UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Diseño y desarrollo de una aplicación de armario de ropa virtual para Android

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Morales Martínez, Rubén

Tutor/a: Herrero Cucó, Carlos

CURSO ACADÉMICO: 2021/2022

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño y desarrollo de una aplicación de armario de ropa virtual

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Rubén Morales Martínez

Tutor: Carlos Herrero

Curso 2021-2022

Resumen

El TFG consiste en el Diseño y desarrollo de una aplicación armario virtual que guarde la información de todas las prendas de ropa del usuario. Esta aplicación Android también será capaz de combinar dichas prendas para crear un atuendo (en adelante *outfit*) de manera automática de entre toda la ropa del usuario, mediante un algoritmo que tendrá en cuenta varios parámetros como la formalidad del evento al que se planea acudir, el clima de ese día, o los colores de cada prenda. Además de esto, la aplicación será capaz de ayudar a los usuarios a organizar su ropa y perder menos tiempo diariamente gracias a su calendario de *outfits*, donde poder planificar la ropa que llevará los días posteriores. Se incluirán también, entre otras, funcionalidades de red social donde se puedan compartir los *outfits* creados mediante la aplicación con los contactos.

Las tecnologías por usar van a ser Android Studio como IDE, Java como lenguaje de programación, y FireBase como base de datos. Todo el proyecto será planificado en un desarrollo ágil, haciendo previamente al desarrollo un análisis de los requisitos de este, marcando fechas para los distintos MVP del proyecto y realizando una validación basada en pruebas con usuarios reales que podrán descargar la aplicación y dar un *feedback* usado para refinar la propuesta.

Palabras clave: Armario virtual; Atuendos virtuales; Ropa; Recomendación; Android; FireBase; Java

Abstract

This TFG consists of the design and development of a virtual closet app that saves the data of every garment of the user. This Android App Will be able to mix those garments to create an outfit automatically choosing from all the garment of the user using an algorithm that considering some parameters like formality of the event that the user is planning to go, the weather of that day, o the colors of the garment. Besides that, the app will be able to help the user to organize their clothes and lose less time daily thanks to the outfit calendar, where it's possible to plan the outfit that you we will wear in the following days. Will also be included other functionalities used in other social networks like sharing the outfits through the Contac application.

The technologies to use will be Android Studio as IDE, Java as programming Language and FireBase as database. All the project will be planned as an agile development, doing previously to the development a requirement analysis, planning dates to the MVPs of the project, and doing validation based on real user who could download the app and give feedback to improve the proposal.

Key words: Virtual Closet, Virtual outfits, Clothes, recommendation, Android, FireBase, Java.

Índices y glosarios

Tabla de contenidos

Tabla de Ilustraciones

1. Introducción

1.1 Motivación

La principal motivación de este proyecto viene dada por una necesidad personal, la necesidad de mejorar la organización de mi armario y las prendas de ropa que lo conforman. Esta necesidad personal se viene haciendo cada vez mayor con el paso del tiempo, pues en repetidas ocasiones me ha ocurrido el encontrar una prenda en lo más profundo del armario que no recordaba que tuviese. Y es que, al final, es difícil recordar todas las prendas que te compras a lo largo de tu vida.

Con este problema en mente, surgió la idea de crear una aplicación que ayudase a organizar el armario de una manera sencilla, guardando datos e imágenes de todas las prendas. Además, esta app podía ayudarme a no perder tanto tiempo en una cosa tan trivial como elegir la ropa que me iba a poner para salir a cenar con los amigos, ir a la universidad, o cualquier otra actividad que requiriese elegir que ropa llevar.

Con esta idea en mente, empecé a investigar sobre el mercado de aplicaciones similares para ver si encontraba alguna que cubriese mis necesidades. Está búsqueda fue insuficiente para mí, pues, aunque existían aplicaciones similares, no veía ninguna que me ofreciese todo lo que buscaba.

Tras no encontrar los resultados esperados, decidí que sería una buena idea intentar recrear esta aplicación que necesitaba para mi uso personal aprovechando los conocimientos adquiridos a lo largo de los últimos años.

1.2 Objetivos

El principal objetivo de este proyecto es el desarrollo de una aplicación para sistemas Android en la cual los usuarios sean capaces de organizar su armario de una manera más eficaz. Este objetivo se puede dividir en los siguientes subobjetivos:

Que la aplicación sea capaz de:

- Permitir subir prendas a los usuarios
- Permitir crear atuendos, en adelante *outfits*, a los usuarios
- Permitir guardar en favoritos los *outfits* creados por los usuarios
- Permitir enseñar todas las prendas subidas de los usuarios a modo de armario aplicando filtros

Como consecuencia de este objetivo principal, nace el objetivo de reducir la cantidad de tiempo que pasa una persona eligiendo que ropa ponerse cada día, y el objetivo de facilitar la información al usuario de que prendas ya posee y cuales podría necesitar, para evitar la compra de prendas de ropa innecesarias, ahorrando así un gasto económico y de materiales con impacto medioambiental.

1.3 Metodología

Para el desarrollo de esta aplicación se ha escogido un enfoque de metodología ágil [1]. Dada la estimación de tiempo necesario para la realización de todos los objetivos que planea cubrir esta aplicación y el tiempo disponible para la entrega del proyecto, se considera que la metodología ágil es la más adecuada pues se podrán abordar los distintos items en orden de importancia y se tendrá rápidamente una versión funcional, aunque no completa, de la aplicación.

Como es costumbre en una metodología ágil, el tiempo total disponible para el proyecto se dividirá en *Sprints*. Un *Sprint* en metodología ágil es un periodo de tiempo fijo en el cual, al principio de este, se marcan unas tareas a realizar , y cuando finalice el plazo establecido, se comprobará el resultado de dichas tareas, para ver si se han completado todo el trabajo estimado. Para este proyecto, cada *Sprint* dispone de una dedicación diaria al proyecto de aproximadamente 3 horas.

En este caso, se han elegido 5 *Sprints* de 14 días, con 2 días de descanso tras la finalización de cada *Sprint*, lo que sumaría una carga de trabajo de 42 horas por *Sprint* y una dedicación total de 210 horas de desarrollo de la aplicación.

Cuando se va a comenzar un Sprint, se eligen las tareas a desarrollar en el mismo, viendo cuales tareas se consideran más prioritarias dentro de las incluidas en el *Backlog*. Se puede observar un ejemplo en la Figura 1.

Lo primero que se ha de realizar con cada tarea es una especificación de requisitos y un prototipado de la interfaz. Cuando se implemente una tarea, se realizarán pruebas de aceptación que analizarán si el funcionamiento de la tarea es correcto y es tal y como se describió en la especificación de requisitos, como se puede ver en la Figura 2.

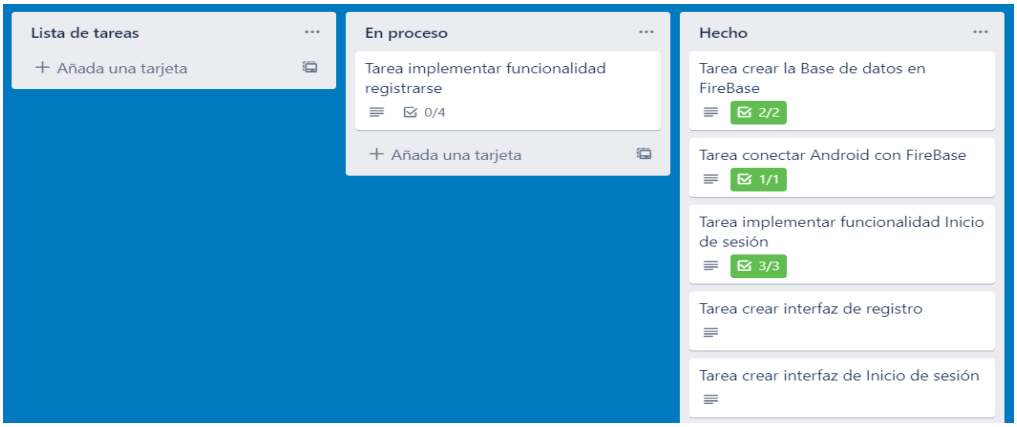

Figura 1. Ejemplo Sprint

Al finalizar cada *Sprint*, se verá el resultado obtenido y se comprobará si se han cumplido todos los objetivos que se tenían pensados a la hora de realizar el *Sprint*. Si no es así y quedan tarea por realizar, ya sea por imprevistos o por una mala estimación de tiempos, se pasarán como prioridad al siguiente *Sprint*.

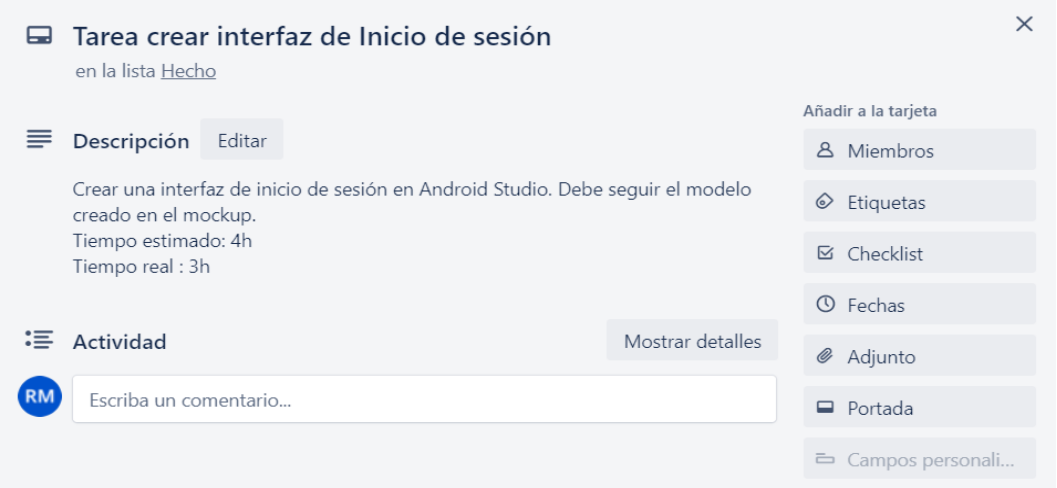

Figura 2. Tarea de un Sprint

Viendo más en detalle cómo se estima la planificación cada *Sprint* para la realización del proyecto:

- *Sprint* 0: Se centra en la evaluación de la idea de negocio propuesta para la aplicación, recaudando información útil para el desarrollo del proyecto, pero sin empezar con la programación de la aplicación.

- *Sprint* 1: Comienzo de la programación de la app. Los primeros items a abordar fueron la conexión de Android Studio [2] con FireBase [3], el inicio de sesión y el registro de usuarios para la aplicación.

- *Sprint* 2: Este *Sprint* se centra en la funcionalidad de creación de *outfits*, implementado el algoritmo que cree el *outfit* para el usuario, y la capacidad de subir fotos a la aplicación.

- *Sprint* 3: En este *Sprint* se realizan las funcionalidades de Favoritos y de Armario Virtual.

- Primer MVP de la aplicación: Tras la realización de los *Sprints* anteriores , se obtiene el primer MVP de la aplicación. Un MVP (Producto Mínimo Viable) es una versión muy básica del producto, con las funcionalidades esenciales, que permite probar la interacción de la aplicación con usuarios reales. Fin del TFG.

- *Sprint* 4: Creación del calendario de *outfits* y capacidad de compartir *outfits* por redes sociales.

- Segundo MVP de la aplicación

1.4 Estructura

Tras la introducción del proyecto en el apartado 1, se presenta un repaso al estado del arte en el apartado 2, donde de comenta la situación actual de la aplicación en el mercado y se compara con otras aplicaciones similares ya existentes. A continuación, en el apartado 3, se puede encontrar un análisis detalla del problema que resuelve la aplicación, teniendo en cuenta el estudio realizado a las aplicaciones similares realizado anteriormente. Tras esto, en el apartado 4, se encuentra el diseño de la solución, donde se elige que herramientas y tecnologías usar para satisfacer los objetivos marcados. En el apartado 5, se comenta el detalle el desarrollo de la aplicación, resaltando los problemas encontrados y como se solucionaron. Después, en el apartado 6, se puede observar el resultado de la implantación del proyecto, con imágenes y comentarios sobre el estado de la aplicación. En el apartado 7 se analiza una encuesta realizada a los usuarios que pudieron probar la aplicación , para obtener un *feedback* directo. Tras esto, en el apartado 8 se comentan las conclusiones del proyecto, y en el 9 se mencionan los trabajos futuros que se realizarán relacionados con este proyecto.

Además, los apartados 10 y 11 podemos encontrar las secciones de referencias y glosario, para terminar con anexo donde se comenta la relación de este proyecto con los objetivos de desarrollo sostenible (ODS).

2. Estado del arte

Para poner en contexto la necesidad que cubre este proyecto, es necesario saber cuáles son los problemas que intenta solucionar. El mundo textil y de la moda lleva mucho tiempo siendo importante para nuestra sociedad, y es que entre ropa y calzado en España se gastan más de 27 mil millones de euros anualmente [4]. Como es de esperar dadas las cifras que maneja la industria de la moda, un porcentaje de la ropa que se compra la gente es ropa que se perderá en los más profundo del armario y no se llegará a usar en la vida.

Varios son los factores que llevan a este gasto innecesario de dinero para comprar accesorios que resultan no sernos útiles, pero el mayor factor es el desconocimiento, es decir, la mayoría de los ciudadanos no podrían saber de memoria cuantas prendas de ropa tienen, o si la prenda que están viendo en la tienda para comprar no la necesitan porque tienen varias prendas parecidas. El *e-commerce* Percentil [5] realizo un estudio donde se descubrió que el 76% de los españoles ha comprado ropa que ha acabado no usando [6], y el 68% considera que tiene más ropa de la que necesita. Estos datos son excesivos y reflejan un malgasto no solo de dinero de los compradores, sino de materias primas, gastos de transporte y gastos de fabricación que no favorecen a los objetivos de sostenibilidad y cuidado del medio ambiente que debería tener nuestra sociedad.

Otro problema que busca solucionar este proyecto es la pérdida de tiempo que realizan las personas al tener que elegir que prenda deben ponerse para salir a trabajar, a una reunión informal, a una cita… Y es que, a parte de la pérdida de tiempo, es frecuente que algunas personas se encuentren perdidas en esta elección de prendas y sufran de estrés por no saber que deberían elegir.

Todos estos problemas son conocidos en nuestra sociedad, y es por eso por lo que existen aplicaciones similares a la propuesta en este TFG, pero todas con enfoques o funcionalidades distintas.

En los siguientes apartados se habla de la situación de mercado de la app a desarrollar, es decir, de aplicaciones con características similares que existan actualmente. Se compararán con la solución propuesta en este TFG, mostrando los problemas que existen en el mercado actual y las soluciones que aporta esta aplicación.

2.1 Aplicaciones similares

Simulando que un usuario desea una aplicación para organizar su ropa y crear *outfits* nuevos para no repetir siempre las mismas prendas, se ha buscado en la Play Store de Android aplicaciones ya existentes que podrían cubrir esta necesidad. Una vez realizada la búsqueda, se han seleccionado las apps más descargadas que se asemejen lo máximo posible a la versión final de la aplicación que se está desarrollando.

2.1.1 Combyne

Combyne [7] (Figura 3 y Figura 4) es una aplicación móvil disponible en Android y iOS enfocada en ser una red social donde compartir tus outfits con amigos que encuentras en la propia aplicación. Permite crear *outfits* con las prendas que se suben a la aplicación mediante un asistente. Además, en la feed, se pueden encontrar diseños de otras personas que estén ganando popularidad.

Figura 3. Interfaz de Combyne

Figura 4. Feed de Combyne

2.1.2 Smart Closet

Smart closet [8] (Figura 5 y Figura 6) es la aplicación de referencia que se tuvo en cuenta a la hora de pensar en el desarrollo de este TFG. Es un armario virtual donde se puede subir fotos de ropa y crear *collages* con ello, para posteriormente compartirlos con otros usuarios.

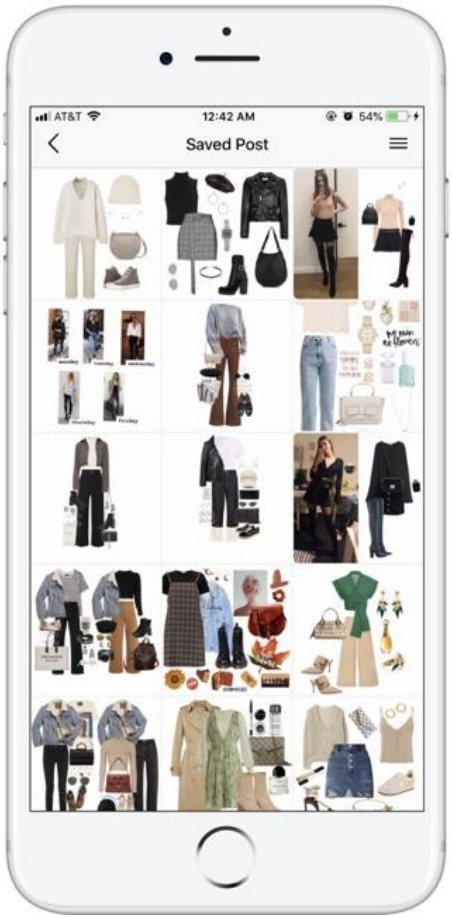

Figura 6. Interfaz de Smart Closet

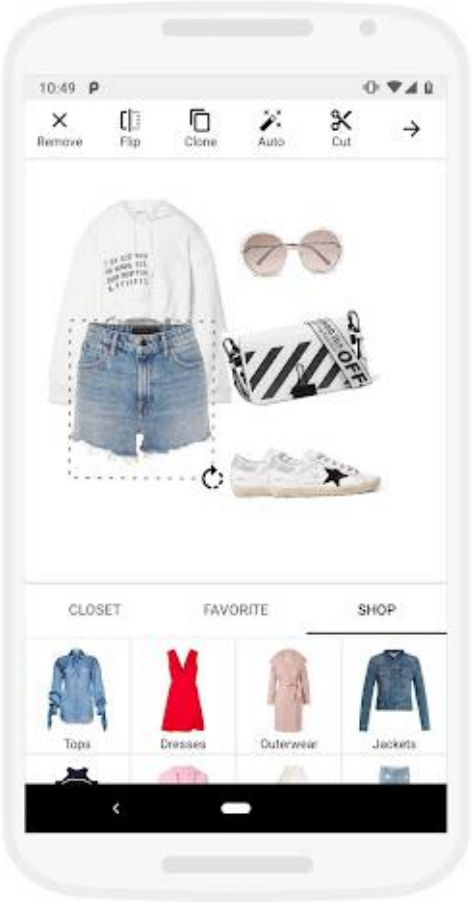

Figura 5. Función de tienda de Smart Closet

2.1.3 ACloset

Acloset [9] (Figura 8 y Figura 7) es un creador de *outfits* manual donde se puede subir fotos de las prendas, pero se centra en la recomendación de ropa en venta en distintas tiendas online. Puede planificar que te gustaría llevar cada día del mes y crear estilos, que son varias prendas agrupadas por ciertas características para poder crear más fácilmente los conjuntos.

Figura 7. Función descubrir tiendas de Acloset

Figura 8. Feed de ACloset

2.2 Crítica al estado del arte

Como se ha podido observar en el punto anterior, ya existen el mercado aplicaciones similares a la que se va a desarrollar, pero, además de que ninguna aplicación de las anteriores tiene una gran popularidad, todas se diferencian en algunas características de este proyecto. Para poder observar fácilmente las similitudes y diferencias de las aplicaciones anteriores con la aplicación desarrollada, llamada *OFM* (*Outfit Manager*), se ha creado esta Tabla comparación otras apps:

CARACTERÍSTICA	Smart Closet	Combyne	ACloset	*OFM*
Armario virtual	X	X	X	X
Crear conjuntos	X	X	X	X
Creador automático outfits			X	X
Reconocimiento colores				X
Filtros por características	X	X	X	X
Recomendar tiendas online	X	X	X	
Calendario outfits			X	X
Sistema de seguidores	X	X	X	
Guardar favoritos	X	X		X
Publicar outfits	X	X	X	X

Figura 9. Tabla comparación otras apps

La mayoría de las apps cubren la necesidad de armario virtual a la perfección, permitiendo subir fotos de las prendas de los usuarios para una mayor organización de la ropa y crear conjuntos con dicha ropa, pero ninguna se centra en la creación de un algoritmo que cree *outfits* con sentido estético y que puedan de verdad ayudar a los usuarios a la hora de elegir sus prendas.

También se puede observar que las funciones propias de una red social se cubren perfectamente en las aplicaciones ya existentes, como serian la capacidad de compartir los *outfits*, un sistema de seguidores o de amigos con los que interactuar, poder explorar o ver *outfits* de otros usuarios.

Por otra parte, dada la reciente explosión de la compra de venta online, la mayoría de las apps también cubren la faceta de recomendar ropa de tiendas online conocidas por el usuario.

Poniendo el foco ahora en otros aspectos a parte de las funcionalidades propias de cada aplicación existente, también se ha notado que las aplicaciones analizadas tienen unas interfaces de usuario complicadas, con mucha sobrecarga de botones, panales u opciones, que pueden llegar a confundir a los usuarios nuevos, no permitiéndoles sentirse cómodos navegando en la aplicación, lo que podría provocar una tasa de retención de usuarios nuevos bastante baja.

En resumen, si se analizan las aplicaciones que compiten con la propuesta de este proyecto, se puede ver que sería fácil que *OFM* se desmarcase de la competencia centrándose en el apartado de la creación automática de conjuntos, ya que es la característica que no se cubre en las demás aplicaciones analizadas.

2.3 Propuesta

A la vista del análisis concluido en los puntos anteriores sobre el estado del arte, se ha decidido que *OFM* se centra las siguientes características:

- Interfaz sencilla: La aplicación debe tener una interfaz muy sencilla para los usuarios, con un tiempo de adaptación muy rápido, y que facilite el uso de todas sus funcionalidades.

- Inicio sesión: La aplicación debe tener una funcionalidad que permita crear cuentas a los usuarios para permitirles guardar toda su información de una manera adecuada en la base de datos, y poder recuperarla posteriormente.

- Subir prendas: La aplicación debe permitir subir fotos de las prendas de los usuarios junto a una serie de parámetros que.

- Creación automática *outfits*: La aplicación debe permitir crear *outfits* de manera automática basándose en las prendas subidas por el usuario y en los filtros aportados al realizar la creación del *outfit*.

- Guardar outfits en favoritos: La aplicación debe permitir guardar los *outfits* creados anteriormente, para que el usuario puede volver a ver el *outfit* en el momento en el que lo desee.

3. Análisis del problema

Después de analizar el estado del arte y detallar cual es la solución propuesta para este proyecto, se ha hecho un análisis más exacto del problema haciendo uso de técnicas que nos ayuden a comprender cual es el problema por resolver.

3.1 Especificación de requisitos

La especificación de requisitos es una técnica que nos ayuda a visualizar cual el comportamiento completo esperado de la app al finalizar su desarrollo. Se va a dividir los requisitos en 2 categorías:

- **Requisitos Funcionales**: Son aquellos que describen los servicios que debe proporcionar el sistema.
  - El sistema debe implementar un sistema de autentificación de usuarios con inicio de sesión y registro mediante correo electrónico y contraseña.
  - El sistema debe permitir subir fotos de las prendas de los usuarios añadiéndoles una serie de parámetros como nombre, formalidad o temperatura ideal.
  - El sistema debe ser capaz mediante un algoritmo de crear *outfits* automáticamente ajustando a los parámetros dados por el usuario.
  - El sistema debe implementar una funcionalidad de Favoritos donde el usuario guarde sus *outfits* aleatorios para poder visualizarlos en cualquier momento.
  - El sistema debe permitir ver todas las prendas que ha subido el usuario a la app, pudiendo aplicarle filtros a la búsqueda.
  - El sistema debe implementar un calendario donde añadir los *outfits* para cada día de la semana
  - El sistema debe permitir compartir los *outfits* creados

- **Requisitos no Funcionales**: Estos requisitos se refieren a los requisitos que no tienen relación con la funcionalidad propia del sistema.
  - Se debe garantizar la seguridad del sistema, para proteger los datos de los usuarios.
  - El tiempo de carga al abrir la aplicación debe ser inferior a 3 segundos.
  - El tiempo de carga al crear un *outfit* aleatorio debe ser inferior a 1 segundo.
  - El tiempo de espera al subir una imagen a la base de datos debe ser inferior a 2 segundos.
  - La aplicación debe estar disponible para todos los dispositivos con un Sistema Operativo mayor o igual a Android Nougat.
  - La aplicación debe tener una disponibilidad mayor del 99%

3.2 Mapa de características

La Figura 10 muestra el mapa de características que se ha realizado usando la técnica de MoSCoW. Esta técnica ayuda a determinar como de prioritaria es una característica basándose en el valor comercial que tendría. Esta priorización de las características se utiliza para poder dar un orden lógico a las funcionalidades, empezando a implementar las más prioritarias, es decir, las del apartado MUST, y desplazándose en orden descendente conforme avanza el desarrollo del proyecto.

El objetivo de este TFG era realizar todas las características de MUST, y una del SHOULD, lo que resultaría en nuestro 1º MVP. Después, una vez finalizado el TFG, el proyecto seguiría desarrollando con las características restantes del SHOULD más las COULD.

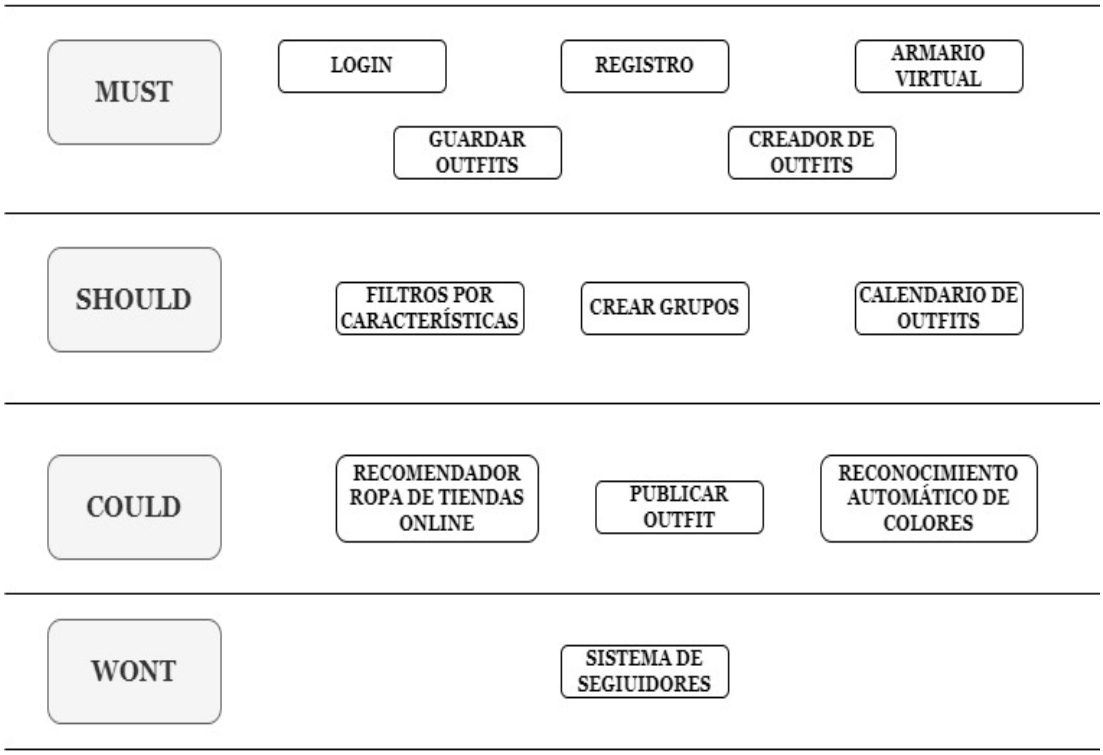

Figura 10. Mapa de características

3.3 Modelo de dominio

Un modelo de dominio es una representación de las clases conceptuales del mundo real. Se ha elaborado este documento porque es muy útil a la hora de dar inspiración para el diseño de software.

Para facilitar la comprensión del problema, también se ha elaborado este modelo de dominio, como se ve en la Figura 11, describiendo las entidades del proyecto y como se relacionan entre ellas.

Como se puede observar, la clase que más se relaciona con las demás es la clase Prendas, pues cada prenda se relaciona con todas sus características, y además las clases Prenda Superior, Prenda Inferior y Calzado heredan de la clase padre Prendas. También se puede ver que el armario está compuesto por prendas, y cada usuario tiene un único armario.

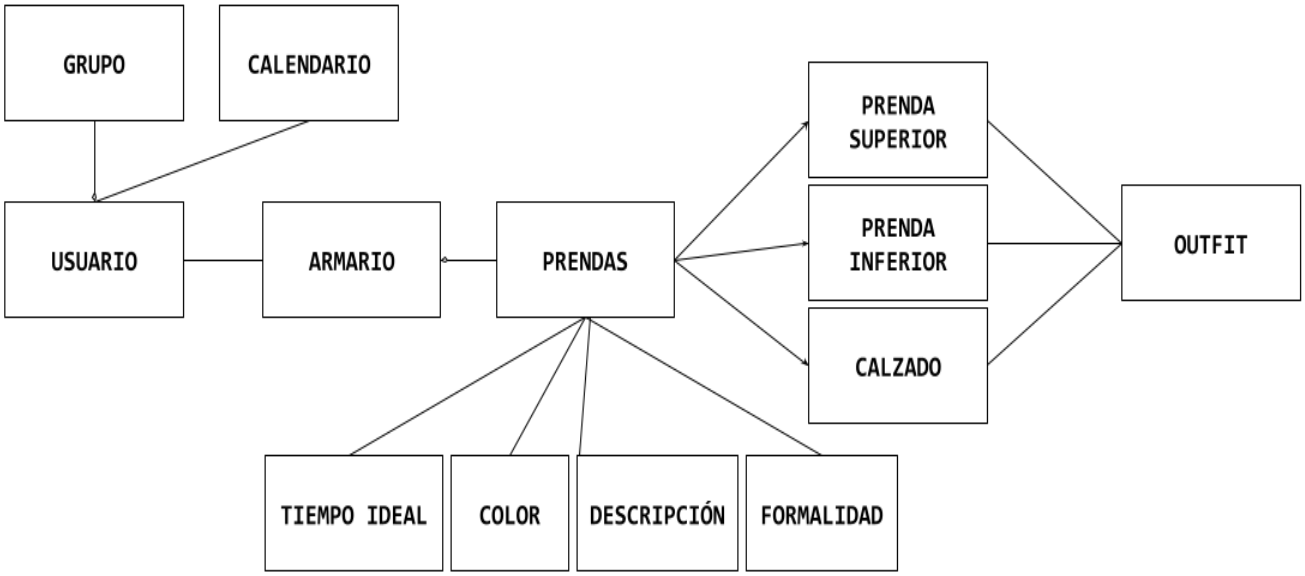

Figura 11. Modelo de Dominio

4. Diseño de la solución

En este apartado se analizan todos los datos obtenidos con anterioridad, y se decide cuál va a ser el diseño elegido para satisfacer los objetivos y requisitos comentados previamente.

4.1 Arquitectura del sistema

La aplicación desarrollada es una aplicación móvil, puesto que la comodidad y rapidez que proporciona una aplicación para Smartphone es muy adecuada con las necesidades del proyecto, ya que el armario virtual debería estar disponible rápidamente en unos pocos clics.

Se ha decidido que, dentro del mundo de desarrollo de aplicaciones móviles, este proyecto se va a orientar a ser una aplicación Android, pues es el SO más utilizado en todo el mundo, y proporciona muchas facilidades para el desarrollo de aplicaciones.

La aplicación para desarrollar se puede dividir claramente en 2 grandes bloques.

4.1.1 Android Studio

Figura 12. Logo Android Studio

Para desarrollar la aplicación se usará Android Studio, pues es el IDE perfecto para el desarrollo en Android, ya que al ser el IDE oficial de desarrollo Android, facilita muchas herramientas para de apoyo al proyecto, como un simulador de dispositivos Android donde probar la app en desarrollo, o un generador de *APKs* para facilitar la descarga de la app a otros usuarios. En el IDE se realizará tanto es diseño de las interfaces mediante los *layouts XML* como las funcionalidades mediante código Java.

4.1.2 FireBase

Se ha decidido que el proyecto Android se basara en FireBase como estructura principal. FireBase es una plataforma en la nube de Google que ayuda y facilita la creación de proyecto móviles. En los siguientes apartados se explicará detalladamente todas las funcionalidades que se usaran de FireBase, pero cabe destacar que esta tecnología integra una gran variedad de herramientas, que agrupan todo lo necesario para el desarrollo de una aplicación Android en una sola tecnología, como se ve en la Figura 14.

Figura 13. Logo FireBase

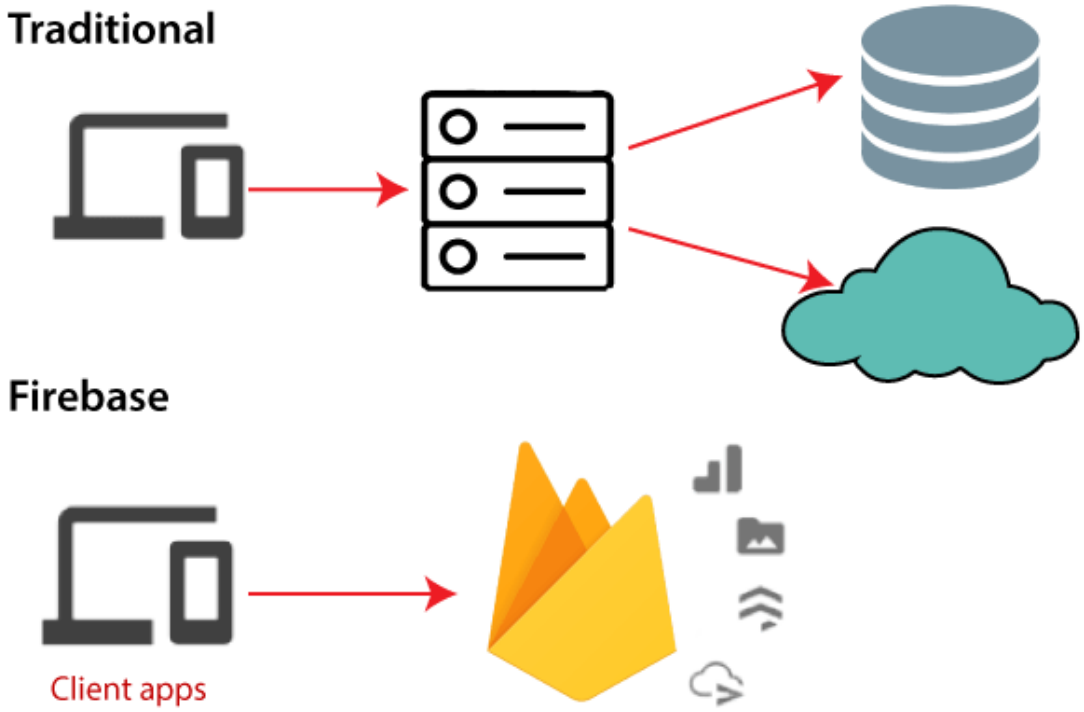

Figura 14. Arquitectura Firebase vs Tradicional

Gracias a FireBase, no es necesario que se use una Base de datos externa como podría ser SQL, ni un sistema de almacenamiento de imágenes, ya que todo esto se nos facilitas al añadir FireBase al proyecto.

4.2 Diseño detallado

En este apartado vamos a detallar las clases y la estructura del proyecto dentro de Android Studio y dentro de FireBase.

4.2.1 Android Studio

Al crear un nuevo proyecto de Android Studio, automáticamente se crean los archivos y directorios necesarios para el proyecto (Figura 16) los más importantes son:

- Archivo Android Manifest: Es un archivo XML que contiene información sobre el producto Android que se está desarrollando, como la versión del *SDK* usada o los permisos que requeridos por la aplicación para su uso.
- Drawable: En esta carpeta se guardarán las imágenes necesarias para el proyecto, como son el logo de la aplicación en distintos formatos o cualquier imagen la cual se vaya a usar en una interfaz.
- Layouts: Como se puede observar en la Figura 15 dentro de la carpeta *Layouts* se encuentran los archivos XML que se usan como interfaz del proyecto. Estos archivos serán enlazados por las clases nativas Java para ser mostrados en la aplicación.

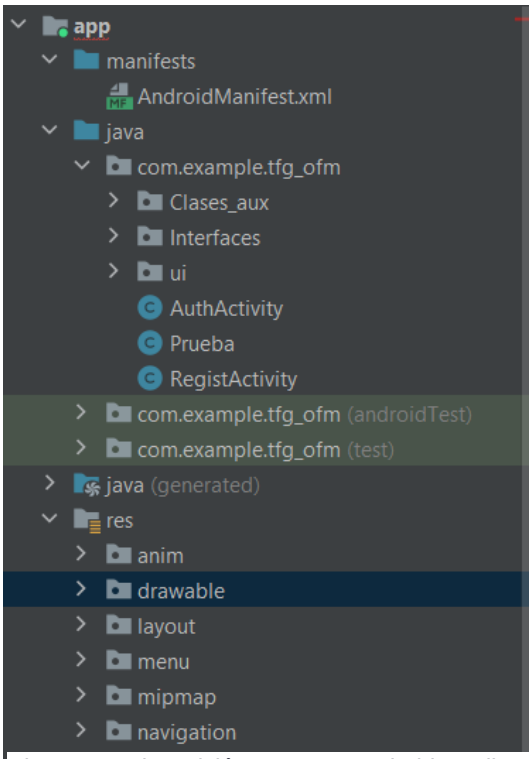

Figura 16. Disposición carpetas Android Studio

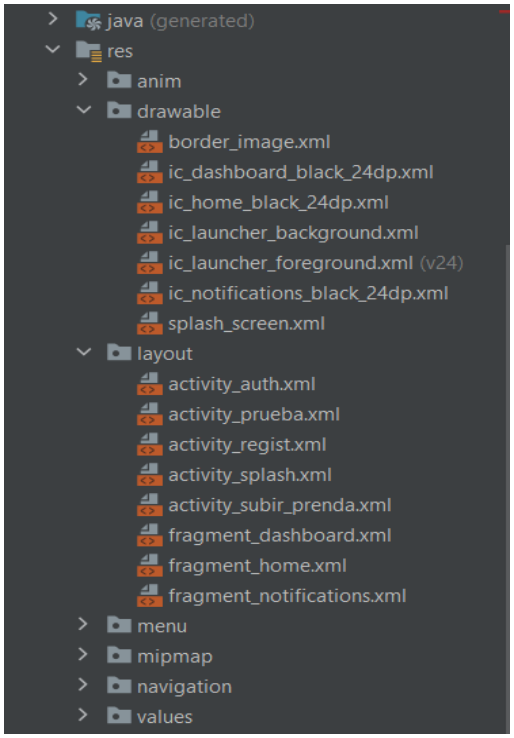

Figura 15. Disposición carpetas Android Studio

Las clases nativas Java se encuentran en la carpeta autogenerada Java. Dentro de la carpeta, se ha optado por crear la siguiente distribución:

-**Clases auxiliares**: En esta carpeta se incluyen tanto los *DTOs* usados en las consultas a la BBDD como las clases necesarias para realizar alguna funcionalidad especifica proveniente de una biblioteca Java.

-**Interfaces**: En esta carpeta se guardan las interfaces usadas en la aplicación.

-**UI**: Esta es la carpeta principal del proyecto, donde se guardan las clases que forman nuestra actividad principal. Esta actividad principal es una implementación de una actividad que ofrece Android Studio la cual se adapta a la perfección con las necesidades de la aplicación. Esta actividad es la 'Botton Navigation Activity', la cual nos proporciona una barra de navegación inferior para ir moviéndose entre las distintas pantallas en unos pocos clics. Esta actividad resultaba perfecta en esta aplicación ya que se evita su principal problema, y es que no es recomendable crear una barra de navegación con más de 4 interfaces principales, pues sobrecarga con demasiados botones la pantalla. Este problema no interfiere en los planes que se tiene del diseño de la aplicación, pues principalmente iba a haber 4 pantallas por la cuales es usuario podría ir deslizándose. Las pantallas recién mencionadas son:

- **Crear tu *outfit***: Primera pantalla que ve el usuario tras el registro. Permite crear *outfits* con las prendas subidas del usuario teniendo en cuenta los parámetros que se escojan. Además, permite añadir los *outfits* a Favoritos, donde podrá verlos posteriormente en la pantalla homónima.

- **Subir Prenda**: Pantalla donde el usuario sube sus prendas tras realizar las fotografías a su ropa.

- **Favoritos**: Pantalla donde se pueden ver todos los *outfits* marcados como favoritos, y eliminar los que ya no se desean en esta categoría. En una versión posterior de la aplicación, desde aquí se podrá compartir los *outfits* que el usuario desee.

- **Armario**: Pantalla donde el usuario podrá ver toda su ropa subida en la app, aplicando filtros para una búsqueda más sencilla. Posteriormente, esta pantalla tendrá un botón flotante donde se abrirá en modo de *popup* un calendario con la ropa prevista para cada día.

Además de esta actividad principal, existen la clase de Registro y la de Autentificación, previas al acceso a la pantalla principal, donde el usuario puede registrarse con una cuenta existente o crear una nueva cuenta.

4.2.2 FireBase

FireBase tiene una forma de almacenar los datos distinta a otro tipo de Base de datos como podría ser SQL. En Firebase la información no se guarda en tablas que se relacionan entre sí, si no que se guarda en Colecciones. A su vez, dentro de cada Colección, se almacenan Documentos, que incluyen los campos donde se guarda la información. Para aclarar el funcionamiento de FireBase, se ha realizado este esquema, como se puede ver la Figura 17, que relaciona la forma en la que se guardan los datos en FireBase con la forma en la que se guardan los datos en una base de datos relacional.

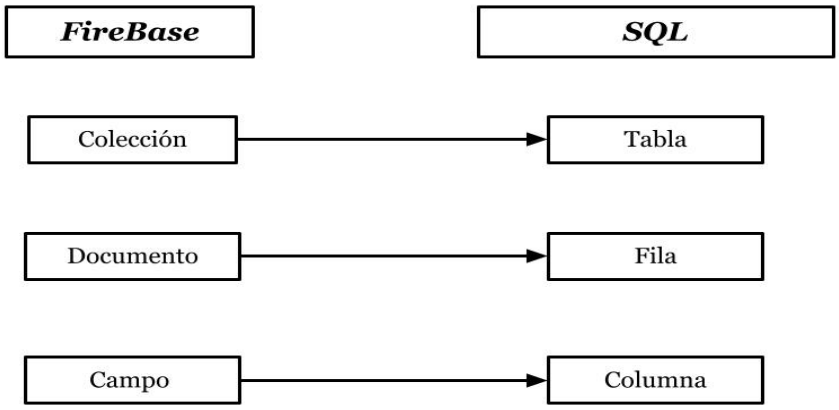

Figura 17. Organización interna FireBase

Una vez visto como se organiza una base de datos de FireBase, se puede ver como se ha realizado para este proyecto (Figura 18):

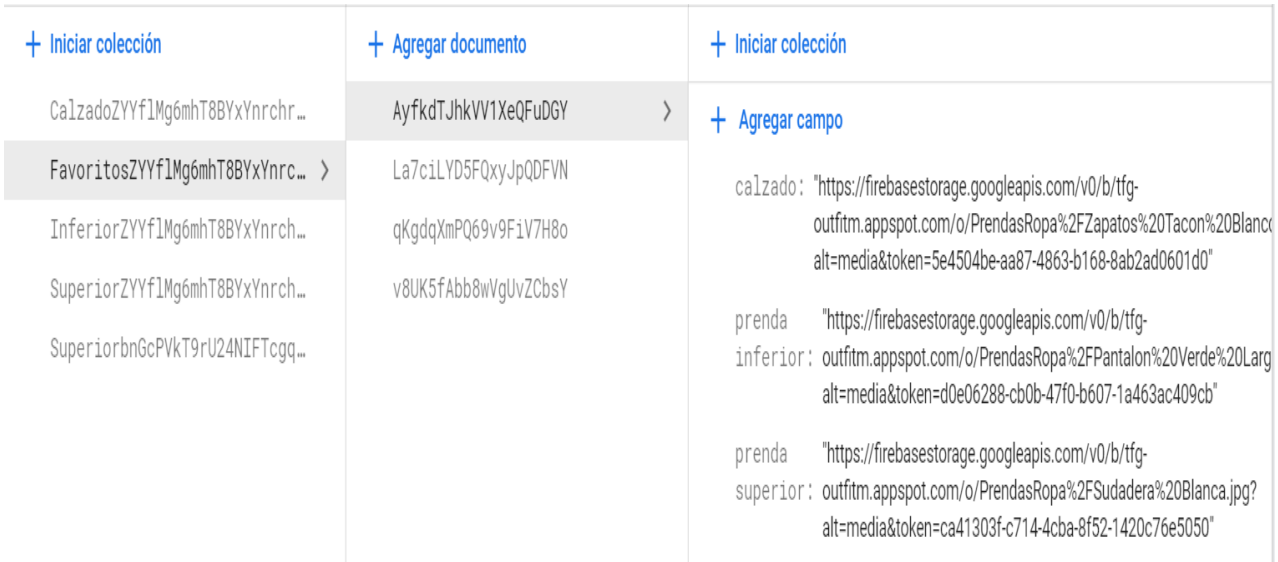

Figura 18. Ejemplo organización en FireBase

Como se puede observar, se ha elegido crear 4 colecciones para cada usuario, una para cada tipo de prenda y otra para los favoritos. A estas colecciones se les identifica con el String que indica que tipo de colección es más una serie de caracteres que son el ID del usuario, autogenerado por la función Auth de FireBase.

Se ha elegido este formato puesto que al ser el ID autogenerado único, la consulta a la base de datos se simplificada ampliamente al separar de esta forma a los usuarios. Esto se debe a que las consultas FireBase se realizan a cada documento, teniendo que indicarle el nombre en el que buscar, y filtrando posteriormente por documentos y/o campos.

4.3 Tecnología utilizada

-Android Studio: Como se ha comentado con anterioridad, Android Studio es el IDE elegido para la realización del proyecto. Este IDE desarrollado por Google fue creado por Google en 2014 y se convirtió en el IDE oficial para el desarrollo de aplicaciones Android. Se ha elegido la versión Android Studio Artic Fox, y, como lenguaje de programación dentro de Android Studio, se ha elegido Java. El nivel de API mínimo para la aplicación es de 24, lo que equivale a una versión de Android de 7.0. Se ha elegido esta versión pues pese a ser lo suficientemente nueva como para tener todas las tecnologías necesarias disponibles, con esta versión la aplicación estaría disponible para el 89% de los dispositivos Android del mundo.

-**FireBase**: Detallando más las funcionalidades que ofrece FireBase, se va a enumerar todas las funcionalidades que se han agregado al proyecto gracias a FireBase.

º **Base de datos**: Firebase ofrece dos posibles herramientas de base de datos para el desarrollo en aplicaciones Android, como son Cloud Firestore y Realtime Database. Vamos a compararlas y ver cual se adapta mejor al proyecto:

-**Realtime Database**: Es la base de datos original de Firestore. Guarda los datos en forma de JSON y los muestra en forma de árbol. Solo soporta filtrar y ordenar por un campo en cada Query. Tiene una fiabilidad y disponibilidad demostrada pues lleva varios años en uso en grandes empresas. El precio de usar estos servicios escala por tamaño y ancho de banda.

- **Cloud Firestore**: Es la base de datos más reciente de Google, basándose en Realtime, pero con un modelo de datos nuevo. La estructura visible cambia, pues pasa de ser una estructura de árbol a tener una consola más sencilla donde ver claramente todos los datos. Permite las consultas compuestas y puede combinar filtros de ordenamiento, al contrario de Realtime. Ofrece una gran escalabilidad de forma automática. El precio es por operación de escritura lectura y eliminado.

Una vez vistas las 2 posibles propuestas, se ha tomado la decisión de elegir Cloud Firestore como BBDD de la aplicación, pues tiene un mayor rendimiento hoy en día, realizando consultas de manera muy rápida y eficaz. Además, el coste de este servicio es nulo hasta que haya una enorme cantidad de operaciones diarias.

º **Authentication**: Esta herramienta de Firebase sirve para el control de usuarios, pues proporciona servicios *Backend*, *SDK* fáciles de usar y bibliotecas de IU para autentificar a los usuarios de la app. Admite tanto la autentificación mediante contraseñas, como mediante entidades federadas populares como Google o Facebook.

º **Cloud Storage:** Es un servicio de almacenamiento de objetos muy potente y simple. Permite crear carpetas donde guardas topo tipo de archivos multimedia. En este caso, se usará para guardar todas las prendas que suban los usuarios a la aplicación.

º **Analytics:** Es una solución de medición de apps, que proporciona estadísticas sobre el uso de las apps y el grado de participación de los usuarios. Como todo lo mencionado anteriormente, está disponible sin cargo, y permite generar gran cantidad de informes para entender cómo se comportan los usuarios dentro de la app, como se puede ver en las ilustraciones Figura 19 y Figura 20.

Figura 19. Ejemplo Analytics I

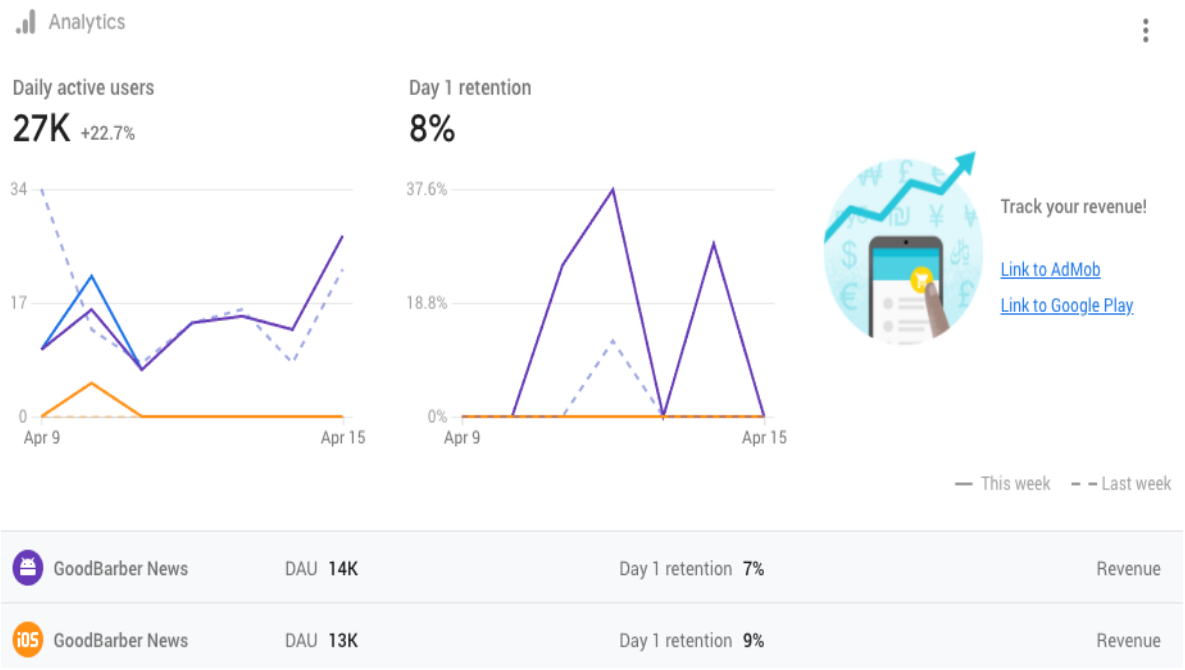

Figura 20. Ejemplo Analytics II

Figura 21. Logo Java

-**Java**: Como idioma de programación se ha optado por Java, pues una de las principales opciones que se nos ofrece al crear un proyecto Android Studio con FireBase. La otra alternativa a Java es Kotlin, pero dado que no hay experiencia previa en desarrollo de aplicaciones Kotlin y no ofrece ninguna ventaja respecto a Java, se ha elegido este último.

Figura 22. Logo Glide

-**Glide** [10]: Glide es una librería externa que soluciona uno de los principales problemas de Android Studio, que es no permitir cargar imágenes desde una URL. Dado que en FireBase Storage se nos facilita una URL por cada imagen que suben los usuarios, Glide es una solución perfecta para poder mostrar esas imágenes en la aplicación. Además, implementa varios métodos que permiten personalizar la forma en la cual se van a cargar las imágenes.

-**Github** [11]: Como sistema de control de versiones, se ha elegido GitHub. Gracias a Github se pueden crear distintas ramas en el proyecto para poder hacer grandes cambios en el código sin tener la preocupación de perder ningún tipo de información. Además, se puede integrar fácilmente a la interfaz de Android Studio, lo que facilita su uso.

Figura 23. Logo Github

-**Moqups** [12]: Moqups ha sido la herramienta elegida para realizar los prototipos de cada interfaz de la aplicación. Se ha elegido esta aplicación porque es gratuita y ofrece una gran cantidad de diseños e imágenes para crear los *mockups* más realistas posibles.

Figura 24. Logo Moqups

-**Trello** [13]: Trello es una herramienta que sirve para gestionar proyectos y aumentar la productividad. En este proyecto se ha utilizado para manejar cada *Sprint*, usando tarjetas para representar a cada ítem.

Figura 25. Logo Trello

5. Desarrollo de solución propuesta

En este apartado se comenta como se ha desarrollado el proyecto, viendo que tareas se realizaron en cada momento, con sus dificultades, soluciones y decisiones tomadas para llevar a cabo la solución propuesta.

Como este proyecto se ha realizado en mediante una metodología ágil, vamos a ver detalladamente como se ha desarrollado cada *Sprint*.

5.1 *Sprint* 0

Durante este *Sprint* del proyecto, se dedicó la totalidad del tiempo a investigar sobre la viabilidad de la aplicación a desarrollar. Se buscó información sobre apps existentes que pudiesen ser parecidas a la solución propuesta, descargaron y probaron esas aplicaciones para ver que funcionalidades ya planeadas para la aplicación implementaban, o que funcionalidades no planeadas podrían también ser útiles en este proyecto. Un ejemplo de esto último fue la funcionalidad de calendario de *outfits*, pues no estaba planeada en la primera idea de la aplicación, pero al ser una idea que encajaba bien con la solución que se estaba implementando, se decidió que esta funcionalidad debería estar en una versión futura de la aplicación.

Durante este *Sprint* también se realizaron los documentos vistos con anterioridad para tener una idea más clara de la solución, como son el mapa de características o el modelo de dominio. También durante este *Sprint* se marcaron los objetivos y las requisitos funcionales y no funcionales de la aplicación.

5.2 *Sprint* 1

En este *Sprint* se comenzó con el desarrollo del proyecto en Android Studio. Lo primero que se realizó fue la conexión de Android Studio con la herramienta FireBase. Esta conexión fue muy sencilla siguiendo un manual que proporciona la propia página web de FireBase.

Una vez se realizó la conexión, las primeras tareas de programación de funcionalidades que se desarrollaron fueron las de Inicio de Sesión y Registro de usuarios. Para realizar estas funcionalidades, muy similares entre sí, primero se realizó el diseño de la interfaz gráfica de cada mediante un archivo XML. Después, se realizó programación de ambas funcionalidades con la ayuda de la herramienta Authentication de FireBase mencionada anteriormente. El desarrollo fue sencillo pues con esta herramienta solo fue necesario instanciar un objeto *FireBaseAuth,* con el cual ya se obtienen los métodos necesarios para realizar las funcionalidades de inicio de sesión y de registro.

El primer problema que se encontró durante el desarrollo del proyecto vino justo después de la realización de la funcionalidad anterior, pues tocaba realizar la fase de pruebas de aceptación, para comprobar el correcto funcionamiento de la aplicación. La primera prueba fue satisfactoria, pues todo parecía funcionar correctamente, pero tras cambiar de Smartphone para la realización de más pruebas, se descubrió que la interfaz gráfica no adaptaba su tamaño al tamaño de la pantalla. Tras investigar porque sucedía esto, se dedujo que había que cambiar el *layout* padre que usaban las interfaces, pues no se estaba usando un *root* que se auto escalase correctamente. Tras realizar este cambio y rehacer las interfaces gráficas, se comprobó que el funcionamiento de estas funcionalidades correspondía a lo que se estaba esperando en las pruebas de aceptación, por lo que se dio por terminado el *Sprint* con todas sus tareas cerradas a tiempo.

En este primer *Sprint* se añadieron las tareas equivalentes al tiempo total disponible en el *Sprint* basándose en la estimación de tiempos que se había realizado de cada tarea, pero a medida que se cerraban las tareas, se descubrió que normalmente se cerraban con una carga de trabajo inferior a lo estimado, por lo que al principio del *Sprint* de sobreestimo el tiempo que se necesitaría para desarrollar dichas tareas. Aunque la diferencia de tiempo no era enorme, ya que se estimaron 40 horas para el *Sprint* y se acabaron realizando 33, esta diferencia no entra en lo ideal en el desarrollo de un proyecto de metodología ágil, pues no debería ser mayor al 5%. Esta sobreestimación en el *Sprint* nos servirá para mejorar la estimación en el resto de *Sprints*.

5.3 *Sprint 2*

En este segundo *Sprint*, como no se arrastró nada del Sprint anterior, se comenzó a realizar las tareas de subir prendas a la app y crear *outfit* aleatorio. Primero se empezó con la tarea de subir prenda, y el primer problema encontrado fue realizando el *mockup* de esta interfaz. Un *mockup* es una maqueta que se realiza antes de empezar a programar una interfaz, en el cual se detalla cómo debe verse la interfaz una vez finalizada su desarrollo. El problema encontrado fue que la interfaz se veía muy sobrecargada, ya que tenía muchas *labels* y selectores en un espacio reducido. Tras realizar varios diseños, se optó por el interfaz mostrado en la Figura 26.

Figura 26. Mockup Interfaz subir prenda

Tras realizar este diseño, se paso a su programacion, que mediante el uso de la herramienta anteriormente mencionada Glide, y la facilidad que aporta FireBase Storage, se llevo a cabo sin mayores problemas.

Despues de terminar con esta funcionalidad, se paso al diseño de la interfaz de la otra funcionalidad preparada para este *Sprint*, la funcionalidad de crear *outfits* aleatorios. El diseño de esta interfaz fue sencillo, y tras finalizarlo, se paso a la implementación de la funcionalidad.

Se comenzo programando el código para para que cuando el usuario pulsase el botón de crear *outfit* aleatorio, un array guardase todas las prendas del usuario, y sin hacer caso a los filtros, sacase aleatoriamente un conjunto . Esta funcionalidad se realizo asi en primera instancia para comprobar como funcionaba la aplicación y cuanto podria tardar en crear un *outfit*. Los resultados que se iban obteniendo eran buenos, creando los outfits de manera rapida y eficaz, pero se detectó un bug durante la fase de programación que hacia que la parte del calzado no se mostrase correctamente. Este bug provoco un retraso signifactivo en el tiempo estimado a esta tarea, ya que fue un error de programación que no se habia comtemplado en un principio, lo que hizo que al terminar el *Sprint*, la funcionalidad funcionase correctamente, pero sacando los outfits sin fijarse aun en los filtros introducidos por el usuario. Esta parte de la funcionalidad de pasó al *Sprint* siguiente como trabajo de alta prioridad.

5.4 *Sprint* 3

Para comenzar con este Sprint, lo primero que se realizó justo después de crear las pruebas de aceptación y *mockups* de las tareas asignadas al *Sprint*, fue terminar la funcionalidad que había quedado pendiente del Sprint anterior por falta de tiempo.

Una vez se terminó esta funcionalidad, se empezó con las tareas que se tenían previstas para este *Sprint*. Primero se empezó con la tarea de Favoritos, haciendo en primer lugar la interfaz de esta funcionalidad, y pasando posteriormente al código. La funcionalidad parecía funcionar correctamente, pero en fase de testeo de pruebas de aceptación, se descubrió que había un bug de carácter grave que hacía que la aplicación se cerrase de manera abrupta si se presiona el botón de pasar al siguiente favorito mientras se estaba cargándola interfaz. Este error venia dado por un fallo en la implementación del código que llamaba a la base de datos, y si se pulsaba rápidamente el botón, la llamada a la base de datos intentaba asignar valores a unas variables inexistentes y producía un error que cerraba la aplicación. Después, se encontró también que sucedía el mismo error si se salía de la pantalla mientras estaba cargando. Para solucionar este error tuvo que hacerse una restructuración del código, pero, aun así, se consiguió cerrar la tarea con un tiempo real de trabajo parecido al estimado.

Después se comenzó a trabajar en la otra funcionalidad esperada para este *Sprint*, la del armario virtual. Viendo que se había dedicado tiempo a una tarea de alta prioridad, el tiempo estimado de esta tarea y el tiempo restante, se decidió dividir esta tarea en 2 más pequeñas para poder cerrar el *Sprint* con una versión sin fallos, aunque más reducida en funcionalidad. Se separo la funcionalidad de mostrar las prendas con la de crear filtros para esa búsqueda de prendas, llevando a cabo solo la primera de estas durante el *Sprint*. La otra mini tarea fue pasada a un siguiente *Sprint* y dejada fuera del primer MVP de la aplicación, que sería posteriormente probado por los usuarios.

Una vez finalizado estos *Sprint*, se realizaron las pruebas de regresión para comprobar que la aplicación seguida funcionando correctamente en todos los aspectos desarrollados.

6. Implantación

Una vez vistos los distintos *Sprints* que han llevado a la realización del proyecto, se puede detallar cómo es el funcionamiento final de la aplicación. Como se puede observar en la Figura 27, lo primero que se ve al abrir la aplicación es la pantalla de Registrase/Inicio de Sesión, dependiendo de si el usuario ya tiene una cuenta creada en *OFM*.

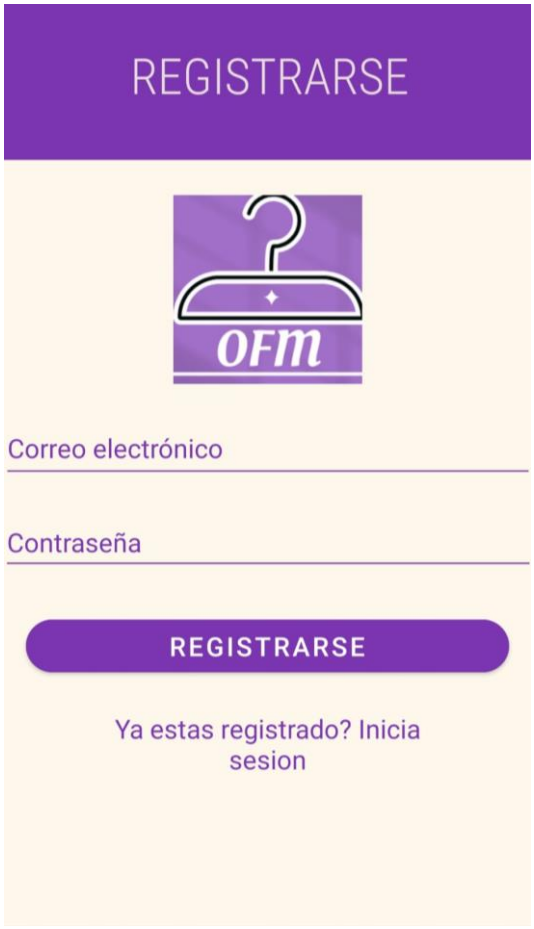

Figura 27. Registrarse en OFM

Cuando el usuario complete este inicio de sesión con un usuario válida, llegara a la actividad principal de la app. Como se ha comentado con anterioridad, se ha usado una actividad por defecto de Android Studio para crear este bloque de pantalla, donde la navegación se hace mediante la barra que se puede observar en la Figura 28.

Figura 28. Barra de navegación

Una vez se ha entrado a esta pantalla, lo primero que ve el usuario es la de creación de outfits (Figura 29) . En esta pantalla el usuario puede crear un *outfit* en base a sus prendas en la base de datos y los parámetros que puede personalizar.

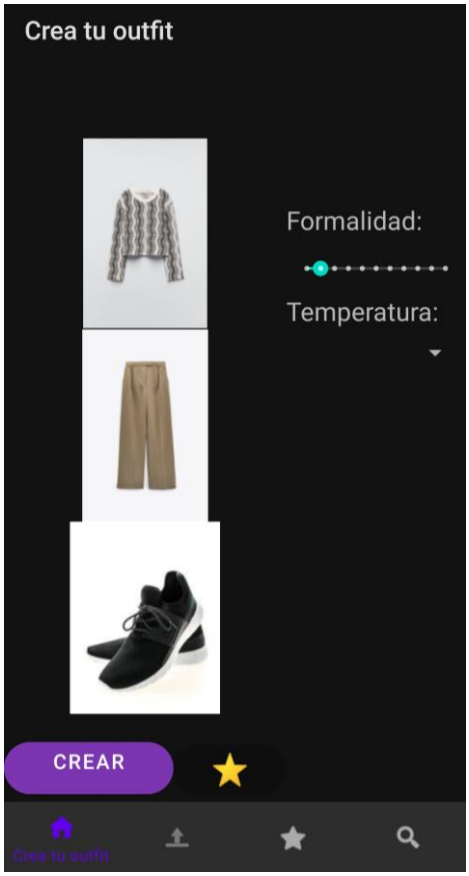

Figura 29. Crea tu outfit

Figura 30. Subir prenda

Si el usuario sigue explorando la barra de navegación, se encuentra con la pantalla de subir prenda (Figura 30) donde rellenando los campos y eligiendo una imagen de la galería del usuario, se puede subir una prenda a la base de datos.

Además, el usuario también puede hacer uso de las funcionalidades de Favoritos, donde puede encontrar todos los *outfits* marcados como favoritos, y la funcionalidad de Armario, donde ver todas las prendas de ropa subidas a la aplicación.

7. Pruebas

En este apartado se pueden observar a ver las pruebas que se han ido realizando para comprobar el correcto funcionamiento de la aplicación en las distintas fases del proyecto. Estas pruebas se pueden dividir en 2 grandes bloques.

7.1 Pruebas de aceptación

Las pruebas de aceptación son un tipo de pruebas propios de la metodología ágil que consiste en, al principio de cada *Sprint* de programación, definir una serie de requisitos que ha de cumplir cada tarea para que se considere que funciona correctamente.

Una vez finalizada una tarea por el programador, pasará a la etapa de *testing* donde, comprobando con las pruebas de aceptación definidas con anterioridad, se decidirá si la tarea tiene un comportamiento adecuado o no, y, por lo tanto, debería volver a la etapa de programación. Se puede observar un ejemplo de una tarea que usa este método en la Figura 31.

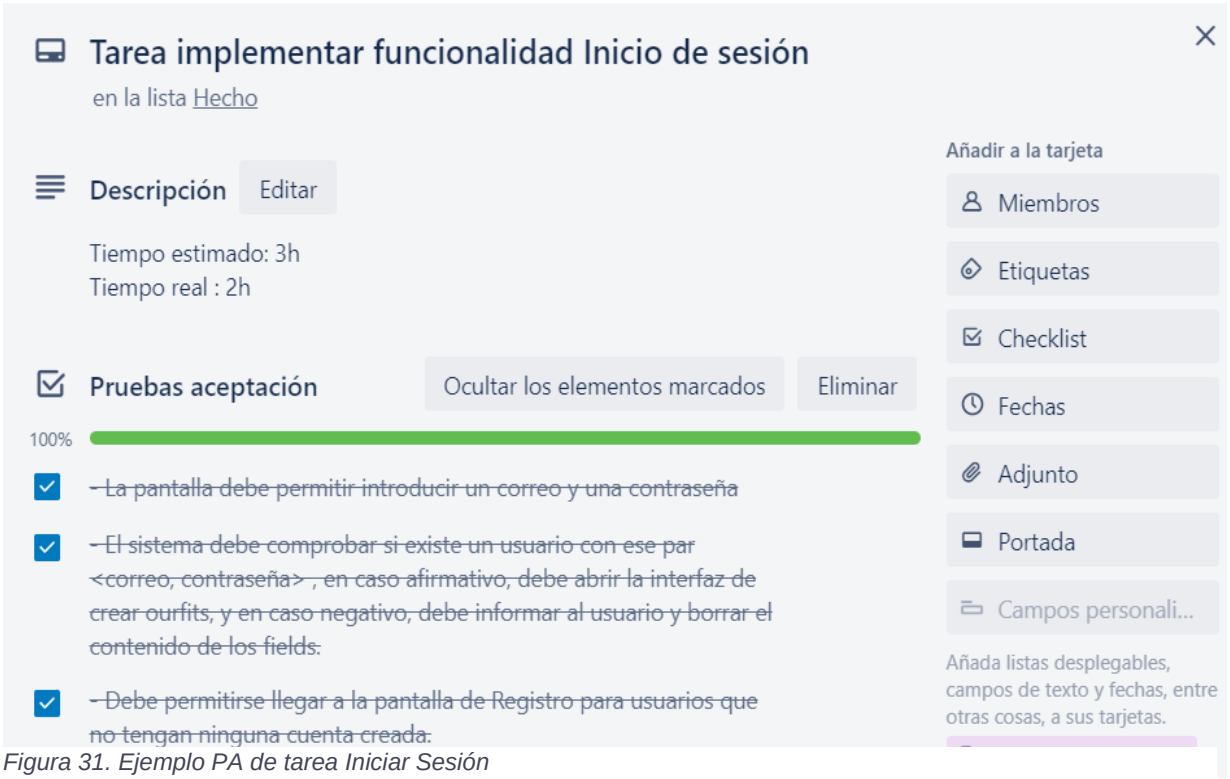

Figura 31. Ejemplo PA de tarea Iniciar Sesión

Una vez realizadas todas las tareas del último *Sprint* antes del primer MVP de la aplicación, se realizaron también las pruebas de regresión, donde se volvió a probar cada PA de todas las tareas para comprobar que seguía funcionando correctamente.

7.2 Pruebas con usuarios reales

Una vez finalizada la programación del primer MVP de la aplicación, se pasó a probar con usuarios reales para conocer su *feedback* de la aplicación. Estas pruebas son muy útiles en el desarrollo ágil pues ayudan a ver posibles debilidades del producto que podrían no haberse tenido en cuenta durante su desarrollo.

Para estas pruebas, se le facilito una *APK* de la aplicación a distintos usuarios, y tras probar la aplicación, rellenaron un cuestionario para conocer como había sido su experiencia tras usar la aplicación. Analizando los resultados obtenidos:

7.2.1 Cuestión I

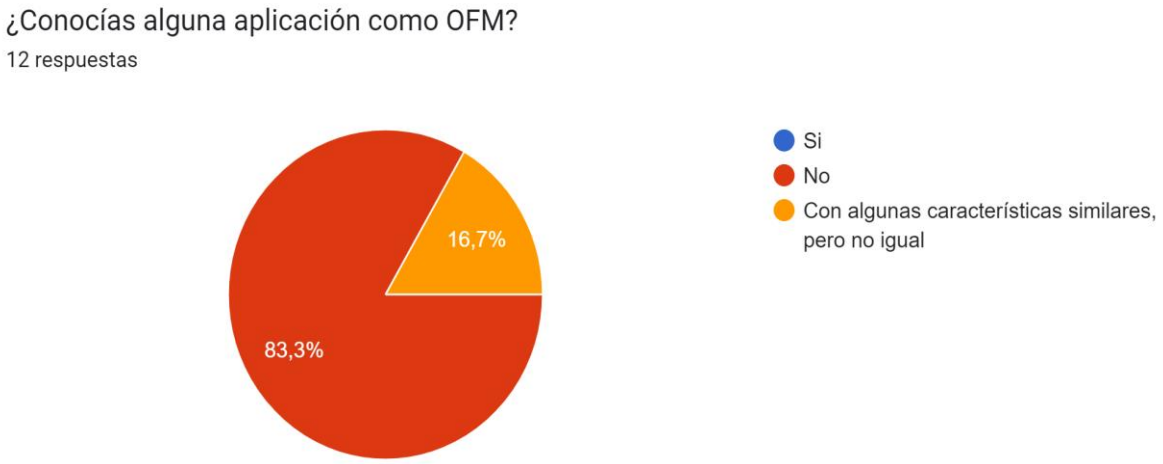

Figura 32. Cuestión 1 Encuesta a Usuarios

Como se puede observar en la Figura 32, la primera pregunta del cuestionario trata de establecer el conocimiento de los usuarios encuestados sobre aplicaciones similares a *OFM*, y como se puede observar por las respuestas de los usuarios, la gran mayoría no conocía ni usaba ninguna aplicación similar, y un pequeño porcentaje sabia de alguna aplicación con características similares, pero no con todas las funcionalidades de *OFM*.

7.2.2 Cuestiones II y III

Figura 34. Cuestión 2 Encuesta Usuarios

Figura 33. Cuestión 3 Encuesta Usuarios

Viendo las respuestas de estas 2 cuestiones (Figura 33 y Figura 34) se puede comprobar que los usuarios no han tenido problemas con la aplicación, es decir, se ha comprobado que funciona correctamente en varios dispositivos. El único problema que ha tenido un usuario ha sido por intentar instalar la aplicación en un dispositivo no Android, lo cual provocaba un fallo ya que la app no está diseñada para dispositivos que no usen un sistema operativo Android.

7.2.3 Cuestión IV

Figura 35. Cuestión 4 Encuesta Usuarios

Como se puede observar en las respuestas a esta pregunta, la Figura 35 muestra que la mayoría de los usuarios consideraría que una aplicación como *OFM* podría ser útil en su vida diaria, aunque en cierto que también consideran que la aplicación necesita mejoras. Este punto es entendible ya que los usuarios solo han tenido contacto con la primera versión de *OFM*, la cual necesita aun mejoras y nuevas funcionalidades aun no añadidas.

7.2.4 Cuestión V

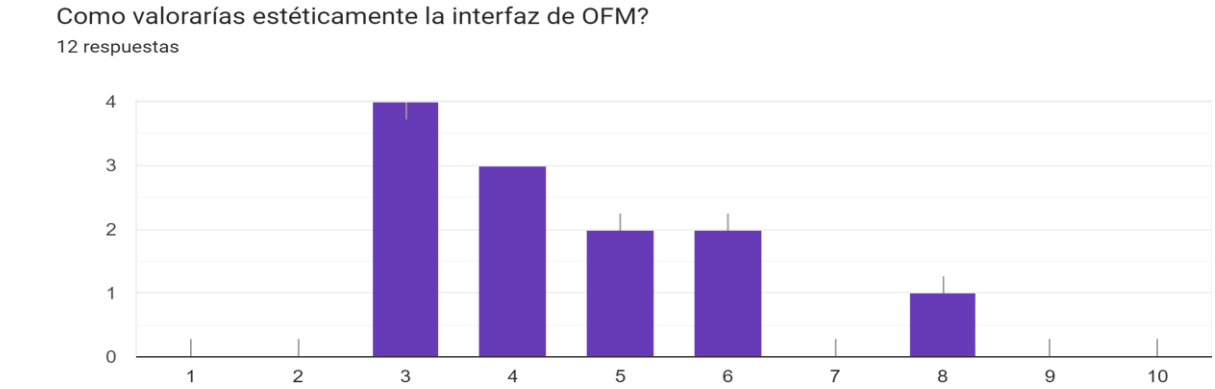

Figura 36. Cuestión 5 Encuestas Usuarios

Esta pregunta sobre la interfaz de la aplicación ha desvelado algo que se sabía ya sobre su diseño, y es que la interfaz de *OFM* pese a ser sencilla, como se puede ver por la valoración de los usuarios en la ilustración Figura 36, es pobre y poco agradable, ya que, dado el tiempo disponible, no se utilizó ningún *Framework* ni se aplicó CSS a los archivos XML que componen la interfaz. Los resultados de la encuesta apoyan la idea de hacer una remodelación de la interfaz aplicando diferentes estilos de cara al 2 MVP de la aplicación.

7.2.5 Cuestión VI

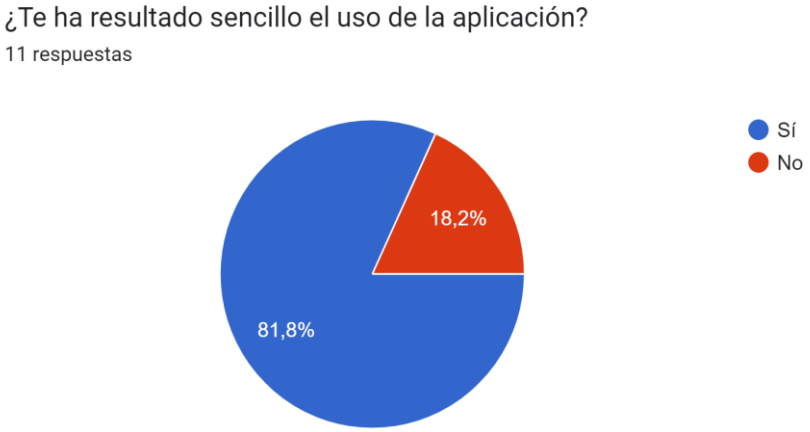

Figura 37. Cuestión 6 Encuesta Usuarios

En esta ilustración, la Figura 37, se puede observar que a la mayoría de los usuarios les ha parecido sencillo e intuitivo el uso de la aplicación.

7.1.1 Cuestión VI

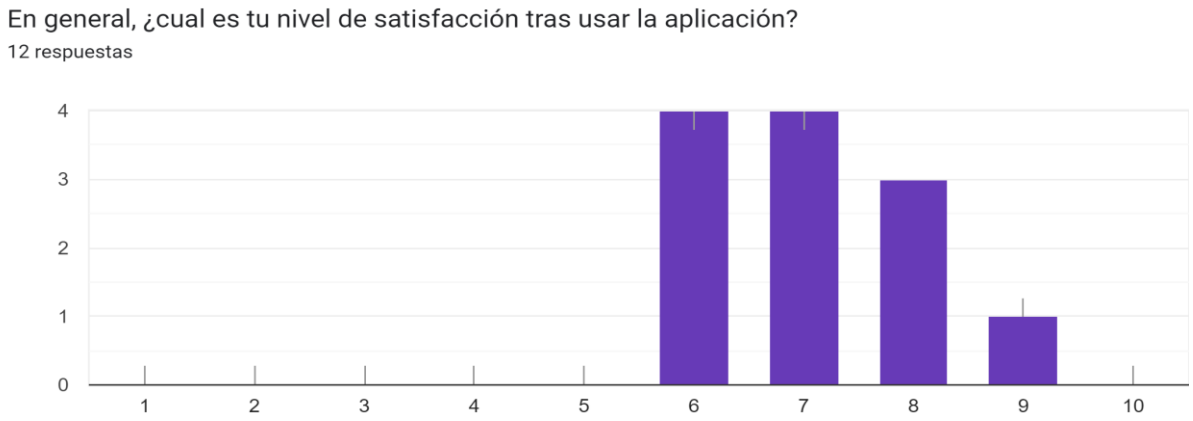

Figura 38. Cuestión 7 Encuesta Usuarios

En Figura 38 se puede ver que, pese a los problemas con la interfaz, el nivel de satisfacción medio con la aplicación es bastante alto.

8. Conclusiones

Para realizar la conclusión de este TFG, se van a recuperar los objetivos que se marcaron en el apartado 1.2 de esta memoria. Primero, se van a ver los subobjetivos que se intentaban cumplir:

-Permitir subir prendas a los usuarios

-Permitir crear *outfits* a los usuarios

-Permitir guardar en favoritos los *outfits* creados por los usuarios

-Permitir enseñar todas las prendas subidas de los usuarios a modo de armario - aplicando filtros

Si se comparan estos pequeños objetivos, se puede observar que se cumple casi en su totalidad el resultado final con lo establecido en los objetivos, a falta del último subobjetivo, pues como se comentó con anterioridad, no se ha implementado la funcionalidad de buscar con filtros en toda la ropa del usuario, por falta de tiempo en el último *Sprint*.

Además de esto, se estableció el objetivo de que la app hiciese a los usuarios ahorrar dinero y tiempo en su día a día, y como se han implementado las funcionalidades necesarias para ello, y la mayoría de los usuarios que han probado la aplicación han respondido correctamente a la encuesta de satisfacción, también se ha completado este objetivo.

8.1 Relación del trabajo desarrollado con los estudios cursados

Sin duda alguna, la realización de este TFG no hubiese sido posible sin el conocimiento obtenido en muchas de las asignaturas cursadas durante estos años, pero dada la naturaleza del proyecto, cabe destacar la importancia que han tenido las asignaturas de Proyecto de ingeniería de Software y Proceso Software, pues con sus simulaciones de proyectos con metodología ágil, han ayudado mucho en la creación de este proyecto.

También existe una gran relación de este proyecto contras asignaturas como Base de datos, que me ayudo en su momento a entender correctamente las bases de datos, por lo que ha resultado más sencillo trabajar con FireBase, o Análisis y especificación de requisitos , que me ha facilitado de identificación de requisitos en este proyecto.

Además, cabe destacar las competencias transversales que me han ayudado durante todo el proyecto, destacando:

- Aplicación y pensamiento práctico
- Diseño y proyecto
- Pensamiento crítico
- Planificación y gestión del tiempo

9. Trabajos futuros

Como se ha comentado con anterioridad durante esta memoria, se planea realizar otro *Sprint* de las aplicaciones para implementar las funcionalidades que no se terminaron en el último *Sprint*, y añadir la funcionalidad de calendario y compartir *outfits*. Además, se debería investigar sobre interfaces en Android Studio y ver que Frameworks permiten dar cambios estéticos a la interfaz actual de la aplicación, para que fuese más agradable.

Como consecuencia de la lista de tareas a realizar que se acaba de mencionar, sería necesario realizar una estimación de tiempo de cada tarea para saber si sería necesario realizar 1 o 2 *Sprints* de programación , pues dada la experiencia previa con los Sprint anteriores y viendo la carga de trabajo restante, es posible que no se pueda realizar todo en ese *Sprint* final.

Una vez finalizada la programación de estas tares, estaremos antes el segundo MVP de la aplicación, que al igual que el primer MVP, debería ser probado por los usuarios reales para ver si existe una mejora real de la aplicación.

Si se considerase que las pruebas con usuarios no fuesen satisfactorias, se podría realizar un mini *Sprint* arreglando los flecos necesarios.

10. Referencias

[1]. Metodología ágil. https://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software [Online, Marzo 2022]

[2]. Android Studio. https://developer.android.com/studio [Online, Marzo 2022]

[3]. FireBase. https://firebase.google.com/docs [Online, Marzo 2022]

[4]. Estadísticas sector moda España. https://fashionunited.es/statistics/estadisticas-de-consumo-y-ventas-de-la-moda-en-espana [Online, Marzo 2022]

[5]. Percentil. https://percentil.com/ [Online, Marzo 2022]

[6]. El 76% de los españoles ha comprado ropa que no ha llegado a estrenar, según un estudio. https://www.europapress.es/economia/noticia-76-espanoles-comprado-ropa-no-llegado-estrenar-estudio-20200505145123.html [Online, Marzo 2022]

[7]. Combyne. https://play.google.com/store/apps/details?id=com.combyne.app&hl=es&gl=US [Online, Abril 2022]

[8]. Smart closet. https://play.google.com/store/apps/details?id=com.rkk.closet&hl=es&gl=US [Abril, Marzo 2022]

[9]. Acloset. https://play.google.com/store/apps/details?id=com.looko.acloset&hl=en_US&gl=US [Online, Abril 2022]

[10]. Glide. https://github.com/bumptech/glide [Online, Abril 2022]

[11]. Github. https://github.com/ [Online, Marzo 2022]

[12]. Moqups. https://moqups.com/es/ [Online, Marzo 2022]

[13]. Trello. https://trello.com/ [Online, Marzo 2022]

11. Glosario de términos

- **APK**: Es un archivo propio de Android que se utiliza para instalar aplicaciones.

- **Backlog**: Una lista priorizada con todas las tareas del proyecto pendientes de asignar a un Sprint

- **DTO**: Objeto que se utiliza para transportar datos entre procesos. En este caso se usa para recibir la información de las consultas a la base de datos, y poder trabajar cómodamente con objetos java.

- **E-Commerce**: Tienda online que se dedica a la compra y venta de productos a través de internet.

- **Framework**: Esquema que ofrece una estructura base para elaborar un proyecto

- **Nivel de API**: Es un valor entero que identifica de manera única la revisión de la Api de Framework que ofrece una plataforma de Android.

- **MVP**: Es una versión mínima de la aplicación a desarrollar con ciertas funcionalidades, preparada para ser enseñada a los usuarios. Durante un desarrollo ágil se suelen crear varios MVP que se utilizan para ver que debería mejorar en la aplicación.

- **SDK**: Es un conjunto de herramientas de desarrollo propio de Android. Algunas herramientas son un depurador, biblioteca, simulador de dispositivo Android, documentación, tutoriales.

- ***Outfit***: Combinación de prendas para un evento.

Anexo I : ODS

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**			X	
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**				X
ODS 4. **Educación de calidad.**				X
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**				X
ODS 9. **Industria, innovación e infraestructuras.**				X
ODS 10. **Reducción de las desigualdades**.				X
ODS 11. **Ciudades y comunidades sostenibles**.				X
ODS 12. **Producción y consumo responsables.**	X			
ODS 13. **Acción por el clima.**			X	
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**				X

Figura 39. Tabla ODS

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

Como se ha podido comprobar en la ilustración anterior, este TFG se puede relacionar con algunos de los ODS, pero destaca sobre todo el ODS 12: *Producción y consumo responsable.*

La razón por la cual este en este TFG se relaciona especialmente con este objetivo es sencilla, uno de los objetivos de este proyecto es reducir la cantidad de ropa que compran los usuarios y acaba en el fondo del armario, sin haberse ni siquiera estrenado.

Con el uso de *OFM* se pretende que los usuarios sean más conscientes de la ropa que poseen actualmente, y no hagan compran que no necesiten, ya que esta acción tiene consecuencias tanto económicas para el usuario que compra ropa que al final será inútil, como consecuencias medioambientales, pues esas prendas que se han comprado han sido fabricadas en un proceso industrial, usando materias primas, y después han sido transportadas hasta la localización donde se venda, generando un gasto de combustible muy perjudicial para el medio ambiente.

La forma en la que *OFM* puede ayudar a que este gasto innecesario no ocurra es mediante la función de Armario Virtual, pues ayudara a los usuarios a mantener una mayor organización de su ropa para saber en todo momento si una prenda de ropa es realmente necesaria o ya tienen algo muy parecido en su armario.

Todo esto mencionado anteriormente sobre el ahorro de un gasto innecesario nos introduce también a otro de los ODS que se relacionan con este proyecto el ODS 1 *Fin de la pobreza*, puesto que la aplicación ayudara a muchos usuarios a tener un mayor control y no realizar gastos innecesarios en ropa. Además, mientras organizan su armario con *OFM*, los usuarios pueden darse cuenta de que ya tienen entre sus pertenencias ropa con características similares. Esta ropa podría ser dada a asociaciones que repartan las prendas entre la gente más necesitada, ayudando a un gran número de personas.

Además, este TFG también se puede relacionar con el ODS 13 *Acción por el clima* pues como se ha mencionado anteriormente, que los usuarios hagan un consumo responsable de las prendas de ropa repercute directamente sobre el medio ambiente, pues el gasto de fabricación y transporte de esas prendas de ropa tiene un impacto elevado en farios factores que perjudican el medio ambiente y agravan el problema del cambio climático, sobre todo destacando el consumo de combustibles fósiles que se realiza en el transporte de esas prendas de ropa.