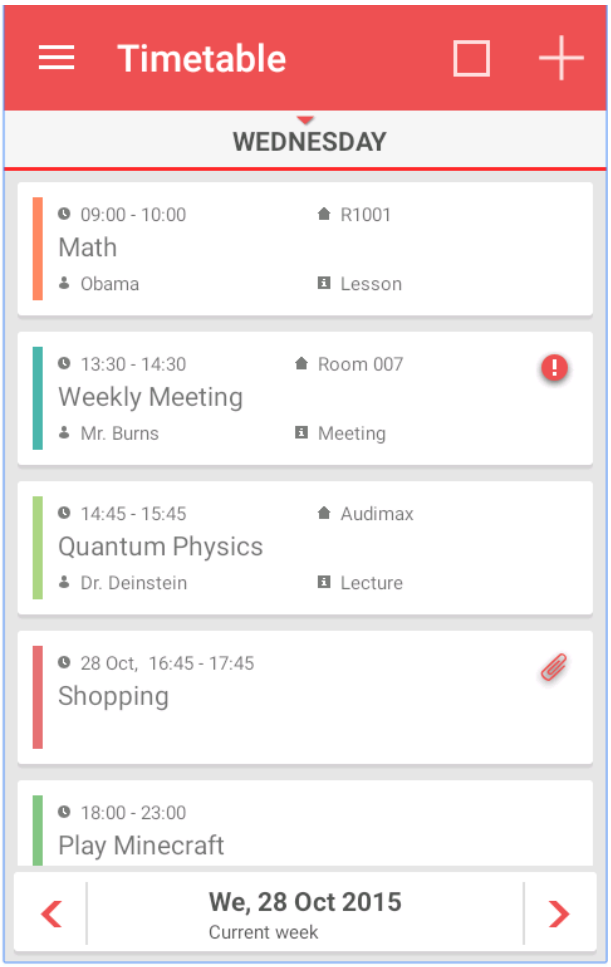
- **Análisis**
- **Fundamentos tecnológicos**: descripción de los distintos elementos tecnológicos en los que se sustenta el trabajo.
- **Scrum**: En este caso explicar mas afondo Scrum y como se ha aplicado.
- **Implementación**: Funcionalidades más destacas.
- **Implementación de la aplicación**
- **Aplicación desarrollada:** Se muestran las pantallas y funcionalidades más destacadas.
- **Conclusión y contraste de objetivos.**
- **Líneas futuras.**

2 Análisis

2.1 ANÁLISIS DE ANTECEDENTES

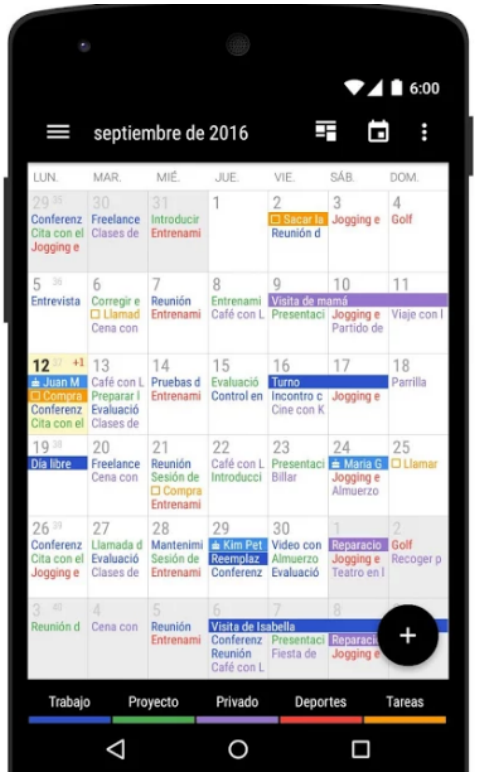
Actualmente existen aplicaciones que muestran información horaria, pero se centran más en la parte meramente contable, en la que se apuntan las horas realizadas o a modo recordatorio. El objetivo de esta aplicación es la de usar esa información para interactuar con compañeros de trabajo o con otras personas para conciliar la vida laboral y personal, así como para indicar cuándo en el entorno laboral se está disponible o no.

2.1.1 TimeTable++ horario



Esta aplicación lleva un control de las diferentes actividades, muy útil para llevar el control de los horarios de clase.

2.1.2 Calendario Business Agenda



Similar a la anterior aplicación y no es más que un calendario con eventos.

2.1.3 Tiempo de Trabajo

Esta aplicación, no hace de agenda, pero si de control exacto de las horas realizadas. El interfaz es muy diferente al indicado de Android y para nada sigue las prácticas de Material Design. No interacciona con otras personas en base a la entrada o salida en la jornada laboral.

2.2 USUARIOS Y CONTEXTO DE USO

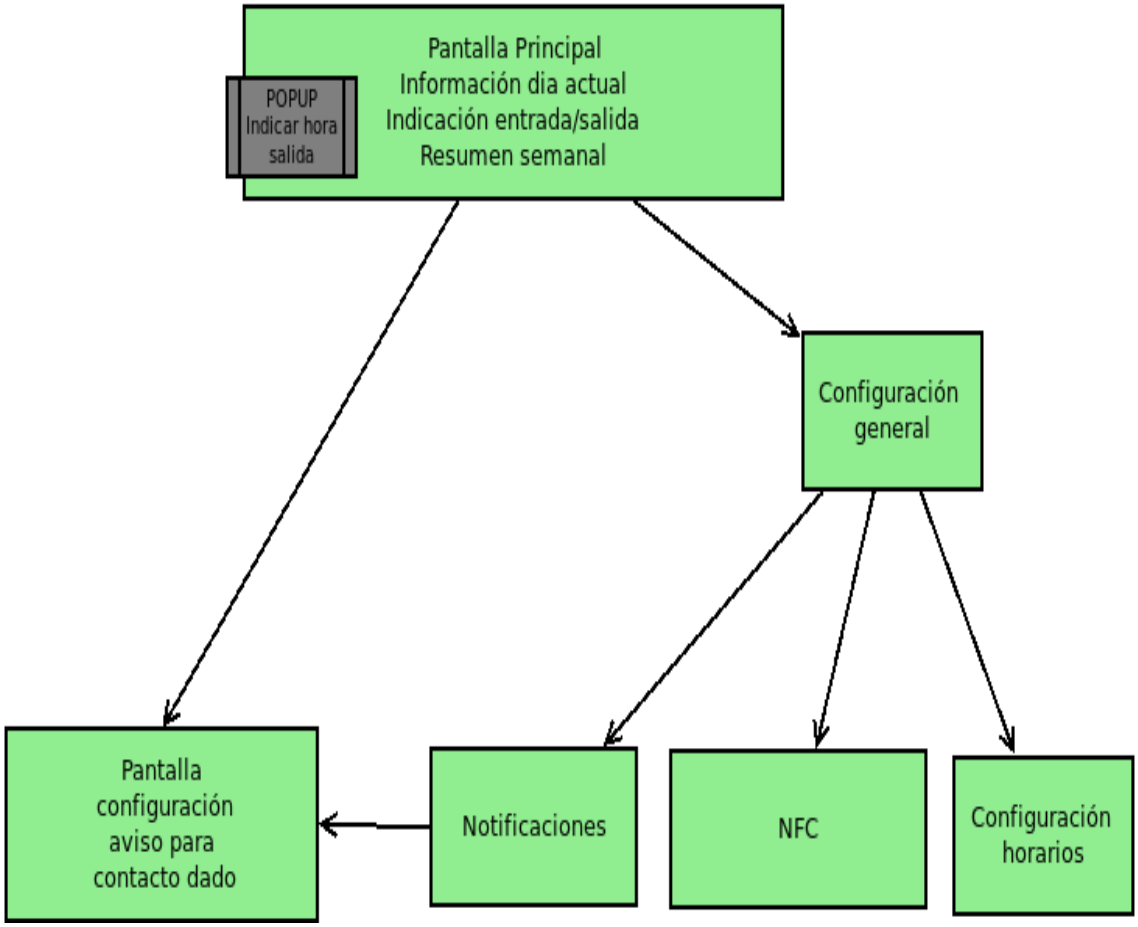
2.2.1 Ficha/s de persona y escenario.

A continuación se describen las fichas de persona y escenario:

Nombre	María.
Edad	37
Profesión	Funcionaria de la administración general del estado.
Descripción de la persona	Es funcionaria de la Administración General del Estado y reside en Madrid. Todos los días coge el Metro hasta su trabajo. En su tiempo libre le gusta ir de compras y pasear al salir de trabajar. Es una usuaria a la que le gusta probar nuevas aplicaciones.
Descripción del escenario	Lunes, a las 8:13 de la mañana, llega al trabajo e indica la hora de entrada en la aplicación. En función de su jornada laboral, le toca salir de trabajar a las 18:13. Con la aplicación avisa a su pareja que ya está en su jornada de trabajo y le indica a partir de que hora sale para estar disponible para realizar un paseo.

Nombre	Daniel
Edad	39
Profesión	Funcionario del ministerio de energía, turismo y agenda digital
Descripción de la persona	Es funcionario y vive en Pontevedra, en una provincia diferente a la de su pareja. Al llegar al trabajo suele avisar a su pareja de que ha entrado, así como a sus compañeros para interaccionar con ellos, puesto que no trabajan ni en la misma oficina ni en la misma provincia. También para conocer cuando su mujer entra a trabajar.
Descripción del escenario	Es Lunes a las 18:30 de la tarde, sale de trabajar y notifica, mediante la aplicación, que sale de trabajar. Esta de forma automática, avisa a su pareja que ya está disponible y a sus compañeros de trabajo que se va y no podrá estar ya a su disposición.

2.2.2 Flujo de interacción



Descripción de las pantallas:

- **Pantalla principal**: Mostrará la información principal de la aplicación:
  - cuántas horas ha realizado tanto en el día actual como semanal.
  - las que quedan por realizar en la jornada
  - Poder marcar entrada o salida, según proceda.

  En esta pantalla, una ventana emergente, tras pulsar el botón principal, permitirá enviar una notificación a los contactos preseleccionados.

  También podrá añadir una nueva notificación para un contacto que seleccione.

- **Configuración general,** podrá establecer la configuración general de la aplicación.

- **NFC:** Podrá activar una etiqueta NFC para poder marcar entrada o salida.
- **Notificaciones:** En esta pantalla podrá ver los avisos a contactos que están configurados.
- **Pantalla configuración aviso para contacto dato**: Para un contacto dado, podrá editar la configuración, tal como texto, etc.
- **Configuración de horarios**: Para indicar las horas a realizar semanalmente, horas a hacer cada día, etc.

2.2.3 Prototipos

2.2.3.1 Entrada a trabajar

Se muestra la pantalla de entrar cuando estamos dentro del rango de entrada a trabajar. El usuario marcará entrar por defecto y si tiene marcado “avisar a mis contactos” avisará a aquellos que estén ya configurados.

Alternativamente puede:

- Indicar otra hora de entrada. Para los casos que se ha olvidado de hacerlo.
- Añadir contacto nuevo para avisar de forma automática.
- Desmarcar avisar a mis contactos.

2.2.3.2 Salir de trabajar

Se muestra la pantalla de salir cuando estamos dentro del rango de salida de trabajar. El usuario marcará salir por defecto y si tiene marcado “avisar a mis contactos” avisará a aquellos que estén ya configurados.

Alternativamente puede:

- Indicar otra hora de salida. Para los casos que se ha olvidado de hacerlo.
- Añadir contacto nuevo para avisar de forma automática.
- Desmarcar avisar a mis contactos.

2.2.3.3 Pantalla de jornada laboral.

Se muestra la pantalla de jornada laboral cuando no tiene que marcar entrada o salida o como pantalla destino cuando ha marcado uno de los dos estados anteriores.

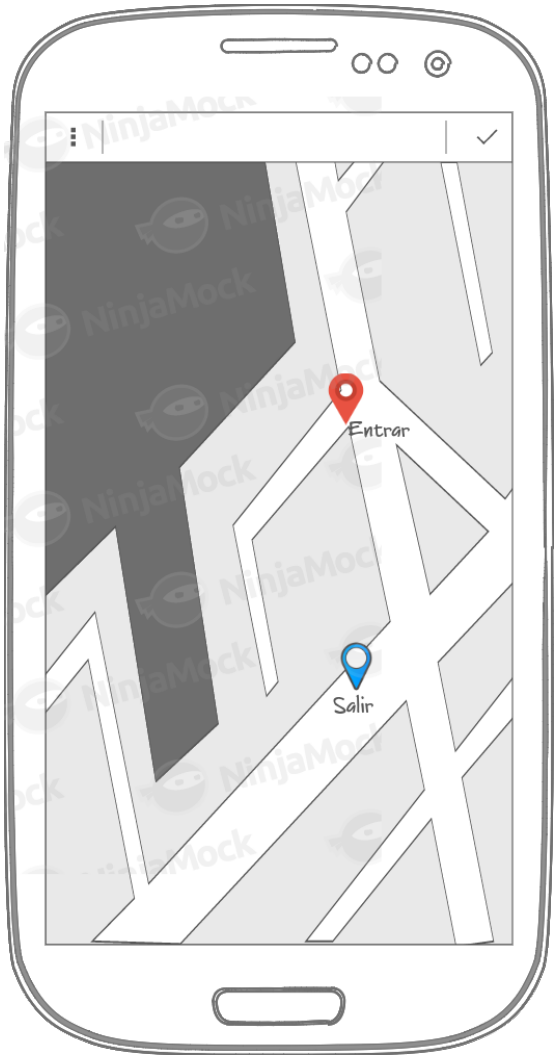
En esta pantalla puede:

- Indicar otra hora de salida. El sistema calcula cuándo saldrá, pero puede que el usuario tenga que salir antes o despues del horario.
- Avisar: Si ha cambiado la hora, o si quiere volver a avisar, marcará este botón.
- Visualizar la información de horas realizadas, tanto en el día actual, como en la propia semana.

2.2.3.4 Pantalla de jornada laboral: Cambio de hora.

Pulsando el botón de cambiar hora de salida, se puede indicar una hora diferente.

2.2.3.5 Configuración de información geoposición



Se muestra la pantalla de configuración de la marca geolocalizada, a partir de la cual indicará que es entrada o salida.

2.2.3.6 Gestión de contactos

Esta pantalla, permite seleccionar los usuarios disponibles para los que se les podrá enviar notificaciones. También es la pantalla a la que se irá cuando se quiera añadir un aviso de forma manual.

2.2.3.7 Configuración de aviso

En esta pantalla, para un contacto dado, se podrá indicar el texto que se le enviará al entra o al salir, asi como marcar cuándo enviar la notificación, si al entrar, o al salir. También se puede elegir que cuando se entre, se le indique o no la estimación de hora de salida.

2.2.3.8 Diseño gráfico.

2.2.3.8.1 Uso de Material Design

La aplicación desarrollada sigue la nueva filosofía de Google presente a partir de su versión 5.0, “Material Design”. Si bien no es un dogma que haya que seguir de forma estricta, sí que sirve de guía para el desarrollo de interfaces, proporcionando un lenguaje visual perfectamente definido. En “Material Design” la sugerencia es siempre usar tres colores: “Primary”, “PrimaryDark” y “Accent”. La siguiente imagen, de la documentación oficial de Google identificame claramente los distintos colores:

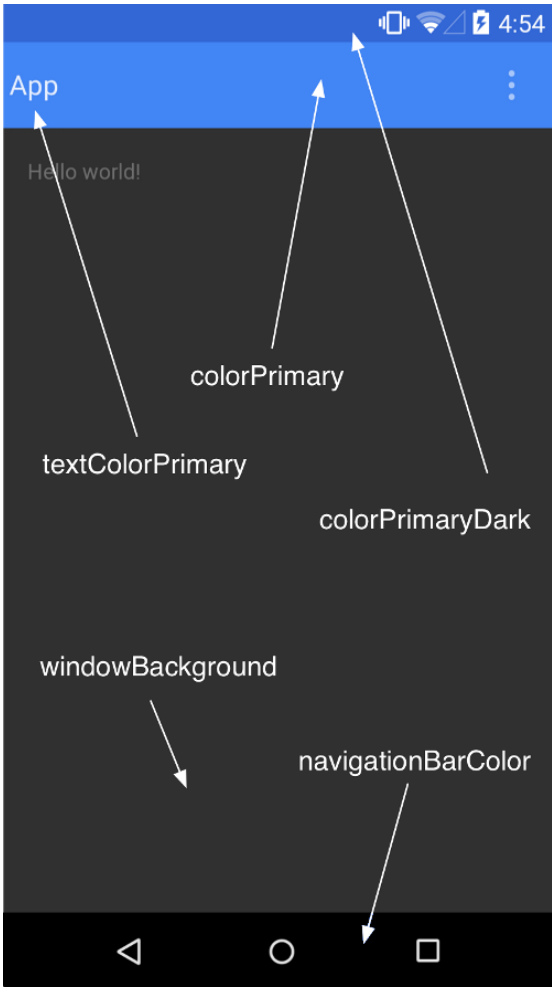


Ilustración 1: Material DesignFuente: Android.com

En el caso de este proyecto los colores definidos mediante el fichero xml de estilo correspondiente han sido:

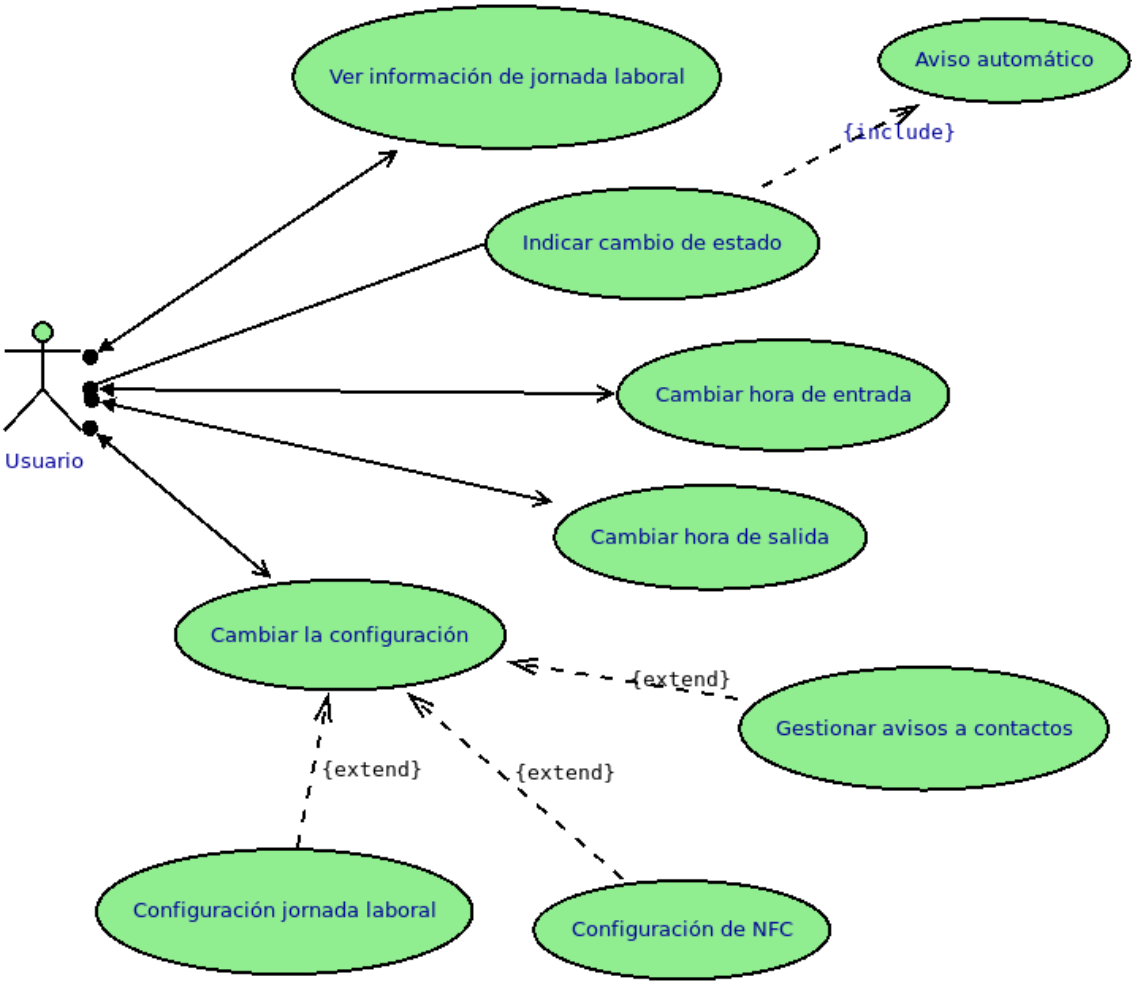
```
<color name="colorPrimary">#039BE5</color>
<color name="colorPrimaryDark">#0288D1</color>
<color name="colorAccent">#FFA000</color>
```

2.2.3.8.2 Barra lateral

En el diseño está contemplado el uso de una barra lateral. Así, se podrá acceder de forma rápida a las opciones más importantes de la aplicación, como es el caso de administrar las notificaciones.

2.3 DIAGRAMAS DE CASOS DE USO

Se emplea la notación UML para realizar el modelado del proyecto. En dicho análisis se emplearon diagramas de casos de uso resumidos en este apartado.



2.3.1 Descripción casos de uso

Nombre	Ver información de la jornada laboral	
Actores	Usuario	
Objetivo	La aplicación permitirá al usuario consultar la información actual de su jornada	
Pre-condición	• El teléfono inteligente cumple los requisitos necesarios para el correcto funcionamiento de la aplicación. • La aplicación está instalada en el teléfono.	
Post-condición		
Escenario		
Paso	**Actor**	**Descripción**
1	Usuario	El usuario arranca la aplicación
3	Sistema	Se muestra la información al usuario
Escenario alternativo		
Paso	**Actor**	**Descripción**

Nombre	Indicar Entrada
Actores	Usuario
Objetivo	La aplicación permitirá al usuario indicar la entrada en la jornada laboral
Pre-condición	• El teléfono inteligente cumple los requisitos necesarios para el correcto funcionamiento de la aplicación, en concreto acceso a internet. • La aplicación está instalada en el teléfono. • La hora actual está dentro del rango del horario laboral. • No ha marcado previamente Entrada
Post-condición	

Escenario		
Paso	**Actor**	**Descripción**
1	Usuario	El usuario arranca la aplicación
2	Sistema	Se muestra la información al usuario, mostrando el botón de Entrar
3	Usuario	Pulsa en el botón “Entrar”
4	Sistema	Se registra la entrada
5	Sistema	Envía las notificaciones a los contactos.
6	Sistema	Se muestra la pantalla de información de jornada laboral

Escenario alternativo		
Paso	**Actor**	**Descripción**
3	Usuario	Desmarca la opción de avisar automáticamente
4	Sistema	Se registra la entrada del usuario
5	Sistema	Se muestra la pantalla de información de jornada laboral.

Nombre	Indicar Salida
Actores	Usuario
Objetivo	La aplicación permitirá al usuario indicar la salida de la jornada laboral
Pre-condición	• El teléfono inteligente cumple los requisitos necesarios para el correcto funcionamiento de la aplicación, en concreto acceso a internet. • La aplicación está instalada en el teléfono. • La hora actual está dentro del rango del horario laboral. • Ha marcado previamente Entrada
Post-condición	

Escenario		
Paso	**Actor**	**Descripción**
1	Usuario	El usuario arranca la aplicación
2	Sistema	Se muestra la información al usuario, mostrando el botón de Salir
3	Usuario	Pulsa en el botón “Salir”
4	Sistema	Se registra la salida
5	Sistema	Envía las notificaciones a los contactos.
6	Sistema	Se muestra la pantalla informativa.
Escenario alternativo		
Paso	**Actor**	**Descripción**
3	Usuario	Desmarca la opción de avisar automáticamente
4	Sistema	Se registra la salida del usuario
5	Sistema	Se muestra la pantalla de información de jornada laboral.

Nombre	Aviso automático
Actores	Sistema
Objetivo	La aplicación enviará una notificación de cambio de estado a los usuarios que se han marcado previamente para ser notificados.
Pre-condición	• El teléfono inteligente cumple los requisitos necesarios para el correcto funcionamiento de la aplicación, en concreto acceso a internet. • La aplicación está instalada en el teléfono. • Ha indicado un cambio de estado • Existen contactos que deben ser notificados
Post-condición	

Escenario		
Paso	**Actor**	**Descripción**
6	Sistema	Envía las notificaciones a los contactos.

Escenario alternativo		
Paso	**Actor**	**Descripción**

Nombre	Cambiar hora de salida
Actores	Usuario
Objetivo	La aplicación permitirá al usuario cambiar la hora prevista de salida, para aquellos casos en los que se le ha pasado hacerlo a la hora, o sabe que saldrá más tarde.
Pre-condición	• El teléfono inteligente cumple los requisitos necesarios para el correcto funcionamiento de la aplicación, en concreto acceso a internet. • La aplicación está instalada en el teléfono. • Ha indicado que ha entrado a trabajar.
Post-condición	

Escenario		
Paso	**Actor**	**Descripción**
1	Usuario	El usuario arranca la aplicación
2	Sistema	Se muestra la pantalla informativa.
3	Usuario	Pulsa en el botón notificar cambio de hora
4	Sistema	Registra la hora prevista de salida

Escenario alternativo		
Paso	**Actor**	**Descripción**

Nombre	Cambiar hora de Entrada
Actores	Usuario
Objetivo	La aplicación permitirá al usuario cambiar la hora de entrada, para aquellos casos en los que se le ha pasado hacerlo a la hora
Pre-condición	• El teléfono inteligente cumple los requisitos necesarios para el correcto funcionamiento de la aplicación, en concreto acceso a internet. • La aplicación está instalada en el teléfono. • No ha indicado que ha entrado a trabajar.
Post-condición	

Escenario		
Paso	**Actor**	**Descripción**
1	Usuario	El usuario arranca la aplicación
2	Sistema	Se muestra la pantalla informativa.
3	Usuario	Pulsa en el botón notificar cambio de hora
4	Sistema	Registra la hora prevista de entrada

Escenario alternativo		
Paso	**Actor**	**Descripción**

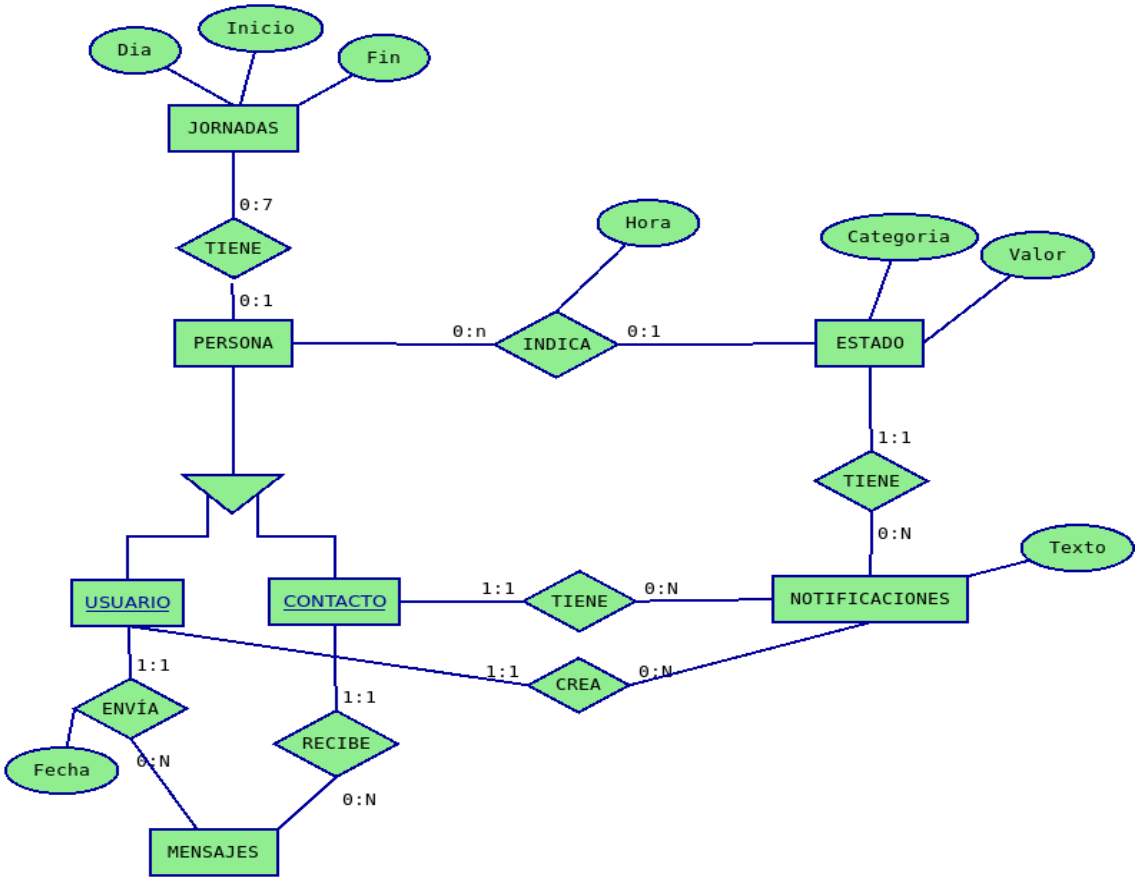
Nombre	Cambiar la configuración
Actores	Usuario
Objetivo	La aplicación permitirá al usuario cambiar los diferentes parámetros de configuración
Pre-condición	• El teléfono inteligente cumple los requisitos necesarios para el correcto funcionamiento de la aplicación, en concreto acceso a internet. • La aplicación está instalada en el teléfono.
Post-condición	

Escenario		
Paso	**Actor**	**Descripción**
1	Usuario	El usuario arranca la aplicación
2	Usuario	Selecciona el elemento de configuración que quiere cambiar
3	Sistema	Muestra la pantalla de configuración específica
4	Usuario	Realiza los cambios pertinentes.
5	Sistema	Registra los cambios

Escenario alternativo		
Paso	**Actor**	**Descripción**

2.4 DIAGRAMA ENTIDAD RELACIÓN

A continuación se muestra el diagrama entidad relación generado para la aplicación a desarrollar:



En el caso de Persona, se han creado subentidades, para indicar que una persona es usuario de la aplicación, en la que crea notificaciones, y otras son contactos que reciben dichas notificaciones.

En el ejemplo anterior, una persona indica un cambio de estado, por ejemplo sería la siguiente tupla:

(Persona: Daniel Díaz, Relación: 09:23, Estado: Trabajo, Entrada)

Otra tupla sería, en la relación usuario con notificaciones:

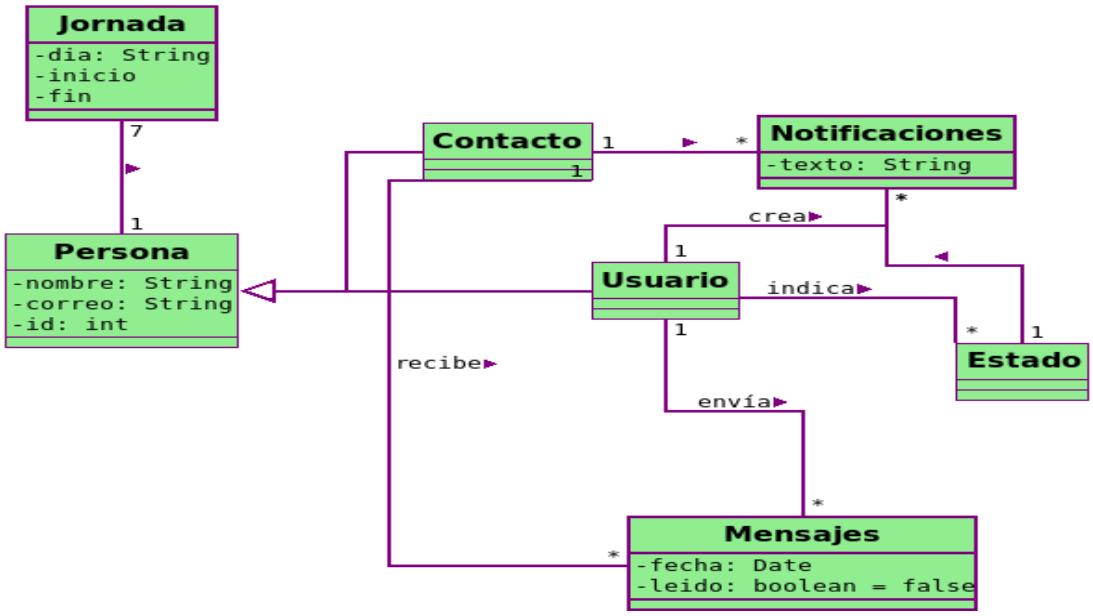
(Usuario: Daniel Díaz, Notificaciones: “Estoy ya en la oficina!”)

Notificación con contacto:

(Notificaciones: “Estoy ya en la oficina!”, Contacto: Pepe)

2.5 DIAGRAMA DE CLASES

En el siguiente diagrama, se muestra el diagrama de clases



Al usar Scrum, y ser un proceso iterativo, el diagrama de clases irá evolucionando en base a las historias definidas en cada ciclo.

2.6 ARQUITECTURA DEL SISTEMA

Se opta por seguir el patrón Modelo-Vista-Controlador. La finalidad con la que surgió este patrón era la de disminuir el coste de programación en la implementación de sistemas múltiples y sincronizados de los mismos datos. MVC separa los datos de la aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos: Modelo, Vistas y Controladores. Cada componente se trata cómo entidades separadas.

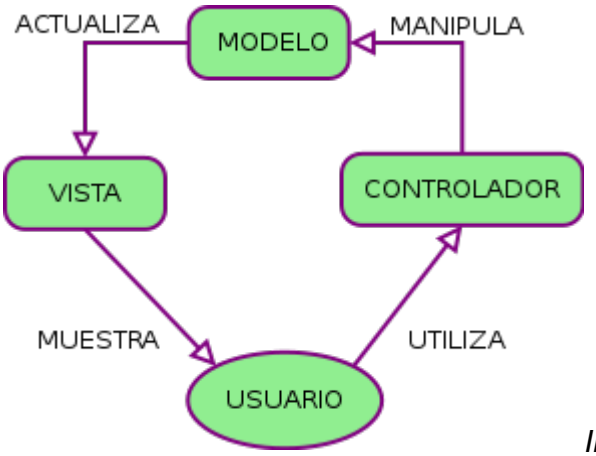


Ilustración 2: Ejemplo colaboración en Modelo - Vista - Controlador

Al estar separado en componentes, permite implementaciones por separado de cada undo de los componentes. También permite que un componente se pueda cambiar o refactorizar sin que afecte al resto de los componentes.

- Modelo: Representa los datos del sistema. No tiene referencia con el resto de los componentes. El sistema es el encargado de notificar a las vistas cuando hay un cambio de modelo.

- Vista: Es la representación visual del modelo. Ej: Página web, Interfaz de ventanas.
- Controlador: Proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Ej: Los sistemas de gestión de bases de datos.

De forma general, aunque se pueden encontrar distintas implementaciones de MVC, el flujo que sigue el control es el siguiente:

1. El usuario interactúa con la interfaz de usuario (ej: pulsa un botón, pincha en un enlace, etc)

2. El controlador recibe, por parte de los objetos de la interfaz-vista, la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, normalmente a través de un gestor de eventos, llamado handler o callbak.

3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (como por ejemplo: actualizar el carro de compra). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se reflejan los cambios en el modelo (ejemplo: produce un listado del contenido del carro de la compra). El modelo no tiene conocimiento directo sobre la vista. Sin embargo, el patrón de observador puede ser empleado para proporcionar cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquiera cambio.

5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

Las Vistas y los Controladores conforman la interfaz de usuario. Un mecanismo de propagación de cambios asegura la consistencia entre la interfaz y el modelo. La separación del modelo de los componentes vista y controlador permite tener múltiples vistas del mismo modelo. Si el usuario cambia el modelo a través del controlador de una vista, todas las otras vistas dependientes deben reflejar los cambios. Por tanto, el modelo notifica a todas las vistas siempre que sus datos cambien. Las vistas, en cambio, recuperan los nuevos datos del modelo y actualizan la información que muestran al usuario.

Este patrón es muy popular y fue portado a una gran cantidad de entornos y frameworks. El MVC es un patrón ampliamente utilizado en múltiples plataformas y lenguajes. Algunos de sus principales beneficios son:

- Menor acoplamiento
  - Desacopla las vistas de los modelos.
  - Desacopla los modelo de la forma en que se muestran e introducen los datos.

- Mayor cohesión
  - Cada elemento del patrón esta altamente especializado en su tarea (la vista en mostrar datos al usuario, el controlador en las entradas y el modelo en su objetivo de negocio).

- Las vistas proporcionan mayor flexibilidad y agilidad
  - Se puede crear múltiples vistas de un modelo.
  - Se puede crear, añadir, modificar y eliminar nuevas vistas dinámicamente.
  - Las vistas pueden anidarse.
  - Se puede cambiar el modo en que una vista responde al usuario sin cambiar su representación visual.

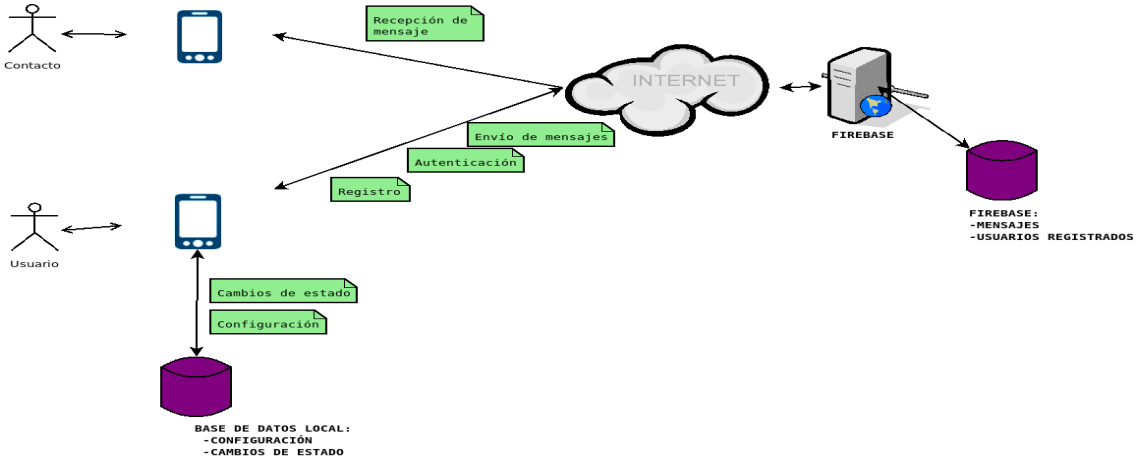
    - Se puede sincronizar las vistas.
    - Las vistas pueden concentrarse en diferentes aspectos del modelo.

- Mayor facilidad para el desarrollo de clientes ricos en múltiples dispositivos y canales, por ejemplo:
    - Una vista para cada dispositivo que puede variar según sus capacidades.
    - Una vista para la Web y otra para aplicaciones de escritorio

- Más claridad de diseño.
- Facilita el mantenimiento.
- Mayor escalabilidad.

2.6.1 Diagrama de arquitectura

Para este proyecto, se presenta el diagrama de arquitectura inicial



Se empleará la solución Firebase para almacenar la información que sea necesaria tener en la nube, como es el caso de los mensajes que leerán otros usuarios y la autenticación.

La información de configuración, así como cambio de estado, etc, se almacenará en una base de datos local.

El usuario, mediante la aplicación indicará cambios de estado, envíos de mensajes, etc. A su vez, los contactos asociados, suscritos a dichas notificaciones, recibirán los pertinentes mensajes de cambio de estado, tal como “entrar a trabajar”, “salir de trabajar”, etc.

3 Fundamentos tecnológicos

3.1 ANDROID

La empresa creadora de Android fue fue adquirida en el año 2005 por Google, que liberó la mayoría del código de Android bajo la Licencia Apache, una licencia libre y de código abierto. Al ser un sistema operativo de código abierto, cualquier desarrollador puede crear y distribuir aplicaciones para Android.

Existe una gran comunidad de desarrolladores creando aplicaciones Java para extender la funcionalidad de los dispositivos con sistema operativo Android.

Google proporciona su plataforma de distribución de aplicaciones, Google Play, que proporciona un mercado global en el que los desarrolladores pueden hacer llegar a los clientes sus aplicaciones tanto gratuitas como de pago. El coste para publicar aplicaciones en Google Play es de 25 Euros, un coste que se paga una sola vez y permite de forma duradera publicar tantas aplicaciones como sean necesarias.

Android[2] está formado por los siguientes componentes:

- Núcleo: Android se basa en Linux para los servicios básicos del sistema, como seguridad, gestión de memoria, gestión de procesos, pila de red, y modelo de controladores. El núcleo actúa además como una capa de abstracción entre el hardware y el resto de la pila de software.
- Bibliotecas: Incluyen funciones para gráficos 2D y 3D, reproducción y grabación de archivos multimedia, bases de datos para almacenamiento estructurado de datos, así como otras funcionalidades accesibles a los desarrolladores a través del framework de aplicaciones de Android.
- Framework de aplicaciones: Permitiendo así la reutilización de componentes.
- Runtime de Android: Android incluye un conjunto de bibliotecas base que proporcionan la mayoría de las funciones disponibles en las bibliotecas base del lenguaje Java. Además, cada aplicación posee su propia instancia de la máquina virtual, mejorando así la seguridad y estabilidad del sistema, al estar aisladas las aplicaciones unas de otras.
- Aplicaciones: Proporciona una serie de aplicaciones base, tales como navegador, mapas, y gestión de contactos.

En la siguiente figura se muestra la arquitectura de Android:

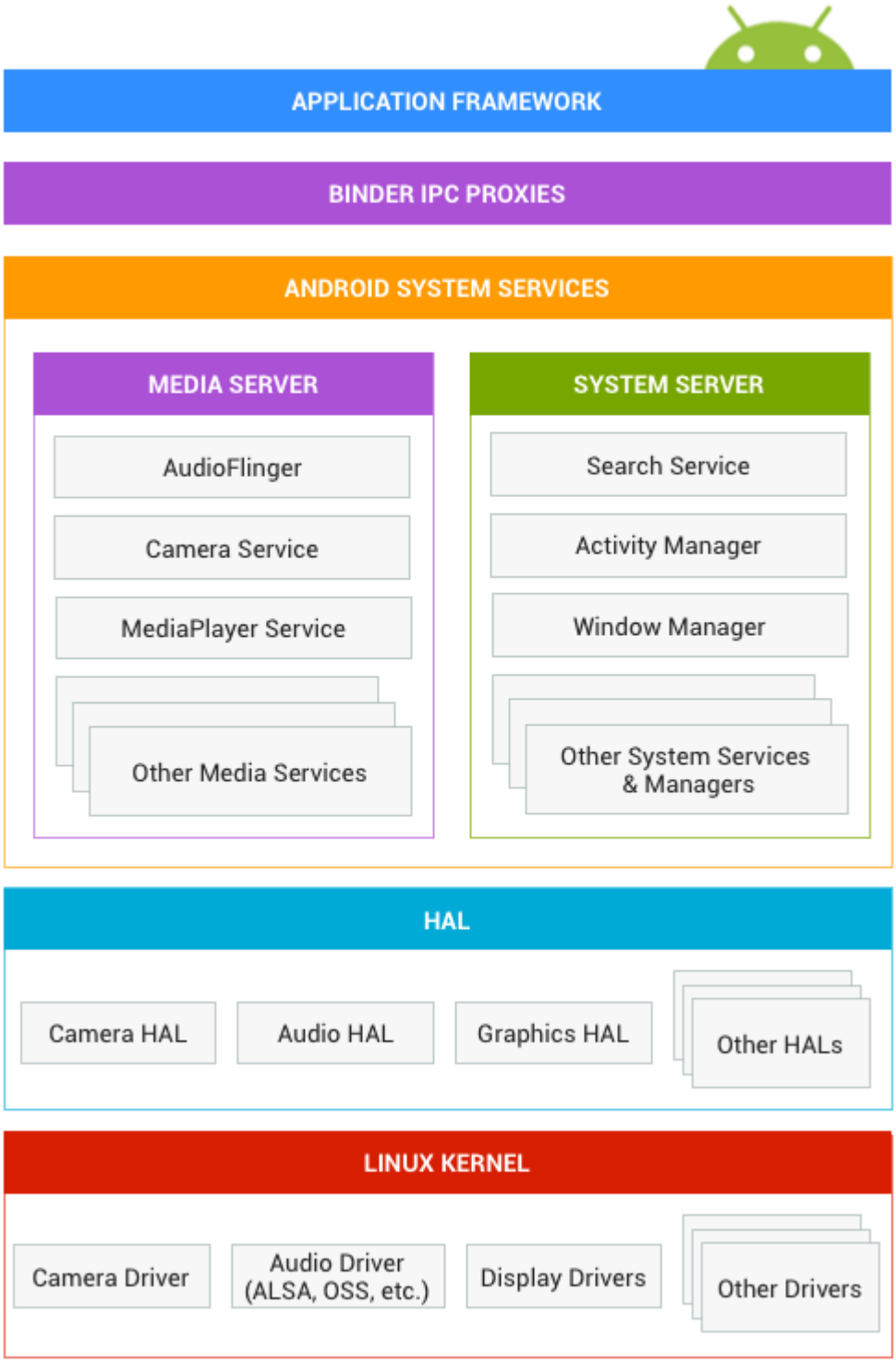


Ilustración 3: https://source.android.com/devices/

4 Scrum

Scrum comparte los principios del desarrollo ágil en el que a partir de la visión de una necesidad del cliente construye la solución de forma incremental mediante iteraciones breves. En Scrum estas iteraciones se llaman Sprints o Ciclos.

Hay tantos sprints como sean necesarios hasta que el cliente considere concluido el trabajo. La duración de los sprints no es menor a dos semanas ni superior a 2 meses. La duración exacta, dentro de ese intervalo, para equipos noveles en Scrum, se determinará en función del intervalo que mejor se adapte a cada equipo de desarrollo tras varios sprints de prueba. Una vez elegida la duración, esta se mantendrá invariable puesto que la duración no debe variar en función de la carga de trabajo o de otros factores.

En cada iteración se implementará una parte del producto, que se considera como un producto entregable, es decir que está suficientemente probada y que proporciona una funcionalidad que el cliente puede usar.

Es muy recomendable emplear un tablón (mejor físico que digital) para realizar el seguimiento de cada ciclo. Este tablón contiene una lista de las historias pendientes de realizar, las que están en curso y también las realizadas. Ello permite ver de una forma sencilla la evolución del proyecto.

4.1 INFORMACIÓN DE SCRUM

4.1.1 Roles

En Scrum se definen los siguientes roles:

- ScrumMaster: Persona encargada de que se siga la metodología de Scrum. Esta persona estará encargada de que las reuniones se hagan a la hora acordada, etc. El ScrumMaster no tiene porqué ser el jefe del equipo, solo es aquella persona encargada de llevar las partes burocráticas del proceso.
- Dueño del producto: Es aquella persona que define y prioriza las historias a introducir en cada ciclo. Sólo puede haber un dueño de producto.
- ScrumTeam: Son los miembros del equipo de programación que hacen el desarrollo indicado.

4.1.2 Los elementos

Para trabajar con Scrum se trabaja con la siguiente información:

- Pila de producto: Una lista, priorizada por el dueño del producto, donde están recogidos todos los requisitos de la aplicación. A esos requisitos se les llama “historias”.

- Pila de Sprint: Es una lista, priorizada y perfectamente estimada por parte de los miembros del equipo, con las historias a realizar en el ciclo.

- Incremento: Es el resultado de cada iteración de sprint y son las funcionalidades realizadas, testeadas y listas para que el cliente las pueda emplear.

- Gráfico de Burn-down: El gráfico de Burn-down es una información que se obtiene en la realización de cada ciclo y consiste en un gráfico que en el eje Y contiene el esfuerzo a realizar en el ciclo, y en el eje X los días de trabajo. Permite realizar un seguimiento del propio sprint. Es un gráfico alimentado por el propio equipo de programación.

- Gráfico de Burn-up: Gráfico que contiene la información de todas las historias previstas para la realización del producto en cada sprint. Este gráfico permite realizar una planificación y seguimiento del producto. Se construye a partir de la pila de producto.

4.1.3 Las reuniones

En cada sprint de Scrum se realiza una serie de reuniones acotadas en tiempo para poder definir tanto cada incremento a realizar como las desviaciones y reflexiones en el proceso.

- Planificación del Sprint: Reunión previa al inicio de cada sprint en la que se define el trabajo a realizar. En función del equipo de trabajo y de los días asignados se define la velocidad de trabajo y se escogen las historias que el dueño de producto indicó como más prioritarias.

- Seguimiento diario: También llamada “Scrum diario” o “Scrum Daily,” es una reunión de 15 minutos a primera hora de la mañana, cada día que dura el sprint para hacer una revisión diaria del estado del proceso, en la que cada miembro debe responder a tres preguntas:
  - ¿Qué hice ayer?
  - ¿Qué haré hoy?
  - ¿Qué impedimentos tengo?. El ScrumMaster debe resolver los impedimentos resultado de responder a esta pregunta.

  Tras esta reunión se actualiza la pila de sprint y el gráfico de evolución.

- Demo: En esta reunión, en la que puede acudir cualquier persona, se enseña a los interesados el incremento realizado. Esta demo es realizada por parte de los miembros del equipo de desarrollo, para que haya una comunicación directa y favorezca la información entre clientes y el equipo de programación.

- Retrospectiva: Los miembros del equipo, junto con el dueño del producto, cuya asistencia es opcional, reflexionan sobre la evolución del sprint, tanto de los aspectos positivos como negativos para mejorar en siguientes ciclos la eficiencia del equipo.

4.2 INFORMACIÓN MÁS DESTACADA DE CADA SPRINT EN ESTE PROYECTO

Para la realización de la PEC de Implementación se estimaron 4 Sprints. A continuación se indican los aspectos más relevantes de cada uno de ellos y como con Scrum se ha podido adaptar a los imprevistos o realizar cambios durante el proceso.

4.2.1 Sprint 4

El Sprint 4 contemplaba los siguientes elementos:

- Autenticación de usuario
- Registro de entrada y salida
- Contabilidad de las horas pendientes de realizar
- Proporcionar información:
  - A que hora se puede salir
  - Si ya se han realizado las horas, avisar de que puede salir.

En este Sprint, además, se tuvo que aprender sobre el uso de firebase, tiempo que no se tuvo en cuenta y que implicó que en este Sprint sólo se pudieron hacer las dos historias más prioritarias (Autenticación y registro de entrada y salida), descartando para el próximo sprint el resto.

En este Sprint, al entregar la posibilidad de entrar y salir, se descubrió que del prototipo realizado había una serie de carencias.

- En el se consideraba que una vez fichado salida, y realizadas las horas, ya no se podía volver a fichar entrada. Nos llevaba a otra pantalla separada, que simplemente nos daba la información de las horas realizadas y pendientes. Con las pruebas se descubrió que no era siempre así. Puedes acabar teniendo que ir a trabajar, y como están ya todas las horas cubiertas, no te muestra la pantalla.
- De la misma manera, consideraba que hasta la hora de salir no lo iba a hacer, por lo que se mantenía en la pantalla de información hasta que fuese la hora de salida. Tampoco era práctico, porque por ejemplo, el usuario puede tener que salir por un asunto particular, y la aplicación no lo tenía en cuenta.

En base a estos elementos previos, se vio la necesidad de realizar un cambio y fusionar la pantalla de información con la de marcar entrada y salida. De tal manera que el usuario puede a la vez ver toda la información y además fichar entrada y salida siempre que lo considere necesario.

En el registro de entrada y salida, así como el resto de información necesaria, inicialmente toda esa información se guardaría en Firebase, pero salvo los

mensajes a enviar a los contactos, el resto de la información no aportaba nada estar en la nube, por los siguientes motivos:

- Consumo de datos innecesario para obtener información que podía estar almacenada en local.
- Puede haber información sensible, como los fichajes de entrada y salida, los datos de contacto que el usuario no quiera almacenar en la nube.

Es por ello que se realizó el estudio de que soluciones para tener persistencia local serían las más útiles y tras el estudio, se implementó Sugar ORM.

4.2.2 Sprint 5

En vista de los resultados del Sprint anterior, se incluyeron las siguientes historias:

- Fusionar pantallas de información y fichar
- Contabilidad de las horas pendientes de realizar
- Proporcionar información:
  - A que hora se puede salir
  - Si ya se han realizado las horas, avisar de que puede salir.
- Funcionalidad de geolocalización para entrada y salida.

Como se puede ver, se creó una historia nueva, en base a lo aprendido de las pruebas del ciclo anterior. Gracias al uso de Scrum, se pudo hacer esta corrección en una etapa temprana del desarrollo, adaptando el producto a las necesidades reales del cliente.

El conocimiento básico en Firebase también mermó la velocidad de este sprint, siendo las horas dedicadas superiores a las estimadas en un principio por errores de principiante.

Se realizaron todas las funcionalides, si bien se descartó la funcionalidad de mediante la geolocalización marcar entrada o salida, por los siguientes motivos:

- Puedes estar de paseo por la zona y el sistema detectar que estás en el lugar y marcar entrada o salida.
- Consumo excesivo de batería.

Se decidió no acabar la implementación de esta pantalla, y se determinó que otras tecnologías, como es el caso de NFC podrían ser más útiles. Por ejemplo, sería útil para marcar entrada o salida pasando el teléfono encima de una etiqueta NFC que podía estar pegada en el lugar de trabajo. Así, al llegar al trabajo simplemente pasando el teléfono de forma consciente encima de una etiqueta NFC, se realizarían las acciones de marcar entrada y a posteriori, realizar los avisos.

En la siguiente imagen se puede ver la pantalla creada de geolocalización que posteriormente se descartó:

En este Sprint se perdió el tiempo dedicado a esta historia, pero se prefirió no continuar y realizar la funcionalidad de NFC.

4.2.3 Sprint 6

En este Sprint se tuvo que realizar, en base a los imprevistos y al tiempo de formación, las historias previstas para el ciclo anterior.

- Creación de acciones automáticas con mensajes personalizados:
  - Avisar a grupos concretos de cuando se entra a trabajar.
  - Notificar cuándo se pretende salir para organizar de forma fácil eventos.

Se detectó que no se había diseñado ni tenido en cuenta que los usuarios también recibirán mensajes de entrada y salida, y no se había creado ni el prototipo, ni se había estimado el tiempo necesario para ello. En este sprint, se añadió una historia a la pila de producto, para poder crear una pantalla en la que se vean los mensajes recibidos de entradas o salidas de otros usuarios.

Además, en vista del retraso acumulado, y de las nuevas historias que iban surgiendo, se tuvo que acortar funcionalidades y alcance de las historias a implementar.

4.2.4 Sprint 7

En este Sprint se realizó la historia de usar la tecnología NFC para marcar entrada o salida. Tras una labor de investigación, se realizó una pantalla que se encarga de grabar un token en una etiqueta NFC y si se detecta este token, se marca entrada o salida.

Tal como se ha indicado, la historia a realizar fue:

- Uso de tecnología NFC para marcar entrada o salida.

4.2.5 Sprint 8

En este ciclo se realizó la historia de la funcionalidad que no se había tenido en cuenta en la estimación que consistía en:

- Recibir notificación de mensajes y mostrar mensajes

4.2.6 Sprint 9

En este último sprint se realizó la siguiente historia:

- Incluir los trabajos realizados en la memoria

Como conclusión final, en este hito se habían estimado realizar 4 Sprints y al final se realizan 2 sprints más. Ello requirió, para cumplir los plazos, realizar los cambios necesarios, además de dedicar más esfuerzo.

La pila de producto al final de este sprint era la siguiente:

Categoría	Historia	Prioridad	Dedicación (días)
Diseño Centrado en Usuario	Usuarios y contexto de uso [Análisis] Diseño conceptual [Diseño] Prototipado [Diseño] Evaluación [Evaluación]	1000	4
Diseño técnico de la aplicación	Definición de los casos de uso	960	5
Diseño técnico de la aplicación	Diseño de la arquitectura de la aplicación	920	2
Documentación	Incluir los trabajos realizados en la memoria	880	2
Desarrollo	Autenticación de usuario	840	1
Desarrollo	Registro de entrada y salida	800	1
Desarrollo	Fusionar pantallas de informacion y fichar	790	3
Desarrollo	Contabilidad de las horas pendientes de realizar	760	1
Desarrollo	Proporcionar información: - A que hora se puede salir - Si ya se han realizado las horas, avisar de que puede salir.	720	2
	Funcionalidad de geolocalización para entrada y salida.		3
Desarrollo	Creación de acciones automáticas con mensajes personalizados: - Avisar a grupos concretos de cuando se entra a trabajar. - Notificar cuándo se pretende salir para organizar de forma fácil eventos.	680	8
Desarrollo	Uso de tecnología NFC para marcar entrada o salida.	640	6
	Recibir notificación de mensajes y mostrar mensajes		3
	Incluir los trabajos realizados en la		

5 Implementación de la aplicación

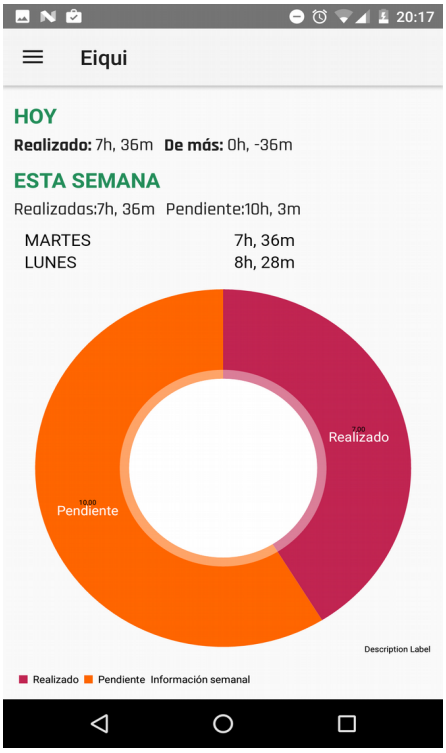
5.1 DESARROLLO

5.1.1 Librerías empleadas

Para el desarrollo de este proyecto se han empleado librerías que permiten incorporar funcionalidades de forma rápida a la aplicación y que simplifican el desarrollo, así como una reducción de costes.

5.1.1.1 Gráficas

Para la realizacion de gráficas se ha usado la librería MPAndroidChart [6], que permite crear gráficas de diferentes tipos. En este caso, se ha usado una gráfica “PieChart” para indicar las horas realizadas y pendientes de la semana actual.



5.1.1.2 Menu lateral

Aunque Android ya proporciona un menú lateral, su uso es algo más complicado que la librería escogida,que es “MaterialDrawer” [7],

5.1.2 Herramienta de gestión de errores

En las aplicaciones móviles, la gestión de los errores y fallos de la aplicación es un proceso complejo, debido a que el desarrollador no puede disponer del terminal que ha generado el error para poder analizar un fallo puntual a un usuario concreto. Sus usuarios pueden estar a cientos de kilómetros, dispersos por todo el planeta. Además, no tiene posibilidad de contactar con ellos, ni estando cercanos, pues es algo que desconoce y que la tienda de aplicaciones no le facilita.

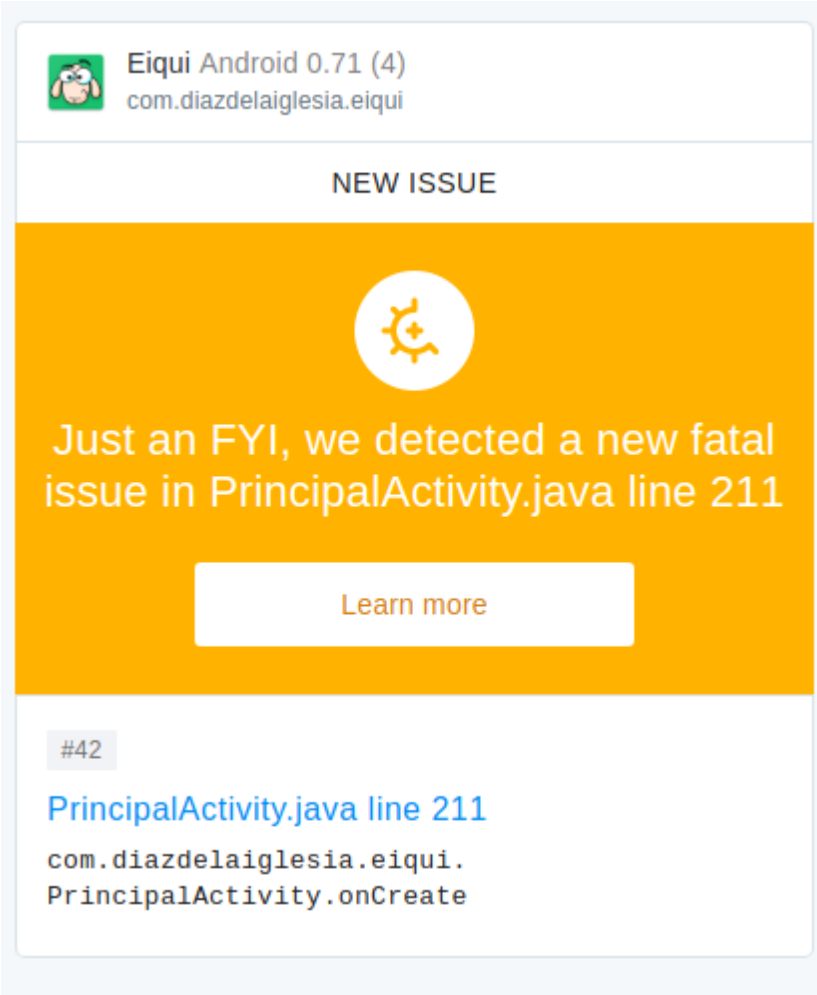
Si bien al publicar la aplicación en Google Play este dispone de una sección para poder ver los errores de la aplicación se ha investigado soluciones de reporting de errores más avanzadas para cubrir las necesidades que la plataforma de Google Play no soporta:

- **Poder disponer de la información y recopilación de errores en el proceso de desarrollo y sin tener que publicar la aplicación en Google Play.** Durante el proceso de desarrollo, la aplicación se ha distribuido a usuarios de prueba, y se necesitaba de una herramienta que pudiese dar información detallada de los errores cuando estos usuarios de prueba usan la aplicación instalándola manualmente. Esto ha permitido tener una gestión centralizada de todos los errores que habían sucedido para poder corregir los bugs. De esta manera se ha eliminado la necesidad de pedirle al usuario que reportase cada uno de los fallos, pues estos se registran automáticamente.
- **Gestión más completa de los errores.** Si bien Google Play muestra los errores de la aplicación, una aplicación de gestión de errores más completa permite realizar filtrado, obtener más información, etc.
- **Independiente de la plataforma:** Si bien Google Play proporcionar una gestión de errores mínima, se quería tener una plataforma común tanto para iOS como para Android, para el caso de que la aplicación, como línea futura, se implementase en iOS, la gestión de errores tuviese un interfaz y punto de entrada común.

Para esta gestión de errores se ha usado Crashlytics [8] de Fabric. Es una solución lanzada en 2014 y que a principios de 2017 ha sido comprada por Google. Es usada por compañías tan importantes como Twitter o Amazon. Su modulo de gestión de errores, Crashlytics ha sido reconocido con el premio de mejor SDK para la estabilidad de aplicaciones, tal como puede verse en “The State of Mobile SDKs in 2016” [9].

Con esta solución, además del panel web desde el que se puede ver toda la información, también envía correos electrónicos, tanto resúmenes diarios,

como correos puntuales de un error concreto. Por ejemplo, el siguiente correo de un fallo de la aplicación:



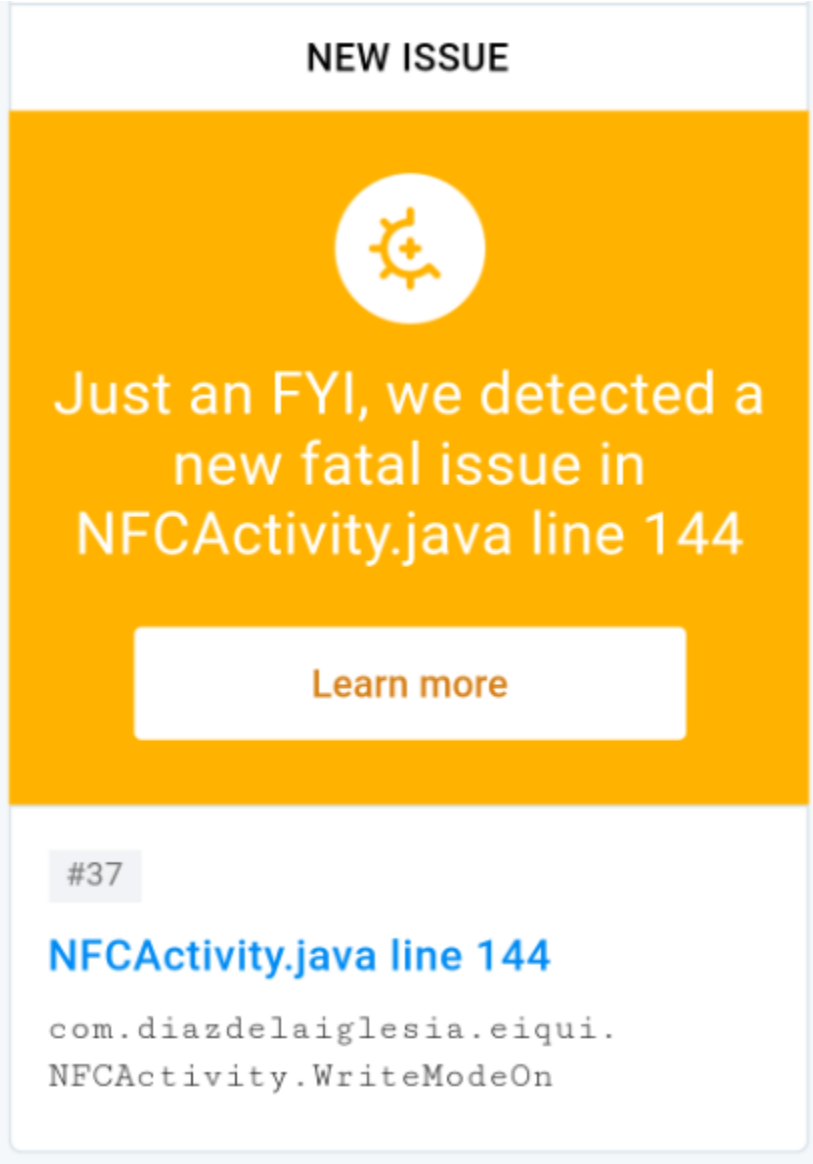


Ilustración 4: Errores - Mensaje recibido por correo

Si pulsamos en ver más, nos lleva al portal web donde podemos obtener más información.

En su panel de administración proporciona información detallada, así como un cuadro de mando, tal como puede verse en la captura de pantalla siguiente:

Panel informativo:

Ilustración 5: Panel de control

La sección de errores de la aplicación ofrece la siguiente información:

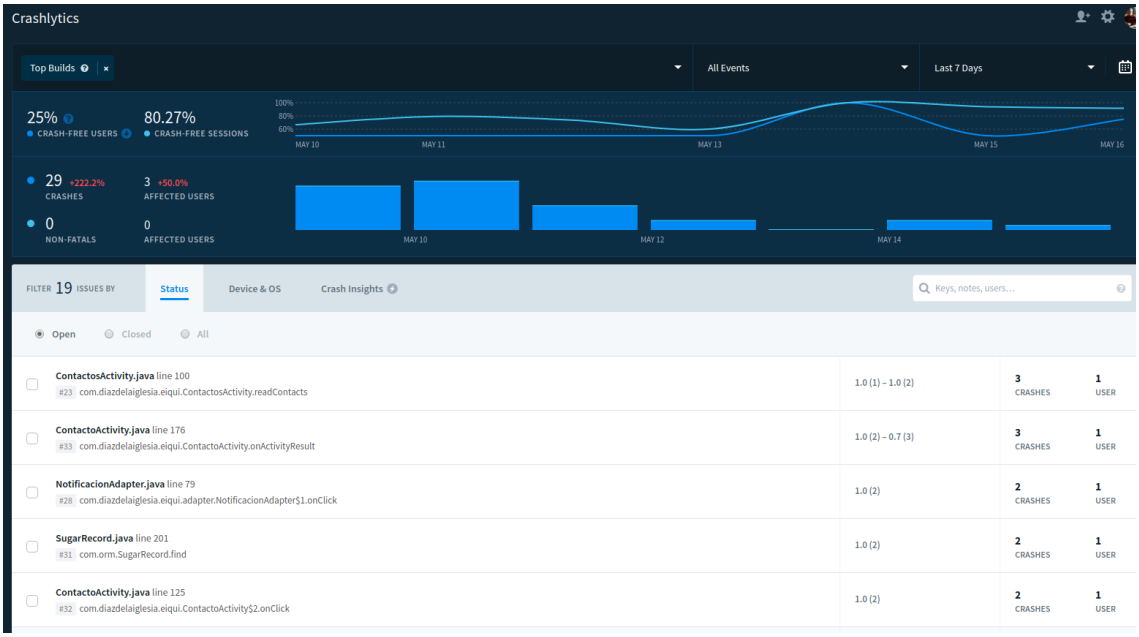


Ilustración 6: Listado de errores

Permite además hacer una gestión de los errores, ya que una vez solucionado un bug, se puede indicar que se ha resuelto.

Para un error concreto, se puede ver información detallada:

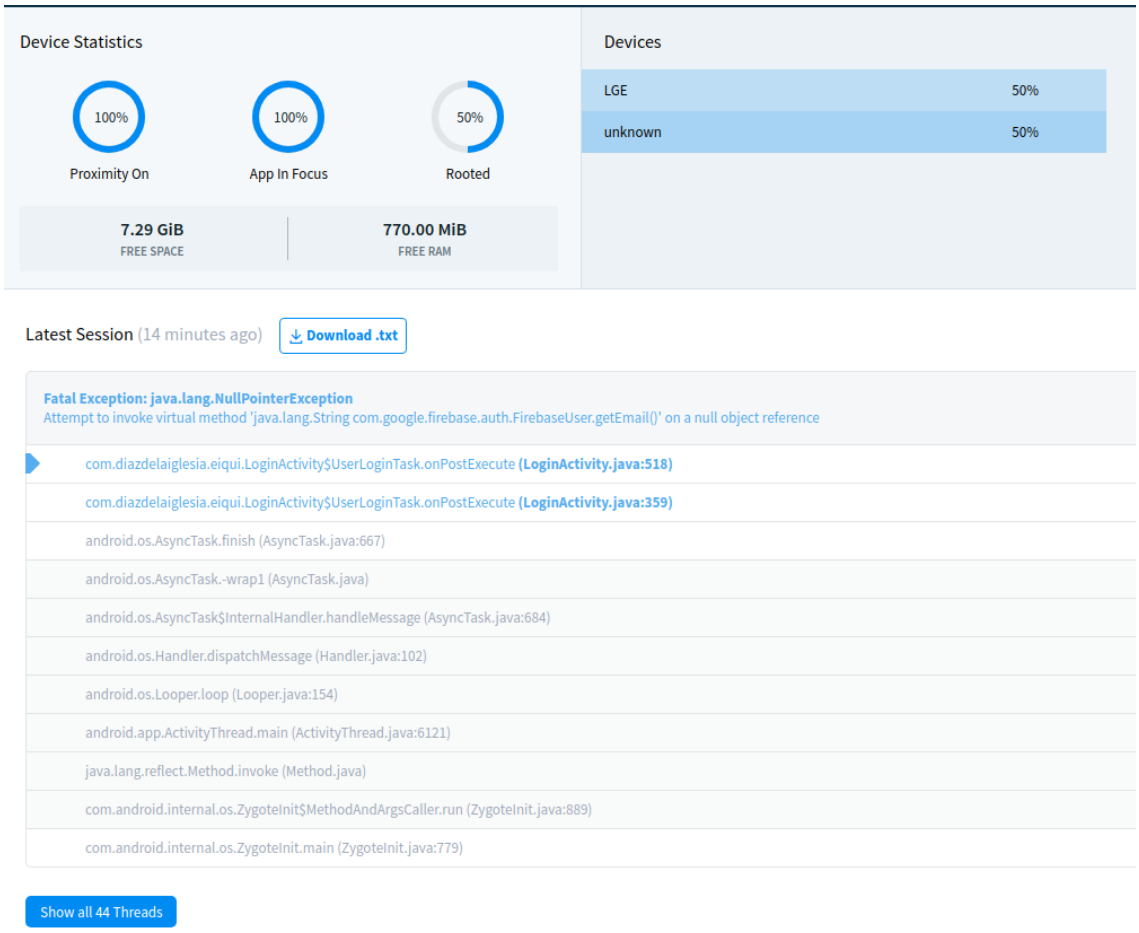


Ilustración 7: Información detallada de un error.

Además de una gestión cómoda y eficaz de los errores, con la información recopilada es capaz de analizar los datos y proporcionar información, por ejemplo de una versión publicada que está produciendo demasiados errores.

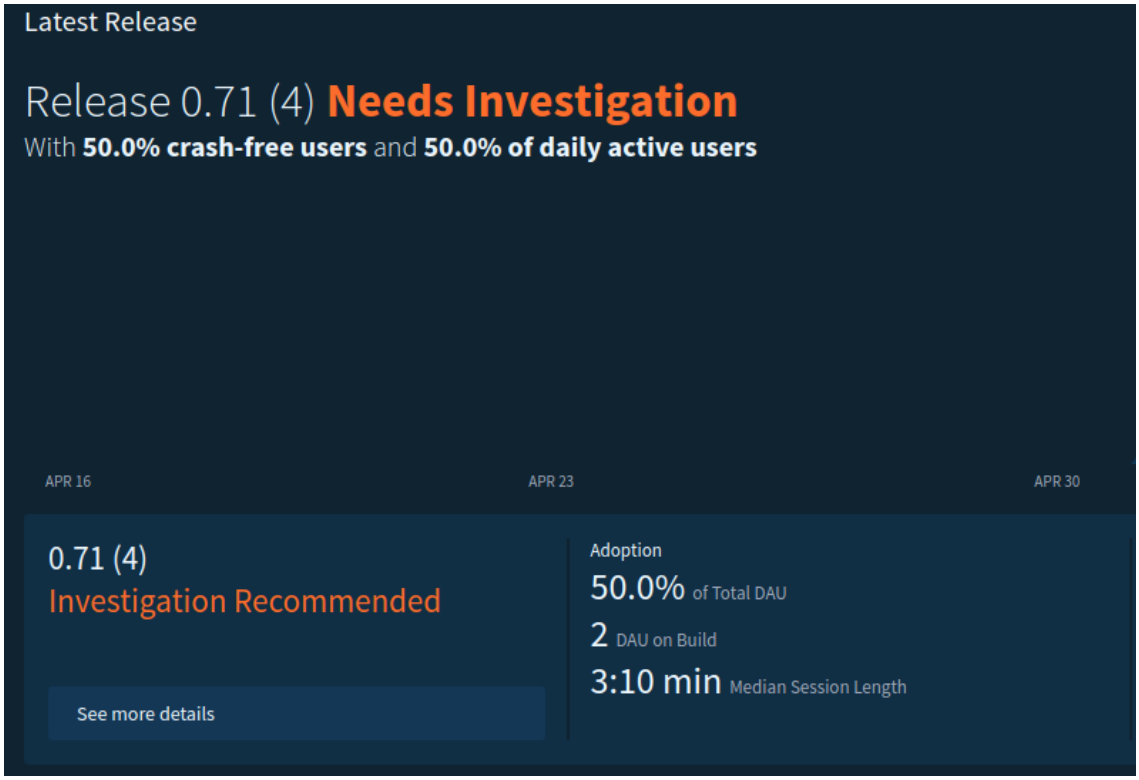


Ilustración 8: Información de que la versión 0.71 está dando muchos fallos

5.1.2.1 Firebase

Firebase [10] , adquirida recientemente por Google, proporciona una serie de funcionalidades que ayudan a crear de forma rápida aplicaciones. En un entorno tan competitivo, es importante sacar un producto al mercado en el menor tiempo posible, sobre todo el mundo de las aplicaciones móviles, en las que otro desarrollador puede sacar una aplicación similar a la que se está desarrollando antes. Como la tienda de aplicaciones es común, el factor tiempo es fundamental para ganar usuarios y que éstos no estén ya en la competencia.

Además de permitir un desarrollo más rápido de aplicaciones, al proporcionar una serie de funcionalidades, con Firebase tenemos un backend que permite no tener que contratar ningún hosting ni programar la parte servidor. En el mundo de las aplicaciones para dispositivos móviles, en las que pequeñas empresas, o desarrolladores individuales crean productos, es importante poder reducir además del tiempo, los costes. En este sentido, Firebase ahorra el gasto de un hosting, y permite escalar de forma sencilla. La versión gratuita es más que suficiente para proyectos, pero si la aplicación fuese un éxito y necesitase de forma rápida crecer, sería posible escalar simplemente eligiendo alguno de los planes que proporcionan, tal como puede verse en la página de precios de firebase, https://firebase.google.com/pricing/:

Products	Spark Plan Generous limits for hobbyists Free	Flame Plan Predictable pricing for growing apps $25/month	Blaze Plan Calculate pricing for apps at scale Pay as you go
Analytics, App Indexing, Dynamic Links, Invites, Remote Config, Cloud Messaging, Authentication, and Crash Reporting.	✓ Included	✓ Included	✓ Included
Realtime Database			
Simultaneous connections	100	Unlimited	Unlimited
GB stored	1GB	2.5GB	$5/GB
GB downloaded	10GB/month	20GB/month	$1/GB
Automated backups	✗	✗	✓
Storage			
GB stored	5GB	50GB	$0.026/GB
GB downloaded	1GB/day	50GB/day	$0.12/GB
Upload operations	20K/day	100K/day	$0.05/10,000
Download operations	50K/day	250K/day	$0.004/10,000
Cloud Functions			
Invocations	125K/month	2M/month	$0.40/million
GB-seconds	40K/month	400K/month	$0.0025/thousand
CPU-seconds	40K/month	200K/month	$0.01/thousand
Outbound networking	Google services only	5GB/month	$0.12/GB
Hosting			
GB stored	1GB	10GB	$0.026/GB
GB transferred	10GB/month	50GB/month	$0.15/GB
Custom domain & SSL	✓	✓	✓

En este proyecto se ha usado las siguientes funcionalidades de Firebase:

- **Realtime database**: Es una base de datos noSQL alojada en la nube. Con el API ha sido sencillo poder almacenar la información en la base de datos en cuestión de poco tiempo. Esta base de datos se ha empleado para guardar los mensajes que se envían de unos usaurios a otros para avisar de sus cambios de estado (entran o salen)
- **Crash Reporting:** Se ha usado inicialmente, para detectar problemas y errores de la aplicación de la app en las diferentes pruebas que han hecho los usuarios. Tras probar y configurar Crashlystics, se ha preferido usar esta última.
- **Authentication:** Solamente es necesario usar la autenticación y registro en la aplicación para acceder a los mensajes que pueda haber recibido el usuario almacenados en la base de datos en tiempo real. Entre las ventajas que tiene usar la autenticación de Firebase, además de ahorrar tiempo en programar en el servidor, es que se puede elegir diferentes modos de autenticación y registro. Por ejemplo, se puede usar métodos tradicionales de autenticación mediante correo y contraseña, pero también, se puede activar, autenticar con Google, Facebook, etc. Esto permitirá que la aplicación pueda crecer de forma rápida adaptándose a las necesidades.
- **Notifications:** Sistema para enviar notificaciones push a los móviles. Se usará para enviar avisos a los destinatarios de mensajes cuando no tengan la aplicación abierta.

5.1.3 Autenticación

Para realizar la autenticación se ha optado por usar la plataforma Firebase, con ello se agiliza el desarrollo:

- No es necesario hacer la parte de servidor de autenticación, ahorrando tiempo y posibles bugs.
- Se simplifica la parte del cliente.

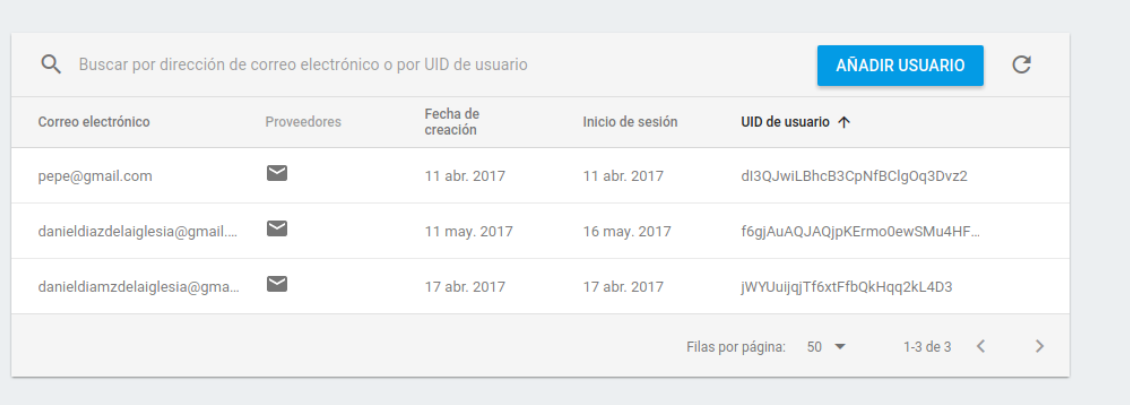


Ilustración 9: Usuarios autenticados usando Firebase

También permite adaptarse a nuevas necesidades, puesto que soporta como proveedores de inicio de sesión los siguientes:

- Autenticación correo y contraseña
- Google
- Facebook
- Twitter
- GitHub
- Anónimo

De esta manera, a futuro, o como mejora continua de la aplicación, se podrán incluir nuevos métodos de autenticación de forma sencilla, simplemente activándolos en el backend y haciendo pequeños cambios en la parte de cliente.

5.1.4 Almacenamiento de la información

5.1.4.1 SharedPreferences

Se ha usado SharedPreferences para almacenar información de configuración. Como es el caso de la hora de salida si se ha establecido manualmente.

5.1.4.2 Firebase

Con el objetivo de minimizar la información a guardar en Firebase, y de esta manera que la aplicación, sólo para las funcionalidades concretas, deba tener una conexión a internet, en Firebase se ha guardado los mensajes a intercambiar. De tal manera que cuando se hace un cambio de estado se guarda en Firebase los mensajes, y los destinatarios reciben dichos mensajes cuando se conectan.

La información del mensaje contiene la siguiente información:

- destinatario: un correo electrónico.
- Fecha del mensaje.
- Id del mensajes
- Si se ha leído o no. Cuando se recibe un mensaje, la aplicación en el puesto cliente la marca como leída.
- Remitente: quién ha enviado el mensaje.
- Texto: El texto está compuesto por la notificación que ha creado el usuario.

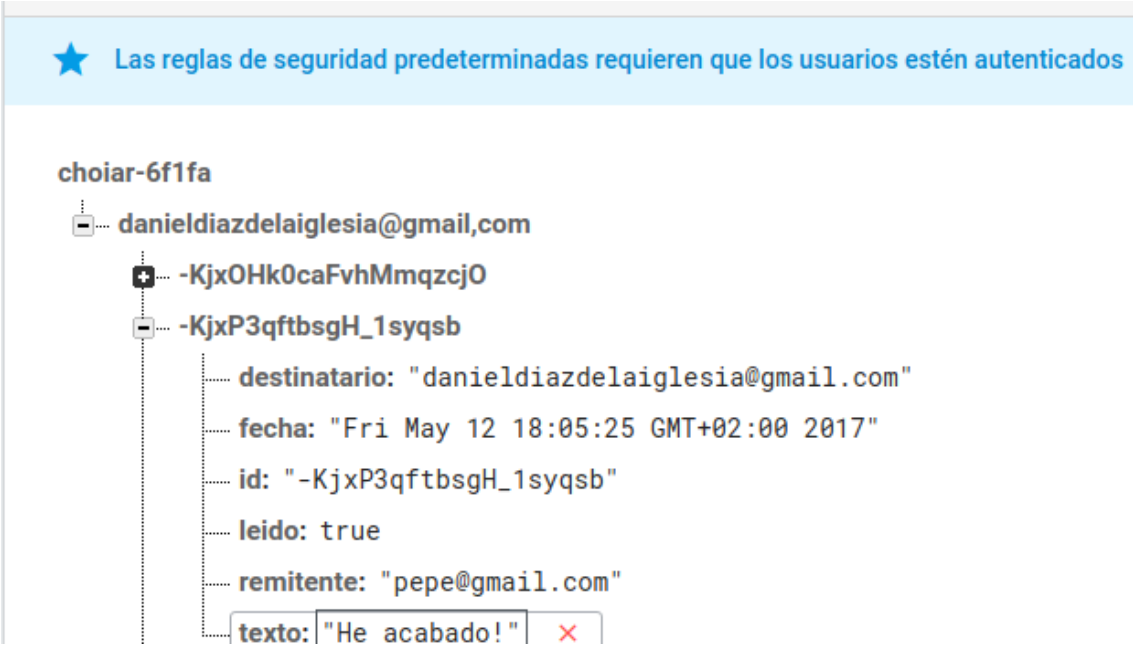


Ilustración 10: Datos de los mensajes almacenados en firebase.

5.1.4.3 Sugar ORM

Se ha usado Sugar ORM para simplificar el acceso a los datos locales. En este caso concreto, el teléfono tiene en local la base de datos de la aplicación, salvo los mensajes enviados, que de esa parte se encarga Firebase. Para el resto de datos (registro de entrada y salida, horario, notificaciones) toda esa información está disponible localmente. Con Sugar ORM se simplifica el acceso a la base de datos, simplemente creando una clase objeto que extienda de SugarRecord, ya es posible guardarla en la base de datos local de forma transparente. Las clases empleadas en este proyecto que se han usado con Sugar ORM son:

- NotificacionItem. La clase Notificaciones se encarga del acceso a esta información (ej: realizar la búsqueda de NotificacionItem por id)
- HorarioItem. Encargándose del acceso la clase Horarios.
- EstadoItem. Encargándose del acceso la clase Estados
- ContactoItem. Encargándose del acceso la clase Contactos.
- CambioEstadoItem. Encargándose del acceso la clase CambioEstados.

5.2 PRUEBAS

Las pruebas buscan como objetivo obtener la máxima calidad del producto en el que se está trabajando. Para eso, deben poder determinar que el producto desarrollado cumple con los requisitos iniciales, es decir que se entrega lo que realmente se pidió. Además, intenta encontrar errores de funcionamiento o anomalías antes de que el producto llegue al usuario final.

Uno de los problemas que tiene la fase de pruebas en los métodos tradicionales de la ingeniería de software es que suelen ser una de las últimas partes del proceso (ejemplo: metodología en cascada), por lo que los posibles retrasos del proyecto, y la consecuente perdida de rentabilidad, se suelen solucionar acortando tiempo a las pruebas, lo que produce un producto que puede no estar bien chequeado. Otro de los problemas de la fase de pruebas es que los propios programadores, o los integrantes del equipo de programación, no les dan la suficiente importancia y suelen aplicar el “método de pruebas de Pandora”, que consiste en poner en producción el software sin probar y esperar a que el propio usuario descubra por sí mismo las deficiencias y las reporte al equipo. Este método provoca que el producto entregado no cumpla con los mínimos de calidad que debería tener y de una mala imagen del equipo de programación. Muchas veces, las pruebas más complicadas de una aplicación, se suelen delegar al propio usuario por no conocer exactamente cómo funciona la aplicación. Esto ocurre porque los equipos de programación no suelen tener una visión global del producto ni una idea de cómo funciona la aplicación (Ej: Un programador puede codificar un formulario de contratación en el CRM de una empresa de telecomunicaciones, pero no sabe como realizar el proceso de contratación concreta para su prueba y así verificar que está bien). Este problema se resuelve en las metodologías ágiles, pues en ellas se intenta que el equipo de programación esté alineado con la empresa, teniendo una visión global del producto y de su funcionamiento. Es mucho más seguro no tener errores si a parte de saber lo que tenemos que hacer para dar una nueva funcionalidad, también sabemos para qué sirve esa funcionalidad y como se emplea.

Como consecuencia de utilizar la metodología ágil Scrum en este proyecto, en cada ciclo, se aplicó un plan de pruebas antes de cerrar el ciclo de desarrollo y con herramientas de gestión de errores, se recopiló la información obtenida del error para corregirlo.

- **Pruebas unitarias:** Durante cada uno de los ciclos de desarrollo hasta obtener el producto final, se realizaron pruebas unitarias en cada nueva funcionalidad incorporada.

- **Pruebas de integración:** También se realizaron pruebas de integración para verificar que los módulos probados por separado, entre sí y en conjunto, el sistema funcionaba de acuerdo a las especificaciones.

- **Pruebas de aceptación:** Las pruebas de aceptación se realizaron al finalizar cada ciclo. Estas pruebas buscan probar que las nuevas funcionalidades están acordes a lo esperado por el dueño del producto.

También se realizaron pruebas de usabilidad con usuarios finales, para eso se escogió dos perfiles de usuario:

- **Usuario no técnico**: Un usuario que desconoce este tipo de aplicaciones y tecnologías, y que es importante sus pruebas para saber sí la aplicación es lo suficientemente intuitiva.

- **Usuario técnico**: Aquel usuario que está acostumbrado al uso de este tipo de aplicaciones y está familiarizado con esta tecnología.

A estos usuarios se les facilitó un banco de pruebas con el que trabajar. En el caso del usuario no técnico se le entregó una documentación mas detallada con pruebas muy definidas.

Estas pruebas de usabilidad permitieron en cada ciclo obtener retroalimentación de la experiencia de usuario y hacer los cambios necesarios a la aplicación hasta llegar a una versión de aplicación totalmente usable.

En lo que respecta a la aplicación móvil, se han realizado pruebas reales sobre un Nexus 5x, sin menoscabar aquellas pruebas realizadas durante todo el proceso de desarrollo con los emuladores que proporciona el SDK.

5.2.1.1 Detalle de pruebas realizadas

Tal como se ha indicado anteriormente al usar Scrum, en cada ciclo de desarrollo se realizaron pruebas exhaustivas, se realizaron las pruebas unitarias por cada nueva programación realizada, las pruebas de integración al finalizar cada historia y las de aceptación en la entrega del producto creado en cada ciclo. A continuación se detallan las pruebas realizadas correspondientes con los casos de uso.

Tarea	Ver información de la jornada laboral
Descripción	Visualizar información de la jornada actual y las horas que lleva en la semana actual.
Resultado esperado	El sistema muestra la información diaria y semanal registrada, así como las horas pendientes. En un gráfico además, se le da una información más visual.
Resultado de la prueba	Correcto.

Tarea	Indicar Entrada
Descripción	Pulsar en el botón de entrar para marcar entrada
Resultado esperado	El botón cambia a “salir” y se indica la fecha de salida prevista.
Resultado de la prueba	Correcto.

Tarea	Indicar Salida
Descripción	Pulsar en el botón de salir para marcar salida, cuando previamente se ha marcado entrada.
Resultado esperado	El botón cambia a “entrar” y se muestra el tiempo realizado.
Resultado de la prueba	Correcto.

Tarea	Cambio de estado mediante NFC
Descripción	Pasar el móvil encima de una etiqueta NFC activada.
Resultado esperado	Se aplica un cambio de estado, como si se hubiese pulsado en el botón.
Resultado de la prueba	Correcto.

Tarea	Aviso automático
Descripción	Al tener marcado aviso automático, y alguna notificación creada, se pulsa en entrar o salir
Resultado esperado	Se realiza automáticamente avisos a las notificaciones creadas.
Resultado de la prueba	Correcto.

Tarea	Cambiar hora de entrada
Descripción	Pulsar en el botón de indicar otra hora de entrada.
Resultado esperado	Se muestra otra hora de entrada al pulsar entrar.
Resultado de la prueba	Correcto.

Tarea	Cambiar hora de salida
Descripción	Pulsar en el botón de indicar otra hora de salida.
Resultado esperado	Se muestra otra hora de salida.
Resultado de la prueba	Correcto.

Tarea	Cambiar la configuración - Notificaciones
Descripción	Pulsar en el menú lateral, luego en notificaciones y dar de alta una notificación.
Resultado esperado	Se crea una nueva notificación.
Resultado de la prueba	Correcto.

Tarea	Cambiar la configuración - Horarios
Descripción	Pulsar en el menú lateral, luego en Horarios y cambiar las horas a realizar para un día concreto.
Resultado esperado	Los cambios guardados se registran.
Resultado de la prueba	Correcto.

Tarea	Cambiar la configuración - NFC
Descripción	Pulsar en el menú lateral, luego en NFC, poner el móvil encima de una etiqueta NFC y pulsar en el botón activar.
Resultado esperado	La etiqueta NFC recibe y almacena la información.
Resultado de la prueba	Correcto.

6 Aplicación desarrollada

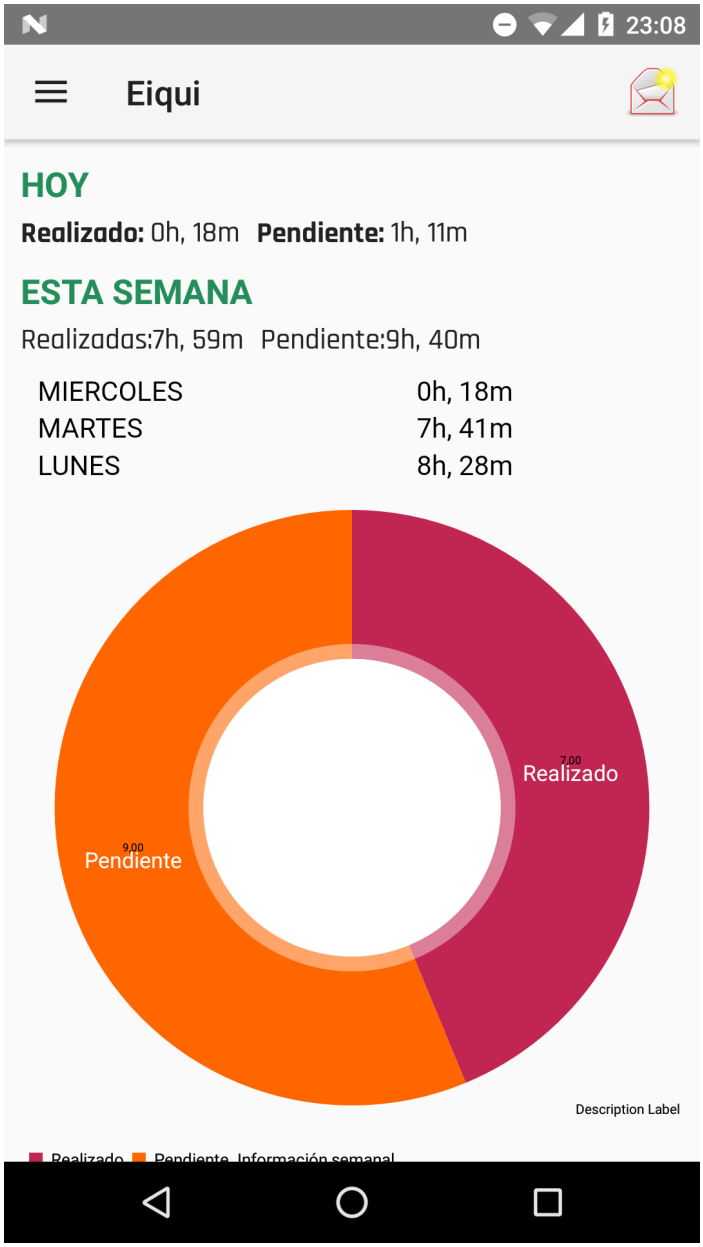
6.1 PANTALLA PRINCIPAL

En esta pantalla de puede ver un mensaje de ánimo, porque aún quedan horas por realizar.

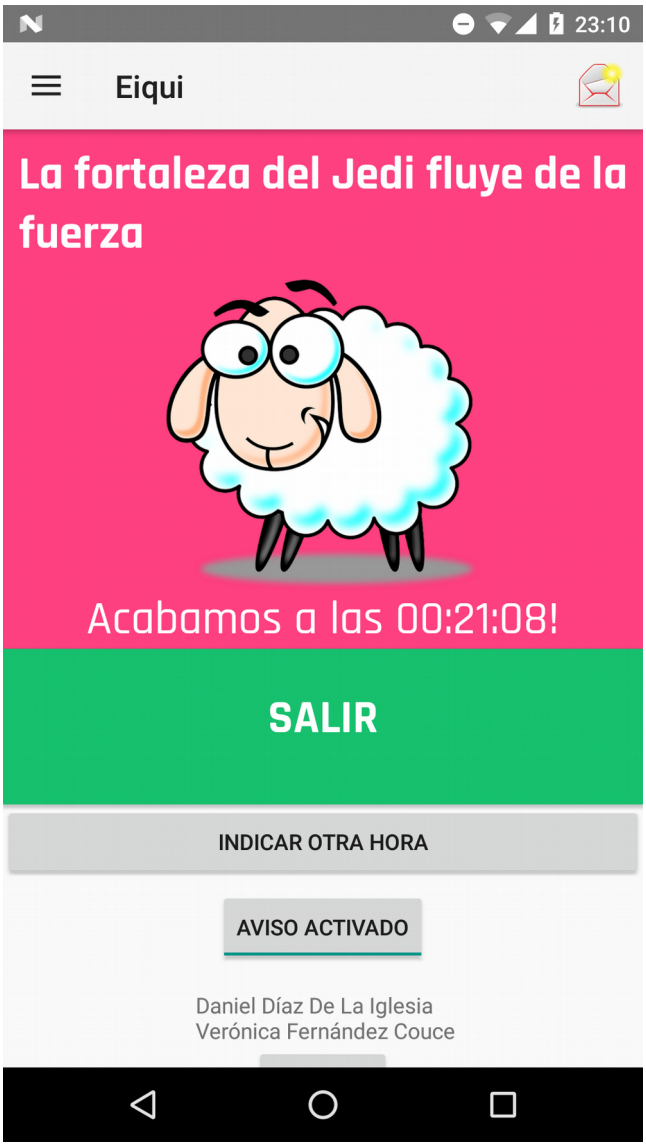
También aparece el botón de Entrar, que deberá pulsar para marcar entrada. Está marcado por defecto “aviso automático” y debajo se ven los contactos a los que avisará. Además, le permite, pulsando en nuevo, añadir más contactos.

En la parte superior derecha, se ve que le han llegado avisos de otros contactos que han marcado entrada o salida.

Desplazándose en la propia pantalla hacia abajo aparece información detallada:

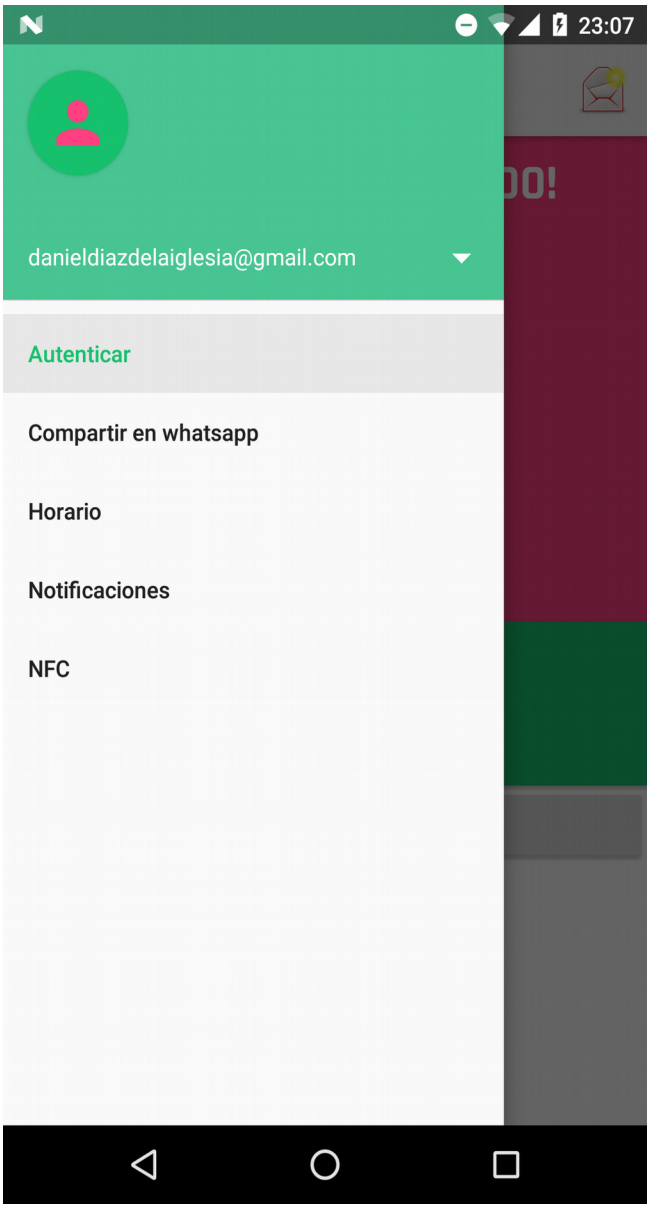


Si marcamos salida, vemos como el mensaje cambia, como aún quedan horas por realizar, proporciona un aviso de la hora prevista.



6.2 MENÚ LATERAL

En el se ven las opciones que dispone el usuario



6.3 MENSAJES RECIBIDOS

La pantalla de mensajes recibidos muestra los mensajes que se han recibido de otros contactos.

6.4 REGISTRO DE HORAS REALIZADAS

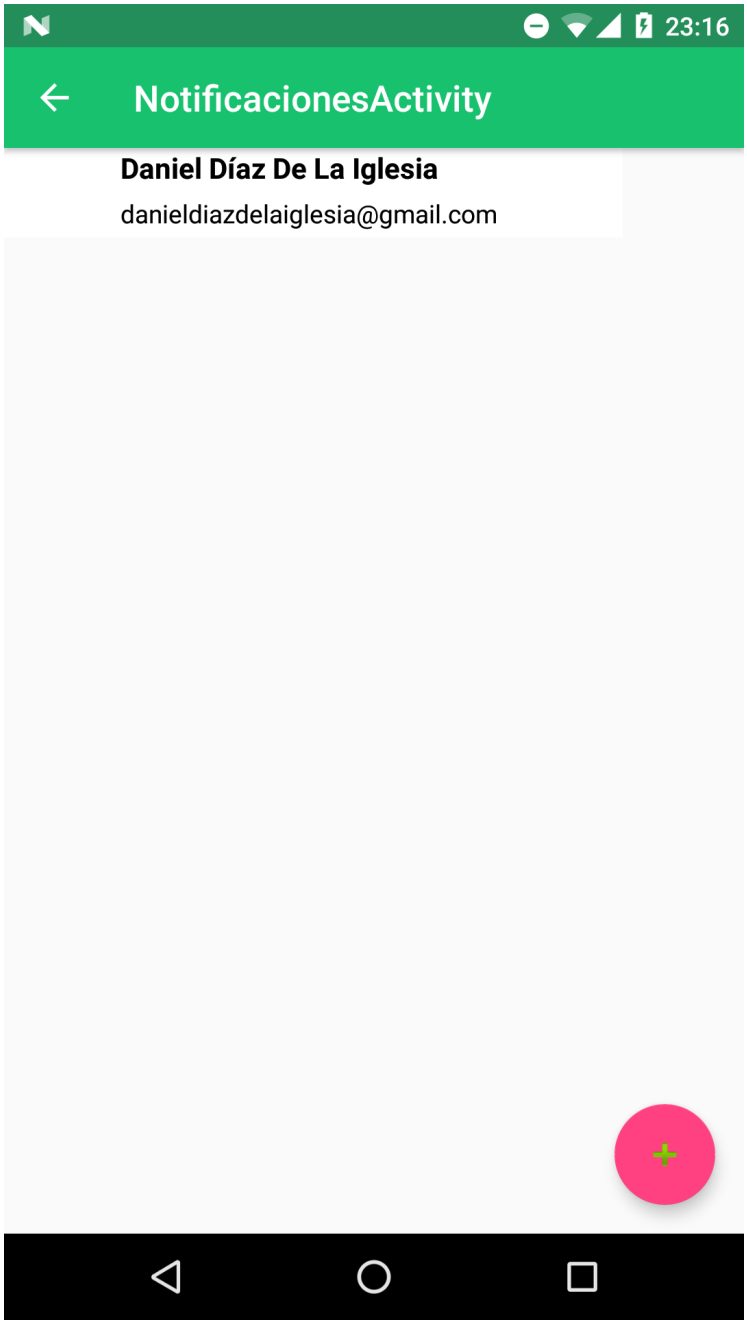
La pantalla de registros permite visualizar los registros de todas las horas realizadas:

6.5 PANTALLA DE HORARIOS

Se pueden marcar los días que son hábiles, así como las horas necesarias:

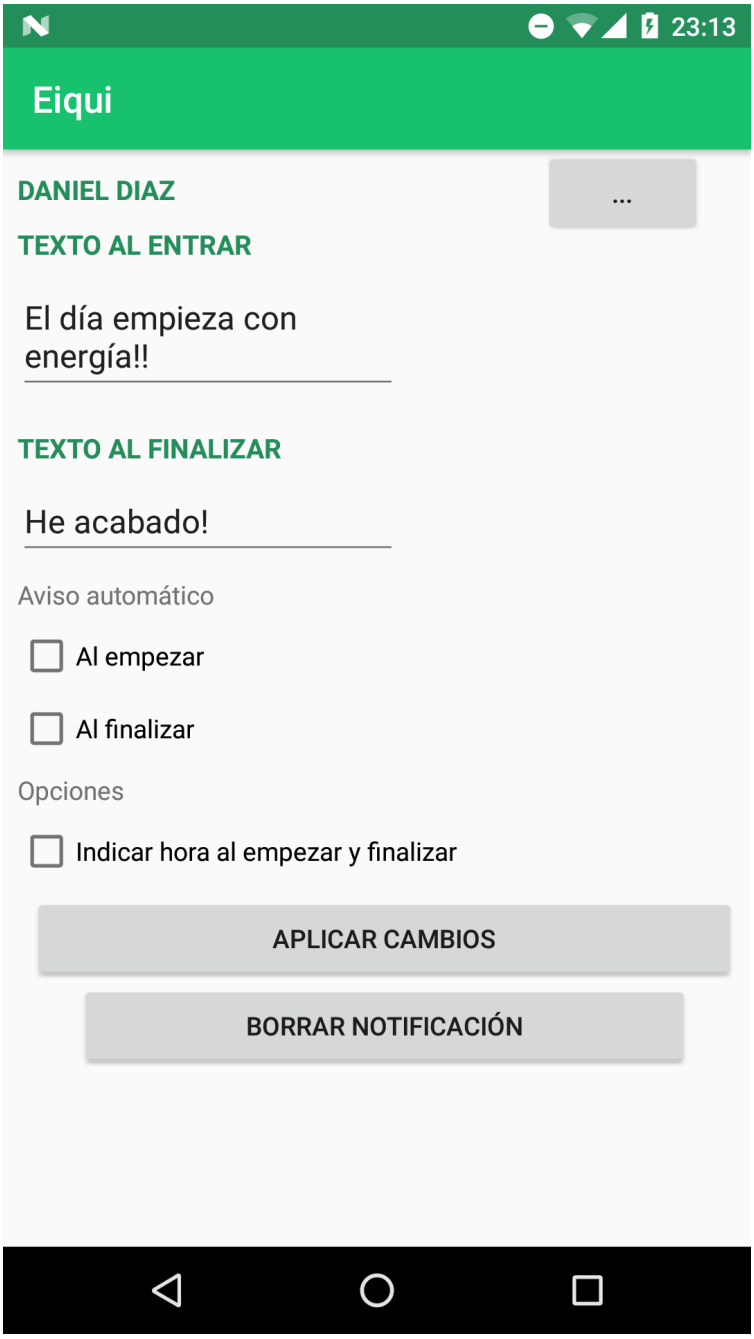
6.6 PANTALLA DE NOTIFICACIONES

Aparecen las notificaciones dadas de alta, y si se pulsa en una de ellas, se puede configurar.
Permite además añadir notificaciones.



6.7 ALTA O MODIFICACIÓN DE NOTIFICACIÓN

Permite configurar la notificación



6.8 NFC

La pantalla de activación de tarjeta NFC permite configurar una tarjeta NFC para poder ser usada por la aplicación para marcar automáticamente entrada y salida.

6.9 NFC

La pantalla de activación de tarjeta NFC permite configurar una tarjeta NFC para poder ser usada por la aplicación para marcar automáticamente entrada y salida.

6.10 APLICACIÓN PARA ANDROID WEAR

Para aprovechar las posibilidades de los dispositivos inteligentes, se ha desarrollado una aplicación para relojes inteligentes, que mediante las APIS de google se comunican con la aplicación principal.

En concreto, esta aplicación para android wear, mediante un botón, lanza de forma automática la acción predeterminada. Si está entrando, avisará de la entrada, como si se usase la tarjeta NFC o se pulsase directamente en el botón.

7 Conclusiones

Durante la realización de este proyecto se han aplicado conocimientos adquiridos durante el Master. Las asignaturas de Android han sido fundamentales para coger soltura en la programación en este lenguaje. Pero en el mismo nivel de importancia, la asignatura sobre prototipado ha demostrado su utilidad en permitir de una forma previa al desarrollo poder tener una visión más clara de la aplicación de qué cosas funcionan y cuáles se deben mejorar.

Con esas bases, en este proyecto se han abordado retos de aprendizaje, como es aprender el uso de soluciones como Firebase o Sugar ORM. Firebase ha permitido el desarrollo más rápido de aplicaciones y no tener que depender de un servicio de hosting. De esta manera, un desarrollador individual puede poner en el mercado una aplicación sin tener que asumir costes recurrentes todos los meses.

Con Sugar ORM, simplifica la persistencia de datos, sin tener que tener que programar el acceso a la base de datos a más bajo nivel.

Respecto a los objetivos planteados, estos consistían en el desarrollo de una aplicación que permitiese tener información de las horas realizadas en el trabajo y poder realizar avisos automáticos. Este objetivo se ha conseguido, con una aplicación amena de usar. Se han aprovechado las posibilidades de los dispositivos móviles, tal es el caso de NFC y de disponer de una aplicación para relojes con Android Wear.

Como reflexión, cuanto más tiempo se hubiese dedicado al diseño, por ejemplo en trabajar más sobre los prototipos, sería posible haber detectado cambios que se hicieron a posteriori. Por ejemplo, en el diseño inicial se contemplaba el uso de la geolocalización para detectar la entrada y salida. Se ha visto que no es tan práctico y que puede suponer un consumo de batería. Si se hubiese analizado mejor esto en la etapa de desarrollo, no se hubiese invertido tiempo en el desarrollo para realizarlo y luego descartarlo. En cambio, se ha visto en la fase de desarrollo que sí que es práctico el uso de NFC y se ha aplicado para que sea más cómodo marcar entrada y salida. Cuanto más tiempo se invierta en fases previas, como es el diseño, será mejor para evitar luego desperdiciar el tiempo de desarrollo en funcionalidades que se acabarán descartando.

Ha sido costoso seguir la planificación y ajustarse a los plazos. El proceso de aprendizaje de soluciones como Firebase ha supuesto un tiempo extra que no estaba contemplado. En la planificación se debió tener en cuenta aquellos casos de formación necesaria.

Gracias a usar la metodología ágil Scrum, en cada iteracción hemos podido corregir desviaciones y entregar un producto en plazo acorde a las necesidades del usuario, así:

- En las entregas de la aplicación, se detectaron funcionalidades que no eran tan interesantes como en un principio. Se descartaron.
- También se detectaron otras necesidades, como es el NFC, que sí son útiles.

- Ha permitido adelantarse a los desvíos, repriorizando historias o descartando otras no tan importantes para el producto y así entregar el producto en plazo.

Con Scrum se tiene una comunicación más fluida con el dueño de producto, pudiendo conocer mejor sus necesidades, y esté, con sus prioridades, poder realizar cambios para tener el mejor producto, en el tiempo pactado.

8 Lineas futuras

Como líneas futuras a realizar para la aplicación, se contemplan las siguientes:

- Añadir más métodos de autenticación. Al usar Firebase, será sencillo añadir otras autenticaciones, como puede ser la de Google, para simplificar el uso de la aplicación por parte del usuario.
- Mejorar la “gamificación” de la aplicación. Ahora la aplicación muestra un mensaje de ánimo diferente si aún no se ha entrado, si se está en medio de la jornada, o si se ha salido. Sería más divertido para el usuario, que mostrase mensajes diferentes en cada caso.
- También sería interesante integraciones con otros sistemas, como puede ser con Telegram, para que la aplicación permitiese interactuar con las diferentes plataformas de mensajería.

9 Glosario

- Android: Sistema operativo para dispositivos móviles creado por Google
- Scrum: Metodología ágil.
- Weareable: Literalmente traducido como “tecnología vestible” son aquellos dispositivos electrónicos que se lleva en el cuerpo, tal como puede ser un reloj inteligente, pulseras de actividad, etc, y que interactáun normalmente con el dispositivo móvil para enviar o recibir información.
- Ludificación: Es el uso de técnicas, propias de los juegos, con el fin de potenciar la motivación, convirtiendo una tarea rutinaria aburrida en un algo divertido.
- Gamificación: Anglicismo, sinónimo de Ludificación.

10 Bibliografía

10.1 PROTOTIPADO

[1] https://www.ninjamock.com

10.2 ANDROID

[2] https://ww.android.com

[3] http://developer.android.com/intl/es/index.html

10.3 ANÁLISIS DE USO DE SISTEMAS OPERATIVOS DE ANDROID

[4] http://www.kantarworldpanel.com/global/smartphone-os-market-share/intro

10.4 METODOLOGÍAS ÁGILES

[5]Scrum y XP desde las trincheras http://www.proyectalis.com/wp-content/uploads/2008/02/scrum-y-xp-desde-las-trincheras.pdf

10.5 LIBRERIAS

[6] https://github.com/PhilJay/MPAndroidChart

[7] http://mikepenz.github.io/MaterialDrawer/

[8] https://fabric.io/kits/android/crashlytics

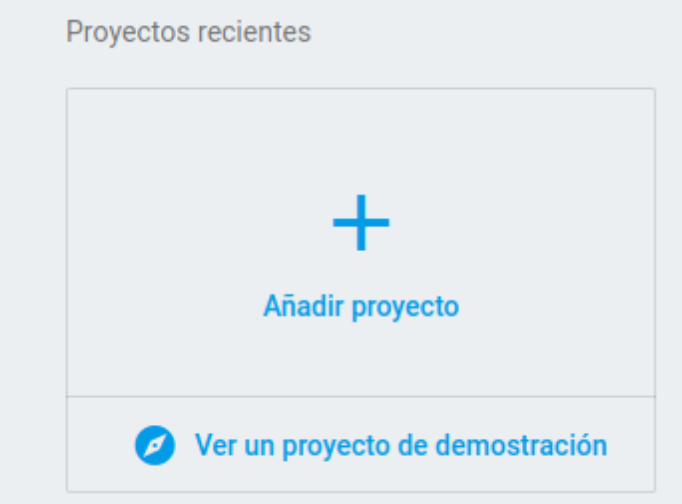
[9]_https://blog.mightysignal.com/the-state-of-mobile-sdks-in-2016-6007a13d8546

[10] https://firebase.google.com/

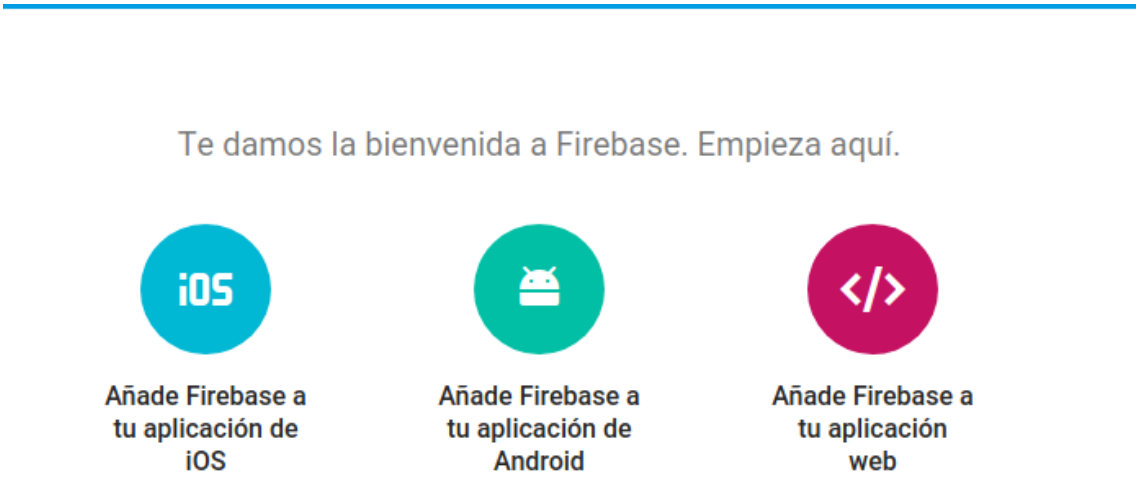
11 Anexos

11.1 CONFIGURACION DE FIREBASE

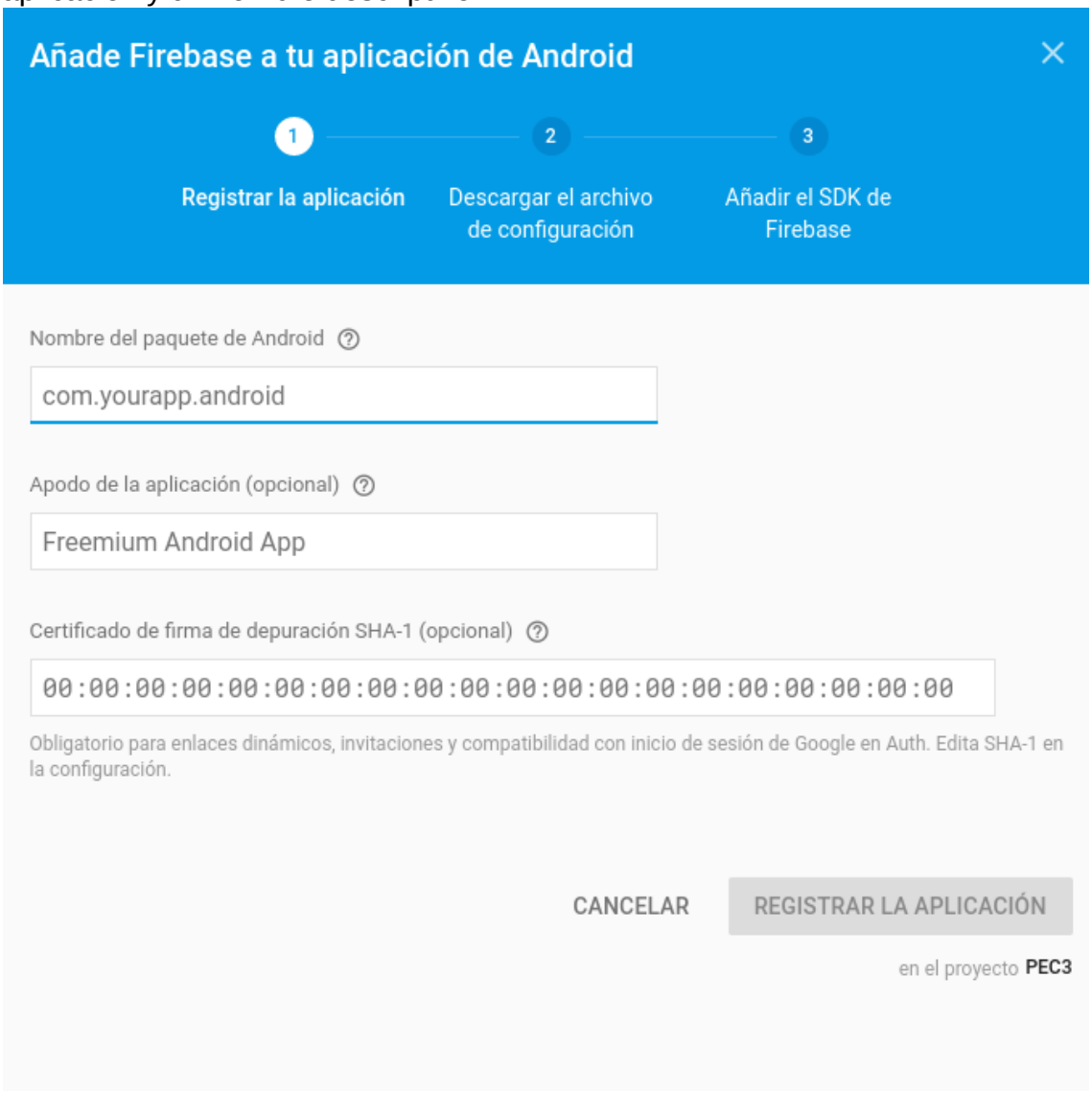
Una vez autenticados en Firebase, se debe crear un nuevo proyecto, pulsando en “Añadir Proyecto”:



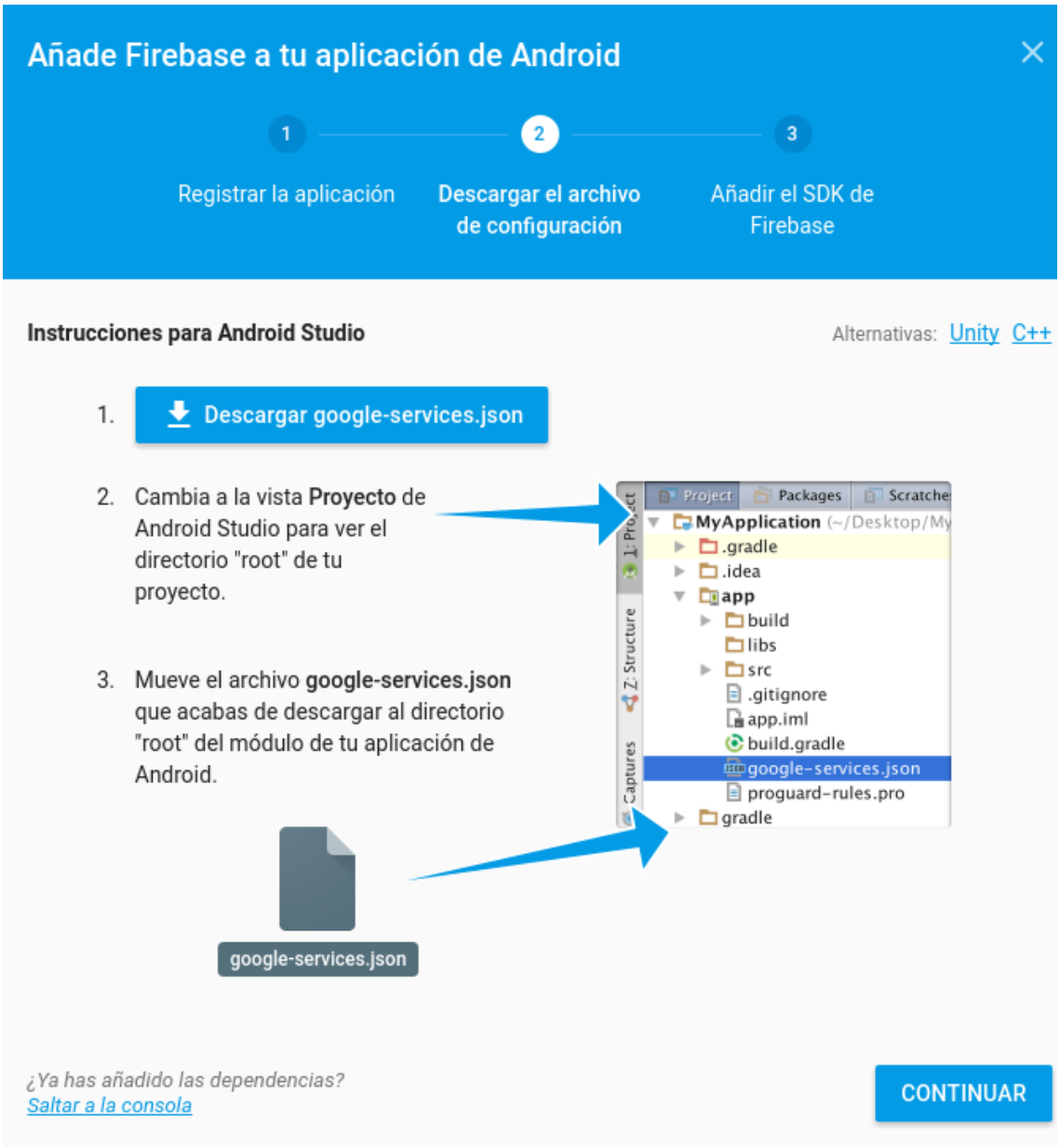
Pulsamos en “añade Firebase a tu aplicación de Android”:



La pantalla siguiente nos pide que introduzcamos el paquete de nuestra aplicación y un nombre descriptivo:



Una vez indicado el paquete, el siguiente paso es descargar el fichero json de configuración y agregarlo al proyecto, tal como indica en la imagen siguiente:



Tal como indica:

- Descargar json
- Cambiar a vista de proyectos
- Dejar el fichero en la carpeta raíz.

El siguiente paso, consiste en dejar en el fichero build.gradle del proyecto la siguiente configuración

```
buildscript {
   dependencies {
      // Add this line
      classpath 'com.google.gms:google-services:3.0.0'
   }
}
```

En el build.gradle de la aplicación se debe añadir la siguiente configuración

```
// Add to the bottom of the file
apply plugin: 'com.google.gms.google-services'
```

Una vez aplicados los cambios, se debe sincronizar.

Con esto, firebase ya estará configurado para poder ser usado en la aplicación Android.

11.2 CONFIGURAR LA APLICACIÓN EN ANDROID STUDIO

Para poder usar el codigo fuente y compilar la aplicación, es necesario realizar los siguientes pasos:

1. Importar la aplicación en Android Studio.
    1. Ir a File → Open
    2. En el diálgo, buscar la carpeta donde está el proyecto de android studio.
    3. Sobre la carpeta del proyecto pulsar en OK.

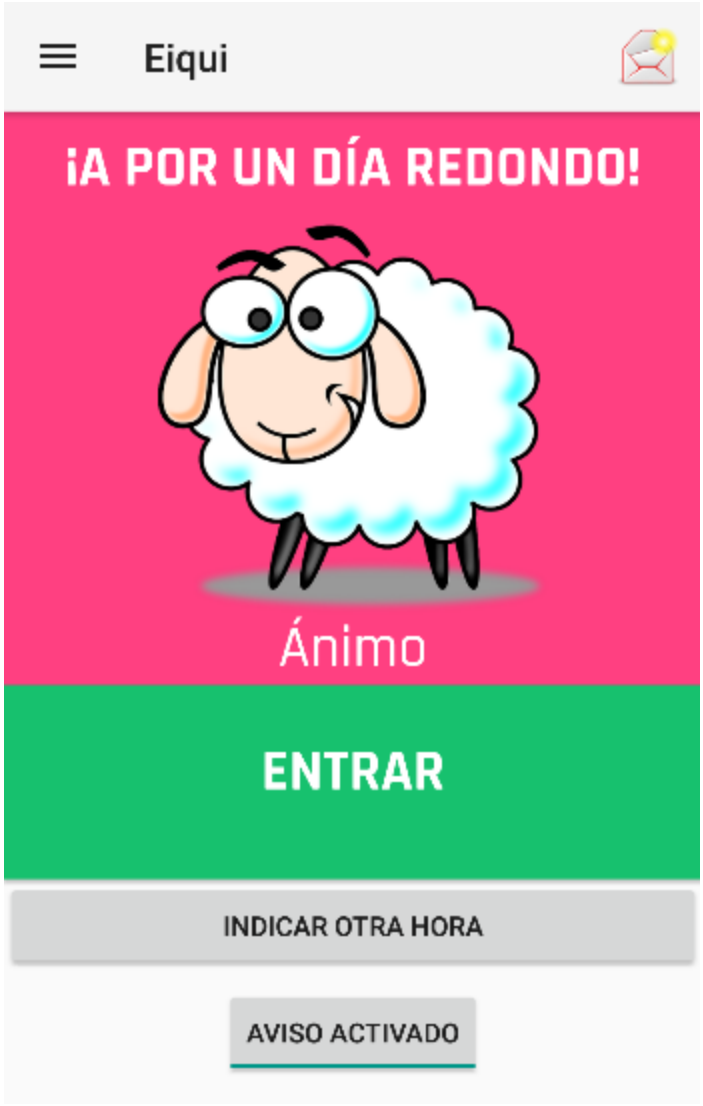
2. En Tools → Android → SDK Manager verificar que están las versiones de Android necesarias:
    1. Para la aplicación de teléfono: Api level 25
    2. Para la aplicación de wear: Api level 25.

3. Si no se tiene dispositivo físico y se desea emulador, es necesario descargar los emuladores de Android 7.1. Para ello hay que ir a Tools→ Android → AVD Manager.

4. Compilar y lanzar la versión móvil o la versión Wear.

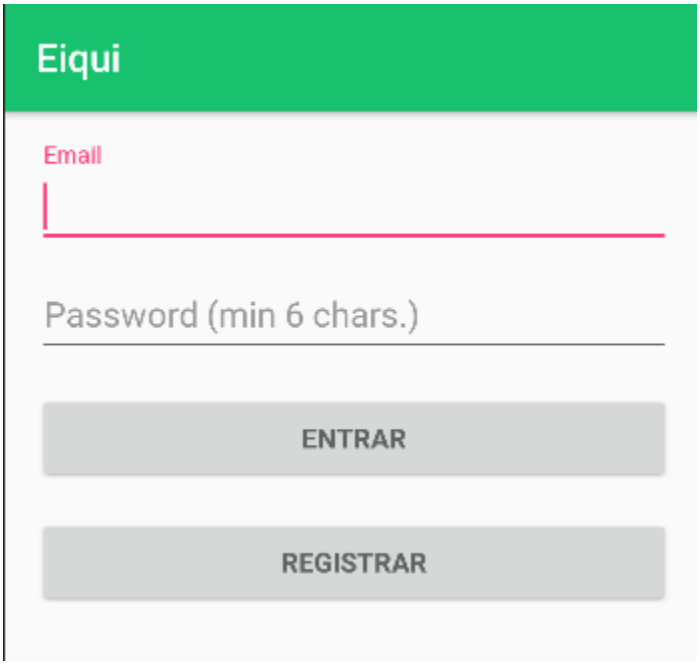
11.3 MANUAL DE USUARIO

11.3.1 Aplicación móvil



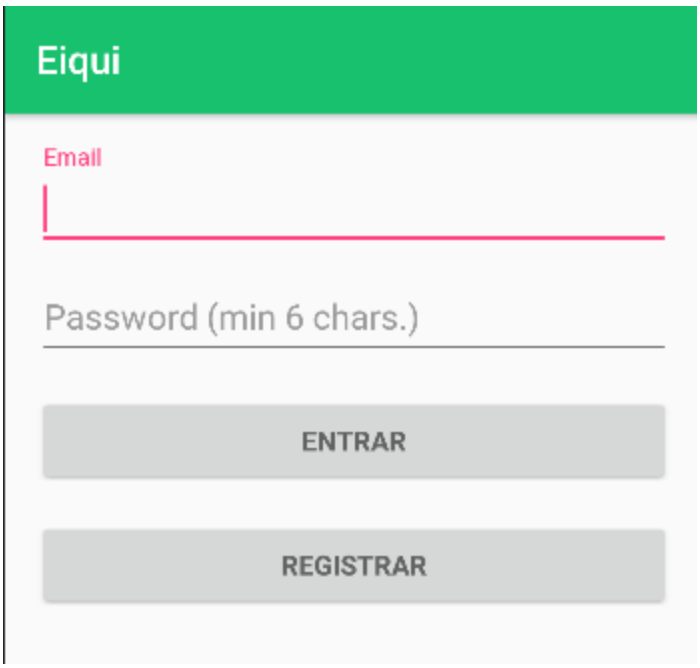
11.3.1.1 Registro

El registro en la aplicación es mediante usuario y contraseña. Para hacerlo, se debe ir al menú, y allí pulsar en autenticar. Cubrimos usuario y contraseña y pulsamos en “REGISTRAR”.



11.3.1.2 Autenticar

se debe ir al menú, y allí pulsar en autenticar. Cubrimos usuario y contraseña y pulsamos en “ENTRAR”.

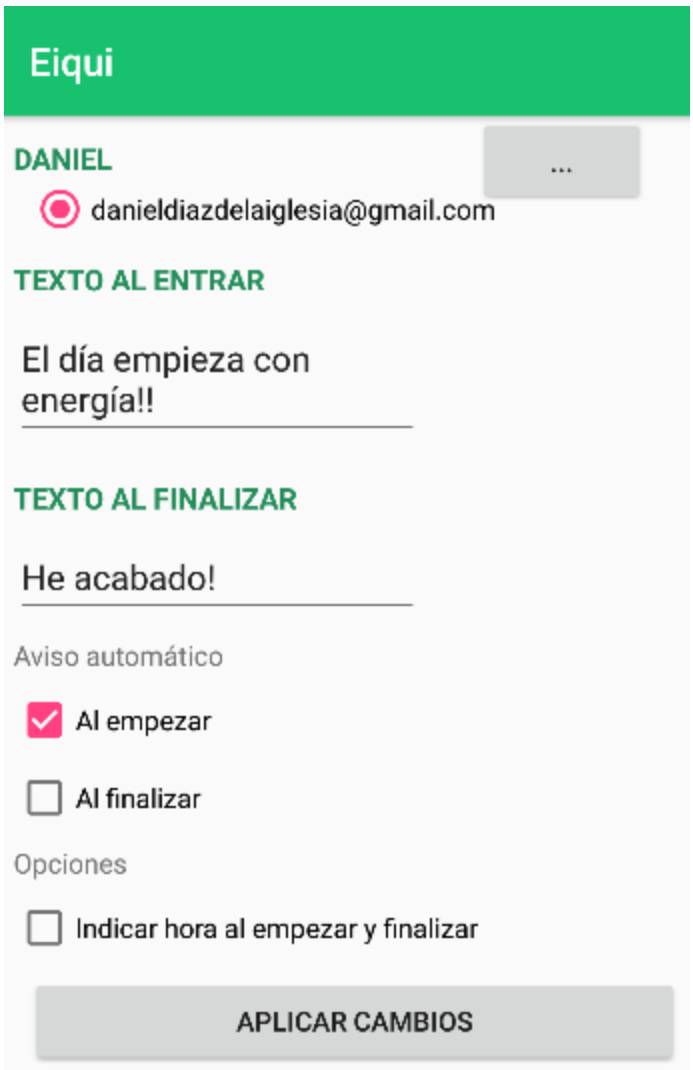


11.3.1.3 Gestión contactos

La aplicación permite añadir contactos a los que se le avisará cuando marquemos entrada o salida. Para cada usuario permite:

- Elegir si queremos avisarlo al entrar
- Elegir si queremos avisarlo al salir.
- Indicar además la hora de salida.
- Escoger un texto concreto para el mensaje de entrada o salida.

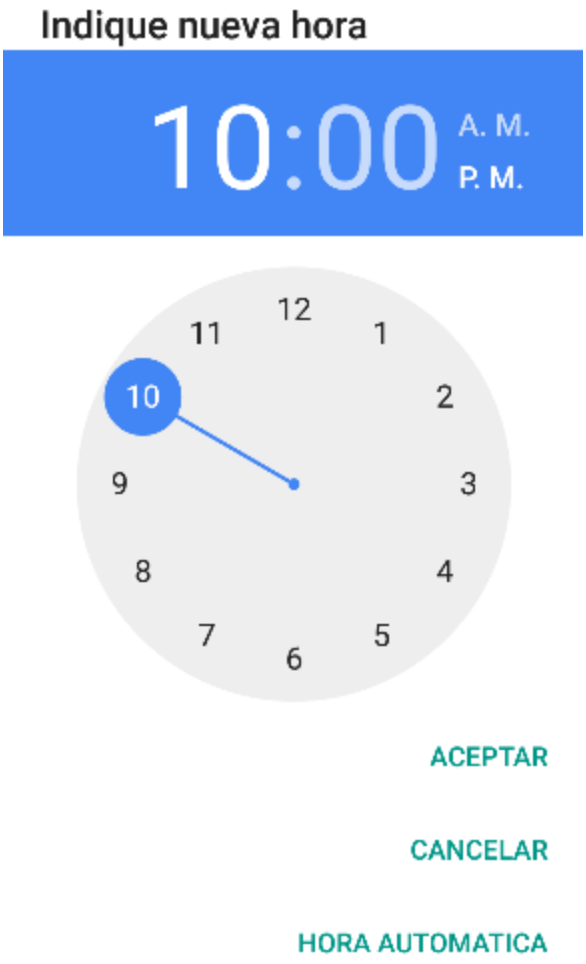
En el menú, “notificaciones” nos permite ir a la gestión de usuarios.


Eiqui
DANIEL
...
danieldiazdelaiglesia@gmail.com
TEXTO AL ENTRAR
El día empieza con energía!!
TEXTO AL FINALIZAR
He acabado!
Aviso automático
Al empezar
Al finalizar
Opciones
Indicar hora al empezar y finalizar
APLICAR CAMBIOS

11.3.1.4 Marcar Entrada

Cada vez que entremos a trabajar, si pulsamos el botón “Entrar” nos registrará la hora de entrada y nos dirá la hora de salida.

Si no hemos marcado justo al entrar, podemos indicar manualmente la hora de entrada.



Si hemos puesto hora manual y queremos volver a la hora automática, no tenemos más que pulsar en “hora automática”.

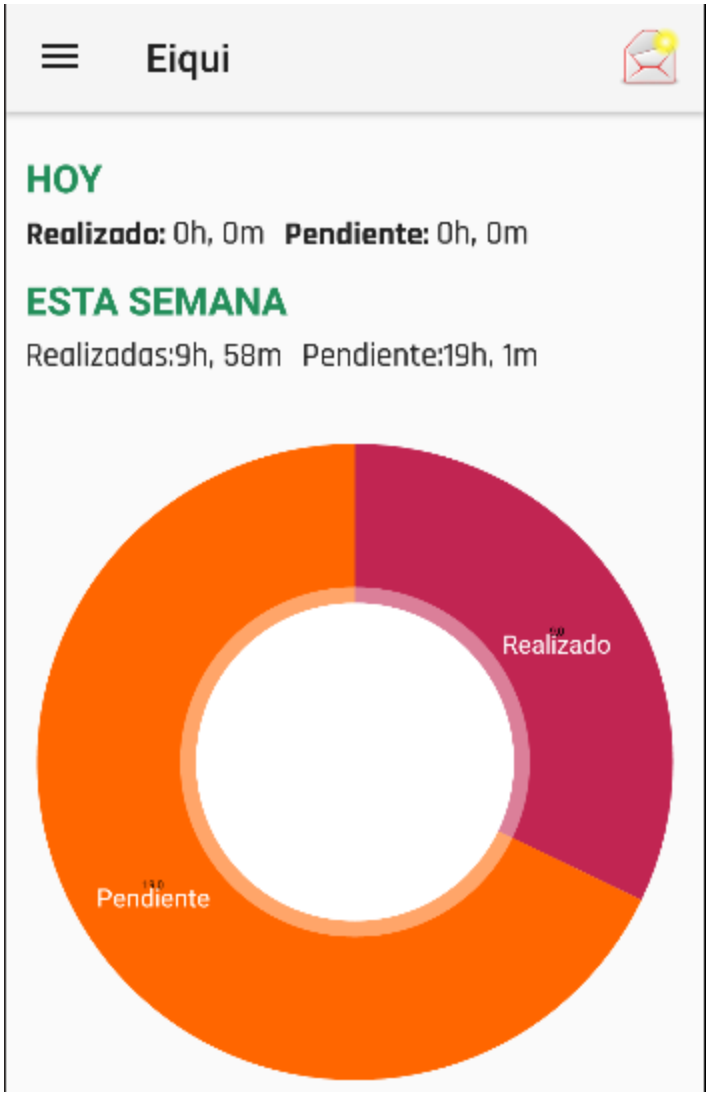
Si no queremos que avise a los usuarios de forma automática al pulsar en entrar, tendremos que desmarcar “aviso activado”. Desde esa pantalla podemos añadir un nuevo contacto de forma rápida sin tener que ir al menú notificaciones.

11.3.1.5 Marcar Salida

El caso de marcar salida es análogo al de marcar entrada, salvo que en este caso se indica la salida del trabajo.

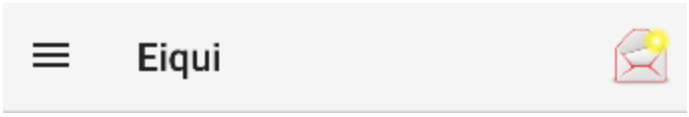
11.3.1.6 Visualizar información semanal.

La pantalla principal nos muestra las horas pendientes y realizadas en el día actual, así como la información semanal, cuantas horas hemos realizado y cuántas nos queda por finalizar.

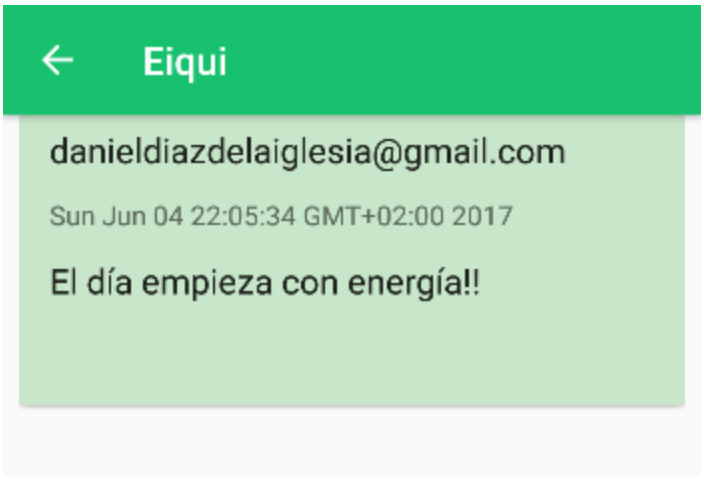


11.3.1.7 Ver avisos

Cuando un usuario nos manda un aviso de que ha entrado o salido. En la pantalla principal aparece un sobre en la parte superior derecha.



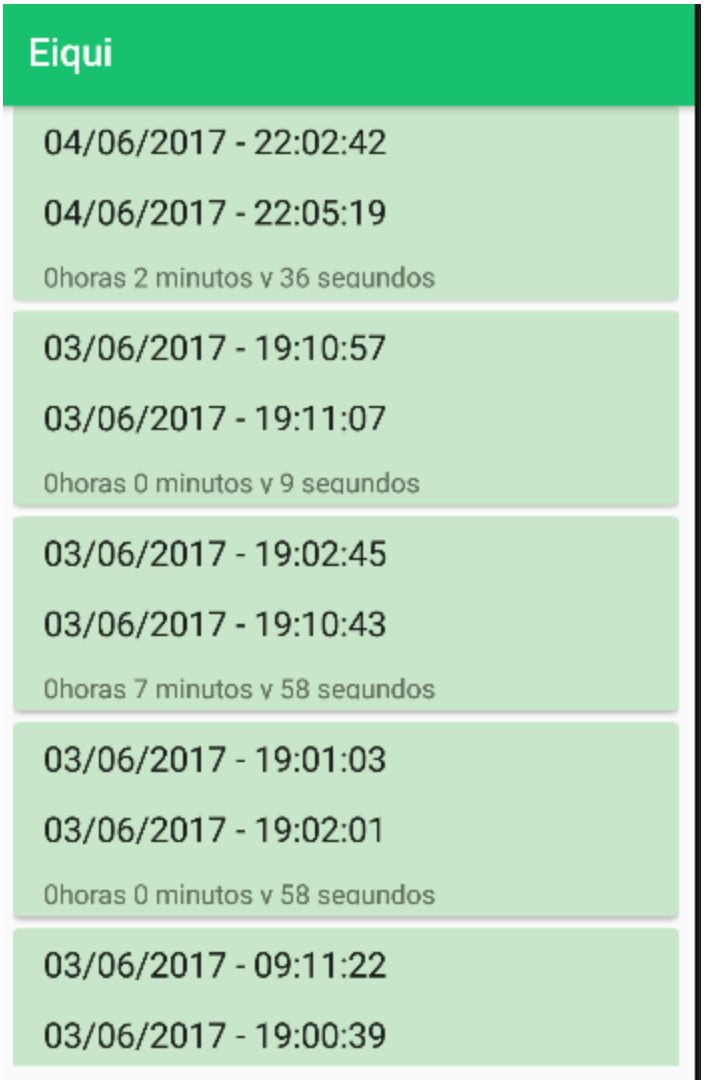
Si pulsamos en el botón, nos mostrará los mensajes recibidos.



Si no tenemos la aplicación abierta, en el área de notificaciones de Android nos llegará un aviso, si pulsamos nos arrancará la aplicación para poder ver los mensajes.

11.3.1.8 Ver registro de entradas y salidas.

En el menú, podemos ver todas las entradas y salidas que hemos marcado.



11.3.1.9 Activar NFC

Sin necesidad de pulsar en el botón o salir, con NFC podemos indicar los cambios de estado. Pasando un teléfono que tenga soporte de NFC por encima de una tarjeta NFC acvitada, realizará la acción automáticamente. Para saber si ha hecho la opción, notaremos una vibración y como se realiza el cambio de estado.

La etiqueta NFC tiene que estar activada, para ello en el menú, en NFC podemos realizarlo:

1. Colocamos el teléfono encima de la tarjeta NFC
2. Pulsamos en activar.

11.3.1.10 Configuración horaria

En el menu, en “horario” podemos realizar las siguientes acciones:

1. Marcar los días que son hábiles para trabajar.
2. Por cada día indicar las horas que deben realizarse.

11.3.2 Aplicación para reloj

La aplicación para reloj permite realizar un cambio de estado. Cada vez que pulsemos el botón, se conectará con el móvil para lanzar un cambio de estado, entrada o salida.