Universitat Oberta de Catalunya

uoc.edu

AllPay, aplicación para gastos compartidos en Android

Javier Parra Patiño
Máster universitario de Desarrollo de aplicaciones para dispositivos móviles
Trabajo final de máster DADM

Eduard Martin Lineros

Carles Garrigues Olivella

06/2019

FICHA DEL TRABAJO FINAL

Título del trabajo:	*AllPay, aplicación para gastos compartidos en Android*
Nombre del autor:	*Javier Parra Patiño*
Nombre del consultor/a:	*Eduard Martin Lineros*
Nombre del PRA:	*Carles Garrigues Olivella*
Fecha de entrega (mm/aaaa):	06/2019
Titulación:	*Máster universitario de Desarrollo de aplicaciones para dispositivos móviles*
Área del Trabajo Final:	*Trabajo final de máster DADM*
Idioma del trabajo:	*Castellano*
Palabras clave	*Gestión, economía, control*

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

En este Trabajo Fin de Máster (TFM) se desarrolla una aplicación con la que dotar a los usuarios de una herramienta con la que poder analizar, controlar, registrar, prever y calcular cualquier gasto económico llevado a cabo por cualquier usuario dentro de un determinado grupo. Además, incluye sincronización en tiempo real entre todos los componentes del grupo, gestión de metas económicas y el análisis del balance total de cada uno de los componentes.

Aunque en el mercado existen multitud de aplicaciones de gestión económica de grupos, cabe destacar que ninguna de ellas ha conseguido dominar claramente este mercado. Esta ausencia de aplicación líder del sector es debido, entre otros factores, a que gran parte de dichas aplicaciones se complementan entre sí en cuanto a sus diferentes funcionalidades, su grado de usabilidad o la forma de gestionar sus usuarios. Ninguna de ella ha logrado totalmente convencer a un público exigente, acostumbrado a tener aplicaciones con multitud de funcionalidades sin perder un ápice de usabilidad. La aplicación desarrollada permite que cualquier usuario pueda, debido a sus numerosas funcionalidades y a su alto grado de usabilidad, realizar todas las actividades económicas asociadas a un determinado grupo independientemente de la naturaleza de estas.

Para el desarrollo de la misma se ha hecho uso del servicio de Google Firebase, de librerías externas como MaterialDrawer y de diferentes componentes incluidos en las últimas actualizaciones de Android, como *CardView o RecyclerView.*

Abstract (in English, 250 words or less):

In this Final Master Project (FMP) an application is developed to provide users with a tool with which to analyse, control, record, forecast and calculate any economic cost carried out by any user within a given group. In addition, it includes real time synchronization between all the components of the group, management of economic goals and analysis of the total balance of each of the components.

Although there are many applications in the market for the economic management of groups, it should be noted that none of them has managed to clearly dominate this market. This lack of industry-leading application is due, among other factors, to the fact that many of these applications complement each other in terms of their different functionalities, their degree of usability or the way they manage their users. None of them has totally convinced a demanding public, accustomed to having applications with a multitude of functionalities without losing an ounce of usability. The developed application allows any user, due to its numerous functionalities and its high degree of usability, to carry out all the economic activities associated with a given group regardless of their nature.

For the development of this application has made use of the service of Google Firebase, of external libraries like *MaterialDrawer* and of different components included in the last updates of Android, like *CardView* or *RecyclerView*.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Hoy en día cuesta imaginar nuestra sociedad sin aplicaciones para dispositivos móviles, muchas de ellas impensables hace tan solo unos años, y que se han convertido en prácticamente imprescindibles para cualquier persona, como por ejemplo, WhatsApp o Facebook. Actualmente, ante la creciente demanda, muchas empresas y organizaciones se encuentran investigando y desarrollando aplicaciones cuyo principal objetivo es satisfacer y facilitar las diferentes necesidades humanas, tales como acceso a información necesaria en cualquier circunstancia, prever los gastos de una comunidad u organizar la economía de un determinado grupo.

Actualmente muchos usuarios pertenecen a diversos grupos heterogéneos, como el grupo de los amigos del colegio o el de los compañeros de trabajo, en los que resulta complicado conocer en todo momento quién ha pagado cada producto o quién debe dinero a otro miembro del grupo. Como consecuencia de esta situación, es frecuente la aparición de problemas de confianza entre los diferentes integrantes del mismo ya que suele ser común que, por ejemplo, uno de los integrantes de dicho grupo no recuerde la cantidad exacta que ha prestado con anterioridad o se haga el despistado para no tener que pagar una deuda pendiente.

Ante dicha problemática este Trabajo Fin de Máster (TFM) propone como objetivo principal la creación de una aplicación con la que dotar a los usuarios de una herramienta con la que poder analizar, controlar, registrar, prever y calcular cualquier gasto económico llevado a cabo por cualquier usuario dentro de un determinado grupo (ver Figura 1). Este gasto económico podrá ser derivado de, por ejemplo, un producto o un servicio.

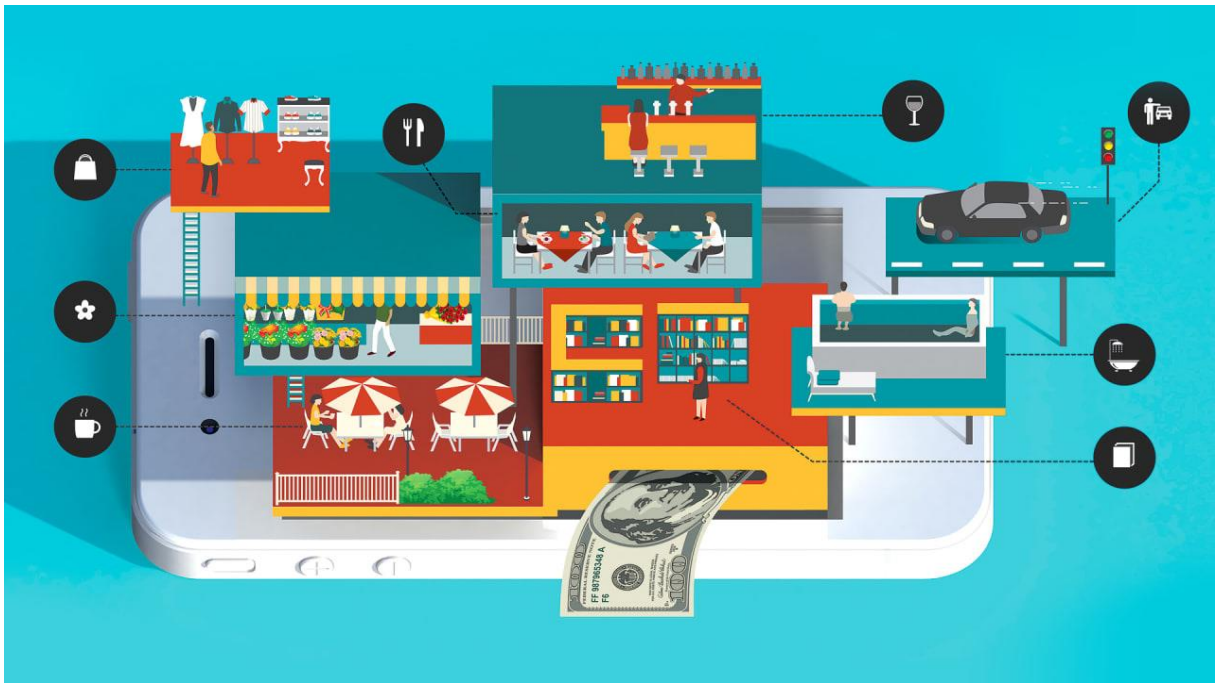

Figura 1: Representación de diferentes gastos económicos compartidos

Aunque en el mercado existen multitud de aplicaciones de gestión económica de grupos, cabe destacar que ninguna de ellas ha conseguido dominar claramente este mercado. Esta ausencia de aplicación líder del sector, como por ejemplo pudiera ser Booking en el sector turístico, es debido, entre otros factores, a que gran parte de dichas aplicaciones se complementan entre sí en cuanto a sus diferentes funcionalidades, su grado de usabilidad o la forma de gestionar sus usuarios. Es por ello que ninguna de estas aplicaciones de gestión económica de grupos ha logrado totalmente convencer a un público exigente, acostumbrado a tener aplicaciones con multitud de funcionalidades sin perder un ápice de usabilidad. Por ejemplo, es común que un usuario utilice una determinada aplicación para controlar la gestión económica en su grupo de amigos y otra distinta para llevar la contabilidad de su núcleo familiar. La aplicación por desarrollar permitirá que cualquier usuario pueda, debido a sus numerosas funcionalidades y a su alto grado de usabilidad, realizar todas las actividades económicas asociadas a un determinado grupo que este desee, independientemente del ámbito de estas. De esta manera, el usuario sólo tendrá que utilizar dicha aplicación en lugar de una diferente para cada grupo, servicio o evento.

Puesto que la aplicación debe de poder ser utilizado por un público generalizado, si se analizan el perfil de un usuario estándar y el patrón de uso existente en este tipo de aplicaciones, lo habitual es que el acceso al mismo se haga a través de un teléfono móvil. Es por ello que se necesita una estrategia de desarrollo que implemente la teoría de empezar por los dispositivos más pequeños (mobile first), priorizando estos dispositivos sobre, por ejemplo, los ordenadores.

1.2 Objetivos del Trabajo

Por ello, el objetivo principal de este TFM es el desarrollo de una aplicación con la que dotar a los usuarios de una herramienta con la que poder analizar, controlar, registrar, prever y calcular cualquier gasto económico llevado a cabo por cualquier usuario dentro de un determinado grupo.

Con la meta de satisfacer este objetivo principal se han definido los siguientes requisitos funcionales clave para el desarrollo de este, además de diferentes requisitos funcionales opcionales y requisitos no funcionales.

Los requisitos funcionales de la aplicación, aquellos que son funcionalidades claves requeridas por el sistema, se exponen a continuación:

- Cada usuario podrá registrar cuánto ha costado el producto consumido, el nombre de este (con una breve descripción), la etiqueta de su categoría, quién lo ha consumido y quién lo ha pagado.

- Sincronización en tiempo real de la información introducida en el producto para todos los miembros de un determinado grupo, en cualquier dispositivo del ecosistema.

- Acceso al historial con todos los productos y servicios consumidos desde la creación del grupo.

- Capacidad para indicar quién debe pagar el producto o servicio la próxima vez. Con esta funcionalidad se pretende igualar el gasto económico de todos los usuarios de un determinado grupo para de esta forma tener la menor cantidad de deudores posible.

- Un grupo de usuarios podrá establecer una meta de dinero común, con el objetivo de, por ejemplo, comprar un determinado producto o realizar un viaje. De esta forma se establecerá un objetivo común para el grupo por lo que todos tratarán de ahorrar para alcanzarlo.

- La aplicación deberá de dar soporte en múltiples idiomas, debido al carácter social de la misma.

Los requisitos funcionales opcionales de la aplicación, aquellos menos importantes para el objetivo principal de esta, se indican a continuación.

- Cualquier usuario podrá crear un nuevo grupo dentro de la aplicación y podrá añadir a tantos usuarios como sea necesario. Además, un usuario podrá pertenecer a tantos grupos de usuarios como desee, pudiendo seleccionar cada uno de estos desde la propia aplicación

- Estadísticas de los diferentes bienes y servicios consumidos, como por ejemplo el producto que más se ha consumido en el último mes, el último servicio pagado o quién ha realizado más anotaciones en el último mes.

- Capacidad de asignación de roles a los usuarios de un determinado grupo. El creador del grupo tendrá el rol de administrador y podrá aplicar este rol a tantos usuarios de su mismo grupo como desee. Los usuarios con dicho rol podrán borrar y modificar los gastos introducidos por cualquier usuario, salvo que este sea administrador, del grupo al que pertenezcan.

- El usuario podrá configurar si el gasto económico es derivado de, por ejemplo, un producto o un servicio a través de etiquetas personalizadas.

- Capacidad de guardar el dinero sobrante de un grupo en un apartado denominado bote con el que pagar el próximo producto. Siempre que se pague un producto, si el bote posee fondos, se le indicará al usuario su existencia y se restará de este, si el usuario así lo seleccionase.

- Posibilidad de que los usuarios pueden pagar a otros usuarios dentro de la propia aplicación haciendo uso del servicio de PayPal.

- Capacidad para indicar quién debe dinero al grupo y quién ha pagado en exceso. Además, en ambos casos, deberá de mostrar dicha cantidad.

Por último, los requisitos no funcionales de la aplicación, aquellos que son características del sistema debido a su propia naturaleza, se exponen a continuación:

- Deberá tener un diseño claro y preciso, con el objetivo de poder registrar los gastos económicos de un determinado grupo lo más rápido posible. De esta forma, el usuario podrá disfrutar con su grupo de amigos en lugar de estar anotando multitud de datos innecesarios en la aplicación.

- De la característica anterior se derivan las características que debe tener el camino, en inglés *path*, que el usuario debe seguir hasta conseguir registrar un determinado gasto. Este debe de ser mínimo, con el objetivo de poder anotar sin tener que pasar por multitud de pestañas.

- Se deberá de utilizar un vocabulario y un léxico neutro con el objetivo de poder ser entendido por todos los niveles culturales. De esta manera cualquier usuario, sea cual sea su nivel académico y social, podrá hacer un uso correcto de la aplicación.

- Deberá utilizar un rango de colores adecuados, para así ayudar a los usuarios que presenten algún tipo de dificultad visual, como por ejemplo el daltonismo. Del mismo modo, se deberá de utilizar una fuente adecuada.

- Es deseable que las funcionalidades clave estén diseñadas y funcionen correctamente en lugar de añadir multitud de funcionalidades adicionales. El usuario desea una aplicación sencilla y con alto grado de usabilidad, en lugar de una aplicación compleja con multitud de opciones adicionales.

1.3 Enfoque y método seguido

Debido a la propia naturaleza del TFM, se necesitará una metodología ágil de desarrollo basada en un desarrollo iterativo e incremental, siendo ésta a su vez sensible a modificaciones. Además, deberá estar centrada en los valores de investigación y prototipado pero sin descuidar en ningún momento aspectos como la gestión o los requisitos. En resumen, la metodología requerida necesitará poseer un origen ágil en la que los requisitos y las soluciones evolucionen en el tiempo según las necesidades propias del proyecto.

De entre el abanico de metodologías ágiles existentes para la realización de este proyecto [1], la metodología elegida es el método de desarrollo de sistemas dinámicos (en inglés Dynamic Systems Development Method (DSDM)) adaptada a las necesidades del mismo. Para solucionar el problema, su idea principal se basa en priorizar el ajuste del tiempo y de los recursos necesarios sobre el proveer multitud de funcionalidades adicionales al sistema [2, 3]. En esta metodología el concepto de adaptación gana peso a lo largo de la vida del proyecto, en detrimento de las clásicas en las que se realiza una previsión fija y estricta. Debido a esta adaptabilidad, se pueden realizar cambios a los requisitos iniciales sin que suponga una gran desventaja, siempre que se cumplan los requisitos del mismo.

La metodología ágil de desarrollo DSDM presenta como pilar fundamental la continua colaboración entre cliente y desarrollador, moldeándose de esta manera el proyecto hasta conseguir el resultado final deseado. En cuanto al equipo de desarrollo, éste es caracterizado por poseer gran libertad de acción y decisión, siendo en la mayor parte de las ocasiones autónomo. También es importante señalar la alta frecuencia de entregas parciales del proyecto con el objetivo de que éstas puedan ser verificadas y aprobadas para cerrar, en cierta media, esa fase del proyecto. Además no se requiere de la completa ejecución de una iteración para poder pasar a la siguiente. Por último se debe indicar que las pruebas son realizadas a lo largo de todo el ciclo de vida del proyecto, no sólo al final del desarrollo del mismo.

Debido a que la metodología se basa en la premisa de ajustar la funcionalidad en función del tiempo y de los recursos disponibles, se debe establecer una jerarquía de requisitos del proyecto estableciendo de este modo una prioridad a cada uno de ellos. Esto se realiza siguiendo el método MoSCoW, el cual presenta los siguientes niveles de priorización, explicados a continuación categorizados de mayor a menor prioridad:

- **Must have.** Indica que los requisitos son fundamentales para el desarrollo exitoso del proyecto, debiendo estar presentes en el producto final sin objeción alguna. Es la mayor de las prioridades.

- **Should have.** Señala requisitos que, aun siendo claves para el producto final, no implican su resolución inmediata. No obstante, deben de resolverse a corto plazo estando presentes en el sistema final.

- **Could have.** Muestra requisitos que, aun siendo importantes en el sistema final, pueden quedarse fuera de este si ocurrieran problemas de tiempo.

- **Want have.** Designan requisitos totalmente opcionales deseables de aparecer en el sistema final, pero sin ser estrictamente necesarios. Es la menor de las prioridades.

Durante el desarrollo de un proyecto que sigue la metodología DSDM se darán claramente cinco fases bien diferenciadas:

1. **Estudio de viabilidad.** El proyecto final es descrito en términos de costes, adaptación a la metodología DSDM, tiempo estimado, requisitos, tecnología necesaria, riesgos asumibles y procedimientos para llegar hasta los objetivos propuestos.

2. **Estudio de negocio.** Durante esta etapa, el equipo de desarrollo reflexiona acerca de lo que supondrá para ellos y para la empresa la realización del proyecto, analizando el impacto que tendrá éste en todas las entidades relacionadas con él.

3. **Iteración del modelo funcional.** Se recogen todos los requisitos identificados en las dos etapas anteriores transformándolos en modelos funcionales. De esta manera, se plantea el contenido y la estrategia a seguir en el proyecto.

4. **Iteración de diseño y construcción del sistema.** Se encargan de recogen los modelos funcionales creados en la etapa anterior y transformarlos en productos que satisfagan completamente las necesidades del usuario, construyendo de este modo la mayor parte del sistema. Como resultado se obtiene un producto con al menos un conjunto mínimo de reglas MoSCoW.

5. **Implementación.** Los usuarios verifican el correcto funcionamiento del sistema final siendo entrenados, si fuera necesario, para su utilización. Si faltara algún requisito crítico por cumplir, se puede volver a ejecutar el proceso desde el principio.

Las dos primeras fases son secuenciales y son realizadas, por norma general, una única vez en todo el proyecto. Las tres últimas son iterativas e incrementales realizadas con cada entrega, y requieren de una revisión por parte del usuario. El gráfico de estas etapas junto a las relaciones que se pueden producir entre ellas puede ser apreciado en la Figura 2.

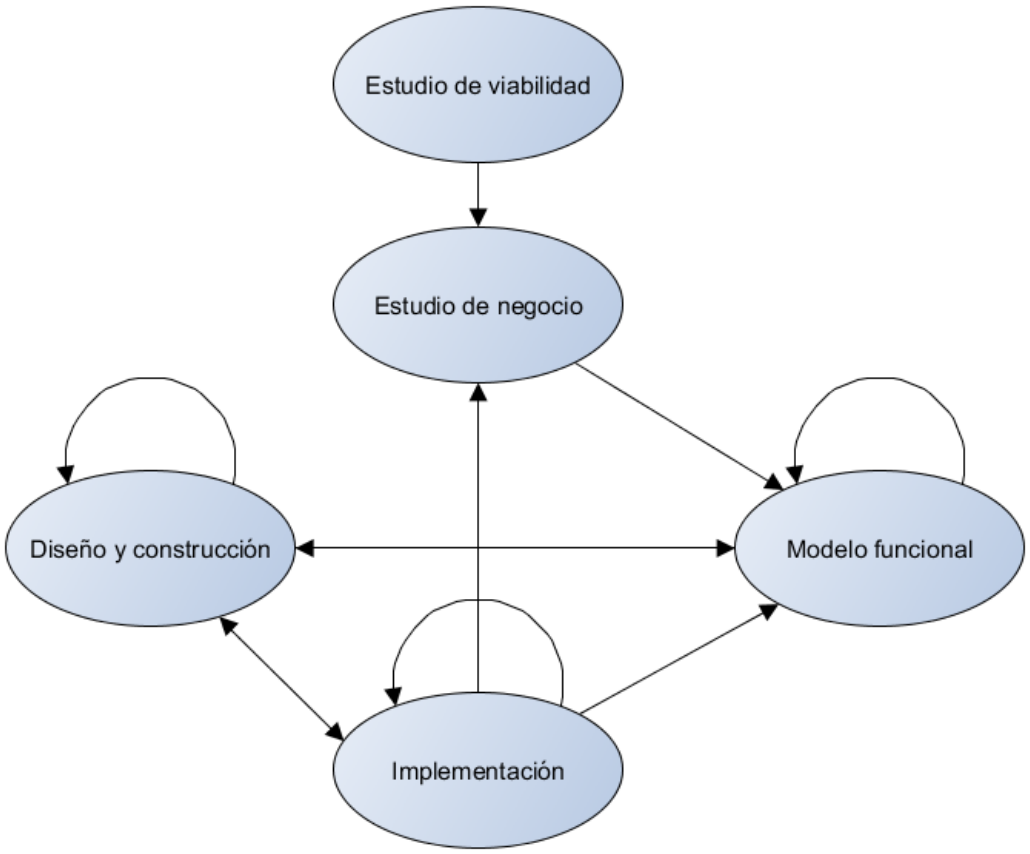

Figura 2: Esquema de las cinco fases de la metodología DSDM

La estrategia de desarrollo elegida es desarrollar un producto nuevo que cumpla con todos los requisitos funcionales principales y con todos los no funcionales. Además, se intentarán integrar tantos requisitos funcionales opcionales en cuanto a los recursos y el tiempo lo permitan. Se ha seleccionado esta estrategia de desarrollo debido a que ninguno de los productos actuales satisface los requisitos funcionales o los requisitos no funcionales requeridos por la aplicación final a desarrollar.

Además, no es posible adaptar dicha aplicación a un rango de mercado determinado puesto que estará enfocada a cualquier usuario. Este enfoque implicará la realización de un diseño válido para todo tipo de usuarios, sea cual sea su edad, género, nacionalidad o estatus social. Además, debido a que cada usuario necesitará analizar las cuentas de los diversos grupos a los que pertenezca, será necesario integrar dentro de la aplicación todos los grupos a los que pertenezca dicho usuario. Para ello fue diseñado de manera precisa, clara y concisa cada una de las funcionalidades y vistas de la aplicación, dotándole de un alto grado de usabilidad. De esta forma, el usuario puede saber en todo momento el grupo que está analizando.

1.4 Planificación del Trabajo

Los medios hardware utilizados durante la realización del TFM son expuestos a continuación:

- **Teléfono móvil**. Se ha utilizado un teléfono móvil con el que realizar las diversas pruebas de la aplicación.

- **Ordenador personal**. Necesario para la búsqueda de la información y el desarrollo tanto de la aplicación como de la documentación. Las características del mismo se pueden observar en la Tabla 1.

Tabla 1: Características técnicas del ordenador personal utilizado

Sistema Operativo	Windows 10 Home 64 bits
Procesador	Intel Core i7-6700HQ CPU 2.600GHz
Gráficos	NVIDIA GeForce GTX 960M
Disco Duro	931 GB
Memoria RAM	20 GB

Los principales medios software utilizados durante el desarrollo del presente proyecto se encuentran citados, junto a una breve descripción de cada uno de ellos, a continuación.

- **Atom**. Editor de texto altamente personalizable.

- **Android Studio**. IDE para la programación en Android.

- **BitBucket**. Aplicación para el control de versiones.

- **yEd**. Editor gráfico utilizado para la realización de los diagramas.

- **Trello**. Gestor de tareas.

- **Balsamiq Mockups 3.** Herramienta para la elaboración de wireframes y prototipos de baja fidelidad.

- **Axure RP 8.** Herramienta para la realización de prototipos de bajo y alta fidelidad interactivos.

Durante el desarrollo del proyecto se seguirán las siguientes fases de trabajo, enumeradas y explicadas a continuación (ver Figura 3).

- **Fase 1. Estudio del estado del arte.** En esta primera fase se analizarán las diferentes aplicaciones existentes para controlar la economía de un grupo con el objetivo de realizar un benchmarking en las diferentes tiendas de aplicaciones.

- **Fase 2. Diseño de la aplicación.** En esta etapa se analizan y seleccionan los diferentes elementos para el desarrollo de la aplicación. Esta fase se

divide en 5 partes las cuales siguen las fases del Diseño Centrado en el Usuario (DCU):

1. Definición del usuario objetivo y de los casos de uso.
2. Diseño conceptual de la aplicación y de su arquitectura.
3. Prototipado.
4. Evaluación del prototipo anteriormente desarrollado.

- **Fase 3. Implementación de la aplicación.** Durante esta etapa se desarrolla y se realizan todas las pruebas necesarias para definir el correcto funcionamiento de la aplicación de la aplicación. En caso de encontrar problemas ocasionados con los recursos o el tiempo, se puede volver a la fase 2.

- **Fase 4. Documentación.** Esta fase se desarrolla en paralelo al resto de fases y consiste tanto en la recopilación de información como en la generación de documentación de las diferentes partes de las que consta el presente TFM.

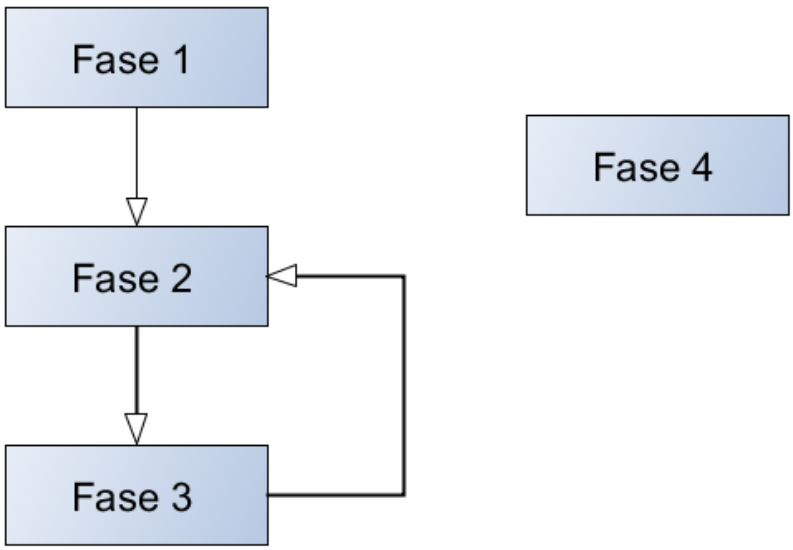

Figura 3: Fases de trabajo durante el proyecto

La planificación temporal con el número de horas que se dedicará a cada una de las tareas se muestra en la Tabla 2. El diagrama de Gantt con el número de horas y la situación en el tiempo de cada una de las tareas a realizar se puede observar en la Figura 4. Se dedicará el mismo número de horas durante los días laborales que durante los días festivos.

Tabla 2: Planificación de las tareas a realizar

Tarea	Duración en horas	Fecha de entrega

Estudio del Estado del Arte	3	
Diseño de la aplicación	40	03/04/2019
Definición del usuario objetivo	3	
Definición de los casos de uso	4	
Diseño conceptual	7	
Diseño de la arquitectura	9	
Prototipado	12	
Evaluación del prototipado	5	
Implementación	70	15/05/2019
Documentación	20	05/06/2019

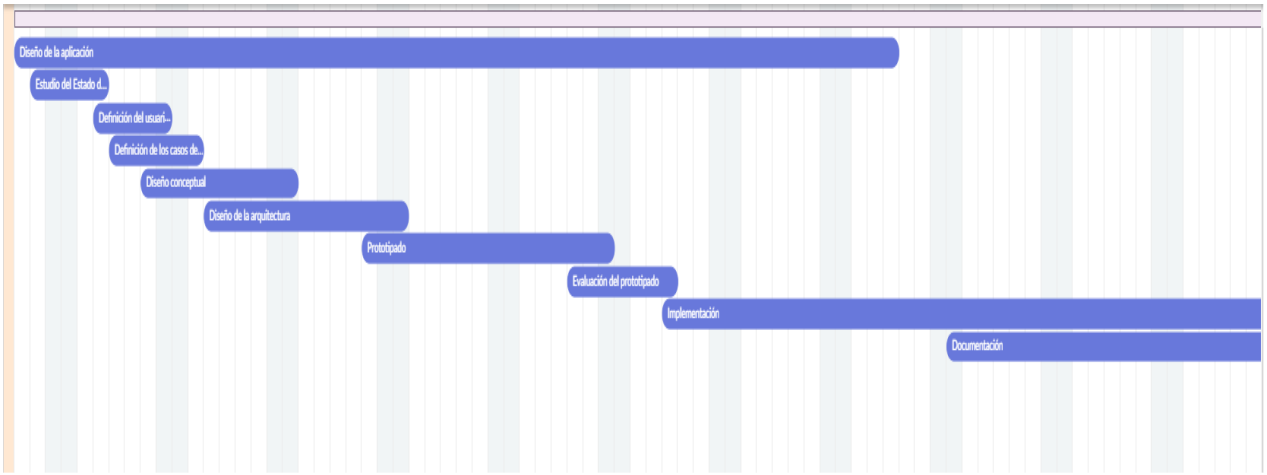

Figura 4: Diagrama de Gantt con la planificación de las tareas a realizar

Aunque toda la documentación se desarrollará al mismo tiempo que el resto del proyecto, durante las dos últimas semanas se terminará y se revisará exhaustivamente.

1.5 Breve sumario de productos obtenidos

Los productos finales obtenidos, ampliamente descritos en el resto de los capítulos, serán:

- Aplicación Android final.
- Memoria con el desarrollo de todas las tareas indicadas en la sección 1.4 Planificación del Trabajo.

1.6 Breve descripción de los otros capítulos de la memoria

El presente TFM contiene los siguientes capítulos, que serán descritos brevemente a continuación:

- **Capítulo 1. Introducción.** Se realiza una presentación general del proyecto a desarrollar, incluyendo la motivación que ha llevado a su desarrollo, los objetivos del mismo, el enfoque y el método elegido, su planificación y un breve sumario de los productos obtenidos.

- **Capítulo 2. Estado del arte.** Se analiza la situación actual de las diferentes aplicaciones existentes para controlar la economía de un grupo con el objetivo de realizar un benchmarking en las diferentes tiendas de aplicaciones.

- **Capítulo 3. Diseño de la aplicación.** Se analizan y seleccionan los diferentes elementos para el desarrollo de la aplicación.

- **Capítulo 4. Implementación de la aplicación.** Se explica el desarrollo de la aplicación de la aplicación.

- **ANEXO 1. Prototipo inicial.** Se muestran las capturas de pantalla del prototipo inicial de la aplicación desarrollada.

- **ANEXO 2. Manual de usuario.** Se indica al usuario la forma de proceder con el fin de instalar y ejecutar correctamente todos los elementos necesarios para la correcta ejecución de la aplicación.

2. Estado del arte

En el presente capítulo se presentan algunas de las aplicaciones realizadas que guardan relación con el objetivo de este TFM. Además, se investigan y se analizan todas las características y funcionalidades involucradas en dichas aplicaciones, examinando con detenimiento las características más importantes de cada una de ellas. Por último se expondrán las conclusiones obtenidas de dicho análisis.

2.1 Benchmarking en las tiendas de aplicaciones

Actualmente existen numerosas aplicaciones destinadas al control y al registro de los gastos producidos en un determinado grupo. Todas estas aplicaciones permiten introducir información sobre quién ha pagado un determinado producto o servicio, cuánto ha costado y cuántos usuarios están implicados en él. Toda esta información es automáticamente almacenada y sincronizaba en la nube, de manera que todos los usuarios pertenecientes a un determinado grupo poseen la información en tiempo real.

En este TFM se distinguen dos grandes grupos de aplicaciones dependiendo del ámbito de uso de la misma.

- **Aplicaciones de ámbito general.** Aplicaciones destinadas al control de cualquier gasto, ya sea de un servicio o producto, producido en un grupo de amigos.

- **Aplicaciones de ámbito específico.** Estas aplicaciones están dedicadas al registro de los gastos producidos por un determinado producto o servicio, como puede ser la compra de un regalo o el pago de una factura en un restaurante. Su principal característica se encuentra en su interfaz, exclusivamente diseñada para albergar un determinado tipo de pago.

Además de su ámbito de uso, en este TFM se distingue un criterio adicional de organización, según el modelo de negocio que posea una aplicación:

- **Aplicaciones de pago.** El usuario descarga la aplicación en su dispositivo previo pago del precio establecido por el desarrollador.

- **Aplicaciones parcialmente gratuitas.** El usuario descarga la aplicación gratuitamente, pero tiene acceso limitado al contenido de esta. Para desbloquear todas sus funcionalidades se debe realizar uno o varios pagos indicados por el desarrollador.

- **Aplicaciones gratuitas.** El usuario descarga la aplicación gratuitamente, sin ningún coste económico. Según su finalidad y la manera de generar nuevos ingresos, esta categoría engloba seis grupos diferentes:

- De marca. Se persigue es obtener un beneficio reputacional, es decir, mejorar la imagen de marca. Muy utilizado por las empresas.

- Servicio. La aplicación es utilizada por la empresa como parte de un servicio de ayuda a los usuarios que así lo necesiten, optimizando el tiempo de respuesta y ofreciendo atención personalizada.

- Publicidad. Se introduce publicidad en la aplicación con el objetivo de obtener beneficio a través del número de clics o del número de usuarios que ven un determinado anuncio.

- Tráfico de datos. Los usuarios obtienen algún tipo de recompensa a cambio de ceder su información o rellenar una serie de encuestas. A su vez, la empresa recibe ingresos al vender a otras compañías dichos datos.

- Comercial. Proporcionan un canal de venta de productos o servicios a los usuarios.

- Audiencias. Su principal objetivo es conseguir una gran base de usuarios para posteriormente utilizarlos con otro fin distinto.

Además de controlar y registrar cualquier actividad económica que el usuario desee, muchas aplicaciones ofrecen la funcionalidad de realizar pagos a otros usuarios a través de transferencias bancarias o de un servicio externo como PayPal. Por este motivo muchos bancos han empezado a desarrollar este tipo de aplicaciones puesto que, además de integrar su servicio de transferencias, posibilitan la captación de nuevos clientes. De este modo, la aplicación se convierte en un elemento de marketing de la entidad. Las dos aplicaciones más populares desarrolladas por entidades bancarias pueden consultarse a continuación.

- **Appatxas**. Desarrollada por Kutxabank, permite registrar el coste de cualquier producto o servicio consumido. Como elemento diferenciador, realiza recomendaciones sobre actividades para llevar a cabo y pagar en grupo [4].

- **Twyp**. Creada y mantenida por ING, permite realizar todas las funciones asociadas al análisis y almacenamiento de las actividades económicas de un determinado grupo. Su funcionamiento se basa en la recarga y en la retirada de crédito dentro de la propia aplicación, de manera similar a PayPal [5].

A pesar de existir las aplicaciones anteriormente citadas desarrolladas por diferentes entidades bancarias, un gran número de usuarios desconfían de ellas puesto que se tratan de aplicaciones desarrolladas por un banco distinto al suyo. Actualmente, debido al factor anteriormente citado, las aplicaciones que gozan de mayor popularidad entre los usuarios son aquellas que han sido desarrolladas por empresas externas. Tras realizar un estudio de mercado tanto en Android

como en iOS, las aplicaciones más populares en cuanto a descarga y a número de usuarios activos son mostradas a continuación [6-7].

- **SettleUp**. Es una de las aplicaciones más populares del mercado, con más de un millón de descargas en Android. Está diseñada principalmente para compartir gastos durante un determinado viaje, puesto que indica quién debería de pagar el siguiente gasto con el objetivo de igualar la aportación económica de todos los usuarios. En Android es gratuita, pero en iOS es de pago [8].

- **Splitwise**. Ofrece la posibilidad de enviar correos electrónicos a las personas que deban dinero, para de este modo recordarles sus deudas. Disponible tanto en iOS como en Android, presenta más de 5 millones de descargas en este último sistema operativo.

- **Billr**. Especializada en compartir los gastos de bares y restaurantes, indica la cantidad que cada comensal debe pagar a partir de los siguientes datos: cuánto cuesta cada plato, el número de ellos y quién los ha compartido, el número de bebidas y la cantidad reservada a la propina. Únicamente disponible para el sistema operativo iOS.

- **Divvy**. También especializada en compartir los gastos de bares y restaurantes, se caracterizada por poder analizar el ticket final a través de la foto subida a la aplicación por parte del cliente. A través de la aplicación de esta técnica de visión por computador, no es necesario que los usuarios introduzcan manualmente el número de platos y el coste de estos. Únicamente disponible para el sistema operativo iOS.

- **Spotme**. Creada principalmente para compañeros de piso, permite la introducción de las facturas comunes y quién las ha pagado para posterior dividirlos entre los usuarios del grupo. Además, muestra gráficos e históricos de los gastos producidos. Únicamente disponible para el sistema operativo iOS.

- **Fairshare**. Especialmente diseñada para compañeros de piso, permite llevar el control tanto de los gastos económicos como de las tareas domésticas a realizar. Igualmente, incorpora un sistema de logros en el que se van otorgando puntos a medida que se hacen las diferentes tareas del hogar.

2.2 Conclusiones obtenidas

Tras analizar las diferentes aplicaciones en el mercado, y estudiar sus principales funcionalidades y características, han sido obtenidas las siguientes conclusiones. Todas ellas se incluirán en el diseño y en el desarrollo de la aplicación final, la cual tendrá entre sus propiedades las indicadas a continuación.

- Debido al número de aplicaciones disponibles en el mercado, la aplicación será liberada gratuitamente en el momento de su lanzamiento. De esta

forma, muchos de los usuarios que no pueden abordar el pago de otras aplicaciones se convertirán en potenciales clientes de la aplicación por desarrollar. Debido a esta característica, los ingresos iniciales serán obtenidos a través de la integración de publicidad no invasiva.

- Deberá de disponer de una amplia gama de funcionalidades, las cuales serán descritas en la sección “Funcionalidades de la aplicación”.

- Como trabajo futuro, se deberá de incluir herramientas de visión por computador con las que detectar y reconocer automáticamente el texto de una determinada factura para de este modo no obligar al usuario a introducir manualmente los gastos.

Además, gracias al proceso de benchmarking realizado, se ha podido extraer que existe un menor número de aplicaciones con la temática analizada para el sistema operativo Android. Del mismo modo, analizando las estadísticas de diversos medios especializados, se puede obtener que el sistema operativo Android se encuentra globalmente más extendido que el sistema operativo iOS (ver Figura 5) [10]. Es por ello que la aplicación será desarrollada para el sistema operativo Android. Como trabajo futuro, teniendo en cuenta el aumento de tiempo y de recursos disponibles, se debería de implementar también dicha aplicación para el sistema operativo iOS.

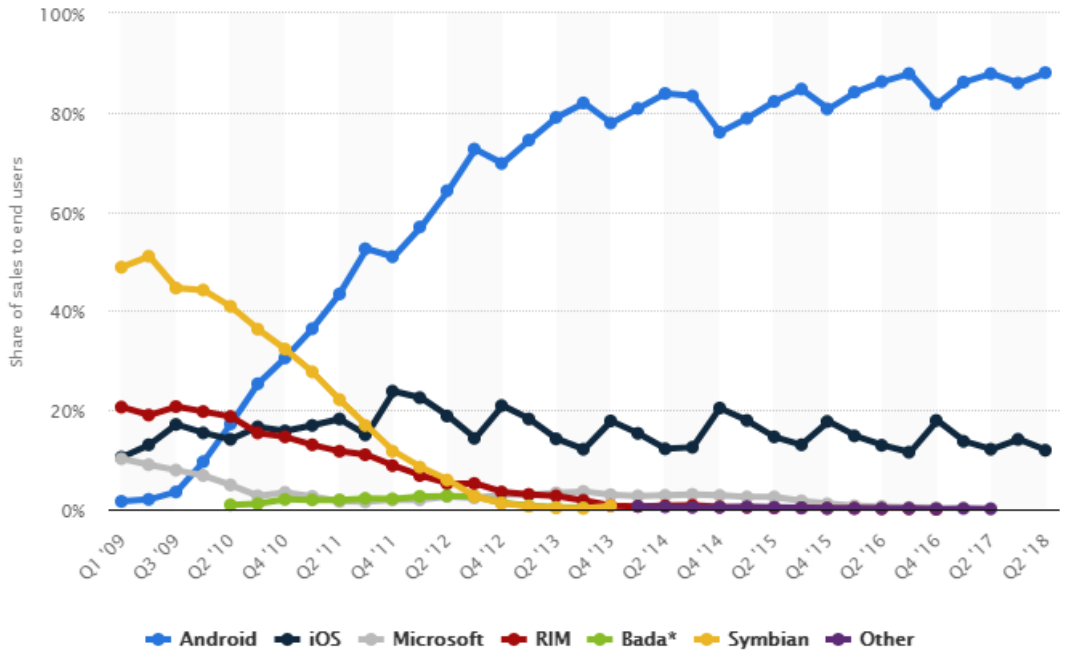

Figura 5: Análisis del sistema operativo de los móviles vendidos entre el primer cuatrimestre de 2009 y el segundo cuatrimestre de 2018 [9].

3. Diseño de la aplicación

Durante el desarrollo del presente capítulo se explicará de forma precisa y detallada el proceso completo de diseño de la aplicación a desarrollar capaz de analizar, controlar, registrar, prever y calcular cualquier gasto económico llevado a cabo por cualquier usuario dentro de un determinado grupo.

Como se ha mencionado en la sección "*1.3 Enfoque y método seguido*", durante el proceso de creación de la aplicación se ha seguido la metodología ágil DSDM, adatándola a las necesidades del proyecto fin de máster. Ésta se basa en un desarrollo iterativo e incremental, en donde los requisitos y soluciones pueden variar en el tiempo según las necesidades impuestas por el proyecto.

La lista de los requisitos de la aplicación, organizados jerárquicamente siguiente la regla *MoSCoW,* puede ser observada a continuación.

- Must have:
    - Capacidad para registrar un producto o bien consumido.
    - Sincronización en tiempo real entre todos los usuarios de un mismo grupo.
    - Acceso al historial con todos los bienes y servicios.
    - Capacidad para indicar quién debe pagar el producto o servicio la próxima vez.
    - Diseño claro y preciso.
    - Funcionalidad para mostrar el balance total sobre cuánto ha pagado cada integrante del grupo.
- Should have:
    - Estadísticas de los diferentes bienes y servicios consumidos por un grupo.
    - Gestión de metas económicas.
    - Asignación de diferentes etiquetas a un bien o servicio consumido.
    - Posibilidad de que los usuarios paguen a través del servicio de PayPal.
    - Envío de un email a los usuarios que deban dinero a otros usuarios de su mismo grupo.
- Could have:

    - Capacidad para indicar si un gasto es presente o futuro.
    - Funcionalidades de gestión de grupos: crear, editar y finalizar.
    - Capacidad para programar un gasto periódico.
    - Gestión de roles.
- Want have:
    - Posibilidad de añadir fotos a un producto o bien consumido.
    - Capacidad para la gestión de un bote común.

Por último, cada una de las secciones que forman este capítulo se corresponden con iteraciones realizadas para el desarrollo completo de la aplicación.

3.1 Usuarios y contexto de uso

Durante la presente sección se mostrará el estudio de requisitos de usuario realizado, para de esta manera conocer cómo será el usuario focal y secundario que harán uso de la aplicación por desarrollar. De este modo, podrán ser adaptados tanto el diseño como las funcionalidades de la aplicación final a sus necesidades. Para ello, se ha hecho uso principalmente de la técnica de observación e investigación contextual, mediante la cual se ha podido extraer cómo utilizan los usuarios las diferentes aplicaciones existentes en el mercado con una funcionalidad similar a la que se pretende desarrollar en su lugar habitual de uso. Además, se ha aplicado la técnica de la entrevista, para de estar forma conocer la opinión de los usuarios en cuanto las características a mejorar en las aplicaciones utilizadas, los fallos encontrados en estas o las funcionalidades innecesarias incluidas.

3.1.1 Definición genérica del público objetivo

La aplicación por desarrollar estará dirigida a cualquier usuario que posea un dispositivo móvil con conexión a internet, necesario para realizar la sincronización entre los diferentes integrantes del grupo. Si no posee conexión a Internet, los cambios se subirán en el momento en el que esta se adquiera, de manera similar a como ocurre con las aplicaciones de mensajería instantánea. Analizando en detalle el tipo de usuario que hará uso de manera más activa del producto, el llamado usuario focal, han sido deducidas las características mostradas a continuación.

- Será un usuario de una elevada iteración social que se relaciona frecuentemente con familiares y amigos.

- Pertenecerá a una edad comprendida entre 18 y 65 años. Los menores de 18 años suelen tener un menor nivel adquisitivo por lo que no necesitarán hacer un uso frecuente de la aplicación a desarrollar ya que los bienes y servicios consumidos durante su tiempo de ocio están usualmente limitados. Por otro lado, las personas de más de 65 años no suelen utilizar un gran número de aplicaciones, por lo que la probabilidad de que posean y utilicen la aplicación a desarrollar es menor que con un rango de edad menor. De este modo, y siguiendo un análisis generalista, dichos grupos no harán un uso frecuénteme de la aplicación.

- El género y el país en el que se haga uso de la aplicación no es importante ya que esta persigue un objetivo común: controlar los gastos de un grupo de usuarios. Es decir, la aplicación no está enfocada a ningún género o nacionalidad. Es por ello que será necesario implementar una traducción de la aplicación a los principales idiomas de comunicación, como el inglés, el francés o el castellano.

- Debido al carácter social del usuario, será necesario la posibilidad de integrar dentro de la aplicación los diferentes grupos a los que pertenezca ya que existe una alta probabilidad de que un usuario se encuadre en más de un grupo de usuarios. Además, cada usuario podrá completamente adaptar la aplicación a sus necesidades, pudiendo configurar, por ejemplo, los diferentes grupos a los que pertenece.

Los principales conceptos de diseño a tener en cuenta durante el desarrollo de la aplicación, relacionados con las motivaciones de uso y las necesidades de los usuarios, son mostrados a continuación.

- La aplicación deberá tener un diseño claro y preciso, con el objetivo de poder registrar los gastos económicos de un determinado grupo lo más rápido posible. De esta forma, el usuario podrá disfrutar con sus familiares o grupo de amigos en lugar de estar anotando multitud de datos poco útiles en la aplicación.

- De la característica anterior se derivan las características que deberá tener el camino de dicha aplicación, en inglés path, que el usuario deberá seguir para poder registrar un determinado gasto. Este deberá ser mínimo, con el objetivo de que pueda anotar sin tener que pasar por multitud de pestañas.

- Se deberá de utilizar un vocabulario y un léxico neutro con el objetivo de poder ser comprendido por cualquier nivel cultural. De esta manera cualquier usuario, sea cual sea su situación académica o social, podrá hacer un uso apropiado de la aplicación.

- La aplicación utilizará un rango de colores adecuados, para así ayudar a los usuarios que presenten algún tipo de dificultad visual, como por ejemplo daltonismo. Del mismo modo, se deberá de utilizar una fuente adecuada.

- Siguiendo los fundamentos de la metodología DSDM seleccionada, es deseable que las funcionalidades clave estén diseñadas y funcionen correctamente en lugar de añadir multitud de funcionalidades adicionales. El usuario desea un producto sencillo y con alto grado de usabilidad, en lugar de uno producto complejo con multitud de opciones superfluas adicionales.

3.1.2 Contexto de uso

Tras la aplicación de las técnicas anteriormente citadas, observación contextual y entrevista, se ha obtenido el principal escenario en el que el usuario focal y secundario harán uso de la aplicación. Este escenario será normalmente un lugar de ocio, ya sea en el interior o en el exterior, por lo que la aplicación será utilizada de manera rápida y concisa, sin demorar la tarea que el usuario quiere realizar. Dicho escenario viene determinado por una serie de características indicadas a continuación.

- Al ser un local de ocio, en la mayoría de las ocasiones el usuario de la aplicación no tendrá mucha libertad de movimientos, es decir, sólo dispondrá de un pequeño espacio físico a su alrededor por el que moverse. Esto significa que, a la hora de diseñar los diferentes menús de la aplicación, las pestañas y las opciones más utilizadas o que posean una relación de consecuencia deberán de estar próximas entre sí.

- Se debe tener en cuenta que, debido a su carácter social, la aplicación será frecuentemente utilizada en un escenario con poca luz, como por ejemplo durante la noche o dentro de un bar, por lo que se deberá de utilizar una relación de colores apropiados.

- Además, en consecuencia con la primera característica, deberá de poder ser utilizada con tan solo una mano, por lo que se tendrá que simplificar al máximo posible el diseño de la misma.

- Se deberá de incluir un sistema de notificaciones, incluso a través del envío de un correo electrónico, para recordar a un usuario determinado el pago de una deuda pendiente. Dichas notificaciones también se podrán configurar para poder ser enviadas a posteriori, como por ejemplo tres días después, puesto que es común que un usuario no quiera estar pendiente de abonar sus deudas en el momento en el que se produzca el gasto, por estar en una fiesta o celebración, dando lugar a su posterior olvido.

- Debido a la naturaleza del escenario principal de uso, durante una fiesta o una celebración, es posible que el usuario de la aplicación no esté con sus capacidades físicas y mentales completas debido al consumo de alcohol. Es por ello que será necesario, además de un diseño simple y

rápido, la inclusión de un léxico claro y de una fuente precisa, con el fin de remarcar, por ejemplo, las funcionalidades más importantes de configuración o de registro del gasto.

3.2 Diseño conceptual

Tras el estudio y el análisis de los requisitos de usuario realizados a lo largo de esta sección, se han configurado dos fichas de persona y dos escenarios a modo de ejemplo del usuario y posibles escenarios de utilización de la aplicación. Ambos representan datos ficticios, basados en los análisis y estudios previamente realizados.

3.2.1 Primera ficha de persona y escenario de uso

La ficha introductoria de la primera persona puede ser consulta en la Figura 6. El escenario principal de esta se puede consultar a continuación.

Figura 6: Ficha de la primera persona

Tras planificarlo durante mucho tiempo, Luis, su novia y sus amigos se han ido de viaje a Dublín durante las festividades de San Patricio. Después de ver el desfile, van a un pub para comer un estofado y tomarse unas pintas al más puro estilo irlandés. Tras la copiosa comida, piden la cuenta. Paga Marta, la novia de Luis, y el gasto es anotado en la aplicación del producto.

A continuación, caminan un poco para bajar el estofado pero, sin previo aviso, comienza a llover por lo que entran en otro pub, para así refugiarse de la lluvia. Como viene siendo habitual durante el viaje, piden una ronda de pintas para todos, anotando el gasto de esta en la aplicación. Tras acabar cada uno su pinta, Luis propone tomar una nueva ronda. Su amigo José indica, con muy mala cara, que no quiere otra pinta, alegando dolores de tripa posiblemente causados por el gran estofado que horas antes se habían comido. Gracias a la aplicación pueden ver que Noelia, novia de José, es la que menos ha pagado por lo que en esta ocasión, a fin de igualar la aportación de cada uno a los gastos comunes, es la seleccionada para pedir la próxima ronda de pintas.

Tras esto, Noelia apunta en la aplicación la ronda de pintas pedida, quitando a su novio de los deudores de dicho producto ya que no han pedido nada para él. En su lugar, José se pide un refresco, pero no lo anota en la aplicación puesto que es un producto que ha pagado exclusivamente para él.

Por la noche, toca el momento del ajuste de cuentas en el grupo. Gracias a la aplicación, analizan todos los gastos producidos durante el día y se produce el pago de las deudas.

3.2.2 Segunda ficha de persona y escenario de uso

La ficha introductoria de la segunda persona puede ser consulta en la Figura 7. El escenario principal de dicha persona se puede consultar a continuación.

Figura 7: Ficha de la segunda persona

Elena, como gran parte de los jueves cuando no están en época de exámenes, queda con sus amigos para salir por la noche. Mientras esperan a que lleguen los demás, ella y su mejor amiga se compran un helado. Su amiga paga y anota la cantidad en la aplicación, puesto que Elena no lleva suelto en ese momento.

Una vez llegan todos, van a cenar al Burger King. Como Elena necesita cambio para poder pagar más tarde en la discoteca, decide pagar toda la cuenta. En lugar de estar haciendo cálculos sobre la mesa, Elena les indica a sus amigos que lo va a registrar en la aplicación para así no perder tiempo e irse a la discoteca.

Mientras Elena baila en la discoteca, una amiga le pide 5€ para poder pagar una cerveza puesto que a ella se le ha olvidado el dinero. Elena se los presta y en menos de un segundo, gracias al intuitivo diseño de la aplicación, lo registra para que no se le olvide.

A los tres días, la aplicación automáticamente manda un correo electrónico a todos aquellos que aún no han pagado a Elena sus deudas. Gracias a dicha aplicación ha podido recuperar su dinero sin necesidad de hablar con cada uno de ellos.

3.2.3 Flujos de interacción

El árbol de navegación de la aplicación (ver Figura 8) representa la estructura de navegación y los flujos de interacción de la misma.

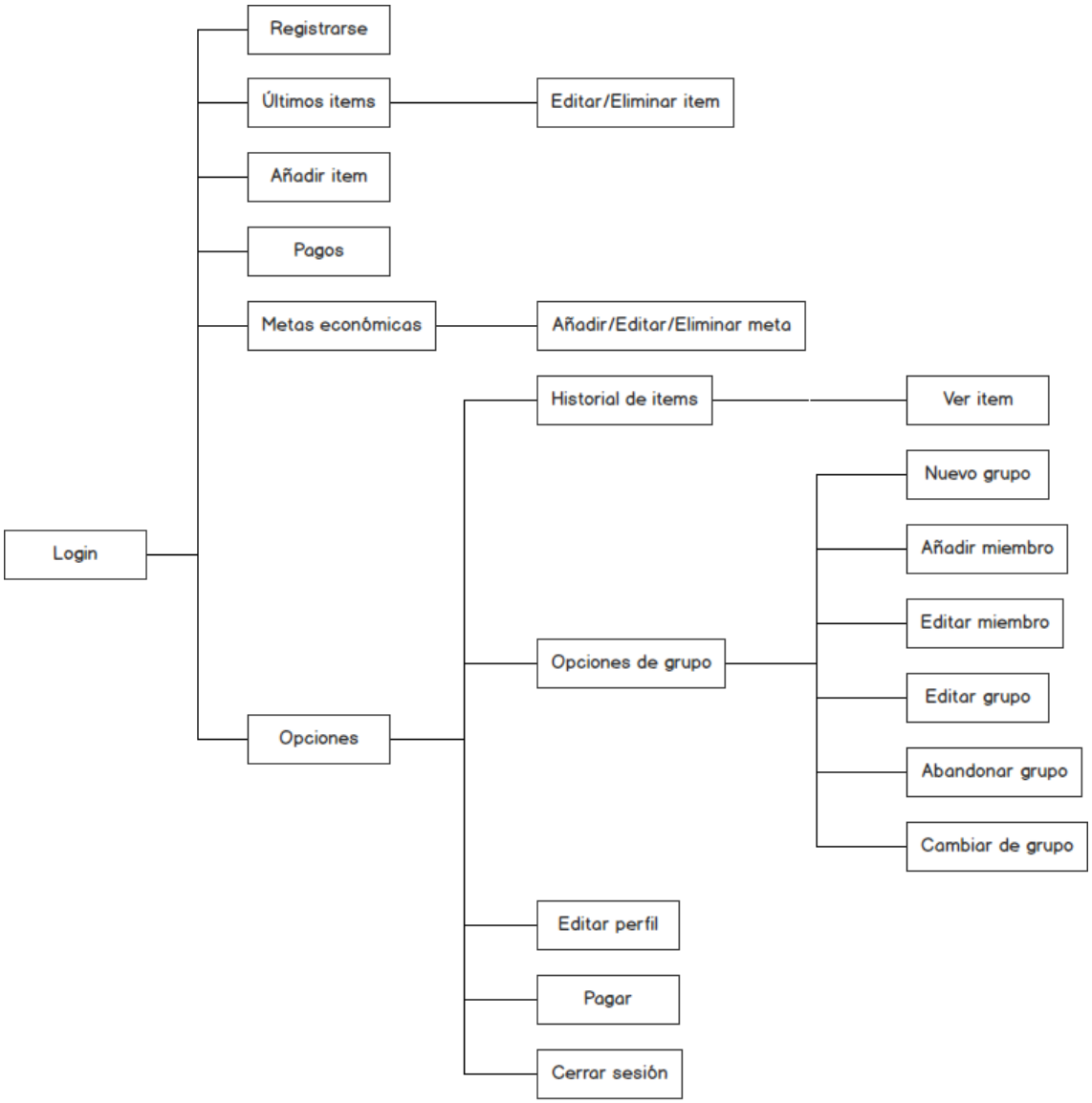

Figura 8: Árbol de navegación

En primer lugar, el usuario es dirigido a la pantalla *Login*. En ella podrá iniciar sesión o, si aún no ha creado la cuenta, darse de alta en el sistema rellenando un sencillo formulario. Una vez iniciado sesión, es redirigido a la pantalla *Últimos items*, la cual es la pantalla por defecto de la aplicación. Si el usuario ha iniciado sesión con anterioridad y no la ha cerrado en el dispositivo, será direccionado directamente a esta pantalla, sin necesidad de hacer login. De este modo, el usuario podrá acceder más rápidamente a la pantalla que desee.

Una vez en la pantalla *Últimos items*, el usuario podrá analizar la situación económica del último grupo que haya visitado. Por ejemplo, podrá revisar la aportación económica de cada uno de los componentes del grupo. Además, podrá visualizar el componente del grupo que deberá pagar el próximo producto o servicio, con el fin de igualar la aportación económica de cada uno de ellos.

Adicionalmente, en dicha pantalla de pestañas, el usuario podrá seleccionar diversas opciones de configuración a través de un menú emergente, llamado *Opciones*. En este menú podrá elegir una de las siguientes cinco opciones, las cuales darán lugar a un cambio de pantalla. Todas ellas son explicadas a continuación.

- *Historial de ítems.* El usuario podrá consultar todos los ítems consumidos por un grupo desde que este se creó.

- *Editar perfil***.** El usuario podrá modificar y actualizar su perfil dentro de la aplicación.

- *Opciones de grupo.* El usuario podrá crear un nuevo grupo de usuarios, modificar diversas opciones de cada uno de los grupos que pertenezca, como por ejemplo su nombre. Además, a través de la opción *Añadir miembro* y *Editar miembro*, podrá añadir o editar respectivamente los componentes del mismo en una nueva pantalla. Adicionalmente, podrá salirle del grupo con opción *Abandonar grupo* o, si es administrador de un determinado grupo, cerrarlo con la opción *Finalizar grupo*. Finalmente, podrá cambiar grupo visualizará cada uno de los grupos de los que sea miembro pudiendo seleccionar cualquiera de ellos.

- *Pagar.* A través de una ventana, el usuario será redirigido a la página de Paypal para poder así pagar a otro usuario.

- *Cerrar sesión*. El usuario podrá cerrar su sesión en el dispositivo actual.

Si el usuario se sitúa en la pantalla con pestañas*,* podrá acceder a otras cuatro pantallas diferentes en las que podrá consultar multitud de información derivada de la situación económica del grupo. Estas cuatro pantallas son organizadas en una estructura de pestañas como se ha comentado anteriormente. Todas ellas se enumeran a continuación.

- *Últimos items*. Es una de las pantallas principales de la aplicación. Su funcionalidad y contenido han sido explicados al comienzo de esta subsección.

- *Añadir producto o servicio*. Es una de las pantallas principales de la aplicación. En ella, el usuario podrá añadir un producto o servicio que haya sido abonado por cualquier componente del grupo junto a una serie de características del mismo, como por ejemplo su precio o el nombre de los usuarios que lo hayan comprado.

- *Pagos*. Es una de las pantallas principales de la aplicación. Cualquier usuario podrá consultar las deudas de cada uno de los componentes del grupo con el objetivo de que, tras el abono de las mismas, todos los usuarios hayan aportado la misma cantidad económica. Es otras palabras, en dicha pantalla se podrá visualizar quién le debe a quién y qué cantidad.

- *Metas económicos*. Podrán ser consultadas todas las metas económicas fijadas en el grupo. A través de las opciones *Añadir meta* o *Editar meta*, se abrirá una nueva pantalla en la que cualquier usuario podrá añadir una nueva meta, como por ejemplo tener en el bote una determinada cantidad de dinero, o editar una existente.

El usuario podrá volver a cualquier página de la estructura en pestañas o del submenú de opciones en cualquier momento y sea cual sea la pantalla en la que esté en ese momento.

3.3 Prototipado

Durante la siguiente subsección se mostrará todo el proceso de prototipado realizado a fin de diseñar todas las distintas funcionalidades y pantallas de la aplicación.

3.3.1 Wireframes

En primer lugar, a fin de tener una idea básica sobre cómo debían ser implementadas las demás pantallas, fueros diseñados y construidos los wireframes de dos de las principales pantallas de la aplicación: la pantalla *Últimos items* (ver Figura 9) y la pantalla para *Añadir Producto o Servicio (*ver Figura 10). Para ambas pantallas se tomó como referencia el árbol de Tomando como referencia el árbol de navegación de la subsección "*3.2.3 Estructura de navegación y flujos de interacción".*

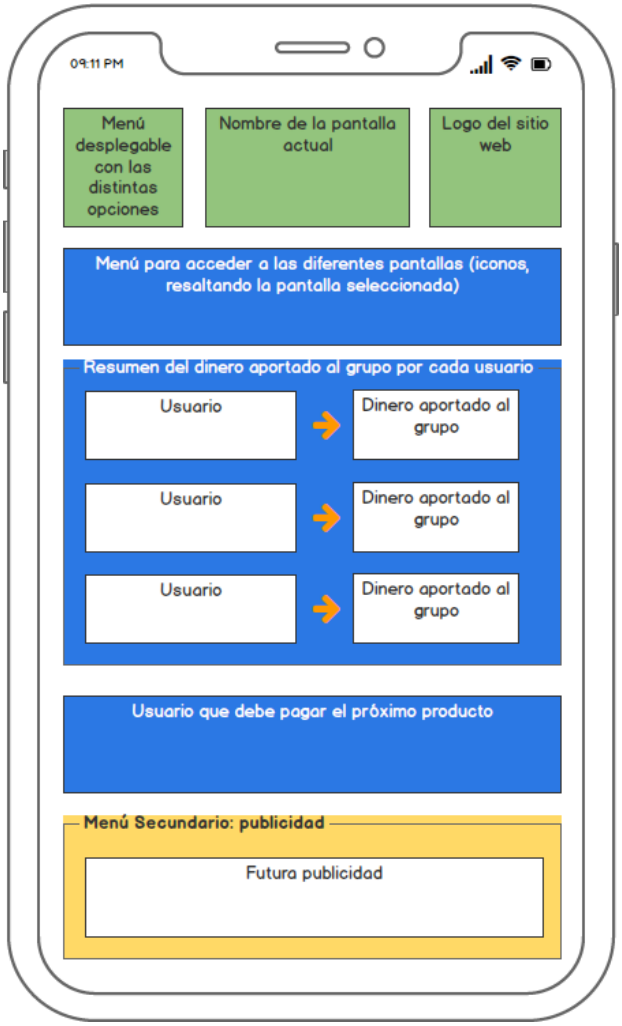

Figura 9: Wireframe de la pantalla *Principal* de la aplicación

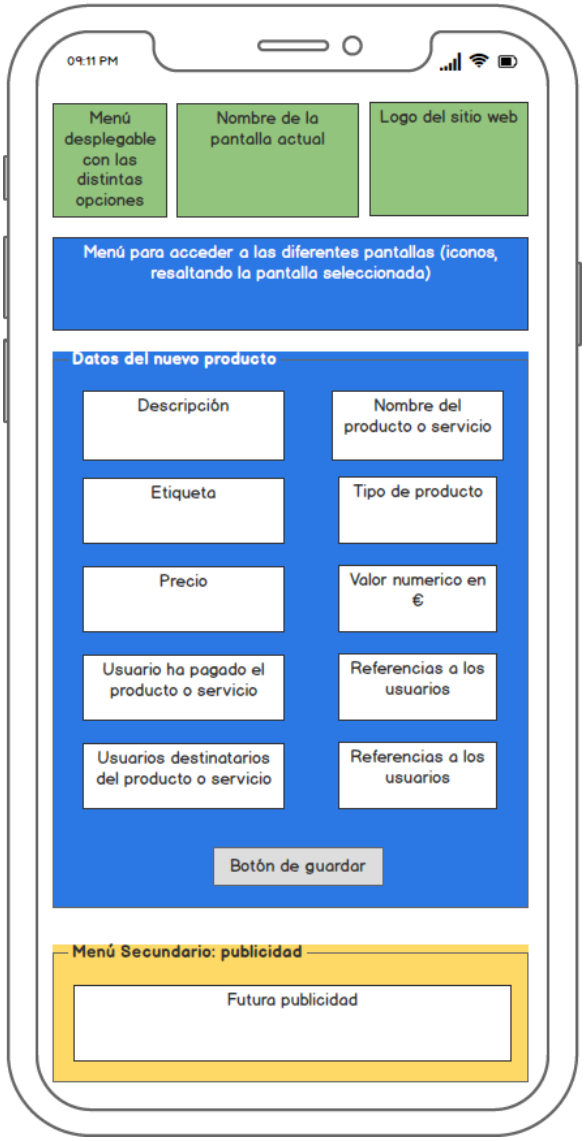

Figura 10: Wireframe de la pantalla *Añadir Producto o Servicio*

Los dos wireframes desarrollados están basados en un diseño responsive, de manera que el contenido de estos se ajusta al tamaño de la pantalla del dispositivo Android. Además, en ambas pantallas se ha remarcado por colores cada uno de los módulos presentes, diferenciando su nivel de jerarquía. Adicionalmente, cada uno de estos colores representa el módulo que primero verá el usuario, y en el cual prestará más atención, al acceder a cualquiera de las dos ventanas. De mayor a menor orden jerárquico, los colores que he utilizado son el azul, el verde y el amarillo.

3.3.2 Prototipo de baja fidelidad

A fin de confeccionar un prototipo de baja fidelidad en el que mostrar las diferentes funcionalidades, pantallas y los diferentes flujos de interacción de la aplicación, fue realizada una etapa de investigación previa. Como resultado de dicha investigación se obtuvo una tabla de inspiración, comúnmente conocidas

como *moodboards*, en el que se muestran los diferentes elementos de diseño de interfaz (ver Figura 11).

Figura 11: Tabla de inspiración de la aplicación a desarrollar AllPay

Después de realizar los primeros esbozos a lápiz de las principales pantallas de la aplicación, fue implementado el prototipo interactivo de baja fidelidad. Las capturas de cada una de sus pantallas pueden analizarse en el **Anexo I**. A sí mismo, dicha interactividad puede probarse en el siguiente enlace https://he4sg1.axshare.com.

3.4 Definición de los casos de uso

En esta sección se detallan los diferentes casos de uso que se han definido en la fase de análisis del proyecto. Un caso de uso representa una clase de funcionalidad dada por el sistema como un flujo de eventos. Así mismo se especifican los diferentes actores que pueden utilizar la aplicación especificando

las tareas que pueden llevar a cabo y las dependencias o relaciones que hay entre ellas. El conjunto de todos los casos de uso constituye el modelo de casos de uso, el cual describe la funcionalidad completa del sistema. Los casos de uso no son solo una herramienta para especificar los requerimientos del sistema, sino que además dirigen el diseño del mismo, su implementación y las pruebas, es decir, guían o dirigen el proceso de desarrollo.

En la Figura 12 se muestra el diagrama de casos de uso en el que se describe la funcionalidad global de la aplicación, en la que aparecen los diferentes casos de uso que se relacionan con los requerimientos funcionales. En ella se puede apreciar que aparece un único usuario, que se corresponde con el que utilice la aplicación y una **Base de Datos (BBDD)** que interviene para dar respuesta a las peticiones de interacción del usuario.

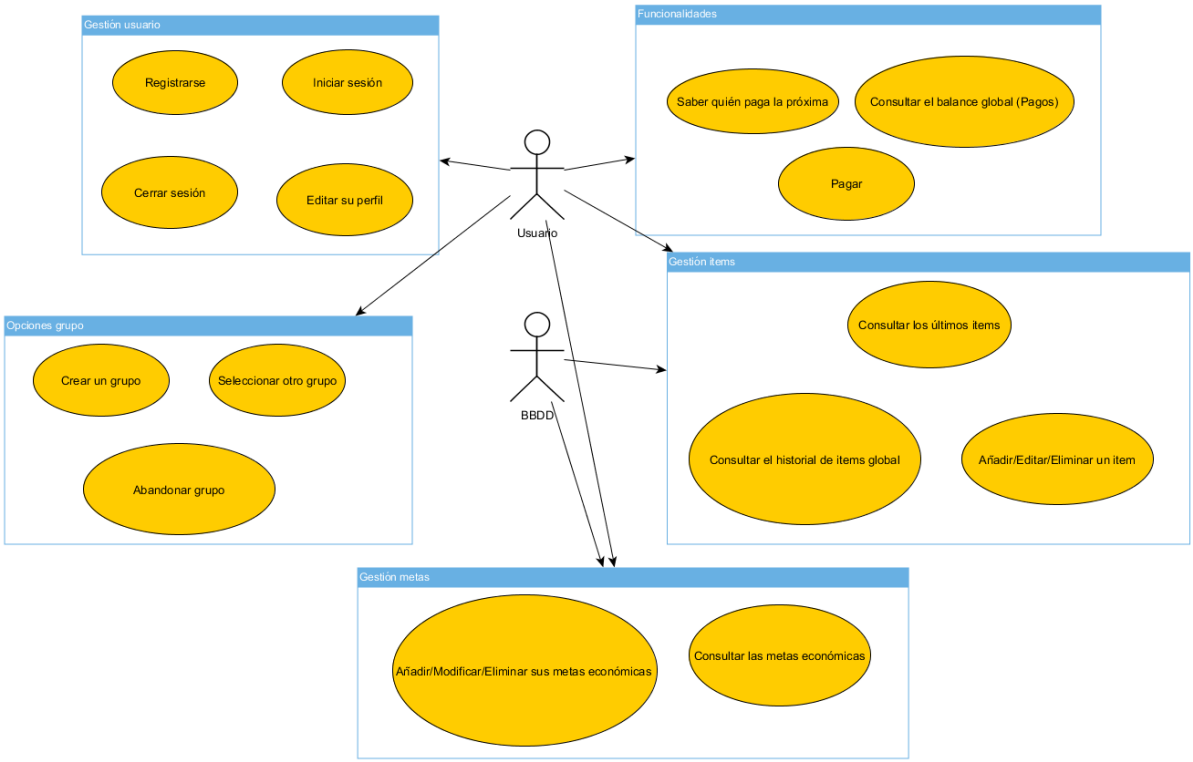

Figura 12: Casos de uso de un usuario

Si un usuario determinado posee el rol de administrador, además de los casos de uso mostrados en la Figura 12, tendrá los casos de uso indicados en la Figura 13.

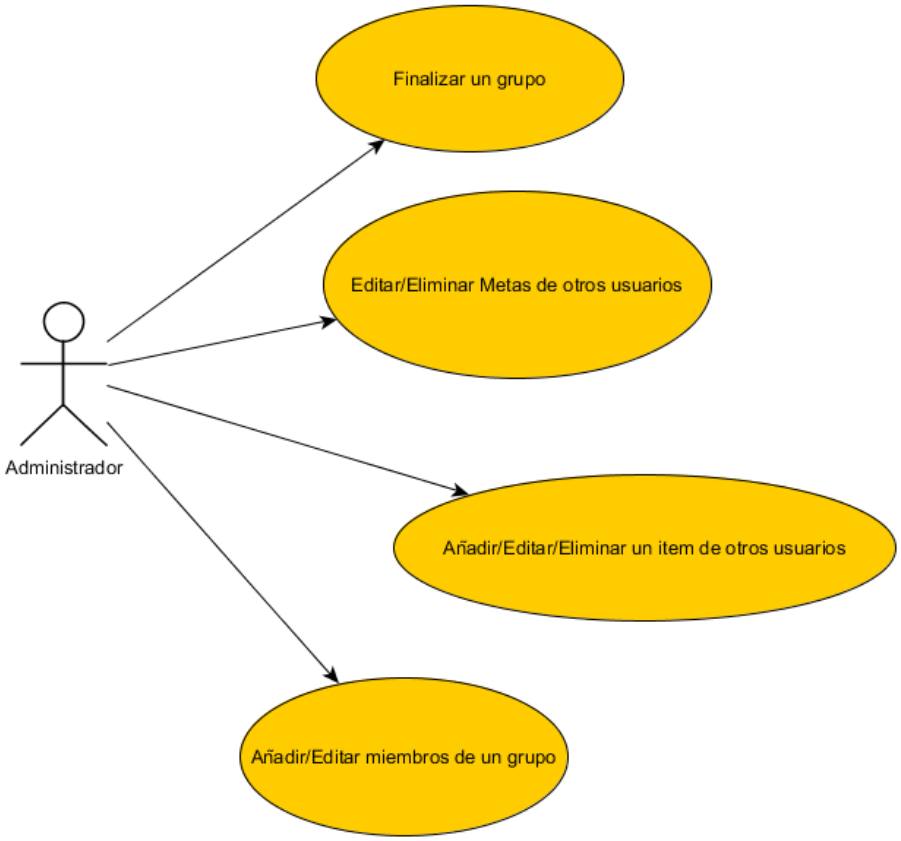

Figura 13: Casos de uso de un administrador

3.5 Diseño de la arquitectura

Debido a las características y funcionalidades de la aplicación expuestas en los anteriores capítulos y subsecciones de esta memoria, para el desarrollo de la aplicación se ha utilizado el patrón de arquitectura **Modelo Vista Controlador (MVC)**, ya que permite separar los datos de una aplicación, la interfaz de usuario y la lógica de negocio en tres componentes distintos, tal como se muestra a continuación. Por tanto, los diagramas de clases obtenidos se dividen en tres capas: presentación, controlador y persistencia.

- **Capa de presentación/Vista.** La capa de presentación se encarga de mostrar la información al usuario y gestionar la interacción del mismo con la aplicación. Por tanto, se incluyen aquí todas las clases que gestionan todas las Activities[1] o Fragments[2], así como las clases Adapter dentro de los Fragments y Activities, que permiten la visualización de los diferentes RecyclerViews[3] y también las clases de los cuadros de diálogos personalizados.

- **Capa de dominio/Controlador.** Contiene la lógica de negocio de la aplicación. Recibe, trata y responde a los eventos enviados por el usuario, y por la propia aplicación.

- **Capa de persistencia/Modelo.** Se compone de las clases que permiten el acceso a la base de datos, para realizar inserciones, modificaciones y eliminaciones de datos. Todas estas peticiones a la Base de Datos remota (alojada en Firebase), así como las cargas y descargas de cualquier tipo de dato, se realizan dentro de las diferentes clases que importan Firebase.

A continuación se muestran los diagramas de paquetes correspondientes a cada una de las capas anteriormente mencionadas.

[1] Una Activity o Actividad es un componente de la aplicación que ofrece la ventana con la que el usuario puede interactuar.
[2] Los *Fragments* o Fragmentos representan un comportamiento de la interfaz en la actividad.
[3] Las RecyclerViews permiten la visualización dentro de la interfaz de las listas dinámicas.

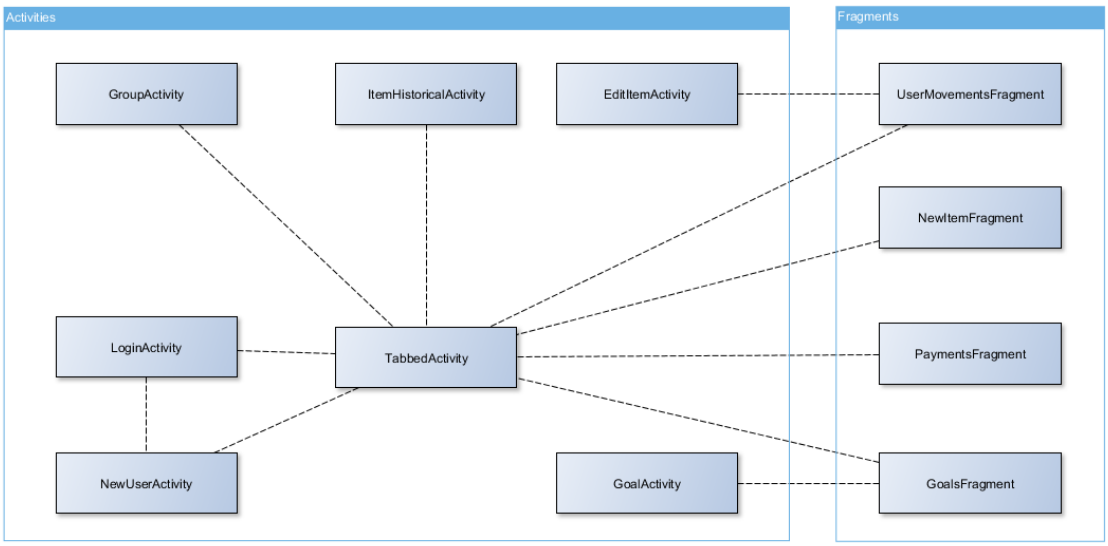

Figura 14: Diagrama de paquetes de Presentación

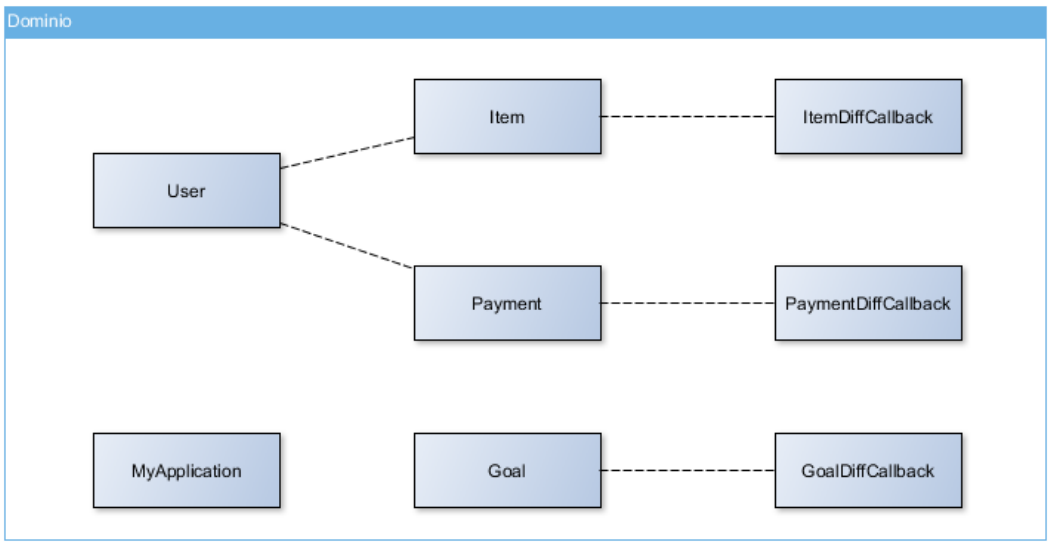

Figura 15: Diagrama de paquetes de Dominio

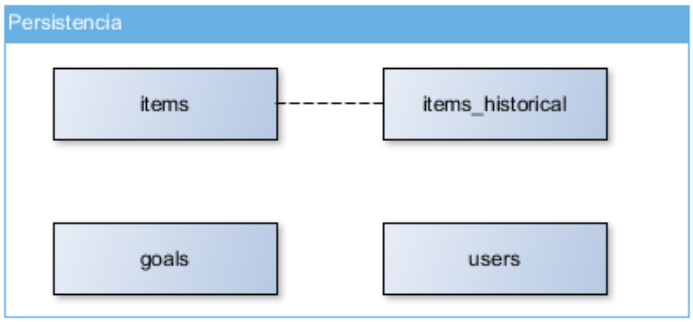

Figura 16: Diagrama de paquetes de Persistencia

3.5.1 Diseño de las entidades y clases

Durante esta subsección se explicarán las capas de Dominio más representativas anteriormente citadas.

La especificación de la clase Goal se muestra en la Figura 17. Dicho clase representa las metas económicas establecidas en un determinado grupo. Tiene como atributos un identificar único (*uid*), la cantidad de dinero total que se quiere recaudar (*goal*), la cantidad de dinero actualmente recaudada (*raised*) y el nombre de dicha meta (*subject*). Como métodos contiene dos constructores y los *getters* y *setters* de cada uno de los atributos.

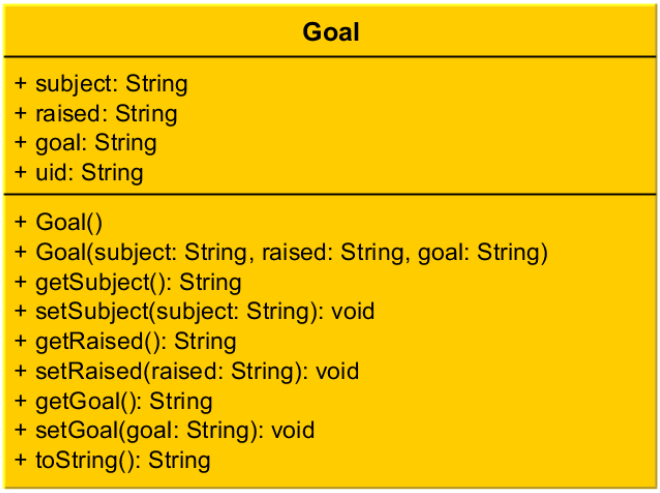

Figura 17: Diagrama de clases de Goal

En la Figura 18 se muestra la clase User, la cual representa a cada uno de los usuarios de la aplicación. Presenta los atributos básicos correspondientes a los datos personales del usuario (nombre, apellidos, email y contraseña) y el *nick* (*username*). De manera análoga a la clase Goal, contiene dos constructores y los *getters* y *setters* de cada uno de los atributos.

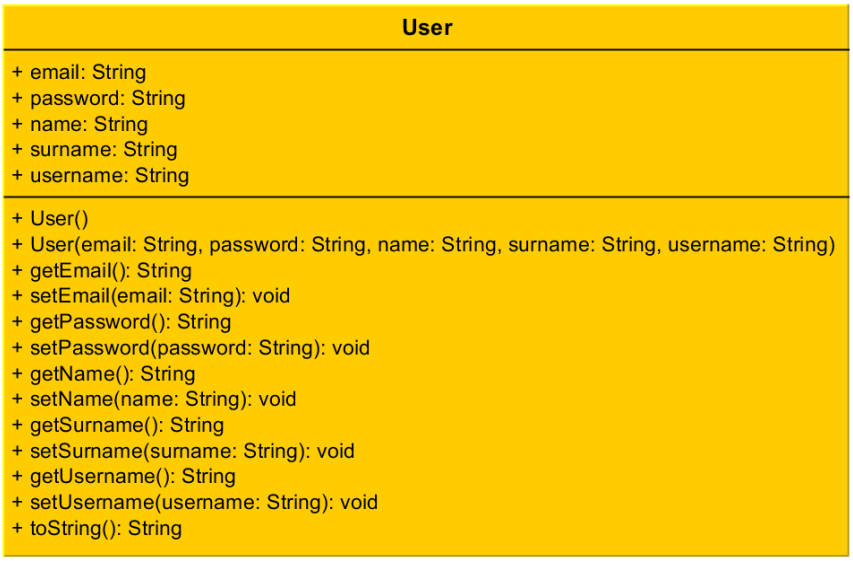

Figura 18: Diagrama de clases de User

La clase Item se muestra en la Figura 19. Tiene como objetivo representar cada uno de los productos o servicios consumidos en un grupo. Como atributos presenta la descripción de dicho item (*description*), si es un producto o un servicio (*type*), el precio (*price*), un identificador único (*uid*), la lista de usuarios que lo han pagado (payers_*users*) y la lista de usuarios que han hecho uso de dicho item (*receivers_users*). Al igual que las dos clases anteriores, contiene dos constructores y los *getters* y *setters* de cada uno de los atributos.

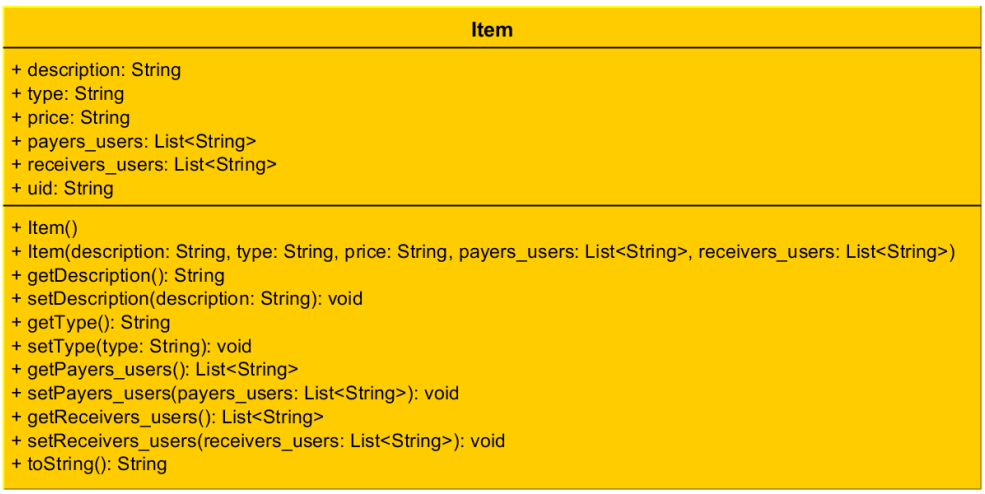

Figura 19: Diagrama de clases de Item

Por último, en la Figura 20 se muestra la clase Payment, la cual simboliza el balance total de cada usuario dentro un grupo, es decir, la cantidad total que han pagado de más o de menos. Tiene como atributos el nick del usuario (*username*) y el balance total de dicho usuario (*balance*). De la misma manera que las demás clases, contiene dos constructores y los *getter* y *setters* de cada uno de sus atributos.

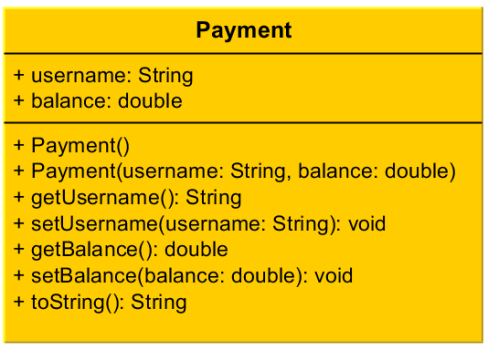

Figura 20: Diagrama de clases de Payment

3.5.2 Diseño de la base de datos

En esta fase se determinó la estructura de los datos o elementos de información con los que va a trabajar la aplicación. Hay que tener presente que se trata de una aplicación orientada a dispositivo móvil, que poseen ciertas limitaciones en cuanto a memoria y capacidad de computación, por tanto, se ha tratado de

simplificar y usar los datos de manera eficiente. Dado que los datos que manejará la aplicación (usuarios, items, metas y pagos) requieren gran capacidad de almacenamiento y que la aplicación debe de ofrecer servicio actualizado en tiempo real, estos deben residir en un servidor y se debe acceder a ellos a través de un servicio web. Por ello, después de realizar el diseño conceptual de la base de datos, se eligió usar el servicio Realtime Database de Firebase. Se ha utilizado para su diseño el modelo Entidad-Relación. El modelo Entidad-Relación (ER) es el modelo conceptual más utilizado y extendido para el diseño conceptual de bases de datos.

En primer lugar se diseñará la tabla **users** (ver Figura 21). En ella se almacenarán los principales datos de los usuarios definidos en la clase User: *email*, *name*, *surname*, *password* y *username*. Los tres primeros elementos se corresponden con los datos personales del usuario: su correo electrónico, nombre y apellidos. Estos tres elementos servirán para ofrecer un *feedback* personalizado al usuario. Además, el email se utiliza, junto a la contraseña, para poder hacer el inicio de sesión en la aplicación y además, para tener constancia de qué usuario utiliza la aplicación de manera inequívoca, ya que el email debe de ser único y nunca será como el de otro usuario.

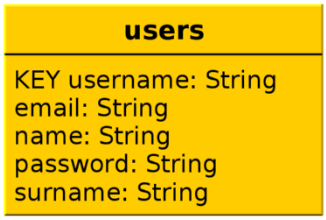

Figura 21: Tabla users

Debido a que Firebase no acepta el carácter punto (.) como clave primaria y con el objetivo de simplificar la referencia de un usuario determinado dentro de la aplicación, fue necesario introducir el atributo username con el que también identificar de manera inequívoca un usuario.

Al tratarse de datos de carácter personal, se aplica la Ley Orgánica de Protección de Datos (LOPD)[4]. Según la cual, todo usuario tiene derecho de acceso, modificación y rectificación de sus datos personales usados en la aplicación. Además, los datos recogidos se utilizarán con el único fin propuesto en la aplicación, quedando así almacenados y protegidos para otros usos y acceso de terceros.

Los campos de tabla **goals** se describen en la Figura 22, definidos en la clase Goal. En ella se almacenarán los atributos de una meta: cantidad de dinero total (*goal*), cantidad de dinero actualmente alcanzada (*raised*), una breve descripción (*subject*) y por último su identificador único (*uid*).

[4] https://www.boe.es/buscar/act.php?id=BOE-A-1999-23750

Figura 22: Tabla goals

De manera similar a las dos tablas anteriores con sus respectivas clases, la tabla **items** guarda gran similitud con la clase Item (ver Figura 23). En ella se almacenarán los atributos de un item: una breve descripción (*description*), la lista de los usuarios que han pagado el item (*payers_users*), su precio (*price*), la lista de los usuarios que han hecho uso de dicho item (*receivers_users*), el tipo de tipo (*type*) y su identificador único (*uid*).

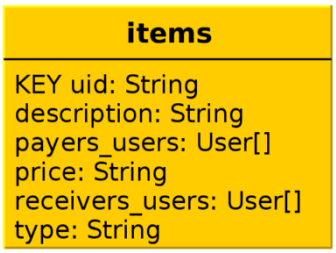

Figura 23: Tabla items

Por último, la tabla **items_historical** almacenará el historial de todos los items consumidos por un grupo desde la creación del mismo (ver Figura 24). Cada vez que se añada, modifique o se elimine un elemento de la tabla items, también se hará de la tabla items_historical. De esta manera, almacenará todos los atributos de un item indicados durante la explicación de la tabla items.

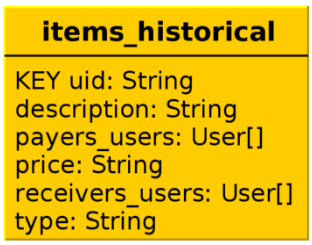

Figura 24: Tabla items_historical

Como se puede apreciar, las tablas items e items_historical referecian a la tabla users. Esto es debido a la asociación existente entre las tres tablas: un item puede tener varios usuarios dentro su atributo *payers_user* y varios usuarios dentro de su atributo *receivers_users*. Dicha asociación se puede observar en la Figura 25, en la que se muestran las distintas relaciones entre las diferentes tablas.

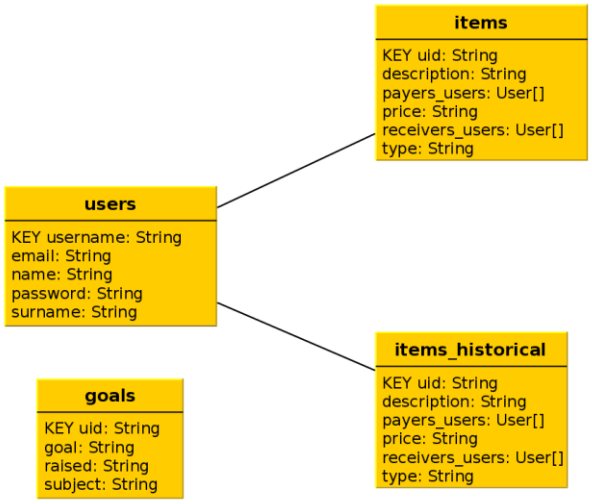

Figura 25: Relaciones entre tablas

Una vez conocidos todos los detalles de diseño del sistema, se pasa a comentar los principales aspectos relacionados con la implementación del mismo.

4. Implementación

Una vez realizadas las etapas anteriores descritas durante esta memoria en las que se establecían los criterios y bases fundamentales sobre las que se sostiene este TFM, se llevó a cabo la fase de implementación. Esto se efectúa por medio de la implementación en un prototipo que servirá para validar si se cumplen los requisitos establecidos en la etapa de análisis descritos anteriormente. En los siguientes apartados se muestran los problemas encontrados en esta fase, así como las diferentes alternativas y el razonamiento de la elección de la solución ante el problema descrito.

La implementación ha requerido el aprendizaje y conocimiento de las tecnologías Android, Java y Firebase. La primera fase, anterior a la implementación, consistió en recordar y reforzar los fundamentos y el funcionamiento de la tecnología Android y, por lo tanto, Java. Con respecto a Firebase, fue necesario realizar un estudio más profundo, para desarrollar la base de datos Realtime y las funciones de Autenticación. Para la implementación del código se ha utilizado el entorno de desarrollo Android Studio, el SDK de la plataforma Android para desarrolladores.

4.1 Desarrollo de la pantalla con pestañas, *Login* y *Registro*

En primer lugar se desarrolló la pantalla con pestañas que contendría a las demás pantallas de la aplicación. Cada una de estas pestañas contendría el *Fragment* con las demás pantallas explicadas durante la etapa de diseño.

Además, se implementó el menú de la aplicación haciendo uso del elemento *MaterialDrawer*[11].

A continuación, se implementaron las pantallas de *Login* y de *Registro* de un nuevo usuario. En ambos se comprobó que, por un lado, todos los campos estén rellenados y, por otro, que los datos introducidos sean correctos (por ejemplo que el email tenga el símbolo @). De no ser así, la aplicación avisa al usuario para que lo corrija.

4.2 Desarrollo de las pantallas principales

Posteriormente fueron desarrolladas las pantallas *Últimos ítems, Añadir ítem, Pagos y Metas económicas.*

Durante esta etapa, y hasta que se integró el servicio Firebase en la aplicación, todas las pruebas fueron realizadas a través de la creación de objetos locales dentro de cada una de las clases: *Item, Goal, Payment* y *User*.

La primera pantalla en desarrollar fue *Añadir ítem,* ya que a priori es la más sencilla. La mayor complejidad de dicha pantalla fue el campo "*Paid by*" y "*Who's it for?*" ya que ambos necesitan un componente *Spinner* que permita seleccionar varios usuarios al mismo tiempo. Después de un largo proceso de análisis e

investigación, se seleccionó una librería externa con la que poder construir un *Multispinner* con el que poder seleccionar varios usuarios el mismo tiempo[12]. Además de este elemento, fue necesario añadir una *TextView* en el que mostrar los usuarios seleccionados (ver Figura 26).

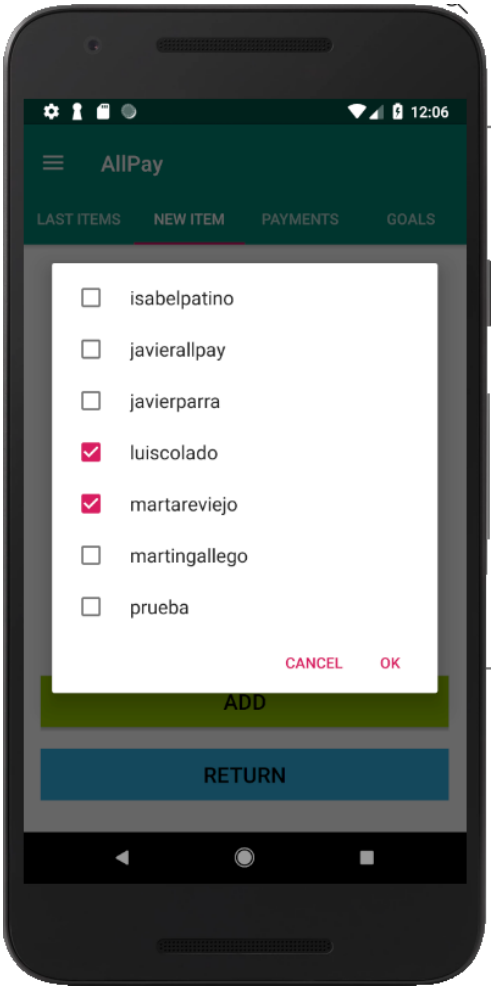

Figura 26: MultiSpinner desplegado en la pantalla "Añadir item"

A continuación se analizó qué componente utilizar para representar los últimos ítems, pagos y metas del grupo. Después de un profundo análisis, se decidió utilizar una *RecyclerView* para dicho fin.

Para representar cada uno de los ítems, pagos y metas de manera individual se utilizó el componente *Cardview* con el que mostrar de una manera precisa todos los datos del elemento que el usuario esté analizando. Para representar la información se ideó que cada una de las *Cardview* siguiera la estructura de la cabecera común de un correo electrónico, ayudando de esta manera a la usabilidad de la aplicación.

4.3 Integración de Firebase

Después de añadir las demás pantallas para añadir, editar y eliminar cada uno de los elementos presentes en la aplicación (*ítems, goals, users y payments)*, se analizaron las posibles opciones para almacenar todos los datos de la aplicación para que estos fueran accesible de manera remota por todos los usuarios del grupo. De entre todas las opciones se eligió Firebase debido a la amplia documentación disponible, rapidez y fácil despliegue.

Los mayores problemas a la hora de integrar Firebase en las demás pantallas de la aplicación fueron debidos a la metodología seguida para recuperar cada uno de los objetos de la base de datos. Específicamente, fue la tabla *items* la que mayor problema ocasionó ya que dicha tabla contiene referencias, en sus atributos *payers_users* y *receivers_users*, a la tabla *users*.

4.4 Implementación de quién paga la próxima y balance total

Dos de las funcionalidades más atractivas para el usuario de la aplicación es la función que esta provee para indicar quién debe pagar el próximo ítem y para mostrar el balance total de un grupo.

En primer lugar se implementó la funcionalidad para indicar quién debe pagar el próximo ítem, con el objetivo de que todos los usuarios gasten la misma cantidad de dinero en el grupo. Después de analizar como representar esta opción en la pantalla *Últimos ítems* con el menor grado de intrusismo posible, se decidió hacer uso del elemento *FloatingActionButton.* De este modo, después de que el usuario pulse sobre él, se calculará quién es el usuario que lleve más tiempo sin adquirir un ítem con el objetivo de mostrarlo (ver Figura 27).

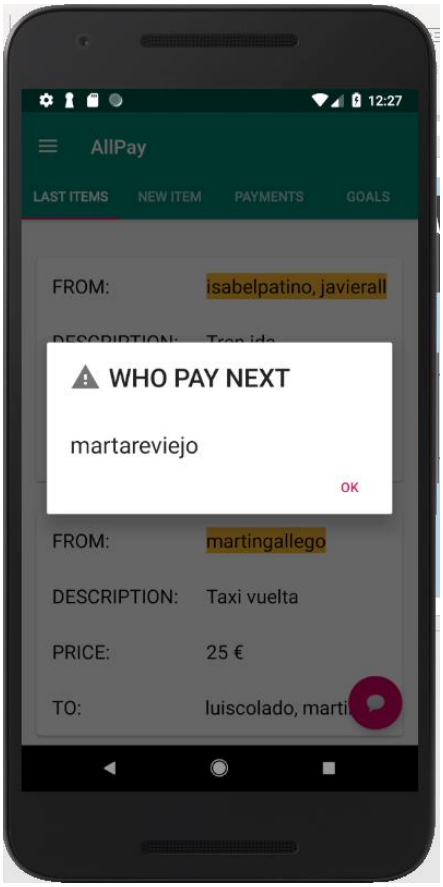

Figura 27: Mensaje para indicar quién paga el próximo item

Con respecto al balance total, la aplicación calcula la cantidad que cada usuario ha pagado y le resta la cantidad total de los bienes que ha consumido. De esta forma, cada usuario sabe cuanto le tienen que pagar o cuanto debe con el objetivo final de que todos los usuarios aporten la misma cantidad económica al grupo.

Además, a través de un *FloatingActionButton,* se incluyó la opción de poder indicar al usuario que todos los pagos se han realizado, es decir, que se han hecho los intercambios entre los usuarios necesarios para que todos los miembros del grupo hayan aportado la misma cantidad económica. De esta manera, además de *resetear* todos los pagos, también lo hacen todos los ítems que se hayan consumido en dicho grupo. Si un usuario quiere consultar el registro global de todos los ítems consumidos por un grupo, junto a todas sus características, lo puede hacer en la pantalla *Historial de ítems* del menú.

4.5 Pruebas y validación

Después de finalizar la aplicación fueron realizadas diversas pruebas a fin de comprobar la robustez de la misma. Gracias a estas pruebas se pudieron mejorar diversos aspectos de la aplicación final.

5. Conclusiones y Trabajo Futuro

Para finalizar, durante el desarrollo del presente capítulo se describirán los objetivos y contribuciones alcanzados durante la realización de este TFM. Además se indicarán las posibles líneas de trabajo futuro a realizar teniendo como base todo lo realizado durante este proyecto.

5.1 Conclusiones

En este TFM se ha desarrollado una aplicación con la que dotar a los usuarios de una herramienta con la cual ser capaz de analizar, controlar, registrar, prever y calcular cualquier gasto económico llevado a cabo por cualquier usuario dentro de un determinado grupo. Las conclusiones obtenidas tras su realización se enumeran a continuación.

1. Se han estudiado las diferentes arquitecturas y diseños con el objetivo de seleccionar e implementar la que mejor se adaptara a las características de la aplicación.

2. Se han analizado y desarrollado cada una de las funcionalidades de la aplicación, con el objetivo de su correcta implementación en la aplicación final.

3. Debido a las necesidades de la aplicación, se ha integrado Firebase con la aplicación, estudiando de este modo como utilizar dicho servicio y las posibilidades que ofrece.

4. Han sido adquiridas competencias complementarias que serán de gran utilidad para el desarrollo de otros proyectos de ingeniería:

    a. Capacidad de aplicar el proceso de ingeniería inversa a la hora de solucionar problemas, estudiar un nuevo método y analizar un componente.

    b. Seguir un desarrollo incremental, realizando poco a poco las tareas y aumentando paulatinamente su dificultad. Esto ha sido reflejado en la implementación de las diferentes versiones creadas como aprendizaje y experimentación antes de la confección del sistema final.

    c. Analizar las causas de un determinado problema surgido durante la realización del proyecto, desarrollando las diferentes hipótesis que solucionan éste para, a continuación, ir analizando e investigando cada una de ellas con el fin de solucionar el inconveniente encontrado de la manera más adecuada.

    d. Documentar todas las decisiones, avances y problemas dados paralelamente al desarrollo del proyecto.

e. Utilización de programas de control de versiones para tener el código almacenado y recuperable en todo momento. Esto ha permitido además la revisión de las modificaciones y evolución del proyecto.

5.2 Líneas de trabajo futuro

Para finalizar, se explican brevemente algunas de las líneas de trabajo futuro a desarrollar teniendo como base este TFM.

- **Posibilitar la gestión de varios grupos**. Una de las principales mejoras a introducir en esta aplicación es la posibilidad de tener varios grupos de usuarios en la aplicación. De esta manera, se cumpliría con uno de los más importantes objetivos opcionales del TFM.

- **Añadir el registro automático a la aplicación.** Otra de las principales mejoras es añadir el registro automático de cualquier usuario en la aplicación sin necesidad de añadirlo manualmente a Firebase.

- **Añadir sistema de roles**. De esta manera, según se ha comentado durante la etapa de diseño, sólo el administrador de un determinado grupo tendría la capacidad para borrar un ítem o meta que él no haya creado, añadir nuevos componentes al grupo o concluir un grupo.

- **Enviar notificaciones y emails**. Una buena funcionalidad sería, a través de Firebase, enviar notificaciones o emails a los usuarios para que nos olviden de pagar una deuda o actualizar una meta.

- **Posibilidad de añadir una imagen a un ítem**. Actualmente, no es posible añadir una imagen como característica de un ítem. Si se añadiera, sería una gran manera de representar gráficamente dicho ítem. De esta manera, el usuario asociaría un momento con el registro que aparece en la aplicación. Además, de este modo, incluir una descripción de dicho ítem sería totalmente opcional ya que con la imagen todos los usuarios sabrían de qué ítem se trata.

5. Glosario

TFM	Trabajo Fin de Máster
DSDM	Dynamic Systems Development Method
BBDD	Base de Datos
MVC	Modelo Vista Controlador
LOPD	Ley Orgánica de Protección de Datos

6. Bibliografía

[1] P. Abrahamsson, O. Salo, J. Ronkainen, y J.Warsta. *Agile software development methods. Review and analysis*. VTT Publications, 2002.

[2] What is DSDM? https://www.dsdm.org/, 2019. Última consulta: 13/03/2019.

[3] S. Messenger. DSDM Consortium: The Right Tools for the Job. Project Manager Today, (January/February 2015):19–20, Feb 2015.

[4] https://www.kutxabank.es/appatxas/#/landing. Última consulta: 08/04/2019

[5] https://www.ing.es/twyp/#. Última consulta: 08/04/2019

[6] http://blog.masmovil.es/no-mas-peleas-7-apps-para-compartir-gastos/. Última consulta: 08/04/2019

[7] http://www.larazon.es/tecnologia/apps/las-5-apps-que-haran-temblar-a-tus-amigos-morosos-CF15766004. Última consulta: 08/04/2019

[8] https://settleup.io/. Última consulta: 08/04/2019

[9] https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/. Última consulta: 08/04/2019

[10] https://deviceatlas.com/blog/android-v-ios-market-share. Última consulta: 08/04/2019

[11] https://github.com/mikepenz/MaterialDrawer . Última consulta: 04/06/2019

[12] https://github.com/thomashaertel/MultiSpinner. Última consulta: 04/06/2019

Anexos

ANEXO 1 – Prototipo inicial

Figura 28: Pantalla de Login

Figura 29: Pantalla de Registro

Figura 30: Pantalla Principal

Figura 31: Pantalla Añadir Producto o Servicio

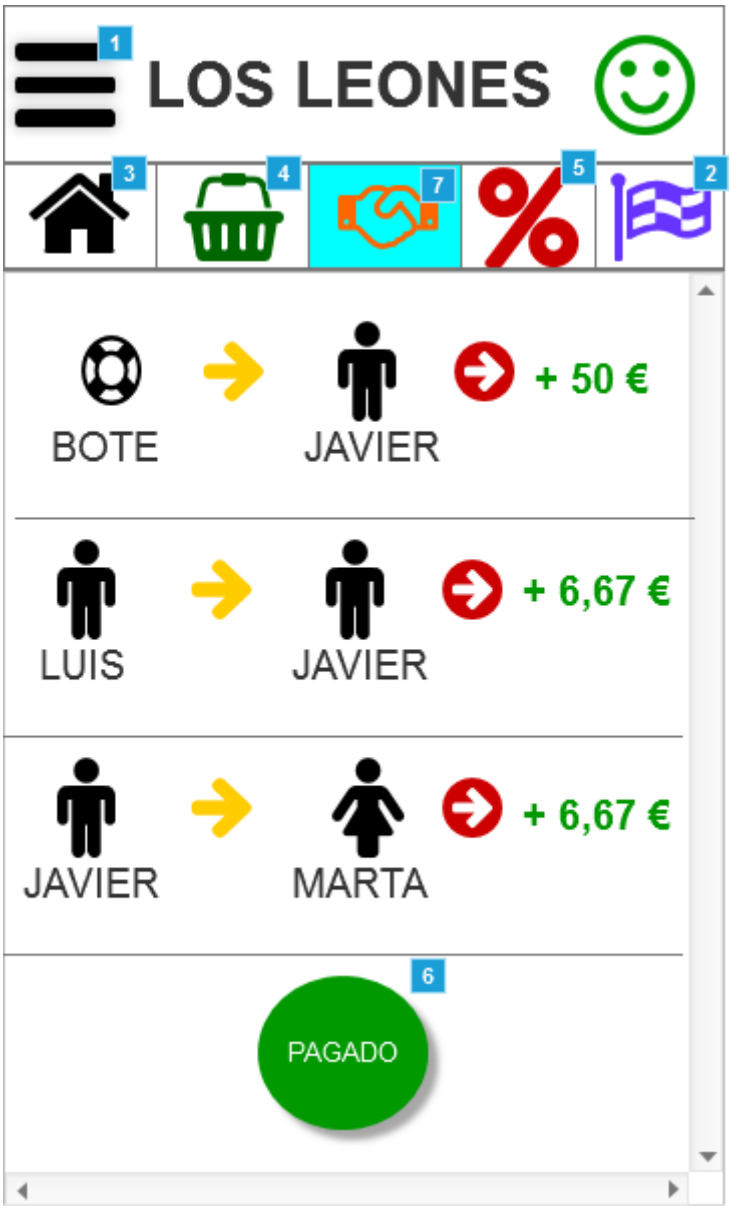

Figura 32: Pantalla Resumen

Figura 33: Pantalla Estádisticas del grupo

Figura 34: Pantalla Metas económicas

Figura 35: Pantalla para eliminar o añadir una meta económica

Figura 36: Pantalla Opciones

Figura 37: Pantalla Editar Perfil

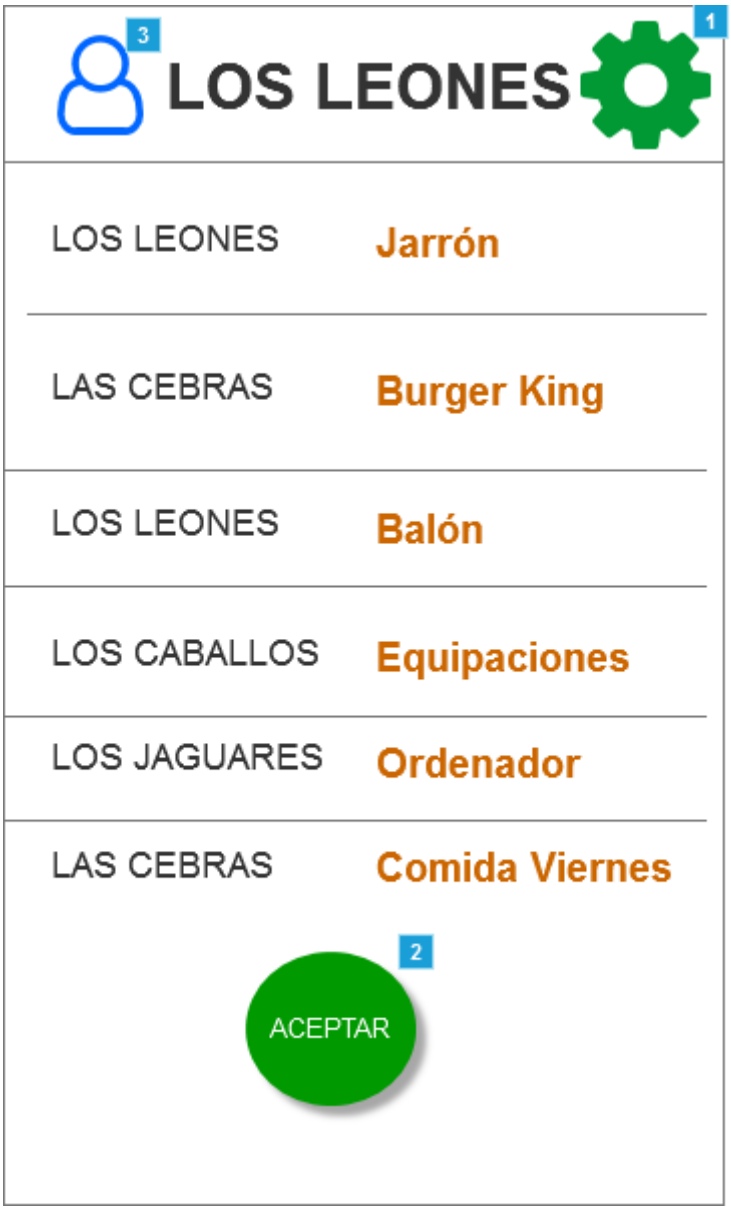

Figura 38: Pantalla Consultar Historial

Figura 39: Pantalla Crear Nuevo Grupo

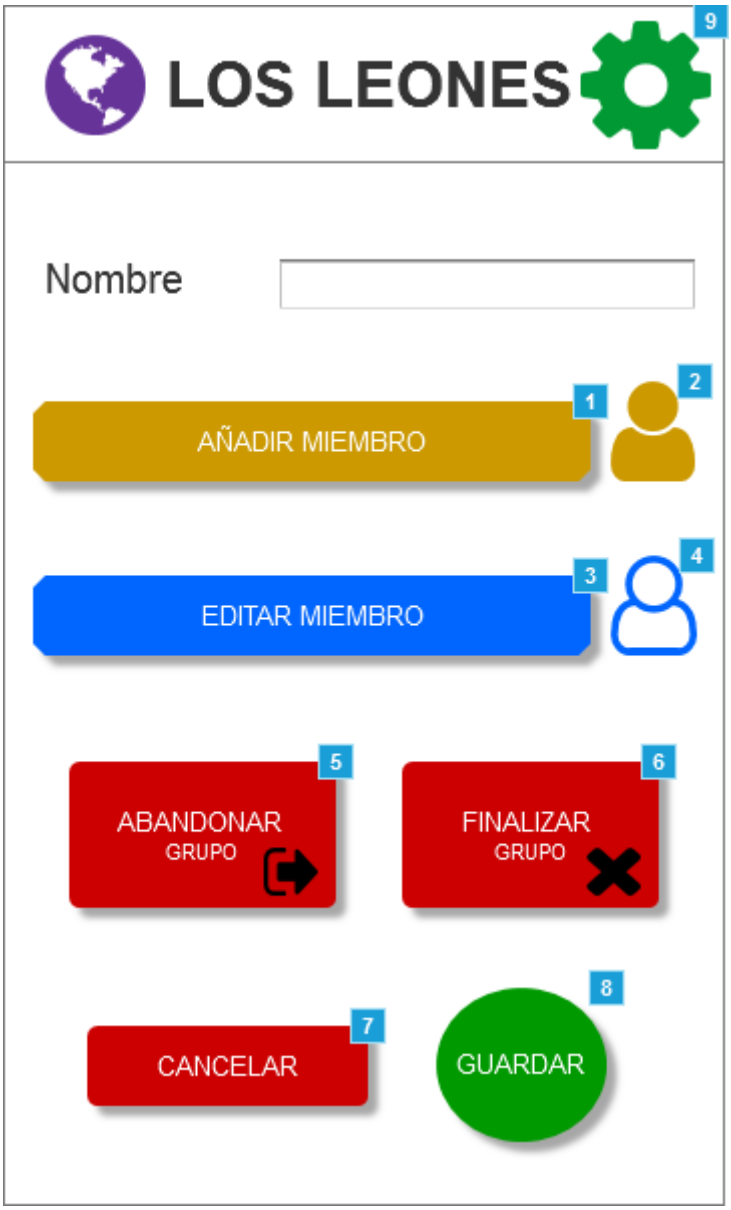

Figura 40: Pantalla Editar Grupo

Figura 41: Pantalla Editar miembro

Figura 42: Pantalla Pagar

Figura 43: Pantalla Cambiar Grupo

ANEXO 2 – Manual de Usuario

En el presente manual se explicarán todas las acciones a llevar a cabo para poder ejecutar la aplicación tanto en un emulador, tomando como ejemplo el incluido en Android Studio, como en un dispositivo real. Adicionalmente se explicará los pasos a seguir para poder analizar la base de datos desarrollada Firebase y cómo utilizar la aplicación.

Configuración del emulador y del dispositivo físico

Para poder utilizar la aplicación, es necesario tener como mínimo la versión Android 7.0 Nougat con el SDK 24.

Si se utiliza un emulador, para poder comunicarse con el servidor Firebase y así poder hacer uso de las diferentes funcionalidades y opciones, es necesario incluir el paquete de funcionalidades “Play Store” a la hora de crearlo. Todos los móviles físicos con Android incluyen dicho paquete de funcionalidades, por lo que no es necesario realizar ninguna configuración extra.

Analizar la base de datos creada en Firebase

Para analizar la base de datos desarrollada en Firebase y así comprobar, por ejemplo, como se van actualizando las diferentes tablas según las funcionalidades de la aplicación, es necesario realizar los siguientes pasos:

1. Abre el navegador y dirígete a https://firebase.google.com/?hl=es-419.

2. Haz clic en “Ir a la consola” y cuando te pidan usuario y contraseña introduce:

   email: javierparra.allpay@gmail.com
   contraseña: tfmallpay

3. Haz clic en el Proyecto “AllPay”.

4. En el menú de la izquierda, dentro de “Desarrolla”, selecciona “Database”.

5. Se abrirá la base de datos “allpay” que contiene todas las tablas desarrolladas en la aplicación.

Uso de la aplicación

Aunque la aplicación es intuitiva de utilizar, se presenta a continuación una breve guía acerca de su uso.

En primer lugar, introduce la siguiente combinación usuario-contraseña para poder realizar el login en la misma:

email: javierparra.allpay@gmail.com
contraseña: tfmallpay

Una vez dentro, se mostrá una ventana en la que hay tres pestañas. Por defecto se muestra la pantalla de *Últimos ítems*. Para moverte entre ellas simplemente es necesario arrastrar o hacer clic sobre ellas.

Tanto en esta pestaña como en las demás, si quieres ver en detalle o editar/borrar un elemento de la lista basta con hacer clic sobre él.

Para desplegar el menú, haz clic en las tres barras horizontales situadas al lado del nombre de la aplicación.