**Turnosync: Aplicación android para la gestión de turnos laborales**

Trabajo de Fin de Grado

GRADO EN INGENIERÍA INFORMÁTICA



Septiembre de 2018

Autor

Julio García Valdunciel

Tutor

Iván Álvarez Navia

D. Iván Álvarez Navia profesor del Departamento de Informática y Automática de la Universidad de Salamanca.

CERTIFICAN:

Que el trabajo titulado “Turnosync: Aplicación android para la gestión de turnos laborales” ha sido realizado por D. Julio García Valdunciel, con DNI 70892288-D y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado de la Titulación Grado en Ingeniería Informática de esta Universidad.

Y para que así conste a todos los efectos oportunos.

En Salamanca, a \_\_\_\_\_\_\_de Septiembre de 2018

D. Iván Álvarez Navia

Dpto. Informática y Automática

Universidad de Salamanca

# Resumen y palabras clave

El proyecto realizado como Trabajo de Fin de Grado consiste en un sistema cliente-servidor donde el rol de cliente lo desempeña la aplicación móvil en el sistema Android. El rol de servidor que incluye la base de datos sobre la que se basa la aplicación lo representa la plataforma Firebase, propiedad de Google.

El objetivo principal de esta aplicación está orientado a ayudar en la gestión y organización dentro de un entorno de trabajo con un modelo por turnos, normalmente siendo una cobertura de 24 horas. Aunque la aplicación no está restringida a ningún entorno ni profesión, la principal fuente de inspiración para este proyecto ha sido el entorno laboral dentro de un hospital, concretamente la profesión de enfermería.

La aplicación permitirá agilizar la comunicación a la hora de establecer y modificar el calendario de turnos dentro de un grupo de trabajo que actualmente suelen estar en formato físico en papel o en una plantilla del tipo Excel.

La aplicación propone sustituir o complementar estos medios llevando este calendario a un formato digital apoyado sobre una base de datos en la nube, de forma que cualquier modificación es accesible en tiempo real por los trabajadores que actúan como usuarios. Se dispone de un calendario personal del usuario y un calendario grupal donde es posible acceder al horario del resto de usuarios del grupo.

Además de la sincronización del calendario la aplicación dispone de un sistema de solicitud de cambios de turnos entre usuarios de forma que para que se haga efectivo deben estar de acuerdo las dos partes y ser aprobado en última instancia por el coordinador.

Palabras clave:

Aplicación móvil, Android, turnos laborales, Firebase, sincronización.

# Abstract and keywords

The proyect developed as Final Degree Project is a client-server system where the client role is played by the Android mobile app. The server role that includes the database upon the app is supported on is represented by the Firebase platform, owned by Google.

The main objective of this app is oriented towards helping at the management and organization within a work environment based on a work shift model, usually it being a 24 hours coverage. Even though the app is not restricted to any environment nor profession, the main source of inspiration for this project has been the work environment within a hospital, being more specific to nursing.

This app will allow to speed up the communication when setting up and modifying the shift calendar within a work group which currently they are usually in physical format on paper or an Excel template.

This app tries to replace or complement this mediums taking this calendar to a digital format supported by a cloud database, so that any modification is accessible in real time by the workers who act as users. It provides a personal calendar as well as a group calendar where each member has access to each other’s shifts.

In addition to calendar synchronization the app also provides a shift change request system between the users so that for it to be effective it’s necessary that both parts agree and in the end it has to be approved by the manager.

Keywords:

Mobile app, Android, work shifts, Firebase, synchronization.

Tabla de contenido

[Resumen y palabras clave - 5 -](#_Toc522120977)

[Abstract and keywords - 6 -](#_Toc522120978)

[1. Introducción - 11 -](#_Toc522120979)

[1.1 Estructura de la memoria - 13 -](#_Toc522120980)

[1.2 Estructura de los anexos - 13 -](#_Toc522120981)

[2. Objetivos del proyecto - 15 -](#_Toc522120982)

[2.1 Objetivos funcionales - 15 -](#_Toc522120983)

[2.2 Objetivos personales - 17 -](#_Toc522120984)

[3. Conceptos teóricos - 19 -](#_Toc522120985)

[3.1 Arquitectura Cliente – Servidor - 19 -](#_Toc522120986)

[3.2 Android y Android SDK - 19 -](#_Toc522120987)

[3.3 Base de datos no relacional - 21 -](#_Toc522120988)

[3.4 Material Design - 21 -](#_Toc522120989)

[4. Técnicas y herramientas - 23 -](#_Toc522120990)

[4.1 Programación orientada a objetos - 23 -](#_Toc522120991)

[4.2 Programas y aplicaciones - 25 -](#_Toc522120992)

[4.2.1 Android studio - 25 -](#_Toc522120993)

[4.2.2 Sublime Text 3 - 26 -](#_Toc522120994)

[4.2.3 Doxygen - 26 -](#_Toc522120995)

[4.2.4 Microsoft Word - 26 -](#_Toc522120996)

[4.2.5 Microsoft Project - 26 -](#_Toc522120997)

[4.2.6 Visual Paradigm - 27 -](#_Toc522120998)

[4.3 Lenguajes de programación - 28 -](#_Toc522120999)

[4.3.1 Java - 28 -](#_Toc522121000)

[4.3.2 XML - 28 -](#_Toc522121001)

[4.3.3 JavaScript - 29 -](#_Toc522121002)

[4.4 Plataformas externas - 30 -](#_Toc522121003)

[4.4.1 Firebase - 30 -](#_Toc522121004)

[4.4.2 GitHub - 33 -](#_Toc522121005)

[4.5 Bibliotecas - 33 -](#_Toc522121006)

[4.5.1 Joda Time - 33 -](#_Toc522121007)

[4.6 Metodología SCRUM - 34 -](#_Toc522121008)

[5. Aspectos relevantes del desarrollo. - 36 -](#_Toc522121009)

[5.1 Resumen general - 36 -](#_Toc522121010)

[5.2 Ingeniería del software - 37 -](#_Toc522121011)

[5.2.1 Marco de trabajo - 37 -](#_Toc522121012)

[5.2.2 Planificación temporal - 38 -](#_Toc522121013)

[5.2.3 Especificación de requisitos - 40 -](#_Toc522121014)

[5.2.4 Análisis del sistema - 47 -](#_Toc522121015)

[5.2.5 Diseño del sistema - 47 -](#_Toc522121016)

[5.2.6 Principales patrones usados - 47 -](#_Toc522121017)

[5.3 Elementos destacables - 47 -](#_Toc522121018)

[5.3.3 Gestión de usuarios - 47 -](#_Toc522121019)

[5.3.4 Sincronización - 47 -](#_Toc522121020)

[5.3.5 Cambios de turno - 47 -](#_Toc522121021)

[5.4 Sprints - 47 -](#_Toc522121022)

[6 Descripción funcional - 47 -](#_Toc522121023)

[7 Conclusiones y líneas de trabajo futuro - 47 -](#_Toc522121024)

[8 Bibliografía - 47 -](#_Toc522121025)

Tabla de ilustraciones

[Ilustración 1 - Arquitectura de capas del sistema Android - 20 -](#_Toc522121026)

[Ilustración 2 - Estilo Material Design - 22 -](#_Toc522121027)

[Ilustración 3 - Vista fragment\_mycalendar.xml - 29 -](#_Toc522121028)

[Ilustración 4 - Cloud Firestore - 32 -](#_Toc522121029)

[Ilustración 5 - Planificación temporal - 40 -](#_Toc522121030)

[Ilustración 6 - Ejemplo de tabla de actores - 41 -](#_Toc522121031)

[Ilustración 7 - Ejemplo de tabla de objetivos - 42 -](#_Toc522121032)

[Ilustración 8 - Ejemplo de tabla de requisitos de información - 43 -](#_Toc522121033)

[Ilustración 9 - Ejemplo de tabla de requisitos de restricción de información - 44 -](#_Toc522121034)

[Ilustración 10 - Ejemplo de tabla de requisitos no funcionales - 45 -](#_Toc522121035)

[Ilustración 11 - Ejemplo de tabla de requisitos funcionales - 46 -](#_Toc522121036)

# Introducción

Este documento recoge la memoria del Trabajo de Fin de Grado de Julio García Valdunciel, estudiantes del Grado en Ingeniería Informática en la Universidad de Salamanca.

El modelo de trabajo por turnos se basa en la necesidad de dar un servicio de normalmente 24h al día, por lo que es necesario que los trabajadores se turnen para cumplir este horario. A continuación se explicara con más detalle este modelo.

Se organizan de las siguientes formas:

* Sistema discontinuo: Es aquel que se interrumpe por la noche y en los fines de semana. Dispone de dos turnos, el de mañana y el de tarde.
* Sistema semi-continuo: Es aquel que solo se interrumpe los fines de semana. Dispone de tres turnos, siendo estos de mañana, tarde y noche dando un servicio de 24h.
* Sistema continuo: Es un servicio 24h que no se interrumpe en fin de semana. Dando un servicio completo todo el año. Estos a su vez pueden ser fijos o rotativos.
  + Fijos: Es el mismo turno a lo largo de la semana. Permite una fácil adaptación pero puede ser un inconveniente a la hora de compaginar la vida familiar si se tienen turnos de tarde.
  + Rotativos: Los turnos varían a lo largo de la semana. Esto supone una perturbación en la vida social de la persona por tener horarios irregulares.

El sistema que esta aplicación tiene más en cuenta es el sistema continuo rotativo. Supone el sistema más irregular y donde se pueden producir más cambios.

La referencia usada para desarrollar diseñar este proyecto ha sido principalmente el entorno laboral de la sección de enfermería dentro de un hospital. Por lo general para organizar los turnos la persona responsable del grupo de enfermería de una planta pone a disposición de los trabajadores una hoja de papel donde se indican los horarios de cada individuo. Si se realizan cambios muchas veces se hacen sobre la hoja ya existente y esto las hace más difíciles de leer.

Además solo se podría acceder a ella en persona o haciendo una foto y distribuirla, el mayor inconveniente vuelve a ser cuando se realizan cambios, para estar consciente de ellos es necesario tomar una nueva foto y volverla a distribuir o estar en persona.

Esta aplicación pretende agilizar el proceso de notificación de los cambios de turnos y ayudar a tener una única versión del calendario. A través de una base de datos en tiempo real los usuarios podrán estar actualizados a la última versión de forma instantánea para que no haya confusiones por no haber sido avisados.

Otro proceso que podría ser agilizado sería el de cambiar turnos entre los usuarios, al usar la aplicación como medio se podrían solicitar cambios de turno, aceptarlos y ser aprobados por el coordinador sin necesidad de contactarse directamente.

## Estructura de la memoria

El contenido y la estructura de este documento son de la siguiente forma:

1. **Introducción**: Descripción del origen del proyecto y del problema que pretende solucionar,
2. **Objetivos**: Explicación de los principales objetivos que se tratan de conseguir y mediante que funcionalidades se intentan resolver.
3. **Conceptos teóricos**: Se describen los conceptos necesarios para la correcta compresión de este proyecto.
4. **Técnicas y herramientas**: Se detallan y exponen técnicas y herramientas usadas para el desarrollo del proyecto así como bibliotecas y la justificación de su uso.
5. **Aspectos relevantes del desarrollo**: En este apartado se detallaran aquellos aspectos destacables en cada fase del desarrollo del proyecto.
6. **Descripción funcional**: Se presentan todas las funcionales de la aplicación en detalle
7. **Conclusiones y líneas de trabajo futuro**: Se comentan las conclusiones obtenidas a partir del desarrollo del proyecto además de las mejoras que podrían llevarse a cabo a continuación.

## Estructura de los anexos

Este proyecto también dispone de cinco anexos que se detallan a continuación:

1. **Anexo I**: Planificación temporal
2. **Anexo II**: Especificación de Requisitos del Software
3. **Anexo III**: Especificación de Diseño
4. **Anexo IV**: Documentación Técnica de Programación
5. **Anexo V**: Manuales de usuario

# Objetivos del proyecto

Este apartado recoge de forma detallada los objetivos planteados que la aplicación debe cumplir y pretende solucionar. Se distinguen los objetivos estrictamente funcionales dados por los requisitos del software y aquellos de carácter técnicos planteados por el alumno.

## Objetivos funcionales

Los objetivos funcionales indican las funcionalidades que debe tener la aplicación para que pueda dar solución a los problemas planteados. Estos objetivos se explican con más detalles en el Anexo II – Especificación de Requisitos del software.

Puesto que se trata de una aplicación donde los usuarios interactúan con el sistema a distintos niveles uno de los objetivos es la gestión de usuarios. Estos pueden registrarse en el sistema desde la aplicación usando un email y contraseña o una cuenta de Google. Posteriormente serán capaces de iniciar sesión para acceder al resto de funcionalidades. Si el usuario ha olvidado la contraseña dispondrá de un sistema de cambio de contraseña donde proveerá su dirección de email y se le enviará un email que le permitirá introducir una nueva contraseña. Para la gestión de usuarios se usa la plataforma Firebase, concretamente el módulo de gestión de usuarios Firebase Auth.

Una vez el usuario ha iniciado sesión correctamente es capaz de crear grupos de trabajo sobre los que se centraran la mayor parte del resto de funcionalidades. Al ser el creador del grupo actuará con el rol de coordinador del grupo. El coordinaron es la figura responsable que maneja la gestión de grupos de trabajo.

Una vez creado un grupo será posible acceder al calendario. Cada usuario dispondrá de un calendario personal dentro del grupo donde únicamente se muestran los turnos asignados a este usuario organizado por meses. Además dispondrá de un calendario grupal donde es posible acceder al horario del resto de usuarios y que se organiza por semanas. Esto le permite al usuario acceder a una visualización del horario laboral propio y de sus compañeros.

El rol de coordinador permite invitar a otros usuarios introduciendo su dirección de email. Si el usuario invitado aún no ha creado una cuenta con esa dirección, se le incluye en el grupo automáticamente en el momento en que la crea. Estos tendrán el rol de usuarios asociado dentro del grupo. Será posible modificar el nombre con el que aparecen estos usuarios además de eliminarlos. De esta manera el coordinador es el responsable de la gestión de los trabajadores.

Para poder asignar turnos a los usuarios, al coordinador se le permitirá la gestión de tipos de turnos, de forma que primero deberá definir los tipos de turno desde una ventana de configuración. Se le pedirá el nombre del turno, una etiqueta identificadora junto con un color para mostrarlo en el calendario además de la hora de inicio y la hora de fin del turno. Estos tipos de turnos serán editables después de su creación así como eliminarlos.

Una vez se han incluido los trabajadores en el grupo y se han definido los tipos de turno el coordinador será capaz de asignar turnos a los usuarios directamente desde el calendario grupal siendo posible asignar el mismo turno a la vez a lo largo de la semana a un usuario. Estos turnos serán posteriormente modificables siendo capaz de cambiar el tipo de turno o transferirlo a otro usuario además de eliminarlo.

Para respetar los derechos de los trabajadores y ayudar al coordinador a organizar los turnos, es posible definir un límite de horas semanales usado para avisar al coordinador cuando las sobrepasa a la hora de asignar turnos. Además los usuarios son capaces de acceder al recuento de horas asignado en cada mes.

El principal atractivo de la aplicación es que una vez este proceso se efectúa, los cambios en el horario se reciben en el cliente del usuario en tiempo real. Esto es posible a través de la plataforma de Firebase de base de datos en tiempo real Cloud Firestore, utilizada como base de datos de la aplicación. De esta forma los usuarios sincronizan sus dispositivos y obtienen su horario instantáneamente junto con una notificación push que les avisa de estos cambios en el calendario.

Con los turnos asignados los usuarios dispondrán de un sistema de solicitud de cambios de turnos donde un usuario solicita el cambio de un turno suyo por un turno de un compañero. Este proceso se hará en tres partes siendo estas la solicitud, la aceptación del otro usuario y la aprobación del coordinador que podrá rechazar el cambio aunque los dos usuarios estén de acuerdo. En cada fase se utilizaran notificaciones para informar a los usuarios del estado de la solicitud y para requerir su intervención para resolverla.

Como se ha comentado la aplicación dispondrá de notificaciones push donde se le indicará al usuario cuando han modificado su horario y durante el proceso de cambio de turno.

## Objetivos personales

En un proyecto del tamaño del Trabajo de Fin de Grado surgen una serie de objetivos personales, especialmente si no se ha realizado algo similar en el pasado.

* Aplicar los conocimientos adquiridos durante el grado, así como la mentalidad metódica propia de una ingeniería.
* Llevar a cabo un proyecto completo de principio a fin mediante una planificación temporal a largo plazo para nuestra experiencia y dando como resultado a un producto funcional y usable.
* Aprender a desarrollar para dispositivos móviles, concretamente para el sistema operativo Android. Así como aprender un nuevo framework.
* Poner en práctica y aprender conocimientos más avanzados en el lenguaje de programación Java tratado en algunas asignaturas durante el grado. Además de usar otros lenguajes como XML(eXtensible Markup Language o Lenguaje de Marcado Extensible) y Javascript.

# Conceptos teóricos

## Arquitectura Cliente – Servidor

La aplicación se basa en una arquitectura cliente servidor donde el cliente es la propia aplicación, sin embargo para su correcto funcionamiento necesita la base de datos y el servicio de autenticación que se alojan en la plataforma Firebase, un servicio de computación en la nube propiedad de Google que es gratuito en una pequeña escala.

La arquitectura cliente – servidor consiste en una comunicación con dos roles diferenciados que son como su nombre indica son el cliente y el servidor. Aunque esta comunicación puede darse en la misma máquina cuando mejor se aprovecha es al aplicarse a una red de máquinas a través de una red local o de internet.

La parte del cliente realiza peticiones al servidor de forma que este entienda y sepa procesarlas para que después este devuelva una respuesta acorde a la solicitud.

## Android y Android SDK

Android es un sistema operativo orientado a dispositivos móviles. Está presente en multitud de dispositivos de distintas marcas y en España tuvo una cuota de mercado del 86% en el primer trimestre de 2018.

Android está basado en el kernel de Linux, lo que lo hace de código abierto y gratuito. El lenguaje de programación para desarrollar aplicaciones en Android es Java junto con XML para definir las vistas. También es posible usar Kotlin, lenguaje creado por JetBrains al igual que IntelliJ IDEA el entorno de desarrollo sobre el que se basa Android Studio que es el entorno oficial para desarrollar aplicaciones en Android.

En la Ilustración 1 - Arquitectura de capas del sistema Android es posible observar la arquitectura en capas del sistema Android.

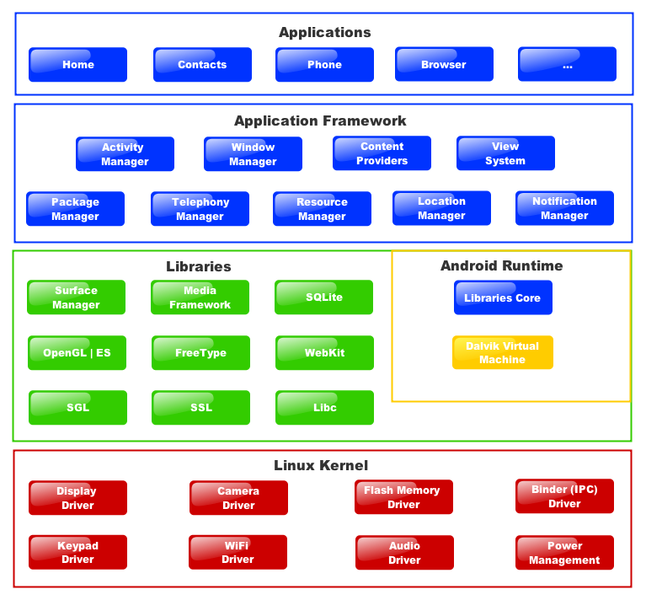


Ilustración 1 - Arquitectura de capas del sistema Android

Para poder desarrollar aplicaciones en Android es necesario obtener el Android SDK (Software Development Kit) que consiste en un conjunto de herramientas para el desarrollo de aplicaciones incluyendo un depurador, bibliotecas, un emulador para simular distintos dispositivos, documentación, ejemplos de código y tutoriales.

El SDK soporta versiones antiguas de Android para alcanzar a un mayor número de consumidores mediante la biblioteca llamada Android Support.

## Base de datos no relacional

Las bases de datos no relacionales también llamadas NoSQL se caracterizan por no tener un esquema rigido en los registros de los datos, de forma que cada registro puede tener campos diferentes a los demás. Esto significa que si un campo no es necesario en un determinado registro, este no aparece aportando claridad al no tener campos vacíos.

Escalan de forma horizontal aumentando el número de nodos (servidores) para incrementar el rendimiento del sistema. Muchos sistemas NoSQL permiten usar consultas tipo Map-Reduce, las cuales pueden ejecutarse en todos los nodos a la vez paralelizando la consulta.

Son sistemas que realizan operaciones directamente en memoria, volando los datos a disco cada cierto tiempo. De esta forma las operaciones son rápidas aunque tiene el inconveniente de que si el servidor falla y se cae se pierdan información y consistencia. Para solventar esto normalmente se requiere que otro servidor haya realizado la operación antes de validarse.

Este tipo de base de datos se expresa en JSON (JavaScript Object Notation) o puede transformarse fácilmente a objetos JSON.

En general son bases de datos con buen rendimiento y flexibilidad, ideal para usarse en tiempo real. Cloud Firebase, la base de datos usada en este proyecto es de este tipo.

## Material Design

Material Design es el estilo de diseño oficial de Android actualmente. Esto fue anunciado durante la conferencia de Google I/O de 2014 e integrado desde Android 5.0 Lollipop ofreciendo una serie de herramientas para poder implementarlo.

Este diseño se basa en objetos materiales que son piezas colocadas en un espacio y con un movimiento determinado. Las principales características distintivas son el uso de la profundidad, superficies, bordes, sombras y colores.

Las animaciones también juegan un papel principal dando una sensación de fluidez y llamando la atención hacia la información más importante.

Este proyecto ha intentado utilizar este estilo en su diseño de la interfaz de la aplicación utilizando las bibliotecas oficiales ofrecidas. En la Ilustración 2 - Estilo Material Design se puede observar un ejemplo.

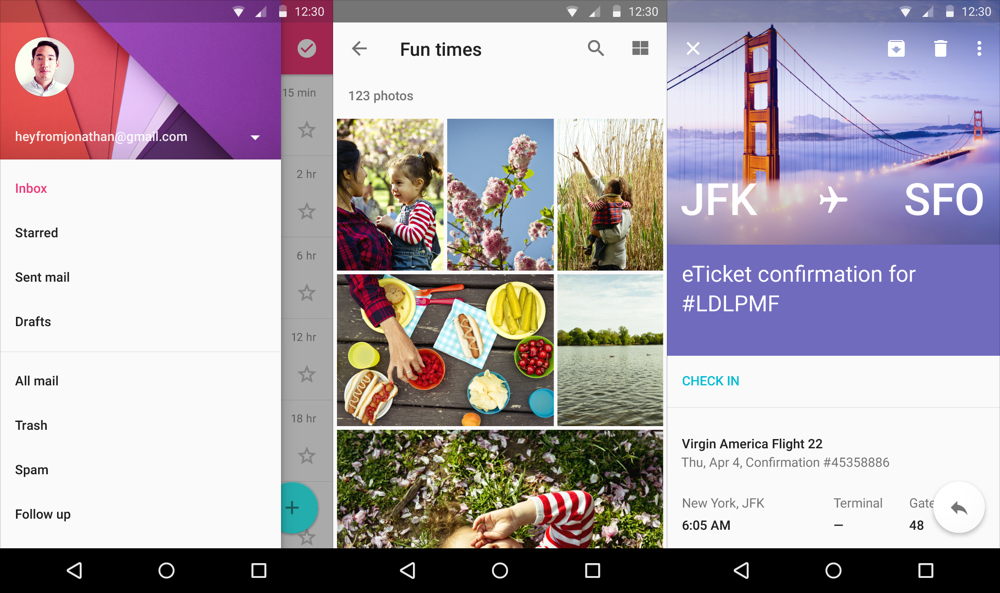


Ilustración 2 - Estilo Material Design

# Técnicas y herramientas

En este apartado se exponen todas las herramientas que han participado en el proyecto, así como lenguajes de programación usados, plataformas además de las técnicas y metodologías usadas en el desarrollo.

## Programación orientada a objetos

La programación orientada a objetos es el paradigma de programación más popular actualmente. Permite construir un sistema expresándolo de una forma cercana a la realidad a través del concepto de objetos. El lenguaje Java utilizado para desarrollar aplicaciones usando el Android SDK es un lenguaje orientado a objetos.

Un objeto es una entidad que contiene en un mismo componente datos y procedimientos de una forma lógica y cercana al mundo real. Los objetos con la misma estructura de datos y comportamiento se agrupan en una clase y se forman a partir de de esta. Se dice que un objeto es la instancia de una clase.

Algunos de los conceptos más importantes son:

* Objeto: entidad que posee un conjunto de propiedades o atributos (datos) y de funcionalidades o procedimientos (métodos). Se corresponden con objetos reales o con conceptos internos.
* Clase: Define las propiedades y comportamiento de un tipo de objeto. Se dice que los objetos son instancias de una clase y se crean a partir de estas.
* Atributos: Datos asociados a un objeto o una clase que indica las características individuales de un objeto.
* Métodos: Algoritmos asociado a un objeto o una clase que se ejecuta tras recibir un mensaje (ser llamado). Indica que es lo que puede hacer un objeto, puede modificar los atributos del objeto o llamar a otros métodos.
* Herencia: Se da entre clases y consiste en definir una clase a partir de otra, de forma que la nueva clase a definir incluye los atributos y métodos de la clase heredada.

Las características más importantes de la programación orientada a objetos son:

* Abstracción: Consiste en centrarse en las características esenciales de un objeto y ocultar los elementos que no lo son. Esto es para poder simplificar los objetos y tener un mejor control sobre la complejidad
* Encapsulación: Consiste en reunir los elementos que pertenecen a una misma identidad para formar un componente unido de forma lógica y aumentar la cohesión en el diseño.
* Principio de ocultación: Cada objeto está aislado del resto de forma que sus características internas solo pueden ser modificadas desde el mismo objeto o a través de una interfaz como interactuar con otros objetos. Esto protege las propiedades del objeto de modificaciones indeseadas por otros objetos y así evitar interacciones inesperadas.

## Programas y aplicaciones

### Android studio

Es el entorno de desarrollo utilizado para este proyecto. Es el entorno oficial para el desarrollo de aplicaciones para Android y está basado en el entorno de desarrollo IntelliJ IDEA de JetBrains. La primera versión fue publicada en diciembre de 2014 con una licencia Apache 2.0.

Este entorno tiene una serie de funcionalidades destacables:

* Soporte de la construcción basado el Gradle, un sistema de automatización para la construcción de proyectos.
* Un emulador para poder lanzar la aplicación en distintos dispositivos.
* Un editor de vistas gráfico junto al editor de textos para XML.
* Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versiones y otros problemas.
* Plantillas para crear diseños comunes de Android y otros componentes.
* Instant Run que permite aplicar cambios sin reconstruir un nuevo APK (Android Application Package)
* Integración con el sistema de control de versiones Git

Un proyecto en android studio se separa principalmente en las carpetas:

* **Manifests**: Contiene el fichero AndroidManifest.xml que proporciona información vital sobre la aplicación como el nombre, la clase principal Application, el icono a mostrar o el estilo visual general. También es donde se registran los permisos que necesita la aplicación.
* **Java**: Carpeta que contiene todo el árbol de paquetes que contienen el código fuente.
* **Res**: Contiene los recursos usados en la aplicación, es donde se definen las vistas usando lenguaje XML además de iconos, cadenas de texto o colores. Permite usar un recurso distinto dependiendo de diferentes factores como el lenguaje, la orientación del dispositivo o la versión de Android entre otros.

### Sublime Text 3

Se ha utilizado Sublime Text para programar en Javascript una serie de funciones ejecutadas en la nube de Firebase. Es un editor de texto que soporta multiples lenguajes de programación. Originalmente era una extensión del editor Vim y se dispone de una versión de evaluación gratuita sin fecha de caducidad que es la que se ha usado en este caso. Es desarrollado por Jon Skinner. Esta licenciado bajo copyright por Sublime HQ Pty Ltd.

### Doxygen

Es una herramienta utilizada para generar la documentación de un proyecto a partir del código fuente en diferentes lenguajes entre los que se encuentran C, C++, Objetive-C, C#, PHP, Java, Python, IDL, Fortran, VHDL y algunos otros. Fue lanzada en octubre de 1997 y desarrollada por Dimitri van Heesch. La documentación es generada en html permitiendo una navegación mediante referencias. Tiene una licencia GNU GPLv2.

### Microsoft Word

Es un programa de procesamiento de textos propiedad de Microsoft que está incluido en el paquete de oficamática Microsoft office. Fue originalmente desarrollado por Richard Brodie para el computador de IBM en el sistema operativo DOS en 1983. Tiene una licencia de propiedad privada. Se ha utilizado para la redacción de esta memoria.

### Microsoft Project

Es un programa para la administración de proyecto de software propiedad de Microsoft. Es utilizado para la gestión de proyectos de software asistiendo a los administradores de proyecto con distintas funcionalidades como la planificación temporal, el calculo de costes y realizar un seguimiento del desarrollo y la carga de trabajo. Tiene una licencia de propiedad privada. Se ha utilizado para realizar la planificación temporal del proyecto.

### Visual Paradigm

Es un programa de herramientas CASE para realizar tareas de ingeniería del software que soporta UML 2, SysML y BPMN. Permite realizar multiples tipos de diagramas donde los mas comunes pueden ser diagramas de clases, de casos de uso, de arquitectura o de paquetes además de varios tipos de diagramas de interacción. Se ha utilizado esta herramienta para realizar toda la parte de ingeniería del software del proyecto.

## Lenguajes de programación

Para este proyecto se han usado distintos lenguajes de programación para las distintas áreas del proyecto.

### Java

El lenguaje de programación predilecto para desarrollar aplicaciones para Android usando el Android SDK es Java aunque es posible usar Kotlin. La aplicación soporta a partir de de la versión 5.0 Lollipop y por tanto la versión de Java utilizada es la 7.

Es un lenguaje de programación orientado a objetos multiplataforma altamente extendido. Se caracteriza por poder ser compilado una vez y que sea posible ejecutarse en distintas maquinas. Para esto se vale del JVM (Java Virtual Machine) que toma el bytecode generado por el compilador de Java y recompilándolo en el código nativo de la plataforma. Fue desarrollado por Sun Microsystems en 1982 y posteriormente adquirida en 2010 por Oracle.

### XML

En este proyecto se usara XML para definir todos los recursos usados por la aplicación, siendo estos las vistas, referencias a cadenas de textos, colores, iconos y otros elementos.

XML(eXtensible Markup Language) es un meta-lenguaje de marcado especificado por W3C (World Wide Web Consortium) utilizado para transportar datos y almacenar datos de forma legible. Proviene del lenguaje SGML y permite definir la gramática de lenguajes específicos. Es un estándar utilizado para intercambiar datos entre diferentes plataformas que da soporte a bases de datos dando facilidades a la comunicación entre distintas plataformas.

Un ejemplo del uso de XML en la aplicación sería una de las vistas de la aplicación apreciable en Ilustración 3 - Vista fragment\_mycalendar.xml.



Ilustración 3 - Vista fragment\_mycalendar.xml

### JavaScript

En este proyecto se usa para programar una serie de funciones que se ejecutan en la nube de Firebase a partir de eventos en la base de datos.

Javascript es un lenguaje de programación orientado a la web que se puede aplicar a un documento HTML y proporcionar funcionalidad y contenidos dinámicos. Fue creado por Brendan Eich, co-fundador del proyecto Mozilla en 1995. Es un lenguaje muy flexible que permite realizar multitud de actividades desde actualizar una vista de forma dinámica a crear juegos y aplicaciones integradas basadas en base de datos.

Existen una gran cantidad de APIs (Application Programming Interface, Interfaz de programación de aplicación) para ampliar aún más la funcionalidad y permitir los desarrolladores integrarse con otros proveedores de contenido como Twitter o Facebook.

## Plataformas externas

El proyecto se ha apoyado en gran medida en la plataforma Firebase usando algunos de sus servicios que se comentarán a continuación además de usar Github como plataforma para alojar el proyecto y utilizar el control de versiones de Git.

### Firebase

Es una plataforma para el desarrollo de aplicaciones web y móviles. Soporta Android, iOS y aplicaciones web. Tiene su origen en la startup de James Tamplin y Andrew Lee den 2011 llamada Evolve. Proporcionaba una API para la integración de sistemas de chat en páginas web. Posteriormente incluyeron un sistema de arquitectura en tiempo real y fue cuando se fundó Firebase en Abril de 2012.

En ocubre de 2014 fue comprado por Google y ha integrado otros equipos como Divshot en 2015 y Tejido y Crashlytics en 2017.

Cuenta con una aplica documentación de sus servicios donde se documentan como incluir cada servicio en el proyecto y la información necesaria sobre cada una de las funciones de su API disponibles al desarrollador. También disponen de proyectos de ejemplo donde se muestran de una forma más práctica las posibilidades que ofrece. Poseen un canal de Youtube donde resuelven dudas planteadas y comentan como utilizar sus servicios además de que algunos de los propios desarrolladores se ocupan personalmente de resolver preguntas en la plataforma StackOverflow.com.

Se ha utilizado esta plataforma principalmente por ofrecer un servicio gratuito por debajo de un volumen de interacción determinado, por su servicio de base de datos en tiempo real y las facilidades que se dan al desarrollador para utilizarlas.

Además dispone de consola de control donde manejar todos los servicios usados por la aplicación como añadir y eliminar usuarios manualmente al igual que editar la base de datos, ver los registros de las funciones en la nube, enviar notificaciones manualmente y consultar los errores que experimentaron los usuarios.

Los servicios que ofrece que se han usado para desarrollar el proyecto son:

#### Firebase Auth

Se trata de un servicio de autenticación que proporciona almacenar datos esenciales de usuarios de forma segura y que admite distintas formas de autenticación como email y contraseña, teléfono, integración con Google Facebook y Twitter entre otros.

Dispone también de un sistema de verificación de direcciones de email utilizado en esta aplicación para que sea necesario confirmar que la dirección de email es válida antes de utilizar la aplicación.

Este servicio asigna un identificador único a cada usuario que será utilizado a través de toda la aplicación para distinguir y tratar con los usuarios.

#### Cloud Firestore

Se trata de una base de datos en tiempo real NoSQL flexible y escalable en la nube. Mantiene los datos sincronizados con los clientes a través de agentes de escucha en tiempo real y ofrece asistencia sin conexión a través de cachés. Permite una integración con otros servicios de Firebase como Cloud Functions.

Actualmente se encuentra en una fase beta y disponen de otro servicio de base de datos NoSQL en tiempo real llamado Realtime Database. La principal diferencia es la estructura de almacenamiento de los datos, en Realtime Database se utiliza una estructura JSON donde se debe evitar la anidación de datos, en Cloud Firestore se organizan en colecciones y documentos, una estructura similar a JSON pero que permite la anidación utilizando subcolecciones y simplificando la organización de los datos.

Dispone de restricciones de acceso de lectura y escritura usando un sistema de reglas, se ha limitado la base de datos para que los usuarios necesiten estar registrados.

En Ilustración 4 - Cloud Firestore se puede observar la estructura de la base de datos desde la consola de control.

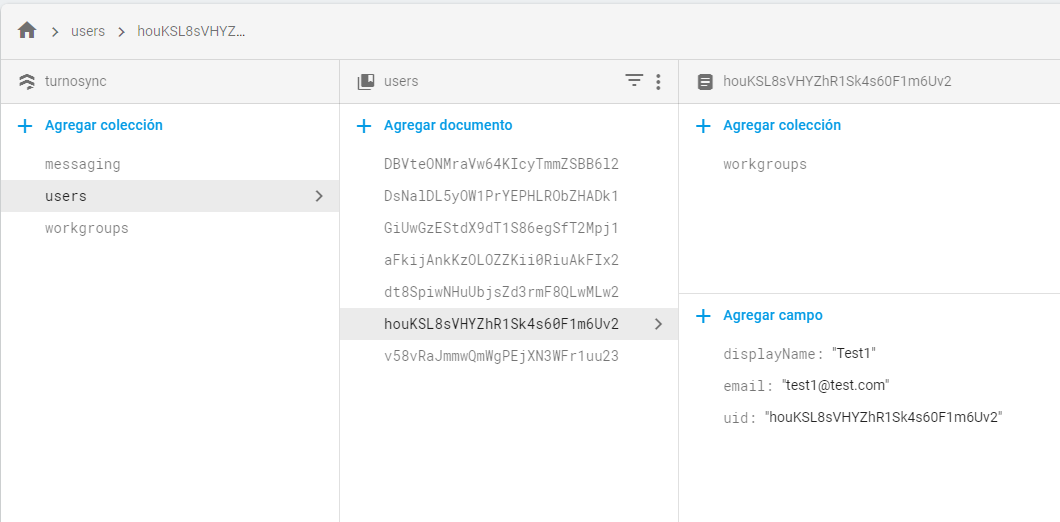


Ilustración 4 - Cloud Firestore

#### Cloud Functions

Permite ejecutar código en respuesta a eventos de otros servicios de Firebase o mediante solicitudes HTTPS. El código se almacena en la nube y se ejecuta en ella. Para desarrollar funciones es necesario inicializar el proyecto usando las herramientas de firebase del SDK de Cloud Functions. Se escribene en local y posteriormente En este proyecto se han utilizado activadores de Firebase Auth al crearse usuarios y con Cloud Firestore escribiendo en base de datos en la creación de usuarios y en la creación, borrado y actualización de datos en otras partes de la aplicación. También se ha utilizado Firebase Cloud Messaging para enviar notificaciones como respuesta a una modificación de la base de datos.

#### Firebase Cloud Messaging

Es la nueva versión de la plataforma llamada Google Cloud Messaging. Es una solución de mensajería multiplataforma que permite notificar una aplicación cliente. Se ha utilizado un sistema integrado con Cloud Firestore para enviar notificaciones cada vez que se le cambia el calendario a un usuario además de informar a los usuarios durante el proceso de cambio de turno.

### GitHub

Es una plataforma de alojamiento de proyectos que usa un sistema de versiones Git. Fue lanzado en 2008 y en junio de 2018 comprado por Microsoft. Los proyectos se almacenan de forma pública aunque con una cuenta de pago se pueden tener repositorios privados.

Github impulsa el código libre al alojar los proyectos de forma pública y facilita la colaboración entre desarrolladores. Dispone de sección de problemas que se usa normalmente para crear tickets por parte de usuarios donde se informa de los errores que puede tener un programa o aplicación y así ayudar a arreglarlos. Cuenta también con una sección de wiki donde el autor puede documentar el proyecto y otras facilidades como facilitar la elección de la licencia usada. También dispone de una sección con una tabla kanban para gestionar un equipo.

## Bibliotecas

Las bibliotecas usadas para desarrollar este proyecto han sido principalmente las que proporcionan distintos componentes de Android y las necesarias para usar los servicios de Firebase. Sin embargo para manejar fechas y horas se ha utilizado la biblioteca Joda Time que actúa como una alternativa al manejo de fechas y horas de Java 7.

### Joda Time

Es una biblioteca muy extendida de Java que pretende sustituir a las clases date y time de Java anteriores a la versión 8. En java 8 las clases referentes al tratamiento de fechas y horas se basaron en esta biblioteca.

Se ha usado esta biblioteca por la facilidad y la simplificación de su uso a la hora de tratar con fechas en la aplicación.

## Metodología SCRUM

La metodología elegida para este proyecto ha sido la metodología ágil SCRUM. Define una serie de prácticas y roles. Las principales características son:

* Es una metodología iterativa e incremental que en lugar de planificar y desarrollar el proyecto de forma secuencial hasta finalizarlo trata de conseguir un producto funcional en cada incremento y se itera por las distintas fases refinando puliendo las tareas ya realizadas.
* Suele emplearse para equipos no demasiado grandes consiguiendo una buena cooperación y comunicación entre ellos.
* Se realizan reuniones regulares con el cliente para comprobar que se cumplen con las expectativas y reuniones diarias con el equipo para coordinarse y comprobar el avance del proyecto.

Se distinguen varios roles donde los principales son:

* **Product owner**: Es la persona que representa al cliente y especifica los requisitos del software ayudando al usuario a escribir las historias de usuarios.
* **Scrum Master**: Es el líder del equipo que realiza la gestión del proyecto y trata de facilitar su desarrollo intentando eliminar los inconvenientes que surjan en el equipo y que se asegura que el proceso se cumple.
* **Equipo de trabajo**: Es el grupo de personas responsable de llevar a cabo el proyecto que poseen las habilidades necesarias para su completo desarrollo y finalización.

En el flujo de trabajo se distinguen varias partes:

* **Sprint**: El sprint es el periodo donde se lleva a cabo el desarrollo del proyecto. Tiene una duración estable de entre 2 y 4 semanas que tiene como objetivo crear un incremento funcional del producto que pueda ser entregado al cliente.
* **Product Backlog**: Es el documento que recoge todos los requisitos del proyecto. Es elaborado por el *product owner* junto al cliente.
* **Sprint backlog**: Es el subconjunto de requisitos que serán desarrollados durante un sprint. Estos requisitos se toman a partir del *product backlog*.
* **Reunión diaria**: Se realiza cada día para comprobar el estado del proyecto de forma breve.
* **Revisión de sprint**: Al final del sprint se realiza una revisión para presentar los trabajos realizados.

# Aspectos relevantes del desarrollo.

En este apartado se presenta de manera detallada el trabajo realizado a lo largo de proyecto comentando primeramente de forma general todo el recorrido y posteriormente entrando en detalle con la parte de ingeniería del software, las distintas iteraciones TODO\*

## Resumen general

El proyecto comenzó a partir de un familiar que trabaja como enfermera que se sentía incómoda con el sistema actual para la gestión de turnos utilizado en su trabajo. El principal problema que tenía era el hecho de que el calendario de turnos se e encontraba en una hoja en papel que únicamente se encontraba en el lugar de trabajo y que para poder distribuirlo se realizaban fotografías y se enviaban entre los trabajadores. Al ser una profesión donde los turnos pueden ser muy cambiantes esta hoja se actualizaba múltiples veces sobre el mismo papel lo que lo hacía más difícil de leer, además de que era necesario realizar una nueva fotografía lo que resultaba en tener varias versiones del mismo.

El desarrollo del proyecto comienza capturando los requisitos necesarios para el desarrollo de una aplicación que pueda ayudar a solucionar algunos de los problemas planteados que también ocurren en otras profesiones de manera que no sea específica de una única profesión. El resultado es en un sistema donde interactúan dos actores principales, el trabajador y la persona encargada que coordina los turnos.

Entonces se plantea un sistema donde se deba sincronizar automáticamente los turnos que establece el coordinador para que el usuario tenga acceso inmediato. Para poder soportar esto se utiliza una base de datos en tiempo real en la nube que es proporcionada por la plataforma Firebase, se utiliza el servicio de base de datos Cloud Firestore.

Junto a este servicio también se usa el servicio de autenticación Firebase Authentication, el servicio de funciones ejecutadas en el servidor Cloud Functions además del servicio que recoge los errores ocurridos en la aplicación en cualquier dispositivo, usado para poder identificar errores fuera del entorno de desarrollo llamado Crashlythics.

Antes de empezar a desarrollar código de este proyecto al no tener experiencia previa en el desarrollo de aplicaciones Android se realizó dos cursos, uno sobre Android y otro sobre Firebase, son online gratuitos y están proporcionados por Udacity, una plataforma que ofrece múltiples cursos online de carácter tecnológico. Los cursos de Android estaban recomendados en la página oficial de Android y están impartidos por profesionales empleados de Google.

El desarrollo se ha realizado siguiendo los objetivos propuestos en cada iteración e intentando dejar el software en un estado con el menor número de errores posibles, de manera que la aplicación era funcional y cumplía con los requisitos propuestos correctamente.

## Ingeniería del software

### Marco de trabajo

Para la realización del proyecto se ha seguido la metodología SCRUM anteriormente confirmada. En este apartado se explicará cómo se ha adaptado al proyecto:

En cuanto a los roles:

* **Product owner**: Este rol corresponde al alumno Julio García puesto que es quien ha definido las historias de usuario y ha tenido contacto con quien se considera el cliente, un familiar con experiencia en un entorno de trabajo regido por turnos que ha planteado los problemas y las posibles soluciones que este proyecto pretende resolver.
* **Scrum master**: Corresponde al alumno que ha sido quien se ha ocupado de intentar cumplir con la metodología y resolver los problemas encontrados a lo largo del desarrollo.
* **Equipo de trabajo**: Corresponde al alumno que ha sido la única persona que se ha ocupado del desarrollo del proyecto a todos los niveles.
* **Stakeholders**: Aunque es un Trabajo de Fin de Grado propuesto por el alumno se puede considerar como cliente al familiar que dio pie al proyecto.

En cuanto al proceso:

* **Sprint**: Se han ido realizando las historias de usuario correspondientes a cada *sprint* al tener sentido lógico entre ellas y ser necesario cumplir unas antes de realizar las siguientes.

Se han intentado cumplir con los plazos establecidos para cada *sprint* aunque a veces no ha sido posible cumplirlos al compaginar el desarrollo del proyecto como la entrega de trabajos de otras asignaturas o la realización de prácticas externas de empresa en el segundo cuatrimestre.

Al terminar cada *sprint* se ha realizado una revisión comprobando lo realizado y lo que se dará prioridad en el siguiente sprint.

* **Reuniones diarias**: Al ser únicamente una persona no se han realizado reuniones como tal pero si se ha revisado lo último implementado y programado el trabajo del día.

### Planificación temporal

La planificación temporal del proyecto se realizó utilizando la herramienta Microsoft Project para establecer el calendario y expresar la planificación como un diagrama de Gant. Esta sección se ha tratado en mas detalle en documento *Anexo I – Plan del Proyecto Software.*

En el calendario se han tenido en cuenta los fines de semana como días hábiles para el proyecto y se han considerado unos periodos reservados para la preparación de exámenes. Sin embargo se ha subestimado el tiempo dedicado a las entregas de trabajos de asignaturas impactando en el desarrollo del proyecto. Además no se tuvo en cuenta al realizar la planificación el periodo de prácticas de empresa que también impactó en el proyecto retrasando la planificación.

Por los motivos anteriormente mencionados el proyecto planificado para ser entregado inicialmente en la convocatoria de julio de 2018 fue retrasado a la convocatoria de septiembre del mismo año.

La planificación recoge las tareas que se desarrollan a lo largo del proyecto y se divide en los distintos *sprints* que a su vez pasan por varias fases del desarrollo. Estas fases son:

* **Aprendizaje previo**: Se adquieren conocimientos necesarios para el desarrollo del sprint.
* **Modelado de negocio**: Principalmente se caracteriza por la reunión con el cliente para la captura de requisitos, su refinamiento y para consultar la opinión del trabajo realizado.
* **Análisis y diseño**: Se realizan las tareas de ingeniería del software donde se formalizan los requisitos y se estructuran los componentes del software.
* **Implementación**: Concentran las tareas de desarrollo del código que otorgan funcionalidad al software.
* **Pruebas**: Se realizan pruebas para comprobar la correcta funcionalidad del código implementado.

Cada tarea de la planificación tiene asignada una duración aproximada en días además de relaciones de precedencia con tareas anteriores que especifican el orden natural del desarrollo. Se indica también a quien se le ha asignado la tarea, que al ser solo un miembro en el equipo, toda la carga ha recaído en el alumno.

Al final de cada iteración cuando se han completado todas las tareas se produce un hito donde la aplicación contiene nuevas funcionalidades completadas y se encuentra en un estado posiblemente entregable.

Las primeras iteraciones del desarrollo están enfocadas al aprendizaje, la captura de requisitos y su análisis y diseño estableciendo las historias de usuario, las iteraciones centrales se enfocan a la construcción y es donde se crea la mayor parte de la aplicación y las últimas es donde se completa la funcionalidad y se busca arreglar los errores y pulir la interacción.

En la siguiente ilustración se pueden observar las primeras 3 iteraciones o *sprints*:



Ilustración 5 - Planificación temporal

### Especificación de requisitos

Para la captura de requisitos se ha tenido contacto con la persona que hace de cliente, un familiar que trabaja como enfermera y ha sido quien ha expresado sus necesidades, con su ayuda se han definido una serie de historias de usuario.

Posteriormente se han establecido de una manera más formal los objetivos así como definido los requisitos siguiendo el método de Duran y Bernárdez y siendo diferenciados en requisitos funcionales, requisitos no funcionales, requisitos de información y requisitos de restricción de información. Junto a los requisitos también se han definido los actores que interactúan con el sistema.

Se han definido:

* 3 Actores
* 6 Objetivos
* 5 Requisitos de información
* 2 Requisitos de restricción de información
* 4 Requisitos no funcionales
* 41 Casos de uso

Este trabajo de definición de requisitos se puede ver en detalle en el *Anexo II - Especificación de Requisitos del Software*.

A continuación se pondrá un ejemplo de cada tipo de tabla empleada.

#### Actores

Los actores son las identidades que interactúan con el sistema. Son los siguientes:

* Usuario
* Coordinador
* Sistema

|  |  |
| --- | --- |
| **ACT-0001** | **Usuario** |
| **Versión** | **1.0 ( 10/11/2017 )** |
| **Autor** | **Julio García Valdunciel** |
| **Fuentes** | **Propuesta Trabajo de Fin de Grado** |
| **Dependencias** | **Ninguno** |
| **Descripción** | **Este actor representa a un usuario que utiliza el sistema y puede realizar diferentes funciones dentro del mismo.** |
| **Comentarios** | **Ninguno** |

Ilustración 6 - Ejemplo de tabla de actores

#### Objetivos

Los objetivos son la funcionalidad a más alto nivel que la aplicación debe cumplir. Estos objetivos se han comentado en la sección de objetivos y son los siguientes:

* Gestión de usuarios
* Calendario de turnos
* **Entorno para la creación de turnos**
* **Sincronización con la base de datos en la nube**
* **Sistema de cambios**
* **Notificaciones**

|  |  |
| --- | --- |
| **OBJ-0001** | **Gestión de usuarios** |
| **Versión** | **1.0 ( 10/11/2017 )** |
| **Autor** | **Julio García Valdunciel** |
| **Fuentes** | **Propuesta Trabajo de Fin de Grado** |
| **Descripción** | **El sistema deberá realizar una gestión de usuarios permitiendo crear cuentas e iniciar sesión.** |
| **Subobjetivos** | **El sistema deberá permitir cambiar la contraseña utilizada en caso de pérdida o por voluntad propia. Iniciar sesión con cuenta de Google.** |
| **Importancia** | **Vital** |
| **Urgencia** | **Vital** |
| **Estabilidad** | **Alta** |
| **Comentarios** | **Ninguno** |

Ilustración 7 - Ejemplo de tabla de objetivos

#### Requisitos de información

Son los requisitos que indican que información se debe almacenar, a partir de estos requisitos se obtiene el modelo. Son los siguientes:

* **Información sobre los usuarios**
* **Información sobre los grupos de trabajo**
* **Información de los turnos de trabajo**
* **Información de los tipos de turnos**
* **Información de las solicitudes de cambio de turno**

|  |  |
| --- | --- |
| **IRQ - 0001** | **Información sobre los usuarios** |
| **Versión** | **1.0 ( 10/11/2017 )** |
| **Autor** | **Julio García Valdunciel** |
| **Fuentes** | **Propuesta Trabajo de Fin de Grado** |
| **Objetivos asociados** | **OBJ-0001** |
| **Descripción** | **El sistema deberá almacenar la información correspondiente a los usuarios.** |
| **Datos específicos** | * **Identificador** * **Nombre** * **Email** * **Contraseña** * **Relación con los grupos de trabajo** |
| **Importancia** | **Alta** |
| **Urgencia** | **Alta** |
| **Estabilidad** | **Alta** |
| **Comentarios** | **Ninguno** |

Ilustración 8 - Ejemplo de tabla de requisitos de información

#### Requisitos de restricción de información

Son los requisitos que indican que información debe limitarse y como se hace efectiva. Son los siguientes

* **Acceso a base de datos**
* **Permisos según rol**

|  |  |
| --- | --- |
| **CRQ - 0001** | **Acceso a base de datos** |
| **Versión** | **1.0 ( 10/11/2017 )** |
| **Autor** | **Julio García Valdunciel** |
| **Fuentes** | **Propuesta Trabajo de Fin de Grado** |
| **Objetivos asociados** | **OBJ-0001, OBJ-0004** |
| **Descripción** | **La información almacenada por el sistema deberá satisfacer la siguiente restricción: solo los usuarios registrados podrán acceder a la base de datos.** |
| **Importancia** | **Alta** |
| **Urgencia** | **Alta** |
| **Estabilidad** | **Alta** |
| **Comentarios** | **Ninguno** |

Ilustración 9 - Ejemplo de tabla de requisitos de restricción de información

#### Requisitos no funcionales

Son los requisitos que no afectan directamente a la funcionalidad pero influyen en el sistema especificando ciertos elementos necesarios. Son los siguientes:

* **Soporte de versiones de android**
* **Base de datos en tiempo real**
* **Material Design**
* **Usabilidad**

|  |  |
| --- | --- |
| **NFR-0001** | **Soporte de versiones de android** |
| **Versión** | **1.0 ( 10/11/2017 )** |
| **Autores** | **Julio García Valdunciel** |
| **Fuentes** | **Propuesta Trabajo de Fin de Grado** |
| **Dependencias** | **Ninguno** |
| **Descripción** | **El sistema deberá soportar versiones de android desde la versión 5.0 Lollipop, donde se implanta Material Design.** |
| **Importancia** | **Alta** |
| **Urgencia** | **Media** |
| **Estabilidad** | **Alta** |
| **Comentarios** | **Ninguno** |

Ilustración 10 - Ejemplo de tabla de requisitos no funcionales

#### Requisitos funcionales

Son los requisitos que definen cada una de las funcionalidades del sistema y cómo interactúan los actores para llegar a usarla. Se especifican como casos de uso.

|  |  |  |  |
| --- | --- | --- | --- |
| **UC-0001** | **Registro** | | |
| **Versión** | **1.0 ( 12/11/2017 )** | | |
| **Autores** | **Julio García Valdunciel** | | |
| **Fuentes** | **Propuesta Trabajo de Fin de Grado** | | |
| **Dependencias** | **OBJ-0001** | | |
| **Descripción** | **El sistema debe comportarse de forma que: El Usuario pueda registrarse en la aplicación mediante email y contraseña.** | | |
| **Precondición** | **Estar en la ventana de inicio de sesión.** | | |
| **Secuencia normal** | **Paso** | | **Secuencia normal** |
| **1** | | **El Usuario selecciona en el menú registrarse.** |
| **2** | | **El Sistema muestra la pantalla de registro** |
| **3** | | **El Usuario introduce su nombre, email y contraseña dos veces.** |
| **4** | | **El Sistema comprueba que los datos introducidos son válidos.** |
| **5** | | **El Sistema envía los datos al servidor donde se registra al usuario.** |
| **6** | | **El Sistema redirige al usuario al caso de uso Iniciar Sesión** |
| **Postcondición** | **El Usuario es registrado en el sistema.** | | |
| **Excepciones** | **Paso** | **Excepciones** | |
| **4** | **Si no son válidos se cancela el registro y se indica el campo no válido. Vuelve al paso 3.** | |
| **Frecuencia esperada** | **Baja** | | |
| **Importancia** | **Alta** | | |
| **Urgencia** | **Alta** | | |
| **Estabilidad** | **Alta** | | |
| **Comentarios** | **Ninguno** | | |

Ilustración 11 - Ejemplo de tabla de requisitos funcionales

### Análisis del sistema

TODO\* Casos de uso?

### Diseño del sistema

Los patrones de diseño arquitectónicos usados ha sido principalmente MVC y MVVM para.

El patrón MVC (Model, View, Controller o Modelo, Vista, Controlador) es un patrón que se puede aplicar en multitud de programas y aplicaciones que se pueda organizar en una vista donde interactúa el usuario, un controlador que realiza la lógica de negocio y operaciones

TODO\* base de datos

### Patrones Android usados

Para realizar el desarrollo del proyecto se han usado varios patrones que son comunes en aplicaciones Android.

#### MVVM

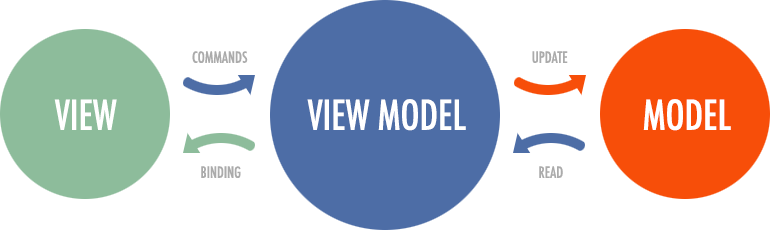


Ilustración 12 - Patron MVVM

El patrón MVVM (Model, View, ViewModel o Modelo, Vista, VistaModelo) es un patrón arquitectónico consiste en el modelo de datos, la vista y el componente vistamodelo cuya tarea es detectar los cambios del modelo y actualizar la vista automáticamente. Es un patrón impulsado desde Android, se proporcionan las clases necesarias mediante la biblioteca de Android *android.arch.lifecycle.*

Este patrón se ha utilizado de forma experimental dentro del paquete MyCalendar principalmente para desacoplar los datos del ciclo de vida de las vistas, de manera que sea más simple poder recrearlas al cambiar la orientación del dispositivo.

#### Databinding

Es una técnica que permite establecer una relación entre la vista y los datos en forma de una clase generada automáticamente cuyos miembros son referencias a elementos de la vista. Esta técnica sustituye a la anterior forma de acceder a la vista que normalmente era utilizando el método *findViewById.* Esta nueva técnica permite actualizar la vista automáticamente al modificar el modelo. Normalmente forma parte del patrón MVVM aunque es posible usarlo por separado.

#### Navigation Drawer

La navegación principal se ha realizado mediante un panel lateral denominado *navigation drawer*. Este patrón consiste en una vista que contiene el panel lateral y un contenedor donde se van sustituyendo las diferentes vistas que corresponden a las opciones de navegación del panel.

En este proyecto el panel lateral está controlado por la clase DrawerActivity. Desde esta clase es posible realizar varias acciones, es posible navegar entre la pantalla principal controlada por HomeFragment, donde se muestran los grupos de trabajo de un usuario y también cambiar a la pantalla de calendarios de trabajo controlada por la clase MyCalendarFragment. Además también es posible acceder a una nueva pantalla con la configuración de la aplicación o desconectarse.

La siguiente ilustración corresponde con el código que controla las opciones del panel lateral.

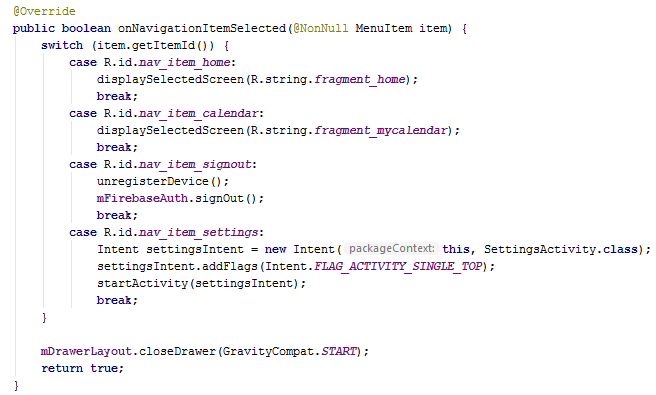


Ilustración 13 – Código de las opciones panel lateral

#### Comunicación con fragmentos

Los fragmentos son vistas que se incluyen dentro de otras, pueden estar dentro de una actividad o de otro fragmento. Un ejemplo es el caso del panel lateral usado comentado en la sección anterior pero se usan a lo largo de toda la aplicación.

En la vista de calendario se utiliza el componente *ViewPager* que permite navegar entre distintas vistas deslizándolas horizontalmente. Cada una de estas vistas es un fragmento.

Para comunicar el fragmento con la vista que lo contiene se ha utilizado un patrón con una interfaz de *callbacks* dentro del fragmento que la actividad o fragmento contenedor deben implementar. Dentro del fragmento se crea una referencia a la clase que implemente la interfaz y se utiliza para enviar datos.

Este patrón está indicado como la forma correcta de comunicación entre las dos clases en la documentación de Android.

#### Adapter

Es un patrón que se encarga de adaptar el contenido a una vista, especificando como se muestran los datos. Es utilizado proveer a vistas de tipo lista los datos y como se muestran estos datos. En la aplicación se ha utilizado este patrón principalmente para proporcional a los *ViewPager* los fragmentos que contienen los calendarios, los propios calendarios están formados por un componente GridView que organiza una lista de vistas en una cuadricula pudiendo crear la cuadricula del calendario además de haberse usado en cada vista con un listado de elementos. El método principal usado es el método *getView* que es donde se carga la vista del elemento y se rellenan sus datos, este método se ejecuta tantas veces como elementos tiene la vista.

Existen distintas implementaciones de *Adapter* siendo las más comunes *ArrayAdapter* donde es necesario proporcionar un objeto *List* o *RecyclerView.Adapter* una implementación especifica del componente *RecyclerView*, un tipo de vista de listado donde solo se carga en memoria los elementos que están activamente mostrados. Además es posible extender la clase *BaseAdapter* que sirve como plantilla para implementar un *Adapter* personalizado.

#### Dialogs

Los diálogos son vistas que se superponen encima de la vista actual sin tapar por completo la vista anterior. En Android se implementan usando un tipo de fragmento denominado *DialogFragment*. En estos diálogos se piden o se muestran datos, como son fragmentos estos datos se envían a la clase contenedora usando las interfaces de comunicación comentadas anteriormente.

Se han utilizado multiples diálogos diferentes, pidiendo datos para crear grupos de trabajo, turnos, invitar personas, diálogos de confirmación al eliminar usuarios del grupo o diálogos donde solo se muestra información.

## Elementos destacables

### Gestión de usuarios

La gestión de usuarios se ha realizado utilizando el servicio de autenticación de Firebase denominado Firebase Authentication. Este servicio se integra dentro de Android proporcionando una API con los métodos que permiten gestionar usuarios como puede ser crear cuentas de usuario o actualizar datos del usuario.

Se han utilizado dos formas de crear una cuenta en la aplicación, la primear utilizando correo electrónico y contraseña que se proporcionan desde la pantalla de registro dentro de la aplicación y la segunda donde se provee la identidad utilizando una cuenta de google. Con la cuenta de Google no es necesario realizar un registro como tal, al intentar iniciar sesión mediante el botón de iniciar sesión con Google desde la pantalla de inicio de sesión si no existía cuenta previamente esta se crea.

También se permite recuperar la contraseña a partir de un email, esta funcionalidad se ha implementado mediante un enlace que recuerda a una página web y que en este caso abre un cuadro de dialogo donde introducir la dirección de email. En el email enviado se incluye un enlace donde introducir una nueva contraseña.

Las cuentas de usuario no están activadas por defecto. Al crear una cuenta con correo y contraseña se enviará un email donde se podrá confirmar la cuenta de correo. Después de hacer esto se le permite al usuario iniciar sesión.

A los usuarios se les permite cambiar su cuenta de correo y el nombre de usuario desde una pantalla de configuración a la que se accede a través del menú lateral.

En el menú lateral también se encuentra la opción para desconectarse de la aplicación llevando al usuario de vuelta a la página de inicio de sesión.

Cuando se crea un usuario en el servicio de autenticación es necesario replicar sus datos en la base de datos para poder usarlos a lo largo de la aplicación. Para lograr esto se han utilizado funciones ejecutadas en el servidor, esto es proporcionado por el servicio de Firebase denominado Cloud Functions. Estas funciones son lanzadas por eventos, se ha establecido una función que se ejecuta cuando se crea un usuario para que cree una entrada en la base de datos

### Invitar usuarios a un grupo

Para incluir a los usuarios en un grupo de trabajo se les invita usando la dirección de email. Después se les añade al grupo usando Cloud Functions de manera que la función se lance cuando se escriba la invitación en base de datos. En este caso sería necesario que el usuario ya tuviera una cuenta, sin embargo para tener él cuenta el caso de invitar a una persona sin cuenta también se lanza esta función en la creación de la cuenta.

### Base de datos y Sincronización

TODO\* base de datos

La aplicación se basa en la sincronización con la base de datos en tiempo real de ciertos elementos, principalmente los turnos de trabajo pero se ha usado en los lugares donde se muestre un listado de elementos que pueda ser actualizado añadiendo, eliminando o modificando elementos.

Para obtener actualizaciones en tiempo real se han utilizado unos agentes de escucha que detectan cuando se producen cambios en la base de datos y producen un evento con los datos asociados. Esto permite mantener actualizado automáticamente los listados y los calendarios de turnos. Un ejemplo de agente de escucha es la Ilustración 14 - Agente de escucha de la base de datos que escucha el listado de grupos de trabajo del usuario.



Ilustración 14 - Agente de escucha de la base de datos

Los datos se reciben como un mapa clave valor que es posible convertir a un objeto designado y añadirlo a un listado. Además del dato se recibe también el tipo de cambio en la colección, siendo este añadido, modificado o eliminado y se actualiza la lista según corresponda.

En el calendario personal la petición se turnos se hace mes a mes, en el calendario grupal se buscan los turnos de la semana correspondiente de todos los usuarios.

### Cambios de turno

El proceso de cambio de turno se ha diseñado de manera que fuera lo más sencillo y cómodo para el usuario. Para realizar una solicitud de cambio se debe seleccionar un turno del propio usuario y un turno de otro usuario manteniendo pulsado un turno desde el calendario grupal. Se ha decidido que la operación se haga manteniendo pulsado para que no se realice por accidente al navegar por las semanas. Una vez se seleccionan turnos válidos se debe confirmar en un cuadro de dialogo que muestra los datos de los dos turnos.

Será posible solicitar un cambio mientras el estado del calendario después del cambio no produzca conflictos como asignar dos turnos al mismo día.

El proceso de cambio de turno comienza con la solicitud de cambio de turno, estas se muestran en una pantalla accesible desde cualquiera de los dos calendarios. El otro usuario implicado tiene la capacidad de aceptar o rechazar el cambio. Si lo acepta entonces el coordinador es quien puede aprobar o no el cambio, teniendo la última palabra. Las solicitudes de cambio de turno poseen un estado que se va modificando a lo largo del proceso, cuando alcanza el estado de aprobado por el coordinador es cuando se hace efectivo. Durante el proceso una serie de notificaciones informan a los usuarios implicados y al coordinador del estado de la solicitud, por ejemplo, cuando el otro usuario acepta la solicitud además de la notificación a este usuario informando que existe una solicitud de cambio dirigida hacia él.

Antes de hacer efectivo el cambio se comprueba que el calendario no haya sido modificado, si no hay ninguna inconsistencia con la solicitud se realiza el cambio y además se anulan las solicitudes con las que tuvieran relación.

### Notificaciones

Las notificaciones push de la aplicación se han implementado utilizando el servicio de Firebase de mensajería denominado FCM (Firebase Cloud Messaging). Este servicio permite enviar mensajes a la aplicación para mostrarlos como notificaciones.

Para poder utilizar este servicio y conocer el destino se han implementado dos servicios que funcionan conjuntamente con la aplicación. El primero es un servicio que obtiene y actualiza el token de registro con FCM que es utilizado para registrar la aplicación en los canales de notificaciones elegidos. Se utiliza un canal por usuario que admite varios dispositivos. De esta forma las notificaciones llegan a todos los dispositivos donde esté conectado.

El segundo servicio es el que se encuentra a la escucha de los mensajes de FCM y la notificación adecuada según su contenido. Este servicio muestra notificaciones esté o no la aplicación en primer plano.

Los casos de notificaciones que se tienen en cuenta son, por una parte cuando el coordinador realiza un cambio en el calendario de un usuario se le notifica, por otra parte se mantienen informados de los cambios del estado de la solicitud de cambio de turno. Se notifica al usuario destinatario de una nueva solicitud de cambio de turno, si la acepta se notica al usuario que lo solicitó además de al coordinador para informarle que depende de este aprobarla o no. Si no se acepta la solicitud se avisa al usuario solicitante sin necesidad de molestar al coordinador, si es el coordinador el que deniega la solicitud se les avisa a los dos usuarios. En la Ilustración 15 - Notificaciones se observa un ejemplo de notificación cuando se le modifica el calendario al usuario.

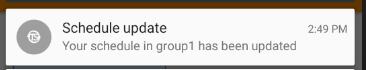


Ilustración 15 - Notificaciones

## Sprints

En este apartado se comentara brevemente todo el proceso de desarrollo comentando en más detalle algunos aspectos de la aplicación y haciendo hincapié en ciertos retos que se consiguieron superar. Al principio de cada sprint se refinaban los requisitos usando la experiencia adquirida en los *sprints* pasados y se planeaba el orden del desarrollo.

### Sprint 1

Aunque no es estrictamente propio del proyecto, previo al desarrollo se realizaron unos cursos online de aprendizaje de desarrollo en Android y sobre el uso de la plataforma Firebase. Estos cursos se realizaron a través de la página Udacity. Concretamente fueron los cursos titulados *Developing Android Apps*, *Advanced Android App Development* y el curso de Firebase titulado *Firebase in a Weekend*.

Se tuvo una reunión para la captura de requisitos, siendo el momento en el que se escribieron las historias de usuario y se definieron formalmente los objetivos del sistema, delimitando la funcionalidad que va a cubrir la aplicación.

### Sprint 2

Se inicializo el proyecto en Android Studio además de en la plataforma Firebase configurando los ficheros necesarios e incluyendo las bibliotecas de Firebase de autenticación. Entonces se hizo un pequeño prototipo inicial para comprobar la conexión con el servidor y se crearon unos usuarios de prueba de forma manual con los que se hicieron pruebas iniciando sesión con una interfaz simple controlada por la actividad de iniciar sesión.

Posteriormente se creó la pantalla de registro controlado por la actividad de registro donde se pedía introducir un nombre que identifique al usuario, un email que no haya sido registrado antes y una contraseña con la que inciar sesión, se pide escribirla para que el usuario esté seguro de su contraseña. Todos estos campos deben validarse comprobando en cada uno diferentes rasgos como que el email tenga el formato correcto y la contraseña sea segura obligando a introducir mayúsculas y números.

Se refina la pantalla de inicio de sesión incluyendo el inicio de sesión usando el inicio de sesión con Google y un enlace que lleva a una nueva actividad donde poder proporcionar el correo para recuperar la contraseña. Esta actividad se sustituye por un cuadro de dialogo más adelante.

### Sprint 3

En este sprint se construye el patrón *navigation drawer* que lo constituye el panel lateral y las pantallas que se van intercambiando. Lo primero que se hizo fue conectar la navegación desde el inicio de sesión, utilizando la escucha de cambio de estado de autenticación se puede detectar cuando se conecta y desconecta el usuario manteniendo la sesión activa. Es esta clase por la que se entra a la aplicación y que lleva al inicio de sesión cuando no hay ningún usuario conectado.

Se aprendió a usar fragmentos, vistas incrustadas sobre otras vistas pudiendo ser actividades u otros fragmentos. Usando fragmentos se le puso contenido al panel lateral, siendo capaz de cambiar entre ellas usándolo.

Primeramente se desarrolló la pantalla principal de inicio, esta pantalla permite visualizar los grupos de trabajo de los usuarios y crearlos utilizando un botón flotante propio de Material Design. El listado de los grupos se implementa utilizando la vista *Gridview* que muestra elementos en una cuadrícula y se utiliza un adaptador personalizado para suplir los elementos a partir de la petición a base de datos de los grupos del usuario conectado.

Se utilizan interfaces en el fragmento para comunicarse con la actividad principal de modo que el título en la barra de la aplicación refleje que la pantalla de inicio está activa.

Inicialmente la base de datos se diseña teniendo una colección de usuarios por un lado y la colección de los grupos por otro, para asociarlos se incluyen la referencia de los usuarios bajo los grupos y se indica el rol. Posteriormente para simplificar las llamadas s base de datos puesto que se requieren todos los grupos del usuario este diseño se invierte referenciando los grupos bajo la colección de usuarios. De este modo es posible obtener todos los grupos de un usuario con una única llamada. Además se mantiene una referencia de los usuarios para poder acceder al listado de usuarios de un grupo.

Además también se refinó el panel lateral incluyendo la opción de desconectarse de la aplicación y se desarrolla una pantalla de configuración donde es posible modificar el email asociado a la cuenta y el nombre a mostrar del usuario.

Al final del sprint se realizaron algunas tareas de corrección de errores referentes a la gestión de usuarios encontrados posteriormente.

### Sprint 4

Se construye el grueso de la interfaz que corresponde a las vistas de los calendarios. A esta vista se accede pulsando sobre uno de los grupos mostrados en la pantalla de inicio o desde el panel lateral que muestra último grupo visitado o si no hay ninguno del primero en la lista.

Primeramente se desarrolla el calendario personal, formado por un *ViewPager*, una vista que permite navegar entre páginas, usado para visualizar un mes por página. Se utiliza un *PagerAdapter*, que es el que construye los fragmentos para las páginas.

Cada página es un fragmento que contiene una etiqueta que identifica el mes y una vista *GridView* para construir el calendario del mes. Además del mes correspondiente también se muestran los días del mes anterior y posterior que completan la semana.

Para conseguir mostrar correctamente todos los días se tuvieron que hacer unos cálculos partiendo del primer día a mostrar teniendo en cuenta que se muestran un máximo de 12 meses con el mes actual en el centro. Para calcular los días del mes anterior se parte de la posición del primer día del mes en la semana, esos días del mes anterior contando por el final son con los que se rellena la semana. De igual manera se hace para los días del mes siguiente. Estos cálculos se realizan utilizando las clases Date y Calendar de Java 7. Más adelante en uno de los últimos sprints se decide migrar la parte de tratamiento de fechas a la bibliote Joda Calendar que simplifica y ayuda a manejar fechas de una forma mucho más sencilla. Se puede comparar esta sección del código con las dos alternativas mediante la Ilustración 16 - Días del mes anterior con Java 7 y la Ilustración 17 - Días del mes anterior con Joda Calendar.

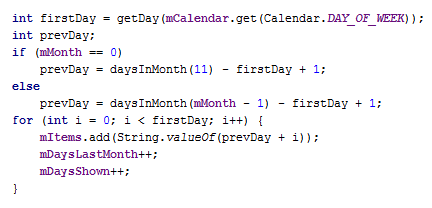


Ilustración 16 - Días del mes anterior con Java 7

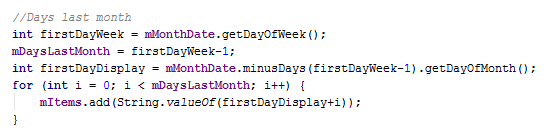


Ilustración 17 - Días del mes anterior con Joda Calendar

Posteriormente se desarrolla la vista del calendario grupal, mostrando una semana por página y mediante un *GridView* donde cada columna representa un día para los usuarios que representan las filas. Para construir este calendario se parte del primer día de la semana y del listado de usuarios del grupo.

La navegación entre los dos tipos de calendario se realiza mediante un botón en el menú que según el calendario actualmente mostrado representa el otro modo de calendario. Si se quiere ver el calendario de semanas se muestra la primera del mes de la página actual siempre que no sea el mes actual, en eso se muestra la semana que corresponde a la fecha real. En el caso contrario se muestra el mes correspondiente de la semana visualizándose.

Los turnos se expresan usando una abreviación de hasta 3 letras junto a un color representativo, esto se muestra en la casilla correspondiente del mes o de la semana según el calendario. Para obtener estos turnos se hace una petición a base de datos filtrando por el periodo a mostrar utilizando el campo fecha de los turnos.

### Sprint 5

Una vez se tiene la construcción de los calendarios primeramente es necesario poder invitar personas al grupo de trabajo. Para ello se desarrolla una pantalla de configuración de grupo, donde se muestra una lista de los usuarios del grupo y una opción del menú que permite invitar usuarios usando su dirección de email.

Esta pantalla es una actividad a la que se navega utilizando un botón del menú de la pantalla del calendario. Se utiliza la actividad como base para mostrar dos fragmentos, primeramente la pantalla con la lista de usuarios y la opción de invitar. En este fragmento además de invitar personas también permite cambiar el nombre a mostrar que consiste en una abreviación de máximo 4 letras y eliminarlas del grupo pero sin borrar las entradas de base de datos, solo se marca como inactivo, para que pueda seguir siendo visible el historial de turnos. La gestión de usuarios está limitada únicamente al coordinador.

Para invitar a un usuario al grupo se escribe la invitación en base de datos incluyendo el identificador de usuario y el grupo al que se le invita. Esta escritura lanza una Cloud Function que resuelve la invitación incluyéndolo en el grupo y al mismo tiempo generando una abreviación del usuario utilizando el nombre del usuario para mostrarla en el calendario grupal. Si el usuario ya tenía una cuenta se le incluye directamente, sin embargo si no dispone aún de cuenta la invitación se mantiene y se le incluye en el momento de la creación de la misma.

### Sprint 6

Una vez la gestión de usuarios del grupo está completada para poder asignar turnos a los usuarios se utilizan como plantilla los denominados tipos de turno, que definen las características de un turno sin asignarse a un usuario. Estas características son el nombre del turno dado, una abreviación y un color para mostrar en el calendario y la hora de inicio y de fin del turno. Para las horas de inicio y de fin se usa un tipo de vista de entrada de datos con un reloj radial de 24 donde se pueden establecer la hora y el minuto exacto.

Para definir estos turnos se utiliza un cuadro de dialogo (Ilustración 18 - Cuadro de dialogo para la creación de tipos de turno) que se abre al usar el botón de menú en la pantalla de tipos de turno. A esta pantalla se accede desde la configuración del grupo con la lista de usuarios y se utiliza para gestionar los tipos de turno mostrando el listado de todos los tipos siendo posible crear nuevos con el botón del menú y modificarlos o eliminarlos desde los botones junto al elemento de la lista.

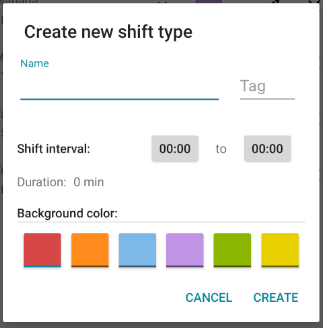


Ilustración 18 - Cuadro de dialogo para la creación de tipos de turno

Después de poder crear tipos de turno se desarrolla la parte de la asignación de turnos a un usuario utilizando el tipo de turno, un usuario y una fecha. El usuario y la fecha se obtienen al pulsar sobre un día del calendario grupal, correspondiendo con la fila del usuario. Para no asignar turnos por accidente se desarrolla un estado de edición utilizando un botón del menú que es necesario usar antes de realizar cualquier modificación del calendario. Esto únicamente está disponible para el coordinador del grupo.

Al pulsar sobre un día aparece un cuadro de dialogo con una lista desplegable con los tipos de turno disponibles y 7 botones correspondientes a los días de la semana, si no hay ninguno tipo disponible se le comunica al usuario y no aparece el cuadro de dialogo. La lista desplegable se implementa mediante el componente *Spinner* y un *SpinnerAdapter* proporcionando los nombres de los tipos de turno. Los botones de los días son utilizados para poder asignar turnos en varios días al mismo usuario de una vez.

Los cambios no se hacen efectivos de inmediato, es necesario guardar los cambios pulsando otra vez sobre el botón editar, que ha cambiado el icono al símbolo de guardar.

Entonces aparece un cuadro de dialogo con una lista de turnos indicando el tipo de modificación del calendario indicando si son nuevos, eliminados o cambios entre usuarios. Este cuadro pide confirmar los cambios para que se hagan efectivos en base de datos, si se cancela los cambio se mantienen y se permite realizar más antes de confirmar.

Una de las complicaciones en la asignación de turnos ha sido comprobar que no había conflictos de turnos al asignarlos en varios días a la vez y anular cambios al añadir y borrar un cambio no confirmado.

Para llevar el registro de cambios se ha utilizado un mapa clave valor donde la clave era el tipo de cambio y el valor una lista de turnos. Se tienen cuatro listas de turnos, los nuevos añadidos, los eliminados, los turnos antiguos que se deben borrar y los nuevos que añadir al modificar un turno asignándoselo a otro usuario. Se recorren estas listas por separado para realizar los cambios correspondientes y según se apliquen se van eliminando de la lista de cambios. Al asignarle un turno de un usuario sobre otro se debe eliminar de los turnos del primero y añadir del segundo, en Ilustración 19 - Aplicar un cambio de turno se puede ver como se aplicarían los cambios.



Ilustración 19 - Aplicar un cambio de turno

### Sprint 7

En este punto la funcionalidad principal de la sincronización de turnos establecidos por el coordinador con los usuarios estaría lista. Sin embargo aún quedaría por desarrollar las funcionalidades que aunque no son tan vitales son importantes como es el sistema de solicitudes de cambios de turno.

Se diseña de manera que para solicitar un cambio de turno el usuario deba pulsar sobre dos turnos en el calendario grupal, un turno suyo y uno de un compañero. Para seleccionar un turno se utiliza el pulsado largo para evitar que suceda sin querer y se comprueba primero que la casilla pulsada tenga asignado un turno y se tiene en cuenta a quien está asignado. Entonces se guarda ese turno como seleccionado distinguiendo si es propio o de otro usuario. Es posible cambiar de turno seleccionado siempre que sea de la misma naturaleza.

Una vez se seleccionan dos turnos de distinta naturaleza aparece un cuadro de dialogo donde se pueden verificar que los dos turnos seleccionados son los correctos antes de confirmar el cambio. Al confirmar el la solicitud se escribe en base de datos los dos turnos involucrados junto con la fecha de solicitud y el estado inicial de solicitado

Estos cambios es posible visualizarlos en una lista dentro de una actividad donde está la gestión de estas solicitudes. Para acceder a esta pantalla se utiliza un nuevo botón de menú en la pantalla de calendario.

En esta pantalla se dónde se interactúa con las solicitudes, el usuario destinatario de la solicitud puede confirmar o rechazar el cambio además de que el propio usuario que solicita puede cancelarla en cualquier momento. Una vez el otro usuario acepta es el coordinador el que debe aprobar el cambio o rechazarlo. Estas opciones aparecen en los elementos de la lista de las solicitudes en forma de dos botones. Todos los usuarios del grupo se les permite visualizar los cambios pero solo los implicados pueden realizar alguna acción.

### Sprint 8

Con el proceso de cambios completado se empiezan a desarrollar funcionalidades menores como el control de tiempos y las notificaciones. Para simplificar todo el tratamiento de tiempos se migra el tratamiento de fechas y tiempos de las clases de Java 7 a la biblioteca Joda Time. Se utiliza principalmente las clase Period y Duration que representan cantidad de tiempo.

Se realiza un recuento de las horas mensuales y semanales partiendo de los turnos que se muestran en los calendarios. Las horas mensuales van dirigidas a los usuarios para que controlen sus horas asignadas y las horas semanales van orientadas al coordinador para que controlen las horas máximas semanales que se les asigna.

Se incluye la opción de establecer las horas máximas semanales que pueden tener asignadas los usuarios desde la configuración de grupo utilizando una vista de entrada de datos con un deslizador vertical de valores donde se elige en número de horas.

El recuento de horas semanales del usuario se muestra al asignar turnos y se indica cuanto tiempo se está asignando, si se va a superar el máximo establecido se le indica al coordinador. Este valor total se va actualizando con cualquier modificación de los días seleccionados.

Además de tiempo se implementan las notificaciones de la aplicación, estas se resumen dos tipos, modificación de calendario y actualización de una solicitud de cambio. Para poder enviar notificaciones se utiliza el servicio FCM de Firebase.

Para comunicarse con FCM se utiliza un *token* y este se obtiene implementando un servicio que extienda FirebaseInstanceIdService, este recibe el *token* cuando se actualice en FCM. Se guarda entonces en base de datos ligándolo a un usuario y dispositivo.

Para recibir las notificaciones se implementa otro servicio que extiende FirebaseMessagingService y que es el encargado de recibir mensajes procedentes de FCM. En el mensaje se indica el tipo de notificación con datos como el grupo de trabajo para más información y según el caso el estado de la solicitud. Entonces según estos datos se muestra la notificación correspondiente. En la Ilustración 20 - Construcción de la notificación se puede observar cómo se construye una notificación, utilizando un título, cuerpo, estilo de la notificación, la acción, la cancelación al pulsar y el icono a mostrar.

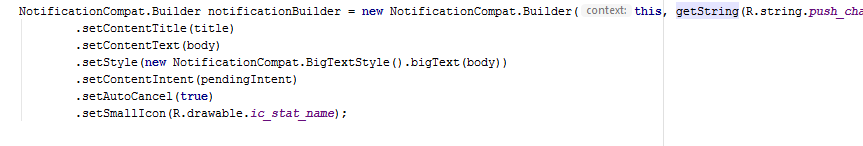


Ilustración 20 - Construcción de la notificación

Estos mensajes se envían desde el servidor al reaccionar a escrituras en base de datos dedicadas a las notificaciones. Al modificar el calendario de un usuario se apunta en esta sección de la base de datos incluyendo el nombre del usuario y el grupo donde se modifica, de esta manera con cada escritura se envía una notificación. También se reacciona a la escritura en las solicitudes de cambio de turno, enviando un mensaje a los dispositivos de los usuarios correspondientes en cada caso según el estado de la solicitud.

### Sprint 9

La iteración final se centra en la corrección de errores y pequeños ajustes después de una última reunión con la persona que hace de cliente para comprobar que la aplicación es funcional y contiene los aspectos que plasmó en los requisitos además de ser fácil de usar.

Además se ha enfocado en realizar un cambio visual de la aplicación que hasta ahora se había desarrollado únicamente la funcionalidad y no su estética. Se trabaja con los colores y símbolos para una interacción agradable y fluida con los usuarios. Se eligen los colores principales de la aplicación y se establece el icono de la aplicación y de notificaciones.

Los cambios técnicos principales son errores de navegación donde al cambiar la orientación del dispositivo o cambiar de pantalla al volver la vista perdía los datos del calendario. Esto se solucionó recuperando los fragmentos creados en el *PagerAdapter* guardando la referencia a los mismos y de esta forma no se recreaban incorrectamente. Junto a esto otros errores menores como arreglar que el indicador de carga no desaparecía cuando fallaba el inicio de sesión con Google y otros errores específicos como que no se manejaba el cambio de zona horaria al leer los turnos y esto producía una lectura incorrecta al comprobar el día con la fecha de los turnos.

Como cambios de diseño visual principales se destacan:

* La pantalla de recuperación de contraseña se migra a un cuadro de dialogo de la pantalla de inicio de sesión después de haber cogido experiencia con el desarrollo de múltiples cuadros de dialogo, en el inicio de sesión y el registro se ajustan las distancias de las entradas de datos para que el teclado tape el mínimo posible.
* En la pantalla principal se estilizan los elementos que muestran los grupos y se colorean según el rol del usuario en ese grupo, rojo para coordinador y azul para usuario.
* En el calendario los elementos de los días se estilizan y además colorean los encabezados. En el calendario grupal se cambia el botón de edición del menú a un botón flotante.
* Los botones de selección de días en la asignación de turnos se rediseñan para que sea más fácil identificar os botones marcados.
* Todas las vistas donde se muestran los datos de un turno se estilizan redondeando los bordes del cuadro con el color asignado.

Por último se procede a redactar esta memoria siguiendo las pautas establecidas por la Guía de Realización y Documentación para el proyecto de final de carrera. TODO\* Ref en biblio

# Descripción funcional

# Conclusiones y líneas de trabajo futuro

## Conclusiones

En este apartado se expondrán las conclusiones que se han podido sacar de la finalización del proyecto del Trabajo de Fin de Grado.

* Se ha conseguido desarrollar una aplicación Android funcional que cumple con los objetivos propuestos inicialmente de manera que pueda cumplir con las necesidades de los usuarios que utilicen la aplicación.
* Se ha conseguido construir una aplicación usable y atractiva utilizando las pautas proporcionadas por el estilo Material Design.
* Se ha comprendido el esfuerzo que lleva desarrollar un proyecto completo de principio a fin de un tamaño mayor a cualquier otro previamente desarrollado utilizando la ingeniería del software.
* Se ha desarrollado la aplicación sin experiencia previa en aplicaciones lo que ha llevado a un proceso de aprendizaje a lo largo de todo el proyecto. Esto ha llevado que en las últimas etapas se tuviera una buena compresión del funcionamiento y fuera posible mejorar algunas partes previamente desarrolladas.
* Se ha conseguido satisfacer los objetivos personales de aprender a desarrollar aplicaciones móviles, ser capaz de desarrollar un proyecto entero poniendo en práctica los conocimientos aprendidos en asignaturas del Grado.

## Líneas de trabajo futuras

La aplicación es perfectamente funcional y capaz de desempeñar el trabajo propuesto, sin embargo considero que no tiene la calidad suficiente para ser competente en el mercado.

Algunas características a mejorar serían por ejemplo:

* Ser capaces de identificar un usuario con una imagen y así ayudar a diferenciarlos
* Más opciones de personalización como cambiar los colores de los turnos del lado de cada usuario, opciones para desactivar las notificaciones desde la aplicación y otros aspectos que den comodidad al usuario.
* Se podría ampliar la funcionalidad del sistema de turnos de manera que en vez de preseleccionar que turnos se intercambian un usuario solicite un cambio y otros le hagan ofertas de cambios con sus turnos.
* Otra funcionalidad que se podría implementar sería un nuevo nivel de usuario que fuera el de “invitado” y que pudiera acceder al calendario de un trabajador. Este rol estaría orientado a familiares o pareja para poder organizarse mejor y que no sea necesario preguntarle al trabajador todo el tiempo.
* Al haberse desarrollado desde el principio sin ninguna experiencia en aplicaciones, la calidad del código admite mejoras en cuanto a modularidad y abstracción.
* La aplicación ha sido desarrollada completamente en inglés salvo en la documentación de las clases y métodos incluyendo toda la interfaz, queda pendiente traducirla al menos al Español.

# Bibliografía

* Turnos de trabajo. Emprendepyme.net – Actualizado el 21/11/2016 <https://www.emprendepyme.net/turnos-de-trabajo.html>
* ¿Qué es Android? Xatakandroid - Actualizado el 9 de febrero 2018 <https://www.xatakandroid.com/sistema-operativo/que-es-android>
* Cuota de mercado de Android. Kantar.com – Actualizado el 03/05 de mayo 2018 <https://es.kantar.com/tech/m%C3%B3vil/2018/mayo-2018-cuota-de-mercado-de-smartphones-en-espa%C3%B1a-primer-trimestre-2018/>
* Anroid SDK. Wikipedia – Actualizado el 09/10/2018 <https://es.wikipedia.org/wiki/Android_SDK>
* NoSQL. Genbeta.com – Actualizado el 28/04/2011 <https://www.genbeta.com/desarrollo/el-concepto-nosql-o-como-almacenar-tus-datos-en-una-base-de-datos-no-relacional?utm_medium=referral&utm_source=pulsenews>
* Material design – Actualizado el 09/11/2014 <https://elandroidelibre.elespanol.com/2014/11/que-es-material-design.html>
* Programación orientada a objetos. Wikipedia - Actualizado el 14/06/2018 <https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos>
* Android Studio. Developer.android.com – Actualizado el 25/04/2018 <https://developer.android.com/studio/intro/?hl=es-419>
* XML. Wikipedia – Actualizado el 20/02/2018 <https://es.wikipedia.org/wiki/Extensible_Markup_Language>
* JavaScript. Developer.mozilla.rog Actualizado 12/07/2018 <https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/JavaScript_basics>
* Firebase. Firebase.google.com – Actualizado en el 2018 <https://firebase.google.com/docs/android/setup?hl=es-419>
* GitHub.Wikipedia – Actualizado el 01/08/2018 <https://es.wikipedia.org/wiki/GitHub>
* Doxygen. Actualizado el 02/07/2018 <http://www.stack.nl/~dimitri/doxygen/>
* Word
* Visual Paradigm
* Project
* Udacity