UNIVERSITAT POLITÈCNICA DE VALÈNCIA

# Título del Proyecto:

**Contadroid**

## Autor:

Cupeiro Durán, Carlos

## Director:

Barroso Carmona, Jorge Juan

**TESINA PARA LA OBTENCIÓN DEL TÍTULO DE:**

**Máster en Desarrollo de Aplicaciones sobre Dispositivos Móviles**

**Septiembre de 2017**

# Contenido

Introducción ..... 1
Descripción del problema ..... 1
Objetivos ..... 2
Motivación ..... 3
Tecnologías utilizadas ..... 4
Arquitectura de la aplicación ..... 5
Esquema del diseño ..... 5
Capa de Presentación ..... 7
Capa de Dominio ..... 7
Capa de Datos ..... 7
Modelo de datos ..... 9
Vistas ..... 11
Vista de Mes Actual ..... 11
Vista de Resumen de gastos anual ..... 14
Vista de Plantilla de gastos ..... 15
Vista de Detalle de un gasto ..... 16
Vista del Menú de navegación ..... 17
Esquema de Navegación ..... 18
Conclusiones ..... 19
Bibliografía ..... 20
Libros ..... 20
Páginas Web ..... 20
Video Youtube ..... 22
Anexos ..... 24
Código fuente en GitHub ..... 24

# Tabla de Ilustraciones

Ilustración 1- Esquema de MVC y MVP ..... 5
Ilustración 2. Arquitectura Clean ..... 6
Ilustración 3- Dibujo de la capas de la aplicación ..... 7
Ilustración 4- Organización de la Arquitectura del proyecto ..... 8
Ilustración 5 - Velocidad en las querys de Realm frente a competidores ..... 9
Ilustración 6 - Gráfica de inserción de registros en Realm frente a competidores ..... 10
Ilustración 7- Pestaña de Gastos Realizados ..... 11
Ilustración 8 - Pestaña de Gastos Pendientes ..... 11
Ilustración 9 - Dialogo de pulsación larga ..... 13
Ilustración 10 - Ventana de Resumen anual ..... 14
Ilustración 11 - Ventana de Plantilla de gastos ..... 15
Ilustración 12 - Diálogo para importar la plantilla ..... 15
Ilustración 13 - Ventana de detalle de un gasto ..... 16
Ilustración 14 - Menú de navegación ..... 17
Ilustración 15 - Dialogo de cierre de la aplicación ..... 17
Ilustración 16 - Navegación entre vistas ..... 18

# Introducción

En las siguientes líneas servirán para introducir en la temática del proyecto donde se tratará de describir el problema sobre el que se sustenta la idea del proyecto, los objetivos que se buscan cumplir y la motivación para la realización de dicho proyecto.

## Descripción del problema

En la actualidad nos encontramos en una situación post-crisis económica en la mayoría de países Europeos. Debido a esta situación aún se está intentado salir a flote y llegar un estado de bienestar en el cuál todos los europeos puedan vivir con unas comodidades mínimas que aseguren una vida completa y feliz.

Esta situación de intentar mejorar nos sitúa en un estado aún con carencias, dichas carencias se notan mucho en los sueldos de la mayoría de los europeos. La gente se ve obligada a realizar malabares con sus ingresos para poder llegar a final de mes o para poder ahorrar un poco pensando en su disfrute vacacional.

Por otro lado nos encontramos en la era tecnológica donde es muy habitual encontrar a todo el mundo con dispositivos móviles, en su gran mayoría Smartphones. Somos una sociedad conectada y quizás dependiente de nuestros dispositivos tecnológicos.

Debido a estas razones es muy común que nos ayudemos de aplicaciones instaladas en nuestros dispositivos para llevar las cuentas de la casa, personales y de ahorro. Pero el afán de conseguir la aplicación completa y que todos quieran descargar origina que las ofertas del mercado sean aplicaciones demasiado complicadas y engorrosas para el usuario común, convirtiéndose en aplicaciones de uso temporal y que acaban siendo eliminadas.

Sería muy interesante poder tener una aplicación sencilla que nos ayudase a llevar los gastos e ingresos mensuales para poder tener una organización correcta y sencilla. Y que fuese cómoda en su uso para que animase al usuario a utilizarla diariamente sin sentirse obligado a ello.

# Objetivos

La creación de esta aplicación surge por la necesidad generada por gente cercana a mi entorno que necesita llevar a cabo un control de sus gastos e ingresos de una manera sencilla y clara. Es por esto que el objetivo principal que sustenta la creación del proyecto es ofrecer una herramienta fácil de utilizar que permita al usuario ver los gastos e ingresos que lleva realizados y cuales están aún pendientes por realizar.

Dichos objetivos se podrían englobar en los siguientes puntos:

- Crear una aplicación para poder apuntar los gastos e ingresos mensuales
- Facilitar el uso de la herramienta con plantillas para que el usuario no se vea obligado a rellenar mensualmente todos sus gastos fijos
- Poder visualizar de manera rápida y sencilla que tal se ha gestionado el usuario en los meses del año actual
- Marcar como objetivo de diseño una aplicación vistosa, pero no saturada, y que sea sencilla en su manejo, prácticamente debe ser intuitivo todo su funcionamiento.

Por último, cabe destacar que la realización del máster ha avivado mis deseos de realizar una buena programación en mis proyectos. Es por esto que otro objetivo importante, pero no dependiente del proyecto en sí, es la creación de una herramienta robusta, ordenada y mantenible. Así que un objetivo, de carácter más profesional, es la creación de una aplicación con una correcta arquitectura para el desarrollo Android y con una modularidad suficiente para que si se precisa de cambios no acarree un trabajo costoso.

# Motivación

Como se ha descrito anteriormente la motivación principal es de carácter personal, ya que la realización del proyecto es por petición de varios familiares cansados de las aplicaciones complejas que encuentran en la actualidad. También me motiva el hecho de poder ofrecer algo útil y funcional para el usuario común de los dispositivos móviles, el cual suele ver como aplicaciones con ideas muy interesantes se acaban convirtiendo en aplicaciones desechables por su complejidad y carga visual.

También existe una motivación más profesional debido a que me gustaría poder plasmar todos mis conocimientos del master realizado en la universidad, de esta forma se puede observar que ha sido útil para mi vida profesional y que he conseguido aprender correctamente todo el contenido que se ha impartido.

# Tecnologías utilizadas

En el siguiente apartado se van a comentar las tecnologías que se han utilizado para la realización del proyecto:

- Android: la aplicación está desarrollada en Android Studio y pensada para ser instalada en dispositivos Android
- Marvelapp (https://marvelapp.com): herramienta online que permite crear un prototipo en imágenes de la herramienta en cuestión. Es muy útil y sencillo para crear las vistas de la aplicación, así como su flujo. Se ha utilizado para plasmar de manera física los diseños de la aplicación y así tener una guía clara sobre la que seguir a la hora de programar. Gracias a esta herramienta ha sido sencillo crear las vistas y aclarar ideas para que la aplicación cumpla los objetivos marcados.
- Material Design: a lo largo de toda la creación de las vistas se han seguido los principios que marca Material Design ya que es el estándar más utilizado. Las pautas que propone son diseños planos, colores uniformes, figuras sencillas y diferentes capas para separar los distintos objetos sencillos. Siguiendo estos principios se consigue una uniformidad que luego se puede transmitir al resto de aplicaciones en los diferentes dispositivos.
- Renderers (https://github.com/pedrovgs/Renderers): librería utilizada en todo el proyecto para la creación e implementación de los RecyclerViews. Es una librería creada por Pedro Vicente Gómez Sánchez, actualmente trabajador de la empresa Karumi, que facilita toda la creación de los RecyclerViews, simplificando toda la parte de los ViewHolder y Adapter focalizando el trabajo del programador en tener una lista de elementos que pasar al Adapter y unos Renderers que se encargarán de pintar las vistas.
- Butterknife: librería que nos permite la inyección de vistas, simplificando enormemente el uso de los elementos de las vista y sus eventos en los Activities y Fragments. Librería creada por Jake Wharton, ingeniero Android, que ha cambiado por completo el trato con las vista desde la programación. Librería muy útil para obtener un código más legible y limpio.
- Dagger 2: librería para gestionar e implementar el patrón de inyección de dependencias en la aplicación. Libería creada por Google, a partir de dagger 1 creado por la gente de Square, mejora en gran medida a su padre ya que toda la inyección la realiza en tiempo de compilación consiguiendo que posibles fallos se detecten antes de ejecutar la aplicación.

# Arquitectura de la aplicación

A continuación se procederá a lo largo del siguiente apartado describir la arquitectura que se ha llevado a cabo en el proyecto y el porqué de las elecciones tomadas.

## Esquema del diseño

Dado que uno de los objetivos del desarrollo del proyecto que se había marcado como imprescindible era la correcta creación de software de calidad y coherente desde el comienzo del proyecto se tuvo claro que el patrón de arquitectura del software que se seguiría es el MVP (Modelo – Vista – Presentador), ya que nos permite realizar una separación muy clara de la parte visual y la parte de negocio.

MVP es un patrón de arquitectura que deriva del MVC (Modelo – Vista – Controlador), el cual ayuda a separar los datos y la lógica del negocio de la parte visual con lo que el usuario interacciona, dando al presentador la funcionalidad de ser el encargado de gestionar toda la lógica de presentación.

MVP se basa en estas tres capas:

- Modelo: es la capa que gestiona los datos, la información que se tiene que mostrar en las vistas.
- Vista: se encarga de mostrar los datos, es lo que ve el usuario en dispositivo
- Presentador: es la capa que conecta las otras dos, se encarga de pedir y actualizar los datos del Modelo y de enviar la información para que la Vista lo pinte en el dispositivo.

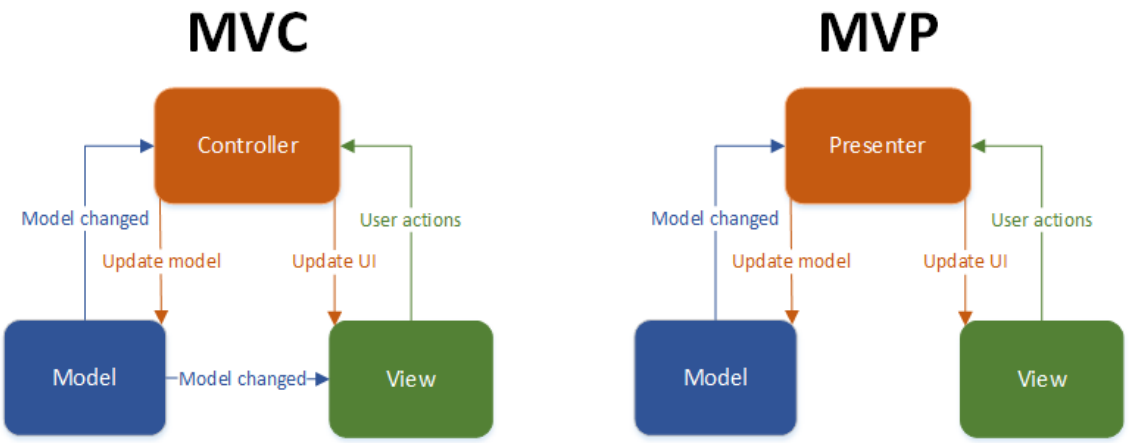


*Ilustración 1- Esquema de MVC y MVP*

Como no se deseaba ser dependiente del framework Android Studio, ya que en un futuro se puede plantear el hecho de llevar la aplicación a otros dispositivos lo que acarrearía cambiar de framework, el proyecto se ha complementado con Clean Architecture. Con Clean lo que conseguimos es dividir de otra forma las capas del proyecto consiguiendo así los siguientes objetivos de arquitectura:

- Independiente del Framework: de esta forma conseguimos que si se deseado llegar a otros dispositivos el trabajo realizado en la parte de la lógica de negocio se pueda utilizar fácilmente.
- Independiente de la Interfaz de Usuario: la interfaz puede cambiar todo lo que se quiera que no va a interferir en la parte de la lógica de negocio
- Testeable: se pueden realizar test separados de la parte de vistas como de la parte de negocio, permitiendo así ser más robustos y que se identifiquen rápidamente los errores que surjan.
- Independiente de la Base de Datos: nos permite utilizar cualquier tipo de almacenamiento de los datos sin que ninguna otra capa se vea afectada. Este punto se ve fácilmente en el proyecto, donde se comenzó con una Base de datos “mockeada” hasta que se implementó la base de datos final en Realm.

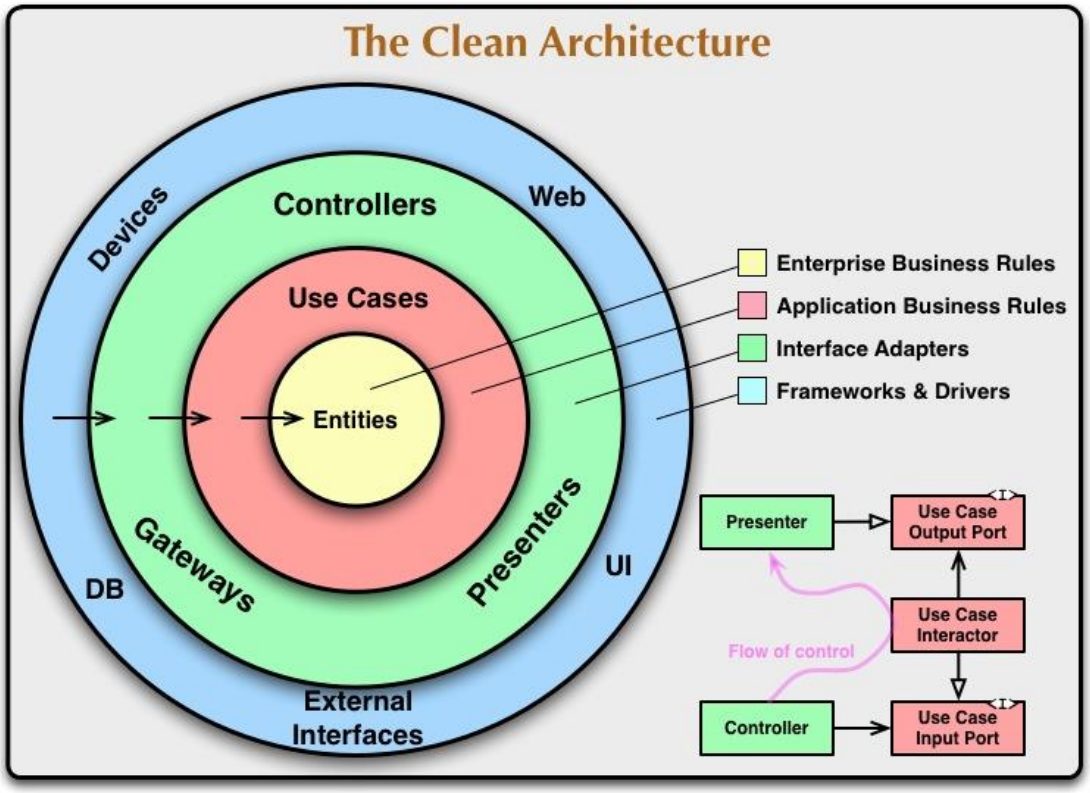


*Ilustración 2. Arquitectura Clean*

Dado que se va a utilizar Clean las capas finales del proyecto se dividirían en tres:

### Capa de Presentación

Se basará en el patrón MVP y será la encargada de tener la vista de la aplicación. En ella se encuentran todos los Activities, Fragments y Presenters de la aplicación

### Capa de Dominio

Contiene todos los objetos Java y los casos de uso que conecta la vista con la capa de datos. Contiene todo lo concerniente a la lógica de negocio de la aplicación.

### Capa de Datos

Esta capa se sustentará sobre el patrón Repository para hacer que la implementación sea independiente del sistema que se utilice para almacenar los datos. Es por esto que existe la interfaz ContadroidRepository, la cual deben implementar todos los sistemas que se utilicen para el funcionamiento con base de datos. Su labor consiste en recoger y almacenar los datos en el medio externo seleccionado, en este caso Realm.

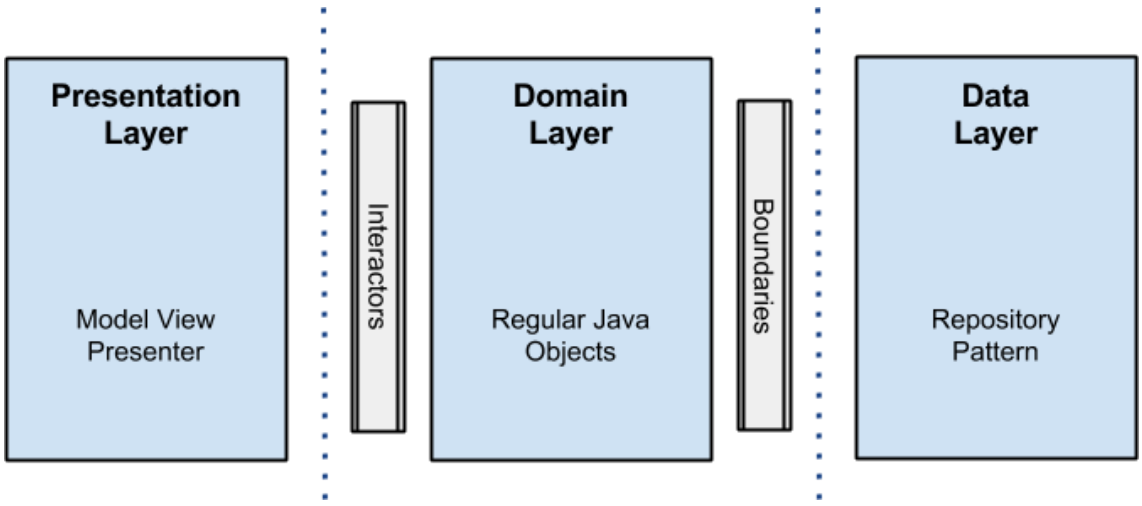


*Ilustración 3- Dibujo de la capas de la aplicación*

Para el proyecto se ha decidido que los paquetes de la aplicación se dividan en las 3 principales funcionalidades:

- Llevar los gastos del mes actual (ActualMonth).
- Llevar unas estadísticas de los gastos en los meses del año (Summary).
- Tener una plantilla que poder añadir al mes actual para no repetir la inserción de gastos fijos (Template).

Antes de mostrar una "vista de pajaro" de la organización del proyecto queda decir que existe dos paquetes generales más:

- Common: contiene las clases generales a todas las funcionalidades del proyecto.
- DI: contiene lo concerniente a la inyección de dependencias generales al proyecto.

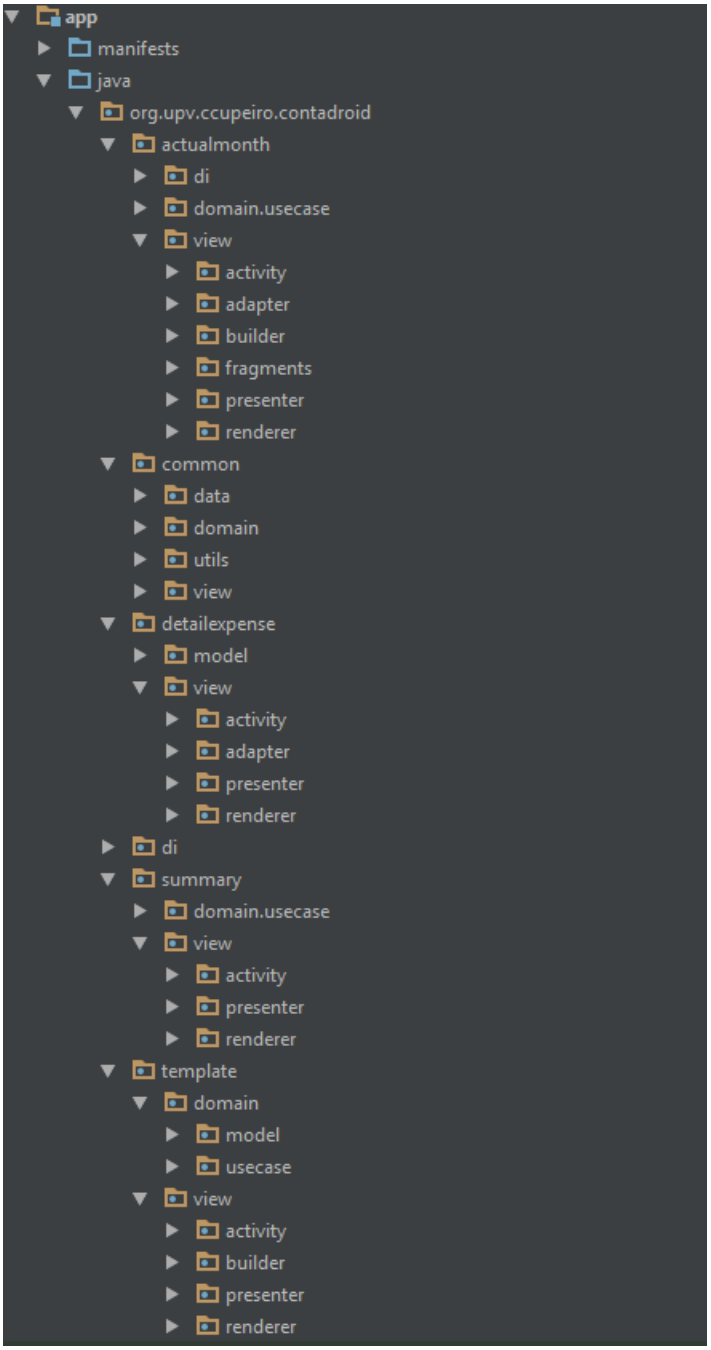


*Ilustración 4- Organización de la Arquitectura del proyecto*

# Modelo de datos

Todo el modelo de datos se sustenta sobre una única tabla definida en la clase Expense. Esto es debido a la sencillez de la aplicación, por lo que no es necesario relaciones ni complejidades en los datos.

Para la implementación del almacenamiento de los datos se ha tomado la decisión de utilizar el framework Realm. Se ha tomado esta decisión por varias razones que se detallan a continuación:

- Sencillez de uso: ha sido muy fácil implementar todo lo relacionado con el modelo de datos ya que con anotaciones Java se crean las tablas que luego se almacenarán.
- Velocidad: se ha testado la velocidad de Realm frente a diferentes ORM's que tiene mucho peso y se puede observar en las siguientes gráficas que a la hora de recuperar datos gana por completo a sus oponentes. En la única parte en la flaquea, respecto de algún competidor, es a la hora de insertar grandes cantidades de registros, tema totalmente innecesario en la funcionalidad del proyecto ya que la inserción de gastos nunca será tan grande como para que se vea afectado la velocidad.

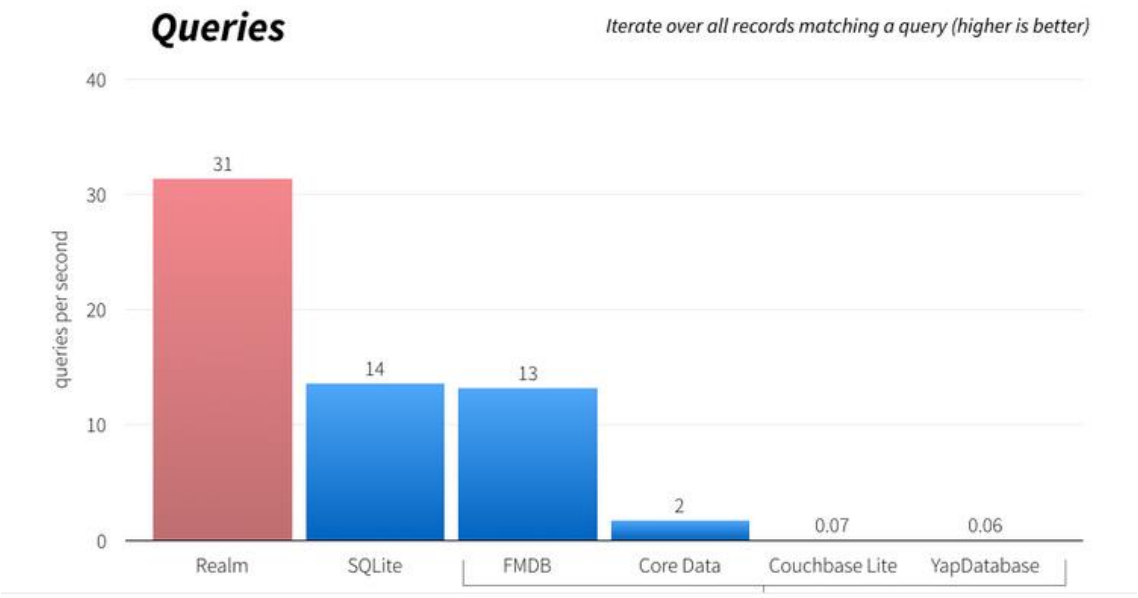


*Ilustración 5 - Velocidad en las querys de Realm frente a competidores*

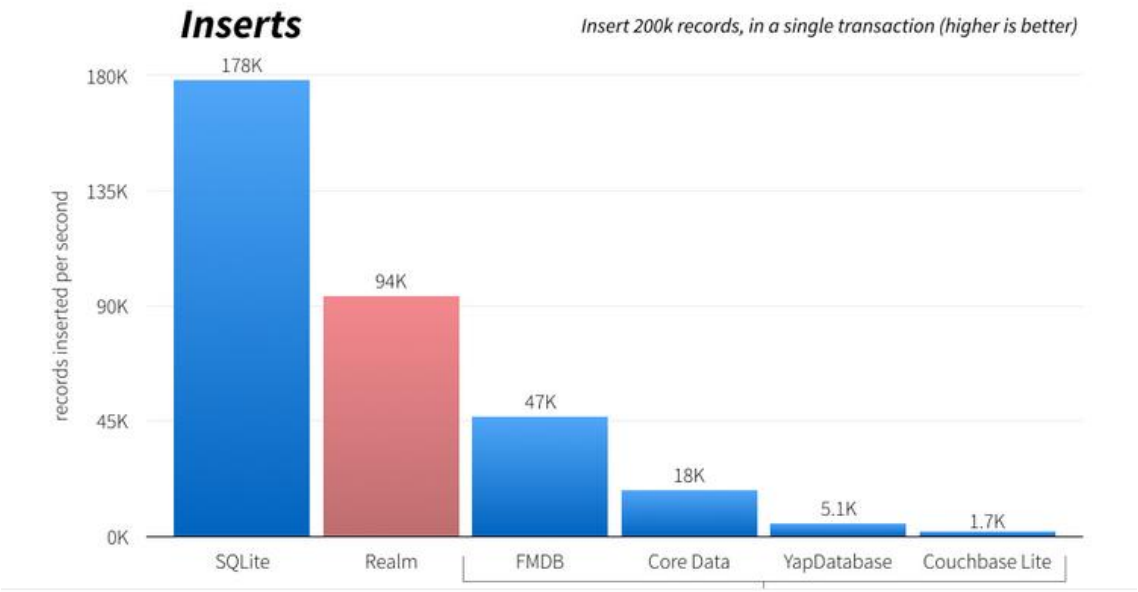


*Ilustración 6 - Gráfica de inserción de registros en Realm frente a competidores*

- Framework actual: se sabe que es uno de los sistemas más actuales y eficientes del mercado, siendo utilizado por empresas como IBM, Amazon o Google.
- Aprendizaje personal: dado que este proyecto es para demostrar mis conocimientos sobre los dispositivos móviles no hay nada mejor que enfrentarse a los sistemas que se utilizan en el entorno laboral y que se desconocen. De esta forma el desarrollo del proyecto resulta ser un reto en el cual se aprende mucho para futuro laboral y personal.

Para separar por completo la implementación de la base de datos de los modelos de negocio se creó una clase POJO exclusivamente para Realm. De esta forma se mapeaba todo de manera simple para que si Realm necesitaba algo especial, como fue la imposibilidad de guardar los grupos como Enum en la base de datos, no afectase a la clase general Expenses.

Pese a que se decidió desde un principio implementar todo lo relacionado con la base de datos en Realm, no se pudo implementar hasta el final dado que hacía falta un pequeño período de aprendizaje. Debido a que nos fijamos como objetivo de arquitectura utilizar Clean, esto no supuso un problema debido a que se partió de una implementación más sencilla con lista fijas que se iniciaban en cada ejecución. Como ambas implementaciones implementaban los métodos definidos en el Repositorio este cambio final entre sistemas fue tan simple como cambiar una sola línea.

# Vistas

En este apartado se van a describir primero las vistas por separado y los diferentes mensajes de la aplicación, para acabar concluyendo con el esquema de navegación entre las vistas.

## Vista de Mes Actual

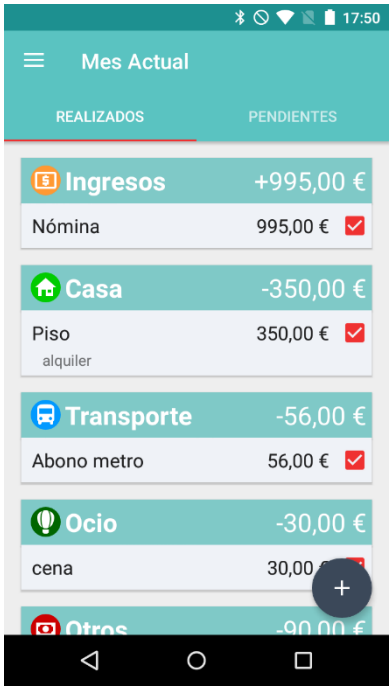


*Ilustración 7- Pestaña de Gastos Realizados*

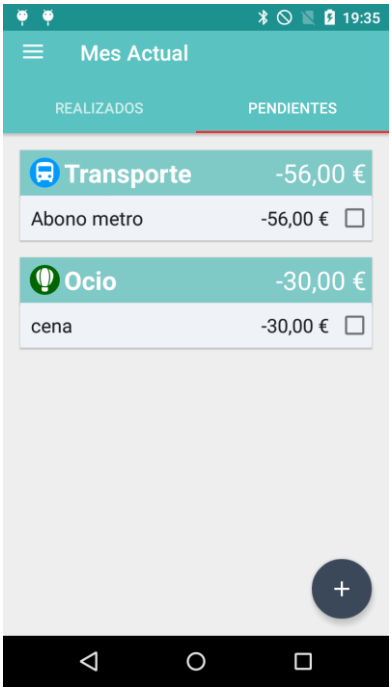


*Ilustración 8 - Pestaña de Gastos Pendientes*

Esta vista contiene los gastos del mes en el que se encuentra el usuario. Está formada por dos pestañas:

- Pestaña Realizados: contiene los gastos que el usuario ya ha pagado y que se contabilizan para el resumen mensual.
- Pestaña Pendientes: contiene los gastos que el usuario aún no ha pagado y que no entran en el resumen mensual.

Como se puede observar en la imagen los gastos se ordenan en tarjetas en función de unas categorías:

- ❖ Ingresos
- ❖ Casa
- ❖ Transporte
- ❖ Comida
- ❖ Compras
- ❖ Ocio
- ❖ Otros

La relación entre las dos pestañas se realiza mediante la activación o desactivación del checkbox que tiene cada gasto. Cada vez que se accione el checkbox hará que el gasto pase de una pestaña a la otra. Para el trato de los gastos se disponen de las siguientes acciones:

- Pulsar sobre el FloatingButton : nos da acceso a la creación de un gasto, la vista que abre se comentará más adelante en el detalle de un gasto.
- Pulsación Larga sobre un gasto: este evento abre un diálogo que permite dos acciones
  - Editar: abre la misma ventana que la creación pero con los datos del gasto ya creado
  - Eliminar: elimina el gasto de la aplicación

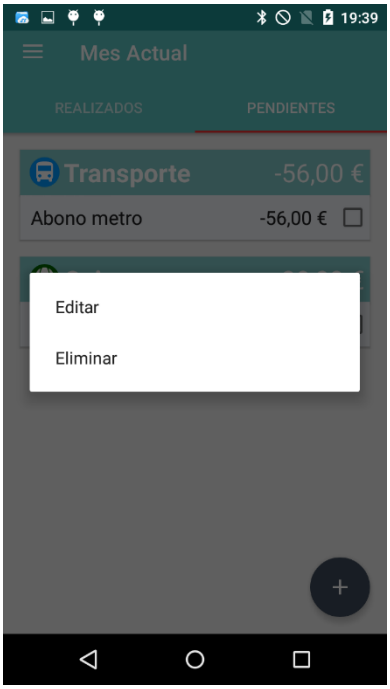


*Ilustración 9 - Dialogo de pulsación larga*

### Vista de Resumen de gastos anual

*Ilustración 10 - Ventana de Resumen anual*

En esta ventana se puede visualizar muy rápidamente como queda el balance mensual en función de los gastos e ingresos que se hayan tenido en el mes. La única categoría que suma es la de Ingresos, por lo que todos los gastos del resto de categorías se identifican como gastos negativos que restan. Existen cuatro estados para cada mes:

1. No hay datos: el mes no tiene datos para evaluar el balance.
2. Verde: el balance del mes es positivo
3. Rojo: el balance del mes es negativo
4. Negro: el balance del mes es neutral, es decir, hay lo mismo de ingresos que de gastos.

Al ser una vista solo informativa no existen eventos para interaccionar con la ventana.

## Vista de Plantilla de gastos

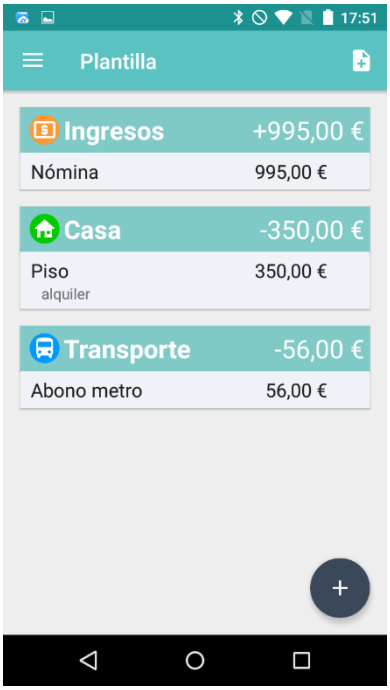


*Ilustración 11 - Ventana de Plantilla de gastos*

A la hora de rellenar los gastos del mes es muy frecuente que los gastos se repitan, ya que es normal tener una serie de gastos e ingresos fijos. Debido a esta casuística existe esta vista, donde el usuario puede realizar una plantilla que más tarde copiar a los gastos de mes si así se desea. Las acciones que se pueden realizar son idénticas a las ya descritas en la vista del Mes Actual, creación, edición y borrado de gastos. Sólo que en esta ventana desaparece el uso del checkbox para indicar que un gasto esta realizado o no y se añade una acción nueva:

- Importar la plantilla al mes actual : con este botón se abre un dialogo que nos permite incluir la plantilla en el mes actual.

*Ilustración 12 - Diálogo para importar la plantilla*

## Vista de Detalle de un gasto

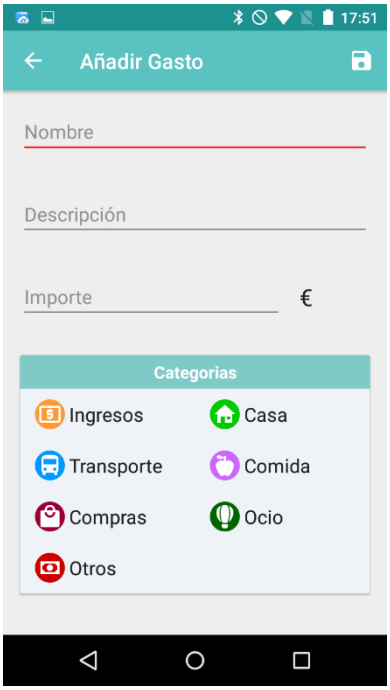


*Ilustración 13 - Ventana de detalle de un gasto*

Esta vista es a la que se accede cuando se quiere crear o editar un gasto, en ella se ofrecen los diferentes campos que definen a un gasto:

- Nombre
- Descripción
- Importe
- Categoría (Solo se permite la selección de una de ellas)

Como se puede observar en la parte superior se ofrecen dos acciones sobre la toolbar:

- Guardar el gasto : ya sea para editar como para crear un gasto nuevo es necesario pulsar sobre este botón que permite el guardado.
- Volver a la ventana anterior : sirve para cancelar la operación de guardado o edición y volver a la ventana anterior.

### Vista del Menú de navegación

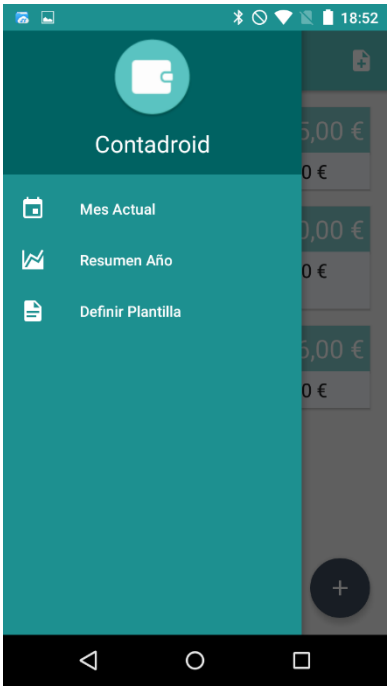


*Ilustración 14 - Menú de navegación*

Cada vez que se pulsa sobre el icono de la toolbar se abre el NavigationDrawer que se muestra en la imagen. A través de dicho menú se puede navegar entre las diferentes funcionalidades del proyecto. De esta forma esta vista se convierte en el nexo de unión de toda la aplicación.

Para concluir con las vistas queda decir que, debido a que no se apilan las actividades principales que tienen acceso al menú de navegación, se ha creado un evento al pulsar sobre el botón de retorno en el móvil por el cual se muestra un dialogo donde seleccionar si realmente se desea salir de la aplicación.

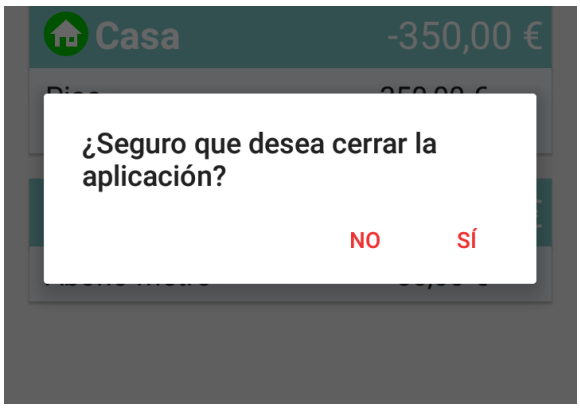


*Ilustración 15 - Dialogo de cierre de la aplicación*

## Esquema de Navegación

*Ilustración 16 - Navegación entre vistas*

# Conclusiones

A lo largo de esta Tesina se puede observar que los objetivos establecidos desde un comienzo se han cumplido por completo. La posibilidad de realizar una aplicación desde cero con una idea muy clara hay sido un trabajo muy educativo. Desde un principio se pensó esta aplicación para un uso diario y que resultase sencillo, lo cual se ha conseguido gracias unos objetivos muy claros y concisos.

Pese a que se han cumplido todos los objetivos marcados, es necesario decir que la aplicación se puede mejorar en algunos ámbitos que se describen a continuación:

- Que sea posible la visualización de meses pasados desde la ventana de resumen. Esta funcionalidad ha sido muy reclamada entre las personas que han visto el proceso de creación de la aplicación. No se ha llevado a cabo para no salir de los objetivos marcados y que la aplicación no se estanque en este proyecto final sino que continúe evolucionando.
- Sería muy interesante poder crear las categorías de los gastos, de esta forma se ofrece una personalización mayor sobre la aplicación. Es lógico que la categorización actual que hay de los gastos quede limitada para algunos usuarios, así que sería muy atractivo para el uso de la aplicación poder disponer de esta funcionalidad.
- Poder loguearse con una cuenta de google para continuar con la personalización de aplicación en función de cada usuario que se ha descrito antes. Quizás no es una funcionalidad necesaria pero dicha personalización puede decantar que guste más o menos una aplicación
- Sería imprescindible en un futuro cercano de la aplicación que todo lo que se almacena en la base de datos tenga persistencia más allá del propio dispositivo. En la actualidad es muy frecuente que un usuario cambie de móvil y es bien conocido lo molesto que puede resultar ese cambio si las aplicaciones no se sincronizan con una cuenta online. De esta forma se podría evitar que en un cambio el usuario decida que la aplicación es prescindible.

Por último, queda comentar que el proyecto final no solo ha sido un mecanismo de aprendizaje y una manera de mostrar mis conocimientos sobre el máster, sino que también ha sido un reto personal poder visualizar mi evolución desde los comienzos del master hasta la actualidad. Siento que he aprendido no solo a desarrollar para aplicaciones móviles, sino también a desarrollar código profesional y de calidad. Esta última tarea del master ha sido dura pero muy gratificante.

# Bibliografía

## Libros


- Martin, R. C. (2008). *Clean Code: A Handbook of Agile Software Craftsmanship* (Edición: 1). Upper Saddle River, NJ: Prentice Hall.


## Páginas Web


- 7.2. Principios S.O.L.I.D (Diseño ágil con TDD). (s. f.). Recuperado 20 de septiembre de 2017, a partir de http://librosweb.es/libro/tdd/capitulo_7/principios_solid.html
- Aplicando Clean Architecture en Android. (s. f.). Recuperado 20 de septiembre de 2017, a partir de https://erikcaffrey.github.io/ANDROID-clean-architecture/
- Aprende a usar Clean Architecture por Kilian Cerdán - Killing the Code. (2017, marzo 5). Recuperado 20 de septiembre de 2017, a partir de http://www.kiliancerdan.com/clean-architecture/
- Dagger ‡ A fast dependency injector for Android and Java. (s. f.). Recuperado 20 de septiembre de 2017, a partir de https://google.github.io/dagger/
- Dependency Injection with Dagger 2 | CodePath Android Cliffnotes. (s. f.). Recuperado 20 de septiembre de 2017, a partir de https://guides.codepath.com/android/Dependency-Injection-with-Dagger-2
- Inyección de dependencias en Android con Dagger 2 | adictosaltrabajo. (s. f.). Recuperado 20 de septiembre de 2017, a partir de https://www.adictosaltrabajo.com/tutoriales/inyeccion-de-dependencias-en-android-con-dagger-2/

- *KataSuperHeroesAndroid: Super Heroes Kata for Android Developers. The main goal is to practice UI Testing*. (2017). Java, Karumi. Recuperado a partir de https://github.com/Karumi/KataSuperHeroesAndroid (Original work published 8 de enero de 2016)
- Material Design Icons. (s. f.). Recuperado 20 de septiembre de 2017, a partir de https://materialdesignicons.com/
- ¿Por qué utilizo clean Architecture? (2016, julio 28). Recuperado 20 de septiembre de 2017, a partir de http://xurxodev.com/por-que-utilizo-clean-architecture-en-mis-proyectos/
- ¿Qué es Material Design? (2014, noviembre 9). Recuperado 20 de septiembre de 2017, a partir de https://elandroidelibre.elespanol.com/2014/11/que-es-material-design.html
- Realm: Create reactive mobile apps in a fraction of the time. (s. f.). Recuperado 20 de septiembre de 2017, a partir de https://realm.io/
- Realm Tutorial: Getting Started. (s. f.). Recuperado 20 de septiembre de 2017, a partir de https://www.raywenderlich.com/112544/realm-tutorial-getting-started
- Rubira, J. (2011, julio 14). Solid, cinco principios básicos de diseño de clases. Recuperado 20 de septiembre de 2017, a partir de https://www.genbetadev.com/paradigmas-de-programacion/solid-cinco-principios-basicos-de-diseno-de-clases
- Sánchez, P. V. G. (2017). *Renderers is an Android library created to avoid all the boilerplate needed to use a RecyclerView/ListView with adapters*. Java. Recuperado a partir de https://github.com/pedrovgs/Renderers (Original work published 28 de febrero de 2014)

- Sánchez, P. V. G. (2017). *Sample project created to show some of the best Android practices to work in the Android UI Layer. The UI layer of this project has been implemented using MVP or MVVM (without binding engine) to sh.*. Java. Recuperado a partir de https://github.com/pedrovgs/EffectiveAndroidUI (Original work published 4 de julio de 2014)
- Varadi, G. (2016, agosto 11). How to use Realm for Android like a champ, and how to tell if you're doing it wrong. Recuperado 20 de septiembre de 2017, a partir de https://medium.com/@Zhuinden/how-to-use-realm-for-android-like-a-champ-and-how-to-tell-if-youre-doing-it-wrong-ac4f66b7f149
- Vísteme con Clean Architecture que tengo prisas – El Blog de Chema. (s. f.). Recuperado 20 de septiembre de 2017, a partir de http://jmperezramos.net/desarrollo-en-android/visteme-con-clean-architecture-que-tengo-prisas/

## Video Youtube

- Android Talks. (s. f.). *MVP + Dagger 2*. Recuperado a partir de https://www.youtube.com/watch?v=_yVE1DRY1v8&t=3093s
- José Dimas Luján Castillo. (s. f.). *Capítulo 309 - Android Bases de datos Realm buscar datos*. Recuperado a partir de https://www.youtube.com/watch?v=TumeZ_HnBHo&t=137s&index=1&list=PLWMrr4TcxwaUE2cIw2gZT302BzANmRUOR
- jotajotavm. (s. f.). *Como usar Butter Knife en Otros Elementos - Inyeccion de Dependencias 05*. Recuperado a partir de https://www.youtube.com/watch?v=Tglof37PFQE&index=5&list=PLraIUviMMM3coR3VfsqtMopnDZL3YaLK9

- slidenerd. (s. f.). *144 Android Navigation Drawer Tutorial Part 1 |*. Recuperado a partir de https://www.youtube.com/watch?v=K8hSIP2ha-g&list=WL&index=52
- SuperAndroidTutorial. (s. f.). *Android Database Tutorial (1/3) - RealmDB (alternative to SQLite)*. Recuperado a partir de https://www.youtube.com/watch?v=9pFJz5VexRw&t=545s&list=WL&index=62
- TwistedEquations. (s. f.). *Dagger 2 Android Tutorial 01 - Setting up*. Recuperado a partir de https://www.youtube.com/watch?v=Qwk7ESmaCq0&t=1s
- VEOTutoriales. (s. f.). *Usar la base de datos Realm en un proyecto Android*. Recuperado a partir de https://www.youtube.com/watch?v=lDv214h4u0g&t=1s&list=WL&index=63

# Anexos

## Código fuente en GitHub

El código fuente de este proyecto se encuentra subido a Github en la siguiente url:

https://github.com/c-cupeiro/contadroid