

# **APLICACIÓN PARA LA GESTIÓN ALIMENTARIA DEL HOGAR (AGAH)**

**Andrea Méndez Sanz**

**Leidy Alejandra Cortés González**

**Mario Marugán Cancio**

**CFGS Desarrollo de Aplicaciones Multiplataforma (DAM)**

**Curso 2020-2021**

**Innovación en Formación Profesional (IFP)**

**Convocatoria de Presentación: junio 2021**



# Índice de contenidos

<b>Índice de contenidos .....</b>	<b>2</b>
<b>Índice de figuras.....</b>	<b>4</b>
<b>Índice de tablas.....</b>	<b>7</b>
<b>Resumen .....</b>	<b>9</b>
<b>Capítulo 1. Introducción .....</b>	<b>11</b>
<b>Capítulo 2. Objetivos .....</b>	<b>13</b>
Sección 2.1 Objetivo principal.....	13
Sección 2.2 Objetivos parciales.....	13
Sección 2.3 Medios utilizados .....	13
<b>Capítulo 3. Estado del Arte .....</b>	<b>15</b>
Sección 3.1 Evolución del desarrollo móvil.....	15
Sección 3.2 Entornos de desarrollo para aplicaciones móviles .....	18
Sección 3.3 Lenguajes de programación para aplicaciones móviles .....	20
Sección 3.4 Otros aspectos tecnológicos .....	21
Sección 3.5 Aplicaciones Similares.....	25
<b>Capítulo 4. Método del Trabajo .....</b>	<b>27</b>
Sección 4.1 Metodología Kanban .....	27
Sección 4.2 Planificación.....	28
<b>Capítulo 5. Resultados.....</b>	<b>31</b>
Sección 5.1 Modelo Entidad/Relación .....	31
Sección 5.2 Modelo de casos de uso .....	31
Sección 5.3 UML.....	32
Sección 5.4 Vistas de la app .....	42
<b>Capítulo 6. Conclusiones.....</b>	<b>63</b>
Sección 6.1 Conclusiones del proyecto.....	63

Sección 6.2 Líneas futuras.....	63
<b>Capítulo 7. Bibliografía .....</b>	<b>67</b>
<b>Anexo 1. Acrónimos .....</b>	<b>71</b>
<b>Anexo 2. Licencia .....</b>	<b>72</b>

## Índice de figuras

Figura 1 Teléfono móvil IBM Simon .....	15
Figura 2 Comparativa del uso web contra el uso de apps en móvil .....	17
Figura 3 Representación de los sensores.....	18
Figura 4 Entorno de desarrollo de Android Studio .....	19
Figura 5 Entorno de desarrollo de Xcode.....	20
Figura 6 Representación de las diversas bases de datos .....	23
Figura 7 Infografía del MVC .....	23
Figura 8 Esquema del MVC .....	24
Figura 9 Infografía del MVP .....	24
Figura 10 Vista de artículos (Out of milk) .....	25
Figura 11 Vista de añadir un artículo (Out of milk).....	25
Figura 12 Vista de recetas (Bring!) .....	26
Figura 13 Vista inicial de la aplicación (Bring!) .....	26
Figura 14 Vista de distintas listas (Listonic) .....	26
Figura 15 Vista de productos (Listonic).....	26
Figura 16 Metodología Kanban.....	27
Figura 17 Organización Kanban .....	29
Figura 18 Estructura en Trello.....	30
Figura 19 Representación del modelo entidad/relación .....	31
Figura 20 Modelo de caso de uso .....	31
Figura 21 Vistas UML .....	32
Figura 22 Login UML .....	33
Figura 23 Util, notificaciones y login UML .....	34
Figura 24 Todos los activities .....	35
Figura 25 Todos los fragments.....	36

Figura 26 Modelos UML.....	37
Figura 27 Adapters UML .....	38
Figura 28 Adapters con relación hacia activitis UML.....	39
Figura 29 Adapters con relación hacia listas UML .....	40
Figura 30 Drawer UML.....	41
Figura 31 Splash .....	42
Figura 32 Vista Inicial .....	43
Figura 33 Registro de usuarios.....	43
Figura 34 Inicio de sesión.....	43
Figura 35 Problema de inicio de sesión .....	44
Figura 36 Canales de la aplicación .....	45
Figura 37 Ajustes de notificaciones chrome .....	45
Figura 38 Drawer.....	47
Figura 39 Infografía de tipos de productos.....	49
Figura 40 Lista de nevera .....	49
Figura 41 Menú funciones .....	50
Figura 42 Buscador.....	51
Figura 43 Item sin caducidad .....	51
Figura 44 Pop-up Añadir cantidad .....	52
Figura 45 Pop-up Eliminar.....	52
Figura 46 Editar productos.....	54
Figura 47 Lista sin productos .....	54
Figura 48 Listas de la compra.....	56
Figura 49 Menú desplegable.....	56
Figura 50 Añadir un nuevo producto .....	57
Figura 51 Creación de evento en el calendario.....	59

<i>Figura 52 Perfil de usuario .....</i>	60
Figura 53 Ejemplo de ejecución en tablet.....	61
Figura 54 Visualización de producto cercano a caducidad .....	64
Figura 55 Representación de fuerza de contraseña .....	65
Figura 56 Licencia CC.....	72

## Índice de tablas

Tabla 1 Comparativa de las aplicaciones .....	17
Tabla 2 Desarrollo de objetivos .....	63



## Resumen

Este documento contiene la memoria del Trabajo Fin de Grado(TFG), que documentará el proceso seguido para el desarrollo de la aplicación, este fue realizado por las alumnas Andrea Méndez Sanz y Leidy Alejandra Cortés González del grado de Desarrollo de Aplicaciones Multiplataforma.

En este caso, el problema que se quiere solventar con nuestra propuesta es uno que se encuentra en la gran mayoría de los hogares españoles: el desperdicio de alimentos [1]. Desgraciadamente en España se tira comida con frecuencia, debido al desconocimiento a veces de lo que se compra o de la caducidad de estos. Para intentar menguar el desperdicio se ha creado la aplicación expuesta en este TFG.

Se ha desarrollado una aplicación Android, tanto para dispositivos móviles como para tablets, con el objetivo de crear una herramienta para la gestión de la nevera o congelador en los hogares. Esta idea nace como alternativa a las diferentes aplicaciones existentes en el mercado móvil, cuya especialidad suele ser la creación de listas de la compra; la aplicación a desarrollar aparte de tener la funcionalidad mencionada, irá un paso más allá, dando el conocimiento a los usuarios de lo que tienen también en sus hogares.

Los usuarios tendrán una forma de identificarse, vía correo. Una vez dentro, el usuario será el encargado de añadir los productos existentes en su nevera o congelador. Una vez el usuario haya notificado a la aplicación que un producto se ha acabado, podrá decidir si añadirlo a la lista de la compra o no.



## Capítulo 1. Introducción

En la actualidad casi el 81.5% de los hogares españoles reconocen tirar alimentos tal y como los compraron [1], este desperdicio alimentario supone un gasto innecesario y poco consecuente con su medio. Además, el porcentaje de personas sin capacidad económica para conseguir una manutención consistente y saludable crece cada día más. Se eligió este proyecto debido a los motivos comentados anteriormente, la ayuda que se plantea para menguar estos problemas es; una lista de la compra optimizada, no sólo para la compra diaria sino para tener un control total de todo el proceso.

El método que se ha diseñado para reducir el desperdicio de alimentos se basa en tomar el control de manera activa sobre lo que ya se tiene en casa y lo que verdaderamente se necesita comprar. Así, a la par que ayudar a la economía familiar para no gastar de más, reduce el impacto generado sobre el medio ambiente, cuantos menos kilos de comida desperdicie una familia, menos producción acabará en la basura.

El objetivo es; a través de la aplicación, tener información a tiempo real de la cantidad, estado y disponibilidad de los productos que se poseen, vinculados a una cuenta personal a la que se pueda acceder a través de un inicio de sesión en cualquier dispositivo compatible.

En resumidas cuentas, se trata de dar una solución a los hogares para identificar que alimentos tienen en su casa desde cualquier parte, y poder así ser más consecuentes a la hora de realizar la compra.

Se ha explicado y representado el tema a tratar en este TFG, antes de empezar se evaluará, a través de un estudio de mercado, las distintas aplicaciones y lenguajes disponibles para la realización de esta aplicación móvil.

Los dos grandes sistemas operativos móviles en la actualidad son Android e iPhone Operating System (iOS), a nivel mundial en 2020, Android posee un 72% del mercado de móviles, mientras que iOS tiene un 27% [2]; en cambio, en el mercado de tablets del mismo año, iOS tiene un 56% compitiendo contra el 43% de Android [3]. Por estos datos se escogerá la opción de desarrollar en Android, para poder tener un posible público más amplio.



## Capítulo 2. Objetivos

En este capítulo se describirán los objetivos a lograr por la realización de este TFG.

### Sección 2.1 Objetivo principal

El objetivo principal de este TFG es el desarrollo de una aplicación que cubra las necesidades de los hogares para ayudar con el control de alimentos que se encuentren en su nevera o congelador.

La aplicación permitirá añadir productos en dos secciones distintas, nevera y congelador, también se tendrá otro apartado en la aplicación para visualizar la lista de la compra. Otro punto importante a tener en cuenta es la interfaz, que debe ser intuitiva y fácil de usar para cualquier usuario, esto repercute positivamente para reducir los tiempos de aprendizaje y minimizar la frustración que pueda llegar a tener el usuario.

De este objetivo principal se derivan otros objetivos parciales que se detallan en la siguiente sección.

### Sección 2.2 Objetivos parciales

- OP1.** Análisis de la idea y de los requisitos que se necesiten para el cumplimiento total de la aplicación.
- OP2.** Comparativa con las demás aplicaciones del mercado e identificar posibles fallas en ellos para la realización de mejoras.
- OP3.** Creación de un diseño visual, atractivo e intuitivo para los clientes.
- OP4.** Realización de la base de datos, usada para almacenar los datos necesarios en la aplicación.
- OP5.** Aprendizaje de Java para su utilización en Android Studio.
- OP6.** Desarrollo de la aplicación cumpliendo con los objetivos anteriores.
- OP7.** Lanzamiento de la aplicación a los clientes cumpliendo con todos los requisitos.

### Sección 2.3 Medios utilizados

Para la realización del TFG se usará todos los medios que se tenga al alcance y sean necesarios para cumplir los objetivos del proyecto.

A continuación, se detallarán los medios hardware que se utilizaron en el desarrollo del proyecto y los medios software escogidos.

#### Medios Hardware

- Ordenador de sobremesa con procesador Intel(R) Core (TM) i7-7700k 4.20GHz memoria RAM de 16GB y SO de 64bits
- Ordenador portátil ASUS con procesador Intel(R) Core (TM) i7-3630QM 2.4GHz memoria RAM de 12GB y SO de 64 bits

## Medios Software

- **Android Studio:** con esta herramienta se desarrollará la aplicación, con las siguientes versiones en mente: [Target SDK(Software Development Kit) 30(API(Application Programming Interface) 30: Android 11.0(R)) | Mini SDK 26(API 26: Android 8.0(Oreo))]
- **Trello:** con esta tecnología se implementará un modo de distribución de tareas en el cual se tendrá semanalmente organizando las metas a realizar.
- **Firebase:** Sistema Gestor de Bases de Datos (SGBD) tipo Non Structured Query Language (NoSQL).
- **Visual Paradigm for UML 16.2:** herramienta que se utilizará para la realización del UML(Unified Modeling Language)
- **Figma:** herramienta de diseño que se usará para visualizar como se desea el diseño de la aplicación final a diseñar.
- **Java 1.8.5:** lenguaje de programación orientado a objetos compatible con Android Studio.
- **GitHub:** se usará esta herramienta para tener un control de versiones y poder trabajar conjuntamente con el equipo que realiza este TFG.
- **FlatIcon:** obtendremos los iconos de nuestra aplicación de este repositorio de imágenes.

## Capítulo 3. Estado del Arte

Para la realización de este TFG y el desarrollo de esta aplicación móvil se utilizarán las tecnologías ya mencionadas anteriormente, además se tendrán en cuenta las diferentes arquitecturas de diseño de software para Android que facilitarán la comunicación entre los componentes de la aplicación y la base de datos(BBDD).

También se decidirá el patrón o la arquitectura a usar, para ello es necesario conocer la evolución del desarrollo móvil junto con los tipos de aplicaciones que existen, a continuación, se abordarán estos puntos.

### Sección 3.1 Evolución del desarrollo móvil

Se considera necesario explicar el concepto de aplicación móvil antes de hablar de la evolución de la misma. Se puede definir como cualquier programa informático (o software) que ha sido diseñado para ejecutarse en un teléfono móvil, la cual puede tener múltiples objetivos, como entretener, suplir un servicio o almacenar datos [4].

Una vez definida, es importante entender el origen de las aplicaciones móviles. Los móviles empezaron a considerarse “smartphones” alrededor de 1994 [5], debido al lanzamiento del: “IBM Simon” [6] (como se puede ver en la Figura 1), un dispositivo que incluía las primeras aplicaciones para suplir necesidades, tales como un calendario, una calculadora, un bloc de notas... entre otras.



Figura 1 Teléfono móvil IBM Simon

Las siguientes mejoras fueron orientadas al entretenimiento, se puede destacar; Tetris en 1984 [7] y Snake lanzada en 1998 [8]. En aquel entonces, era remarcable el poder tener un videojuego al alcance de la mano, por lo tanto, las compañías peleaban por tener ese puesto, el Tetris fue introducido en: “*Hagenuk MT-2000*”, en cambio Snake fue desarrollado por la compañía de telecomunicaciones sueca conocida como Nokia para varios de sus dispositivos móviles.

El punto álgido del desarrollo de aplicaciones móviles vino con el “*iPhone*”, cuando en 2008 lanzó la “*Apple store*”, donde se podían descargar y subir aplicaciones, esto fue una revolución, ya que, en ese momento, todas las aplicaciones venían preinstaladas.

En la actualidad, existen distintos tipos de aplicaciones móviles que afectan a la manera de desarrollar y a la manera de distribuirlas. A continuación, se definirán los tipos principales y sus características más remarcables.

### 3.1.1 Aplicaciones nativas vs Progressive Web App vs híbridas

**Aplicaciones nativas:** las aplicaciones nativas son aquellas que están dirigidas a un SO móvil en concreto, es decir, si está desarrollada para Android, no podrá ser ejecutada en iOS [9].

Esto facilita a los desarrolladores poder exprimir todo el potencial del SO destino; en la aplicación, implica un mejor rendimiento, mayor coherencia estética y funcional, buena experiencia de usuario, acceso a todos los sensores, acceso a la configuración del dispositivo...

**Progressive Web App (PWA):** este tipo de aplicaciones están desarrolladas en HTML5, CSS3, JavaScript, etc. Se las puede describir como webs embebidas en una aplicación móvil.

Una de sus desventajas es que no se pueden instalar como una aplicación nativa o híbrida, pero se puede acceder a ellas fácilmente creando un acceso directo, esto deriva en otra ventaja; el poco espacio en memoria que ocupan, ya que están alojadas en un servidor web, el único espacio que pueden llegar a ocupar es; al realizar un acceso directo [9].

Otra característica destacable, es la posibilidad de guardar en caché la página web para su posterior uso en modo offline.

**Aplicaciones híbridas:** estas aplicaciones son, por definición, la combinación de aplicaciones nativas y basadas en web (PWA). Lo cual supone que para desarrollarlas se deban programar tanto en lenguajes webs como orientados a objetos.

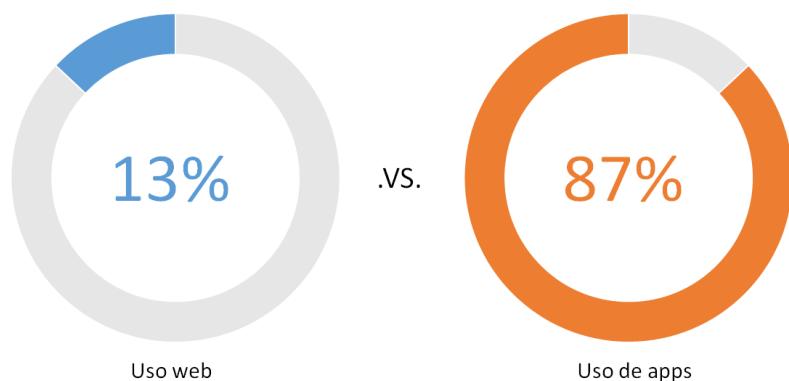
Su principal característica: sería el soporte tanto en aplicaciones móviles como en web, pudiendo llegar así a más gente, a pesar de esta ventaja, suelen tener un rendimiento inferior y a menudo hay problemas con la interfaz debido a la poca similitud de tamaño entre un dispositivo móvil y el monitor de un portátil. Estas aplicaciones a diferencia de PWA, sí que se deben instalar en el dispositivo, aunque ocupan menos espacio que una aplicación nativa [9].

Para mejor visualización y comparativa de los distintos tipos de datos, se ha realizado la siguiente tabla, véase en Tabla 1. [10]

*Tabla 1 Comparativa de las aplicaciones*

Característica	PWA	Híbridas	Nativas
<b>Soporte multiplataforma</b>	Sí	Sí	No
<b>Necesidad de internet</b>	No	Solo para eCommerce	Solo para eCommerce
<b>Instalable</b>	No	Sí	Sí
<b>Método de distribución</b>	Web	App store	App store
<b>Notificaciones push</b>	No	Sí	Sí
<b>Sensores</b>	Parciales	Parciales	Sí

Se ha explicado los tipos más significativos de aplicaciones móviles, para la correcta elección de un tipo se deben tener en cuenta varios factores; primero de todo, se va a analizar cómo usan los usuarios el teléfono móvil día a día, en España el 87% del tiempo de uso móvil (representado en Figura 2), se hace en una aplicación y el 13% en la web [11]. Un buen ejemplo de esta práctica, es el uso de Google Maps, normalmente en sistemas portátiles como móviles o tablets, se accede a ella a través de una app, mientras que en el ordenador se usa desde la web [12].



*Figura 2 Comparativa del uso web contra el uso de apps en móvil*

También se ha tenido en cuenta la desventaja principal de las PWA sobre las aplicaciones nativas, las PWA no pueden acceder a los sensores de un teléfono móvil (véase en Figura 3), a los únicos que son capaces de acceder es al GPS y al micrófono.



*Figura 3 Representación de los sensores*

Debido a los motivos descritos, se ha optado por realizar una aplicación nativa para Android.

### Sección 3.2 Entornos de desarrollo para aplicaciones móviles

Un entorno de desarrollo es un software que provee todas las herramientas necesarias para crear una aplicación móvil, generalmente incluyen soporte para uno o varios lenguajes de programación, permitiendo compilar, depurar y realizar pruebas [10]. Algunos de los entornos más usados en la actualidad son:

**Android Studio:** es un entorno de desarrollo integrado oficial para el desarrollo de apps centrado en android y basado en intelliJ IDEA. En este Integrated Development Environment(IDE) se contará con las herramientas necesarias para crear aplicaciones. Esto incluye desde el código, al diseño de la interfaz de usuario [13].

Android Studio tiene distintas funcionalidades, todas ellas relacionadas con la creación de aplicaciones para el SO Android, algunas características [14] que tiene son:

- Un sistema de compilación flexible basado en Gradle.
- Un emulador rápido, facilitandonos así las pruebas en diversos dispositivos.
- Un entorno unificado donde se puede desarrollar para todas las versiones Android.
- Integración con GitHub.
- Compatibilidad con Google Cloud Platform, que facilita la integración con Google Cloud Messaging y App Engine.
- Soporta Java y Kotlin como lenguajes de programación.

En la Figura 4 se muestra la UI de Android Studio.

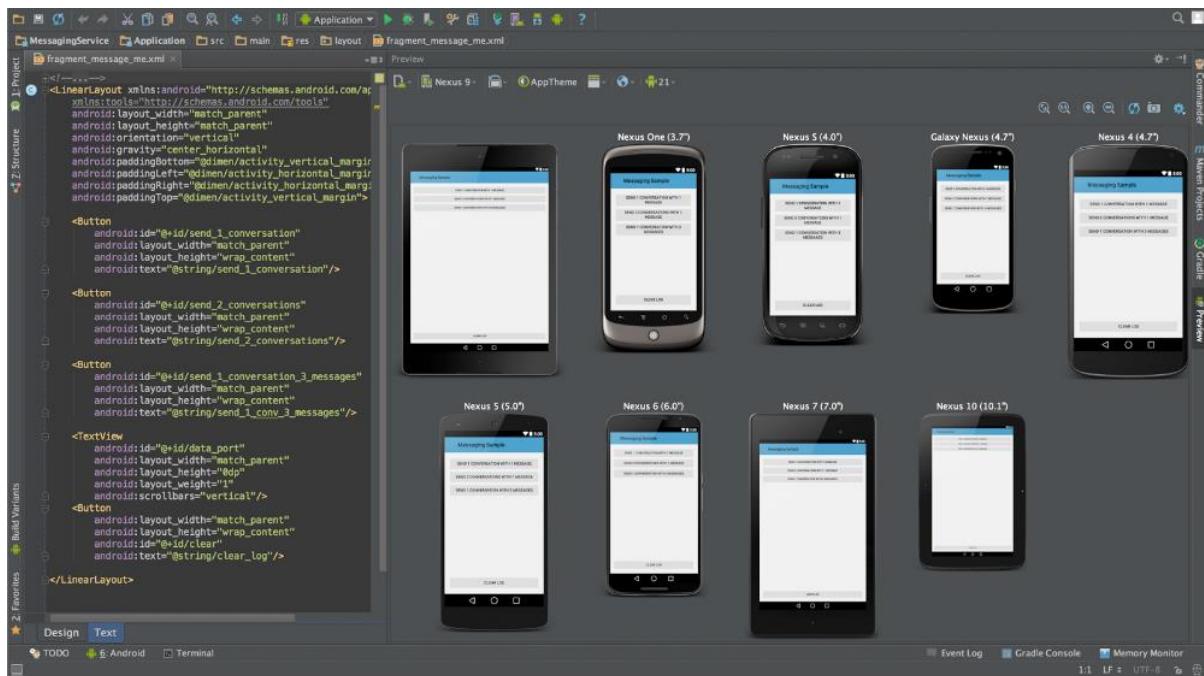


Figura 4 Entorno de desarrollo de Android Studio

**Xcode:** es un IDE para el desarrollo de aplicaciones para Apple, por ende es usado para realizar aplicaciones en: Mac, iPhone, iPad, Apple Watch, Apple TV... y todos sus productos asociados. XCode combina las funcionalidades de diseño de la interfaz de usuario, programación, testing, depuración y publicación en la App Store [15].

Algunas de las características [16] principales son:

- Los lenguajes de programación que puede compilar entre otros muchos, serían Objective-C, C, C++, Java, Python, Ruby y Swift.
- Contiene Interface Builder Built-In, que brinda la posibilidad de crear la UI de forma gráfica, y conectarla con nuestro código de la misma manera.
- Depura tu aplicación directamente desde el editor.
- Simulación Virtual de iOS.

En la Figura 5 se muestra la UI de Xcode.

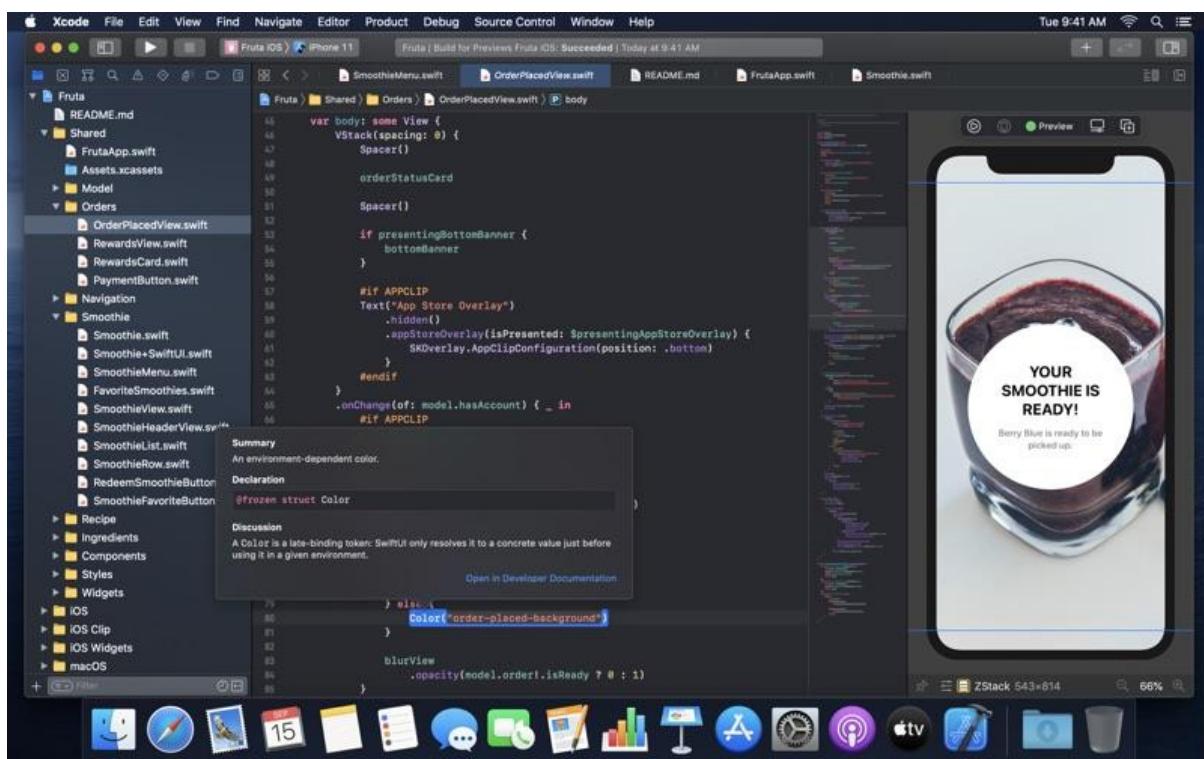


Figura 5 Entorno de desarrollo de Xcode

**Xamarin Studio:** es una herramienta centrada en realizar aplicaciones multiplataforma, soportando iOS, Android o Windows Phone. Xamarin Studio usa los recursos nativos de cada plataforma, por lo que si quieres crear aplicaciones para iOS uno de los requisitos es contar con un sistema Mac OS X [17].

Las principales características [14] de Xamarin Studio son:

- Soporta un único lenguaje de programación, C#.
- Está compilado de forma nativa, lo que genera que se pueda crear aplicaciones de alto rendimiento con aspecto nativo.
- Puede ser usado con un gran número de características útiles de .NET (Framework de Microsoft).
- Xamarin.Forms: se trata de un producto independiente diseñado para crear prototipos o aplicaciones sencillas que comparten el 100% del código en iOS y Android.

### Sección 3.3 Lenguajes de programación para aplicaciones móviles

Existen diferentes lenguajes de programación que se pueden utilizar para desarrollar una aplicación móvil, algunos de los más utilizados en la actualidad son Java, Kotlin y Swift. Cada uno de ellos posee características distintas, las cuales se explicarán en esta sección.

**Java:** es un lenguaje de programación soportado, entre otros, por Android Studio y Xcode, como cualquier otro lenguaje, tiene su propia estructura, reglas de sintaxis y paradigma de programación que se basa en el concepto de orientación a objetos (OOP). Además, es un lenguaje derivado de C por lo que muchas de sus reglas de sintaxis son similares [18].

Java posee las siguientes características:

- **Simple:** debido a que es un lenguaje derivado de C y C++, pero sin contar con las características menos utilizadas y más confusas de estos lenguajes haciéndolo mucho más sencillo.
- **Orientado a objetos:** Java trata los datos como objetos y soporta las tres características del paradigma de orientación objetos que son: herencia, polimorfismo y encapsulación.
- **Robusto:** Java realiza verificaciones en busca de problemas de compilación o ejecución, la comprobación de tipos ayuda a detectar errores en su ciclo de desarrollo, obligando así a la declaración explícita de métodos, reduciendo la posibilidad de error.

**Kotlin:** es uno de los lenguajes de programación más reciente en el sector de desarrollo de aplicaciones, está orientado a objetos, y pese a su reciente creación ya se puede encontrar en algunas aplicaciones disponibles en el mercado, como Coursera o Trello [19].

**Swift:** es un lenguaje que se integra a la perfección con código escrito en Objective-C. Se puede ver en funcionamiento en las apps para iOS de Airbnb o LinkedIn [20].

Algunas de las características más importantes de este lenguaje son:

- **Multiparadigma:** Swift acepta varios paradigmas: la orientación a objetos, la orientación a protocolos y también la orientación a programación funcional.
- **Fuertemente tipado:** es debido a que no admite ambigüedades en la declaración de variables o propiedades, aún así, Swift permite cambiar el tipo de variable, también conocido como “typecasting”.

## Sección 3.4 Otros aspectos tecnológicos

En este apartado, se explicarán otros aspectos tecnológicos destacables para la comprensión y realización del TFG.

### 3.4.1 Bases de datos

Las bases de datos son una manera de organizar información estructurada o datos que típicamente están almacenados en un sistema informático, usualmente está controlada por un sistema gestor de base de datos (DBMS o SGBD); las BBDD se modelan típicamente en filas y columnas en una serie de tablas para el procesamiento y consulta de datos sea eficaz, con esto se tiene la facilidad de acceder, administrar, modificar, actualizar, controlar y organizar fácilmente los datos. La mayoría de bases de datos utilizan lenguaje de consulta SQL para realizar las funciones Create, Read, Update and Delete (CRUD) [21].

Existen diferentes tipos de BBDD según el modelo de administración de datos; obedeciendo a su estructuración, la forma en la que se guardan los datos o los métodos de almacenamiento [22]. Algunas de estas son:

**Bases de datos relacionales:** el modelo más usado en la actualidad a la hora de implementar bases de datos es el modelo relacional, el cual permite crear relaciones entre los datos, esto proporciona una manera más eficiente y flexible de acceder a información de manera estructurada; existen varios gestores de bases de datos relacionales entre los cuales los más usados son Db2, Microsoft SQL Server, MySQL, PostgreSQL, Oracle Database o SQLite [23].

- **DB2 IBM:** es una BBDD relacional que ofrece funciones avanzadas de análisis y administración de datos, creada por la empresa IBM [24].
- **Microsoft SQL Server:** es un DBMS de tipo relacional desarrollada por la empresa Microsoft, el lenguaje utilizado es Transact-SQL (TSQL) y está disponible con una licencia Microsoft de pago.
- **MySQL:** es el SGBD más utilizado a nivel global, debido a que tiene una versión de código abierto. En este caso particular, se cuenta con dos tipos de licencia, una para código abierto y otra versión más comercial diseñada por Oracle.
- **PostgreSQL:** es un sistema de código abierto de administración de base de datos de tipo relacional, siendo mantenida por una comunidad de desarrolladores, colaboradores y organizaciones comerciales de forma libre y desinteresada.
- **Oracle Database:** es un DBMS de tipo objeto-relacional desarrollado por Oracle Corporation, dicho programa está fundamentado en la tecnología relacional de cliente y servidor.

**Bases de datos jerárquicas:** esta BBDD almacenan los datos en una estructura jerarquizada, uno de los principales objetivos es gestionar grandes volúmenes de datos. Este tipo almacena los datos en forma de registros dentro de una estructura, estos registros se conocen como nodos, se pueden definir como puntos asociados entre sí conectados para formar una estructura de árbol invertida, conteniendo la información que está enlazada a partir del nodo raíz a otros nodos descendientes como padres e hijos, los nodos padres pueden tener varios hijos, pero un nodo hijo solo puede tener un nodo padre.

**Bases de datos NoSQL:** en los últimos años el desarrollo moderno ha cambiado radicalmente, esto es debido a que estamos en tendencia a almacenar, procesar o actualizar cada vez mayores volúmenes de datos queriendo mantener la velocidad, en consecuencia, a esto lo mejor es tener una BBDD NoSQL, que, en lugar de usar tablas tradicionales, usan técnicas más flexibles como, por ejemplo; documentos, gráficos, pares de valores o columnas este tipo de sistemas [25].

Este tipo de BBDD son ideales para aplicaciones en las que procesan grandes volúmenes de datos y que requieren estructuras flexibles, como los sistemas NoSQL hacen uso de clústeres de hardware y servidores de nube, las capacidades se distribuyen de manera uniforme, existen numerosos sistemas NoSQL distintos en todo el mundo, normalmente de código abierto, aunque no existan regulaciones uniformes los distintos conceptos NoSQL se pueden clasificar en cuatro categorías principales:

- **Orientadas a documentos:** los datos están almacenados directamente en documentos de diferentes longitudes, para localizar datos se asignan distintos atributos, denominados “Tags”, que facilitan esta búsqueda; este es utilizado para sistemas de gestión de contenidos y blogs, una de la más usada es JavaScript Object Notation (**DBMS**) como formato archivo ya que permite el intercambio de datos rápido.
- **Grafos:** el entramado de datos se organiza mediante puntos nodulares y sus conexiones entre sí, este tipo es más utilizado en el ámbito de las redes sociales, por ejemplo, para representar relaciones entre los seguidores de Twitter o Instagram.
- **Clave-Valor:** las bases de datos de clave-valor guardan los datos como pares de claves y valores. Los valores individuales están asignados a claves específicas y funciona como clave (key) y representan un valor (value), cuando se crea la clave se crea de manera única un índice que se usa para facilitar la búsqueda en la base de datos.
- **Orientadas a columnas:** los datos se guardan en columnas, por cada entrada de datos que se tenga se crea una columna por lo que los datos se van guardando uno debajo de otro, Estas suelen ser utilizadas cuando hay que realizar muchas transacciones rápidamente en grandes

cantidades de datos debido a que si se desea leer y evaluar un registro basta con cargar un bloque y no es necesario leer la base de datos completa [26].

En la Figura 6 se muestra una infografía de los distintos tipos de BBDD.



Figura 6 Representación de las diversas bases de datos

### 3.4.2 Patrones de arquitectura

**modelo-vista-controlador (MVC):** permite separar la aplicación en componentes, un ejemplo de esto podría ser un usuario que solicita un dato, el controlador se encargará de trabajar con el modelo para realizar las acciones que el usuario haya solicitado, el controlador también seleccionará la vista a mostrar y los datos del modelo. Se puede ver un ejemplo en la Figura 7 . [27] [28] [29] [30]

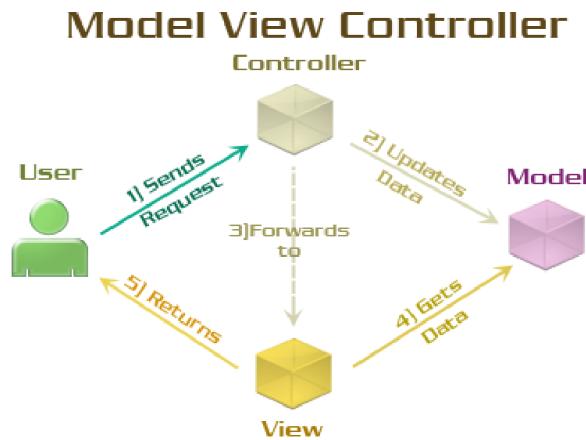


Figura 7 Infografía del MVC

Como su propio nombre indica en MVC, tenemos tres componentes principales mostrados en la Figura 8:

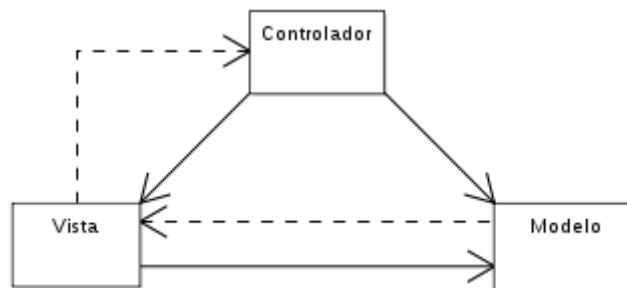


Figura 8 Esquema del MVC

- La **vista**, se refiere a la parte visible de una aplicación, se podría calificar como la UI la cual muestra la información del modelo.
- El **modelo** tiene la responsabilidad de relacionar los datos con los que la aplicación va a operar, aquí se establece la lógica de negocio y representa el estado de la app. El modelo normalmente es independiente de los demás componentes.
- El **controlador** responde a eventos que se encuentran en la vista, es el vínculo que une al modelo con la vista, por lo tanto, selecciona qué vista se muestra en cada momento. Solo existe un controlador para manejar todas las vistas.

**MVP**: esta arquitectura es una variante del MVC, en este caso la vista no se relaciona con el modelo, y existe una capa de presentación por cada vista a mostrar. Se muestra un ejemplo de interactividad en la Figura 9. [31] [28] [30]

## Model View Presenter



Figura 9 Infografía del MVP

MVP se forma por tres componentes principales:

- El modelo mantiene la responsabilidad de relacionar los datos y representar el estado de la app.
- La vista es quien recibe la interacción del usuario, y esta se comunicará con la capa de presentación.
- El presentador es quien reacciona a los eventos mandados por la vista, se encarga de pedir la información al modelo y de recibirla.

**MVVM:** Su característica principal es el desacoplamiento de la interfaz de usuario con respecto a la lógica de la aplicación [32] [30]. Se compone de tres partes:

- El modelo
- La vista
- El modelo de vista, se trata de un actor intermediario entre ambos, en este apartado se desarrolla toda la lógica de presentación.

### Sección 3.5 Aplicaciones Similares

Se han elegido tres aplicaciones móviles con funcionamientos similares a la aplicación a desarrollar, todos con sistemas de guardado de productos o listas de la compra [33]. En este apartado, se explicará el funcionamiento de estas aplicaciones.

**Out of milk:** es una aplicación disponible actualmente tanto en dispositivos Android como en iOS [34], y entre sus funcionalidades principales se encuentran:

- Escanear los códigos de barras de los productos para añadirlos a la lista de compra.
- Administrar distintas categorías de alimentos.
- Contiene un libro de recetas donde se pueden guardar las recetas favoritas del usuario.
- Permite compartir la lista de compra con diferentes personas, teniendo control de acceso de cada una de ellas.



Figura 10 Vista de artículos (Out of milk)

Figura 11 Vista de añadir un artículo (Out of milk)

**Bring!:** es una aplicación que simplifica la lista de compra organizando los productos en diferentes categorías, está disponible en dispositivos Android y en iOS [35], una de sus características principales es su vinculación con asistentes de voz como Google o Amazon, algunas de las funciones más distinguidas son:

- Permite tener varias listas de la compra en caso de tener más de una vivienda.
- Cuenta con categorías para productos que no son alimenticios como Aseo y Salud.

- Tiene un apartado de inspiración de recetas para añadir los ingredientes de dichas recetas a lista de compra.
- Una vista con los alimentos utilizados recientemente.



Figura 13 Vista inicial de la aplicación (Bring!)



Figura 12 Vista de recetas (Bring!)

**Listonic:** una aplicación con la cual se pueden administrar distintas listas de compra, puede ser instalada en dispositivos Android como en iOS [36], la gran característica de esta aplicación es la posibilidad de ver catálogos de distintos supermercados para buscar ofertas, además se puede encontrar con un apartado de basura donde el usuario podrá encontrar las listas que se hayan eliminado anteriormente, en caso de querer recuperarlas.

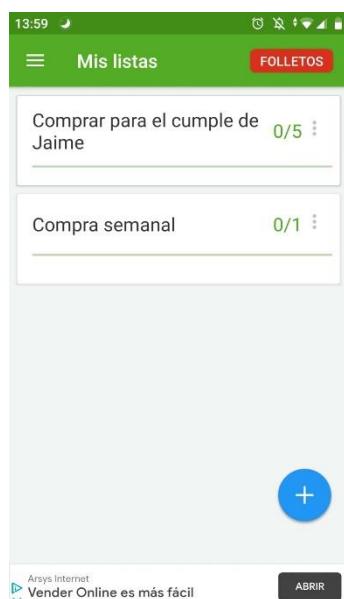


Figura 14 Vista de distintas listas (Listonic)

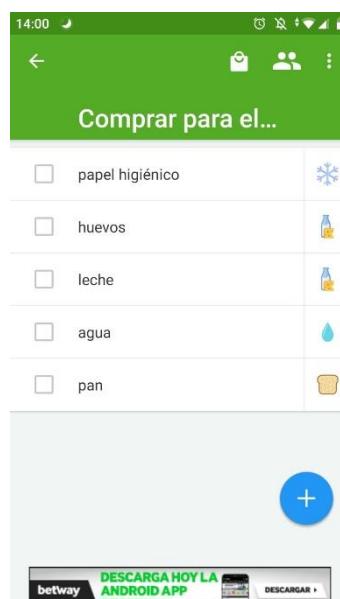


Figura 15 Vista de productos (Listonic)

## Capítulo 4. Método del Trabajo

En este capítulo se expondrá la metodología que se va a utilizar para la realización de este proyecto. En este caso se ha elegido la metodología Kanban, últimamente en alza debido a su sistema de tarjetas llevado a sistemas electrónicos, como el caso de Trello, el cual se utiliza en el desarrollo de este proyecto.

### Sección 4.1 Metodología Kanban

Kanban es una metodología ágil, el nombre del origen japonés está compuesto por las palabras Kan (visual) y Ban (tarjetas), por lo que se refiere a la implementación de tarjetas visuales con el propósito de gestionar un proyecto.

Es una metodología muy simple, el tablero se divide en columnas y filas donde cada columna visualiza una fase de su proceso y las filas representan diferentes tipos de actividades, las columnas más usadas son: tareas pendientes, en proceso y finalizadas [37], como se puede ver en la Figura 16.

Para la aplicación a desarrollar es importante el orden en el cual se efectúan las tareas y la organización del equipo, por este motivo se utilizará la metodología Kanban, ya que, permite que cada miembro del grupo se enfoque en las actividades que tiene por hacer generando una producción altamente efectiva y eficiente [38].

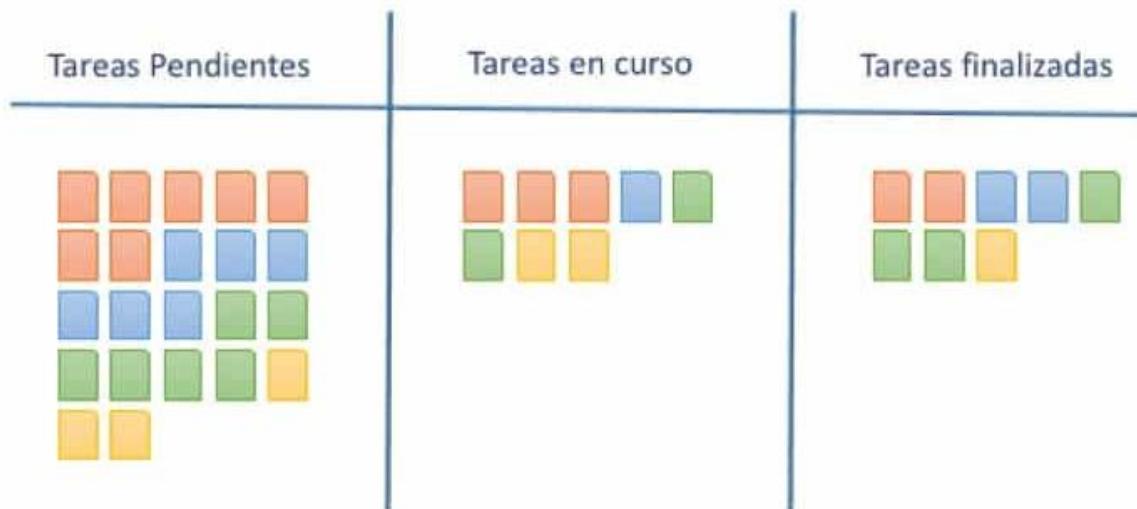


Figura 16 Metodología Kanban

#### 4.1.1 Principios básicos del método Kanban

El método Kanban se enfoca en cubrir un proyecto de manera global, esto asegura un buen resultado y calidad gracias a las diferentes fases con las que cuenta para poder agilizar tareas con sus cinco principios básicos que se definirán a continuación [39]:

- **Visualización:** Kanban es totalmente transparente, de forma que se tiene acceso a todas las tareas en cualquier momento. Permitiendo así, organizarse en los diferentes bloques y realizar modificaciones en cualquier momento para el buen funcionamiento del equipo.
- **Eliminar interrupciones:** cambiar el enfoque de las tareas puede dañar seriamente el proceso al igual que la multitarea podría provocar alguna pérdida de tiempo por esta razón Kanban maneja límites de trabajo en el proceso para que un usuario no tenga demasiadas tareas en uno solo columna.
- **Gestionar el flujo:** por flujo se refiere al movimiento de las tareas entre las columnas dependiendo del estado de la misma.
- **Hacer las políticas explícitas:** los procesos deben estar bien definidos para que todos los integrantes entiendan lo que se debe hacer a continuación. Cuando todos los integrantes estén familiarizados con el objetivo común, se podrá trabajar y tomar decisiones con respecto a los cambios.
- **Circuitos de retroalimentación:** con esto se hace referencia a realizar reuniones periódicamente para la transferencia de conocimiento entre el equipo a través de reuniones para sincronizar los diferentes procesos.
- **Mejorar colaborando:** se debe tener una visión compartida acerca de los problemas que deben superarse, los equipos que tienen un entendimiento compartido de las teorías sobre el trabajo, el flujo de trabajo, el proceso y el riesgo, tienen más probabilidades de crear una comprensión compartida de un problema y sugerir acciones de mejora que pueden acordarse por consenso.

### Sección 4.2 Planificación

Después de definir la metodología se explicará a detalle cómo se implementará en el proyecto

#### 4.2.1 Fase I Visualización

**Tareas necesarias (TN):** se analizará cuáles son los pasos a seguir antes de realizar la aplicación.

- TN1.** Identificar el público objetivo al que queremos dirigirnos.
- TN2.** Identificar entre ese público objetivo, qué dispositivos y versiones Android son las que más se utilizan.
- TN3.** Establecer unos requisitos mínimos con la intencionalidad de llegar al máximo público, pero sin perder funcionalidades en la aplicación.
- TN4.** Establecer el tipo de aplicación que desea el cliente.
- TN5.** Establecer en qué sistemas operativos va a funcionar.

#### 4.2.2 Fase II Eliminar las interrupciones

Para la realización de esta fase se tuvo en cuenta la cantidad de tareas que se debían hacer por semana para llegar a la finalidad del proyecto en el tiempo adecuado, teniendo esto organizado se dividen en 2 (al ser dos personas en el equipo) y estas se reparten teniendo en cuenta que el límite de tareas por persona es dos, teniendo que estar las tareas relacionadas entre ellas para evitar la multitarea. Se puede ver un ejemplo de la metodología puesta en acción a través de Trello en la Figura 18.

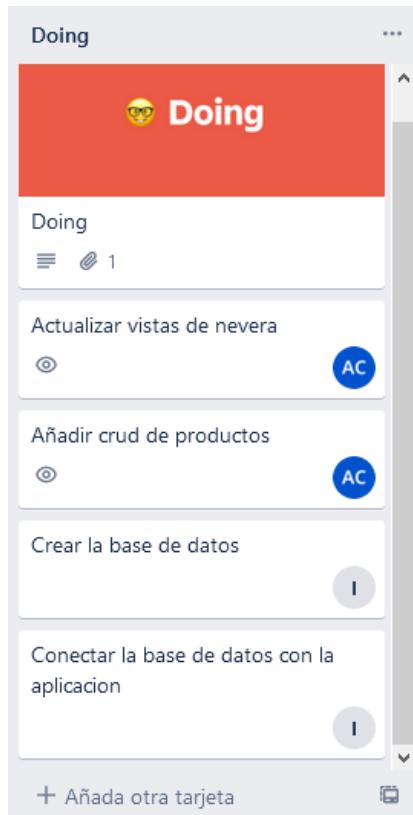


Figura 17 Organización Kanban

#### 4.2.3 Fase III Gestionar el flujo

El flujo del desarrollo se basó en tener en cuenta la cantidad de tareas que tiene cada integrante del equipo y del tiempo en finalizarla, como se explicó anteriormente, se tiene un límite de dos tareas semanales, pero si una tarea se termina antes de lo previsto se continúa con las tareas que se encuentren en el apartado de “To do” del tablero, para mantener el ritmo de desarrollo y evitar la pérdida de tiempo que puede ser útil para las partes finales del proyecto.

#### 4.2.4 Fase IV Hacer las políticas explícitas

Esta fase se desarrolló a lo largo de proyecto debido a que al terminar un proceso se comparte la información de cómo se ha desarrollado con el compañero, esto para tener claro que se está haciendo y como se está haciendo, esta información le facilita al grupo entender a qué se quiere llegar en caso de que en un proceso haya un cambio inesperado.

#### 4.2.5 Fase V Circuitos de retroalimentación

En relación con la fase anterior para el desarrollo de este TFG se realizaron reuniones semanales para la sincronización de las tareas o procesos que se plantean en el tablero, ya que la comunicación con el grupo es fundamental.

#### 4.2.6 Fase VI Mejorar colaborando

En esta fase se subdividió el tablero en tres bloques primordiales que son: el diseño, el código y la memoria. Podría llegar a ser contradictorio tenerlo todo en las mismas listas, un integrante del grupo puede estar desarrollando código y haciendo cambios en el diseño al mismo tiempo, así se consigue optimizar el tiempo.

Con esta estructura se genera una buena compresión por parte de los integrantes del grupo al momento de desarrollar una tarea de alguno de estos tipos, como se muestra en la Figura 18.

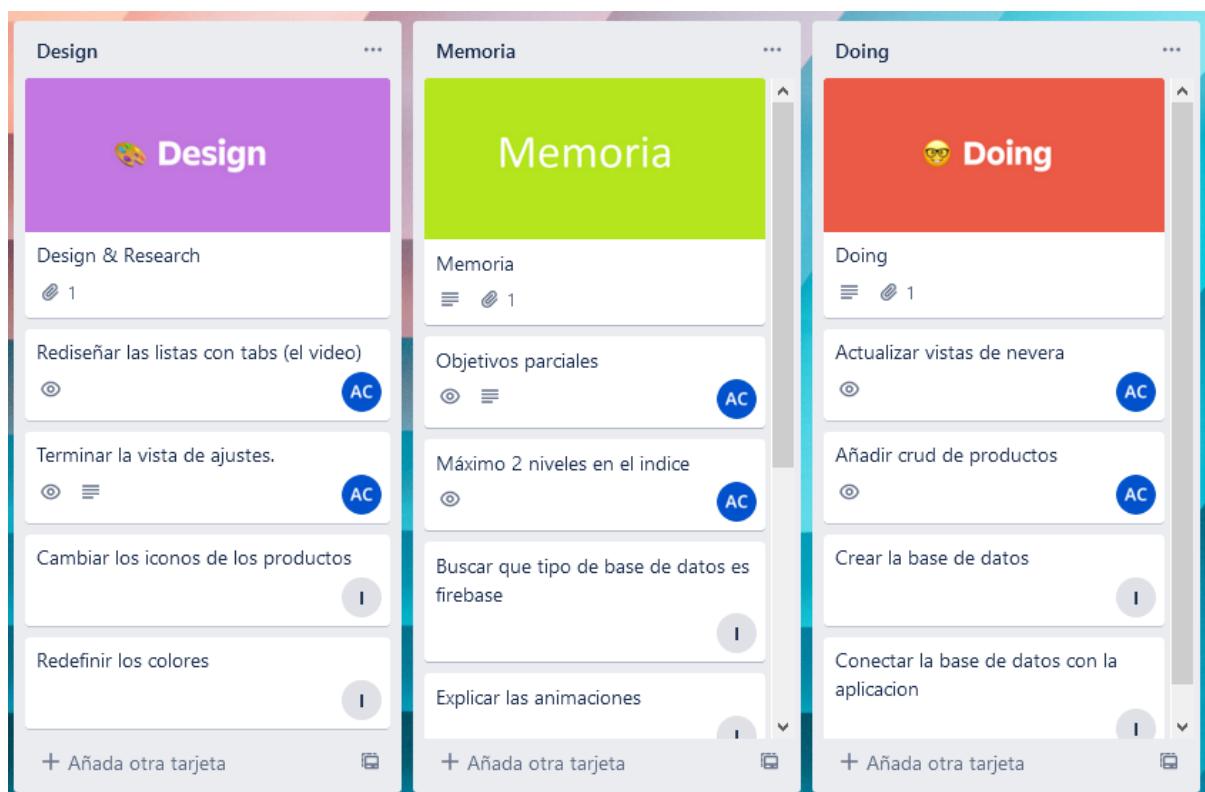


Figura 18 Estructura en Trello

## Capítulo 5. Resultados

### Sección 5.1 Modelo Entidad/Relación

A continuación, se mostrará el ER en la Figura 19 realizado para obtener un correcto funcionamiento a la hora de almacenar los datos.

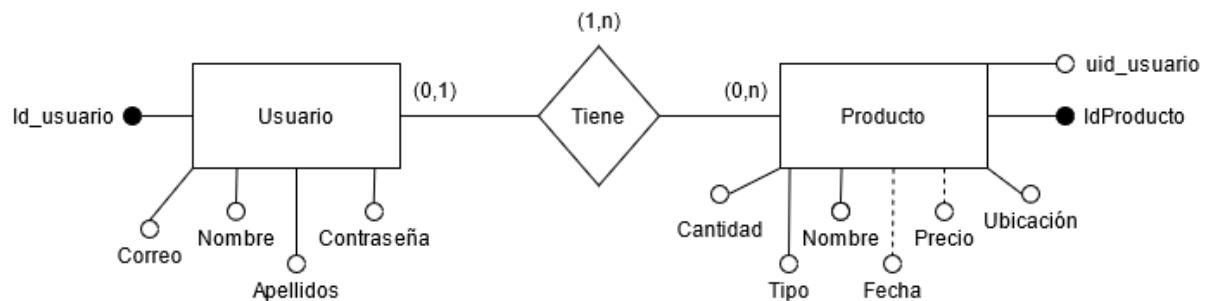


Figura 19 Representación del modelo entidad/relación

### Sección 5.2 Modelo de casos de uso

En esta sección se mostrará el modelo de casos de uso visible en la Figura 20, como único actor un usuario el cual podrá registrarse, cambiar su contraseña, logearse y gestionar los productos.

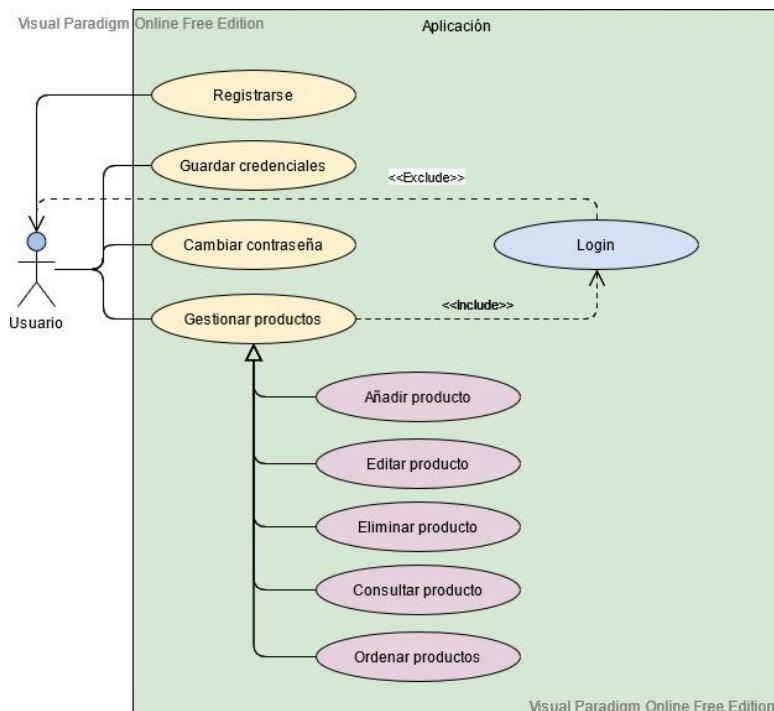


Figura 20 Modelo de caso de uso

## Sección 5.3 UML

### 5.3.1 Vistas

Representación de las vistas utilizadas para el desarrollo del proyecto, estas vistas son utilizadas tanto por activitis y fragments, para darle la parte visual. Figura 21.

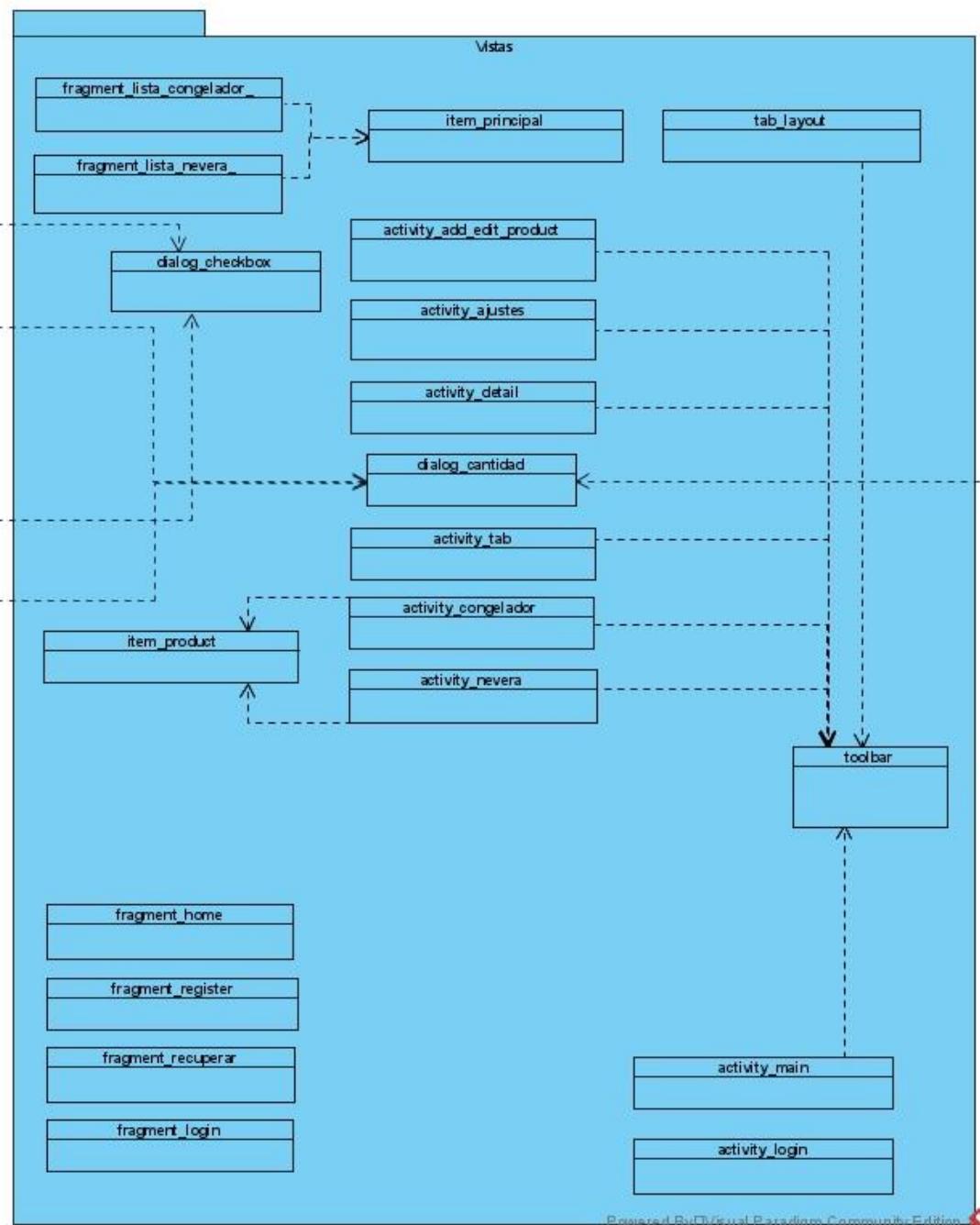


Figura 21 Vistas UML

### 5.3.2 Login

Representación de las relaciones del LoginActivity con los fragments que se pueden visualizar desde el. Figura 22.

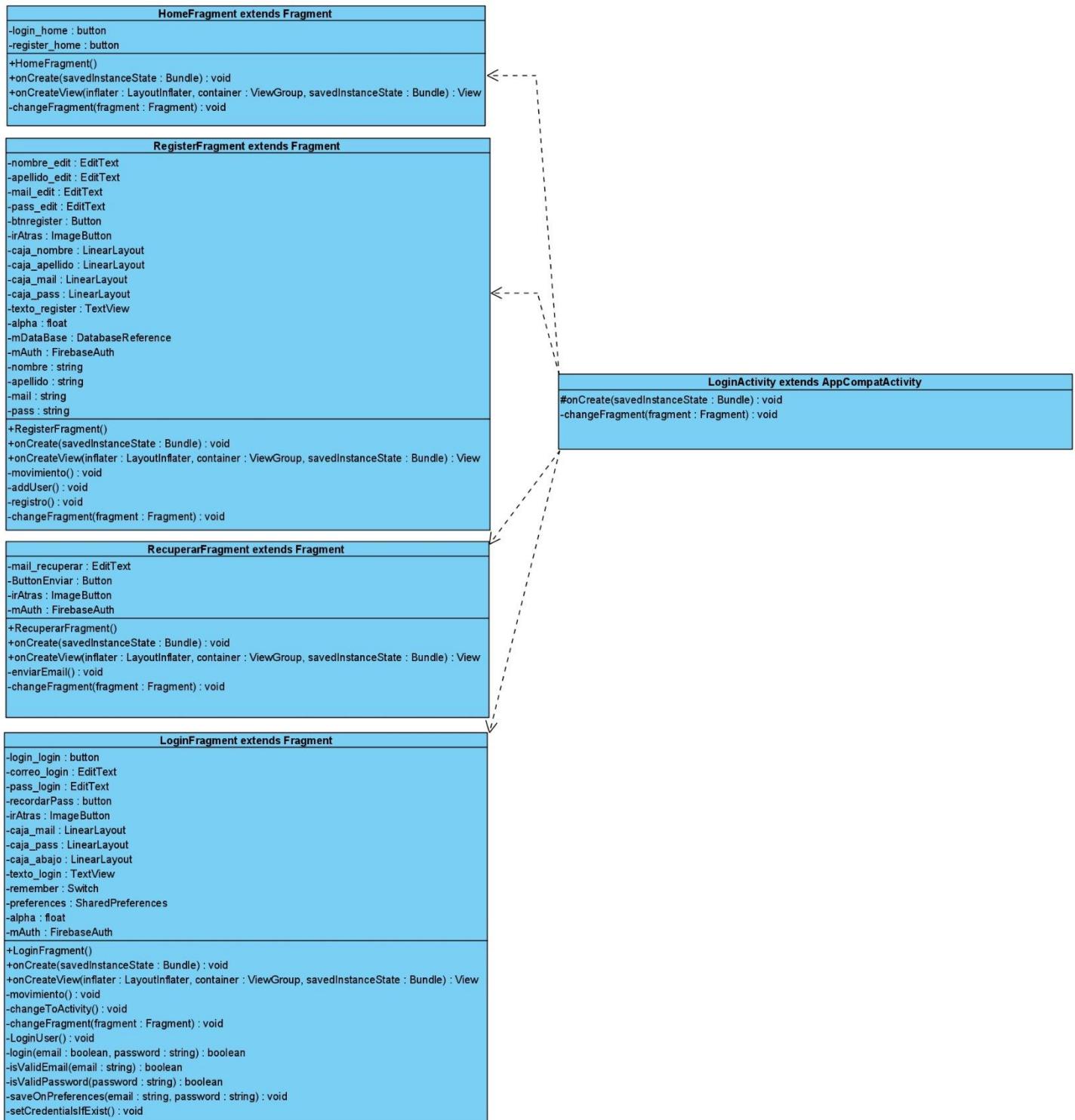


Figura 22 Login UML

### 5.3.3 Útil

Representación de las relaciones y clases que se encargan del Login y Notificaciones, las cuales tienen relación con Splash debido a que es donde se ejecutan. Figura 23.

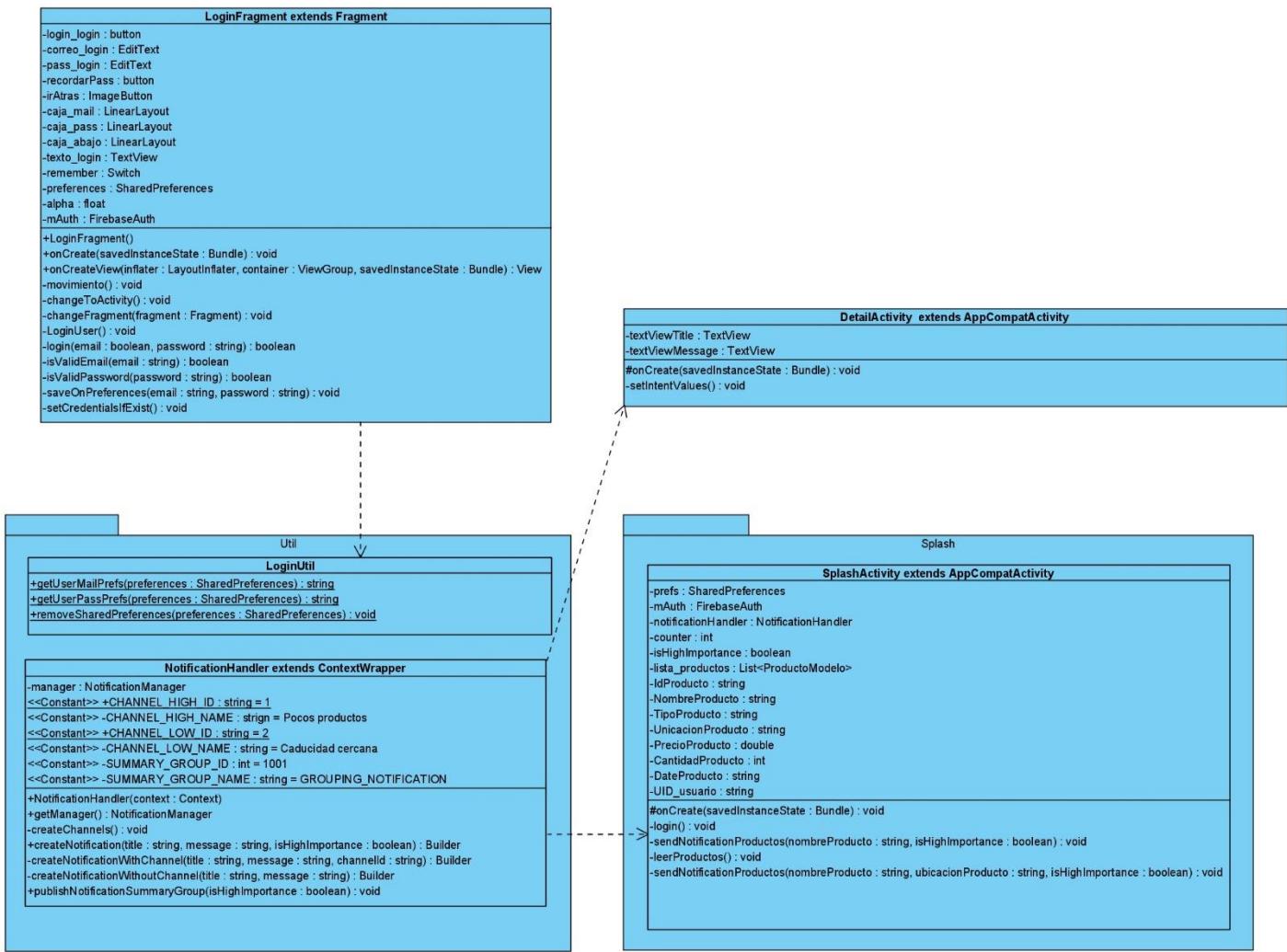


Figura 23 Util, notificaciones y login UML

### 5.3.4 Activities

A continuación se ven todos los activities encontrados en la aplicación, cada uno tiene relación con su layout, se ha cortado para que sea más legible. Se puede observar en la Figura 24.

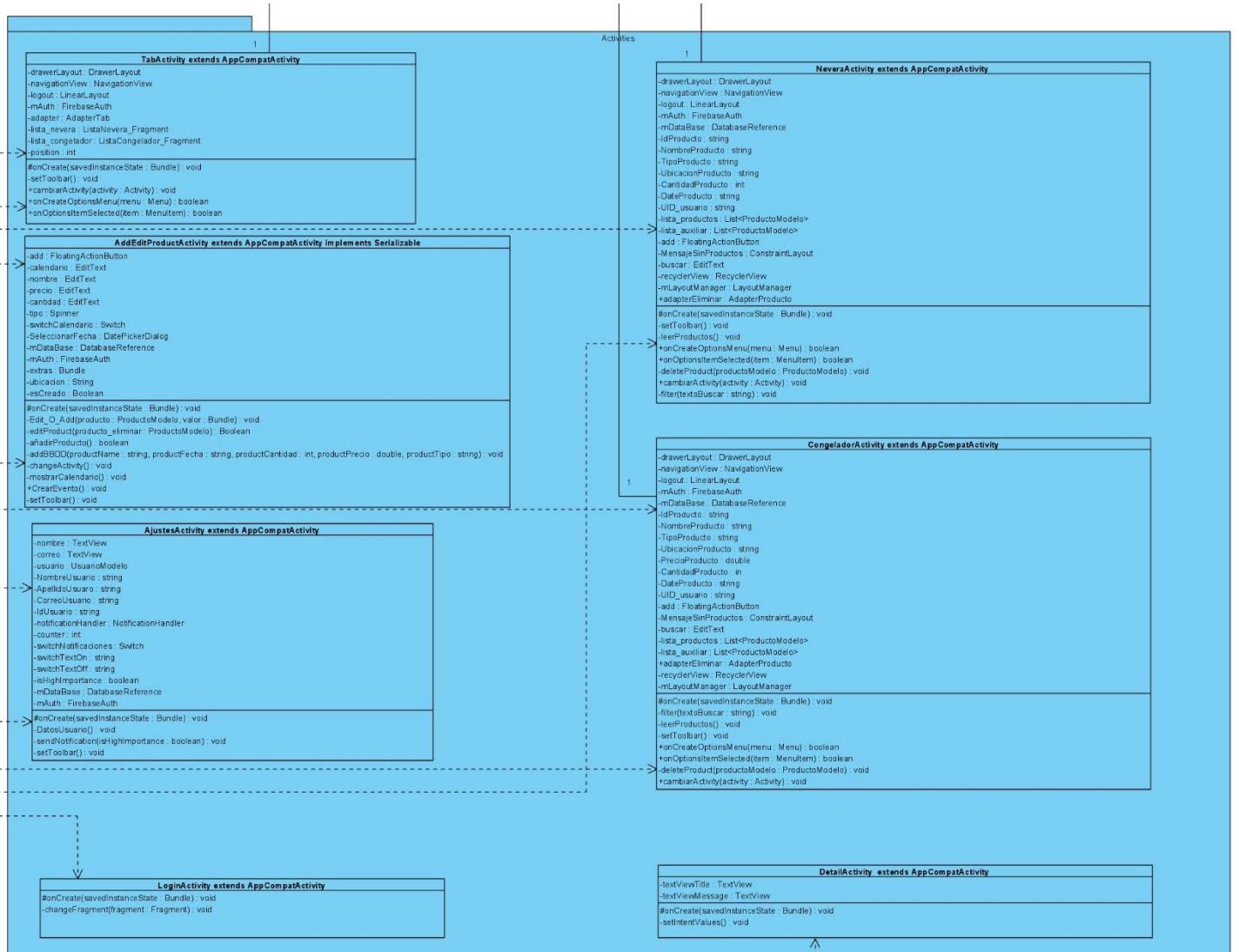


Figura 24 Todos los activities

### 5.3.5 Fragments

En esta parte se observa todos los fragments de la aplicación, cada uno está relacionado con su layout, para más legibilidad se ha omitido en la imagen. A su vez los fragments están anidados en un activity, es una manera de cambiar de pestaña más rápidamente y consumiendo menos recursos. Figura 25.

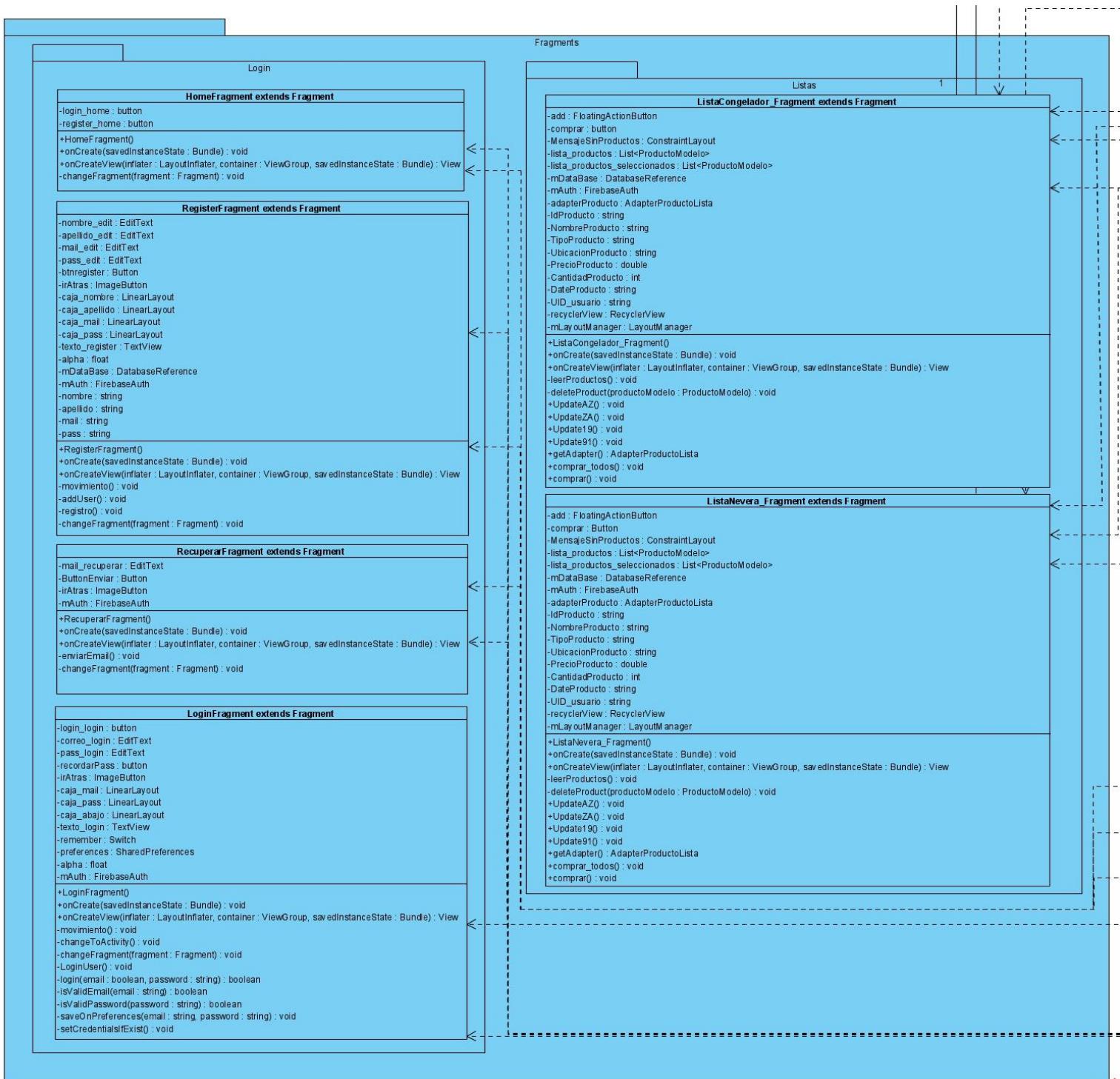


Figura 25 Todos los fragments

### 5.3.6 Modelos

Representación de los modelos, en este caso se tienen dos, uno para productos y otro para usuarios. Productos lo utilizan todas las clases que visualicen productos. Y usuarios se utiliza en todas aquellas clases del login. Figura 26.

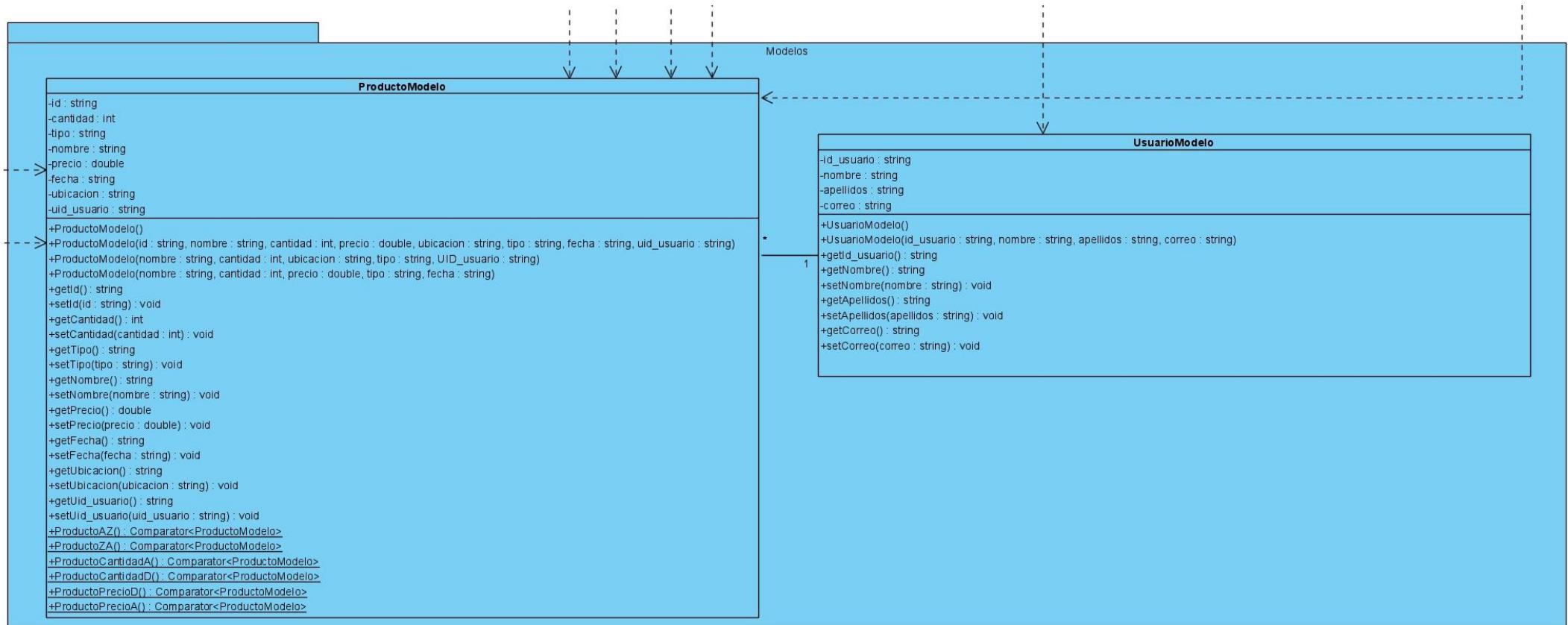


Figura 26 Modelos UML

### 5.3.7 Adapters e interfaces

Se muestra la relación de los adapters con sus views y con las interfaces correspondientes para controlar los eventos. Figura 27.

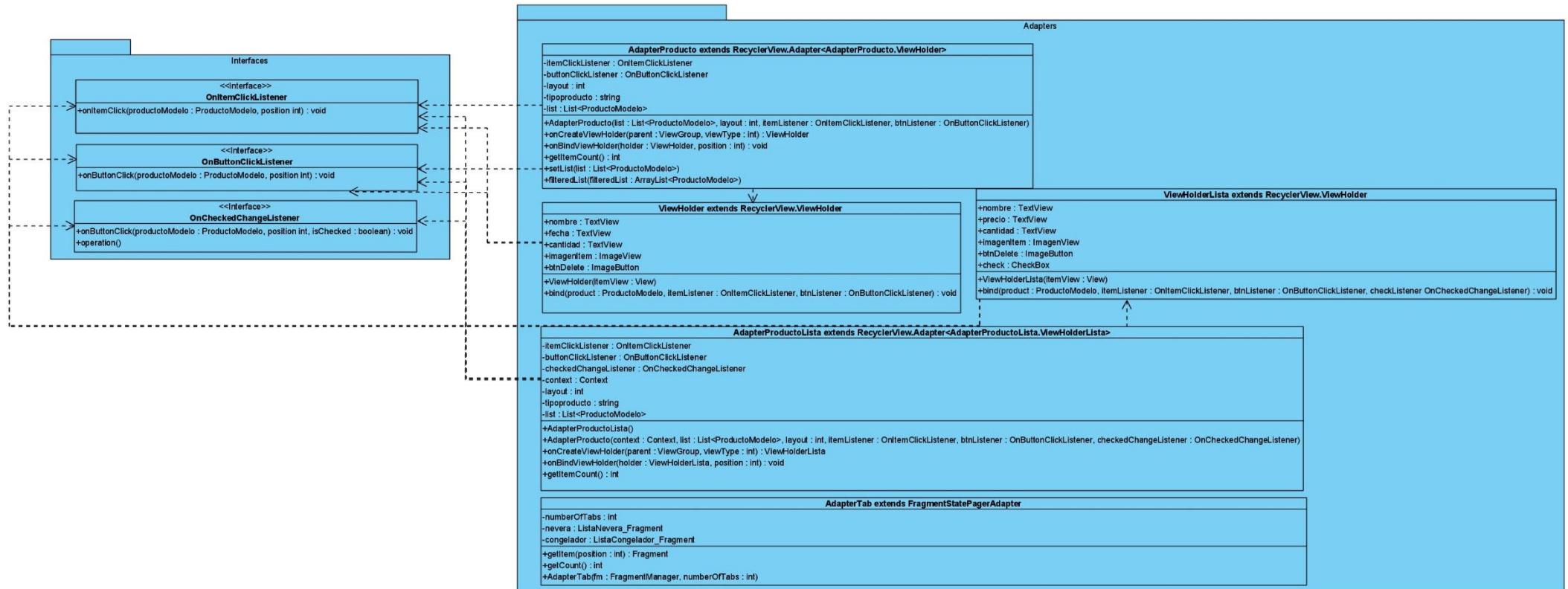


Figura 27 Adapters UML

### 5.3.8 Adapters con activities

Los adapters son los encargados de controlar los eventos de los productos, en este caso los activities tienen dos eventos, onClick que controla el editar producto y el OnButtonClick que controla el eliminar producto. Figura 28.

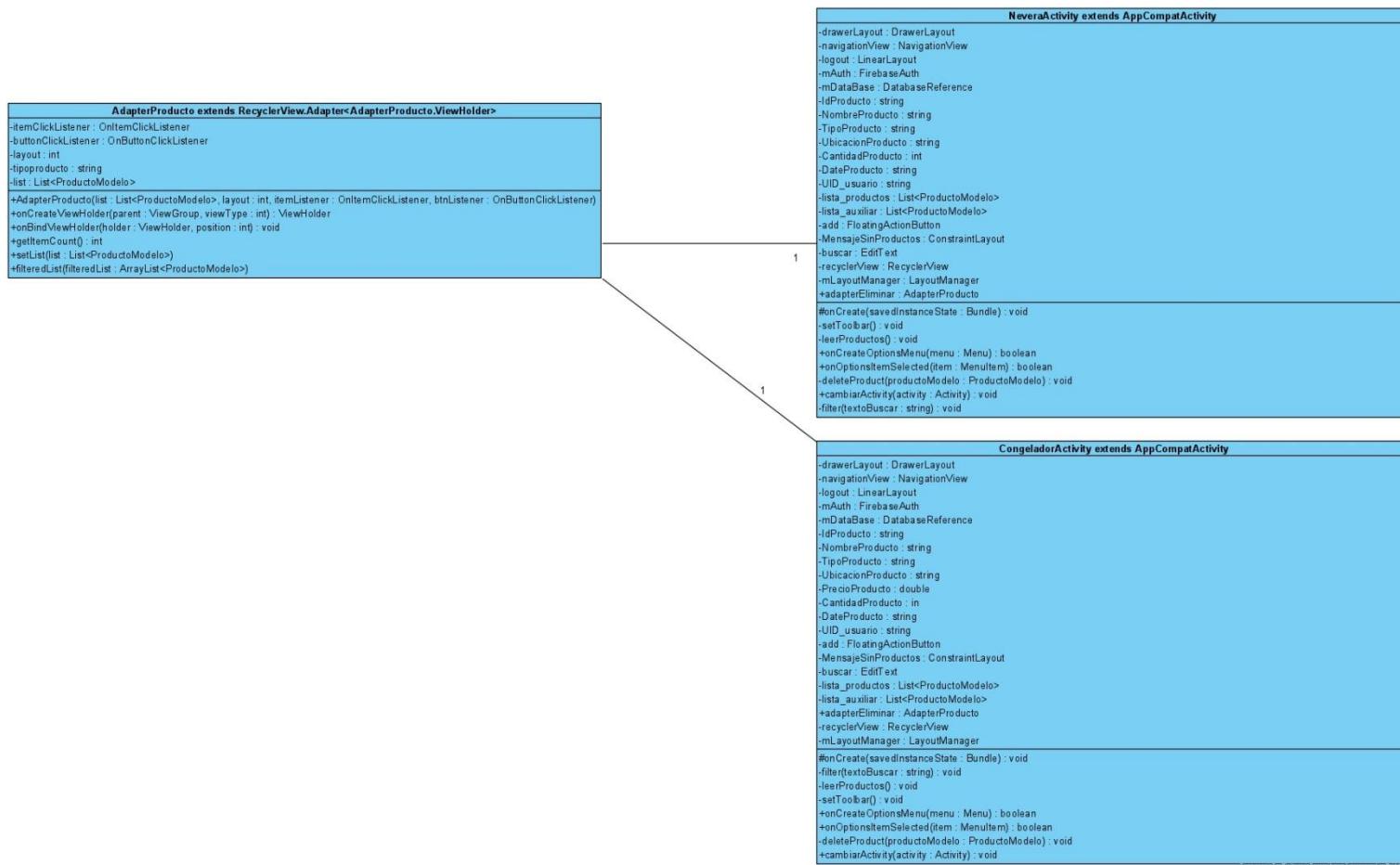


Figura 28 Adapters con relación hacia actividades UML

### 5.3.9 Adapters con listas

En esta relación, el adapter a parte de tener los mismos controladores de eventos que el adapter de activitis, tienen un evento del check, que controla que productos están seleccionados para comprarlos. Figura 29.

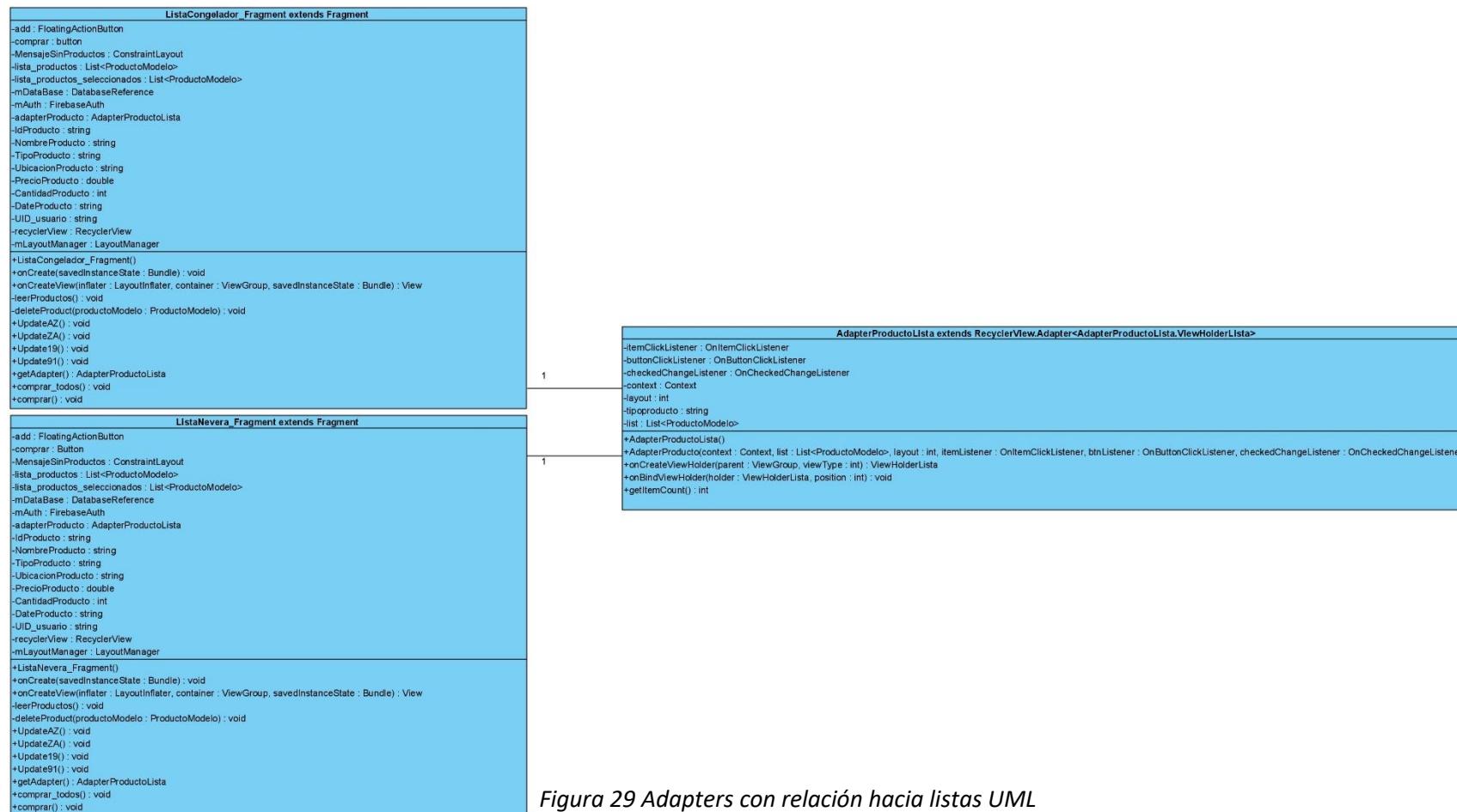


Figura 29 Adapters con relación hacia listas UML

Powered by CrossMark

### 5.3.10 Drawer

Representación de las clases que tienen la opción de mostrar el drawer. Figura 30.

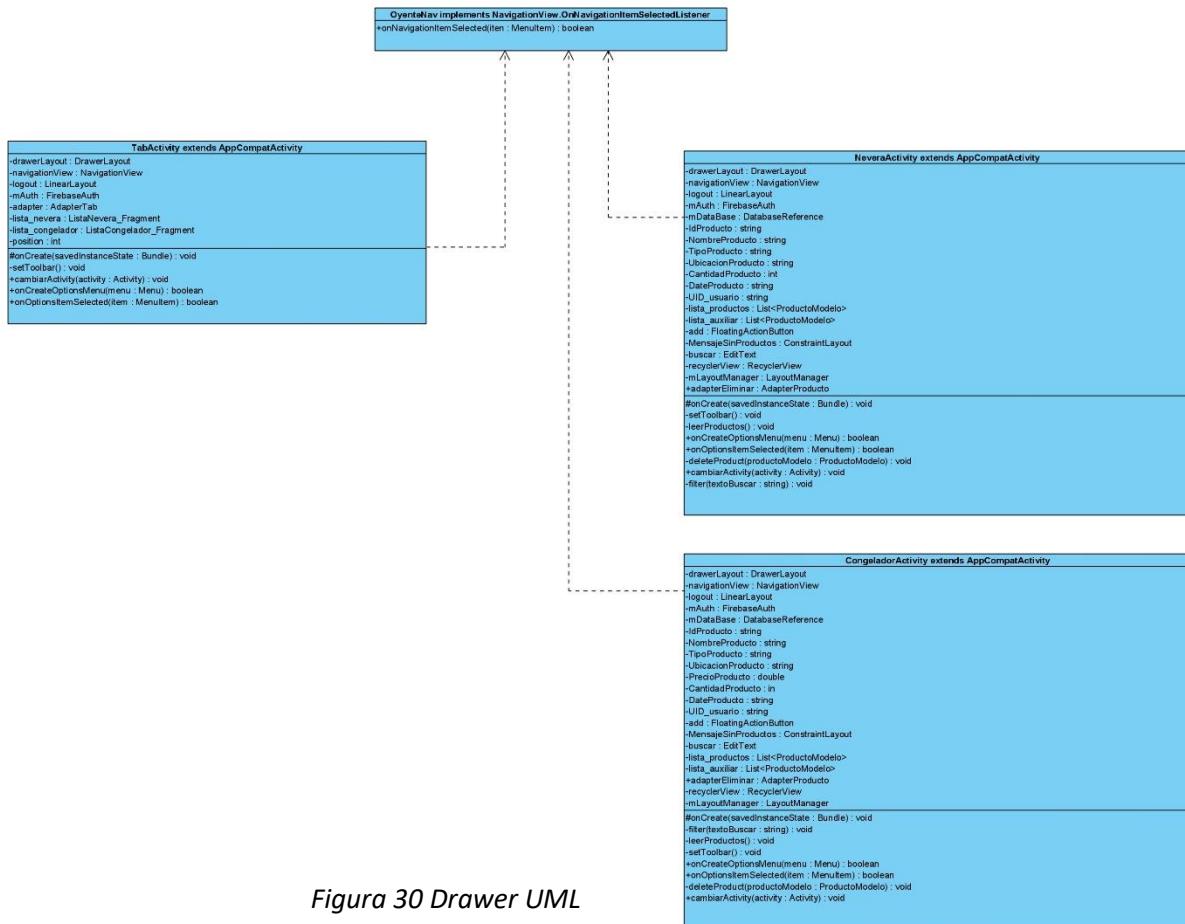


Figura 30 Drawer UML

---

## Sección 5.4 Vistas de la app

Se explicarán y mostrarán las vistas realizadas añadiendo partes del código para mejor entendimiento.

### 5.3.1 Splash

La aplicación cuenta con un Splash activity el cual permite cargar datos antes de que inicie completamente la aplicación, como por ejemplo las notificaciones, se puede ver la vista del Splash en la Figura 31.

En el splash se puede ver el logotipo representativo de la aplicación, al terminar de cargar pasará a la vista inicial de inicio de sesión.



Figura 31 Splash

### 5.3.2 Inicio de sesión

Para acceder a la aplicación es necesario estar logeado con un usuario, para esto se desarrolló un login en el cual se puede iniciar sesión con los usuarios registrados en la BBDD, para registrar a los mismos se creó una vista register.

Al momento de iniciar sesión se tendrán que llenar los campos para el correo y la contraseña del usuario, como se puede visualizar en la Figura 34, en caso de que el usuario quiera guardar su sesión puede hacer check en el switch de “Recordame”. Si el usuario decide guardar la sesión, la próxima vez que abra la aplicación, no le pedirá las credenciales y se logeará automáticamente.

Si un nuevo usuario quiere registrarse deberá llenar los datos de nombre, apellido, correo y contraseña, véase la Figura 33. Cuando estos datos estén rellenos terminará el registro al hacer click en el botón de registrarse.



Figura 32 Vista Inicial

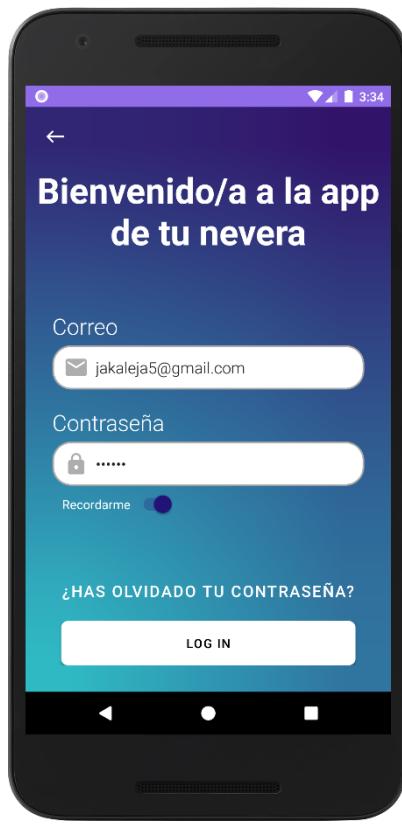


Figura 34 Inicio de sesión



Figura 33 Registro de usuarios

El login se realiza con la ayuda de métodos propios de Firebase, debido a que se ha decidido usar su sistema de usuarios, para así proporcionar más seguridad a los usuarios.

A continuación, se muestra el método principal de logeo:

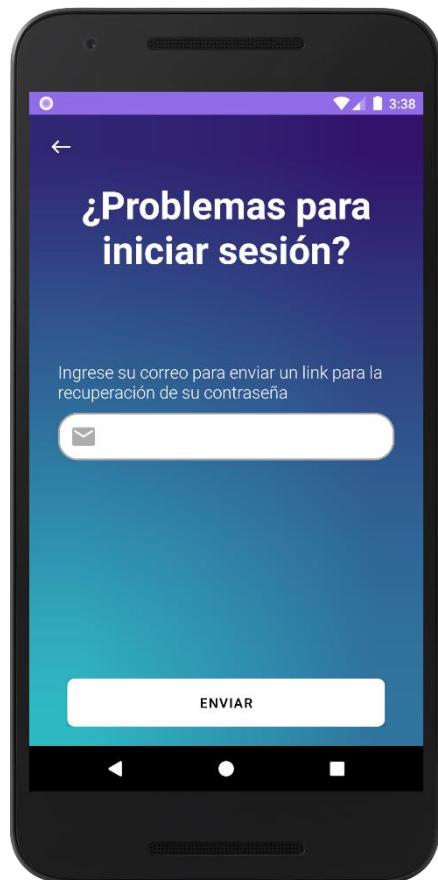
```
//Método para loguear el usuario
private void LoginUser() {
    //Obtenemos lo que ha escrito el usuario
    String email = correo_login.getText().toString();
    String pass = pass_login.getText().toString();

    if(login(email,pass)) {
        mAuth.signInWithEmailAndPassword(email,
        pass).addOnCompleteListener(getActivity(),new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {

                    saveOnPreferences(correo_login.getText().toString().trim(),pass_login.getText().toString().trim());
                    changeToActivity(); //Si el usuario y contraseña son correctos,
                    se carga el NeveraActivity.
                }
            }
        });
    }
}
```

```
        } else {
            Toast.makeText(getApplicationContext(), "Error, compruebe el usuario o
contraseña", Toast.LENGTH_SHORT).show();
        }
    });
}
}
```

En caso de que un usuario olvide su contraseña podrá recuperar su sesión en el ítem de “¿Has olvidado tu contraseña?” en esta Figura 35 podrá llenar sus datos y recibirá un correo para recuperar la contraseña.



*Figura 35 Problema de inicio de sesión*

### 5.3.3 Notificaciones

A partir de Android 8.0 todas las notificaciones deberán pertenecer a un canal, sino no aparecerán, esto es debido a que a partir de esta versión se da la posibilidad al usuario de deshabilitar en una misma aplicación distintos tipos de notificaciones.

Se visualiza en la Figura 37, en este caso es Chrome.

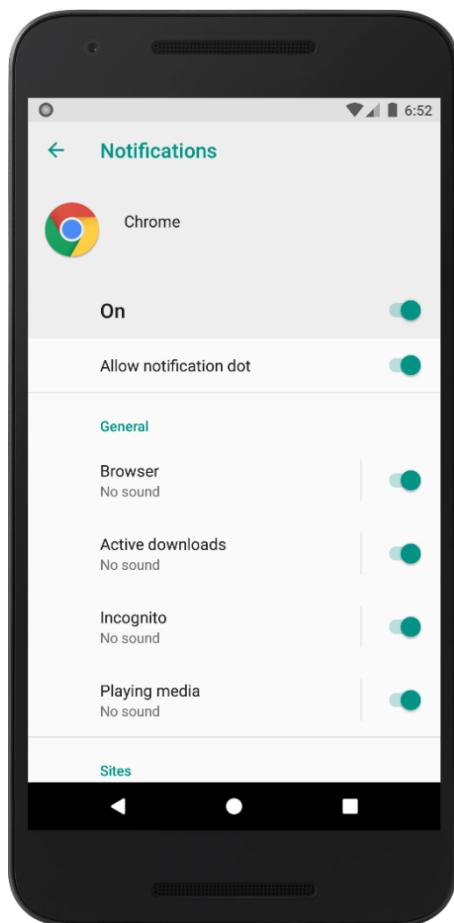


Figura 37 Ajustes de notificaciones chrome

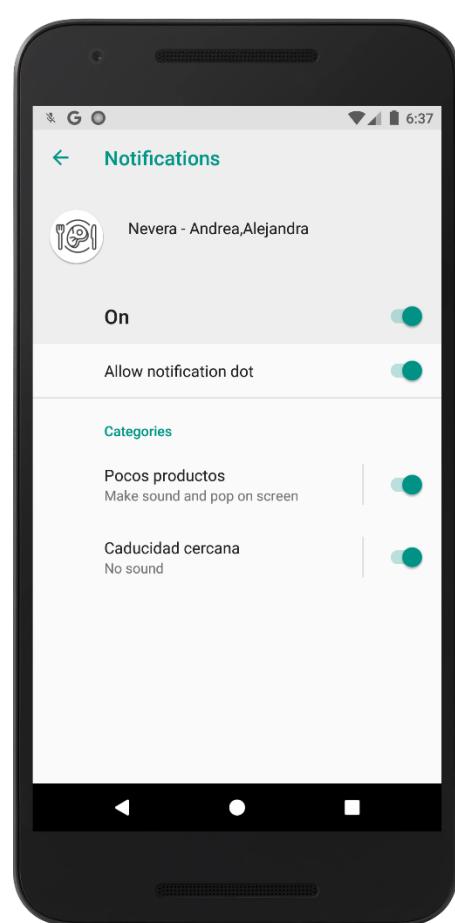


Figura 36 Canales de la aplicación

Teniendo esto en cuenta, y por la comodidad de nuestros usuarios, se ha decidido crear los siguientes canales, se puede visualizar en la Figura 36.

Con estos canales el usuario podrá decidir si desea recibir notificaciones acerca los productos o la caducidad de ellos.

A continuación, se enseña por código como se crean las notificaciones:

```
private Notification.Builder createNotificationWithChannel(String title, String message, String channelId, PendingIntent pendingIntent) {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        Intent intent = new Intent(this, DetailActivity.class);
        intent.putExtra("title", title);
        intent.putExtra("message", message);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);

        //Para la funcionalidad de ir a la app al pulsar la notificacion
        PendingIntent pIntent = PendingIntent.getActivity(this, 0, intent,
PendingIntent.FLAG_CANCEL_CURRENT);

        return new Notification.Builder(getApplicationContext(), channelId)
            .setContentTitle(title)
            .setContentText(message)
            ..setStyle(new Notification.BigTextStyle()
            .bigText(message)
            .setBigContentTitle(title))
            .setColor(getColor(R.color.teal_700))
            .setSmallIcon(android.R.drawable.stat_notify_chat)
            .setGroup(SUMMARY_GROUP_NAME)
            .setAutoCancel(true)
            .setContentIntent(pendingIntent);
    }
    return null;
}
```

### 5.3.4 Drawer

La aplicación cuenta con un Drawer, es la sección que se encuentra en el lado izquierdo de la aplicación, donde el usuario puede navegar entre las distintas pantallas que cuenta la aplicación, desde aquí se podrá dirigir a las vistas de los productos de nevera, productos de congelador, listas de compra y ajustes. También se puede encontrar con la funcionalidad situada abajo a la izquierda de desloguearse del el usuario actual, esta opción devolverá al login dando la posibilidad al usuario de logearse con otro perfil o registrarse.

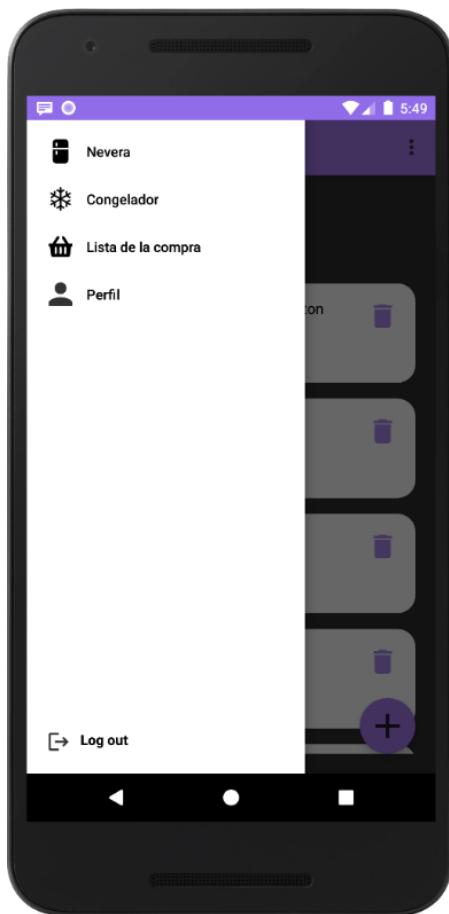


Figura 38 Drawer

El oyente del Drawer es utilizado desde el los activitis principales, este oyente es el que se encarga de cambiar las diferentes vistas dependiendo a donde se dirija el usuario:

```
//Este oyente es el del drawer
public class OyenteNav implements NavigationView.OnNavigationItemSelectedListener
{
    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        //Obtenemos la posicion del menu
        Activity activity = null;
        //Se usa un switch para hacer mas eficiente el codigo
        switch (item.getItemId()) {
```

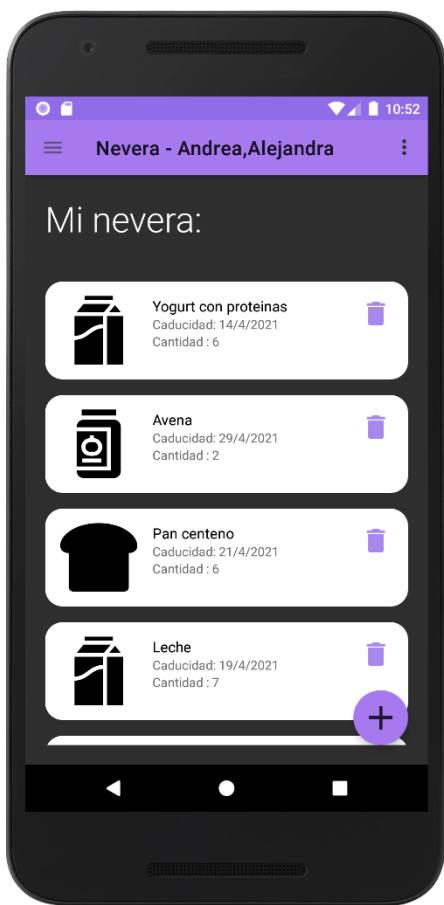
```
case R.id.menu_nevera:  
    activity = new NeveraActivity();  
    cambiarActivity(activity);  
    break;  
case R.id.menu_congelador:  
    activity = new CongeladorActivity();  
    cambiarActivity(activity);  
    break;  
case R.id.menu_lista:  
    activity = new TabActivity();  
    cambiarActivity(activity);  
    break;  
case R.id.menu_ajustes:  
    activity = new AjustesActivity();  
    cambiarActivity(activity);  
    break;  
}  
}  
return true;  
}  
}
```

### 5.3.5 Listas de productos

Los productos de cada usuario están organizados en listas. Dependiendo de su ubicación ya sea nevera o congelador, se visualizarán en una vista u otra. Estas vistas se llevan a cabo implementando un “Recycler View”, el cual se encarga de organizar los productos en forma de lista con una orientación vertical.

```
<androidx.recyclerview.widget.RecyclerView  
    android:id="@+id/item_product_nevera"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:layout_margin="20dp"  
    android:layoutAnimation="@anim/layout_caer"  
    android:scrollbars="vertical" />
```

En estas listas, como se puede ver en la Figura 40, se mostrará los atributos de cada producto, cuenta con su nombre, fecha de caducidad y la cantidad. Se ha decidido que la manera más óptima de mostrar el tipo de producto es a través de una imagen diferente para cada tipo distinto, como se ve en la Figura 39 Infografía de tipos de productos.

*Figura 40 Lista de nevera**Figura 39 Infografía de tipos de productos*

A la derecha de cada producto se observa un ícono de papelera el cual eliminará el producto.

En la parte inferior derecha de la pantalla se tiene un botón que añade nuevos productos, este botón es flotante y al bajar en la lista puede aparecer y desaparecer para darle fluidez a la aplicación, se verá con detenimiento más adelante.

En la parte superior derecha, situado en la toolbar, se cuenta con un botón que nos dará más opciones a la hora de ordenar los productos o buscar un producto en concreto, como se puede visualizar en la Figura 41.

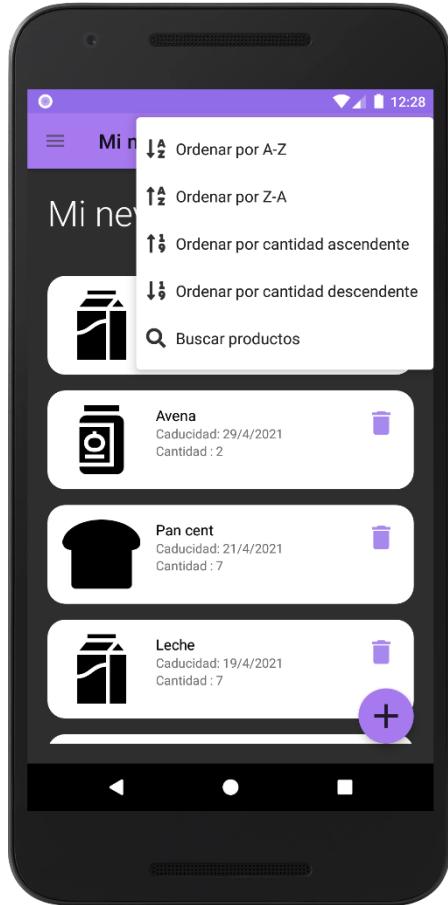


Figura 41 Menú funciones

Estas funciones son administradas por el modelo del producto, donde se realiza un compareTo para organizar los productos de diferentes maneras, un ejemplo de esto sería el compareTo de la A-Z, como se puede ver a continuación:

```
public static Comparator<ProductoModelo> ProductoAZ = new
Comparator<ProductoModelo>() {
    @Override
    public int compare(ProductoModelo p1, ProductoModelo p2) {
        return p1.getNombre().compareToIgnoreCase(p2.getNombre());
    }
};
```

La función buscar se llama desde el Adapter Producto, es la clase que ayuda a visualizar y tratar los eventos del item, en este se tiene un método que filtra los productos dependiendo de lo que se esté buscando el usuario.

```
public void filterList(ArrayList<ProductoModelo> filteredList) {
    list = filteredList;
    notifyDataSetChanged();
}
```

La opción de buscar pondrá un buscador en la parte superior derecha en el cual se podrán hacer las búsquedas por nombre del producto que deseé. Se puede ver un ejemplo de ello en la Figura 42.

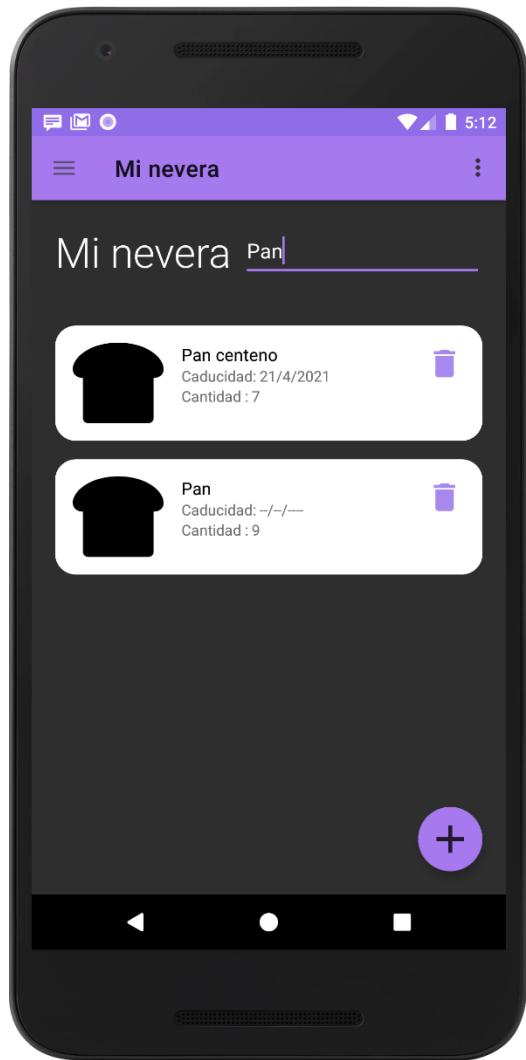


Figura 42 Buscador

En caso de que el producto no tenga precio o fecha de caducidad al guárdalo se visualizara como en la Figura 43.

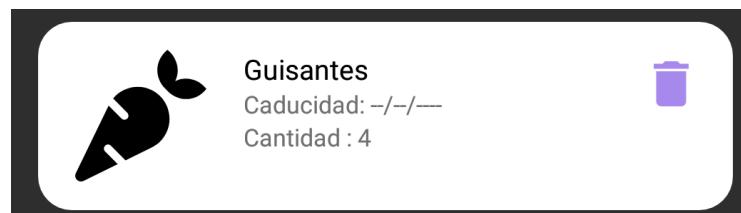


Figura 43 Item sin caducidad

Al momento de eliminar un producto saltará un pop-up, el cual le preguntará al usuario si está seguro de querer eliminar ese alimento y si desea añadirlo a la lista de compra. En caso de querer añadirlo a la lista de compra saldrá otro pop-up donde le preguntará al usuario que cantidad desea añadir.

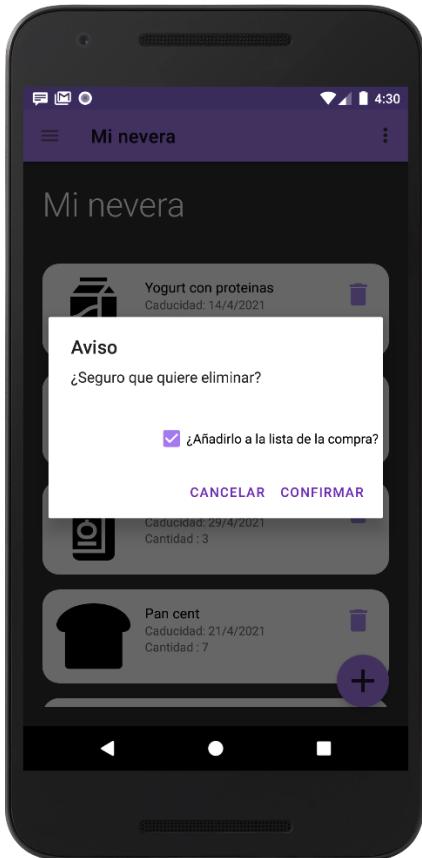


Figura 45 Pop-up Eliminar

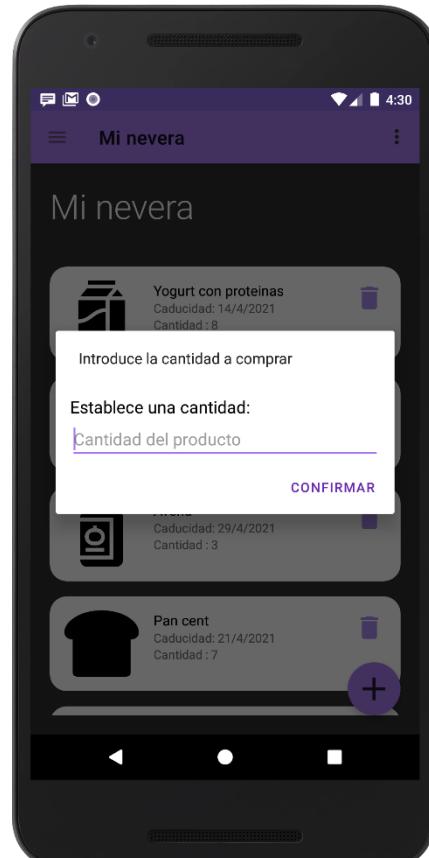


Figura 44 Pop-up Añadir cantidad

Estos pop-up son llamados desde los métodos de eliminar el producto y el código será explicado a continuación:

```
//Creamos el primer alertDialog
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setMessage("¿Seguro que quiere eliminar?")//Es el título
        .setTitle("Aviso")//Es el cuerpo
        .setView(checkBoxView) //Ponemos el checkbox
        ..setCancelable(false)//Con esta opción no se puede salir aunque pulse
        atras
        .setPositiveButton("Confirmar", new DialogInterface.OnClickListener() {
//En caso de que de a confirmar
    public void onClick(DialogInterface builder, int id) {
        //Comprobamos si existe el producto
        if(lista_productos.size()==1){
            lista_productos.clear(); //La limpiamos
        }
        //Oyente del check, si esta checkeado se creará otro alertDialog
    }
})
```

```

        if (checkBox.isChecked()) {
            AlertDialog.Builder builder2 = new
AlertDialog.Builder(CongeladorActivity.this);
            builder2.setMessage("Introduce la cantidad a comprar")
                .setView(edit_cantidad)
                .setCancelable(false)
                .setPositiveButton("Confirmar", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int
which) {
                        //Cambiamos la cantidad
                        try {
                            m DataBase.child("Producto").child(productoModelo.getId()).child("cantidad").setValue(Integer.parseInt(cantidad.getText().toString().trim()));

                            //Si esta seleccionado el check, lo
                            cambiamos de ubicacion

                            m DataBase.child("Producto").child(productoModelo.getId()).child("ubicacion").setValue("lista_congelador").addOnCompleteListener(new OnCompleteListener<Void>() {
                                @Override
                                public void onComplete(@NonNull
Task<Void> task) {
                                    if(task.isSuccessful()) {

                                        Toast.makeText(CongeladorActivity.this, "Se ha añadido satisfactoriamente",
                                        Toast.LENGTH_SHORT).show();
                                    }
                                }
                            });
                        } catch (NumberFormatException e){
                            Toast.makeText(getApplicationContext(),
                            "Debe introducir una cantidad", Toast.LENGTH_LONG).show();
                        }
                    }
                }).show();

        } else { //Si no esta seleccionado el check

            m DataBase.child("Producto").child(productoModelo.getId()).removeValue().addOnCompl
eteListener(new OnCompleteListener<Void>() {
                @Override
                public void onComplete(@NonNull Task<Void> task) {
                    if(task.isSuccessful()){
                        //Creamos un toast, para informar de que se ha
                        eliminado
                        Toast.makeText(CongeladorActivity.this, "Se ha
                        eliminado satisfactoriamente", Toast.LENGTH_SHORT).show();
                    }
                }
            });
        }
    });
builder.setNegativeButton("Cancelar", null).show();

```

Si el usuario desea editar un producto podrá realizarlo haciendo click sobre el ítem que deseé modificar y obtendrá la vista para editarlo, se puede ver esta vista en la Figura 46, aquí estarán predefinidos los campos con los valores anteriores del producto para que la edición sea más sencilla.

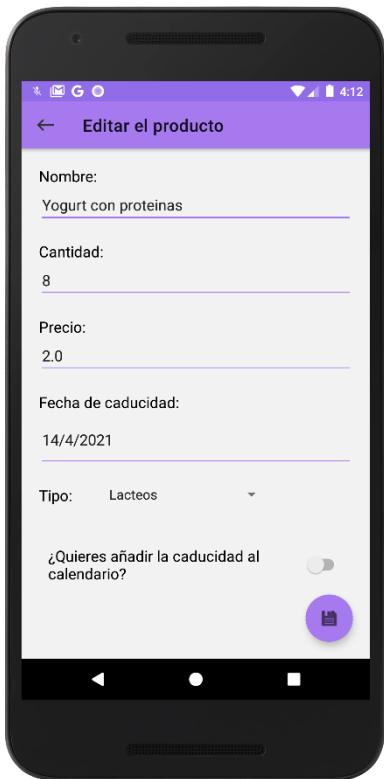


Figura 46 Editar productos



Figura 47 Lista sin productos

Existe la posibilidad de que un usuario no tenga productos en sus listas, en este caso lo que se visualizará es, un ícono con un amable mensaje, como se puede ver en la Figura 47.

### 5.3.6 Listas de la compra

Para visualizar los productos que se han terminado y han sido agregados a la lista de la compra se tendrá un activity con funcionalidad de tabs que le permitirá al usuario navegar entre las listas de la compra para nevera o congelador.

Dentro de estas vistas el usuario podrá seleccionar los productos que haya comprado a través de un check, los productos que estén checkeados al pulsar el botón de comprar se pasaran automáticamente a su vista de nevera o congelador, también podrá eliminar productos que considere que no sean más necesarios para su compra.

Los tabs funcionan debido a que el adapter controla los eventos del cambio de los fragments en la vista dependiendo de la posición.

```
@Override  
public Fragment getItem(int position) {  
  
    switch (position) {  
        case 0:  
            return nevera;  
        case 1:  
            return congelador;  
        default:  
            return null;  
    }  
}
```

En la esquina inferior derecha tenemos un botón para añadir nuevos productos a la lista de compra que el usuario pueda necesitar.

En los tres puntos que se encuentran en la parte superior derecha se puede ver diferentes opciones como se tiene en el activiy de nevera o congelador que tiene la capacidad de organizar los productos dependiendo del precio o la letra inicial, con la diferencia de aquí tenemos la opción de comprar todos y en vez de ordenar por cantidad se ha considerado más útil ordenar por precio. Se puede ver el menú desplegable en la Figura 49.

En el lado derecho de los ítems se puede ver un check con el cual el usuario puede seleccionar que productos que desea comprar, así, en el momento en el que el usuario le dé al botón comprar, solo se añadirán a su despensa los productos checkeados. Se puede ver la lista en acción en la Figura 48.

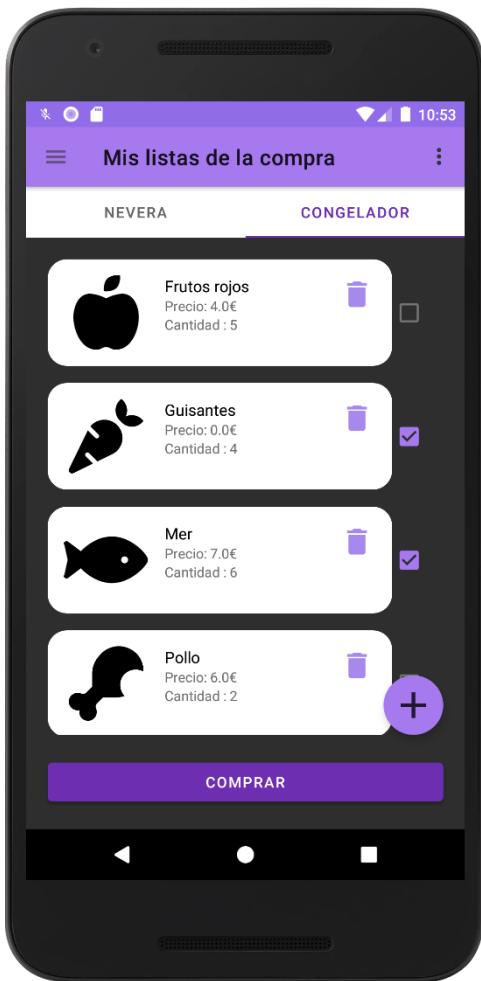


Figura 48 Listas de la compra

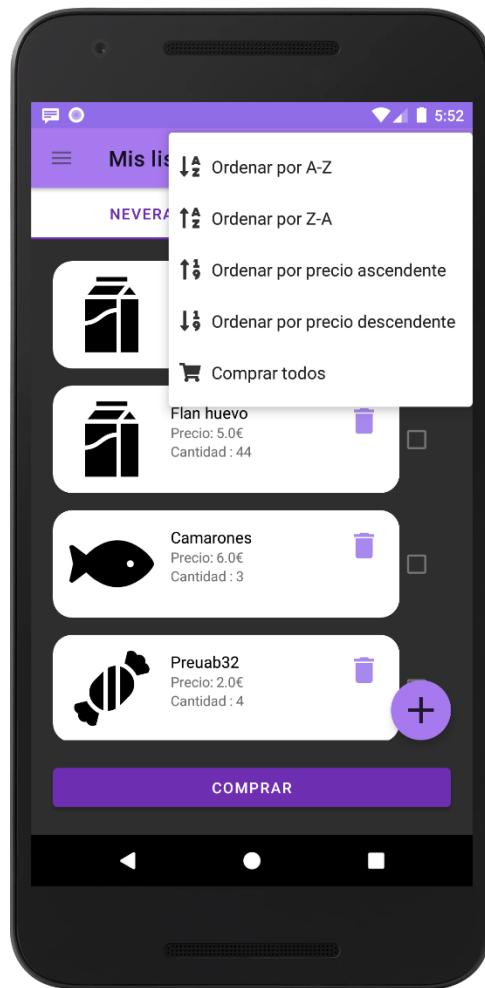


Figura 49 Menú desplegable

En caso de añadir un producto nuevo (tanto en las listas de la compra, como en la nevera o congelador) se tiene una vista en la cual se puede ingresar los datos del producto que se desea añadir. Los productos tienen nombre, cantidad, precio, fecha de caducidad y tipo, este último se selecciona con un desplegable. Se visualiza en la Figura 50.

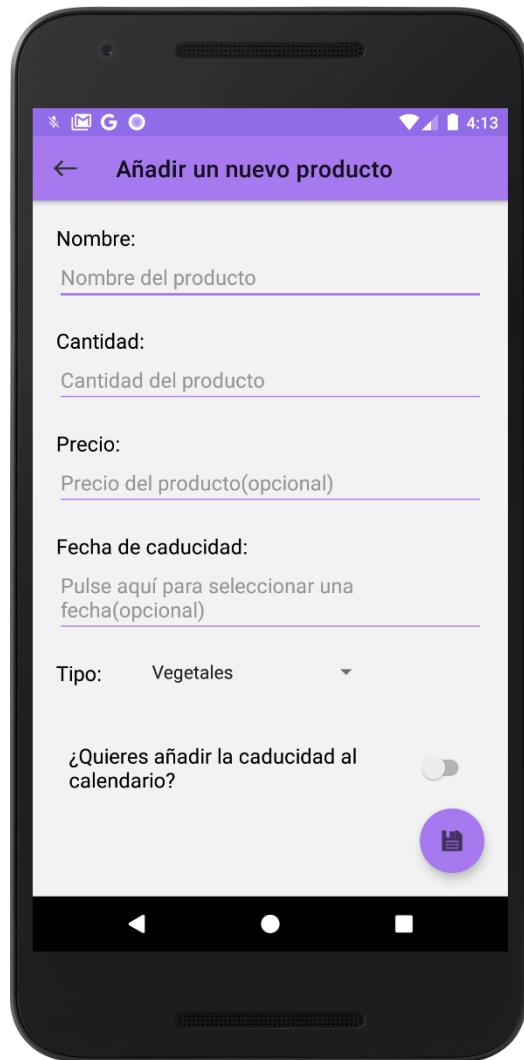


Figura 50 Añadir un nuevo producto

El usuario también tiene la opción de seleccionar si desea crear una notificación en su calendario para la fecha de caducidad de dicho alimento. Esta funcionalidad se logra al poder conectar la aplicación con Google calendar, que genera el evento con la fecha de caducidad del producto.

```
public void CrearEvento(){
    //Creamos la instancia del calendario, para poder establecer la fecha del
    evento
    Calendar cal = Calendar.getInstance();
    Intent intent = null;
    try{
        //Obtenemos la fecha
        String[] fecha = calendario.getText().toString().split("/");
        //Establecemos la fecha
        cal.set(Calendar.DAY_OF_MONTH, Integer.parseInt(fecha[0]));
        cal.set(Calendar.MONTH, Integer.parseInt(fecha[1])-1);
        cal.set(Calendar.YEAR, Integer.parseInt(fecha[2]));
        intent = new Intent(Intent.ACTION_EDIT);
        intent.setType("vnd.android.cursor.item/event");
        intent.putExtra("title", nombre);
        intent.putExtra("description", "Expiración de " + nombre);
        intent.putExtra("date", DateUtil.dateToString(cal.getTimeInMillis()));
        intent.putExtra("calendar_id", 1);
        intent.putExtra("account_name", "Google");
        intent.putExtra("event_color", "#0000FF");
        intent.putExtra("has_end_time", false);
        intent.putExtra("has_recurring", false);
        intent.putExtra("location", "");
        intent.putExtra("start_time", DateUtil.dateToString(cal.getTimeInMillis()));
        intent.putExtra("synced", true);
        intent.putExtra("title", nombre);
        intent.putExtra("url", "http://www.google.com/calendar");
        intent.putExtra("vcal", "BEGIN:VCALENDAR\nVERSION:2.0\nPRODID:-//Google Inc//GCal//EN\nMETHOD:PUBLISH\nCALSCALE:GREGORIAN\nBEGIN:VEVENT\nDTSTART;TZID=UTC:" + DateUtil.dateToString(cal.getTimeInMillis()) + "\nDTEND;TZID=UTC:" + DateUtil.dateToString(cal.getTimeInMillis()) + "\nSUMMARY:Expiración de " + nombre + "\nDESCRIPTION:Expiración de " + nombre + "\nLOCATION:\nURL: http://www.google.com/calendar\nEND:VEVENT\nEND:VCALENDAR");
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
    } catch (Exception e){
        e.printStackTrace();
    }
}
```

```
        cal.set(Calendar.YEAR, Integer.parseInt(fecha[2]));
        cal.set(Calendar.MONTH, Integer.parseInt(fecha[1]) - 1); //Es menos 1,
debido a que enero es 0
        cal.set(Calendar.DAY_OF_MONTH, Integer.parseInt(fecha[0]));

        //Creamos el evento
        intent = new Intent(Intent.ACTION_INSERT);
        intent.setData(CalendarContract.Events.CONTENT_URI);
        intent.putExtra(CalendarContract.Events.ALL_DAY,true);
        intent.putExtra(CalendarContract.Events.TITLE,"El producto " +
nombre.getText() + " se caduca pronto");
        intent.putExtra(CalendarContract.Events.DESCRIPTION, "Comer antes de
que se ponga malo.");
        intent.putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME,
cal.getTimeInMillis()); //Establecemos el día del evento

        //Comprobamos que tenga aplicación para ello
        if(intent.resolveActivity(getApplicationContext()) != null)
            startActivity(intent);
        else
            Toast.makeText(getApplicationContext(), "No hay aplicacion para
crear un evento de calendario", Toast.LENGTH_LONG).show();
    }catch (Exception e){
        Toast.makeText(getApplicationContext(), "Fecha invalida",
Toast.LENGTH_LONG).show();
    }
}
```

El evento como se ha comentado se genera en el calendario, se abrirá la pestaña y si el usuario lo desea podrá cambiar los parámetros, pero se ha decidido que para mayor comodidad con esta funcionalidad se auto rellenarán los datos para el título, la fecha y la descripción, y el evento tendrá un atributo de todo el día. En la Figura 51 se muestra un ejemplo de creación del evento.

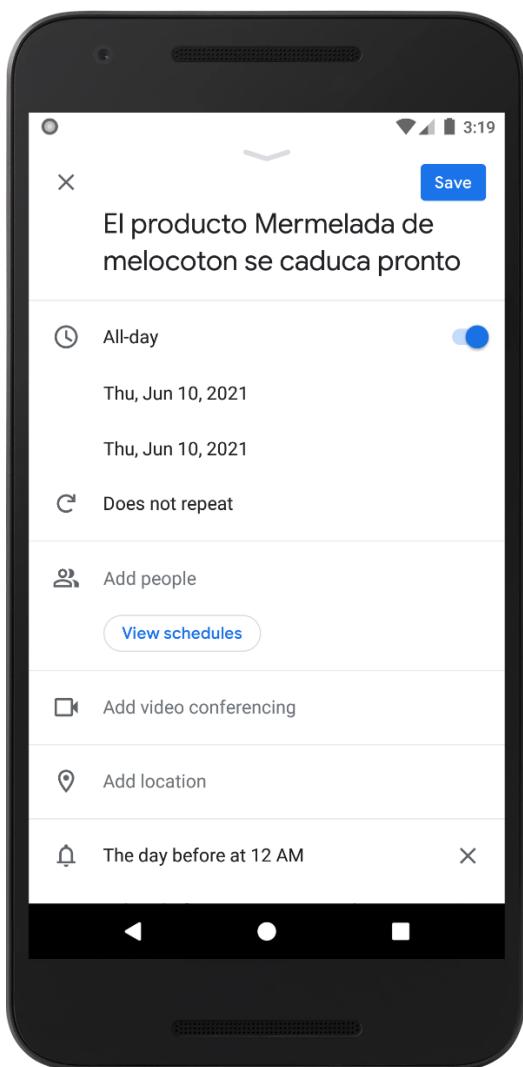
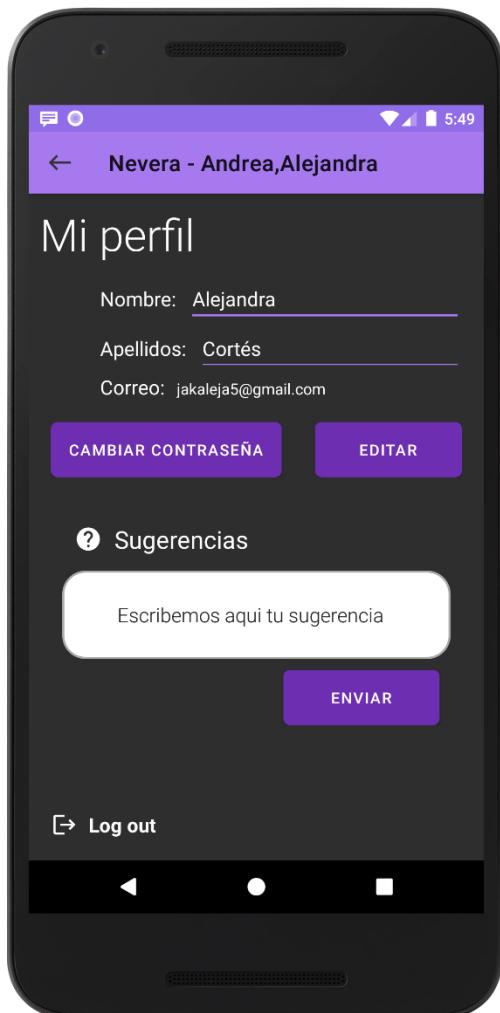


Figura 51 Creación de evento en el calendario

En caso de que el usuario no tenga una aplicación disponible para crear un evento de calendario, saldrá un mensaje avisándolo y continuará en la aplicación.

### 5.3.7 Perfil de usuario

En esta vista el usuario puede visualizar sus datos personales y tiene la opción de poder editar su nombre, apellido y contraseña de su perfil además puede cerrar sesión desde esta vista *Figura 52*.



*Figura 52 Perfil de usuario*

En el apartado de sugerencias, el usuario tiene la opción de escribir una pequeña reseña del uso de la aplicación para mandar por correo electrónico, esto será posible a través del método enviarEmail:

```
private void enviarEmail(){
    //Instanciamos un Intent del tipo ACTION_SEND
    Intent emailIntent = new Intent(android.content.Intent.ACTION_SEND);
    //Aqui definimos la tipología de datos del contenido dle Email en este caso
text/html
    emailIntent.setType("text/html");

    // Indicamos con un Array de tipo String las direcciones de correo a las
cuales enviar
    emailIntent.putExtra(Intent.EXTRA_EMAIL, new String[]{"jakaleja5@gmail.com"});
    // Aqui definimos un titulo para el Email
```

```

emailIntent.putExtra(android.content.Intent.EXTRA_TITLE, "Sugerencias");
// Aqui definimos un Asunto para el Email
emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT, "Sugerencia de App
Mi Nevera");
// Aqui obtenemos la referencia al texto y lo pasamos al Email Intent

emailIntent.putExtra(android.content.Intent.EXTRA_TEXT, textSugerencia.getText().to
String());
try {
    //Enviamos el Correo iniciando una nueva Activity con el emailIntent.
    startActivity(Intent.createChooser(emailIntent, "Enviar Correo..."));
} catch (android.content.ActivityNotFoundException ex) {
    Toast.makeText(AjustesActivity.this, "No hay ningun cliente de correo
instalado.", Toast.LENGTH_SHORT).show();
}
}

```

### 5.3.8 Responsive

Los layouts han sido creados con ConstraintLayout para facilitar que se adapte a todas las pantallas, sin importar sus dimensiones. A continuación se muestra un ejemplo de ejecución en Tablet horizontal, en la Figura 53.

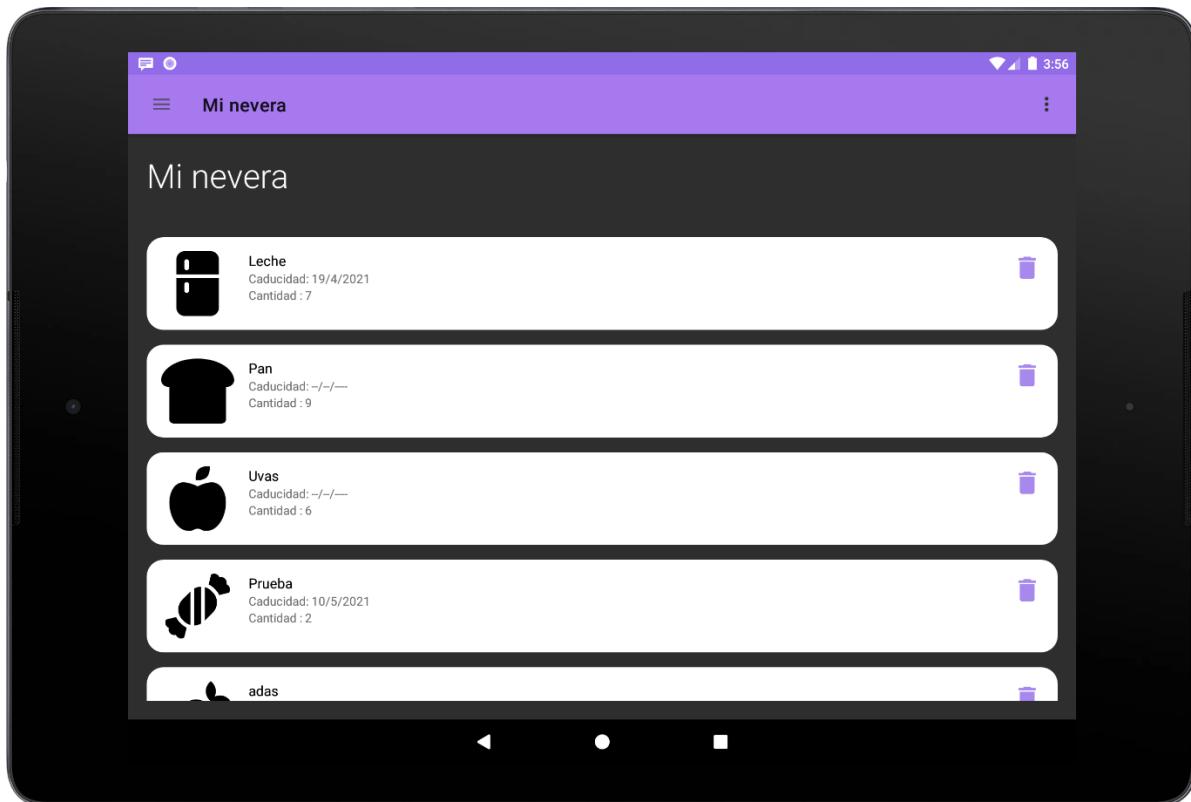


Figura 53 Ejemplo de ejecución en tablet

Se han utilizado herramientas como las guidelines, para establecer porcentajes de pantalla y así crear verdaderamente una interfaz responsive



## Capítulo 6. Conclusiones

En este capítulo se expondrán las conclusiones llegadas tras el desarrollo de la aplicación.

### Sección 6.1 Conclusiones del proyecto

Cumpliendo con los objetivos parciales propuestos en el Capítulo 2, se ha logrado desarrollar una aplicación simple de usar que les facilite a los usuarios el control de la cantidad de alimentos que tiene en su nevera/congelador sin tener que estar en casa.

A continuación, el desarrollo de cada objetivo con su debida justificación.

*Tabla 2 Desarrollo de objetivos*

Objetivo	Justificación	¿Éxito?
<b>OP1.</b> Análisis de la idea y de los requisitos que se necesiten para el cumplimiento total de la aplicación.	Se elaboró un estudio en el Capítulo 1 en el cual, se demostró la cantidad de comida que se desperdicia en España y las causas de esta.	Si
<b>OP2.</b> Comparativa con las demás aplicaciones del mercado e identificar posibles fallas en ellos para la realización de mejoras.	En el Capítulo 1 se realizó un estudio de mercado en el cual no se encontró ninguna aplicación que diera una solución efectiva al problema planteado.	Si
<b>OP3.</b> Creación de un diseño visual, atractivo e intuitivo para los clientes.	Se desarrollaron diferentes vistas interactivas para la comodidad del usuario. Explicadas en el Capítulo 5.	Si
<b>OP4.</b> Realización de la base de datos, usada para almacenar los datos necesarios en la aplicación.	Se realizó un diagrama Entidad-Relación que ayuda a comprender los diferentes datos de la aplicación.	Si
<b>OP5.</b> Aprendizaje de Java para su utilización en Android Studio.	Se desarrollaron diferentes prácticas durante el grado que ayudaron a la puesta en práctica de este TFG.	Si
<b>OP6.</b> Desarrollo de la aplicación cumpliendo con los objetivos anteriores.	Se realizó cumpliendo con la metodología de trabajo planteado en el Capítulo 4 y se desarrolló como se puede evidenciar en el Capítulo 5.	Si
<b>OP7.</b> Lanzamiento de la aplicación a los clientes cumpliendo con todos los requisitos.		

### Sección 6.2 Líneas futuras

En esta sección se explicarán algunas posibles mejoras que podrá tener la aplicación al tener una posible comercialización, debido a que este TFG se ha planteado con la finalidad de crear listas para el control de alimentos se han dejado algunos temas sin tocar que no están relacionados directamente con el control alimenticio.

**Iconos indicadores de caducidad:** Esto dará un aspecto más visual al usuario para distinguir que producto está cerca de caducar, ya que en una nevera con muchos productos es posible no reconocerlo a tiempo. Una previsualización de cómo podría quedar esto se ve en la Figura 54.

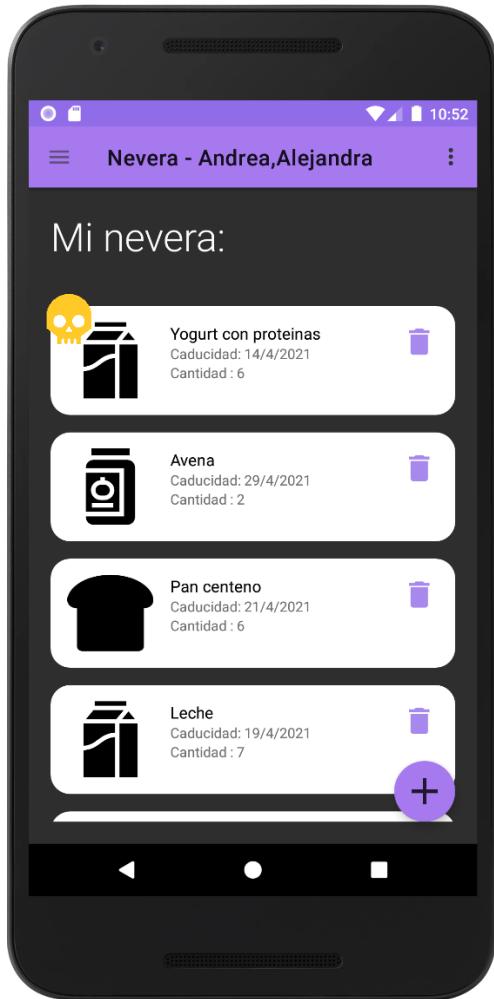
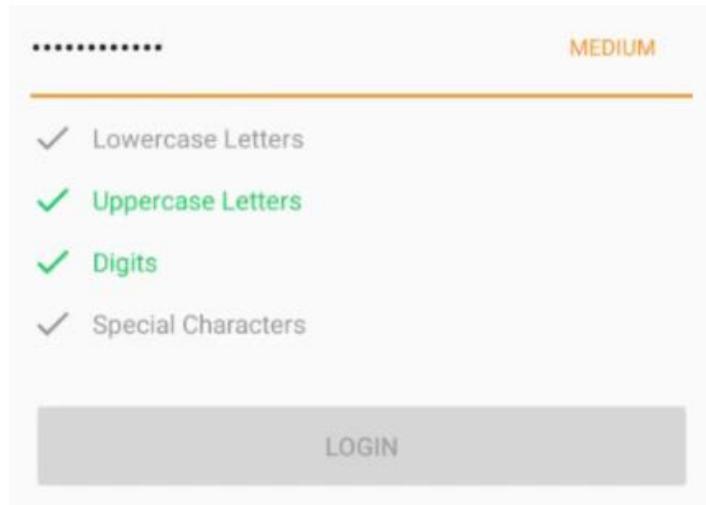


Figura 54 Visualización de producto cercano a caducidad

**Recetas:** Muchas veces los usuarios suelen tener alimentos en la nevera, pero no saben bien cómo combinarlos o en qué receta usarlos, para esto se propone incluir un apartado de recetas donde con los alimentos que se tenga en la nevera se busquen diferentes recetas para mostrarle al usuario.

**Valoración en el Google Play:** En el momento en que el usuario envié una sugerencia pueda valorar la aplicación mediante estrellas y esta sea visible en el Google Play para darle una referencia a los demás usuarios de la calidad de la app.

**Ampliación del login:** En un futuro, se plantearía añadir distintos inicios de sesión por medios como google, Facebook o twitter, para dar más facilidades al usuario y así integrarlo más con sus cuentas habituales. También se debería tener en cuenta la implementación de un método para calcular la fuerza de la contraseña, para darle más seguridad al usuario, se muestra una representación en la Figura 55.



*Figura 55 Representación de fuerza de contraseña*



## Capítulo 7. Bibliografía

- [1] G. d. España, «El desperdicio alimentario en los hogares españoles aumentó un 8,9% en 2018,» [En línea]. Available: <https://www.mapa.gob.es/es/prensa/ultimas-noticias/-el-desperdicio-alimentario-en-los-hogares-espa%C3%B1oles-aument%C3%B3-un-89-en-2018/tcm:30-510668>.
- [2] statcounter, «Mobile Operating System Market Share Worldwide,» 2020. [En línea]. Available: <https://gs.statcounter.com/os-market-share/mobile/worldwide>.
- [3] statcounter, «Tablet Operating System Market Share Worldwide,» 2020. [En línea]. Available: <https://gs.statcounter.com/os-market-share/tablet/worldwide>.
- [4] Wikipedia, «Mobile app,» [En línea]. Available: [https://en.wikipedia.org/wiki/Mobile\\_app](https://en.wikipedia.org/wiki/Mobile_app).
- [5] Wikipedia, «History of mobile phones,» [En línea]. Available: [https://en.wikipedia.org/wiki/History\\_of\\_mobile\\_phones](https://en.wikipedia.org/wiki/History_of_mobile_phones).
- [6] Wikipedia, «IBM Simon,» [En línea]. Available: [https://en.wikipedia.org/wiki/IBM\\_Simon](https://en.wikipedia.org/wiki/IBM_Simon).
- [7] Wikipedia, «Tetris,» [En línea]. Available: <https://en.wikipedia.org/wiki/Tetris>.
- [8] Wikipedia, «Snake (video game genre),» [En línea]. Available: [https://en.wikipedia.org/wiki/Snake\\_\(video\\_game\\_genre\)](https://en.wikipedia.org/wiki/Snake_(video_game_genre)).
- [9] GoMage, «PWA vs Native vs Hybrid vs Responsive Website: Full Comparison,» [En línea]. Available: <https://www.gomage.com/blog/pwa-vs-native-vs-hybrid-vs-responsive-website/>.
- [10] A. Casans, «Las mejores herramientas de desarrollo de apps móviles,» 4 Julio 2020. [En línea]. Available: <https://www.hiberus.com/crecemos-contigo/mejores-herramientas-de-desarrollo-de-apps-movil/>.
- [11] comscore, «Global State of Mobile,» 2019. [En línea]. Available: [https://www.amic.media/media/files/file\\_352\\_2199.pdf](https://www.amic.media/media/files/file_352_2199.pdf).

- [12] Google Chrome Developers, «When should you use a PWA? - Progressive Web App Training,» 11 Marzo 2019. [En línea]. Available: <https://www.youtube.com/watch?app=desktop&v=DfFIBWCQjzA&ucbcb=1>.
- [13] Appnet Blog, «Android Studio: El entorno de desarrollo oficial de Android,» [En línea]. Available: <https://www.tu-app.net/blog/android-studio/>.
- [14] M. Ashraf, «Best IDEs for Mobile App Development | Our Top 5 Picks,» [En línea]. Available: <https://www.appverticals.com/blog/best-integrated-development-environments/>.
- [15] Wikipedia, «Xcode,» [En línea]. Available: <https://es.wikipedia.org/wiki/Xcode>.
- [16] Apple Developer, «Xcode IDE features,» [En línea]. Available: <https://developer.apple.com/xcode/features/>.
- [17] okdiario, «¿Qué es Xamarin?,» [En línea]. Available: <https://okdiario.com/tecnologia/que-xamarin-2022974>.
- [18] J. S. Perry, «Conceptos básicos del lenguaje Java,» 3 Diciembre 2012. [En línea]. Available: <https://developer.ibm.com/es/languages/java/tutorials/j-introtojava1/>.
- [19] «Case Studies (Kotlin),» [En línea]. Available: <https://kotlinlang.org/jp/mobile/case-studies/>.
- [20] «Swift,» [En línea]. Available: <https://www.apple.com/es/swift/>.
- [21] Oracle, «Definición de base de datos,» [En línea]. Available: <https://www.oracle.com/mx/database/what-is-database/>.
- [22] A. I. Sordo, «Qué es una base de datos y cuáles son los 3 tipos utilizados por empresas,» 25 Enero 2021. [En línea]. Available: <https://blog.hubspot.es/marketing/tipos-base-datos>.
- [23] Ionos, «Bases de datos relacionales: el modelo de datos en detalle,» 9 Mayo 2019. [En línea]. Available: <https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/bases-de-datos-relacionales/>.

- [24] «¿Qué es DB2 IBM?,» [En línea]. Available: <https://www.comparasoftware.com/db2-ibm>.
- [25] Ionos, «NoSQL: la tendencia hacia un soporte de datos estructurado,» 20 Abril 2020. [En línea]. Available: <https://www.ionos.mx/digitalguide/hosting/cuestiones-tecnicas/nosql/>.
- [26] Ionos, «Base de datos columnar,» 23 Marzo 2020. [En línea]. Available: <https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/base-de-datos-columnar/>.
- [27] Wikipedia, «Modelo–vista–controlador,» [En línea]. Available: <https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador>.
- [28] Mike Minutillo, «What are MVP and MVC and what is the difference?,» [En línea]. Available: <https://stackoverflow.com/questions/2056/what-are-mvp-and-mvc-and-what-is-the-difference>.
- [29] S. Smith, «¿Qué es el patrón de MVC?,» 12 Febrero 2020. [En línea]. Available: <https://docs.microsoft.com/es-es/aspnet/core/mvc/overview?view=aspnetcore-5.0>.
- [30] iKenshu, «Arquitecturas de Software en Android: MVC, MVP y MVVM,» [En línea]. Available: <https://platzi.com/blog/arquitecturas-de-software-en-android-mvc-mvp-y-mvvm/>.
- [31] Wikipedia, «Modelo–vista–presentador,» [En línea]. Available: <https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93presentador>.
- [32] Wikipedia, «Model–view–viewmodel,» [En línea]. Available: <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93.viewmodel>.
- [33] R. Rai, «10 Best Grocery Shopping List Apps in 2021 (Android/iOS),» [En línea]. Available: <https://gadgetliv.com/best-grocery-list-apps/>.
- [34] «Out of Milk,» [En línea]. Available: <https://www.outofmilk.com/>.
- [35] «Bring!,» [En línea]. Available: <https://play.google.com/store/apps/details?id=ch.publisheria.bring&hl=es&gl=US>.

- [36] «Listonic,» [En línea]. Available: <https://listonic.com/es/>.
- [37] kanbanize, «Qué es un tablero Kanban: Fundamentos,» [En línea]. Available: <https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-tablero-kanban>.
- [38] kanbanize, «Qué es Kanban: Definición, Características y Ventajas,» [En línea]. Available: <https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban>.
- [39] R. APD, «¿En qué consiste la metodología Kanban y cómo utilizarla?,» 30 Enero 2019. [En línea]. Available: <https://www.apd.es/metodologia-kanban/>.
- [40] RyteWiki, «Modelo en Cascada,» [En línea]. Available: [https://es.ryte.com/wiki/Modelo\\_en\\_Cascada](https://es.ryte.com/wiki/Modelo_en_Cascada).
- [41] P. Domínguez, «En qué consiste el modelo en cascada,» 2 Junio 2020. [En línea]. Available: <https://openclassrooms.com/en/courses/4309151-gestion-a-tu-proyecto-de-desarrollo/4538221-en-que-consiste-el-modelo-en-cascada>.

## Anexo 1. Acrónimos

En este apartado se encontrará una lista de los acrónimos presentes en este proyecto:

- **TFG:** Trabajo de Fin de Grado
- **E/R:** Modelo Entidad-Relación
- **UML:** Unified Modeling Language
- **SDK:** Software Development Kit
- **API:** Application Programming Interface
- **SO:** Sistema Operativo
- **iOS:** iPhone Operating System
- **XML:** eXtensible Markup Language
- **IDE:** Integrated Development Environment
- **CRUD:** Create, Read, Update and Delete
- **NoSQL:** Non Structured Query Language
- **JSON:** JavaScrip Object Notation
- **SGBD:** Sistemas Gestores de Bases de Datos
- **DBMS:** Database Management System
- **HTML:** HyperText Markup Language
- **CSS:** Cascading Style Sheets
- **TSQL:** Transact-SQL
- **MVC:** Modelo-Vista-Controlador o Model-View-Controller
- **MVP:** Modelo–Vista–Presentador o Model View Presenter
- **MVVM:** Modelo–Vista–Modelo de vista o Model View ViewModel
- **OOP:** Object-Oriented Programming

## Anexo 2. Licencia



*Figura 56 Licencia CC*

© Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-Compartirlgual 4.0 Internacional, Figura 56.