UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

HOMEBUDGET

Aplicación móvil para la gestión del hogar

Alumno: Daniel Vallecillo Meneses

Tutora: María Luisa Martín Pérez

“Making the simple complicated is commonplace; making the complicated simple, awesomely simple, that’s creativity” – Charles Mingus

“Si es bueno vivir, todavía es mejor soñar, y lo mejor de todo, despertar” – Antonio Machado

“If the opportunity doesn’t knock, build the door” – Milton Berle

“The only way to do great work is to love what you do” – Steve Jobs

“We know what we are, but not what we may be” – Shakespeare

AGRADECIMIENTOS

A mis padres, pues sin ellos recorrer este camino no habría sido posible. A mi hermano, por haber sido siempre un ejemplo a seguir y ser una gran persona. A todos mis amigos, que directa o indirectamente me han ayudado a seguir adelante. A mis compañeros, que han formado parte de esta etapa de mi vida, en especial a Luis, llegué sin nada y regresé con un gran amigo.

Por supuesto a los profesores de este grado, ellos son el motor que ha hecho posible este aprendizaje, y en especial gracias a Marisa, mi tutora, por su tiempo, dedicación, cariño, profesionalidad y simpatía.

RESUMEN

El objetivo principal de este proyecto es la creación de una aplicación móvil Android para la gestión del hogar con la finalidad de poder llevar un control de los gastos cotidianos de una manera sencilla y visual.

Por otro lado, se desea dibujar un marco contextual del estado del arte actual en el desarrollo de aplicaciones móviles poniendo el foco en las desarrolladas para la plataforma Android.

Por último, se desea proporcionar las herramientas necesarias para la administración de la aplicación y además desarrollar ambas partes utilizando las últimas tecnologías, librerías y patrones de diseño disponibles de manera que éstos aporten el máximo valor posible tanto a usuarios finales, cómo a administradores.

Palabras clave: aplicación móvil, Android, Google Analytics, hogar, presupuesto.

ABSTRACT

The main objective of this project is the creation of an Android mobile application for home management in order to be able to keep track of daily expenses in a simple and visual way.

On the other hand, we want to draw a contextual framework of the current state of the art in the development of mobile applications, focusing on those developed for the Android platform.

Finally, we want to provide the necessary tools for the administration of the application and also to develop both parts using the latest technologies, libraries and design patterns available so that they contribute the maximum possible value to both end users and administrators.

Keywords: mobile application, Android, Google Analytics, home, budget.

ÍNDICE GENERAL DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

CAPÍTULO 1

INTRODUCCIÓN

Hoy en día, las personas cada vez sienten más dependencia de sus smartphones, el mundo ha cambiado. La compra de productos a través de comercios electrónicos, los pagos usando el smartphone y las microtransferencias están a la orden del día.

Esta revolución tecnológica se disparó a partir de 2007 tras la salida al mercado de la primera generación de iPhone y continuaría años venideros con Android. Si bien es cierto que ya en aquel entonces la economía y los mercados estaban globalizados, los smartphones terminaron por romper las barreras de localización y accesibilidad. Reinventaron las normas. A nadie nos sorprende ya ver a gente comprando con su smartphone a través de aplicaciones o plataformas móviles pertenecientes tanto a comercios físicos como virtuales o simplemente pagando en supermercados y otras tiendas físicas sin tener que utilizar tarjetas o dinero en efectivo.

Esta situación se traduce en una mayor facilidad para gastar nuestro dinero desde cualquier lugar y en cualquier momento, pero, también se traduce en ausencia de recibos físicos y en un mayor descontrol sobre la gestión de nuestro presupuesto.

Además, otros gastos derivados del hogar como facturas de internet, alquileres y seguros de todo tipo pasan por nuestra cuenta antes incluso de que nosotros podamos verlo, indiferentemente de si contábamos con ese gasto o no. La magia de lo digital.

Si el dinero lo tienes en formato físico jamás podrás asumir un gasto mayor del total del que dispones, pero hace mucho que las cosas dejaron de funcionar así, por ello, la necesidad de una herramienta que nos permita controlar nuestros gastos y ofrecernos con un simple vistazo un resumen de nuestra situación económica personal actual es una necesidad.

1.1 ORGANIZACIÓN DEL DOCUMENTO

En esta sección encontramos un pequeño resumen de los diferentes capítulos y contenidos en los que se subdivide este documento para ofrecer así, una visión global del conjunto. El documento está dividido en diez capítulos que abordan los principales puntos referentes al proceso de desarrollo de software.

Capítulo I Introducción: En este capítulo se muestra una introducción completa al proyecto, se habla sobre la motivación para la realización de este proyecto, los objetivos marcados y alcance del mismo, se respondes a cuestiones necesarias para entender el conjunto del proyecto cómo que es Firebase, para que lo vamos a usar en nuestro proyecto y porqué.

Capítulo II Contexto del documento: En este capítulo trata de ofrecer al lector una visión global del marco tecnológico en el momento de comenzar este proyecto, en él, se habla sobre las opciones para desarrollar este tipo de aplicaciones que están disponibles en el mercado, tendencias en el diseño de interfaces, las librerías más utilizadas actualmente y arquitecturas empleadas en la actualidad. Además, se ofrece un resumen rápido del estado del arte.

Capítulo III Gestión del proyecto: En este capítulo se aborda los temas relacionados con la gestión del proyecto, entre los que se incluyen la metodología empleada para el desarrollo del proyecto, la planificación temporal realizada, la estimación presupuestaria y finalmente el coste real resultante al terminar el proyecto.

Capítulo IV Análisis: En este capítulo se trata todos los aspectos relacionados con el análisis del sistema correspondiente a la fase de elicitación previa al comienzo del desarrollo.
Se definen los posibles actores del sistema, los requisitos de usuario, los casos de uso derivados de los mismos, una especificación y diagrama de éstos, se definen los requisitos no funcionales del sistema y los de información.

Capítulo V Diseño: En este capítulo se aborda todo lo relacionado con el diseño del sistema que se realizó una vez la fase de análisis finalizó.
La arquitectura lógica y física del sistema, un diagrama de clases, el diseño de modelo lógico de datos y el diseño de la interfaz.

Capítulo VI Implementación: En este capítulo se trata todo lo relativo a la implementación del sistema, desde las herramientas utilizadas para ese fin o la estructura del proyecto hasta detalles de diferentes componentes importantes de la aplicación.

Capítulo VII Pruebas: En este capítulo se recogen todas las pruebas de caja blanca y caja negra a las que se ha sometido a la aplicación para certificar un correcto funcionamiento de la misma.

Capítulo VIII Manuales: En este capítulo se detallan los pasos necesarios para la instalación desde cero del sistema (manual de instalación), el manual de usuario de la aplicación para un correcto uso y además un manual de administrador para mostrar el uso de la consola de Firebase en las tareas generales de administración del sistema.

Capítulo IX Conclusiones: En este capítulo se habla sobre los problemas afrontados durante el desarrollo del proyecto, las conclusiones obtenidas a su finalización y las mejoras futuras que podrían mejorar la aplicación y permitirla seguir creciendo.

Capítulo X Referencias: Recoge el conjunto de referencias utilizadas a lo largo del proyecto.

1.2 MOTIVACIÓN

Para explicar la motivación de este TFG hace falta echar la vista atrás y entender cómo la tecnología ha evolucionado en los últimos años hasta poder identificar un claro patrón común. Y es que, la tendencia de los últimos años se ve representada en la reutilización, la modularización, la portabilidad, la escalabilidad y la disponibilidad. Esta tendencia en los diferentes ámbitos del desarrollo de software ha alcanzado estos años su máximo exponente en las plataformas móviles.

Estamos ante un panorama en el que las plataformas móviles cambian cada año con cada versión de los sistemas operativos. Por ello, decidí investigar acerca de los avances referentes a los patrones de arquitectura Android, nuevos elementos, nuevas funcionalidades y oportunidades. Bajo ese contexto, entendí, que no debía preocuparme en exceso de cómo sería mi producto final, sino, de cómo iba a hacer posible que el producto siguiera evolucionando antes de que estuviese construido.

Desechar prácticas usadas hasta antes de ayer y tratar de implementar los últimos avances en busca de alcanzar las características anteriormente mencionadas era el objetivo.

Encontré una gran oportunidad en la propuesta de proyecto de María Luisa Martín Pérez ya que éste me permitía poner en práctica cosas como la implementación del proyecto empleando *Firebase*, nuevas animaciones, patrones de diseño y otras funcionalidades que han sido incluidas en la aplicación.

Pero no sólo eso, además iba a permitirme obtener una visión global completa del proceso de desarrollo de software ya que debía cubrir por mi mismo cada una de las etapas, y eso, era una fantástica oportunidad.

Por todo ello, bajo el marco anteriormente descrito y con el tema del proyecto de la mano, entendía que esta propuesta era una oportunidad para poder desarrollar una herramienta que ofreciera un mayor control sobre los gastos cotidianos y los periódicos.

1.3 OBJECTIVOS Y ALCANCE

El principal objetivo de este proyecto es el estudio de las nuevas tendencias en el desarrollo de las tecnologías Android y posterior utilización de las mismas en el desarrollo de un producto final, una aplicación para la gestión del hogar y su correspondiente herramienta de administración.

En sí misma, la aplicación tendrá como objetivo **recabar los datos financieros** de los usuarios y permitir ser visualizada de manera simple y concisa ofreciendo una idea clara de la situación económica actual así cómo de las principales fuentes de gastos. Además, otro de los objetivos principales de la aplicación es proveer al usuario de un sistema de almacenamiento de la información que **no dependa del dispositivo utilizado** para el acceso ni del SO.

Además de todo ello, la **escalabilidad** tiene que ser un objetivo indispensable en nuestro proyecto, permitirnos crecer rápidamente en función de los usuarios es una necesidad en este mercado de viral. Muchos usuarios, en muy poco tiempo, a menudo se traduce en un deterioro del servicio si éste no es lo suficiente escalable.

Sin embargo, el crecimiento de una aplicación no sólo se ve representado por el aumento de usuarios, también por el aumento de calidad de la misma y esto al final se traduce en cubrir las necesidades de los usuarios de la aplicación. Por ello, la toma de decisiones adecuada a menudo también marca la diferencia, delimita o propulsa un producto.
Si queremos construir una herramienta pensando en el futuro debe haber otro aspecto fundamental que tener en cuenta: El **análisis de los datos**.

Analizar los datos de la aplicación nos va a permitir entender mejor el comportamiento de los usuarios y entender mejor sus necesidades, no obstante, no podemos limitarnos a analizar únicamente el comportamiento de los usuarios. Después deben poder sacarse conclusiones al respecto, y eso no es posible, si los datos recabados no tienen la suficiente calidad.

Por último, la aplicación debería poder ser usada por cualquier usuario con un dispositivo Android con acceso a Internet sin importar el rango de edad o localización. Por ello la interfaz debe de ser sencilla de utilizar.

1.4 CARACTERÍSTICAS PRINCIPALES DEL SISTEMA

El producto desarrollado resultante del estudio adjunto se trata de una aplicación móvil llamada HomeBudget, que da soporte a cualquier versión del sistema operativo Android, aunque, ofreciendo a partir de su versión 5.0 una experiencia más completa. Encontramos el marco de funcionalidad de la aplicación en el contexto de la gestión de pagos en el hogar y en la cotidianidad del día a día.

Para satisfacer estos objetivos principales la aplicación deberá de ofrecer las siguientes características:

- Permitir a los usuarios de la aplicación registrarse.
- Permitir a los usuarios de la aplicación gestionar sus gastos o ingresos (movimientos).
- Permitir a los usuarios de la aplicación autenticados un correcto acceso a la información de su cuenta.
- Permitir a los usuarios de la aplicación visualizar de manera gráfica el historial de gastos.
- Permitir a los usuarios de la aplicación recibir notificaciones.
- Permitir a los administradores gestionar las cuentas de usuario.
- Permitir a los administradores gestionar de manera sencilla la base de datos.
- Permitir a los administradores enviar notificaciones push a los usuarios.
- Permitir a los administradores gestionar los datos de Google Analytics recogidos por el sistema.

1.5 FIREBASE EN NUESTRO SISTEMA

¿QUÉ ES FIREBASE?

Firebase entra dentro de la categoría de los últimamente tan populares SaaS (Software as a Service) que facilitan el desarrollo de software ofreciendo ciertas funcionalidades ya preparadas para ser consumidas por nuestros productos.

Esta herramienta se centra en ofrecer servicios a diferentes niveles.
Desarrollo, **Crecimiento** y **Ganancias** constituyen las principales áreas de Firebase y para cada una de ellas ofrece múltiples funcionalidades

- Las funcionalidades ofrecidas dentro de la categoría de desarrollo están enfocadas a facilitar el desarrollo rápido de apps manteniendo una alta calidad.
- Las funcionalidades ofrecidas dentro la categoría de crecimiento tienen como objetivo permitir aumentar la base de usuarios sin esfuerzo ni dificultad, nos aportan escalabilidad.
- Las funcionalidades ofrecidas dentro de la categoría de ganancias están orientadas a monetizar la app, a obtener beneficios a través de la misma.

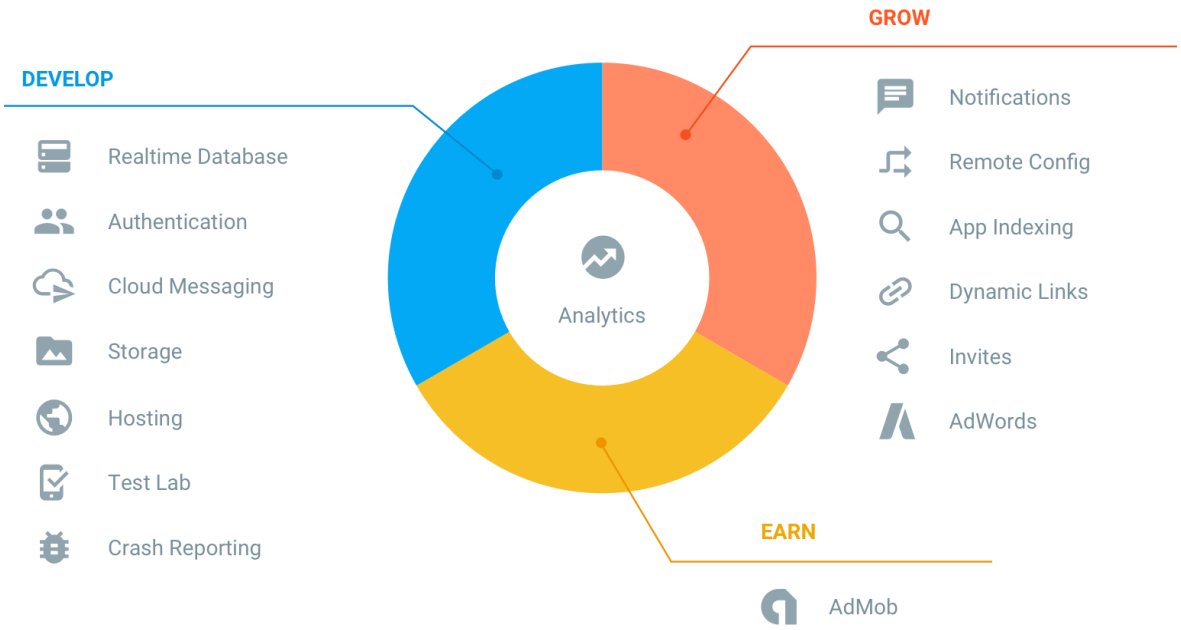


Ilustración 1 - Servicios Firebase

¿POR QUÉ FIREBASE EN NUESTRO PROYECTO?

Uno de los pilares básicos sobre los que construir aplicaciones móviles hoy en día es la escalabilidad, a menudo vemos ejemplos de aplicaciones que se viralizan y terminan rehaciendo su aplicación o presentando otra versión de la misma debido a que la inicial no era escalable y no es capaz de afrontar la demanda de usuarios.

Encontrar el éxito y la viralización de una aplicación hoy en día es un tremendo reto en un mercado tan saturado, pero si encima ésta no escalable, nuestra aplicación habrá muerto antes de ver la luz.

Firebase pertenece a Google, no obstante, no sólo está disponible para Android sino que también para iOS, web y hoy en día también está disponible para plataformas como Unity. Además, la herramienta ofrece la confianza y madurez necesaria contando con el apoyo de un gigante como Google y siendo integrada en multitud de aplicaciones móviles globalmente utilizadas y líderes de sus mercados como son:

Ilustración 2 - Duolingo logotipo

Ilustración 3- The New York Times logotipo

Ilustración 4- Shazam logotipo

Ilustración 5- Aliexpress logotipo

Nos permite construir aplicaciones de manera rápida sin tener que preocuparte por la infraestructura, pues estarás usando las de Google y además podrás gestionar desde una misma consola todos los servicios que se han implementado en la aplicación.

Aunque Google ya ofrecía un servicio de base de datos en tiempo real a través de Firebase, llamado Realtime Database, ahora tiene un nuevo servicio en fase de Beta, es una nueva versión del mismo que facilita el escalado global y facilita la sincronización entre dispositivos online y offline, se llama Cloud Firestore y es el que emplearemos en nuestro proyecto.

¿QUÉ HERRAMIENTAS DE FIREBASE SE USAN EN EL PROYECTO?

- **Cloud Firestore**: Nos permitirá utilizar Firebase como BaaS (Backend as a service) en nuestra aplicación para tener en la nube nuestros datos de manera que podremos acceder a ellos desde cualquier dispositivo. En definitiva, lo emplearemos como servidor en la nube.
- **Google analytics**: Nos ayudará a tomar en el futuro decisiones inteligentes basadas en el comportamiento de los usuarios.
- **Gestión de usuarios**: Se utiliza el módulo de autenticación para gestionar la misma, editar las plantillas de correos enviados a los usuarios.
- **Notificaciones Push**: Nos dará la posibilidad de enviar notificaciones push a grupos de usuarios.
- **Integración con servicios de redes sociales:** Aprovecharemos esta integración con servicios de redes sociales en combinación con la gestión de usuarios para entre otras cosas ofrecer autenticación a través de Facebook.

1.6 CONTENIDO DEL SOPORTE DIGITAL

El soporte digital elegido va ser el DVD, en la raíz de mismo se incluirán tres archivos:

- Carpeta “Proyecto”: que incluirá la carpeta con todo el código fuente de la aplicación Android y el archivo .apk de la aplicación.

- Carpeta Memoria: Incluirá el archivo PDF de esta memoria.

- Leéme: Se trata de un breve archivo .txt con el link y claves de acceso a la herramienta de administración, así como, pares de claves correo/contraseña ya preparadas para la aplicación en caso de desear ahorrarse el paso del registro. También un importante aviso de usabilidad ya que se ha incluido una funcionalidad extra para facilitar el testeo de la app.

CAPÍTULO 2

CONTEXTO DEL DOCUMENTO

2.1 MARCO TECNOLÓGICO

Como parte del inicio del proyecto se realizó un estudio sobre las diferentes opciones disponibles para desarrollar el proyecto. Desde diferentes entornos de desarrollo, librerías externas, servicios, últimas tendencias en desarrollo Android etc… Este capítulo pretende ser un resumen de todo ello.

Además, al final del capítulo también se van a analizar las diferentes soluciones enfocadas a la gestión de gastos que se encuentran disponibles en el mercado señalando sus similitudes, diferencias, ventajas y desventajas sobre nuestra aplicación.

2.1.1 ENTORNOS DE DESARROLLO

Android Studio

Es el entorno de desarrollo para Android de Google, apareció como sustituto oficial de Eclipse y viene con todas las bibliotecas y herramientas necesarias para el desarrollo de la aplicación de principio a fin. Actualmente se encuentra en su versión 3.0, en ella presenta novedades tales como soporte para el lenguaje de programación Kotlin, Java 8, soporte para las instaApps, gestor de fuentes y un asistente para indexar la aplicación a Firebase entre otras muchas novedades.

Titanium SDK

Este entorno ofrece a los desarrolladores la posibilidad de utilizar JavaScript para crear aplicaciones multiplataforma. Pertenece a Appcelerator y aunque hasta hace poco estaba bajo licencia Apache hoy en día existe una política de precios. Ofrece depuración de código, sistema de pruebas, monitorización y recopilación de datos. Es una opción bastante utilizada para el desarrollo de apps multiplataforma.

Xamarin

Es una herramienta de desarrollo de aplicaciones nativas que apuesta por el desarrollo en C#, aunque es posible utilizar Java, Objective-C o Swift. Xamarin proporciona interfaces nativas y una experiencia a la altura de la misma. Además, facilita la reutilización de código y al igual que sus principales competidores cubre todo el ciclo de vida de una app.

En nuestro caso Android Studio será la opción elegida por las siguientes razones:

- Ejecución de app en tiempo real.
- Posibilidad de ejecutar la aplicación directamente en el smartphone.
- Estabilidad.
- Renderizado en tiempo real.
- Herramientas Lint.
- Integración con gitHub.
- Pruebas unitarias con JUnit.

- Facilita la traducción de la app a varios idiomas.

2.1.2 TENDENCIAS EN ARQUITECTURAS MÓVILES

Sin duda se trata de uno de los temas más controvertidos de los últimos años. Ha surgido una necesidad cada vez más imperiosa, de intentar construir apps Android que sigan los conceptos planteados por Robert C. Martin en su ya famoso artículo acerca de Clean Architecture y SOLID en Android.

Una arquitectura Clean es una arquitectura que ha sido construida siguiendo una metodología aplicada, siguiendo una serie de principios, una metodología que aboga por la separación de dependencias y reutilización por bandera.
El objetivo no es solo conseguir no sólo calidad de código, sino asegurar un cómodo mantenimiento y crecimiento en el tiempo. No se trata de un dogma y aunque todas las implementaciones comparten unas características en común, no hay dos iguales.

El problema es que el diseño original de Android no pretendía ser Clean ni cumplir SOLID. Bajo este marco, en los primeros años los desarrolladores Android comenzaron a hacer sus propias implementaciones y surgieron algunas arquitecturas que tenían mucho más en común de lo que podía parecer. Algunos ejemplos son:

- Hexagonal Architecture de Alistair Cockburn y adoptada por Steve Freeman y Nat Pryce en su libro “Growing Object Oriented Software”.
- Onion Architecture por Jeffrey Palermo.
- DCI de James Coplien y Trygve Reenskaug.
- BCE de Ivar Jacobson mostrada en su libro “Object Oriented Software Engineering: A Use-Case Driven Approach”.

A pesar de sus diferencias todas ellas:

1. Son independientes de frameworks. La arquitectura no depende de la existencia de librerías que aporten características software permitiendo usar estos frameworks como herramientas y no como una necesidad.
2. Son testeables. Las reglas de negocio pueden ser testeadas con independencia de la interfaz de usuario, base de datos, servidor web o cualquier otro elemento externo.
3. Son independientes de la interfaz. La interfaz puede ser modificada fácilmente sin modificar el resto del sistema, sin afectar a la lógica de negocio.
4. Existe independencia con la base de datos. Puedes migrar de una base de datos a otra sin afectar a las reglas de negocio.
5. Mantienen independencia de elementos externos. Las reglas de negocio no saben nada acerca de nada exterior a ellas.

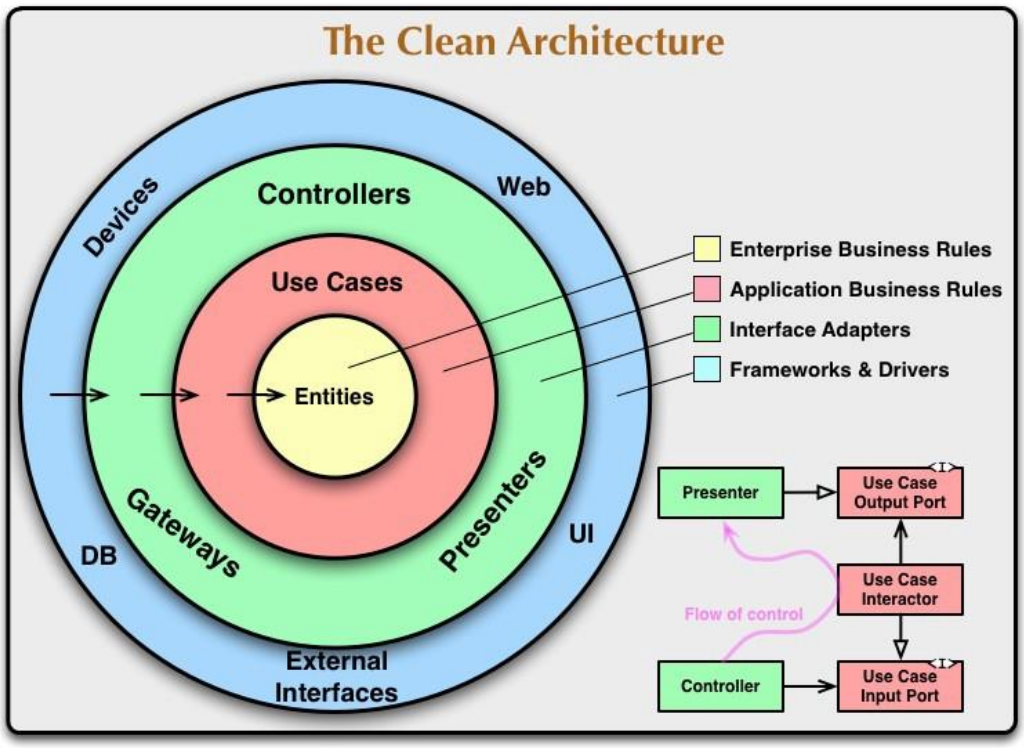


Ilustración 6- Arquitectura clean

Cabe destacar que todas las implementaciones de arquitecturas que reflejan esta metodología están basadas estos círculos concéntricos de separación de dependencias aparecidos por primera vez en el artículo anteriormente mencionado.

2.1.3 ESTADO DE LAS LIBRERÍAS

En estos tiempos es común el uso de bibliotecas en Android que nos proporcionan mecanismos que facilitan el desarrollo en tareas tan comunes como el consumo de APIs, el tratamiento de imágenes o para evitar el tedioso boilerplate (Código que se repite, pero debe de ser incluido en el código aun así).

Algunas de las bibliotecas que colman el panorama actual en el desarrollo Android son:

- **Retrofit**: Es una de las librerías más populares en Android, nos permite realizar de manera sencilla peticiones a una API REST.

- **Glide**: Es una librería enfocada a la gestión de imágenes. Permite la implementación de caché, redimensionar imágenes, soporte para GIFs…

- **GSON**: Es una librería desarrollada por Google empleada para la conversión de POJOs en (Plain Old Java Object) en JSON (JavaScript Object Notation) y viceversa. Es un complemento ideal para Retrofit.

- **LeakCanary**: Es una librería desarrollada por Square que nos permite detectar fugas de memoria y gestionar los errores asociados.

- **DBFlow**: Se trata de ORM (Object-Relational Mapping) muy potente que nos facilita la persistencia de nuestros objetos y evita la necesidad de escribir nosotros mismos las querys SQL.

- **Butterknife**: Se trata de una librería que sirve para reducir el anteriormente mencionado boilerplate, concretamente evita realizar la llamada “findViewById” para cada vista.

- **RXJava**: Es una librería muy empleada en el desarrollo de componentes reactivos.

- **Dagger2**: Nos facilita la inyección de dependencias en nuestro proyecto, aunque la versión inicial fue desarrollada por Square al igual que Retrofit y LeakCanary, actualmente es mantenida por Google.

Para nuestro proyecto se aprovecharán las características ofrecidas algunas de las anteriormente mencionadas y otras no tan conocidas.

2.1.4 TENDENCIAS EN EL DISEÑO

Material Design

Material Design es un lenguaje visual compuesto por un conjunto de guías y buenas prácticas de diseño que tienen el objetivo de dotar a las aplicaciones de una identidad común. Aunque fue presentado por Google en la conferencia Google I/O de 2014, aún sigue vigente y actual.

Está basado en la representación de objetos materiales donde las sombras, la luz y el color toman un papel principal. Se permite la superposición de objetos, pero nunca que se atraviesen los mismos, las animaciones deben ser lógicas y los colores suelen ser llamativos. Google ha desarrollado precisas guías de cómo llevarlo a la práctica y uno de los puntos fuertes es su transversalidad, pues mantiene el detalle independientemente del tamaño de la pantalla.

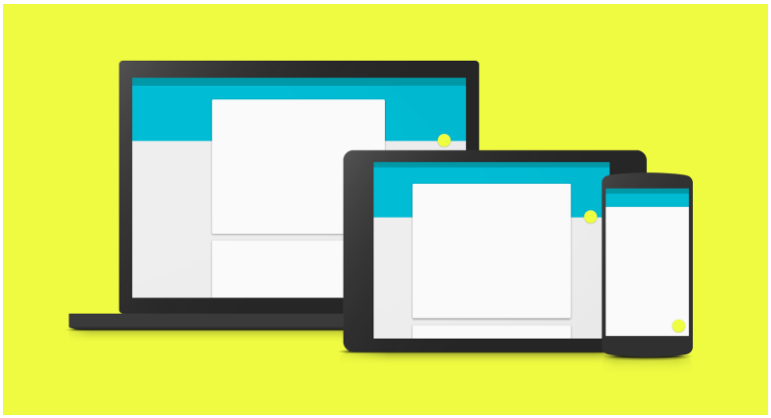


Ilustración 7- Material Design example

Normalización de iconos

Se trata de una novedad presentada con Android Oreo 8.0, y es que el diseño de iconos ha sido controversial en los últimos años para los diseñadores, pues no existían unas directrices claras a seguir más allá de respetar las dimensiones detalladas. Esto se traducía en iconos con forma

circular, con bordes redondeados o cuadrados, tirando por tierra el esfuerzo de Material Design por crear una cierta homogeneidad e identidad.

Eso ha cambiado, ahora deberán estar diseñados en dos capas, en una primera el icono de la aplicación y en una segunda capa el fondo. De esta manera los iconos podrán adaptarse a la interfaz de usuario y además permitirá a los diseñadores crear animaciones para cuando el usuario accede a la aplicación, la arrastra o mueve el cajón de aplicaciones.

Fragments

En primer hay que entender la tendencia al cada vez más común empleo de los fragments como una necesidad tanto a nivel de eficiencia, como a nivel de reutilización.

Pongámonos en contexto antes de la aparición de este elemento.
En los últimos años Android ha sido adoptado como sistema operativo en multitud de dispositivos (no solamente smartphones) con una cantidad increíble de diferentes tamaños de pantalla a la vez que los propios smartphones salían al mercado con pantallas cada vez más grandes.
Todo ello hizo que crear aplicaciones que dieran soporte a un amplio rango de tamaños diferentes fuera insufrible. Es aquí es donde entraron a actuar los fragments, estos nos permiten admitir diseños de UI más dinámicos para las pantallas más grandes.

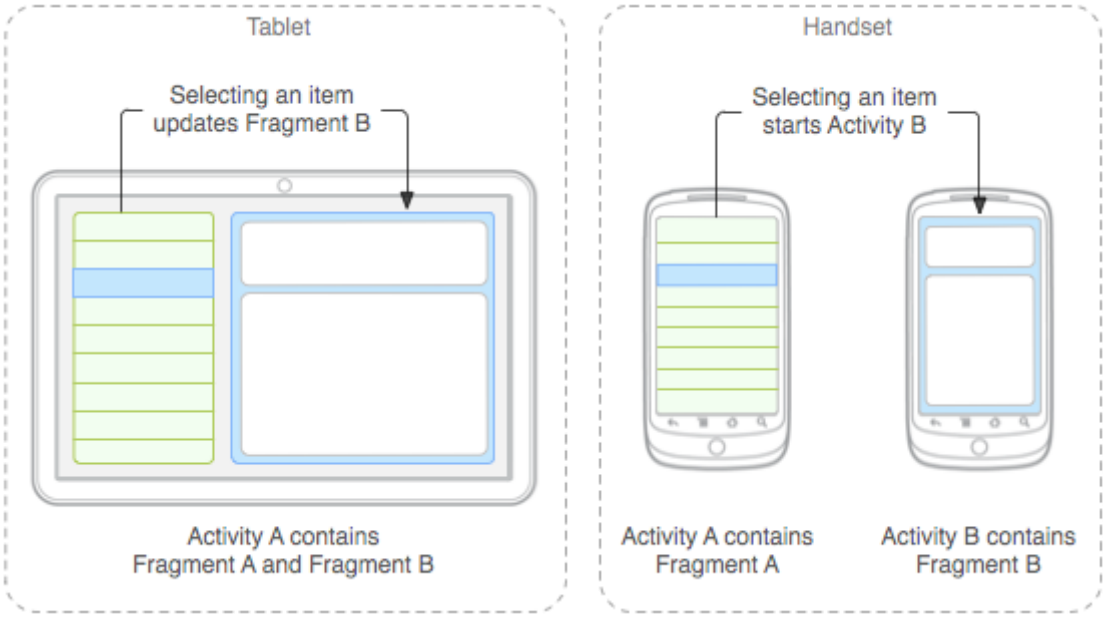


Ilustración 8- Fragments

La clave del funcionamiento de los fragmentos es que puedes modificar el aspecto de la actividad durante la ejecución en función del tamaño de la pantalla u orientación.

Un ejemplo clásico es un gestor de correos electrónicos en el que en la actividad se muestra una lista de correos y al hacer tap sobre alguno de ellos nos lleva a otra actividad en el que nos muestra en detalle el correo seleccionado de la lista. Sin embargo, en la pantalla de una tablet o un smartphone con la pantalla suficientemente grande en orientación horizontal muestra la lista de correos y a la vez al lado el correo seleccionado en detalle todo dentro del mismo activity.

BottomNavigationView

Es un nuevo patrón de navegación, que a pesar de que en Marzo de 2016 fue incluido por Google en su guía de Material Design, no había estado disponible una implementación oficial para este componente hasta inicios de 2017.

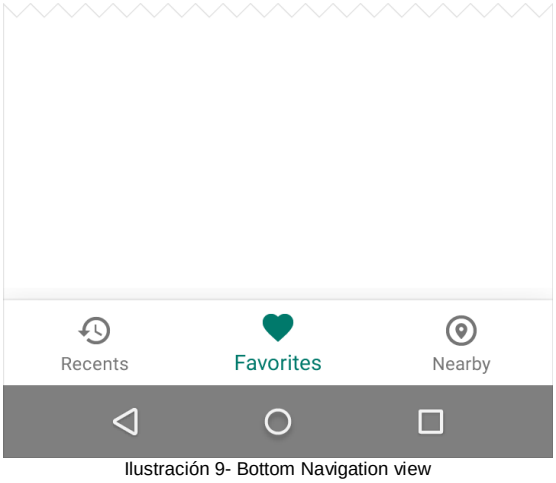


Ilustración 9- Bottom Navigation view

Consiste en una barra de navegación situada en la parte inferior de la pantalla que nos facilita el cambio de vistas de nivel superior. Podemos encontrar un claro ejemplo de su uso en la aplicación Instagram. Es una opción ideal para ofrecer entre tres y cinco acciones en menú. La ventaja de este patrón es que nos permite explorar y cambiar entre vistas de alto nivel con una sola pulsación. También es una opción ideal para opciones que requieran acceso directo desde cualquier parte de la aplicación.

ConstraintLayout

El uso de este tipo de layouts para las vistas son relativamente nuevo, fueron anunciadas en el Google I/O de 2016 y su uso ha sido muy extendido a lo largo de este 2017.
Esta gran adopción es debida en parte a su rendimiento, pero para entenderlo mejor debemos de comprender primero cómo Android dibuja las vistas. Veamos:

1. Cuando llamamos a una vista el framework de Android realiza un recorrido vertical a través del árbol de la vista para determinar cómo de largo es el viewgroup y sus hijos.

2. Después se realiza otro recorrido vertical para determinar las posiciones de los hijos de cada viewgroup (Determinar las posiciones de los elementos).

3. Por último, un último recorrido también vertical en el que se crea un objeto canvas para cada objeto del árbol en función de los tamaños y posiciones determinados en los dos primero recorridos.

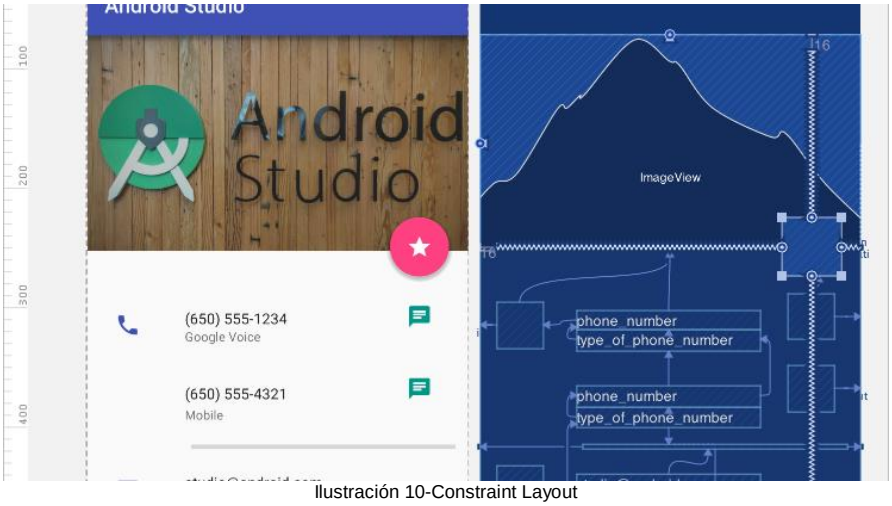


Ilustración 10-Constraint Layout

Es decir, para cada fase del proceso de pintado de la vista se requiere realizar un recorrido al árbol de la vista, por ello, cuantas más anidaciones tengas más tiempo y recursos destinará el sistema para pintarlas.

ConstraintLayout te permite construir layouts complejas sin tener que anidar vistas y elementos viewgroups. En otras palabras, permite construir layouts complejas teniendo jerarquías planas siendo así mucho más eficiente. Factor fundamental a tener en cuenta en todo dispositivo móvil. Esa esa la razón principal por la que hemos decidido incluir este elemento en nuestro proyecto.

NavigationDrawer

Se trata un patrón de navegación y concretamente, se trata de un panel deslizante que aparece desde la izquierda para ofrecer al usuario diferentes opciones de navegación. También permite el acceso al mismo tocando el icono correspondiente en la barra de acciones.

Este elemento ha ido cobrando importancia, las aplicaciones con más peso fueron adoptándolo y hoy en día se ha convertido en casi un “standard” aunque no es la mejor opción si las acciones ofrecidas en el menú son entre tres y cinco.

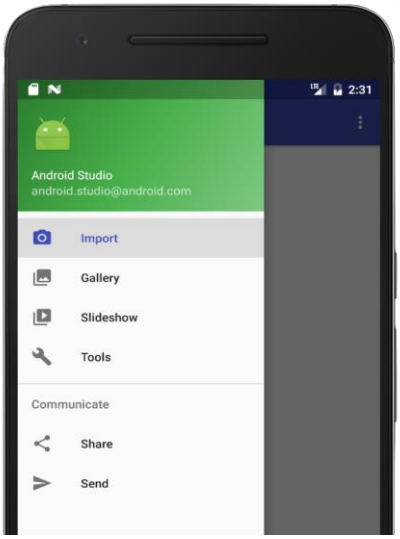


Ilustración 11- Navigation drawer

RecyclerView

Hasta hace poco era común seguir viendo ListViews implementadas en aplicaciones Android para mostrar dato en forma de lista, sin embargo, ese viewgroup ha sido sustituido casi en su totalidad por RecyclerView.

RecyclerView es una versión más flexible y avanzada de ListView su librería de soporte nos proporciona un conjunto de objetos que nos permiten diseñar e implementar interfaces dinámicas que hacen a la aplicación ser más eficientes.

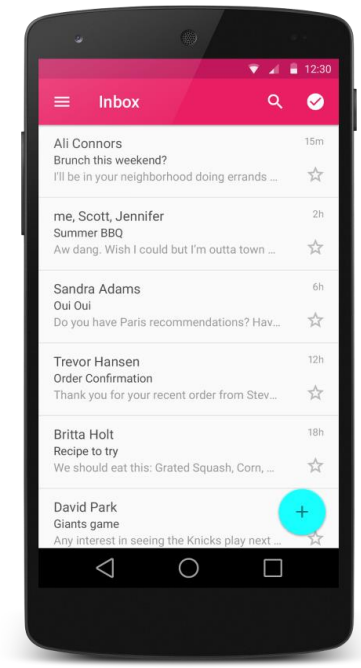


Ilustración 12- Recyclerview

En el fondo se trata de un contenedor para mostrar grandes conjuntos de datos que permite una experiencia de usuario fluida. La clave de funcionamiento de este elemento es que únicamente

permite mantener una cantidad limitada de vistas ya que el usuario solamente, puede visualizar una determinada cantidad de ellas a la vez y no tiene sentido cargar todas.

2.1.5 OTRAS TENDENCIAS EN EL DESARROLLO

App Indexing

En los últimos tiempos se ha convertido en una necesidad más que una tendencia si se quiere captar la atención de un público que cada vez tiene más opciones para escoger. Consiste en integrar la aplicación en las búsquedas de Google. Primero fue el posicionamiento SEO para webs el que sentó sólidos cimientos y mostró las bondades de éste. App indexing va más allá, no sólo posiciona sino que además si los usuarios todavía no tienen tu app permite la instalación directa desde los resultados de búsqueda de Google.

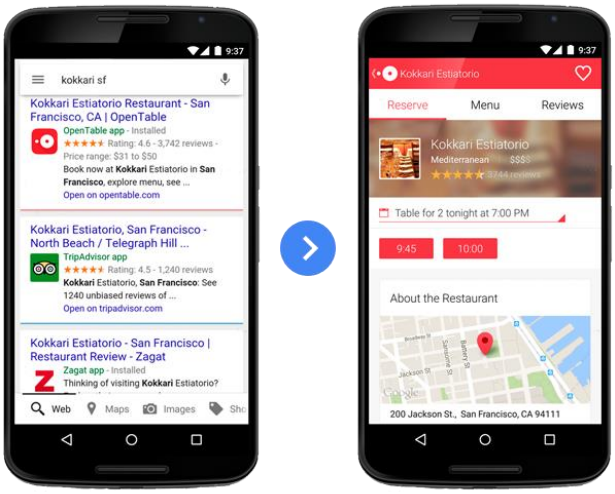


Ilustración 13- App indexing

Integración con redes sociales

Hoy en día las redes sociales, es una de las partes más importantes a tener en cuenta para los desarrolladores, pues la tendencia a permitir loguearse en la aplicación a través de ellas es cada vez más común. Para los usuarios los registros son tediosos y facilitar información personal nunca es plato de gusta. De esta manera, se automatiza el proceso de registro y logueo de manera automática generando una experiencia de usuario muy positiva y además de cara al usuario parece que no está facilitando demasiada información, aunque en realidad no sea así.

Por otro lado, permitir a los usuarios compartir la aplicación en sus redes siempre es un plus para atraer nuevos usuarios. Por ello, esta tendencia será implementada en nuestro producto.

Integrar herramientas Analytics

A menudo pensamos que el éxito de una aplicación depende de que el usuario decida descargarla y la use, sin embargo, no podríamos estar más equivocados. El éxito de una aplicación depende de la fidelización del usuario, de la capacidad que tengamos de retenerlo.

Analizar la interacción del usuario con nuestra aplicación permite conocer qué es lo que hace el usuario con la aplicación y puede ayudarte a tomar decisiones en base a ello.
Hay varios aspectos interesantes en los que enfocarse, algunos de ellos son:

- Qué partes de tu aplicación son más usadas y enfocarte en ellas.
- Qué partes son menos usadas y rediseñarlas o eliminarlas.
- Descubrir problemas de navegación o acceso a la información.

Y algunas de las preguntas que conviene resolver con los datos recogidos son:

- ¿Se registra y no vuelve a usar la aplicación?

- ¿Cuántas sesiones abre y cada cuánto tiempo?
- ¿Qué acciones se producen en nuestra app después del envío de una notificación?
- ¿Qué tipo de usuarios están usando la aplicación?

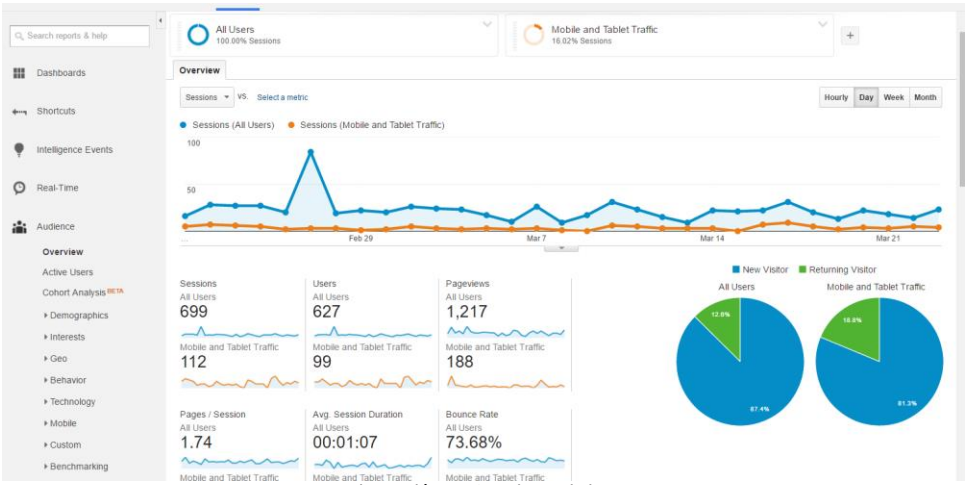


Ilustración 14-Google Analytics

Por ello, tener integrado en la aplicación una herramienta de Analytics es fundamental para un correcto progreso de la aplicación más allá de la versión inicial. Algunas de las más usadas actualmente son:

- Google Analytics para aplicaciones móviles.
- Mixpanel.
- Amplitude.
- Flurry analytics.

Para nuestro proyecto elegiremos Google Analytics ya que ofrece integración con Firebase y además la aplicación para gestionar toda esta información es bastante consistente y no nos veremos limitados a un cliente web.

2.2 ESTADO DEL ARTE

Hoy en día, hay en el mercado aplicaciones Android que cumplen una función similar a la que se va a implementar a nuestro sistema y que procederemos a analizar en esta sección.

Fintonic [1]

Se trata quizás de la más conocida de toda la lista, ésta fue galardonada por Google en 2015 con el premio a la mejor aplicación de finanzas de los premios Start Ups Mobile Innovation.
Consiste en facilitar el seguimiento de los gastos e ingresos.

[1] https://www.fintonic.com/es-ES/inicio/

Ilustración 15- Fintonic

Hoy en día es el modelo de funcionamiento y monetización a seguir, pues, aunque su uso es gratuito, ofrece también servicios de seguros y financiación.
Integra funcionalidades realmente interesantes cómo "FinScore", que te ayuda a conocer el sentimiento o consideración que tienen sobre ti los bancos y entidades financieras en función de tus balances económicos y otros factores similares.

Su gran punto a favor es que permite la integración los pagos derivados de cualquiera de tus tarjetas o cuentas bancarias.

Aunque se trata sin duda de la más completa, sus numerosos servicios y features pueden espantar a un usuario final que simplemente busque sencillez para llevar a cabo la tarea y que por supuesto depende la conexión a internet para su funcionamiento.

Mooverang[2]

Aunque en el momento de escribir esta memoria, la aplicación y su servicio web ha puesto fin a su actividad profesional, me gustaría destacar la misma ya que contaba con cerca de 100.000 usuarios al momento de su cierre y ofrecía una funcionalidad prácticamente igual a la ofrecida por este proyecto.

Es un caso curioso, pues ésta fue lanzada por nada más ni nada menos que la OCU, la Organización de Consumidores y Usuarios, con el objetivo de ayudar a las personas a ahorrar.

Mint[3]

Se trata de una solución muy completa junto a Fintonic, que también permite integración de pagos con la mayoría de entidades financieras de EEUU.
Cuenta con casi 20 millones de usuarios en todo el mundo y tiene una ventaja competitiva importante, Mint si tiene soporte sin conexión.

[2] http://www.eleconomista.es/banca-finanzas/noticias/5774520/05/14/La-OCU-lanza-una-app-para-que-los-consumidores-gestionen-mejor-sus-economias.html
[3] https://www.mint.com/how-mint-works

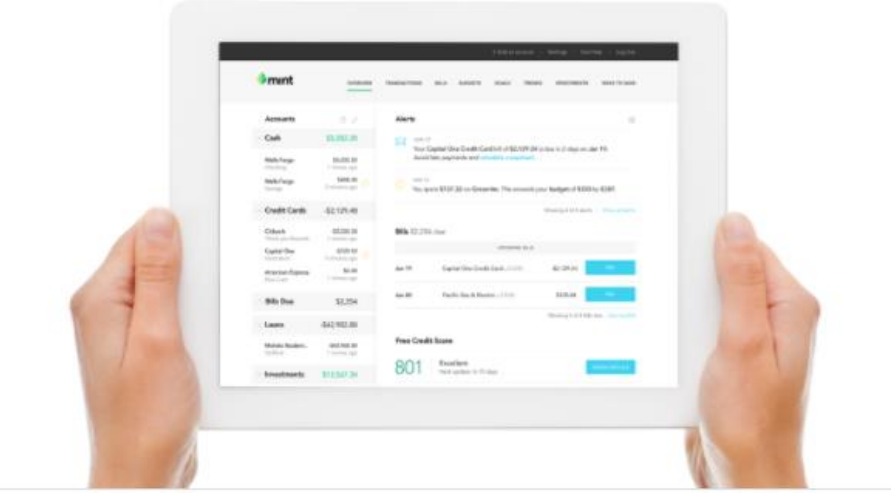


Ilustración 16- Mint

Otro gran punto a favor de esta aplicación, es su diseño adaptado a tabletas, realmente el diseño es muy limpio y la experiencia de usuario es realmente buena.

Dos características que también son dignas de destacar de esta aplicación es la posibildiad de incluir pagos futuros con sus correspondientes avisos y la clasificación de gastos de manera automática.

Monefy [4]

Dejando atrás las “grandes” soluciones ofrecidas a nivel global para la plataforma, encontramos Monefy, una solución rápida y sencilla para el registro de nuestros gastos.

Ofrece soporte en diferentes dispositivos de la plataforma Android gracias a la sincronización con Dropbox, en nuestro caso, esa sincronización es directa gracias a Firebase.

Ilustración 17- Monefy

Representa una ventaja competitiva para todos aquellos usuarios recelosos a incluir su información bancaria en una aplicación de este tipo. En este caso, se trata de una preocupación compartida y

[4] http://www.monefy.me/

creo que, con fundamentos. La solución es permitir añadir nuestros propios gastos de manera rápida e intuitiva sin necesidad de configuraciones previas en categorías previamente predeterminadas que permiten el posterior análisis de la información que se sirve al usuario.

Wally [5]

Es similar a Monefy pero intenta destacar entre las demás ofreciendo una atractiva y cuidada interfaz de usuario. Respecto a nuestra aplicación, ofrece una funcionalidad prácticamente igual y aunque parece que no podría completir con Fintonic o Mint, es la aplicación de finanzas número 1 en 22 países y top 10 en 52 países. Números increíbles que demuestran una vez más que acompañar la funcionalidad sencilla. con una atractiva funcionalidad siempre es una buena opción.

Cómo característica a destacar de esta aplicación, es la posibilidad de añadir una foto de los recibos y como elemento negativo la falta de sincronización entre diferentes dispositivos y entre plataformas.

Esa precisamente es una de las preocupaciones y consideraciones que se tuvieron en cuenta a la hora de desarrollar este proyecto.

Ilustración 18- Wally

Un detalle de la interfaz realmente destacable es el diferente tamaño que muestran en pantalla las entradas de la lista en función de la cantidad, permite de esta manera reconocer rápidamente cuales han sido los mayores gastos de un simple vistazo sin tener que mirar en las gráficas.

[5] http://wally.me/

CAPÍTULO 3

GESTIÓN DEL PROYECTO

3.1 METODOLOGÍA

Para el desarrollo de este proyecto se establecido una metodología basada en modelos incrementales.

Este modelo combina elementos del modelo en cascada, pero, aplicando la filosofía de interacción utilizada en la construcción de prototipos. El concepto es realizar una primera interacción siguiendo una secuencia lógica de análisis, diseño, implementación, testing, despliegue y evaluación que nos proporcione un producto final funcional pero muy básico, un prototipo.

Posteriormente se sigue realizando incrementos sobre el prototipo inicial, siguiendo las fases anteriormente mencionadas una a una sin pasar a la siguiente hasta haber acabado la anterior, obteniendo como resultado de cada interacción una versión mejorada del prototipo final anterior.

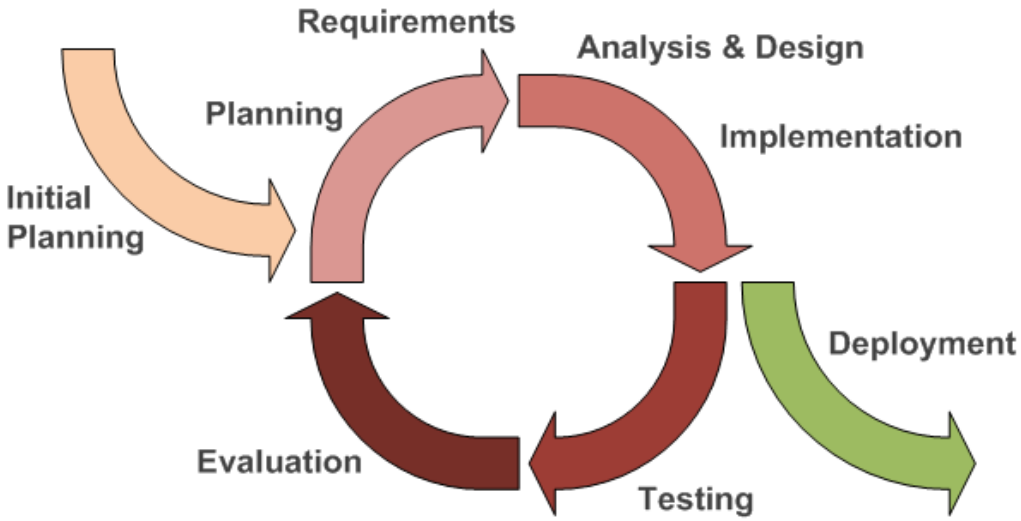


Ilustración 19- Metodología

¿Por qué se ha elegido esta metodología de desarrollo?

- Permite una fácil gestión y distribución de las tareas en cada interacción.
- Permite materializar cambios en poco tiempo.
- Es un modelo que permite adaptarse a los cambios.
- Cada elemento se construye sobre aquél que ya ha sido probado, disminuyendo riesgos.
- El tiempo de desarrollo inicial, que suele ser largo en otras metodologías, en ésta se ve claramente reducido.
- Ideal para proyectos con recursos humanos reducidos.

Riesgos y desventajas

- Es un modelo con el que adquieres experiencia sobre el sistema con cada interacción, pero requiere experiencia previa para la primera interacción.

- Requiere de una buena evaluación para priorizar las interacciones.
- Difícil de evaluar el coste total.
- Requiere de mucha gestión y planificación extra si hay muchas personas involucradas en el proyecto.

FASE DE ANÁLISIS

Durante esta fase se realiza un análisis de los requisitos que son necesarios para el desarrollo del proyecto en el análisis inicial, y de los requisitos necesarios en para la implementación de las nuevas funcionalidades en interacciones posteriores.

Esta fase se subdivide en las siguientes tareas:

- Identificar los actores implicados.
- Análisis de los requisitos de usuario.
- Análisis de los requisitos funcionales.
- Análisis de los requisitos no funcionales.
- Análisis de los requisitos de información.

FASE DE DISEÑO

En esta fase se especifica y/o revisan los diseños de la arquitectura, lógico y físico de manera que éstos permitan implementar correctamente los requisitos analizados en la fase anterior.

FASE DE IMPLEMENTACIÓN

Esta fase recoge todos aquellos procesos necesarios para la implementación de los nuevos requisitos en el sistema, escribir el código, depuración, control de versiones... etc.

FASE DE TESTING

Esta fase recoge todos los procesos que tienen relación con el testeo de las nuevas modificaciones con el objetivo de detectar fallos en tiempo de producción y ser solucionados antes de ser desplegado el producto final como una nueva release.

FASE DE EVALUACIÓN

Es una de las más cortas, pero también una de las más importantes, ésta consiste en ver si realmente se están cumpliendo los objetivos marcados, si el sistema tiene la calidad esperada, marcar los próximos objetivos a implementar y pasos a seguir… etc.
En esta fase cobran importancia las métricas de éxito, los requisitos de negocio y las reglas de negocio.

3.2 PLANIFICACIÓN

La planificación inicial se realizó fijando el inicio del proyecto en octubre, debido a que por motivos laborales fue imposible comenzarlo en septiembre. Por ello, se estimó una primera interacción que duraría un mes, en ésta tendría un mayor peso la investigación y la fase de elicitación tras la que se construiría un prototipo muy básico que apenas permitiría la integración con Cloud Store y la autenticación de usuario para acceder a la aplicación.

El resto de interacciones se estimaron de 20 días cada una aproximadamente hasta un máximo de cinco interacciones para no alargar el proyecto demasiado, en las cuales, la aplicación iría recibiendo funcionalidades al final de cada una.

La estimación aproximada para la finalización del proyecto era el 10 de febrero y si bien se han cumplido los plazos, la documentación del proyecto quedaba fuera de las estimaciones temporales.

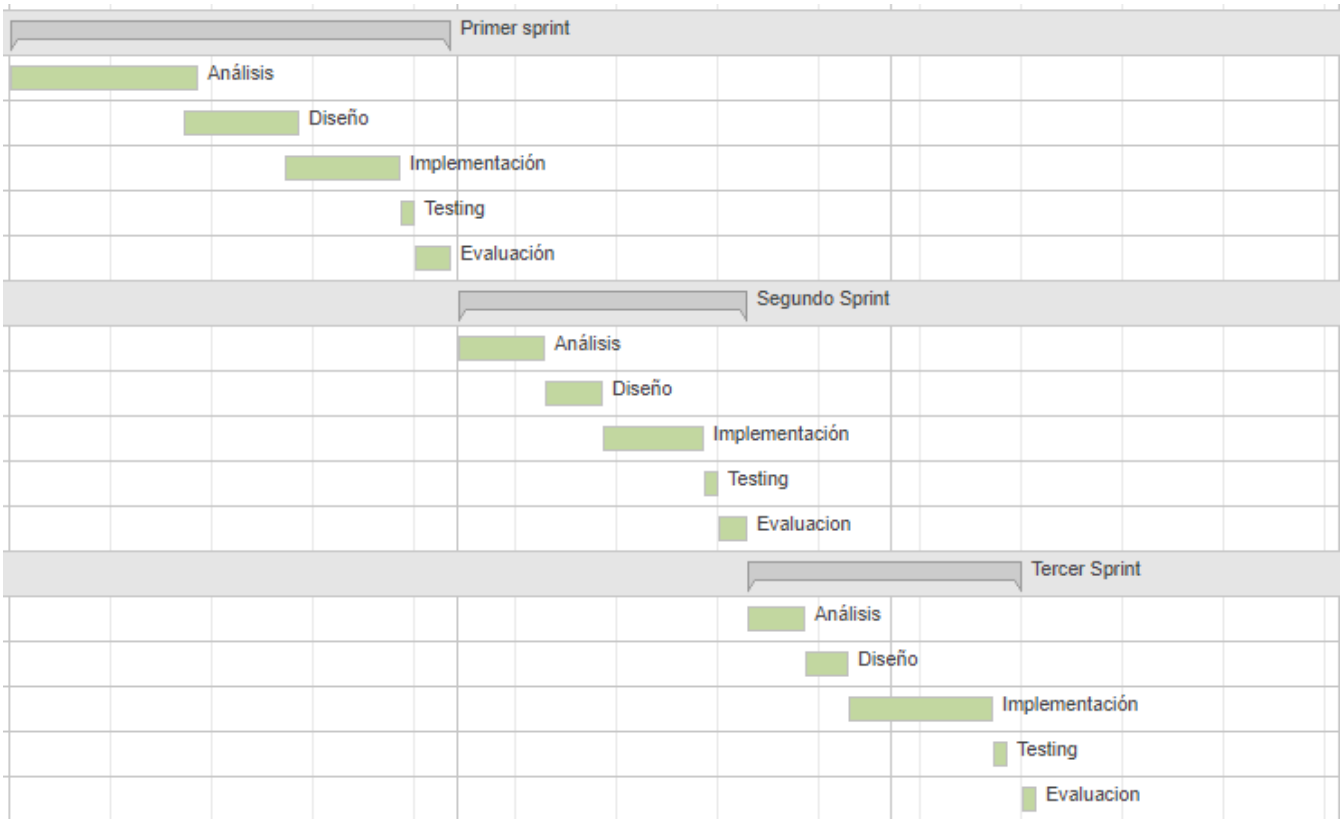
A veces, trabajar sólo en un proyecto como este en el que tienes que adoptar diferentes roles, realizar por ti mismo diferentes fases del ciclo de desarrollo puede ser un auténtico caos, por ello, se utilizó una herramienta de planificación que sirviese como guía, mostrando los objetivos temporales a cumplir, el orden establecido y los deadlines. Veamos:

	Primer sprint	**10/01/17**	**10/31/17**
	Análisis	10/01/17	10/13/17
	Diseño	10/13/17	10/20/17
	Implementación	10/20/17	10/27/17
	Testing	10/28/17	10/28/17
	Evaluación	10/29/17	10/31/17
	Segundo Sprint	**11/01/17**	**11/20/17**
	Análisis	11/01/17	11/06/17
	Diseño	11/07/17	11/10/17
	Implementación	11/11/17	11/17/17
	Testing	11/18/17	11/18/17
	Evaluacion	11/19/17	11/20/17
	Tercer Sprint	**11/21/17**	**12/10/17**
	Análisis	11/21/17	11/24/17
	Diseño	11/25/17	11/27/17
	Implementación	11/28/17	12/07/17
	Testing	12/08/17	12/08/17
	Evaluacion	12/10/17	12/10/17
	Cuarto Sprint	**12/11/17**	**12/31/17**
	Análisis	12/11/17	12/15/17
	Diseño	12/16/17	12/20/17
	Implementación	12/21/17	12/29/17
	Testing	12/30/17	12/30/17
	Evaluacion	12/31/17	12/31/17

	Quinto Sprint	**01/01/18**	**01/20/18**
	Análisis	01/01/18	01/04/18
	Diseño	01/05/18	01/08/18
	Implementación	01/09/18	01/18/18
	Testing	01/19/18	01/19/18
	Evaluacion	01/20/18	01/20/18
	Sexto Sprint	**01/21/18**	**02/10/18**
	Análisis	01/21/18	01/26/18
	Diseño	01/27/18	01/29/18
	Implementación	01/30/18	02/08/18
	Testing	02/09/18	02/09/18
	Evaluacion	02/10/18	02/10/18

Ilustración 20- Sprints

Para ilustrar esta planificación temporal de una manera más gráfica y sencilla hemos realizado un diagrama de Gantt.



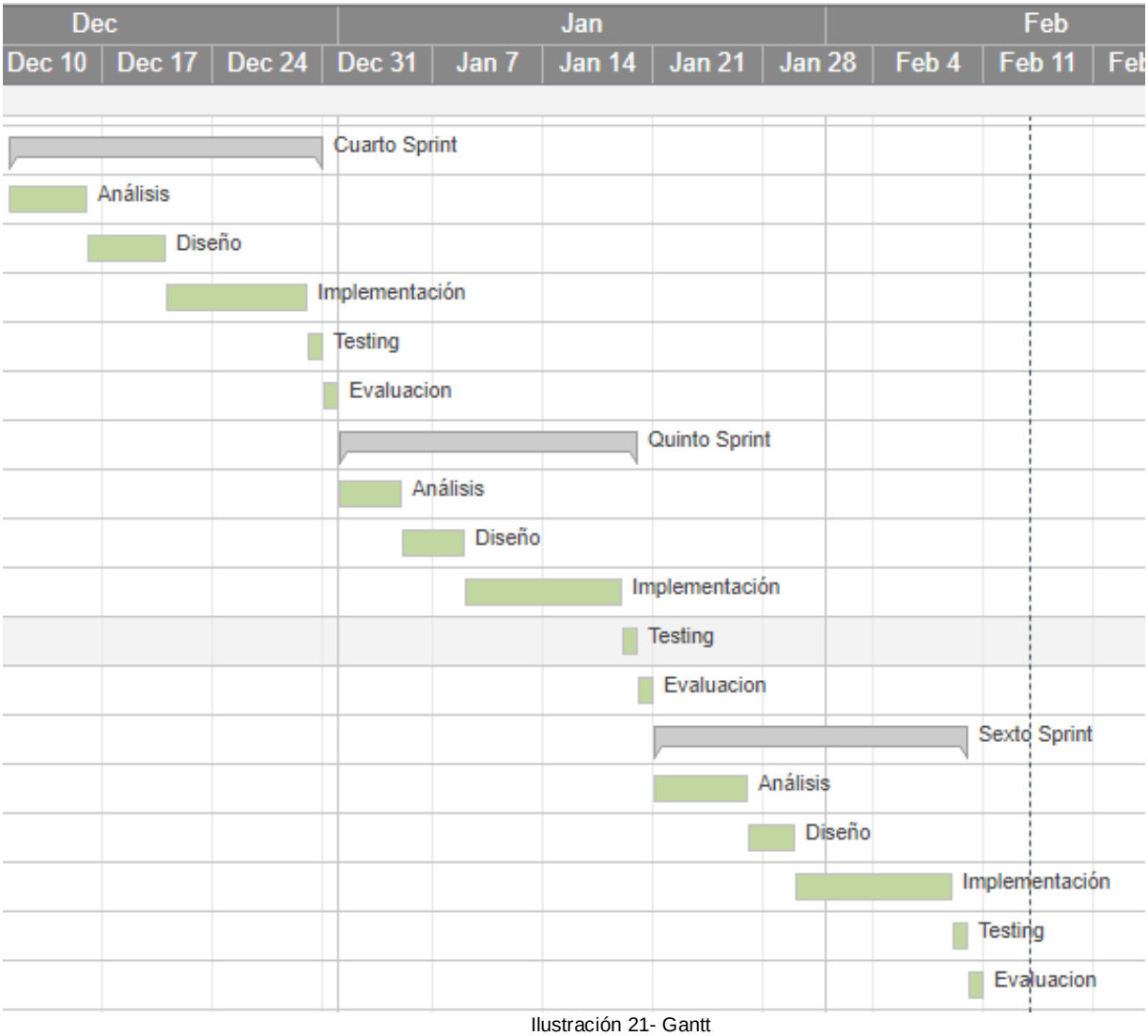


Ilustración 21- Gantt

Como podemos observar, en los primeros sprints el tiempo dedicado a las fases de análisis, diseño y evaluación es mayor. Esto encuentra su explicación lógica en el aprendizaje adquirido con cada interacción inherente a este tipo de metodología de desarrollo, además, en la medida que avanzaba el proyecto el sistema se encontraba más definido y solamente se tenía que analizar y diseñar las nuevas funcionalidades a implementar en ese sprint.

Por otro lado, el tiempo empleado en la fase de desarrollo aumenta cuando el proyecto está más avanzado. Esto se debe a que las funcionalidades o requerimientos a cumplir eran cada vez más complejos al tratarse de un modelo evolutivo en el que las funcionalidades más básicas se implementan primero y las más complejas se implementan en las últimas fases del proyecto, esto requiere un mayor tiempo de desarrollo.

Finalmente observamos que el tiempo dedicado al testing es más o menos constante a lo largo del proyecto y no representa un gran porcentaje de las horas.

En la siguiente tabla se muestran las horas que han sido invertidas cada día en cada uno de los sprints realizados durante el desarrollo.

Periodo	*Días*	*Horas / Días*	*Horas totales*
Primer sprint	31	3	93
Segundo sprint	20	3	60
Tercer sprint	21	3	63
Cuarto sprint	21	3	63
Quinto sprint	20	3	60
Sexto sprint	21	3	63
			Total: 402 horas

Tabla 1 - Horas por sprint

3.3 PRESUPUESTO BASADO EN LA PLANIFICACIÓN

Este apartado consta de varias partes claramente diferenciadas. En primer lugar, se va analizar los costes derivados de los componentes de software y hardware involucrados en el proyecto; Posteriormente se realizará una estimación mediante puntos de función para estimar el coste temporal del desarrollo del proyecto que contrastaremos con nuestra planificación y que usaremos junto a una tabla de coste por rol de cada recurso humano para estimar un coste económico. La tabla de coste/rol será un reflejo de cuáles son los sueldos del sector en este momento.

3.3.1 Coste de los recursos humanos

Pese a que el proyecto ha sido desarrollado íntegramente por una sola persona, ésta ha desempeñado diferentes roles, por ello, para dotar al presupuesto de una mayor precisión se diferencian éstos, tanto en horas cómo en coste.

Por un lado, el rol de analista ha sido desempañado para la fase de elicitación y diseño de las arquitecturas. La aplicación móvil cuenta con numerosas animaciones, elementos gráficos externos, paleta de color personalizada y otros elementos de la interfaz que han sido diseñados fuera del entorno de desarrollo Android, para ello, se ha asumido el rol de diseñador UI.

Además, el rol de desarrollo para la fase de implementación, testing y evaluación, aunque ésta última bien pudiera ser más apropiado haber sido realizada por un Product Owner.

ROL	**Coste / Mes**	**Sueldo / Hora**
Analista	4000 €	23,8 € / hora
Desarrollador	3400 €	20,23 € / hora
Diseñador UI	3500 €	20,83 € / hora
Project Manager	4300€	25,59 € / hora

Tabla 2 - Coste por roles

En primer lugar, se fija el coste que supone un analista, un desarrollador y un diseñador. No sólo los sueldos que cobran, sino el gasto que implica mantenerlos en plantilla. De ahí se extrae cuánto deberíamos cobrar por hora, contando con una jornada estándar de 8h-21días para sacar el coste por hora trabajada.

Una vez sabemos cuánto cuesta la hora trabajada de cada recurso humano, multiplicamos por las horas que se estima que han trabajado cada uno en nuestro proyecto (recordamos que, la estimación temporal, eran 402 horas) y obtendremos una estimación presupuestaria aproximada del coste de nuestros recursos humanos en función de nuestra planificación temporal.

Se han obtenido que trabajando 3h / día cada día (la jornada es tan reducida ya que es un proyecto realizado fuera de horario laboral y fin de semanas), durante el tiempo que dura la estimación temporal (en total 402 horas), asumiendo los diferentes roles, el total del coste de los recursos humanos según nuestra planificación temporal sería:

ROL	Coste / Mes	Sueldo / Hora	Horas de proyecto	Total
Analista	4000 €	23,8 € / hora	105	2499 € / proyecto
Desarrollador	3400 €	20,23 € / hora	202	4086,46 € / proyecto
Diseñador UI	3500 €	20,83 € / hora	65	1353,95 € / proyecto
Project Manager	4300 €	25,59 € / hora	30	767,7 € / proyecto
			Total:	**8707,11 € / proyecto**

Tabla 3 - Coste por roles

Como consideramos que ésta no es todo lo representativa que se espera para un proyecto de esta magnitud en la que hay que tener en cuenta otros muchos factores, posteriormente se va a realizar una estimación temporal mediante puntos de función y aplicaremos nuevamente esta fórmula.

3.3.2 Coste de los componentes hardware

Para calcular los costes de los componentes de hardware involucrados en este proyecto se va a tener en cuenta el coste total de los mismos, el porcentaje que representa el uso durante este proyecto respecto a la totalidad de su vida útil y de esa manera estimaremos el coste que supone su utilización durante el desarrollo del proyecto.

Elemento	Coste unitario	Vida útil media	% de uso en el proyecto	Coste real
Ordenador portátil	800€	5 años	8.33% (5 meses de uso frente a 60 de vida útil)	66.64 €
Disco duro externo SSD 512GB	120€	7 años	5.95%	7.14 €
Infraestructura de Firebase	0€			

Conexión a Internet	30€ / mes	5 meses	100%	150 €
			Total:	**223,78 €**

Tabla 4 - Coste componentes hardware

De manera que los costes de hardware durante el proyecto serían:

(**66.64 €** ordenador + **7.14 €** disco duro) x persona (**1** en este caso)

+ **150 €** Conexión a internet (1 para todas las personas)

223,78 € en total

Lo que se quiere reflejar es que, aunque en este caso solamente una persona (aunque con diferentes roles) está involucrada en el proyecto, al hacer éste cálculo con los componentes hardware, se tiene que tener en cuenta también éste factor, ya que cada uno de las personas necesitarían ese material hardware mientras que la conexión sería compartida.

3.3.3 Coste de los componentes software

Para realizar el cálculo de los costes de los componentes de software empleados durante el desarrollo de éste proyecto se seguirá el mismo proceso que para los componentes de hardware, con la diferencia de que en los de software las licencias de uso no las consideramos individuales sino licencias de empresa u organización, por lo que no habría que tener en cuenta el factor del número de personas.

Elemento	**Coste unitario**	**Vida útil media**	**% de uso en el proyecto**	**Coste real**
Android Studio	0€	Ilimitado		
Windows 8	130€	Ilimitado	100%	130 €
Adobe Photoshop	24,90€ / mes	5 meses	100%	124,5€
Adobe After Effects	24,90€ / mes	5 meses	100%	124,5 €
			Total:	**379 €**

Tabla 5 - Coste componentes software

Por tanto, el coste aproximado total sería:

Costes componentes de **software** + coste de **hardware** + coste **recursos humanos**

379 € + **223,78 €** + **8707,11 €** = **9309,89 € coste total del proyecto** (402 horas estimadas)

Esta cantidad la compararemos en el siguiente apartado.

3.4 PRESUPUESTO BASADO EN LA ESTIMACIÓN MEDIANTE PUNTOS DE FUNCIÓN

3.4.1 Estimación mediante puntos de función

La estimación mediante puntos de función consiste en el cálculo del coste de un proyecto software a través de la evaluación de todas las funciones del sistema. Aunque, nuestro sistema está formado por un conjunto de una herramienta de administración y una aplicación Android, estimaremos solamente las funciones de la aplicación Android ya que toda la implementación para dar funcionalidad a la herramienta de administración, se ha realizado sobre la aplicación Android en forma de integración de los diferentes servicios ofrecidos por Google Firebase.

Cada función del sistema tendrá asociado un determinado número de puntos de función, cuanto más costosa sea de implementar la funcionalidad, más puntos de función tendrá.

El proceso a seguir es el siguiente:

1. Calcular los puntos de función no ajustados.
2. Calcular los factores de ajuste.
3. Calcular los puntos de función ajustados.

Para calcular de los puntos de función no ajustados se definirán cinco grupos, cada una de las cuales tendrá un peso diferente que veremos después y se clasificaran nuestras funciones. Las categorías son las siguientes:

- Entradas de usuario: Son todas aquellas funciones en las que el usuario introduce datos en el sistema. En nuestro caso serán todas aquellas actividades o fragmento que hagan uso de elementos EditText.
- Salidas de usuario: Son todas aquellas funciones en las que se muestran datos por pantalla a los usuarios. En nuestro caso todas aquellas en las que se haga uso de elementos Textview.
- Consultas externas: Se definen como tal, aquellas funciones que realicen intercambio de información con sistemas externos. En nuestro caso, todas aquellas que realicen peticiones a algún servicio de Firebase que no sea Cloud Firestore (Es un base de datos y se considera fichero lógico externo).

- Ficheros lógicos internos: Son todas aquellas funciones que hacen uso de almacenamiento interno.

- Ficheros lógicos externos: Aquellas funciones que hacen uso de sistemas de almacenamiento externos. En nuestro caso todas aquellas que hacen uso de Cloud Firestore.

TIPO DE FUNCIÓN	**COMPLEJIDAD**	**PESO**
Entradas de usuario	Baja	3
	Media	4
	Alta	6
Salidas de usuario	Baja	3
	Media	4
	Alta	6
Consultas de usuario	Baja	4
	Media	5
	Alta	7
Ficheros internos	Baja	7
	Media	10
	Alta	15
Ficheros externos	Baja	5
	Media	7
	Alta	10

Tabla 6 - Ponderación puntos de función

A continuación, nuestros PFNA:

GRUPO	COMPONENTE	COMPLEJIDAD
Entradas	Formulario de registro	Media
	Formulario de inicio de sesión	Media
	Formulario añadir movimiento	Media
	Formulario editar movimiento	Media
Salidas	Pantalla de movimientos	Alta
	Pantalla de perfil	Baja
	Pantalla de estadísticas	Alta
	Pantalla de detalles	Media
	Pantalla de edición	Media
	Pantalla categorías	Media
Consultas externas	Autenticación	Media
	Movimientos	Alta
	Datos estadísticos	Alta
	Autenticación de Facebook	Media
Ficheros internos	Caché de la aplicación	Media
Ficheros externos	Cloud Firestore	Media
	Servicio de notificaciones	Alta
	Servicios de Google Analytics	Baja

Tabla 7 - Datos de puntos de función

A modo de resumen, para facilitar la visualización del cálculo de pesos se facilitan las siguientes tablas:

Tipo	***nº de entradas de complejidad baja***	***nº de entradas de complejidad media***	***nº de entradas de complejidad alta***
Entradas	x0	x4	x0
Salidas	x1	x3	x2
Consultas externas	x0	x2	x2
Ficheros internos	x0	x1	x0
Ficheros externos	x1	x1	x1

Tabla 8 - Numero de datos por tipo

Tipo	***Peso de la complejidad baja x nº de entradas***	***Peso de la complejidad media x nº de entradas***	***Peso de la complejidad media x nº de entradas***
Entradas	3x0	4x4	6x0
Salidas	3x1	4x3	6x2
Consultas externas	4x0	5x2	7x2
Ficheros internos	7x0	10x1	15x0
Ficheros externos	5x1	7x1	10x1
	8 pfna	55 pfna	36 pfna
		Total:	**99 PFNA**

Tabla 9 - Total puntos de función

Una vez obtenido el número de puntos de función no ajustados, deberemos obtener el factor de ajuste (FA), este es la suma de los pesos de los 14 factores de complejidad (FC) que deberemos ponderar entre 1 y 5.

Algunos factores de complejidad pueden resultar ambiguos, por ello se van a describir brevemente los mismos:

- Comunicación de datos: Cuanta mayor sea la presencia de comunicación de datos presente en el proyecto, más alta deberá ser la ponderación de este factor de ajuste.

- Procesamiento distribuido: Si hay presente gran cantidad de procesos distribuidos, el factor será próximo a cinco.

- Rendimiento: Será más alto cuando los requisitos de rendimiento sean altos. Velocidad de respuesta, velocidad de carga, procesos batch por segundo…etc.

- Carga de trabajo: Se refiere a la intensidad de la carga de trabajo que tendrá que soportar el sistema. Concurrencia.

- Frecuencia de transacciones
- Entrada de datos online: Hace referencia a la cantidad de datos que son registrados online respecto a la totalidad.
- Manejo del usuario final: Si es un sistema que requiere de alta capacitación tendrá una ponderación superior.
- Actualizaciones online: Hace referencia a la cantidad de ficheros que son actualizados cuando se realizan entradas de datos online o consultas.
- Procesos complejos: Referencia a la complejidad de los procesos del sistema.
- Utilización de otros sistemas: Cuantos más sistemas externos se empleen más alta será la ponderación.
- Facilidad de mantenimiento.
- Facilidad de operación: Cómo de sencilla resulta la instalación y conversión.
- Instalación múltiple: Referencia si es posible la utilización multidispositivo de la herramienta o es posible operar con ella en diferentes entornos. Cuanto más distribuida sea, más alta la ponderación.
- Facilidad ante cambios: Referencia tanto la escalabilidad como capacidad de ser modificada la herramienta.

Factor de complejidad	***Ponderación***
Comunicación de datos	5
Procesamiento distribuido	3
Rendimiento	3
Carga de trabajo	2
Frecuencia de las transacciones	4
Entrada de datos online	5
Manejo del usuario final	2
Actualizaciones online	4
Procesos complejos	3
Utilización de otros sistemas o servicios	4
Facilidad de mantenimiento	4
Facilidad de operación	3
Instalación múltiple	3
Facilidad ante cambios	3
	Total:44

Tabla 10 - Factor de ajuste

El factor de ajuste que buscábamos es obtenido aplicando la siguiente fórmula en la que interviene directamente el factor de complejidad que se acaba de calcular:

$$\mathbf{FA} = 0.65 + (0.1 \times \Sigma FC) = 0.65 + 0.01 \times 44 = \mathbf{1.09}$$

Por último, debemos de obtener los puntos de función ajustados utilizando los puntos de función no ajustados y el factor de ajuste, de la siguiente manera:

PFA = **PF** x **FA** = 99 x 1.09 = **107.91 Puntos de función ajustados**

Una vez tenemos ya los puntos de función ajustados de nuestro proyecto, basándonos en éstos, debemos de realizar el cálculo del esfuerzo, el cálculo de la duración del proyecto y el cálculo del presupuesto:

Aunque para determinar la equivalencia puntos de horas por puntos de función, es necesario disponer de un histórico, en nuestro caso se tomará una equivalencia estándar de 4 horas por cada punto de función.

107.91 PFA x **4** h/PFA = **431.64** horas totales de duración del proyecto

431.64 horas totales / nº de personas = **431.64** horas de esfuerzo por persona

ROL	Coste / Mes	Sueldo / Hora	Horas de proyecto	Total
Analista	4000 €	23,8 € / hora	106,64	2538 € / proyecto
Desarrollador	3400 €	20,23 € / hora	213	4308,99 € / proyecto
Diseñador UI	3500 €	20,83 € / hora	70	1458,1 € / proyecto
Project Manager	4300 €	25,59 € / hora	42	1074,78 € / proyecto
			Total:	**9379,87 € / proyecto**

Tabla 11 - Presupuesto total del proyecto

A lo que habría que sumar igualmente los costes de hardware y software:

9379,87 € + **379** € + **223.78** € **= 9982,65 €** Presupuesto total puntos de función ajustados

3.4.2 Conclusiones presupuestarias

Cómo conclusión y cierre a este capítulo, podemos decir que la estimación temporal y por tanto presupuestaria obtenida a través de la estimación de los puntos de función, se acerca más a la realidad que la planificada.

Se siguió fielmente la planificación temporal inicial, pero en la fase final las horas dedicadas cada día aumentaron ligeramente y, por otro lado, se excluyó de la misma la realización de esta memoria. Por ello, se entiende que la estimación temporal por PFA es bastante aproximada a la realidad.

CAPÍTULO 4

ANÁLISIS

4.1 ACTORES DEL SISTEMA

En el sistema se contemplan tres tipos de actores que pueden interactuar con el mismo, de tal manera que toda actividad queda limitada a estos tres tipos de actores: Usuarios no autenticados, usuarios autenticados y administradores.

ACT - 01	Usuario no autenticado
Versión	1.0 (05/02/2018)
Autor	Daniel Vallecillo Meneses
Descripción	Este actor representa a todos aquellos usuarios que usan el sistema sin haberse autenticado (Iniciado sesión).
Comentarios	Tiene acceso a las pantallas de Login y Registro de la aplicación.

Tabla 12 - ACT01

ACT - 02	Usuario autenticado
Versión	1.0 (05/02/2018)
Autor	Daniel Vallecillo Meneses
Descripción	Este actor representa a todos aquellos usuarios que usan el sistema después de haberse autenticado (haber iniciado sesión en la aplicación con sus credenciales)
Comentarios	Tiene acceso a todas las pantallas de la aplicación excepto Login y Registro.

Tabla 13 - ACT02

ACT - 03	Administrador
Versión	1.0 (05/02/2018)
Autor	Daniel Vallecillo Meneses
Descripción	Este actor representa a aquellos usuarios que tengan el rol de administrador del sistema.
Comentarios	Tiene acceso a la herramienta de administración de la aplicación en la web.

Tabla 14 - ACT03

4.2 REQUISITOS DE USUARIO

En este apartado se especificarán todos los requisitos de usuario que el sistema debe de cumplir dónde entendemos por requisito de usuario, todos los objetivos o tareas que una clase de usuario podrá realizar con el sistema. Éstos desembocan en casos de uso.

Usuarios no autenticados

RU-1. Un usuario no autenticado podrá autenticarse en el usuario haciendo uso de sus credenciales de acceso.

RU-2. Un usuario no autenticado podrá registrarse en el sistema a través de un formulario de registro.

RU-3. Un usuario no autenticado podrá recibir notificaciones push de la aplicación.

RU-4. Un usuario no autenticado podrá recibir correos electrónicos de la aplicación.

Usuarios autenticados

RU-5. Un usuario autenticado podrá cerrar sesión.

RU-6. Un usuario autenticado podrá ver su información de perfil.

RU-7. Un usuario autenticado podrá visualizar una lista de sus gastos e ingresos.

RU-8. Un usuario autenticado podrá visualizar un balance de sus gastos e ingresos.

RU-9. Un usuario autenticado podrá añadir gastos e ingresos.

RU-10. Un usuario autenticado podrá editar sus entradas de gastos o ingresos.

RU-11. Un usuario autenticado podrá eliminar sus entradas de gastos o ingresos.

RU-12. Un usuario autenticado podrá revisar en detalle cada una de sus entradas de gastos o ingresos.

RU-13. Un usuario autenticado podrá visualizar una gráfica de sus gastos en el tiempo.

Usuarios administradores

RU-14. Un usuario administrador podrá acceder a un panel de gestión de cuentas.

RU-15. Un usuario administrador podrá inhabilitar una cuenta.

RU-16. Un usuario administrador podrá habilitar una cuenta.

RU-17. Un usuario administrador podrá eliminar una cuenta.

RU-18. Un usuario administrador podrá definir plantillas para el envío de emails a los usuarios.

RU-19. Un usuario administrador podrá acceder a un panel de gestión de la base de datos.

RU-20. Un usuario administrador podrá gestionar a tiempo real la base de datos.

RU-21. Un usuario administrador podrá acceder a un panel de visualización de analíticas recogidas de los usuarios.

RU-22. Un usuario administrador podrá enviar notificaciones push a los grupos de usuarios deseados.

4.3 DIAGRAMA DE CASOS DE USO

Si bien en el apartado anterior se recogían las necesidades de los usuarios, en forma de requisitos de usuario, en este apartado vamos a ver un diagrama de casos de uso que nos muestre una visión clara de cómo los actores referidos en los apartados anteriores caracterizan las acciones que se ejecutan en el sistema.

Técnicamente, entendemos como diagrama de casos de uso al diagrama de comportamiento UML que representa los casos de uso derivados de los requisitos de usuario.

Ilustración 22- Diagrama de casos de uso

4.4 ESPECIFICACIÓN DE LOS CASOS DE USO

En este apartado se va a especificar cada uno de los casos de uso que han sido plasmados en el diagrama anterior.

CU-1: Inicio de sesión.

CU-2: Registro.

CU-3: Recibir notificación push.

CU-4: Recibir email.

CU-5: Ver información de perfil.

CU-6: Cerrar sesión.

CU-7: Ver lista de movimientos.

CU-8: Añadir movimiento.

CU-9: Editar movimiento.

CU-10: Eliminar movimiento.

CU-11: Revisar movimiento.

CU-12: Visualizar gráficos.

CU-13: Acceder a panel gestión de cuentas.

CU-14: Inhabilitar cuenta.

CU-15: Habilitar cuenta.

CU-16: Eliminar cuenta.

CU-17: Administrar plantillas de emails.

CU-18: Acceder al panel de base de datos.

CU-19: Gestionar base de datos.

CU-20: Acceder al panel de analytics.

CU-21: Visualizar gráficas analíticas.

CU-22: Enviar notificaciones push.

CU-01	**Iniciar sesión**
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario no autenticado iniciar sesión facilitando sus credenciales de acceso.
Precondiciones	No estar autenticado en el sistema y estar registrado en el sistema.
Secuencia normal	1. El usuario no autenticado abre la aplicación. 2. El usuario rellena los campos de credenciales. 3. El usuario pulsa Sign in. 4. El sistema comprueba si existe una cuenta habilitada con esos credenciales. 5. El usuario es autenticado y se actualiza al interfaz.
Secuencia alternativa	
Postcondiciones	El usuario se autentica.
Excepciones	1. Si los credenciales están vacíos se indica que el usuario o contraseña no son correctos. 2. Si los credenciales están rellenos pero no coinciden, indica error de autenticación. 3. Si no se dispone de conexión a internet, indica error de autenticación.
Frecuencia	Alta
Importancia	Alta

Tabla 15 - CU01

CU-02	**Registrarse**
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario no autenticado crear una cuenta registrando sus datos a través de un formulario.
Precondiciones	No estar autenticado en el sistema.
Secuencia normal	1. El usuario no autenticado abre la aplicación. 2. El usuario pulsa Sign up. 3. El usuario rellena los campos de registro. 4. El usuario pulsa create account. 5. El usuario crea su cuenta y se realiza inicio de sesión automática con la misma. 6. El usuario recibe en su correo un email del equipo de desarrollo de la aplicación.
Secuencia alternativa	
Postcondiciones	El usuario crea una cuenta y se autentica.
Excepciones	1. Si los campos de registro están vacíos o no cumplen las condiciones requeridas se indica. 2. Si el campo password y confirm password no coinciden, se muestra el error. 3. Si no se dispone de conexión a internet, indica error de autenticación. 4. Si el usuario pulsa el botón atrás, el sistema le devuelve a la pantalla de inicio de sesión.
Frecuencia	Baja
Importancia	Alta

Tabla 16 -CU02

CU-03	**Recibir notificaciones push**
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario no autenticado recibir notificaciones push de la aplicación.
Precondiciones	Haber descargado la aplicación y permitir las notificaciones.
Secuencia normal	1. El usuario recibe una notificación. 2. El usuario abre la notificación. 3. El usuario hace click sobre la misma. 4. El sistema abre la aplicación.
Secuencia alternativa	
Postcondiciones	Ninguna.
Excepciones	
Frecuencia	Media
Importancia	Baja

Tabla 17 - CU03

CU-04	**Recibir email**
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario no autenticado recibir emails del equipo de desarrollo de la aplicación.
Precondiciones	Haberse registrado previamente en la aplicación.
Secuencia normal	1. El usuario se registra. 2. El sistema envía un email. 3. El usuario recibe un email de confirmación.
Secuencia alternativa	
Postcondiciones	El usuario recibe un email de confirmación.
Excepciones	1. Si el email de registro cumple los patrones de verificación pero no es su cuenta de correo real no recibirá el email.
Frecuencia	Baja
Importancia	Baja

Tabla 18 - CU04

CU-05	**Ver información de perfil**
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario autenticado ver su información de perfil.
Precondiciones	Estar autenticado en el sistema.
Secuencia normal	1. El usuario accede a la pestaña profile. 2. El sistema muestra la pantalla con los datos del perfil.
Secuencia alternativa	
Postcondiciones	Ninguna.
Excepciones	
Frecuencia	Alta
Importancia	Alta

Tabla 19- CU05

CU-06	Cerrar sesión
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario autenticado cerrar su sesión activa.
Precondiciones	Estar autenticado en el sistema y encontrarse en la pestaña de perfil.
Secuencia normal	1. El usuario presiona Sign out. 2. El sistema cierra la sesión. 3. El sistema redirige al usuario a la pantalla de inicio de sesión.
Secuencia alternativa	
Postcondiciones	El usuario deja de estar autentificado.
Excepciones	
Frecuencia	Media
Importancia	Alta

Tabla 20- CU06

CU-07	Visualizar lista de movimientos
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario autenticado visualizar una lista de todos los ingresos y gastos ordenados por orden cronológico.
Precondiciones	Estar autenticado en el sistema.
Secuencia normal	1. El usuario pulsa sobre la pestaña Home. 2. El sistema muestra la pantalla con todos los movimientos.
Secuencia alternativa	
Postcondiciones	
Excepciones	
Frecuencia	Alta
Importancia	Alta

Tabla 21- CU07

CU-08	**Añadir movimiento**
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario autenticado añadir movimientos (gastos o ingresos).
Precondiciones	Estar autenticado y encontrarse en la pestaña home.
Secuencia normal	1. El usuario pulsa sobre el botón flotante de “+”. 2. El sistema muestra una pantalla de formulario. 3. El usuario introduce la información requerida. 4. El usuario pulsa “add”. 5. El sistema crea el movimiento y lo añade a lista. 6. El sistema devuelve al usuario a la pestaña de home.
Secuencia alternativa	
Postcondiciones	El movimiento se crea y se añade a la lista
Excepciones	1. El sistema es asíncrono y tiene incorporada una caché, si no hay conexión a internet se procesa, funciona y se muestra igual de cara a los usuarios, pero no se añadirá a la base de datos hasta que no haya conexión. 2. Si el usuario pulsa el botón atrás, el sistema le devuelve a la pestaña de home. 3. Si el usuario introduce información no válida en los campos el sistema muestra el aviso correspondiente.
Frecuencia	Alta
Importancia	Alta

Tabla 22- CU08

CU-09	**Editar movimiento**
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario autenticado editar sus movimientos (gastos o ingresos).
Precondiciones	Estar autenticado y encontrarse en la pantalla de detalle de un movimiento.
Secuencia normal	1. El usuario pulsa sobre el icono de editar. 2. El sistema muestra una pantalla con la información del movimiento en formato editable. 3. El usuario edita la información. 4. El usuario pulsa “end editing”.

	5. El sistema actualiza el movimiento. 6. El sistema devuelve al usuario a la pestaña de home.
Secuencia alternativa	
Postcondiciones	El movimiento se actualiza con la nueva información.
Excepciones	1. El sistema es asíncrono y tiene incorporada una caché, si no hay conexión a internet se procesa, funciona y se muestra igual de cara a los usuarios pero los cambios no se añadirán a la base de datos hasta que no haya conexión. 2. Si el usuario pulsa el botón atrás, el sistema le devuelve a la pestaña de home. 3. Si se introduce información no válida en los campos de edición el sistema muestra el error.
Frecuencia	Media
Importancia	Alta

Tabla 23- CU09

CU-10	**Eliminar movimiento**
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario autenticado eliminar sus movimientos (gastos o ingresos).
Precondiciones	Estar autenticado y encontrarse en la pestaña de home.
Secuencia normal	1. El usuario hace “long press” sobre el movimiento que desea eliminar. 2. El sistema muestra un mensaje de confirmación. 3. El usuario confirma. 4. El sistema elimina el movimiento.
Secuencia alternativa	
Postcondiciones	El movimiento es eliminado
Excepciones	1. El sistema es asíncrono y tiene incorporada una caché, si no hay conexión a internet se procesa, funciona y se muestra igual de cara a los usuarios pero el movimiento no es eliminado de la base de datos hasta que no haya conexión. 2. Si el usuario no confirma la eliminación o se pulsa fuera del dialogo, el sistema devuelve al usuario a la vista de home.
Frecuencia	Media
Importancia	Alta

Tabla 24- CU10

CU-11	**Revisar movimiento**
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario autenticado revisar sus movimientos (gastos o ingresos).
Precondiciones	Estar autenticado y encontrarse en la pestaña de home.
Secuencia normal	1. El usuario hace “click” sobre el movimiento que desea revisar. 2. El sistema muestra una pantalla con la información completa del movimiento.
Secuencia alternativa	
Postcondiciones	
Excepciones	
Frecuencia	Alta
Importancia	Alta

Tabla 25- CU11

CU-12	**Visualizar gráficos**
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario autenticado visualizar un gráfico que muestre sus gastos en el tiempo.
Precondiciones	Estar autenticado.
Secuencia normal	1. El usuario hace click sobre la pestaña de “charts” 2. El sistema sirve la vista deseada.
Secuencia alternativa	
Postcondiciones	
Excepciones	1. Si no existen movimientos que graficar se mostrará una pantalla de aviso. 2. Si no se dispone de conexión a internet y se pulsa “Tap to update” se mostrará una pantalla de aviso ya que es una condición indispensable para actualizar los mismos. Aunque el sistema de gestión de movimientos es asíncrono el de actualización de gráficas es independiente. 3. Los movimientos nuevos son añadidos automáticamente pero si se eliminar movimientos, que no es tan común, deberá actualizarse manualmente.
Frecuencia	Media
Importancia	Alta

Tabla 26- CU12

CU-13	**Acceder al panel de gestión de cuentas**
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario autenticado como administrador en la consola de Firebase acceder al panel de gestión de cuentas.
Precondiciones	Estar autenticado en Firebase como administrador del proyecto.
Secuencia normal	1. El usuario accede a Firebase. 2. El usuario hace click en “Go to console” 3. El sistema sirve la vista de la consola. 4. El usuario hace click en “Authentication” 5. El sistema sirve la vista del panel.
Secuencia alternativa	
Postcondiciones	
Excepciones	
Frecuencia	Baja
Importancia	Alta

Tabla 27- CU13

CU-14	**Inhabilitar cuenta**
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario autenticado como administrador en la consola de Firebase inhabilitar una cuenta.
Precondiciones	Estar autenticado y encontrarse en el panel de gestión de cuentas.
Secuencia normal	1. El usuario se posiciona sobre una de las cuentas de la lista. 2. El sistema muestra opciones a la derecha de la misma. 3. El usuario hace click sobre “…” y selecciona inhabilitar cuenta. 4. El sistema inhabilita la cuenta.
Secuencia alternativa	
Postcondiciones	La cuenta es inhabilitada
Excepciones	1. Si la cuenta ya está inhabilitada muestra la opción de “habilitar” en vez de “deshabilitar”.
Frecuencia	Baja
Importancia	Alta

Tabla 28- CU14

CU-15	**Habilitar cuenta**
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario autenticado como administrador en la consola de Firebase habilitar una cuenta.
Precondiciones	Estar autenticado y encontrarse en el panel de gestión de cuentas.
Secuencia normal	1. El usuario se posiciona sobre una de las cuentas de la lista inhabilitadas. 2. El sistema muestra opciones a la derecha de la misma. 3. El usuario hace click sobre "…" y selecciona habilitar cuenta. 4. El sistema habilita la cuenta de neuvo.
Secuencia alternativa	
Postcondiciones	La cuenta es habilitada
Excepciones	1. Si la cuenta ya no está inhabilitada muestra la opción de "deshabilitar" en vez de "habilitar".
Frecuencia	Baja
Importancia	Alta

Tabla 29- CU15

CU-16	**Eliminar cuenta**
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario autenticado como administrador en la consola de Firebase eliminar una cuenta.
Precondiciones	Estar autenticado y encontrarse en el panel de gestión de cuentas.
Secuencia normal	1. El usuario se posiciona sobre una de las cuentas de la lista. 2. El sistema muestra opciones a la derecha de la misma. 3. El usuario hace click sobre "…" y selecciona eliminar cuenta. 4. El sistema elimina la cuenta.
Secuencia alternativa	
Postcondiciones	La cuenta es eliminada
Excepciones	
Frecuencia	Baja
Importancia	Alta

Tabla 30- CU16

CU-17	Administrar plantillas de emails
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario autenticado como administrador en la consola de Firebase administrar las diferentes plantillas de emails disponibles.
Precondiciones	Estar autenticado y encontrarse en el panel de gestión de cuentas.
Secuencia normal	1. El usuario sobre el panel de “authentication” pulsa sobre la pestaña “plantillas”. 2. El sistema sirve la vista de las plantillas. 3. El usuario hace click sobre la plantilla que desea modificar. 4. El sistema sirve la vista de la plantilla. 5. El usuario hace click sobre el icono de editar. 6. El sistema pone la plantilla en modo edición. 7. El usuario edita los campos permitidos de la plantilla. 8. El usuario hace click sobre “guardar”. 9. El sistema guarda los cambios en la plantilla.
Secuencia alternativa	
Postcondiciones	La plantilla es modificada.
Excepciones	
Frecuencia	Baja
Importancia	Baja

Tabla 31- CU17

CU-18	Acceder al panel de gestión de la base de datos
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario autenticado como administrador en la consola de Firebase acceder al panel de gestión de la base de datos.
Precondiciones	Estar autenticado en Firebase como administrador del proyecto.
Secuencia normal	1. El usuario accede a Firebase. 2. El usuario hace click en “Go to console” 3. El sistema sirve la vista de la consola. 4. El usuario hace click en “Data base” 5. El sistema sirve la vista del panel.
Secuencia alternativa	
Postcondiciones	
Excepciones	
Frecuencia	Media
Importancia	Alta

Tabla 32-- CU18

CU-19	Gestionar base de datos
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario autenticado como administrador en la consola de Firebase gestionar la base de datos.
Precondiciones	Estar autenticado en Firebase como administrador del proyecto y encontrarse en el panel de “database”.
Secuencia normal	1. El usuario selecciona un documento o colección y hace click sobre “…” 2. El sistema muestra las opciones disponibles dependiendo de si es un documento o una colección. 3. El usuario hace click sobre la opción deseada. 4. El sistema ejecuta la acción.
Secuencia alternativa	
Postcondiciones	El documento o colección es borrado, añadido o modificado.
Excepciones	
Frecuencia	Media
Importancia	Alta

Tabla 33- CU19

CU-20	Acceder al panel de analytics
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario autenticado como administrador en la consola de Firebase acceder al panel de gestión de analytics.
Precondiciones	Estar autenticado en Firebase como administrador del proyecto.
Secuencia normal	1. El usuario accede a Firebase. 2. El usuario hace click en “Go to console” 3. El sistema sirve la vista de la consola. 4. El usuario hace click en “Dashboard” 5. El sistema sirve la vista del panel.
Secuencia alternativa	
Postcondiciones	
Excepciones	
Frecuencia	Baja
Importancia	Alta

Tabla 34- CU20

CU-21	Visualizar gráficas analíticas
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario autenticado como administrador en la consola de Firebase acceder a las gráficas analíticas generadas.
Precondiciones	Estar autenticado en Firebase como administrador del proyecto y encontrarse en la pestaña Dashboard.
Secuencia normal	1. El usuario selecciona el tipo de información que desea ver en detalle. 2. El sistema sirve la vista con la gráfica deseada.
Secuencia alternativa	
Postcondiciones	
Excepciones	
Frecuencia	Alta
Importancia	Media

Tabla 35- CU21

CU-22	Enviar notificaciones push
Versión	1.0
Autor	Daniel Vallecillo
Descripción	El sistema permitirá a un usuario autenticado como administrador en la consola de Firebase enviar notificaciones push a grupos de usuarios.
Precondiciones	Estar autenticado en Firebase como administrador del proyecto y encontrarse en la pestaña “Notifications”.
Secuencia normal	1. El usuario hace click en “mensaje nuevo”. 2. El sistema muestra una pantalla formulario con diferentes opciones. 3. El usuario configura la notificación. 4. El usuario hace click en enviar. 5. El sistema envía la notificación push al grupo de usuarios deseado.
Secuencia alternativa	
Postcondiciones	El sistema envía la notificación a los usuarios.
Excepciones	
Frecuencia	Alta
Importancia	Baja

Tabla 36- CU22

4.5 REQUISITOS NO FUNCIONALES

Un requisito funcional o atributo de calidad es la descripción de una propiedad que debe de tener el sistema o una restricción que debe de respetar. Si los funcionales indican que hace el sistema los no funcionales indican cómo de bien lo hace: Disponibilidad, usabilidad, seguridad, rendimiento, etc…

RNF-1. El sistema deberá ser escalable y soportar la concurrencia de usuarios sin importar la magnitud.

RNF-2. El sistema deberá almacenar las contraseñas cifradas.

RNF-3. El sistema no deberá el registro de más de una cuenta por correo electrónico.

RNF-4. La autenticación se realizará mediante email y contraseña.

RNF-5. La estructura del sistema deberá permitir la integración de redes sociales para la autenticación.

RNF-6. El sistema deberá tener una disponibilidad de 24 horas al día y 7 días a la semana.

RNF-7. El sistema deberá permitir una gestión asíncrona de los movimientos de gastos e ingresos independiente de internet.

RNF-8. La base de datos debe de permitir la modificación de la misma sin afectar a la lógica de negocio.

RNF-9. El sistema debe de respetar la ley orgánica de protección de datos.

RNF-10. El sistema deberá registrar y procesar toda la información de los usuarios, así como sus interacciones.

RNF-11. El sistema debe de ofrecer analíticas de la información recogida.

RNF-12. El sistema deberá poder ser administrado desde la web.

RNF-13. El sistema deberá adoptar la simbología del Euro cómo forma de representación del dinero.

RNF-14. El sistema deberá incluir obligatoriamente la lengua inglesa, y opcionalmente, aceptar también el Español.

RNF-15. El sistema deberá de ofrecer un tiempo de respuesta a las peticiones de red razonablemente bajo.

4.6 REQUISITOS DE INFORMACIÓN

Se tratan de todos aquellos requisitos que describen las restricciones del sistema en relación a los datos almacenados o procesados por el mismo.

RI-01. El sistema almacenará los datos de registro del usuario (correo, contraseña, nombre, apellidos, país y fecha de nacimiento).

RI-02. La contraseña del usuario deberá tener un mínimo de 6 caracteres.

RI-03. El email del usuario deberá de presentar un patrón ‘xxx@xxx.com’.

RI-04. El dinero será representado con números de hasta dos decimales.

RI-05. Los movimientos (ingresos o gastos), deberán almacenar la siguiente información: concepto, fecha, descripción, gasto, cantidad.

RI-06. La propiedad descripción podrá tener hasta un máximo de 250 caracteres.

RI-07. La propiedad Nombre y Apellidos deberán estar limitados a 30 y 50 caracteres respectivamente.

RI-08. El formato de todas las fechas del sistema será dd/mm/aa.

CAPÍTULO 5

DISEÑO

5.1 ARQUITECTURA LÓGICA

La arquitectura lógica es la manera en que se organizan los componentes lógicos de una solución[6], es el diseño estructural que ofrece tantos detalles como sea posible sin restringir la arquitectura a una tecnología o entorno particular, por ejemplo, un diagrama que ilustre la relación entre los componentes que conforman el sistema.

Veamos cuál es la arquitectura lógica que emplea nuestro sistema:

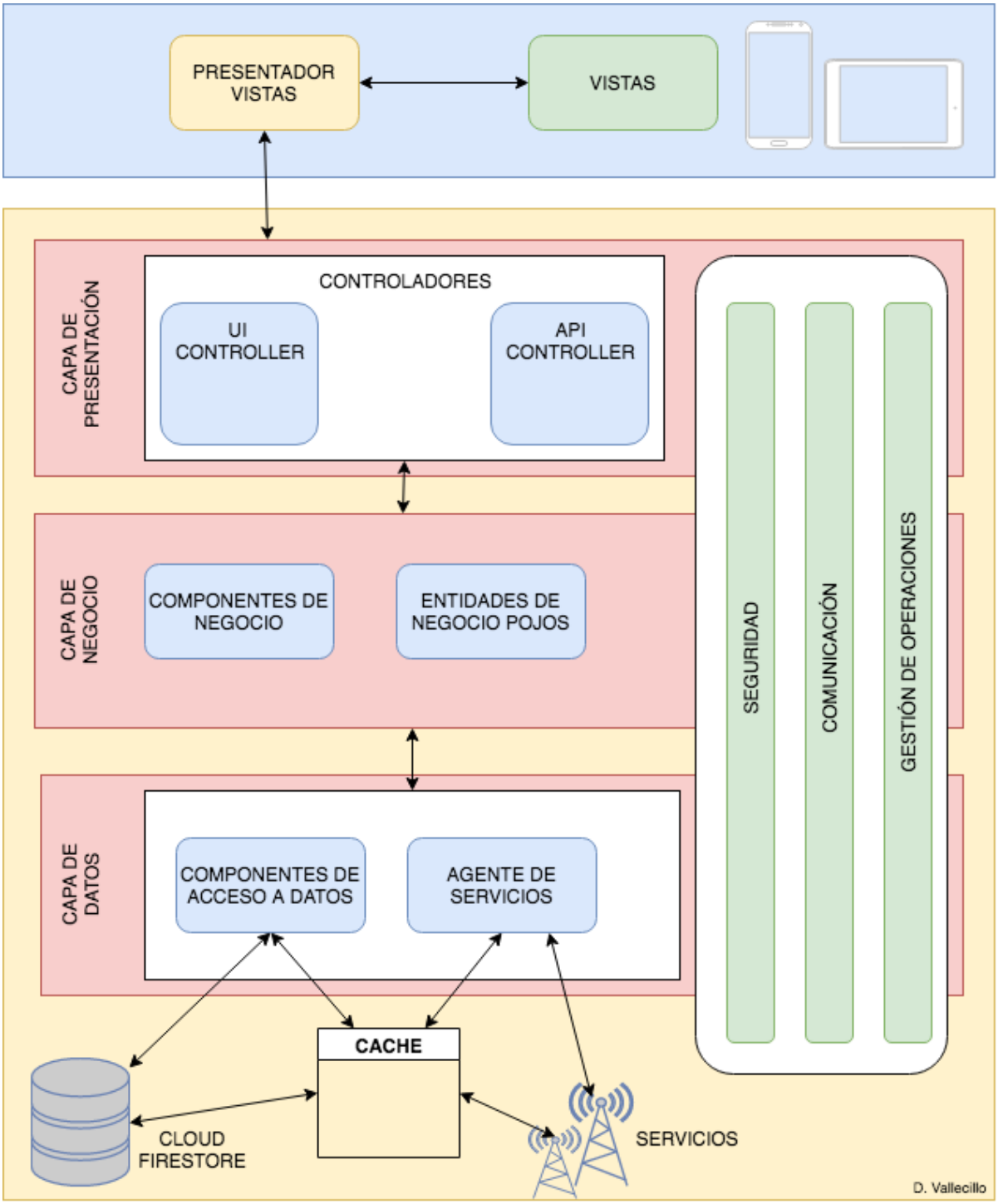


Ilustración 23- Arquitectura lógica

[6] https://technet.microsoft.com/es-es/library/cc261976(v=office.12).aspx

Esta arquitectura está formada principalmente por tres cuatro capas y la interacción entre ellas es la siguiente:

El usuario interactúa con la aplicación a través de las vistas, el presentador es el intermediario entre las vistas y el resto. De esta manera se aísla la interfaz de la lógica de negocio y permite la modificación de las vistas sin afectar al resto de la aplicación.

- La capa de presentación: Está formada por los controladores de la interfaz y de la API, que reciben peticiones del presentador de las vistas, éstos controladores interpretan que es lo que se debe servir o procesar y se lo pide a la capa de negocio.

- La capa de negocio: En ésta capa está implementada la lógica de negocio y es la que interactúa con la capa de datos, procesa los datos o realiza las operaciones una vez recibidos de la capa de datos y se lo sirve de vuelta al presentador.

- La capa de datos: Recibe peticiones de la capa de negocio y es la única que está en contacto directo con las fuentes de datos o servicios, de esta manera, el acceso a los datos está totalmente aislado ya que cualquier petición debe pasar primero por esta capa.

- Caché, fuente de datos y servicios: Reciben peticiones de la capa de datos y manda de vuelta la información.

Al final se trata de un tipo de arquitectura MVP

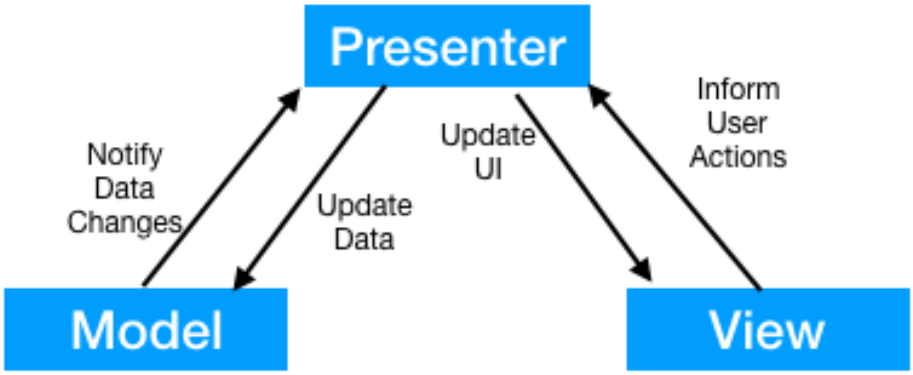


Ilustración 24- MVP

- View corresponde a nuestras vistas.

- Presenter corresponde al presentador de las vistas y los controladores.

- Model corresponde a nuestra capa de negocio y nuestra capa de datos o servicios (cualquiera de las dos acepciones es válida para referir a la capa que está en contacto con las fuentes de datos o servicios).

La anterior imagen y la descripción de la interacción, muestra cuál es la relación entre los distintos componentes involucrados en la aplicación, si bien es cierto que nuestro sistema está formado también por una herramienta de administración, ésta, no deja de ser un servicio de Google al que hacemos peticiones. Su aspecto lógico es el siguiente:

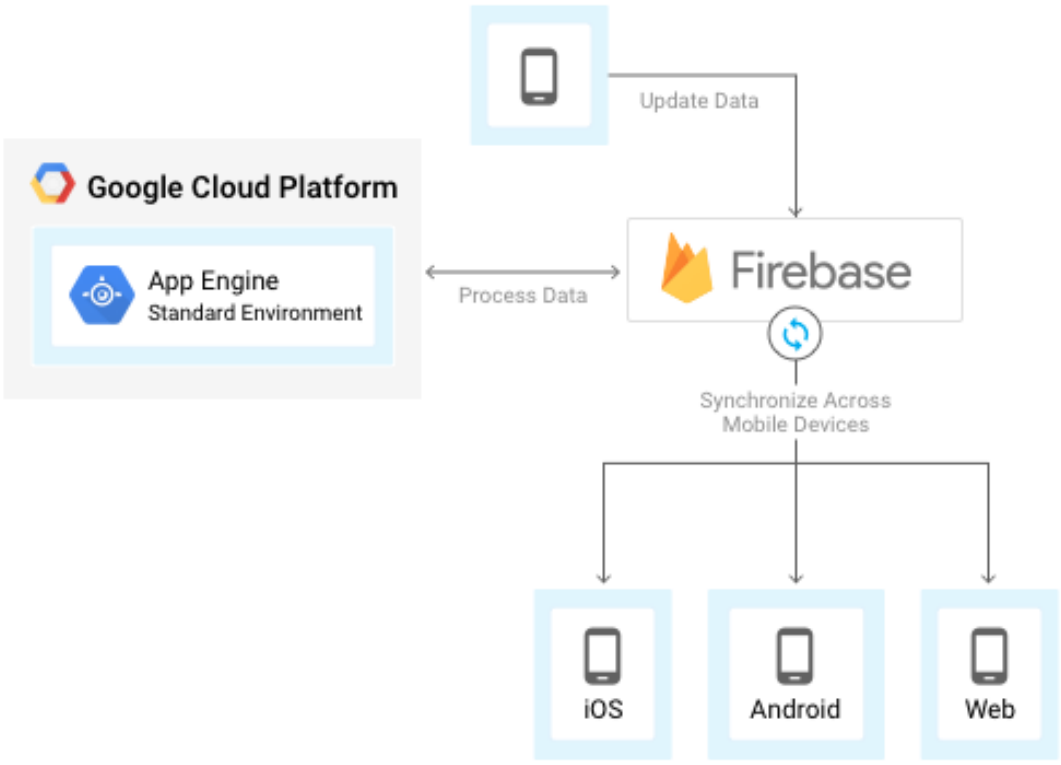


Ilustración 25 - Interacción Google Cloud
[7]

Dónde el dispositivo que se encuentra en la parte superior, es el que realiza las peticiones a través de la consola de Firebase, éste puede ser cualquier dispositivo con acceso a internet, no tiene por qué ser un ordenador.

5.2 ARQUITECTURA FÍSICA

La arquitectura física es la manera en la que está implementada la arquitectura de una tecnología, por ejemplo, la especificación de los servicios y diferentes componentes de un sistema. Podríamos decir que es una solución concreta para acomodar la arquitectura lógica y satisfacer las necesidades del sistema. Ésta se implementa a través de elementos físicos que proporcionan la solución al producto o al servicio.

La arquitectura física de nuestro sistema es un sencillo cliente-servidor que no necesita mayor explicación. Como detalle, destacar que además de los servicios proporcionados por Google Cloud Platform, también consume servicios de Facebook al estar implementada la autenticación con Facebook en la aplicación.

[7] https://cloud.google.com/solutions/mobile/mobile-firebase-app-engine-flexible

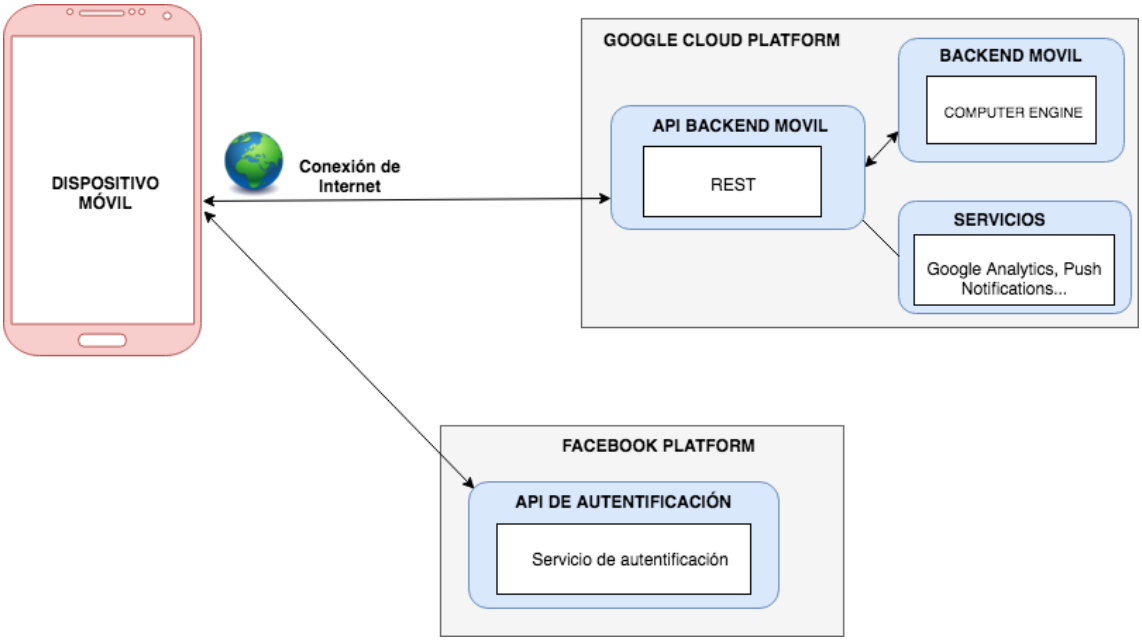


Ilustración 26 - Arquitectura Física

5.3 DIAGRAMA DE CLASES

Se trata de un tipo de diagrama que describe la estructura de un sistema, mostrando las clases, atributos, métodos y la relación entre ellos. Para este fin se va a mostrar una versión simplificada del mismo en el que simplemente los atributos y métodos (al ser numerosos en cada clase hacen ilegible el diagrama) aparecen ocultos para una correcta visualización del mismo.

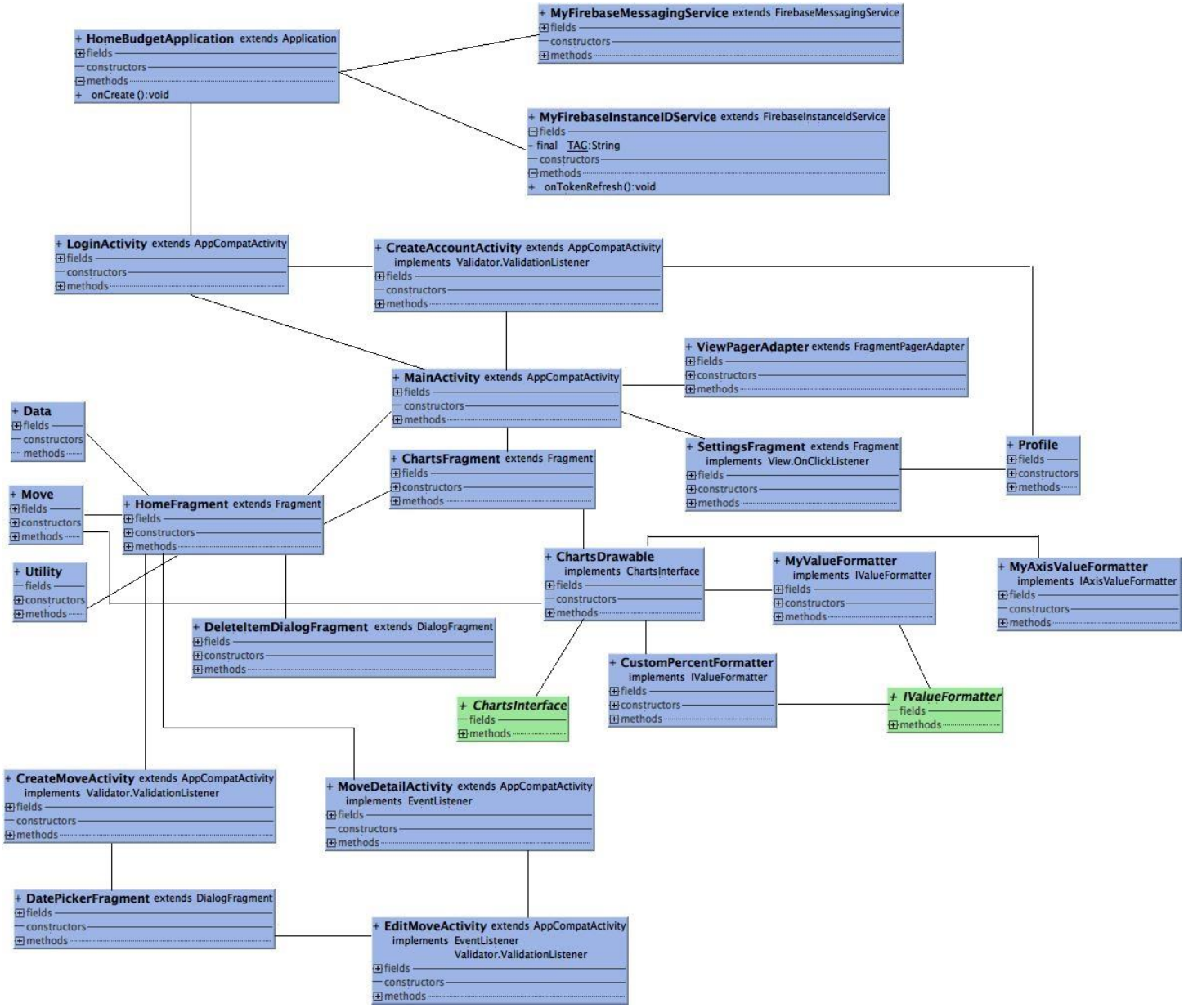


Ilustración 27 - Diagrama de clases

5.4 MODELO LÓGICO DE DATOS

En este apartado presentamos el diccionario de datos basado en un diseño relacional, mostrando las diferentes entidades que componen la base de datos. En este punto es importante aclarar que nuestra base de datos es NoSQL, es decir, es una base de datos no relacional. No contiene tablas aunque a continuación lo vayamos a representar de esta manera "estándar" para un mejor entendimiento.

En nuestro caso tenemos una base de datos orientada a documentos, ésta colecciones de documentos en el que cada documento almacena pares de clave-valor y además puede contener subcolecciones con documentos.

De esta manera el diccionario de datos junto con la decisión de consumir un BAS (Back-end As a Service) simplifica en gran medida el tratamiento de datos, el diccionario es el siguiente:

Users collection
Users: Tipo colección de documentos

Users documents	
Country:	clave-valor (String)
Date:	clave-valor (String)
GlobalPosition:	clave-valor (Double)
Name:	clave-valor (String)
LastName:	clave-valor (String)
Mail:	clave-valor (String)
Moves	sub-colección de documentos

Moves documents	
Amount:	clave-valor (Double)
Date:	clave-valor (String)
Concept:	clave-valor (String)
Category:	clave-valor (String)
Description:	clave-valor (String)

Al final lo que nos encontramos es con una colección de documentos en el que cada documento representa a un usuario y tiene unos pares clave-valor que almacenan la información del usuario en cuestión.
A su vez, cada documento usuario tiene asociado una subcolección de documentos "movimiento" (Gasto o ingreso de dinero). Y cada documento "movimiento" encontramos una serie de pares clave-valor que almacena la información referente al mismo. Es un paradigma diferente pero cada vez más común.

Detallar que no se ha indicado si el campo puede ser nulo o no ya que la base de datos nos permitirá escribir cada documento como nosotros deseamos, aunque nosotros escribiremos nuestros documentos con el mismo formato para recoger la información y tener una cierta organización de una manera más sencilla, la realidad es que literalmente podríamos escribir cada documento de cada colección de manera diferente y almacenar una información diferente.

Esto sería impensable para una base de datos relacional en la que cada tabla-entidad tiene que recibir unos parámetros concretos. Veamos una representación a modo de ejemplo para que quede más claro.

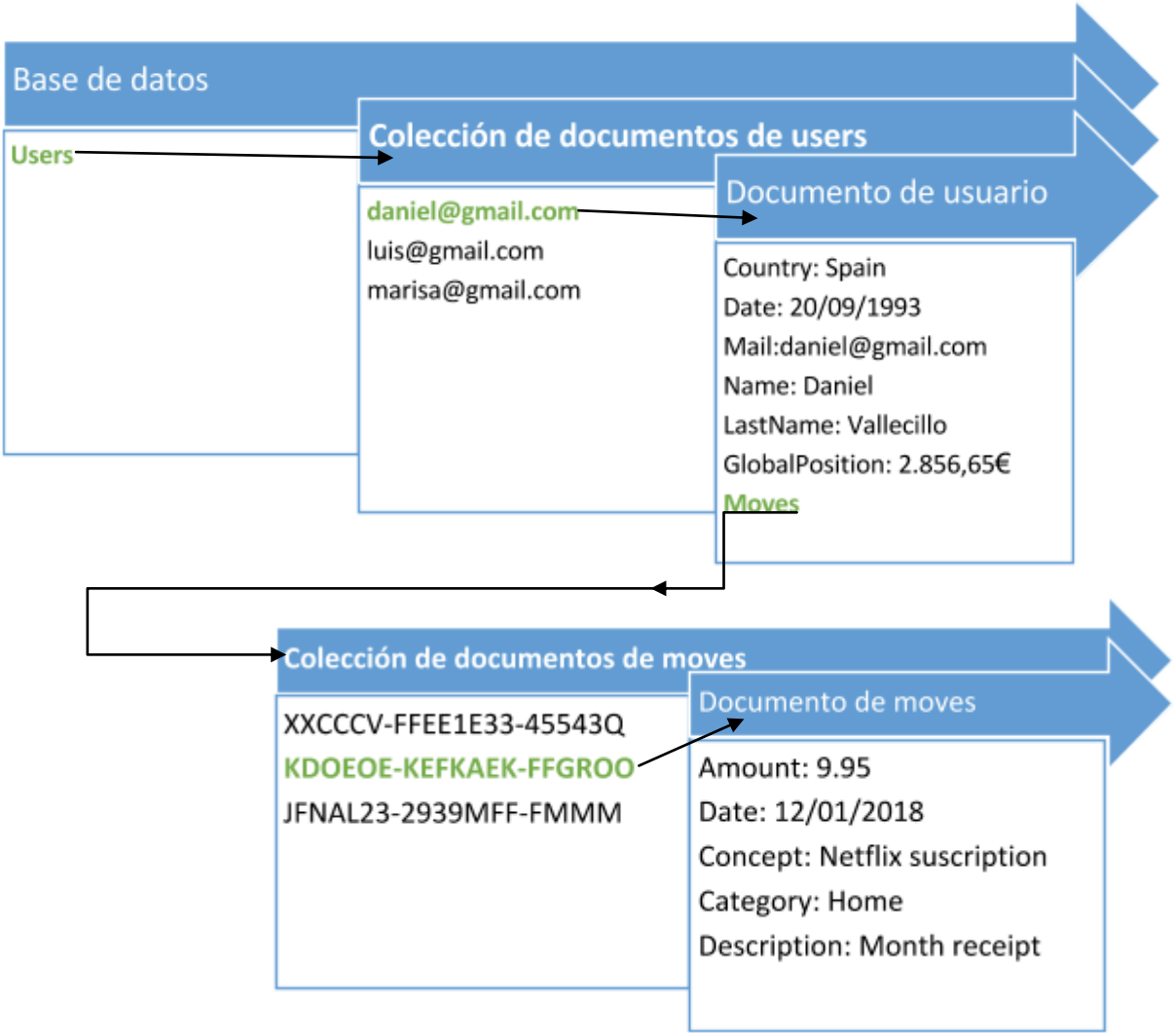


Ilustración 28 - Modelo lógico de datos

5.5 DISEÑO DE LA INTERFAZ

Al momento de crear nuestro proyecto se selecciona como API mínima la API 15 conocida como “Ice Cream Sandwich” ya que nos permitiría que nuestra aplicación funcionase en el 100% de los dispositivos según la infografía facilitada por Android Studio.

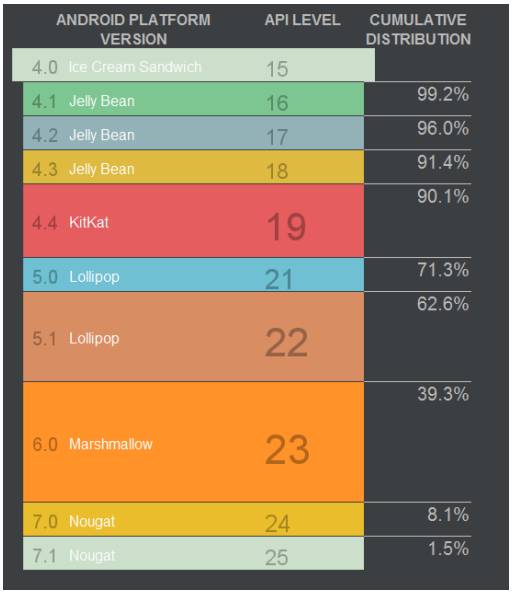


Ilustración 29 - Fragmentación de Android

Hemos incluido esto en el apartado de diseño porque Material Design surgió a partir de Android 5.0, lo que significa que necesitaremos dar compatibilidad a Material Design en versiones anteriores a la API 20 del sistema. La librería que nos va a dar esta compatibilidad es Them.AppCompat.

Hemos seguido la filosofía Material Design y algunas de sus guías disponibles en la web[8] , pensando en dotar a la aplicación de una identidad visual y sensación de uniformidad entre las diferentes partes, veamos el ejemplo de la pantalla de Home y la de Charts:

Ilustración 30 - Vista de home

[8] https://developer.android.com/training/material/get-started.html?hl=es-419

Ilustración 31 - Vista de charts

La primera decisión que se tomó a nivel de diseño fue la selección de una correcta paleta de colores con la que poder trabajar.

Para ello, se debía de elegir un color primario, un primario oscuro y un color de acentuación que fueran la base de nuestra paleta de colores.

Se eligió un color blue light #03A9F4 como color primario, a partir de éste se obtiene el color primario oscuro. Todos los colores básicos (blue light lo es) de Material Design están ponderados con el número 500, para elegir el primario oscuro debemos de tomar un tono 200 puntos por encima en la paleta de la tonalidad, es decir el número 700 de la paleta de azules.

Mejor veámoslo en la siguiente imagen:

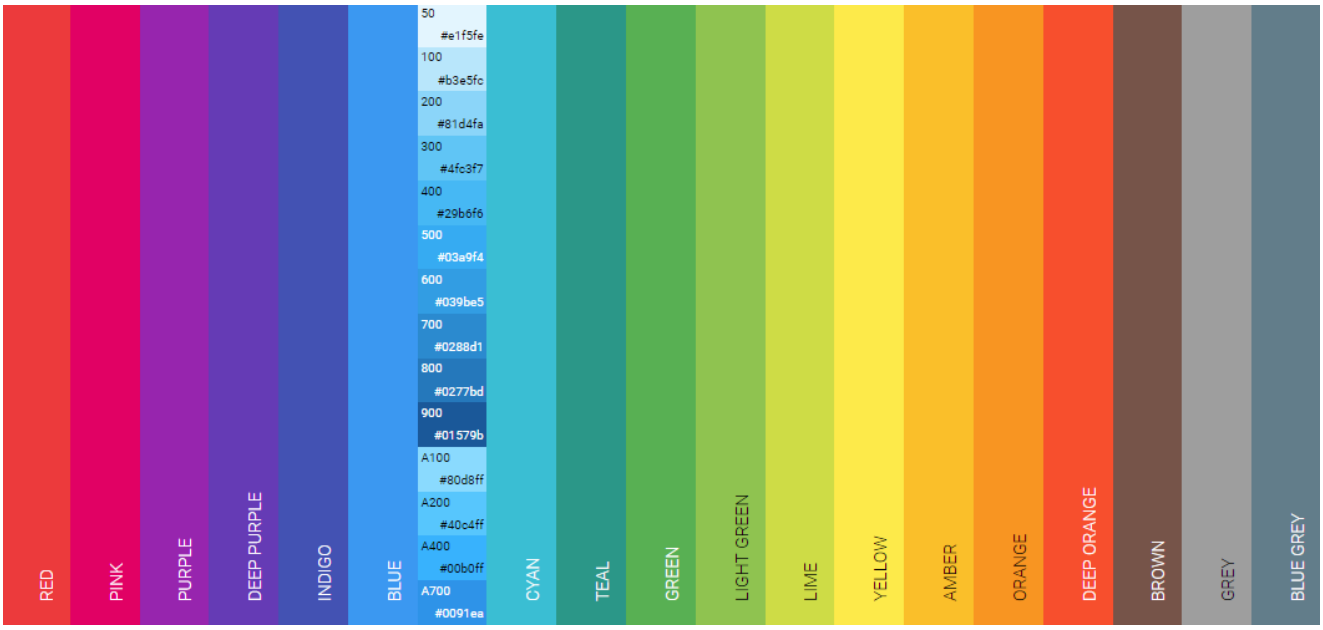


Ilustración 32 - Paleta de colores

Una vez obtenidos los dos colores primarios se debe de elegir el color de acentuación, en nuestro caso se creyó conveniente que el “Amber” podía quedar bien y mostrar al usuario de forma inequívoca cuando está seleccionado y a la vez resaltar los detalles.

El resto de la paleta se completó incluyendo algunos tonos de rojo y ver para las cantidades y botones. Blancos, negros y grises para uso general.

Respecto al patrón de diseño, se ha empleado el patrón Bottom navigation view para el menú de navegación y se ha combinado con un view pager en el que se cargan las diferentes pantallas y nos permite hacer swipe entre ellas como si fuera un libro. Está inspirado en el diseño actual de Instagram[9].

[9] https://www.instagram.com/?hl=es

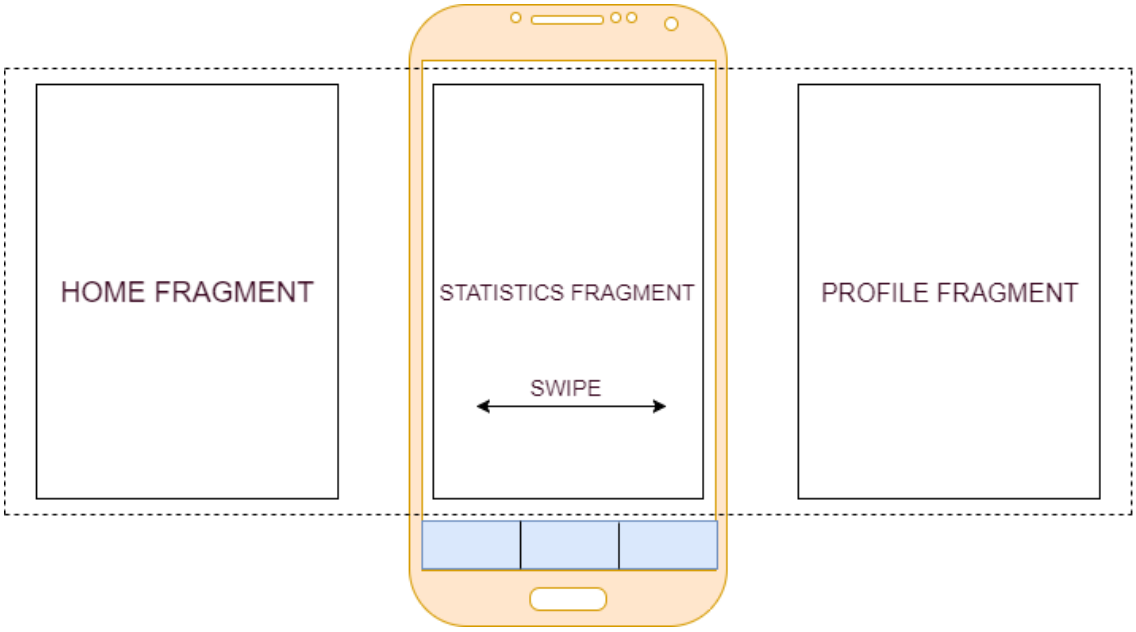


Ilustración 33 - Patrón de diseño

Además, para añadir interactividad a la aplicación, se han añadido algunas transiciones y animaciones personalizadas cómo la que nos encontramos cuando aún no hemos añadido gastos o hemos decidido borrar los que teníamos.

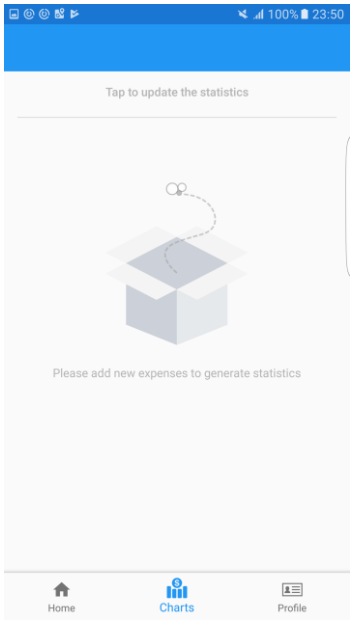


Ilustración 34 - Vista vacía

A continuación, vamos a realizar una explicación detallada de las pantallas que componen la aplicación:

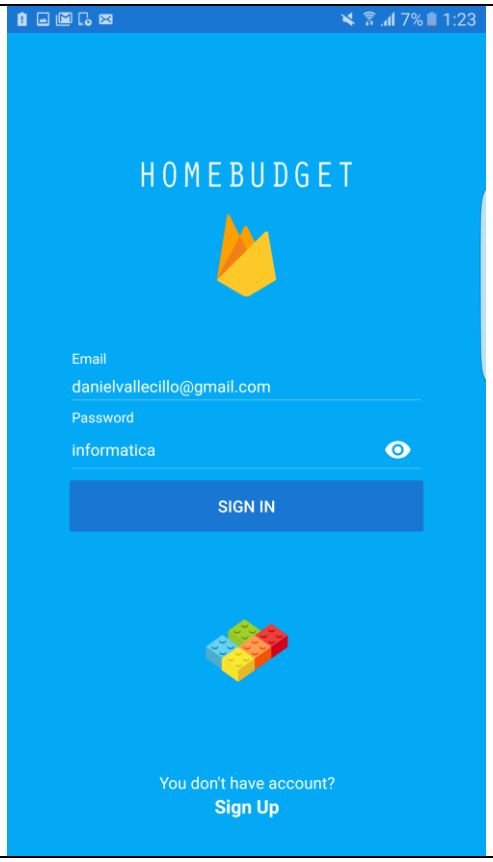
INICIO DE SESIÓN	
Descripción	Es la pantalla inicial de la aplicación cuando no estemos autentificados, ésta nos permite acceder al resto de la aplicación previo logueo o ir a la pantalla de registro si aún no tenemos cuenta. Si tenemos una sesión activa podemos acceder a la aplicación sin pasar por esta pantalla.
Activación	Al abrir la aplicación o pulsar sign out desde nuestro profile.
Diseño	
Eventos	• Pulsación sobre sign up para ir al registro. • Pulsación sobre sign in para acceso a la aplicación. • Eventos de validación. • Evento sesión activa disponible. • Evento no conexión a internet.

Tabla 37 - Vista de inicio de sesion

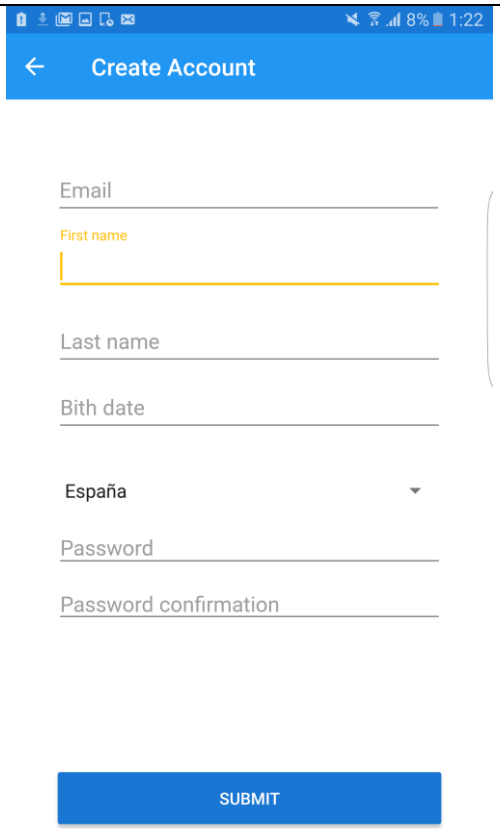
REGISTRO	
Descripción	Esta pantalla nos ofrece un formulario para poder registrarnos en la aplicación, si se cumplen las condiciones de validación y conexión, nos permite registrar nuestra cuenta, crea un profile con nuestra información de registro, inicia sesión automáticamente con nuestra nueva cuenta y nos manda a la pantalla principal de la aplicación.
Activación	Al pulsar sign up desde la pantalla de inicio de sesión.
Diseño	
Eventos	• Pulsación sobre submit para enviar la información. • Pulsación sobre birth date para mostrar un date picker. • Pulsación sobre el spinner para mostrar la lista de países. • Eventos de validación. • Evento no conexión a internet.

Tabla 38 - Vista de registro

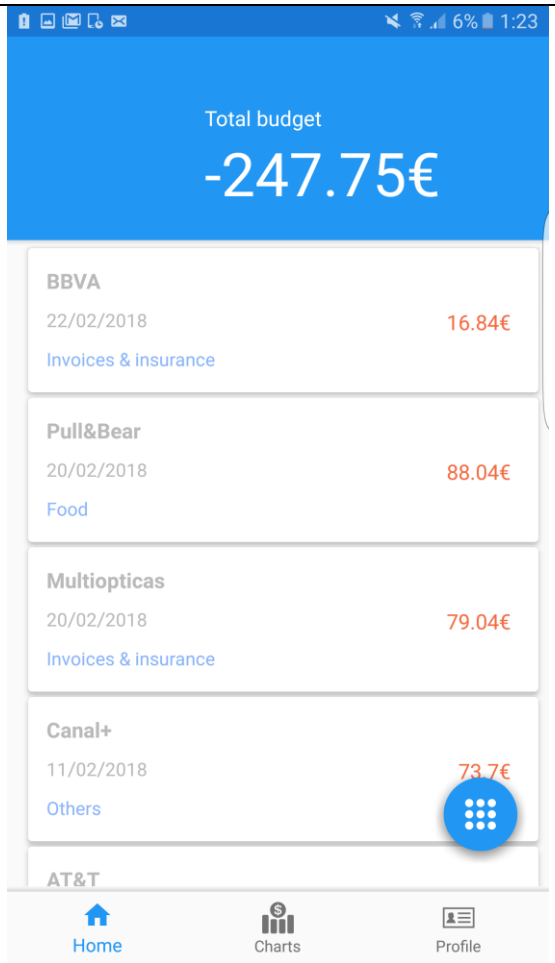
HOME	
Descripción	Es el fragmento inicial de la pantalla principal de la aplicación, en él se muestran los gastos e ingresos, y permite ir a ver los detalles de los mismos, borrarles o añadir nuevos.
Activación	Al abrir la aplicación con una sesión activa, al completar un inicio de sesión, cuando es llamada desde el botón de la bottom bar o se hace swipe desde la pantalla de las estadísticas.
Diseño	
Eventos	• Pulsación sobre el botón flotante. • Pulsación corta sobre un elemento de la lista. • Pulsación larga sobre un elemento de la lista. • Eventos de scroll. • Evento de swipe. • Evento escucha modificaciones de los movimientos.

Tabla 39 - Vista de home

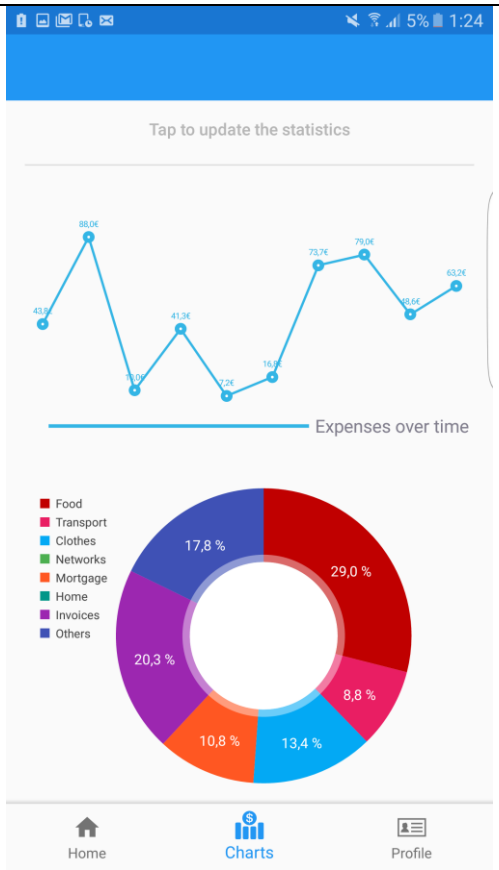
GRAFICOS	
Descripción	Es el fragmento de la pantalla principal de la aplicación dónde se recogen las estadísticas generadas de los gastos.
Activación	Cuando es llamada desde el botón de la bottom bar o se hace swipe desde el fragmento de home o el de profile.
Diseño	
Eventos	• Pulsación sobre actualizar. • Pulsaciones sobre el gráfico de sectores. • Evento de swipe.

Tabla 40 - Vista de gráficos

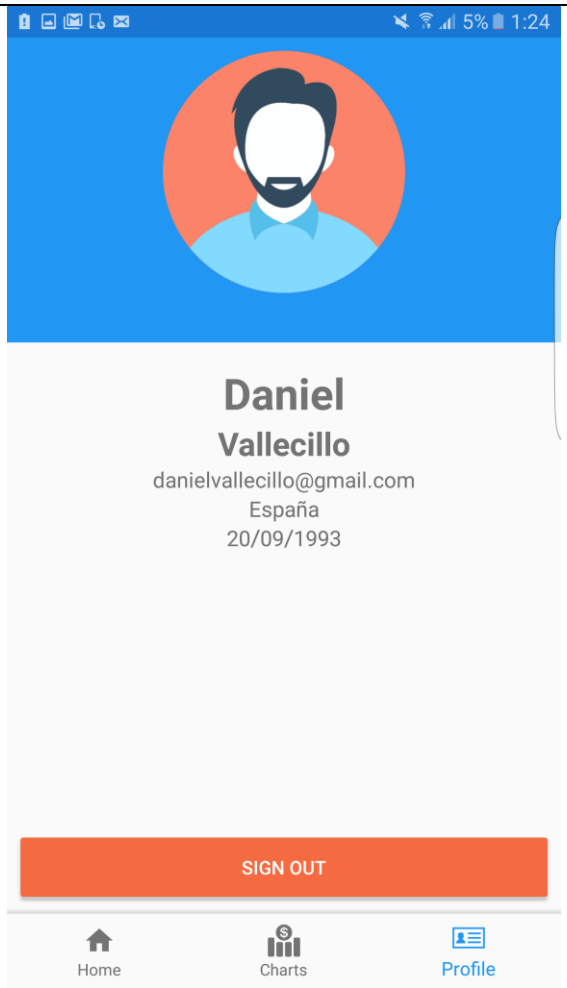
PERFIL	
Descripción	Es el fragmento de la pantalla principal de la aplicación dónde se muestra la información del perfil del usuario y se ofrece la opción de cerrar la sesión.
Activación	Cuando es llamada desde el botón de la bottom bar o se hace swipe desde el fragmento de charts.
Diseño	
Eventos	• Pulsación sobre sign out. • Evento escuchador modificaciones sobre el perfil. • Evento de swipe.

Tabla 41 - Vista de perfil

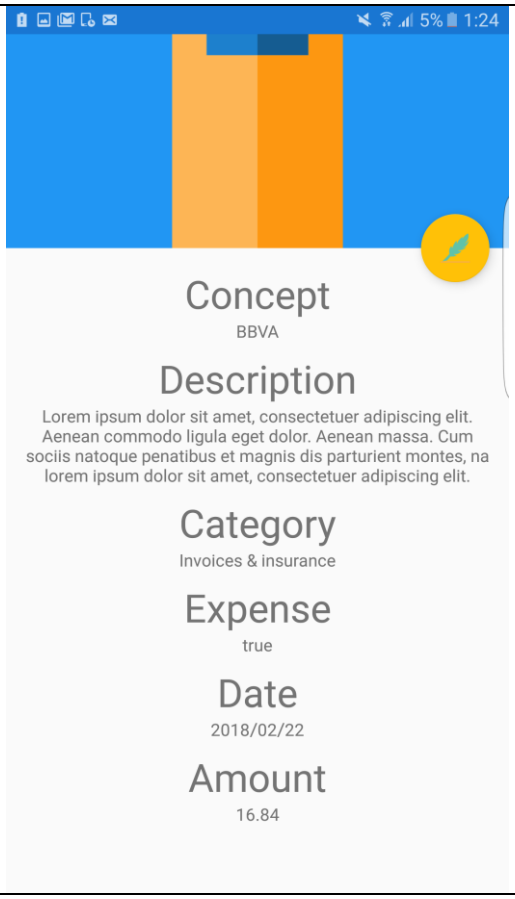
DETALLES DE MOVIMIENTO	
Descripción	Es la pantalla que muestra los detalles de un movimiento y nos ofrece la opción de editarlo.
Activación	Cuando es llamada a través de una pulsación sobre un elemento del recyclerview que contiene los movimientos.
Diseño	
Eventos	• Pulsación sobre editar. • Pulsación sobre el botón up de la barra (ir atrás). • Evento de scroll. • Evento modificaciones.

Tabla 42 - Vista de detalles de movimiento

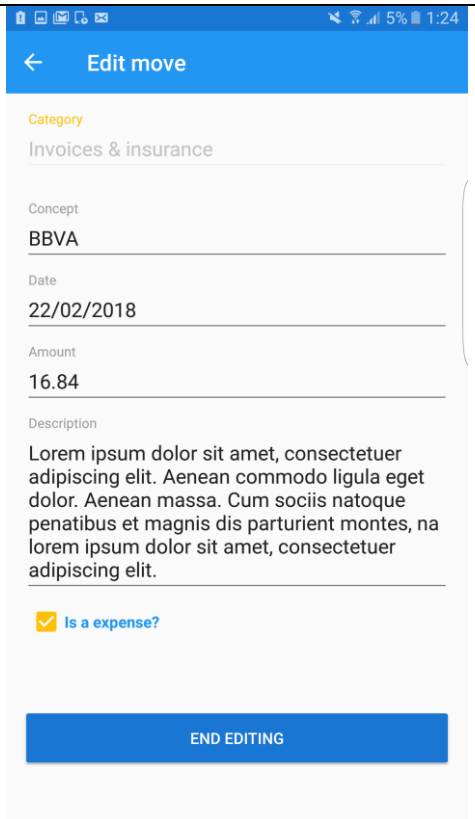
EDITAR MOVIMIENTO	
Descripción	Es la pantalla que muestra los detalles de un movimiento sobre campos editables.
Activación	Es llamada desde la pantalla de detalles de un movimiento.
Diseño	
Eventos	• Pulsación sobre end editing. • Pulsación sobre el botón up de la barra (ir atrás). • Pulsación sobre date para mostrar un pick date.

Tabla 43 - Vista de editar movimiento

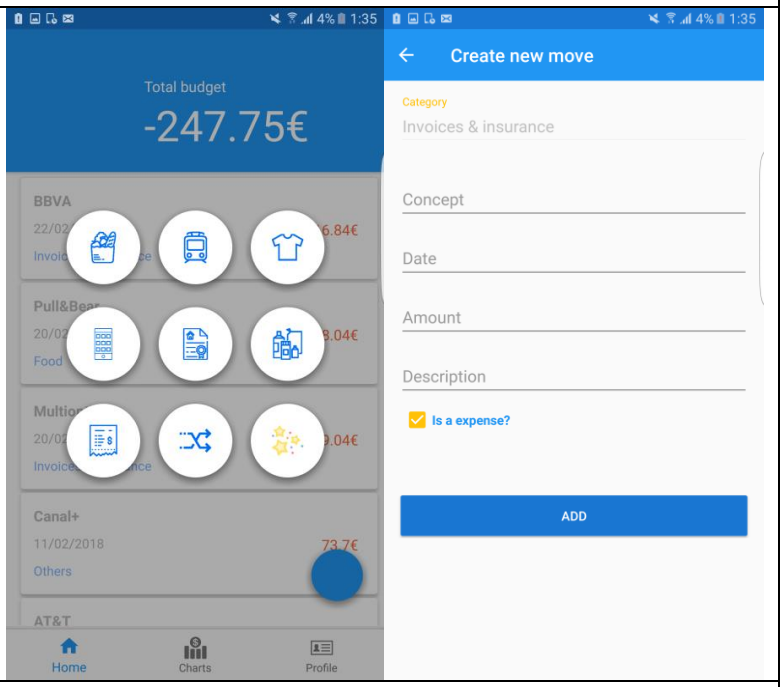
AÑADIR MOVIMIENTO	
Descripción	Es la pantalla que nos permite añadir un movimiento rellenando los campos editables que se muestran.
Activación	Es llamada desde la pantalla de home tras seleccionar una categoría
Diseño	
Eventos	• Pulsación sobre el botón de crear. • Pulsación sobre el botón up de la barra (ir atrás). • Pulsación sobre date para mostrar un pick date.

Tabla 44 - Vista de añadir movimiento

CAPÍTULO 6

IMPLEMENTACIÓN

En este capítulo vamos a tratar todo lo referente a la implementación del proyecto.

En primer lugar, se va a proporcionar un contexto en relación a la toma de decisión referente a la inclusión de incluir una tecnología u otra o unos componentes u otros. Una vez proporcionado este contexto se va a realizar una descripción técnica de cómo implementar esos componentes y de su funcionamiento. En concreto me gustaría poner la atención en la base de datos y en el "módulo" que genera los gráficos de los gastos.

En segundo lugar, se describirá con más profundidad algunos de los componentes considerados como clave en la aplicación, se verán detalles de implementación a nivel de código.

6.1 HERRAMIENTAS EMPLEADAS

Herramientas y tecnologías

- **Android Studio:** Se trata de la primera, pero también la más relevante, es el entorno de desarrollo integrado oficial que reemplazó hace ya tres años a Eclipse. Basado en IntelliJ IDEA de JetBrains tiene una licencia Apache 2.0 y se encuentra disponible para todas las plataformas.

- **Android OS:** Sistema operativo que nació para dispositivos móviles con pantalla táctil pero que hoy día también ofrece soporte para tabletas, televisiones, relojes inteligentes, automóviles…

- **Visual Studio**: Es un editor de código ligero desarrollado por Microsoft, inicialmente en exclusiva para Windows pero que actualmente cuenta con soporte multiplataforma. Tiene embebido soporte para control de versiones.

- **Git:** Es un software de control de versiones diseñado por el mismísimo Linus Torvalds, permite llevar un registro de todas las modificaciones que se realizan sobre los archivos. Es software libre.

- **Github:** Es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el control de versiones Git.

- **StarUML:** Se trata de una herramienta que permite realizar de diagramas UML de todo tipo.

- **Smartsheet:** Es una herramienta online que ofrece todo tipo de funcionalidades para la planificación de los proyectos, contiene plantillas para planificaciones de desarrollos ágiles, kanbas… etc.

- **Firebase plattform:** Es una plataforma para desarrollo de aplicaciones web y móviles desarrollada por una empresa que fue adquirida por Google en 2014.

- **Google Analytics:** Es una herramienta de analítica web de la empresa Google que ofrece información agrupada del tráfico y comportamiento de los usuarios de una página web, aplicación web y ahora también de aplicaciones móviles.

- **Google Cloud Messaging:** Plataforma para el manejo de mensajes y notificaciones para Android, iOS y aplicaciones web.

- **Firebase Auth:** Es un servicio actualmente ofrecido desde la plataforma de Firebase que permite autenticar a los usuarios únicamente utilizando código desde el lado del cliente. Incluye soporte para la autenticación con servicios de redes sociales.

- **Facebook Login:** Se trata de un servicio oficial de Facebook que nos permite implementar la autenticación a través de la cuenta de la conocida red social. Aunque no visible para el usuario final, esta funcionalidad está implementada y funcionando correctamente.

- **Cloud Firestore:** Se trata de una base de datos NoSQL flexible y escalable para almacenar y sincronizar datos entre el cliente y el servidor que se encuentra aún en fase Beta.

- **JSON:** JavaScript Object Notation, es un formato de texto ligero estandarizado a menudo empleado cuando hay servicios RES entre otros de por medio.

- **Java 8:** Principal lenguaje de programación empleado para el desarrollo del proyecto, aunque Klotin hoy en día es también una opción real y muy válida para el desarrollo de aplicaciones Android.

- **XML:** Extensible Market Language, se trata del lenguaje utilizado para la definición de todas las vistas de la aplicación.

Dependencias utilizadas en el proyecto

- **MPAndroid Chart 'com.github.PhilJay:MPAndroidChart:v3.0.3':** Es una librería que nos permite realizar diferentes tipos de gráficos a partir de una entrada de datos debidamente tratada y customizar su aspecto.

- **ButterKnife 'com.jakewharton:butterknife-compiler:8.7.0':** Es una de las librerías más utilizadas por los desarrolladores, nos permite realizar un "binding" de las vistas de una manera más ordenada y eficiente evitando la continua llamada a la conocida función .findViewById() para poder realizar una ligadura entre nuestros elementos visuales y nuestras variables.

- **Lottie 'com.airbnb.android:lottie:2.3.1':** Se trata de una librería de Airbnb que permite parsear animaciones de After Effects (Conocido programa de animación profesional que forma parte de Adobe Suite) a formato JSON pudiendo así ser incluidas y manejadas dentro de nuestros proyectos.

- **BoomMenu 'com.nightonke:boommenu:2.1.1' :** Es una librería que permite realizar el "boom menú" que se ha utilizado para seleccionar la categoría.

Además de las dependencias correspondientes a los servicios implementados de Firebase y Facebook.

6.2 REQUISITOS HARDWARE Y SOFTWARE

Estos son los requisitos mínimos que deberá tener un Smartphone o Tablet para poder correr correctamente la aplicación.

Requisitos Software	**Sistema Operativo**	**1.2GHz Dual-Core**
	Otros	1.5Gb
Requisitos Hardware	CPU	Conexión a Internet
	RAM	Android API 15 o superior
	Otros	-

Tabla 45 - Requisitos de hardware y software

6.3 DETALLES DE LA IMPLEMENTACIÓN

En primer lugar, debemos de recalcar el término sistema, cómo conjunto, puesto que se trata de un conjunto de una aplicación Android para que el usuario final pueda administrar el presupuesto del hogar, tener una contabilidad de los ingresos y gastos, pero por otro lado se proporciona también una herramienta de administración que nos permite entender el sistema cómo un producto completo.

Al final, se consideró que no tenía ningún sentido desarrollar una aplicación final para el usuario sin tener en cuenta la administración, mantenimiento y modificación de la misma. El sistema no sería un producto completo sin esa parte “administrativa” que nos permita crecer, poner en producción la misma y por supuesto, también mantener.

6.3.1 LA BASE DE DATOS

Tal cómo se comentó brevemente anteriormente, el sistema utiliza una base de datos NoSQL en la nube llamada Cloud Firestore. Vamos a tratar de entender esta decisión antes de entrar en detalles.

6.3.1 - ¿POR QUÉ UNA BASE DE DATOS EN LA NUBE?

Esta decisión se entiende en el contexto de querer aportar a la aplicación la mayor persistencia posible. Al tratarse de sistema que permitiese al usuario final almacenar su histórico de gastos e ingresos cómo mayor atractivo, se pensó que no sería la mejor opción almacenar los datos en la memoria del teléfono cómo modo principal de persistencia ya que el simple hecho de cambiar de teléfono o borrar la aplicación eliminaría todo rastro de nuestro histórico de datos, que posiblemente contuviese información recogida a lo largo de mucho tiempo.

Era una idea realmente atractiva, pero tampoco terminaba de ser una solución completa porque requería de una conexión a internet permanente para realizar algo tan simple como visualizar, añadir o eliminar movimientos de dinero.

Por ello, decidí buscar una base de datos en la nube que ofreciese asistencia sin conexión y permitiese tener a la aplicación móvil capacidad de respuesta sin importar la latencia de red ni la conectividad a internet.

6.3.1 - ¿Y POR QUÉ UNA BASE DE DATOS NOSQL?

- Una base datos NoSQL a diferencia de una SQL nos permite tener estructuras de datos variables.
- Es mucho más eficiente que una base de datos SQL.
- Nos permite la captura y procesado de eventos, es decir nos permite implementar una base de datos a tiempo real.
- Son más fácilmente escalables.
- Las consultas para grandes cantidades de datos están mucho más optimizadas.
- No generan cuellos de botella.
- Se pueden ejecutar en clusters de máquinas baratas.

A cambio perdíamos varias características que consideré que no eran excesivas en relación con las que ganábamos. Al no usar una base de datos SQL perdimos:

- Experiencia: Al tratarse de bases de datos relativamente “nuevas”, que no poco utilizadas ni con falta de soporte.
- Falta de compatibilidad: Mientras que las bases de datos SQL comparten muchos de los estándares, el mundo de las bases de datos NoSQL es todo lo contrario.
- Un lenguaje de consultas estándar. En este caso nos tendríamos que adaptar al lenguaje de consultas proporcionado por la base de datos elegida.
- Atomicidad de las instrucciones e integridad de los datos. Cómo hemos comentado antes, si quisiéramos, literalmente podríamos almacenar pares de clave-valor distintos.

Finalmente, después de investigar diferentes soluciones se encontró un producto aún en fase Beta, desarrollado por Google que se adaptaba a todas nuestras necesidades. Éste, venía a ser una evolución de otro producto anterior.

En el momento de finalizar este proyecto aún se encuentra en fase Beta, con las limitaciones y dificultades que esto representa: posibles errores y fallos, falta de documentación por parte de la comunidad (Que no por parte de Google que tiene una buena documentación), dificultad de solventar problemas complejos para los que el producto aún no está preparado o la incertidumbre de si puede realizarse. A pesar de todo ello, era lo idóneo.

Veamos que nos ofrece y aporta Cloud Firestore a nuestro proyecto y porque se adaptaba al contexto anterior:

Flexibilidad	El modelo de datos de Cloud Firestore admite estructuras de datos flexibles y jerárquicas. Almacena tus datos en documentos, organizados en colecciones. Los documentos pueden contener objetos anidados complejos, además de subcolecciones.
Consultas expresivas	En Cloud Firestore, puedes usar consultas para recuperar documentos individuales específicos o para recuperar todos los documentos de una colección que coinciden con los parámetros de la consulta. Tus consultas pueden incluir varios filtros en cadena y combinar los filtros con criterios de orden. También se indexan de forma predeterminada, por lo que el rendimiento de las consultas es proporcional al tamaño de tu conjunto de resultados, no del conjunto de datos.
Actualizaciones en tiempo real	Al igual que Realtime Database, Cloud Firestore usa la sincronización de datos para actualizar los datos de cualquier dispositivo conectado. Sin embargo, también está diseñado para ejecutar consultas de recuperación únicas y sencillas de manera eficiente.
Asistencia sin conexión	Cloud Firestore almacena en caché datos que usa tu app de forma activa, por lo que la app puede escribir, leer, escuchar y consultar datos, aunque el dispositivo se encuentre sin conexión. Cuando el dispositivo vuelve a estar en línea, Cloud Firestore sincroniza todos los cambios locales de vuelta a Cloud Firestore.
Diseño para ajustarse a escala	Cloud Firestore te ofrece lo mejor de la poderosa infraestructura de Google Cloud Platform: replicación automática de datos multirregión, garantías de coherencia sólida, operaciones atómicas por lotes y asistencia real sobre transacciones. Diseñamos Cloud Firestore para controlar las cargas de trabajo de las bases de datos más complejas de las apps más grandes del mundo.

Tabla 46 - Características Cloud Firebase

Ahora veamos a modo de resumen cuales son los pasos a seguir para realizar una implementación y configuración de la misma detallando cada uno de los puntos:

6.3.1 – CREAR UN PROYECTO DE FIREBASE

Accedemos a https://firebase.google.com/ y nos dirigimos arriba a la derecha dónde pone “Go to console”, hacemos click e iniciamos sesión con nuestra cuenta de Google o cuenta de acceso proporcionada por el autor del proyecto.

En este punto se nos mostrará los diferentes proyectos de Firebase asociados a nuestra cuenta, cada proyecto puede contener varias aplicaciones. Esto es así porque al dar soporte a diferentes clientes Android, IOS, aplicaciones web o incluso Unity podemos encontrarnos con un proyecto en el que se tenga diferentes clientes, por ejemplo, uno para iOS y otro para Android, pero con un base de datos en común.

En esta pantalla simplemente dar a añadir proyecto

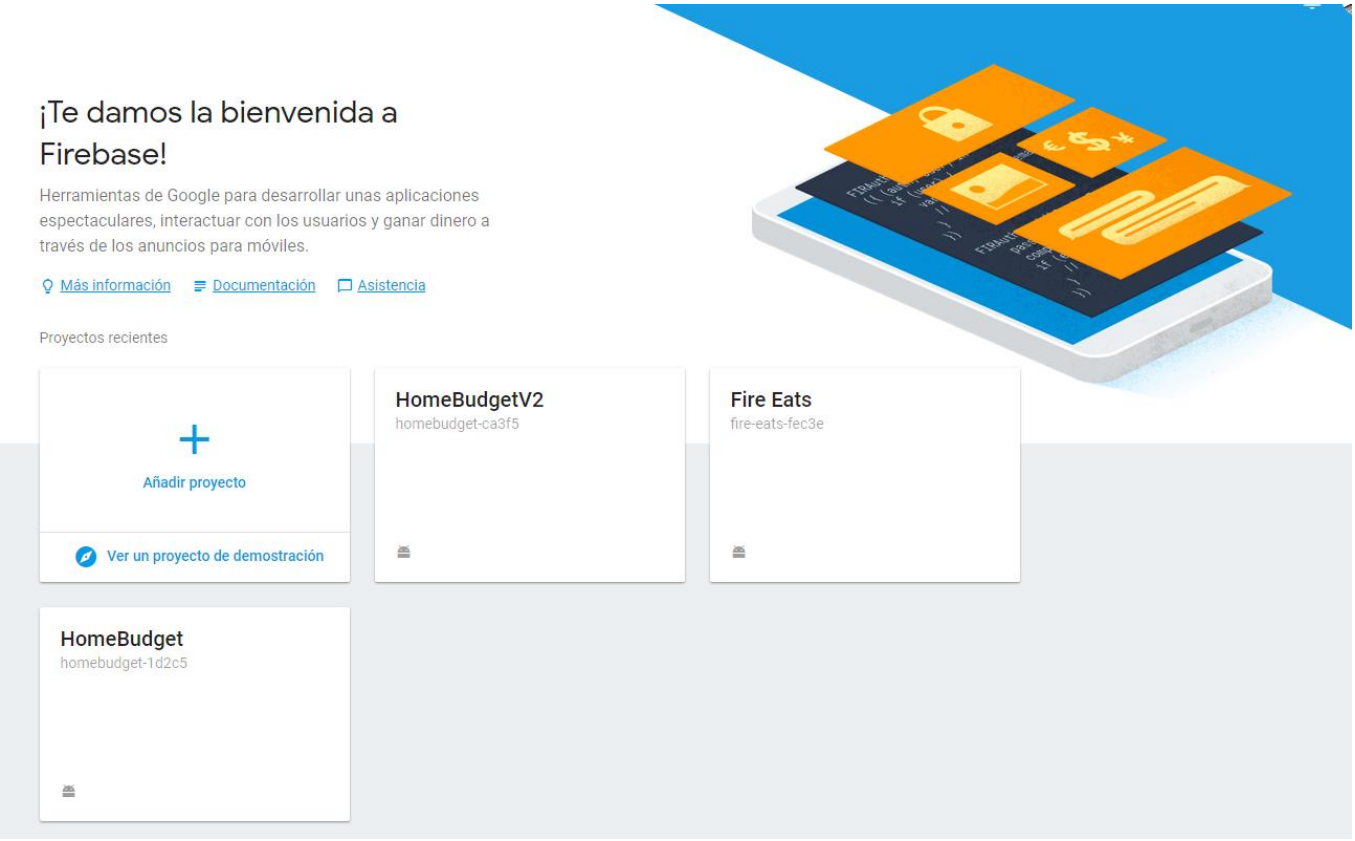


Ilustración 35 - Configuraración de proyecto 1

Se configura el proyecto, dándole nombre y un país dónde deseas que se encuentren los servidores que alojarán tus datos, lo más común es elegir un país cercano posible a dónde se encuentra el grueso de los usuarios de la aplicación.

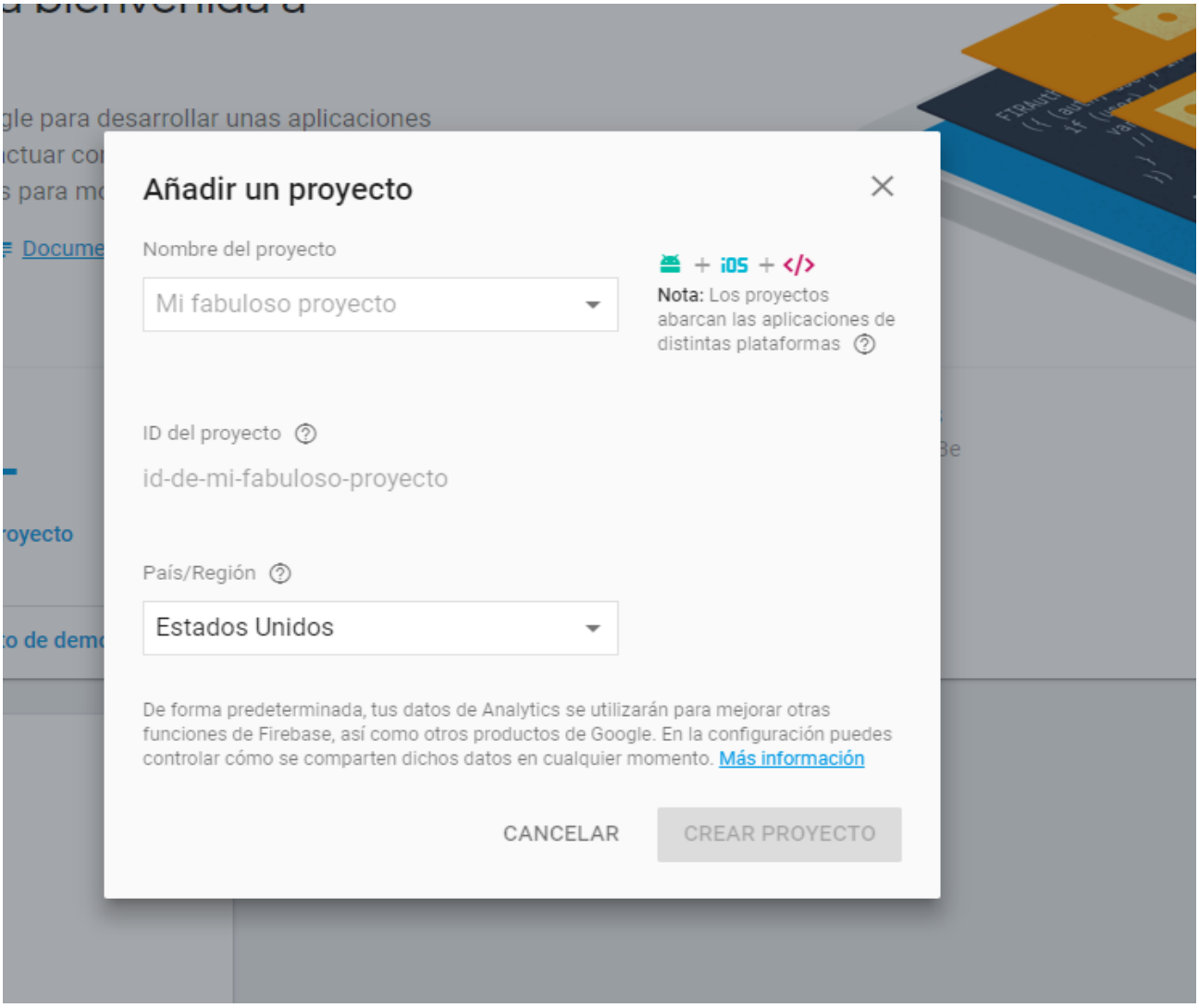


Ilustración 36- Configuraración de proyecto 2

Se nos configurará el proyecto y cuando esté listo nos aparecerá una pantalla como esta dónde daremos click en “Añade Firebase a tu aplicación de Android”

Ilustración 37- Configuraración de proyecto 3

6.3.1 – INTEGRAR EL SDK DE CLOUD FIRESTORE EN LA APLICACIÓN ANDROID

Continuamos dónde lo dejamos en el punto anterior, aquí rellenaremos con el nombre del paquete de Android de nuestra aplicación, un apodo para la aplicación y una firma digital que sólo es obligatoria para algunas funcionalidades muy específicas por lo que no será necesario.

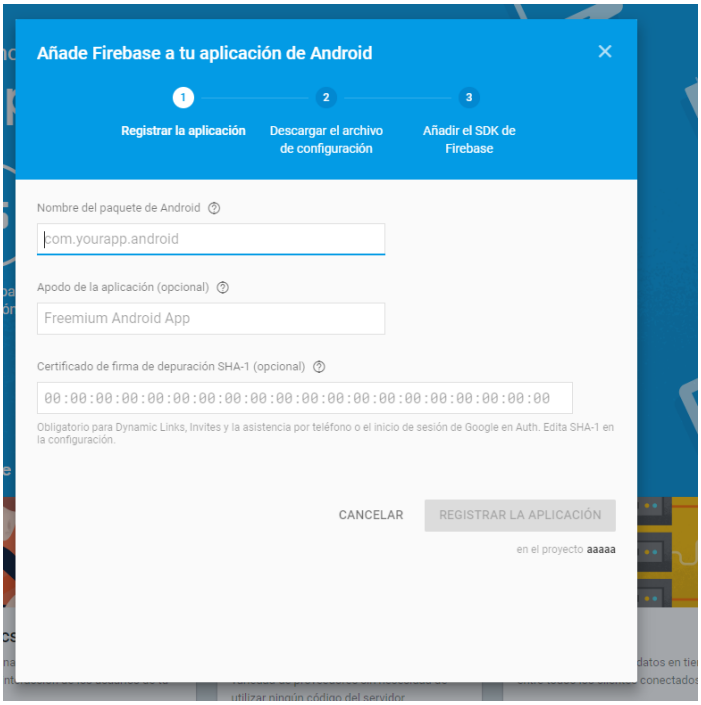


Ilustración 38- Configuraración de proyecto 4

Y aquí viene uno de los puntos quizás un poco confusos, se nos mostrará una pantalla como ésta para descargar un archivo json que deberemos incluir en el proyecto.

Varias cosas, la primera es que la inclusión del archivo es la clave para realizar la ligadura entre el proyecto en la web que acabamos de crear de Firebase y nuestra aplicación Android.

La segunda, la pantalla indica que debes de mover el archivo que has descargado “google-services.json” al root de tu aplicación. El problema es que por defecto Android Studio muestra la vista de Aplicación y no la vista de proyecto.

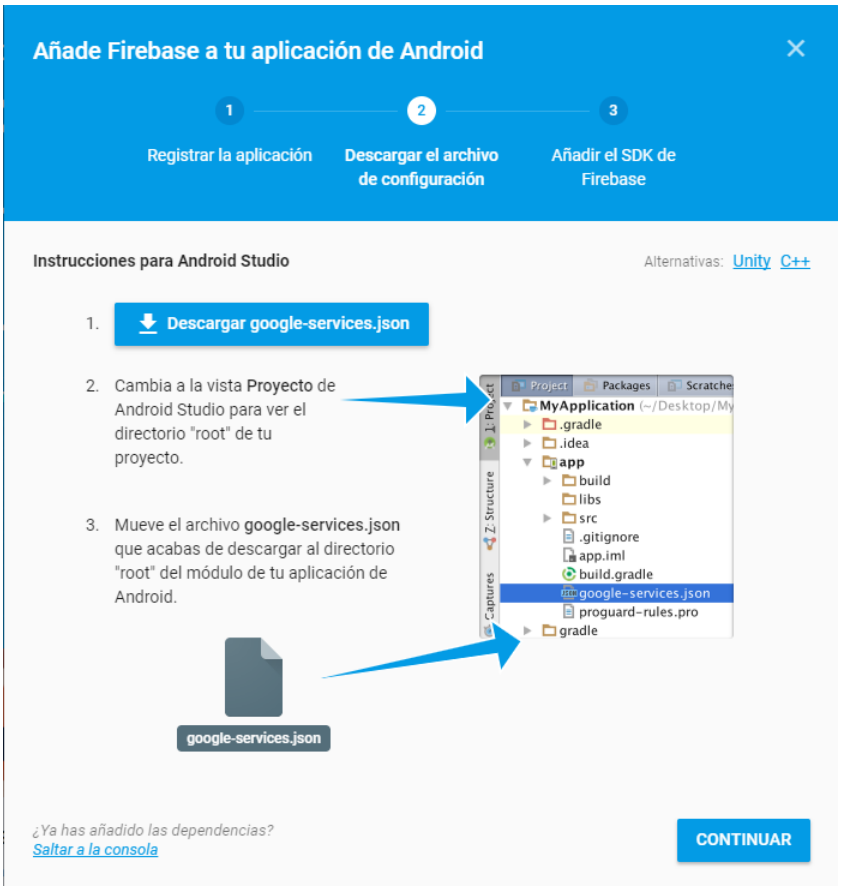
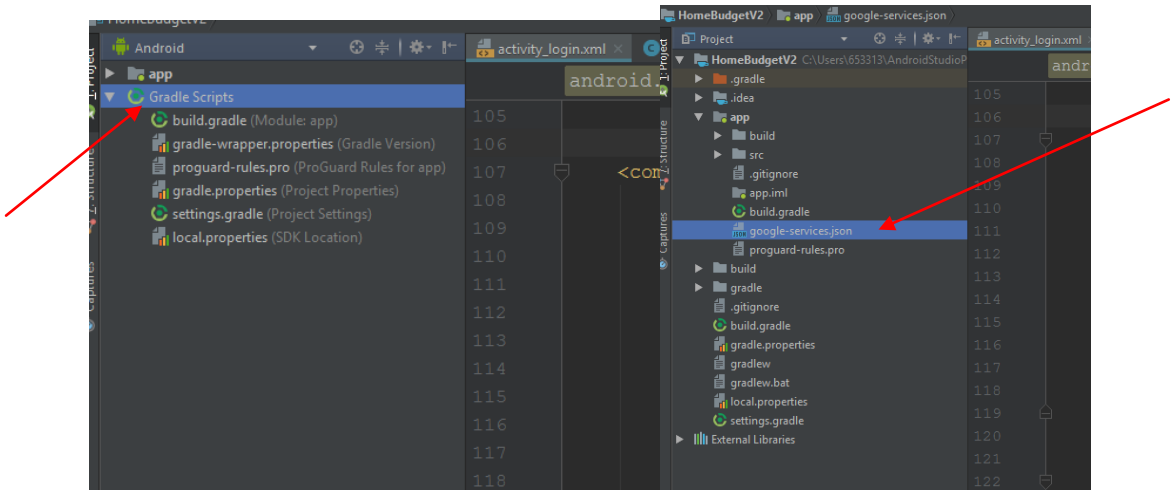


Ilustración 39- Configuraración de proyecto 5

Para ello vamos a nuestro Android Studio, cargamos el proyecto de nuestra aplicación, cambiamos a vista del proyecto haciendo click dónde indica la flecha roja en la siguiente imagen y pegamos el archivo en esa ruta.



Finalmente, para terminar de integrar el SDK en nuestra aplicación deberemos añadir las dependencias correspondientes y activar los Google services. Para ello:

En el archivo build.gradle de nuestro proyecto de Android añadir:

```
buildscript {
  dependencies {
    // Add this line
    classpath 'com.google.gms:google-services:3.1.0'
  }
}
```

Y en el archivo build.gradle de la aplicación añadir:

```
apply plugin: 'com.google.gms.google-services'
```

Y con eso habríamos finalizado la integracion del SDK de Cloud Firestore en nuestra aplicación. Nuestro siguiente paso será crear en nuestra aplicación los métodos referentes a la inserción de datos, recuperación y escucha.

6.3.1 – AGREGAR Y RECUPERAR DATOS

Aunque Cloud Firestore permite el uso de objetos Map de Java para representar a los documentos no suele ser la mejor opción y conviene definir tus objetos personalizados y si tienen la correcta notación en su definición, Cloud Firestore convertirá internamente los objetos en tipos de datos compatibles.

Primero debemos de definir nuestra clase Move, que forma parte de nuestro modelo de datos de la siguiente manera:

```
public class Move {

    private String ID;
    private String concept, description, date, category;
    private Double amount;
    private Boolean expense;

    public Move(){}

    public Move(String concept, String date, String description, Double amount,
String category, Boolean expense){

        this.concept = concept;
        this.date = date;
        this.description = description;
        this.amount = amount;
        this.category = category;
        this.expense = expense;
```

```
        this.ID = String.valueOf(UUID.randomUUID());
    }

    public String getID() {
        return ID;
    }

    public void setID(String ID) {
        this.ID = ID;
    }

    public String getConcept() {
        return concept;
    }

    public void setConcept(String concept) {
        this.concept = concept;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public String getCategory() {
        return category;
    }

    public void setCategory(String category) {
        this.category = category;
    }

    public Double getAmount() {
        return amount;
    }

    public void setAmount(Double amount) {
        this.amount = amount;
    }

    public Boolean getExpense() {
        return expense;
    }
```

```
    public void setExpense(Boolean expense) {
        this.expense = expense;
    }

    public float getValueY() {

        double result = amount;
        float y = (float)result;
        return y;
    }
```

Análogamente se haría lo mismo para cualquier otra clase del modelo de datos que fuera a incluirse en la base de datos.

De esta manera, primero en el método onCreate() o onCreateView() de nuestra actividad o fragmento, creamos una instancia de la base de datos para poder manejarla.

```
      db = FirebaseFirestore.getInstance();
```

En nuestro caso además recuperamos el usuario actual que está autenticado en la aplicación, para saber en todo momento si está autenticado y poder usar su correo para insertar la información en el documento correspondiente, pero eso no es obligatorio, solo es nuestro caso particular.

```
mUser = mAuth.getCurrentUser();
```

De esta forma solo podrán escribir o consultar en la base de datos los usuarios autenticados, además de que utilizaremos ese estado de la variable mUser para otras cosas cómo actualizar la vista dependiendo el usuario, por un lado nos permite entrar a la aplicación sin hacer login si existe una sesión activa del usuario y por otro lado abre la puerta a mostrar diferentes interfaces a diferentes usuarios por ejemplo, usuarios con roles diferentes.

Ahora, se va a mostrar una método general de inserción de información en la base de datos. Hay diferentes variantes según las necesidades, diferentes escuchadores… etc.

```
db.collection("users")
                .document("" + currentUser.getEmail().toLowerCase())
                .collection("moves").document(move.getID()).set(move)
                .addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void aVoid) {
                        Log.d(TAG, "Document successfully written!");
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Log.w(TAG, "Error writing document", e);
                    }
                });
```

Dónde se utiliza la instancia de la base de datos de la que antes hablamos, se le pide que mire en la colección “users”, en el documento cuyo nombre es igual al correo electrónico en minúsculas del usuario que tiene la sesión activa y se inserta un monumento en esa localización con un identificador generado aleatoriamente.

Por último a esta operación se le añade un escuchador de resultado, si la operación resulta exitosa addOnSuccessListener se activa y crea un Log que registra la operación, por lo contrario si la operación resulta fallida se activa addOnFailureListener y registra la excepción a través de un Log.

Ahora vamos a ver un método general que puede ser utilizado para realizar una petición y recuperar los datos deseados.

```
private DocumentReference mMoveRef;
```

```
// Get reference to the document which contains our focus move
        mMoveRef = mFireStore.collection("users")
                .document("" + mCurrentUser.getEmail())
                .collection("moves")
                .document(ID);
```

Para ello se declara una variable del tipo DocumentReference, dónde almacenaremos la referencia a un documento de nuestra base de datos de dónde deseamos recuperar información.
A continuación se llama a la instancia de la base de datos “mFireStore” previamente inicializada y se le pasa la “ruta” dónde se encuentra el documento del que deseamos recuperar información, ésta es asignada a la variable mMoveRef.

Bien ya tenemos la referencia al documento lista, ahora se añade a la lista un escuchador de eventos, al que luego más tarde le pasaremos la referencia del documento para que esté escuchando si hay modificaciones.

```
implements EventListener<DocumentSnapshot>
```

Al implementar este escuchador, debemos de implementar de manera obligatoria el método onEvent()

```
@Override
    public void onEvent(DocumentSnapshot documentSnapshot, FirebaseFirestoreException
e) {

        if(e != null){
            Log.d(TAG, "move:onEvent", e);
            return;
        }

        onMoveLoaded(documentSnapshot.toObject(Move.class));
    }

    private void onMoveLoaded(Move move) {

        // to do whatever
    }
```

En el método onEvent, es decir, cuando se produce una modificación de algún documento, se llama al método onMoveLoaded que recibe por parámetro un snapshot del documento parseado a la clase Move que es la clase que nosotros utilizamos, ya que en este caso estábamos recogiendo información de un movimiento en concreto.

En el método onMoveLoaded() simplemente hacer lo que necesites, éste recibe por parámetro un objeto tipo movimiento que corresponde al que almacena el documento que pedimos.

Por ello podríamos coger sus propiedades, y ponerlas en los campos de la vista, por ejemplo, para mostrar los detalles de un movimiento por pantalla o actualizar automáticamente la información cuando hay alguna modificación.

Sin embargo, falta un último detalle, activar y desactivar el escuchador. Esto lo haremos en el método onStart() de la actividad para que empiece a escuchar cuando la actividad está activa y se desactiva el escuchador en el método onStop() de la aplicación para así desligarlo cuando la actividad no esté en pantalla.

```
private ListenerRegistration mMoveRegistration;
```

```
  @Override
    public void onStart() {
        super.onStart();

        mMoveRegistration = mMoveRef.addSnapshotListener(MoveDetailActivity.this);
    }
```

```
@Override
    public void onStop() {
        super.onStop();

        if (mMoveRegistration != null) {
            mMoveRegistration.remove();
            mMoveRegistration = null;
        }
    }
```

6.3.1 –EDITAR Y ELIMINAR DATOS

Consiste en recuperar la ruta dónde del documento o colección que se desea modificar, alamacenarla en una variable del tipo DocumentReference y operar con ella. Veamos un ejemplo:

```
DocumentReference docRef = db.collection("users")
                .document("" + currentUser.getEmail())
                .collection("moves")
                .document(itemID);

        docRef.delete()
                .addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void aVoid) {
                        Log.d(TAG, "DocumentSnapshot successfully deleted!");
                        Toast.makeText(getActivity(), "Move deleted",
Toast.LENGTH_SHORT).show();
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Log.w(TAG, "Error deleting document", e);
```

```
                    Toast.makeText(getActivity(), "Could't possible delete the
move!", Toast.LENGTH_SHORT).show();
                }
            });
```

Se llama al método delete() del objeto documento de referencia y se añaden los escuchadores que se crean oportunos.

Para el caso de la edición el problema no es la edición en sí, puesto que simplemente se tiene que llamar al método update desde el documento o colección de referencia y especificar los parámetros que se desean modificar y su nuevo valor tal que así:

```
mMoveRef.update(
                "concept", concept,
                "date", date,
                "amount", amount,
                "description", description,
                "category", category,
                "expense", expense
        );
```

Realmente sencillo, el problema es, saber que elemento se tiene que modificar. Es decir, cómo saber cuál es en una base de datos cuyos identificadores son aleatorios y el orden en el que se encuentran en la base de datos es diferente al orden al que se muestra al usuario por pantalla, el elemento que en pantalla se indicó que se desea eliminar.

6.3.1 –PROTEGER LOS DATOS

Consiste en añadir restricciones a la base de datos con el objetivo de limitar el acceso a la misma y proporcionarla plena seguridad. Esto se hace desde la consola de Firebase, en el panel de autenticación.

```
service cloud.firestore {
  match /databases/{database}/documents {
    // Anyone can read a restaurant, only authorized
    // users can create or update. Deletes are not allowed.
       match /restaurants/{restaurantId} {
         allow read: if true;
         allow create, update: if request.auth.uid != null;
    }

    // Anyone can read a rating. Only the user who made the rating
    // can delete it. Ratings can never be updated.
    match /restaurants/{restaurantId}/ratings/{ratingId} {
         allow read: if true;
      allow create: if request.auth.uid != null;
         allow delete: if request.resource.data.userId == request.auth.uid;
         allow update: if false;
    }
  }
}
```

6.3.1 –OPTIMIZAR LAS CONSULTAS

Esta quizás sea la parte más sencilla en cuanto a trabajar a la base de datos se refiere una vez se han implementado correctamente las consultas.

Para entender que vamos a hacer debo introducir el concepto de query.
Una query es una consulta la base de datos, es la otra manera de hacer consultas y obtener datos, están enfocadas a devolver cantidades de datos grandes filtrados, por ejemplo todos los movimientos que sean un gasto y no un ingreso de un usuario en concreto, pero no permite recuperar documentos singulares. Por eso dependiendo las necesidades o para qué se trabaja con con referencias a documentos o colecciones o con queries.

Una query en Cloud Firestore puede tener este aspecto:

```
Query capitalCities = db.collection("cities").whereEqualTo("capital", true);
```

Pues bien, para optimizar las consultas solamente tenemos que escribir las queries que deseamos optimizar que tengan varias opciones de filtrado y ejecutarlas.

Algo así, por ejemplo:

```
citiesRef.whereGreaterThanOrEqualTo("state", "CA").whereGreaterThan("population",
100000);
```

De manera que la aplicación nos mostrará un error, que es normal:

```
com.google.firebase.example.fireeats W/Firestore Adapter: onEvent:error
com.google.firebase.firestore.FirebaseFirestoreException: FAILED_PRECONDITION: The
query requires an index. You can create it here:
https://console.firebase.google.com/project/...
```

Esto sucede porque Cloud Firestore no permite consultas complejas que no estén optimizadas para de esta forma evitar cuellos de botella y ofrecer una experiencia fluida.
Daremos click sobre el link que se muestra y se nos mostrará la pantalla siguiente:

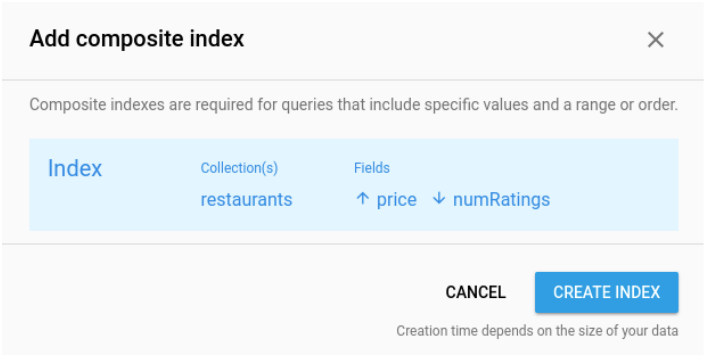


Ilustración 40 - Añadir índices de consultas

Al dar sobre create index nos creará automáticamente el índice de la consultta optimizada.

6.4 ESTRUCTURA DEL PROYECTO

En esta sección se detalla cómo es la estructura interna del proyecto, es decir, cómo están organizados los directorios de archivos que componen el proyecto.

El código fuente de la aplicación se encuentra se encuentra en el directorio del proyecto:

- ***~/HomeBudgetV2/app/src***

Aunque el resto de archivos pertenecientes al proyecto contenidos en *~/HomeBudgetV2/* y *~/HomeBudgetV2/app/* son también necesarios para la construcción del mismo, archivos de build y del gradle de Android Studio.

A continuación, detallamos la organización interna del mismo y comentamos brevemente el contenido de cada una de los archivos.

- ***~/HomeBudgetV2/app/test*** *y* ***~/HomeBudgetV2/app/AndroidTest*****:** contienen archivos relacionados con la definición y ejecución de test del proyecto.

- ***~/HomeBudgetV2/app/src/main/*** **:** Contiene el código fuente y recursos de la aplicación propiamente dicha. Para hacernos una idea más clara se facilita una imagen de la vista del proyecto desde Android Studio.

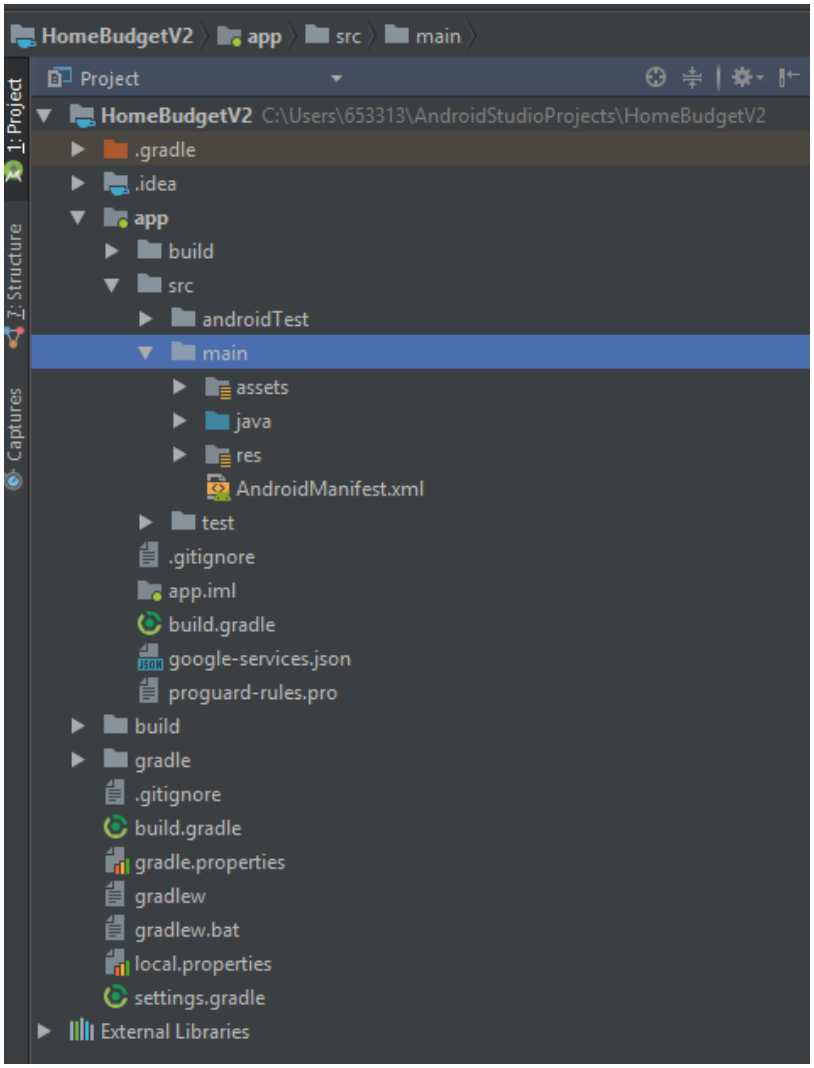


Ilustración 41 - Estructura del proyecto 1

- ***~/HomeBudgetV2/app/src/main/assets:*** encontramos todos los archivos referentes a las animaciones incluídas en la aplicación y en ***~/HomeBudgetV2/app/src/main/res*** todos los archivos referentes a las vistas de la aplicación (archivos .xml) y recursos (textos, imágenes etc…).

- ***~/HomeBudgetV2/app/src/main/java/com.vallecillo.homebudgetv2*:** encontramos finalmente todos nuestros archivos .java de nuestra aplicación y se describirán brevemente a continuación.

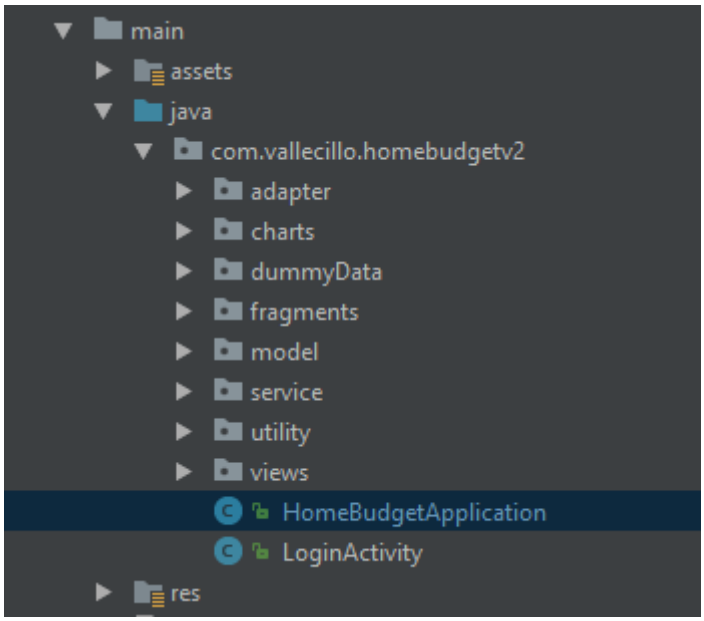


Ilustración 42 - Estructura del proyecto 2

- **HomeBudgetApplication.java:** Se trata de nuestra clase aplicación, clase desde la que controlamos los cambios del estado de la autentificación en toda la aplicación.

- **java/com.vallecillo.homebudgetv2/views:**

  En este directorio se organizan todas clases relativas a la parte lógica de la aplicación y contenedoras de la interfaz gráfica. Además, en MainActivity será contenedora de otros fragmentos que se irán “inflando” en ella con el patrón de diseño “bottomBar”.

  - LoginActivity
  - CreateAccountActivity
  - CreateMoveActivity
  - EditMoveActivity
  - MoveDetailActivity
  - MainActivity

- **java/com.vallecillo.homebudgetv2/utility:**

  En este directorio se encuentras las clases que contienen funciones útiles que se usan en el resto de la aplicación y que se contienen aquí para no replicar el código y para tener un sencillo acceso a las mismas.

  - Utility.java

- **java/com.vallecillo.homebudgetv2/service:**

  En este directorio encontramos las clases relativas al servicio de notificaciones, un servicio es un proceso en segundo plano que capturan eventos o realizan acciones en función de los mismos.

  - MyFirebaseInstanceIDService.java
  - MyFirebaseMessagingService.java

- **java/com.vallecillo.homebudgetv2/model:**

  En este directorio encontramos las clases relativas a la definición de los modelos entidad utilizados en la aplicación y cuya definición es estándar para que Cloud Firestore los procese internamente como objetos en la base de datos, en nuestro caso la definición es la de Profile o usuario y la de Move.

  - Move.java
  - Profile.java

- **java/com.vallecillo.homebudgetv2/fragments:**

  En este directorio encontramos todas las clases que definen nuestros fragments y que usaremos para reemplazarlos sobre nuestra actividad principal MainActivity.java o inflar sobre una actividad en un momento determinado (como sucede con los datepicker y dialog fragments).

  - ChartsFragment.java
  - DatePickerFragment.java
  - DeleteItemDialogFragment.java
  - HomeFragment.java
  - ProfileFragment.java

- **java/com.vallecillo.homebudgetv2/dummyData:**

  En este directorio se encuentra la clase en la que almacenamos datos de muestra para ser usados en la función de generación de movimientos aleatorios que se ha creado con el propósito de facilitar el testeo de la aplicación.

  - Data.java

- **java/com.vallecillo.homebudgetv2/charts:**

  En este directorio encontramos todas las clases referentes a la graficación y creación de los charts, es una implementación customizada de la librería MPAndroidChart.

  - ChartsInterface.java
  - ChartsDrawable.java
  - MyAxisValueFormatter.java
  - MyValueFormatter.java

- **java/com.vallecillo.homebudgetv2/adapter:**

  En este directorio encontramos las clases adapter que hacen de puente entre los datos y las vistas contenidas en listas. En nuestro caso es un adapter para el viewpager usado en MainActivity y que sirve de contenedor para inflar los principales fragments.

  - ViewPagerAdapter.java

- **java/com.vallecillo.homebudgetv2/fragments:**

  En este directorio encontramos todas las clases que definen nuestros fragments y que usaremos para reemplazarlos sobre nuestra actividad principal MainActivity.java o inflar sobre una actividad en un momento determinado (como sucede con los datepicker y dialog fragments).

  - ChartsFragment.java
  - DatePickerFragment.java
  - DeleteItemDialogFragment
  - HomeFragment
  - ProfileFragment

Por último, vamos a describir brevemente el contenido de la carpeta **res** anteriormente mencionada:

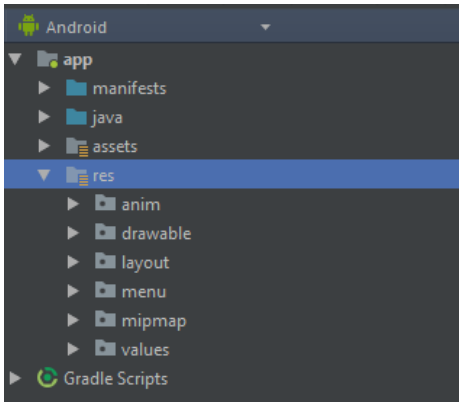


Ilustración 43 - Estructura del proyecto 3

- **Res/**
  - **anim**: En este directorio se almacenan las animaciones de transición utilizadas en la transición entre activities (No confundir con animaciones gráficas que ya explicamos anteriormente que se encontraban en “assets” a la altura de nuestra carpeta java).

  - **drawable**: En este directorio se encuentran todos los recursos gráficos que utiliza la aplicación a excepción de los iconos y animaciones gráficas.

  - **layout**: En este directorio se encuentran todas las vistas de la aplicación a excepción de las de los menús, éstas tienen formato xml.

  - **menú**: En esta carpeta podemos encontrar las vistas referentes a los menús de la aplicación, también en formato xml.

  - **mipmap**: En este directorio concentramos toda la iconografía empleada en la aplicación.

  - **values**: Es una de las carpetas más importantes pues en ésta se encuentran los archivos de definición de cadenas de texto, colores, estilos y dimensiones.

- **AndroidManifest.xml**: Este es el archivo dónde se recoge toda la información general sobre la aplicación que se sirve al sistema operativo, describe los componentes de la aplicación, las clases que implementa cada uno de ellos, permisos…

CAPÍTULO 7

PRUEBAS

7.1 PRUEBAS DE CAJA BLANCA

Las pruebas de caja negra consisten en la prueba individual de cada uno de las funcionalidades internas. Son principalmente pruebas unitarias y de integración, en ellas se comprueban todos los posibles caminos lógicos que pueden tomar los métodos o los diferentes tipos de valores que se pueden pasar.

Todas estas pruebas se realizan durante el tiempo de implementación y no a posteriori cómo las de caja negra.

Las pruebas de caja blanca que se han realizado a lo largo de este proyecto se resumen en:

- Comprobación de la validación de entrada de datos en cada uno de los formularios.
- Comprobación del comportamiento del sistema sin conexión.
- Comprobación de las salidas del sistema tras la introducción forzosa de datos truncados en la base de datos.
- Comprobación de la comunicación del sistema en los diferentes escenarios posibles.
- Comprobación del funcionamiento de los servicios de manera aislada.
- Comprobación del control de la sesión.
- Comprobación de la concordancia entre las entradas de datos y la información guardada en Cloud Firestore.
- Comprobación del comportamiento de la caché de la aplicación.
- Comprobación individual de los comportamientos no dependientes de nuestro sistema (funcionalidades ofrecidas en la consola de Firebase).

7.2 PRUEBAS DE CAJA NEGRA

Las pruebas de caja negra son un método de prueba de software en el que la estructura/diseño/implementación interna del ítem que se está probando no es conocida por el probador. Estas pruebas pueden ser funcionales o no funcionales, aunque generalmente funcionales. Esa es la diferencia principal respecto a las pruebas de caja blanca.

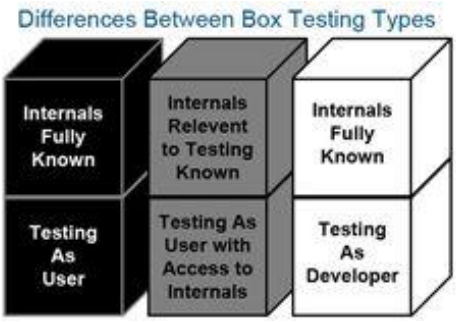


Ilustración 44 - Caja blanca vs caja negra

Por ello, para la realización de estas pruebas se ha solicitado a compañeros de trabajo cuyo conocimiento del proyecto era nulo, que sean ellos quien prueben el sistema limitando mi participación a la anotación de los resultados. Todas estas pruebas han sido realizadas obviamente después de haber sido implementadas las funcionalidades en el sistema.

Veamos una descripción de las pruebas que se han realizado:

Prueba de registro	
Objetivo	Realizar el registro del usuario en el sistema.
Precondiciones	No tener una sesión activa.
Entrada de datos	Correo: danielvallecillo@gmail.com Nombre: Daniel Apellidos: Vallecillo Fecha de nacimiento: 20/09/1993 País: España Constraseña: informatica Confirmación: informatica
Postcondiciones esperadas	1. Crearse la cuenta. 2. Recoger los datos y crear perfil. 3. Realizar automaticamente inicio de sesión con la nueva cuenta. 4. Abrir la pantalla principal de la aplicación.
Secuencia	1. Rellenar formulario 2. Realizar validaciones 3. Enviar información al servicio 4. Recibir respuesta 5. Mostrar aviso visual del resultado 6. Realizar acciones consecuentes
Resultado	El resultado fue satisfactorio aunque en conexiones con baja latencia existe un bug documentado[10] en la versión 11.8 de Firestore (Recordemos que es un servicio aún en fase Beta).

Tabla 47 - Prueba de registro

[10] https://stackoverflow.com/questions/47973887/firebase-error-w-bichannelgoogleapi-firebaseauth-getgoogleapiformethod-r

Prueba de inicio de sesión	
Objetivo	Realizar el inicio de sesión del usuario en el sistema.
Precondiciones	No tener una sesión activa.
Entrada de datos	Correo: danielvallecillo@gmail.com Contraseña: informatica
Postcondiciones esperadas	1. Completar el inicio de seisón. 2. Ir a la pantalla principal de la aplicación.
Secuencia	1. Rellenar formulario 2. Realizar validaciones 3. Enviar información al servicio 4. Recibir respuesta 5. Mostrar aviso visual del resultado 6. Realizar acciones consecuentes
Resultado	El resultado fue satisfactorio.

Tabla 48 - Prueba de inicio de sesión

Truncar inicio de sesión	
Objetivo	Comprobar el funcionamiento de las validaciones del sistema cuando se realiza el inicio de sesión
Precondiciones	No tener una sesión activa.
Entrada de datos	Correo: danielvallecillo@gmail.com Contraseña:
Postcondiciones esperadas	1. Las validaciones que se realizan en el cliente impide el envio de datos al servicio. 2. Se muestra aviso visual.
Secuencia	1. Rellenar formulario 2. Realizar validaciones 3. Mostrar aviso visual del resultado
Resultado	El resultado fue satisfactorio.

Tabla 49 - Prueba truncar inicio de sesión

Truncar el registro	
Objetivo	Comprobar el funcionamiento de las validaciones del sistema cuando se realiza el registro.
Precondiciones	No tener una sesión activa.
Entrada de datos	Correo: danielvallecillo@gmail Nombre: Daniel Apellidos: Vallecillo Fecha de nacimiento: 20/09/1993 País: España Constraseña: informatica Confirmación: informatica
Postcondiciones esperadas	1. Las validaciones que se realizan en el cliente impide el envio de datos al servicio. 2. Se muestra aviso visual.
Secuencia	1. Rellenar formulario 2. Realizar validaciones 3. Mostrar aviso visual del resultado
Resultado	El resultado fue satisfactorio.

Tabla 50 - Prueba truncar registro

Visualizar lista de movimientos	
Objetivo	Comprobar el funcionamiento del recyclerview que contiene la lista de movimientos.
Precondiciones	Tener una sesión activa y encontrarse en el fragmento de “Home”.
Entrada de datos	Ninguna
Postcondiciones esperadas	Mostrar la lista de movimientos
Secuencia	1. Iniciar sesión o completar registro o entrar en la aplicación con una sesión activa.
Resultado	El resultado fue satisfactorio.

Tabla 51 - Prueba visualizar movimientos

Eliminar movimiento	
Objetivo	Comprobar la eliminación de movimientos del sistema.
Precondiciones	Tener una sesión activa y encontrarse en el fragmento de “Home”.
Entrada de datos	Ninguna
Postcondiciones esperadas	El sistema elimina el movimiento.
Secuencia	1. Realizar una pulsación larga sobre un elemento de la lista. 2. Aceptar la eliminación del movimiento desde el dialogo flotante.
Resultado	El resultado fue satisfactorio.

Tabla 52 - Prueba eliminar movimiento

Añadir movimiento	
Objetivo	Comprobar el funcionamiento del sistema cuando se intenta añadir un movimiento.
Precondiciones	Tener una sesión activa y encontrarse en el fragmento de “Home”.
Entrada de datos	Categoría: Transport. Concept: Bus Alsa. Description: Trayecto Valladolid – Segovia. Date: 12/02/2018 Amount: 12.95 Expense: true
Postcondiciones esperadas	El sistema añade el movimiento.
Secuencia	1. Iniciar sesión o completar registro o entrar en la aplicación con una sesión activa. 2. Presionar el botón flotante. 3. Seleccionar una categoría. 4. Rellenar el formulario. 5. Pulsar el botón de enviar.
Resultado	El resultado fue satisfactorio.

Tabla 53 - Prueba añadir movimiento

Truncar añadir movimiento	
Objetivo	Comprobar el funcionamiento del sistema cuando se intenta añadir un movimiento.
Precondiciones	Tener una sesión activa y encontrarse en el fragmento de “Home”.
Entrada de datos	Categoría: Transport. Concept: Description: Trayecto Valladolid – Segovia. Date: 12/02/2018 Amount: 12.9544 Expense: true
Postcondiciones esperadas	Las validaciones impiden que el movimiento se añada.
Secuencia	1. Iniciar sesión o completar registro o entrar en la aplicación con una sesión activa. 2. Presionar el botón flotante. 3. Seleccionar una categoría. 4. Rellenar el formulario. 5. Pulsar el botón de enviar. 6. Mostrar aviso visual del resultado.
Resultado	El resultado fue satisfactorio.

Tabla 54 - Prueba truncar añadir movimiento

Añadir movimientos aleatorios	
Objetivo	Comprobar el funcionamiento del sistema cuando se intenta generar movimientos aleatorios.
Precondiciones	Tener una sesión activa y encontrarse en el fragmento de “Home”.
Entrada de datos	Ninguna.
Postcondiciones esperadas	El sistema añade el movimiento.
Secuencia	1. Iniciar sesión o completar registro o entrar en la aplicación con una sesión activa. 2. Presionar el botón flotante. 3. Seleccionar la categoría movimientos aleatorios.
Resultado	El resultado fue satisfactorio.

Tabla 55 - Prueba añadir movimientos aleatorios

Visualizar información del perfil	
Objetivo	Comprobar el funcionamiento de la función que nos permite visualizar el perfil.
Precondiciones	Tener una sesión activa y encontrarse en el fragmento de “Profile”.
Entrada de datos	Ninguna
Postcondiciones esperadas	Mostrar el perfil del usuario.
Secuencia	1. Iniciar sesión o completar registro o entrar en la aplicación con una sesión activa, hacer dos veces swipe hacia la derecha dos veces o pulsar en el boton de la bottom bar “profile”.
Resultado	El resultado fue satisfactorio.

Tabla 56 - Prueba visualizar información del perfil

Visualizar detalles del movimiento	
Objetivo	Comprobar el funcionamiento del sistema cuando se accede a los detalles de un movimiento.
Precondiciones	Tener una sesión activa y encontrarse en el fragmento de “Home”.
Entrada de datos	Ninguna.
Postcondiciones esperadas	El sistema muestra los detalles del movimiento seleccionado.
Secuencia	1. Iniciar sesión o completar registro o entrar en la aplicación con una sesión activa. 2. Presionar sobre un movimiento del recyclerview. 3. El sistema muestra la información del movimiento.
Resultado	El resultado fue satisfactorio.

Tabla 57 - Prueba visualizar detalles del movimiento

Editar movimiento	
Objetivo	Comprobar el funcionamiento del sistema cuando se intenta editar un movimiento que ha sido añadido previamente.
Precondiciones	Tener una sesión activa y encontrarse en el fragmento de “Home”.
Entrada de datos	Opcional
Postcondiciones esperadas	El sistema muestra los detalles del movimiento seleccionado.
Secuencia	1. Iniciar sesión o completar registro o entrar en la aplicación con una sesión activa. 2. Presionar sobre un movimiento del recyclerview. 3. El sistema muestra la información del movimiento. 4. Pulsar sobre el botón de edición. 5. El sistema muestra una pantalla con los detalles esa vez sobre campos editables. 6. Se editan los campos. 7. Presionar finalizar edición. 8. El sistema te lleva a la vista de Home dónde aparece el movimiento actualizado en caso de haber sido modificado.
Resultado	El resultado fue satisfactorio.

Tabla 58 - Prueba editar movimiento

Truncar la edición movimiento	
Objetivo	Comprobar el funcionamiento del sistema cuando se intenta editar un movimiento que ha sido añadido previamente.
Precondiciones	Tener una sesión activa y encontrarse en el fragmento de “Home”.
Entrada de datos	Categoría: Concept: Bus. Description: Trayecto Valladolid – Segovia. Date: 12/02/2018 Amount: 12.95erv Expense: true
Postcondiciones esperadas	El sistema realiza las validaciones y no permite el envio de la información modificada.
Secuencia	1. Iniciar sesión o completar registro o entrar en la aplicación con una sesión activa. 2. Presionar sobre un movimiento del recyclerview. 3. El sistema muestra la información del movimiento. 4. Pulsar sobre el botón de edición. 5. El sistema muestra una pantalla con los detalles esa vez sobre campos editables. 6. Se editan los campos. 7. Presionar finalizar edición. 8. El sistema valida los campos. 9. El sistema muestra los errores.
Resultado	El resultado fue satisfactorio.

Tabla 59 - Prueba truncar la edición del movimiento

Visualizar estadísticas de los gastos	
Objetivo	Comprobar el funcionamiento del sistema cuando se desea visualizar las estadísticas de los gastos.
Precondiciones	Tener una sesión activa y encontrarse en el fragmento de “Charts”.
Entrada de datos	Ninguna.
Postcondiciones esperadas	El sistema muestra la información de las estadísticas de los gastos en forma de gráficos.
Secuencia	1. Iniciar sesión o completar registro o entrar en la aplicación con una sesión activa. 2. Hacer swipe hacia la derecha o presionar el botón correspondiente en la bottom bar. 3. El sistema muestra la información de los gastos en forma de gráficos.
Resultado	El resultado fue satisfactorio.

Tabla 60 - Prueba visualizar estadísticas de los gastos

Actualizar estadísticas de los gastos	
Objetivo	Comprobar el funcionamiento del sistema cuando se desea actualizar las estadísticas de los gastos.
Precondiciones	Tener una sesión activa y encontrarse en el fragmento de “Charts”.
Entrada de datos	Ninguna.
Postcondiciones esperadas	El sistema actualiza la información de las estadísticas de los gastos en forma de gráficos.
Secuencia	1. Iniciar sesión o completar registro o entrar en la aplicación con una sesión activa. 2. Eliminar o editar movimientos. 3. Hacer swipe hacia la derecha o presionar el botón correspondiente en la bottom bar. 4. El sistema muestra la información de los gastos en forma de gráficos. 5. Presionar sobre el botón de actualización 6. El sistema genera los gráficos con la información actualizada.
Resultado	El resultado fue satisfactorio.

Tabla 61 - Prueba actualizar estadísticas

CAPÍTULO 8

MANUALES

8.1 MANUAL DE INSTALACIÓN

Puesto que el backend se consume como servicio, para el uso del sistema solamente será necesario la instalación de la aplicación. Para ello, el proceso no puede ser más simple, se proporciona un archivo apk, que es un archivo empaquetado en un formato apto para ser instalado en un dispositivo Android.

De manera que para instalar la aplicación, solamente será necesario ejecutar este archivo con extensión .apk en nuestro teléfono.

Para ello, antes de nada se debe de aclarar que es necesario habilitar o tener habilitado en el sistema la opción que nos permite instalar aplicaciones no provenientes de Google Play.

Ajustes → Seguridad → Orígenes desconocidos , ésta es la opción que debemos de activar o tener activada.

Veámoslo de una manera un poco más gráfica:

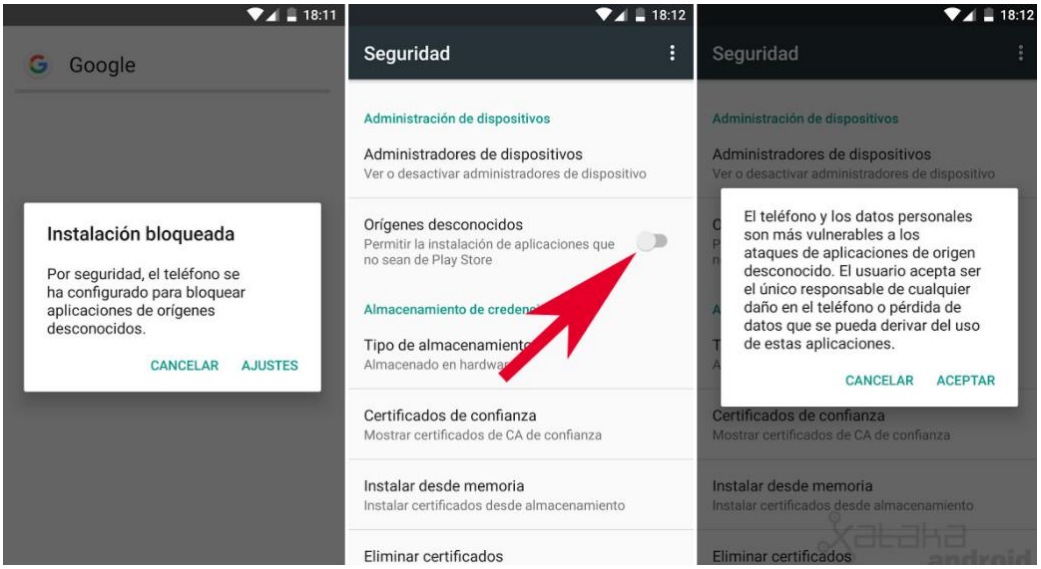


Ilustración 45 - Activación origenes desconocidos

Una vez ejecutado este paso, mediante conexión USB o una tarjeta SD, se puede introducir el archivo en el Smartphone y simplemente ser ejecutada para su instalación.

Es posible que sea necesario también actualizar Google Services si no estuviesen actualizados. Eso dependerá del dispositivo. Esto sucede porque la aplicación utiliza servicios de Google.

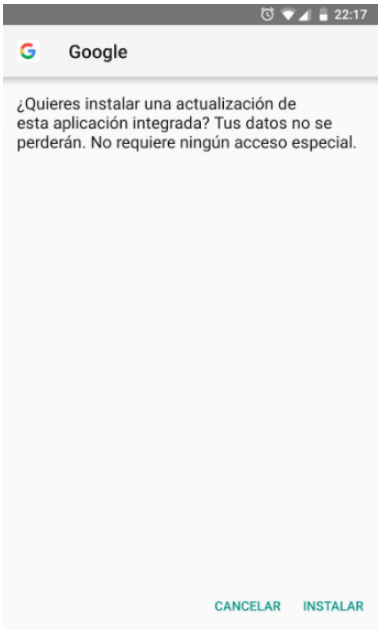


Ilustración 46 - Aceptar instalación

Eso sería todo en lo que a la instalación del sistema se refiere.

8.2 MANUAL DE USUARIO

En esta sección se va a cubrir lo relativo al uso de la aplicación, ya que todo lo que respecta a la herramienta de administración de Firebase será tratado en el siguiente capítulo.

- Inicio de sesión: Deberemos de proveer nuestros credenciales y pulsar sign in.

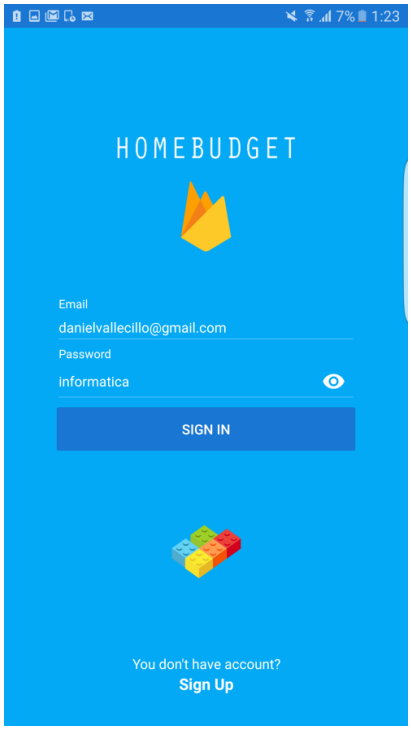


Ilustración 47 - Manual de usuario inicio de sesión

- Registro en el sistema: Aunque se proporcionan credenciales de acceso ya preparados, se puede registrar en el sistema completando los campos del formulario, importante hacer caso a los avisos visuales de las validaciones. El formulario requiere completar la información de todos sus campos.

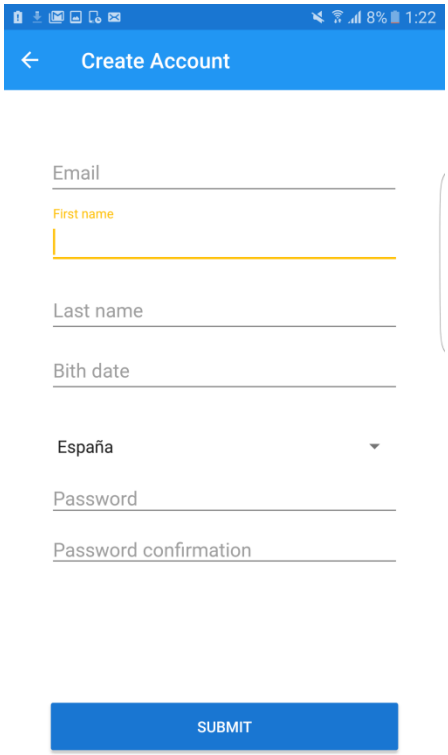


Ilustración 48 - Manual de usuario registro

- Añadir movimiento: Una vez iniciada la sesión deberemos pulsar sobre el botón flotante de la pantalla de home para que se nos muestren las categorías disponibles.

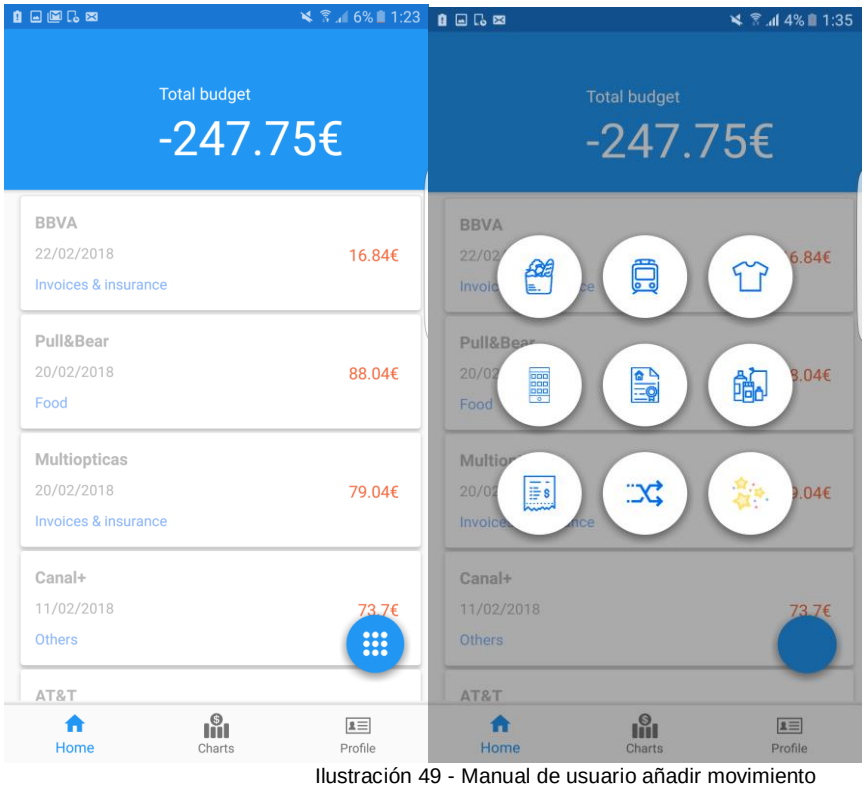


Ilustración 49 - Manual de usuario añadir movimiento

A continuación, tras seleccionar la categoría correspondiente se nos mostrará una pantalla formulario dónde deberemos dar entrada a los datos requeridos para registrar el movimiento en el sistema.

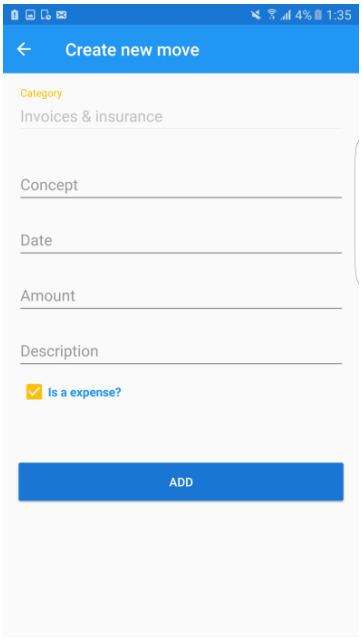


Ilustración 50 - Manual de usuario formulario añadir movimiento

- Eliminar movimiento: Se deberá hacer una pulsación larga sobre cualquiera de los movimientos de la lista y darle a aceptar al dialogo que aparece.

- Ver detalles: Se deberá hacer una pulsación simple sobre cualquiera de los movimientos de la lista.

- Editar movimiento: Una vez situados sobre la pantalla de detalles, se deberá pulsar sobre el icono de la pluma (edición) y se nos mostrará la pantalla que nos permitirá editar los campos.

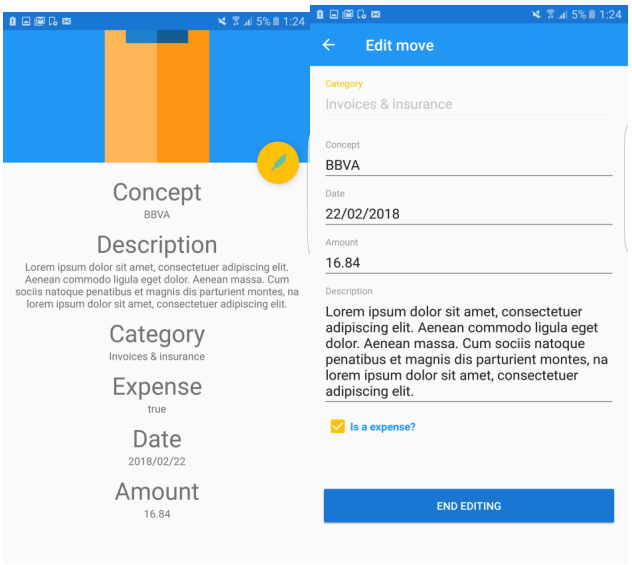


Ilustración 51 - Manual de usuario editar movimiento

8.3 MANUAL DE ADMINISTRADOR

En esta sección se cubre la parte referente al uso de la herramienta de administrador.

- Acceder al proyecto de Firebase: Primeramente deberemos acceder desde un navegador o navegador móvil a la dirección https://firebase.google.com/ pulsaremos sobre acceso y deberemos iniciar sesión con la cuenta de google asociada a este proyecto que se ha creado exclusivamente para este fin.

  Email: proyectodanielvallecillo@gmail.com
  Contraseña: proyectodaniel

  Estos credenciales serán revocados de acceso tras la defensa de este TFG.

  Una vez iniciada la sesión pulsamos sobre “Ir a la consola”

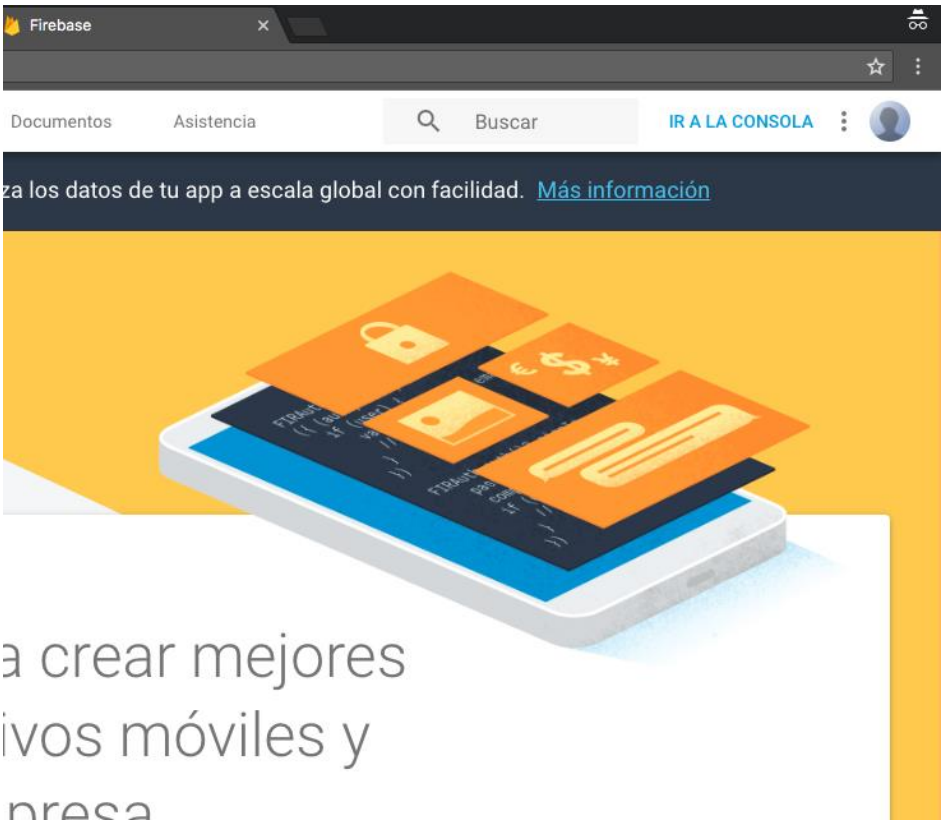


Ilustración 52 - Acceso Firebase

Nos encontramos ya dentro de Firebase y pulsamos sobre el proyecto Homebudget V2 para acceder a nuestro proyecto.

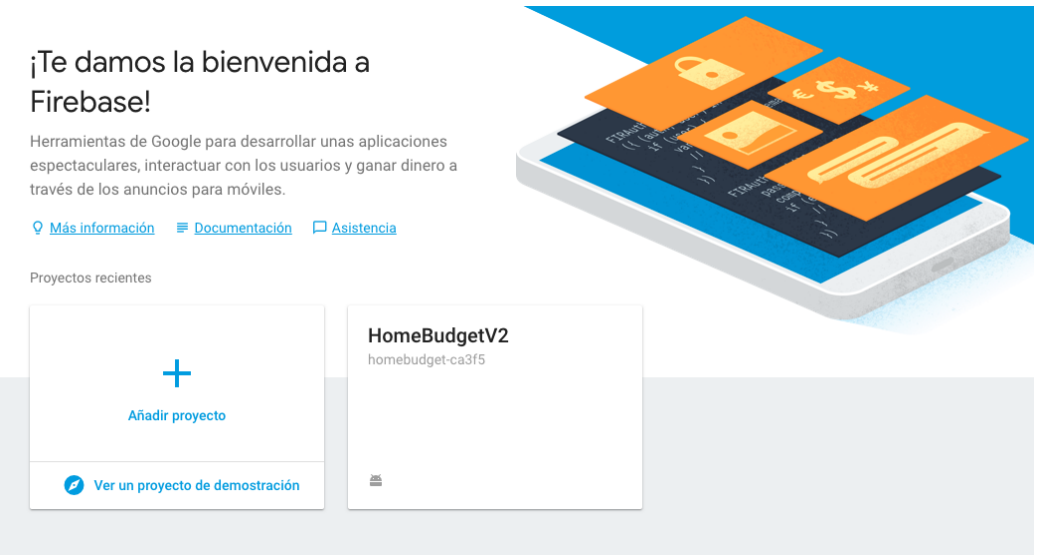


Ilustración 53 - Vista de proyecto Firebase

- Acceder a las funcionalidades principales de Firebase implementadas en el sistema:

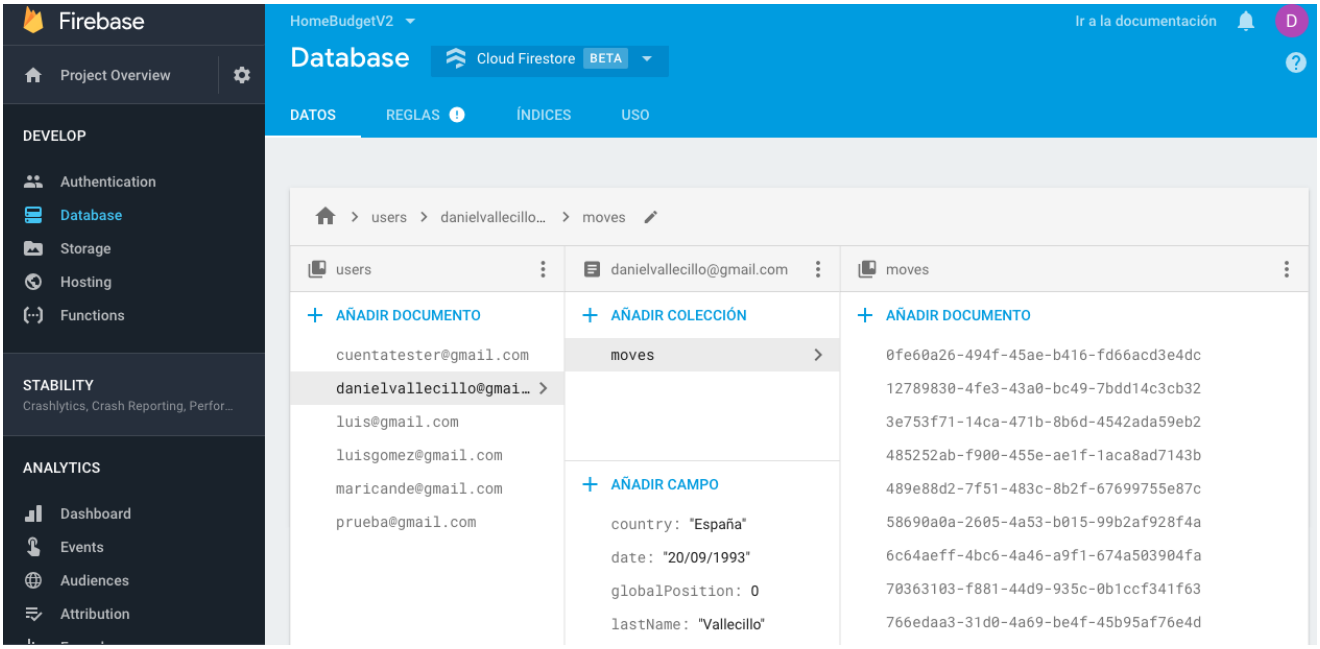


Ilustración 54 - Vista panel authentication

En el panel lateral se encuentran los diferentes servicios ofrecidos, en nuestro caso encontraremos información en los apartados “Database”, “Authentication”, “Dashboard” y “Notifications”. Desde ahí podremos ejecutar y visualizar todas las opciones que han sido implementadas.

En el panel de Database: Visualizar, añadir, modificar y eliminar cualquier tipo de información, añadir reglas a la base de datos y crear índices personalizadas para optimizar las consultas complejas.

En el panel de Dashboard: Visualizar e interactuar con todas las estadísticas que han sido recogidas en la aplicación.

En el panel de Autentication: Eliminar, inhabilitar y habilitar cuentas, modificar las plantillas de emails.

En el panel de Notifications: Enviar notificaciones y ver información relativa a las mismas.

CAPÍTULO 9

CONCLUSIONES

9.1 CONCLUSIONES

A lo largo de la implementación del sistema se han encontrado numerosos inconvenientes que en su mayoría terminaron resolviéndose con horas de investigación, estudio o simplemente prueba y error.

En su mayoría, estas dificultades vinieron dadas en la relación a la base datos Cloud Firestore. Pese a que la simple inserción o consulta de datos es muy sencilla, hacer que la base de datos fuese real time y ofreciese soporte sin conexión a internet fue una tarea ardua por la falta de ejemplos ya implementados. Recordemos que aún está en fase Beta y la única documentación de momento disponible es la ofrecida por Google.

Otra pequeña dificultad fue la adaptación al uso de algunas librerías externas integradas en la aplicación, que pese a tener un uso relativamente sencillo, al estar aún en desarrollo o mantenidas por la comunidad, los cambios en la documentación entre las diferentes versiones y en los métodos de llamada para algunas funciones supusieron un contratiempo.

Por último, aunque la autentificación con Facebook está implementada (pero no disponible para esta versión), se presentó un problema al no tener aún un sistema de unificación de cuentas.

A pesar de todo lo anterior, puedo concluir que ha sido un proyecto realmente satisfactorio, tanto a nivel personal, por lo duro del camino, por el sacrificio extra después del horario laboral para poder realizarlo y sobre todo porque me ha ayudado indirectamente a conocerme mejor a mi mismo. Son muchas horas pensando en soluciones de problemas que yo mismo me había creado o de las que no veía la salida cuando la tenía ante mis ojos.

También a nivel académico, pues me ha brindado la oportunidad de investigar, estudiar, afianzar los conocimientos básicos de la programación Android que tenía y sobre ha creado la inquietud de seguir querer aprendiendo esta tecnología y creer un poco más en mi mismo. Eso es oro.

9.2 MEJORAS FUTURAS

Si bien es cierto que cuanto más avanzado estaba el proyecto y más conocimiento había sido adquirido, el abanico de opciones se multiplicaba y muchas grandes ideas tuvieron que ser desechadas, por lo avanzado del mismo y por la falta del tiempo que habría sido necesario para llevarlas a cabo. Algunas de ellas son las siguientes:

- En el proyecto, aunque no visible, se ha implementado la lógica necesaria para la generación de gráficos de otros cuatro tipos diferentes de datos. Una idea que siempre ha rondado mi cabeza es encontrar una utilidad para cada uno de ellos, pues no todos son válidos para mostrar cualquier tipo de información.

- Integrar la unificación de cuentas para poder utilizar tanto la autenticación con Facebook como la de correo-contraseña indiferentemente en la misma cuenta.

- Declarar más eventos para capturar mediante Google Analytics y obtener así una información más precisa sobre el comportamiento de los usuarios.

- Integrar Firebase Predictions para aprovechar al máximo toda la información recogida por el servicio de Google Analytics.

- Implementar añadir pagos futuros y notificaciones que nos lo recuerden.

CAPÍTULO 10

REFERENCIAS

10.1 BIBLIOGRAFÍA

- Gargenta, M. (2011). *Learning Android*. Sebastopol, Calif.: O'Reilly
- Ableson, F. Collins, C. and Sen, R. (2011). *Android*. Madrid: Anaya Multimedia
- Mark Lawrence Murphy (2008). The Busy Coders Guide to Android Development: CommonsWare.
- Ed Burnetter (2010). Hello Android: Introducing Google's Mobile Development Platform.: Pragmatic Bookshelf.

10.2 WEBGRAFÍA

- Librerías que debes conocer si eres desarrollador Android. (n.d.). Retrieved December 12, 2017, from http://bemobile.es/blog/2016/10/10-librerias-que-debes-conocer-si-eres-desarrollador-android/
- Custom Activity transition when up button is pressed - Stack Overflow. (n.d.). Retrieved February 7, 2018, from https://stackoverflow.com/questions/40882733/custom-activity-transition-when-up-button-is-pressed
- *android-guidelines: Architecture and code guidelines we use at ribot when developing for Android*. (2017). ribot. Retrieved from https://github.com/ribot/android-guidelines (Original work published 2014)
- Announcing Architecture Components 1.0 Stable. (n.d.). Retrieved November 16, 2017, from https://android-developers.googleblog.com/2017/11/announcing-architecture-components-10.html
- Arquitectura Clean de Uncle Bob. (2016, January 26). Retrieved November 17, 2017, from http://jmperezramos.net/programacion/arquitectura-clean-de-uncle-bob/
- Bauer, V., & Bauer, V. (n.d.). RecyclerViewTemplate | Android-Arsenal.com. Retrieved September 12, 2017, from https://android-arsenal.com/details/1/6204
- cadenas, victor garibay. (2016a, July 6). Consumiendo una API con Retrofit 2 en Android. Retrieved December 12, 2017, from https://stories.devacademy.la/mi-primer-app-con-retrofit-y-android-ac61a8954a2c
- cadenas, victor garibay. (2016b, July 6). Consumiendo una API con Retrofit 2 en Android. Retrieved December 12, 2017, from https://stories.devacademy.la/mi-primer-app-con-retrofit-y-android-ac61a8954a2c
- Clean Architecture. The Bemobile Way. (n.d.). Retrieved December 12, 2017, from http://bemobile.es/blog/2016/11/clean-architecture-the-bemobile-way/

- Gallegos, D. (2016, August 1). How to use GitHub like a proper human being. Retrieved December 12, 2017, from https://stories.devacademy.la/how-to-use-github-like-a-proper-human-being-1a9c895c4e13

- Intents y navegación entre actividades. (n.d.). Retrieved February 9, 2018, from http://www.jtech.ua.es/dadm/restringido/android/sesion02-apuntes.html

- Introducción a los componentes de arquitectura Android: go Clean! (2017, June 22). Retrieved December 7, 2017, from https://www.paradigmadigital.com/dev/introduccion-los-componentes-arquitectura-android-go-clean/

- java - ¿Cómo prevenir ir a Activity anterior en Android? - Stack Overflow en español. (n.d.). Retrieved February 6, 2018, from https://es.stackoverflow.com/questions/3427/cómo-prevenir-ir-a-activity-anterior-en-android

- Leiva, A. (2017). *androidmvp: MVP Android Example*. Retrieved from https://github.com/antoniolg/androidmvp (Original work published 2014)

- Miličić, D. (2016, February 3). A detailed guide on developing Android apps using the Clean Architecture pattern. Retrieved December 12, 2017, from https://medium.com/@dmilicic/a-detailed-guide-on-developing-android-apps-using-the-clean-architecture-pattern-d38d71e94029

- Molina, C. (2016, June 27). Analítica móvil: qué medir en una app y con qué herramientas. Retrieved December 18, 2017, from https://mediossociales.es/analitica-herramientas-app/

- NoSQL vs SQL: Principales diferencias y cuándo elegir cada una de ellas. (2015, November 18). Retrieved February 6, 2018, from https://blog.pandorafms.org/es/nosql-vs-sql-diferencias-y-cuando-elegir-cada-una/

- Putting it All Together: Wireframing the Example App | Android Developers. (n.d.). Retrieved September 12, 2017, from https://developer.android.com/training/design-navigation/wireframing.html

- *quickstart-android: Firebase Quickstart Samples for Android*. (2018). Firebase. Retrieved from https://github.com/firebase/quickstart-android (Original work published 2016)

- Rodríguez, T. (2015, April 23). Tendencias de desarrollo en Android y qué mejorar de la plataforma: los ponentes de la Droidcon Spain 2015 nos dan su visión. Retrieved December 12, 2017, from https://www.genbetadev.com/desarrollo-aplicaciones-moviles/tendencias-de-desarrollo-en-android-y-que-mejorar-de-la-plataforma-los-ponentes-de-la-droidcon-spain-2015-nos-dan-su-vision

- Taberski, K. (n.d.). Four Steps to a Successful Business Process Integration Solution. Retrieved September 13, 2017, from http://www.ingrammicroadvisor.com/unified-communications-and-collaboration/four-steps-to-a-successful-business-process-integration-solution

- The Clean Architecture | 8th Light. (n.d.). Retrieved December 12, 2017, from https://8thlight.com/blog/uncle-bob/2012/08/13/the-clean-architecture.html

- Wach, L. (2018). *hellocharts-android: Charts/graphs library for Android compatible with API 8+, several chart types with support for scaling, scrolling and animations*. Retrieved from https://github.com/lecho/hellocharts-android (Original work published 2014)

10.3 REPOSITORIOS DE CÓDIGO UTILIZADOS

Los repositorios de código utilizados durante este proyecto corresponden a librerías de terceros, ejemplos de Firebase proporcionados por Google a través de su repositorio oficial y proyectos de código abierto.

- *quickstart-android: Firebase Quickstart Samples for Android*. (2018). Firebase. Retrieved from https://github.com/firebase/quickstart-android (Original work published 2016)
- *android-guidelines: Architecture and code guidelines we use at ribot when developing for Android*. (2017). ribot. Retrieved from https://github.com/ribot/android-guidelines (Original work published 2014)
- *BoomMenu: A menu which can ... BOOM! - Android*. Retrieved from https://github.com/danielvallecillo77/BoomMenu(Original work published 2017)
- *Platzigram.* Retrieved from https://github.com/danielvallecillo77/Platzigram-1 (Original work published 2017)
- *lottie-android: Render After Effects animations natively on Android and iOS, Web, and React Native*. Retrieved from https://github.com/danielvallecillo77/lottie-android (Original work published 2017)
- *android-saripaar: UI form validation library for Android*. Retrieved from https://github.com/danielvallecillo77/android-saripaar (Original work published 2018)
- *MPAndroidChart: A powerful Android chart view / graph view library, supporting line-bar- pie- radar- bubble- and candlestick charts as well as scaling, dragging and animations*. Retrieved from https://github.com/danielvallecillo77/MPAndroidChart(Original work published 2018)
- *FirebaseUI-Android: Optimized UI components for Firebase*. (2018). Firebase. Retrieved from https://github.com/firebase/FirebaseUI-Android (Original work published 2015)
- Leiva, A. (2017). *androidmvp: MVP Android Example*. Retrieved from https://github.com/antoniolg/androidmvp (Original work published 2014)