

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

**Andrea Méndez Sanz**

**Leidy Alejandra Cortés González**

**Mario Marugán Cancio**

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

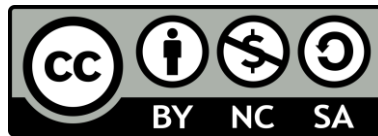
**Curso 2020-2021**

**Innovación en Formación Profesional (IFP)**

**Convocatoria de Presentación: junio 2021**

---

## Licencia



© Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional

---

## Índice de contenidos

Licencia .....	2
Índice de contenidos .....	3
Índice de figuras .....	5
Índice de tablas .....	7
Resumen .....	8
<b>Capítulo 1. Introducción .....</b>	<b>9</b>
<b>Capítulo 2. Objetivos .....</b>	<b>10</b>
Sección 2.1 Objetivo principal.....	10
Sección 2.2 Objetivos parciales.....	10
Sección 2.3 Medios utilizados .....	10
<b>Capítulo 3. Estado del Arte .....</b>	<b>12</b>
Sección 3.1 Evolución del desarrollo móvil.....	12
Sección 3.2 Entornos de desarrollo para aplicaciones móviles .....	15
Sección 3.3 Lenguajes de programación para aplicaciones móviles .....	18
Sección 3.4 Otros aspectos tecnológicos .....	18
Sección 3.5 Aplicaciones Similares.....	22
<b>Capítulo 4. Método del Trabajo .....</b>	<b>25</b>
Sección 4.1 Método de desarrollo en cascada .....	25
Sección 4.2 Planificación .....	26
<b>Capítulo 5. Resultados .....</b>	<b>29</b>
Sección 5.1 Modelo Entidad/Relación .....	29
Sección 5.2 Modelo de casos de uso .....	29
Sección 5.3 Vistas de la app .....	30
<b>Capítulo 6. Conclusiones .....</b>	<b>49</b>
Sección 6.1 Conclusiones del proyecto.....	49

---

Sección 6.2 Líneas futuras.....	50
<b>Capítulo 7. Bibliografía .....</b>	<b>52</b>
Acrónimos .....	56

---

## Índice de figuras

Figura 1 Teléfono móvil IBM Simon .....	12
Figura 2 Comparativa de PWA con nativa.....	14
Figura 3 Comparativa del uso web contra el uso de apps en móvil .....	15
Figura 4 Representación de los sensores.....	15
Figura 5 Entorno de desarrollo de Android Studio .....	16
Figura 6 Entorno de desarrollo de Xcode.....	17
Figura 7 Representación de las diversas bases de datos .....	20
Figura 8 Infografía del MVC .....	21
Figura 9 Esquema del MVC .....	21
Figura 10 Infografía del MVP .....	22
Figura 11 Vista de artículos (Out of milk) .....	23
Figura 12 Vista de añadir un artículo (Out of milk).....	23
Figura 13 Vista de recetas (Bring!) .....	23
Figura 14 Vista inicial de la aplicación (Bring!) .....	23
Figura 15 Vista de distintas listas (Listonic) .....	24
Figura 16 Vista de productos (Listonic).....	24
Figura 17 Infografía del modelo en cascada .....	25
Figura 18 Representación del modelo entidad/relación .....	29
Figura 19 Modelo de caso de uso .....	29
Figura 20 Splash .....	30
Figura 21 Vista Inicial .....	31
Figura 22 Registro de usuarios.....	31
Figura 23 Inicio de sesión.....	31
Figura 24 Problema de inicio de sesión .....	32
Figura 25 Canales de la aplicación .....	33

---

Figura 26 Ajustes de notificaciones chrome .....	33
Figura 27 Drawer.....	35
Figura 28 Infografía de tipos de productos.....	37
Figura 29 Lista de nevera .....	37
Figura 30 Menú funciones .....	38
Figura 31 Buscador.....	39
Figura 32 Item sin caducidad .....	39
Figura 33 Pop-up Añadir cantidad .....	40
Figura 34 Pop-up Eliminar.....	40
Figura 35 Editar productos.....	42
Figura 36 Lista sin productos .....	42
Figura 37 Listas de la compra.....	44
Figura 38 Menú desplegable.....	44
Figura 39 Añadir un nuevo producto .....	45
<i>Figura 40 Perfil de usuario .....</i>	<i>47</i>
Figura 41 Visualización de producto cercano a caducidad .....	50
Figura 42 Representación de fuerza de contraseña .....	51

---

## Índice de tablas

Tabla 1 Comparativa de las aplicaciones .....	14
Tabla 2 Desarrollo de objetivos .....	49

---

## 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.

Una vez se ha decidido que Android será el sistema operativo(SO) en el que se desarrollará, se determinó usar Android Studio como entorno de desarrollo por la familiaridad con el mismo, una gran comunidad y facilidad a la hora de usarlo. En ese momento se tuvo que elegir el lenguaje de programación, principalmente se plantearon dos opciones; Java y Kotlin. Se decantó por Java debido a sus características técnicas, de escalabilidad y de adaptación a frameworks.

---

## 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 Figura 1 Teléfono móvil IBM Simon), 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.

### Aplicaciones nativas vs PWA 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.

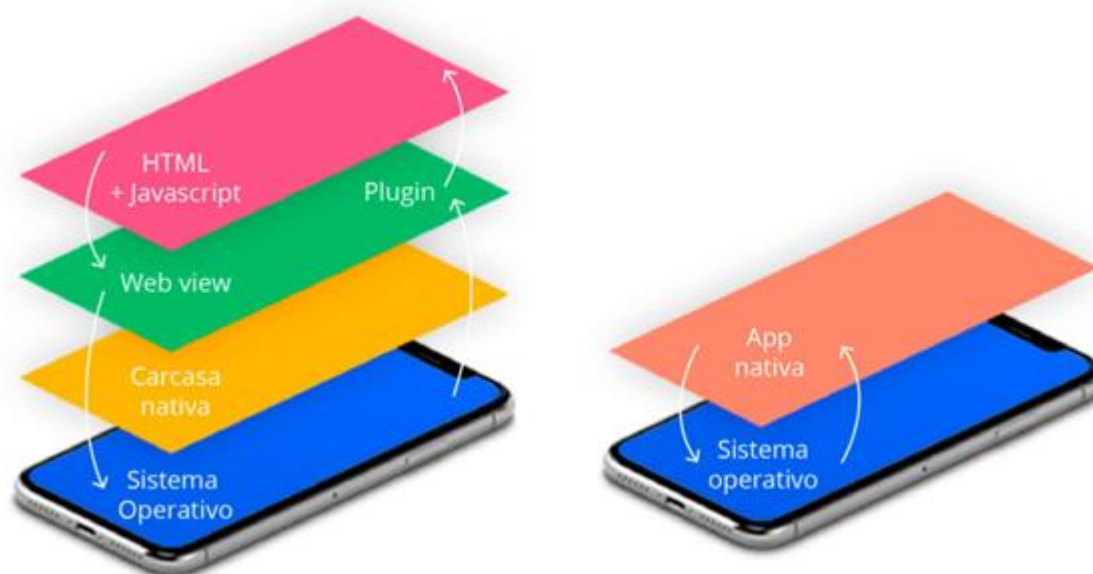
**Aplicaciones híbridas:** estas aplicaciones son, por definición, la combinación de aplicaciones nativas y basadas en web (PWA). Lo cual supone que para desarrollarlas se deban programar tanto en lenguajes webs como orientados a objetos.

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]

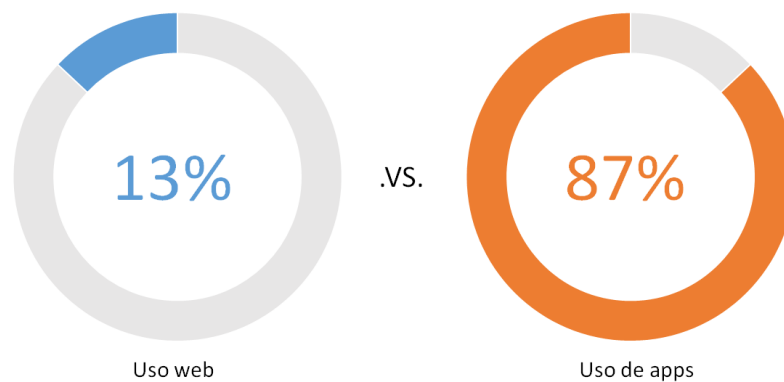
*Tabla 1 Comparativa de las aplicaciones*

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>	No		Sí



*Figura 2 Comparativa de PWA con nativa*

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 como usan los usuarios el teléfono móvil día a día, en España el 87% del tiempo de uso móvil (representado en Figura 3), 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 3 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 4), a los únicos que son capaces de acceder es al GPS y al micrófono.



*Figura 4 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

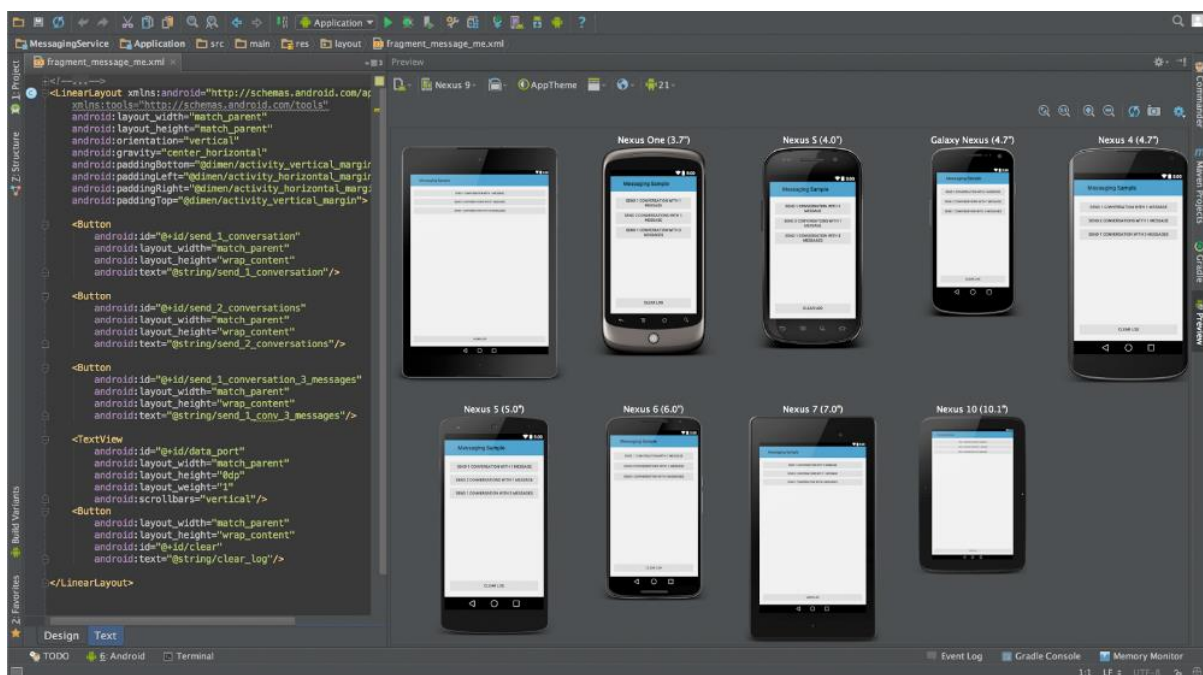
Un entorno de desarrollo es un software que provee todas las herramientas necesarias para crear una aplicación móvil, generalmente incluyen soporte para uno o varios lenguajes de programación, permitiendo compilar, depurar y realizar pruebas [10]. Algunos de los entornos más usados en la actualidad son:

**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 5 se muestra la UI de Android Studio.



*Figura 5 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 6 se muestra la UI de Xcode.

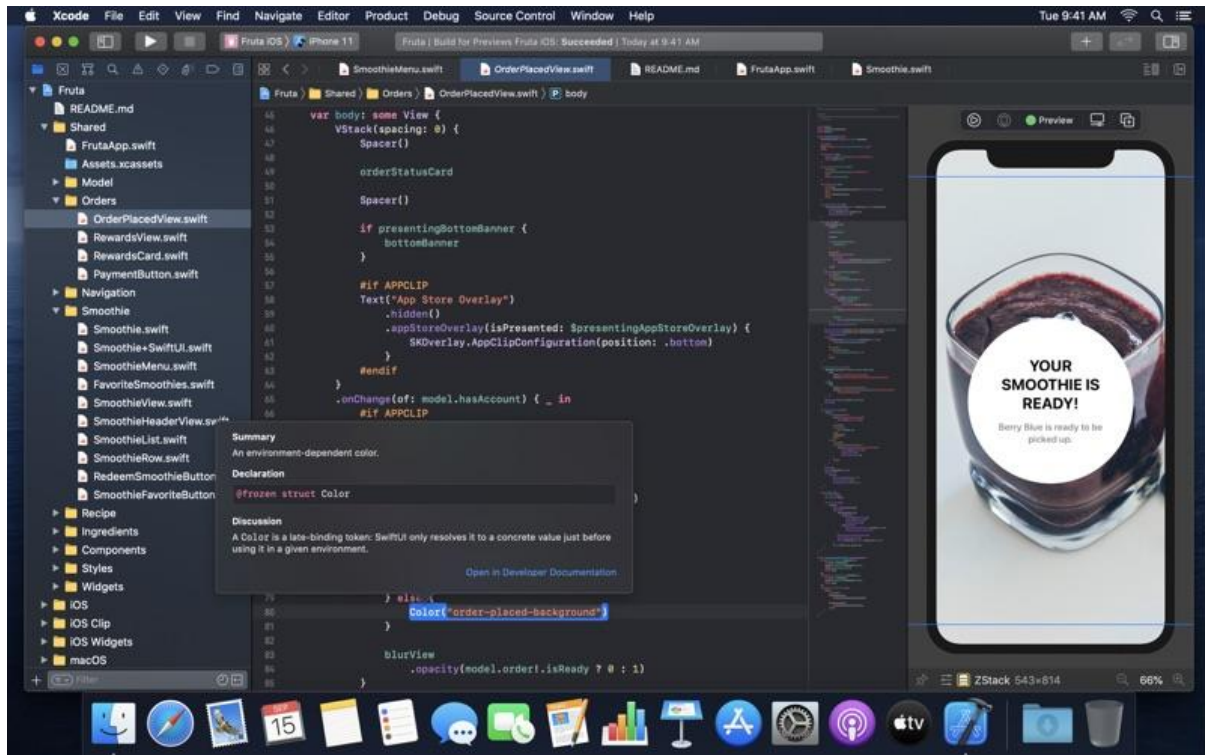


Figura 6 Entorno de desarrollo de Xcode

**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.

#### 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 unívoca 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].

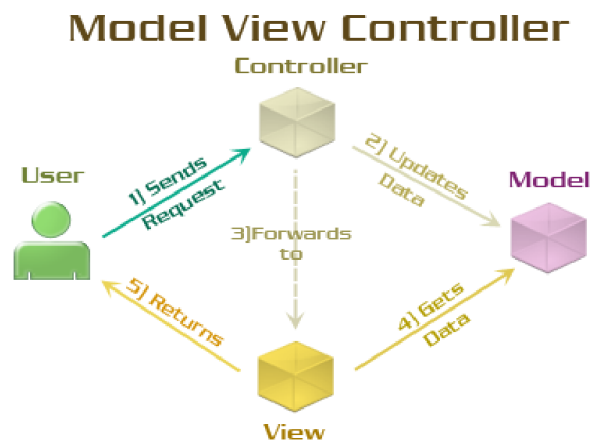
En la Figura 7 se muestra una infografía de los distintos tipos de BBDD.



*Figura 7 Representación de las diversas bases de datos*

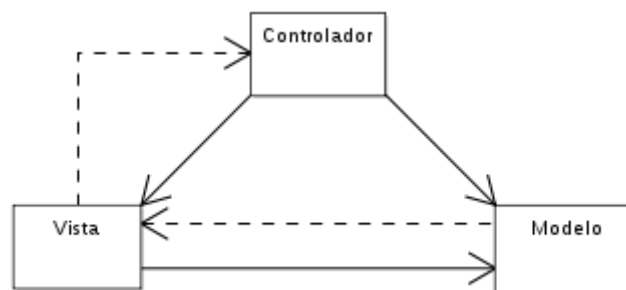
## Patrones de arquitectura

**MVC:** es el modelo-vista-controlador 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 8 . [27] [28] [29] [30]



*Figura 8 Infografía del MVC*

Como su propio nombre indica en MVC, tenemos tres componentes principales mostrados en la Figura 9:



*Figura 9 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 10. [31] [28] [30]

## Model View Presenter



Figura 10 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.

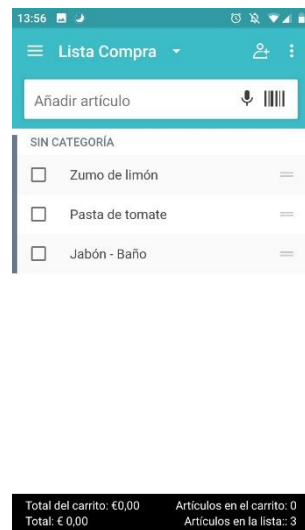
### Sección 3.5 Aplicaciones Similares

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 11 Vista de artículos (Out of milk)*



*Figura 12 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.

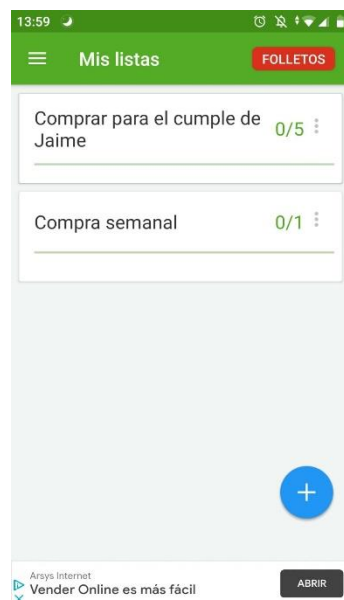


*Figura 14 Vista inicial de la aplicación (Bring!)*

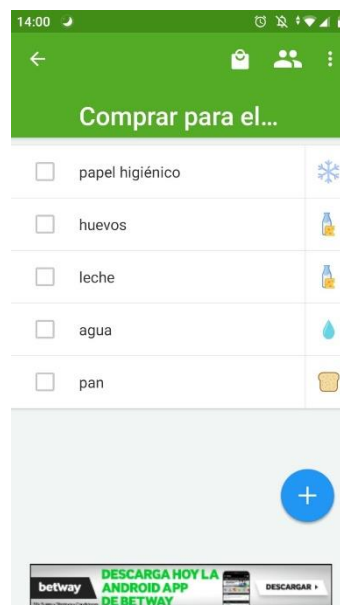


*Figura 13 Vista de recetas (Bring!)*

**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 15 Vista de distintas listas (Listonic)*



*Figura 16 Vista de productos (Listonic)*



## Capítulo 4. Método del Trabajo

La metodología que se va a utilizar para la realización de este proyecto será explicada en este capítulo.

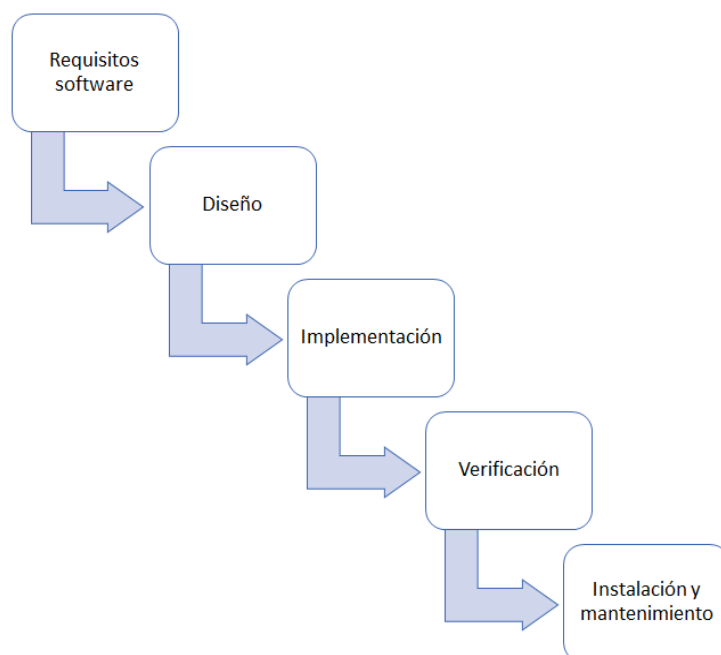
### Sección 4.1 Método de desarrollo en cascada

La metodología de desarrollo en cascada es una metodología secuencial, la cual ordena rigurosamente las etapas del proceso para el desarrollo de software, necesitando así que para iniciar con una etapa la anterior debe ser completamente finalizada, debido a esto se hará una revisión al finalizar cada etapa para comprobar que se puede avanzar a la siguiente fase [37].

Para la aplicación a desarrollar es importante el orden en el cual se efectúan las fases, por este motivo se utilizará el desarrollo en cascada, ya que, si desde el inicio del desarrollo se tienen claros los diseños tanto de la base de datos como de la parte visual al momento de desarrollar e implementar no se generarán dudas sobre los siguientes pasos a realizar.

#### Características del modelo en cascada

El modelo en cascada tiene un enfoque clásico, relacionado con el método de desarrollo lineal y secuencial, el cual se divide en distintas fases, estas están definidas por tareas y objetivos, estos objetivos deben finalizar en su totalidad para continuar con la siguiente fase. Este modelo anteriormente contaba con 7 fases, pero en la actualidad algunas de ellas se han fusionado quedando únicamente 5 fases [38], que son las siguientes representadas en la Figura 17:



*Figura 17 Infografía del modelo en cascada*

---

## Sección 4.2 Planificación

### Fase I

**Requisitos software:** en esta fase se va a captar las necesidades del cliente para determinar las características del software que se desarrollará, especificando los requisitos necesarios para la finalidad del producto, en este proceso se debe tener especial cuidado y atención ya que, la mayoría de las veces no se puede modificar los requisitos en el proceso de desarrollo.

**Requisitos necesarios (RN):** se deberá analizar cuáles son los requisitos mínimos para el correcto funcionamiento de la aplicación.

**RN1.** Identificar el público objetivo al que queremos dirigirnos.

**RN2.** Identificar entre ese público objetivo, qué dispositivos y versiones Android son las que más se utilizan.

**RN3.** Establecer unos requisitos mínimos con la intencionalidad de llegar al máximo público, pero sin perder funcionalidades en la aplicación.

**RN4.** Establecer el tipo de aplicación que desea el cliente.

**RN5.** Establecer en qué sistemas operativos va a funcionar.

### Fase II

**Diseño:** teniendo los requisitos establecidos, se debe tomar decisiones sobre la arquitectura de la información, las decisiones más importantes a tomar son; la definición, la organización y la estructura de todos los elementos que fuesen necesarios para el desarrollo del software en cuestión, estos elementos se deben tener correctamente relacionados con la estructura de datos y con el diseño de nuestra interfaz gráfica.

**Entorno de desarrollo (ED):** se deberá realizar una serie de pasos antes del desarrollo de la propia aplicación.

**ED1.** Instalación y configuración correspondiente de Android Studio como herramienta de desarrollo, posteriormente se deberá vincular con nuestro gestor de versiones.

**ED2.** Instalación y configuración de GitHub con GitKraken, creando un repositorio al que tendrán acceso todos los integrantes del TFG.

**ED3.** Configuración y vinculación de nuestra base de datos (Firebase) con el proyecto de Android Studio.

**ED4.** Creación de un nuevo tablero en Trello, se utilizará como gestor de trabajo para la asignación de tareas y la clasificación de las mismas.

**Diseño de la Base de Datos (DBD):** se debe pensar y analizar previamente qué datos se necesitarán para poder llevar a cabo la aplicación deseada.

**DBD1.** Se analizarán los requisitos para cada conjunto de datos que necesitemos, teniendo en cuenta en todo momento que nuestro gestor de base de datos es NoSQL.

**DBD2.** Diseño del modelo entidad-relación (E/R)

**DBD3.** Creación de cada una de las colecciones necesarias.

---

**DBD4.** Habilitación de las funciones de autenticación de usuarios en Firebase.

**Diseño Gráfico de la aplicación (DG):** en este apartado se realizará la interfaz de la aplicación a realizar. Susceptible a cambios durante todo el desarrollo.

**DG1.** Representación aproximada a través de la herramienta elegida para ello, en este caso Figma.

**DG2.** Se deberá sopesar para cada pantalla realizada, si deberemos crearla como un activity o un fragment, para el correcto rendimiento de la aplicación.

**DG3.** Realización de la interfaz con eXtensible Markup Language (XML), en este apartado empezaremos a darle funcionalidad a la interfaz, se harán cambios para adaptarlo a los distintos dispositivos.

### Fase III

**Implementación:** cuando la estructura de datos esté correctamente planteada, el siguiente paso a realizar sería empezar con la parte de programación, donde se harán actividades como el análisis de condiciones, creación de algoritmos y la implementación de estos en el lenguaje de programación elegido, en este caso Java.

**Desarrollo de la Aplicación (DA):** se incluyen los pasos aproximados a tomar para la implementación de código realizados con los medios softwares ya establecidos.

**DA1.** Creación de activities y fragments necesarios para cumplir el diseño deseado.

**DA2.** Vinculación de las zonas interactivas desde el XML al código, pudiendo así darle funcionalidad.

**DA3.** Configuración e implementación de las dependencias necesarias para trabajar con Firebase, que serían autenticación, base de datos y almacenamiento.

**DA4.** Creación de clases complementarias para la gestión de datos (como los adapters).

### Fase IV

**Verificación:** habiendo terminado la fase anterior, se empezará la fase de verificación, en esta fase se probará y ejecutará el código final, a la vez se verificará el correcto funcionamiento y que cumpla con los objetivos iniciales planteados por el usuario, el motivo de esta fase es encontrar defectos y mejorar la calidad del software antes de ser lanzado como producto final.

**Pruebas de Base de Datos (PBD):** se incluye los pasos a realizar para la validación de datos.

**PBD1.** Verificación de que los datos han sido añadidos/editados correctamente según el esquema ER (Véase en figura x).

**PBD2.** Verificación de carga correcta de los datos con los que se va a trabajar en cada una de las tablas.

**Pruebas de Desarrollo (PD):** se comprueba el correcto funcionamiento de las distintas herramientas utilizadas en el desarrollo.

**PD1.** Verificación de la conexión de la aplicación con la base de datos utilizada, en este caso Firebase.

**PD2.** Verificación de la conexión de la aplicación con la gestión de autenticación de usuarios.

**PD3.** Verificación del correcto funcionamiento de Github como sistema de control de versiones.

**Pruebas de Aplicación (PA):** se analizará las características principales de la aplicación una vez desarrollada.

**PA1.** Verificación del correcto funcionamiento de las funciones asociadas a la autenticación de usuarios (login, register, recuperar contraseña y logout).

**PA2.** Verificación de las funciones CRUD con un usuario de pruebas.

**PA3.** Verificación del correcto funcionamiento de notificaciones y ajustes.

## Fase V

**Instalación y mantenimiento:** cuando se haya completado la fase de verificación, se continuará con la fase de instalación, aquí se instalará y se comprobará que la aplicación funciona correctamente en los entornos elegidos en la fase de requisitos; en el caso del mantenimiento, se deberá destinar recursos para poder realizar modificaciones o corregir errores después de que se haya entregado el producto al cliente final.

**Fase de Mantenimiento(FM):** esta fase se realizará durante toda la vida útil de la aplicación una vez lanzada.

**FM1.** Analizar los reportes de incidencias mandados por los usuarios.

**FM2.** Actualización de versiones debido a los sistemas operativos, es probable que alguna funcionalidad cambie por ello se debe tener cuenta.

**FM3.** Realización de informes de evolución de la aplicación.

**FM4.** Mantener soporte por correo o similares.

## Capítulo 5. Resultados

### Sección 5.1 Modelo Entidad/Relación

A continuación, se mostrará el ER en la Figura 18 realizado para obtener un correcto funcionamiento a la hora de almacenar los datos.

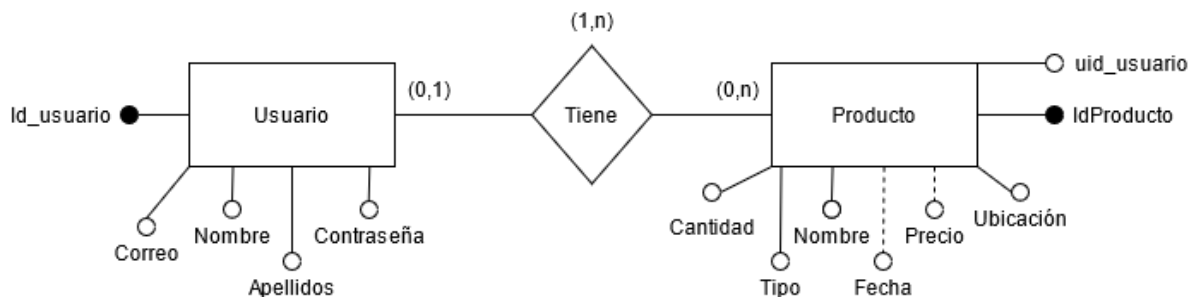


Figura 18 Representación del modelo entidad/relación

### Sección 5.2 Modelo de casos de uso

En esta sección se mostrará el modelo de casos de uso visible en la Figura 19, como único actor un usuario el cual podrá registrarse, cambiar su contraseña, logearse y gestionar los productos.

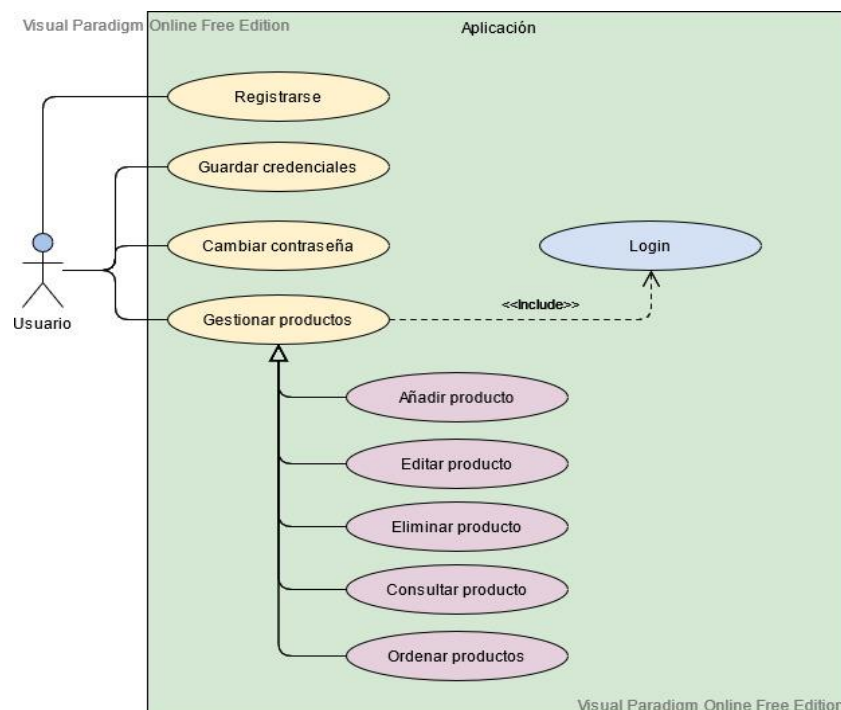


Figura 19 Modelo de caso de uso

## Sección 5.3 Vistas de la app

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

### 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 20.

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 20 Splash*

### 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 rellenar los campos para el correo y la contraseña del usuario, como se puede visualizar en la Figura 23, 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 22. Cuando estos datos estén rellenos terminará el registro al hacer click en el botón de registrarse.



Figura 21 Vista Inicial



Figura 23 Inicio de sesión

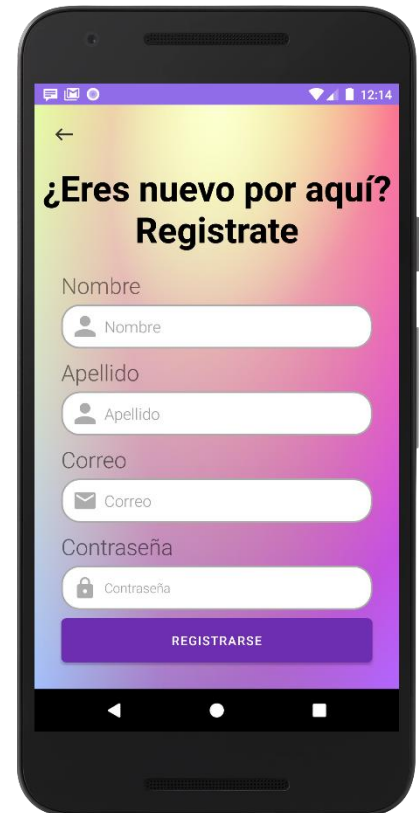


Figura 22 Registro de usuarios

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.

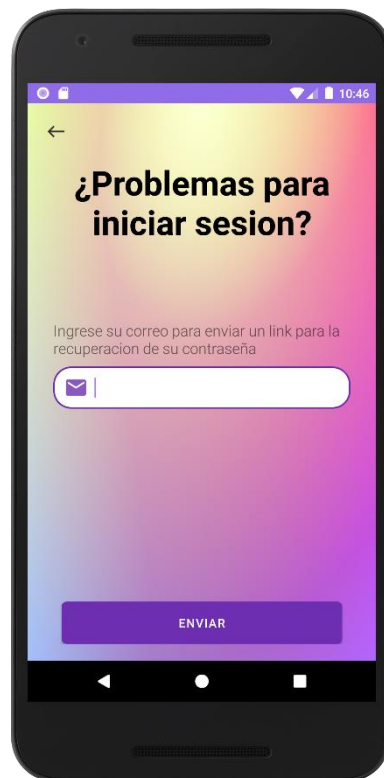
A continuación, se muestra el método principal de logeo:

```
//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().to  
String().trim());  
        changeToActivity();//Si el usuario y contraseña son correctos,  
se carga el NeveraActivity.  
    } else {  
        Toast.makeText(getContext(), "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 24 podrá rellenar sus datos y recibirá un correo para recuperar la contraseña.



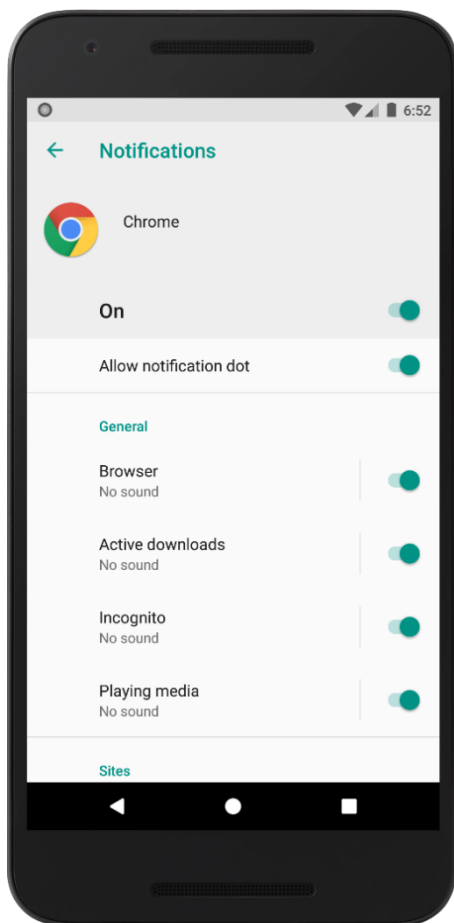
*Figura 24 Problema de inicio de sesión*



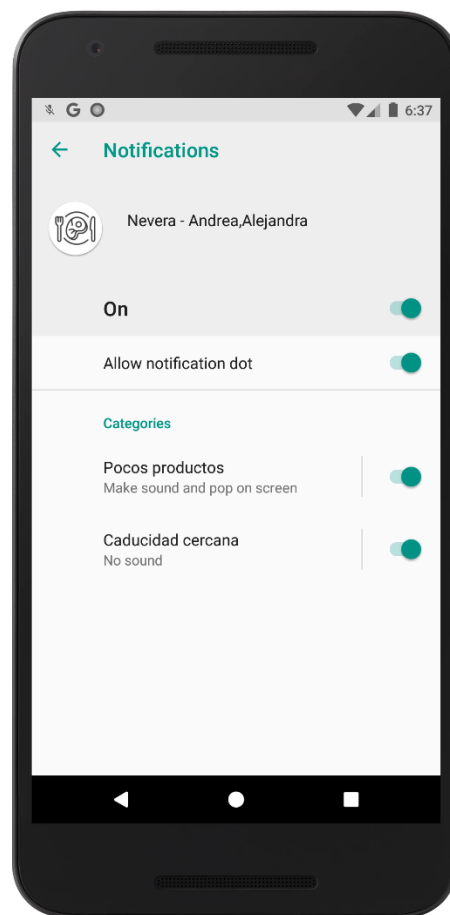
## Notificaciones

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 26, en este caso es Chrome.



*Figura 26 Ajustes de notificaciones chrome*



*Figura 25 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 25.

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;
}
```

## 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.

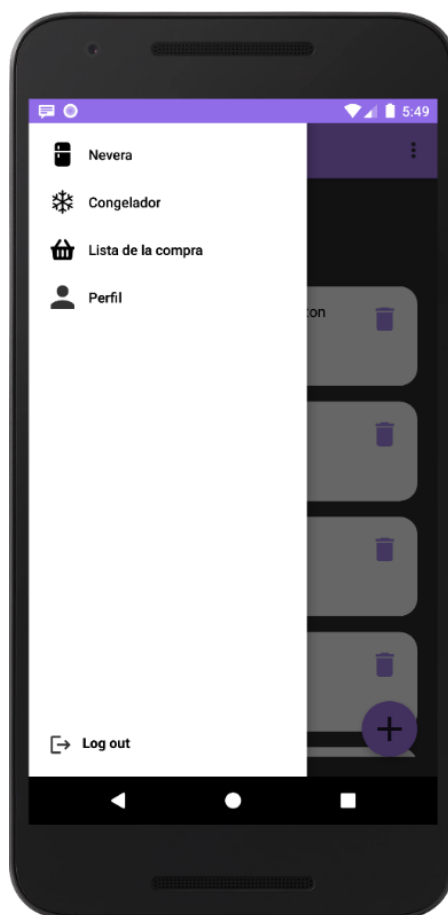


Figura 27 Drawer

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;
}
```

## 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 29, 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 28 Infografía de tipos de productos.

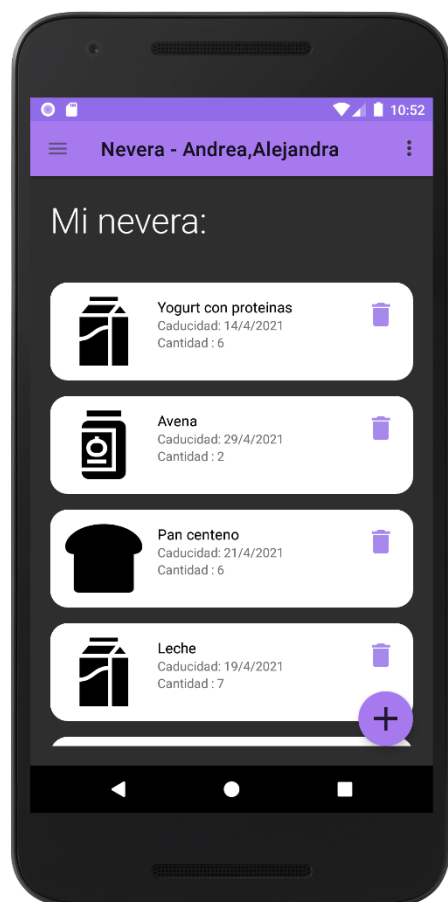


Figura 29 Lista de nevera



Figura 28 Infografía de tipos de productos

A la derecha de cada producto se observa un icono 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 30.

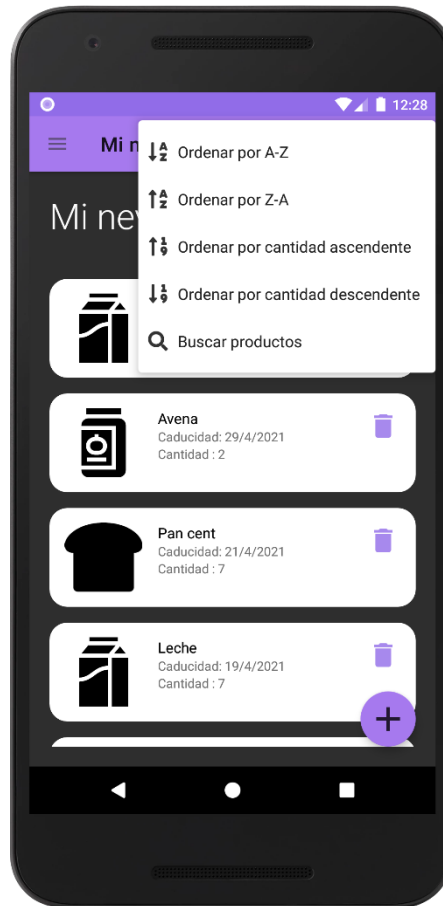


Figura 30 Menú funciones

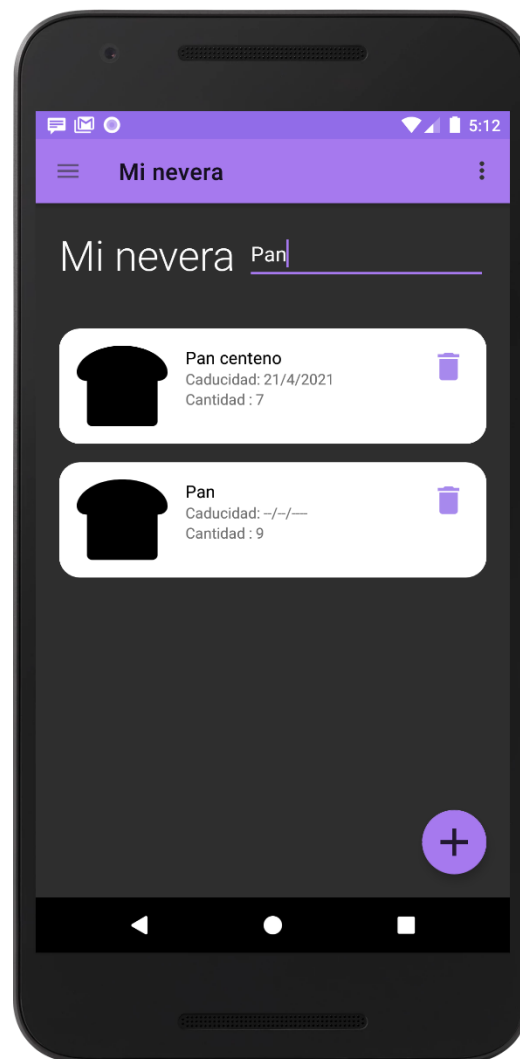
Estas funciones son administradas por el modelo del producto, donde se realiza un compareTo para organizar los productos de diferentes maneras, un ejemplo de esto sería el compareTo de la A-Z, como se puede ver a continuación:

```
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.

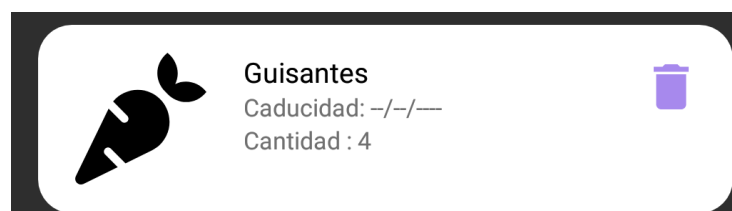
```
public void filterList(ArrayList<ProductoModelo> filteredList) {
    list = filteredList;
    notifyDataSetChanged();
}
```

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 desee. Se puede ver un ejemplo de ello en la Figura 31.



*Figura 31 Buscador*

En caso de que el producto no tenga precio o fecha de caducidad al guárdalo se visualizara como en la Figura 32.



*Figura 32 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.

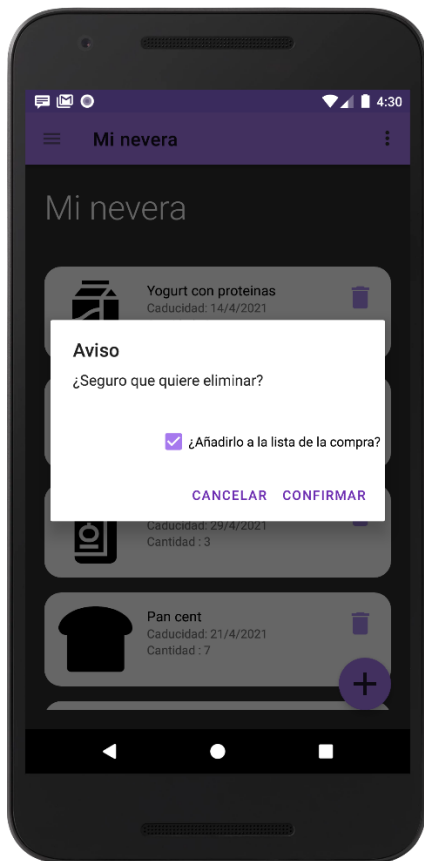


Figura 34 Pop-up Eliminar

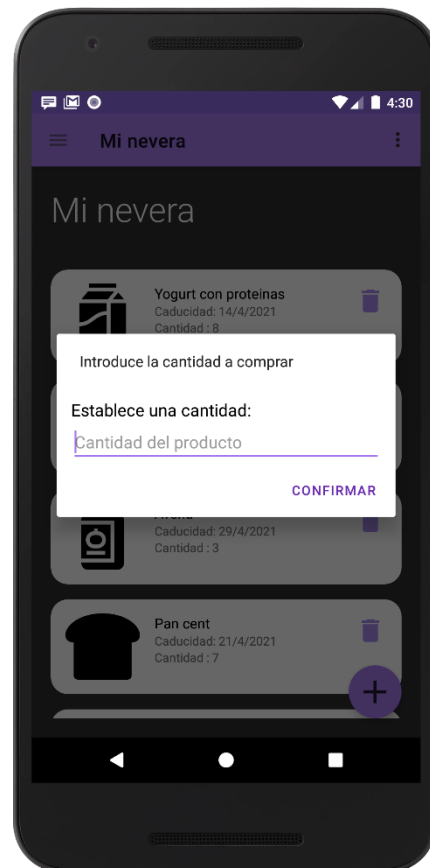


Figura 33 Pop-up Añadir cantidad

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 titulo
.setTitle("Aviso")//Es el cuerpo
.setView(checkBoxView) //Ponemos el checkbox
.setCancelable(false)//Con esta opción no se puede salir aunque pulse
atrás
.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 {

mDataBase.child("Producto").child(productoModelo.getId()).child("cantidad").setVal
ue(Integer.parseInt(cantidad.getText().toString().trim()));

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

mDataBase.child("Producto").child(productoModelo.getId()).child("ubicacion").setVa
lue("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();

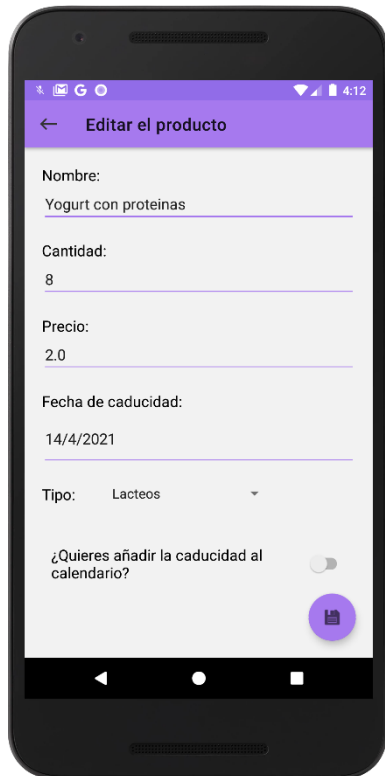
        } else { //Si no esta seleccionado el check

mDataBase.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 desee modificar y obtendrá la vista para editarlo, se puede ver esta vista en la Figura 35, aquí estarán predefinidos los campos con los valores anteriores del producto para que la edición sea más sencilla.



*Figura 35 Editar productos*



*Figura 36 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 icono con un amable mensaje, como se puede ver en la Figura 36.

---

## 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 38.

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 chequeados. Se puede ver la lista en acción en la Figura 37.

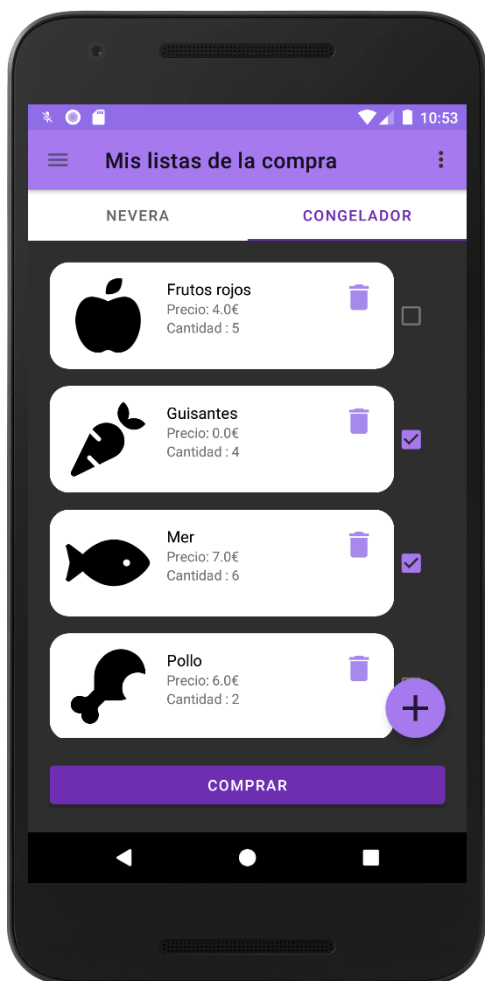


Figura 37 Listas de la compra

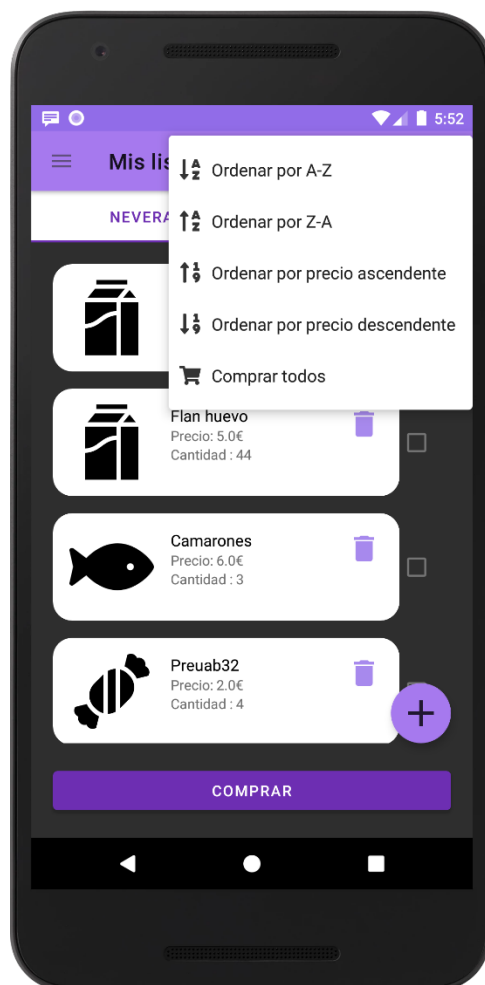


Figura 38 Menú desplegable

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 39.

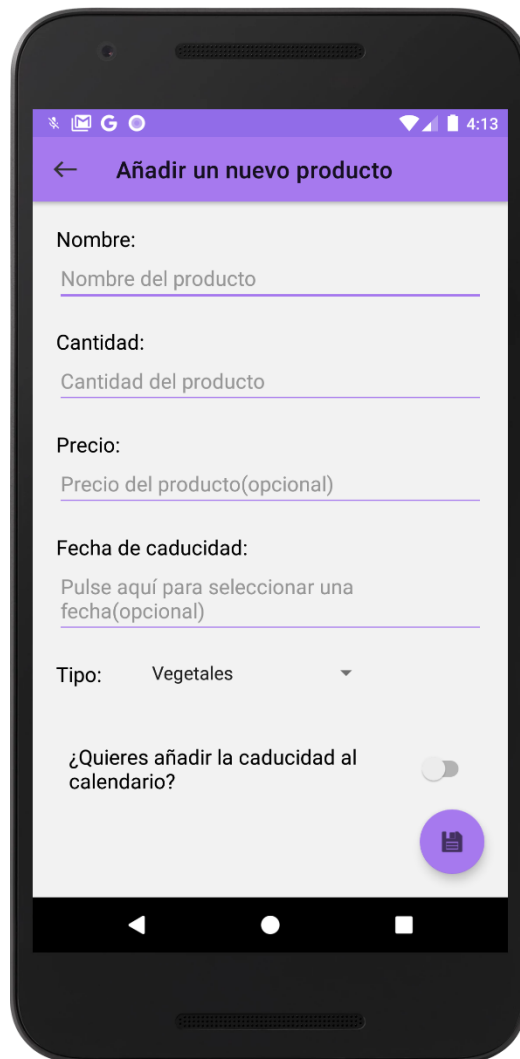


Figura 39 Añadir un nuevo producto

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
```

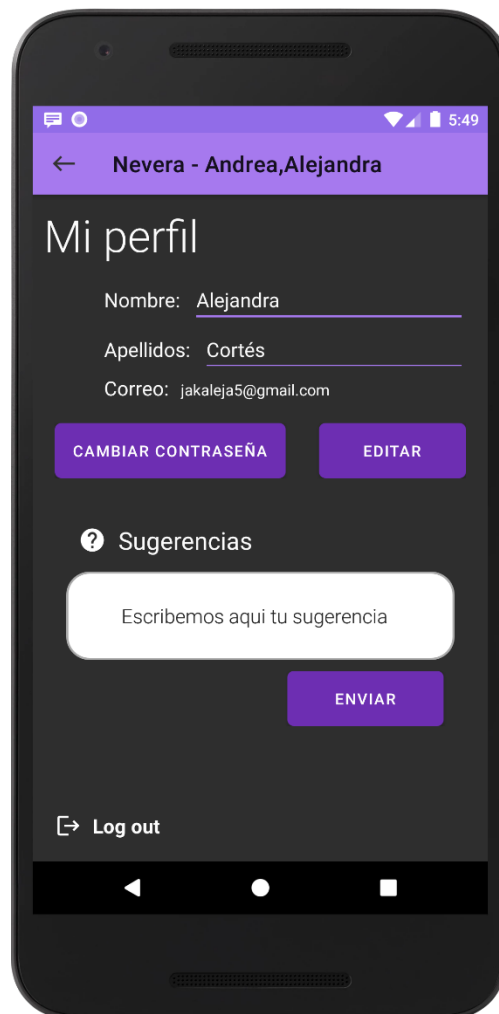
```
        cal.set(Calendar.YEAR, Integer.parseInt(fecha[2]));
        cal.set(Calendar.MONTH, Integer.parseInt(fecha[1]) - 1); //Es menos 1,
        debido a que enero es 0
        cal.set(Calendar.DAY_OF_MONTH, Integer.parseInt(fecha[0]));

        //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.DESRIPTION, "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(getPackageManager()) != 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();
    }
}
```

## 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 40*.



*Figura 40 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);
    //Aquí 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"});
    // Aquí definimos un título 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();
    }
}
```



## 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.

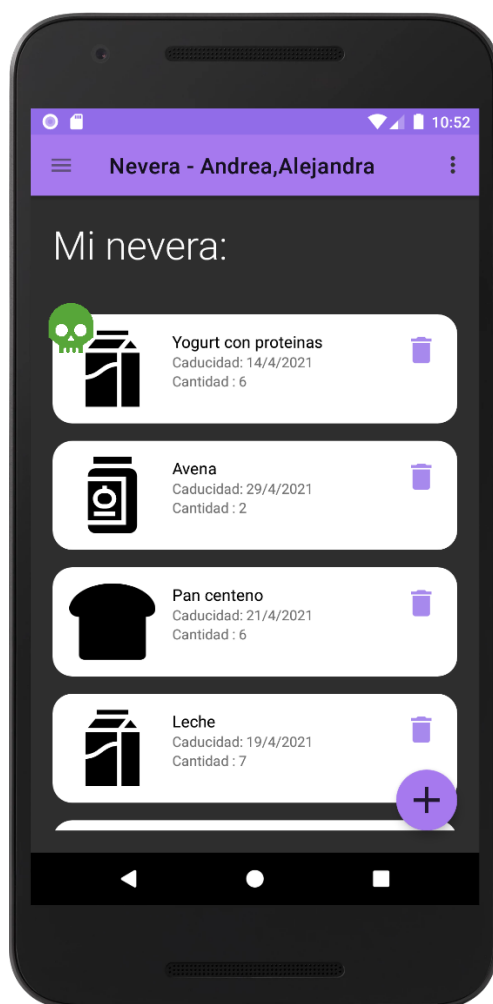
*Tabla 2 Desarrollo de objetivos*

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 afectiva 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 41.

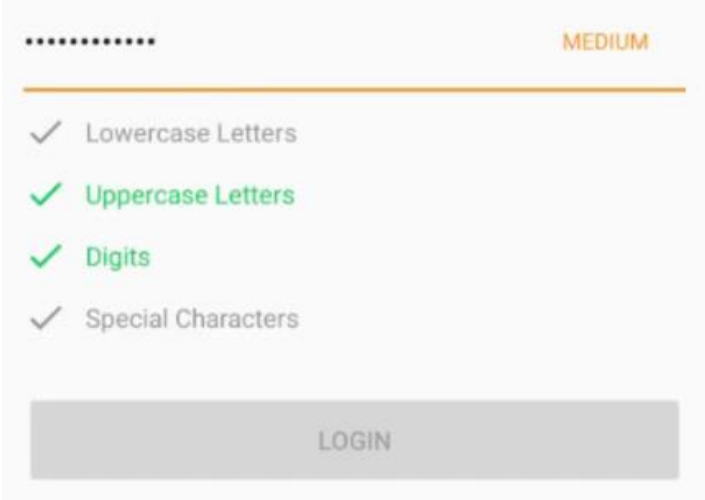


*Figura 41 Visualización de producto cercano a caducidad*

**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 envíe 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 42.



The image shows a password strength indicator interface. At the top, a password field is represented by a series of dots, followed by the word "MEDIUM" in orange text. Below this, a horizontal orange line separates the header from the requirements list. The list includes four items, each with a green checkmark icon and text: "Lowercase Letters", "Uppercase Letters", "Digits", and "Special Characters". At the bottom of the interface is a grey rectangular button with the word "LOGIN" in white capital letters.

*Figura 42 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/lp/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%93viewmodel>.
- [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] RYTEWiki, «Modelo en Cascada,» [En línea]. Available: [https://es.ryte.com/wiki/Modelo\\_en\\_Cascada](https://es.ryte.com/wiki/Modelo_en_Cascada).

[38] P. Domínguez, «En qué consiste el modelo en cascada,» 2 Junio 2020. [En línea]. Available: <https://openclassrooms.com/en/courses/4309151-gestiona-tu-proyecto-de-desarrollo/4538221-en-que-consiste-el-modelo-en-cascada>.

---

## 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