

# Universidad de Alcalá

## Escuela Politécnica Superior

Grado en Ingeniería de Computadores

### Trabajo Fin de Grado

Solución IoT para la gestión de alimentos en el hogar

**Autor:** Adrián Llana Ben

**Tutor/es:** Bernardo Alarcos Alcázar

2023



UNIVERSIDAD DE ALCALÁ  
Escuela Politécnica Superior

Grado en Ingeniería de Computadores

Trabajo de Fin de Grado

Solución IoT para la gestión de alimentos en el hogar

Autor: Adrián Llana Ben

Tutor/es: Bernardo Alarcos Alcázar

2023

**TRIBUNAL:**

**Presidente: Juan Ramón Velasco Pérez**

**Vocal 1º: Luis de la Cruz Piris**

**Vocal 2º: Bernardo Alarcos Alcázar**

**FECHA: Septiembre – 2023**



# Índice

1.	Introducción .....	1
2.	Objetivo.....	2
3.	Estado del arte .....	3
3.1	Orígenes de la nevera inteligente .....	3
3.2	Neveras inteligentes en la actualidad .....	4
3.3	Estudio de mercado .....	5
4.	Desarrollo .....	8
4.1	Hardware .....	8
4.1.1	Computador de placa única SBC.....	9
4.1.2	Pantalla táctil 5 pulgadas.....	12
4.1.3	Cámara .....	13
4.1.4	Disipador de calor .....	14
4.1.5	Micrófono.....	15
4.1.6	Controlador a distancia.....	16
4.2	Software .....	17
4.2.1	Sistema operativo .....	17
4.2.2	Entorno de desarrollo .....	18
4.2.3	Repositorio de código.....	18
4.2.4	Base de datos.....	20
4.2.5	Conexión a la BBDD.....	21
4.2.6	Lenguaje.....	22
4.2.7	Interfaz .....	22
4.2.8	Código y estructura .....	23
4.2.9	Servicios externos .....	33
4.2.10	Uso de Logs.....	35
5.	Conclusiones .....	36
6.	Trabajo futuro.....	37
6.1	Aprovechamiento de servicios externos .....	37
6.2	Funcionalidades a largo plazo .....	37
6.3	Configuraciones del sistema.....	37
6.4	Mejora de componentes .....	38
6.5	Mejoras en la interfaz.....	38
	Bibliografía .....	1
	Anexos.....	2
1.	Anexo I – Coste del proyecto.....	1
1.1	Coste de material.....	1

1.2	Coste de personal .....	2
1.3	Costes generales .....	3
1.4	Coste total.....	3
2.	Anexo II - Manual de usuario .....	4
2.1	Pantalla principal.....	4
2.2	NEW PURCHASE .....	5
2.3	STORED PRODUCTS .....	11
2.4	RECIPE IDEAS .....	14

## Índice de Figuras

Figura 3.1: Noticia de 1998 sobre las primeras neveras inteligentes .....	3
Figura 3.2: The LG Internet Digital DIOS (also known as R-S73CT).....	3
Figura 3.3: Función FamilyHub - View Inside.....	5
Figura 3.4: Función FamilyHub - Lista de la compra .....	5
Figura 3.5: Función FamilyHub - Comunicación familiar.....	6
Figura 3.6: Función FamilyHub - Calendario .....	6
Figura 3.7: Función FamilyHub - Smart View .....	7
Figura 3.8: Función FamilyHub - Smart Things .....	7
Figura 4.1: Esquema funcional del proyecto.....	8
Figura 4.2: Diagrama del hardware del proyecto .....	9
Figura 4.3: Pinout Raspberry Pi 4 Model B .....	10
Figura 4.4: Dimensiones Raspberry Pi 4 Model B.....	11
Figura 4.5: Raspberry Pi 4 Model B .....	11
Figura 4.6: Pantalla táctil .....	12
Figura 4.7: Pantalla y Raspberry conectadas.....	12
Figura 4.8: Montaje pantalla táctil .....	13
Figura 4.9: Módulo de cámara Pi Noir.....	13
Figura 4.10: Dimensiones disipador de calor.....	14
Figura 4.11: Conexiones disipador de calor .....	15
Figura 4.12: Placa con disipador de calor conectado .....	15
Figura 4.13: Micrófono USB .....	16
Figura 4.14: Rii X8 Mini.....	16
Figura 4.15: Diagrama del software del proyecto .....	17
Figura 4.16: Captura Raspberry Pi Imager.....	18
Figura 4.17: Archivo .gitignore .....	20
Figura 4.18: Logotipo Firebase .....	20
Figura 4.19: Captura de ejemplo de base de datos del proyecto .....	21
Figura 4.20: Ejemplo de interfaz desarrollada .....	22
Figura 4.21: Estructura de la aplicación.....	23
Figura 4.22: Carpeta IAResults.....	24
Figura 4.23: Carpeta components.....	24
Figura 4.24: Carpeta components/MenuOptions.....	25
Figura 4.25: Carpeta screens .....	26
Figura 4.26: Carpeta solutionAlerts .....	27
Figura 4.27: Ejemplo correo de alerta enviado .....	28
Figura 4.28: Captura bot de Telegram.....	29
Figura 4.29: Carpeta solutionDB .....	29
Figura 4.30: Carpeta styles.....	30
Figura 4.31: Configuraciones comunes.....	30
Figura 4.32: Configuraciones según apartado de la aplicación.....	31
Figura 4.33: Estilos aplicados en el menú principal.....	31
Figura 4.34: Estilos aplicados 'NEW PURCHASE' .....	32
Figura 4.35: Estilos aplicados 'STORAGE PRODUCTS' .....	32
Figura 4.36: Estilos aplicados 'RECIPE IDEAS' .....	32
Figura 4.37: Logo OpenAI.....	33
Figura 4.38: Logo OpenFoodFacts.....	34
Figura 4.39: Información del bot de Telegram.....	35
Figura 4.40: Ejemplo de logs registrados tras una ejecución .....	35
Figura 6.1: Aproximación mejora tabla de productos .....	38

Figura 6.2: Icono ‘borrar’ .....	39
Figura 6.3: Resultado de ejemplo con icono .....	39
Figura 2.1: Pantalla principal de la aplicación .....	4
Figura 2.2: Pantalla principal de ‘NEW PURCHASE’ .....	5
Figura 2.3: Preparación para el reconocimiento de voz .....	5
Figura 2.4: Pantalla mientras la aplicación escucha .....	6
Figura 2.5: Pantalla de confirmación de reconocimiento de voz .....	6
Figura 2.6: Pantalla insertar fecha .....	7
Figura 2.7: Ejemplo fecha incorrecta .....	7
Figura 2.8: Mensaje fecha incorrecta .....	7
Figura 2.9: Mensaje fallo en el reconocimiento de voz .....	8
Figura 2.10: Mensaje productos registrados.....	8
Figura 2.11: Preparación para la inserción por teclado .....	9
Figura 2.12: Inserción por teclado.....	9
Figura 2.13: Pantalla de confirmación entrada teclado .....	10
Figura 2.14: Preparación para el escaneo de productos .....	10
Figura 2.15: Pantalla mientras la aplicación escanea .....	11
Figura 2.16: Pantalla principal de ‘STORED PRODUCTS’ .....	11
Figura 2.17: Visualización de productos almacenados.....	12
Figura 2.18: Selección de un producto.....	12
Figura 2.19: Mensaje informativo de borrado de producto .....	13
Figura 2.20: Mensaje de advertencia antes de borrar todos los productos.....	13
Figura 2.21: Pantalla para eliminar todos los productos .....	14
Figura 2.22: Pantalla principal de ‘RECIPE IDEAS’ .....	14
Figura 2.23: Visualización listado productos .....	15
Figura 2.24: Modo de selección .....	15
Figura 2.25: Selección listado de productos para ideas de recetas .....	16
Figura 2.26: Resultado ideas de recetas 1/2 .....	16
Figura 2.27: Resultado ideas de recetas 2/2 .....	17



## Resumen

---

El Proyecto consiste en la creación de un sistema IoT para la gestión de alimentos en el hogar utilizando una Raspberry Pi. Se pretende incorporar en un frigorífico común las funcionalidades de uno inteligente haciendo uso de este dispositivo externo, al igual que, un Chromecast convierte en *Smart TV* un televisor corriente.

Esta solución permite llevar un registro de productos y sus fechas de caducidad, facilitando la planificación de compras y comidas.

Cuenta con otras funcionalidades como notificar al usuario con alertas cuando alguna de las fechas de caducidad esté próxima o proponer recetas que se puedan preparar con los ingredientes presentes en el sistema mediante IA.

## Palabras clave

*IoT, gestión de alimentos, Raspberry, IA.*

## Abstract

---

The Project involves developing an IoT system for home food management using a Raspberry Pi. The aim is to integrate smart refrigerator features into a regular one by making use of an external device, just as a Chromecast turns a regular TV into a Smart TV.

This solution allows keeping record of products and their expiration dates, making groceries and meal planning easier.

It also offers additional functionalities such as alerting the user when any of the expiration dates are approaching or suggesting recipes that can be prepared with the available ingredients using AI.

## Key words

*IoT, food management, Raspberry, IA.*

## **1. Introducción**

La tecnología IoT (Internet de las cosas) ha dejado de ser una novedad. En las últimas décadas ha sufrido un impresionante avance, cada vez podemos encontrarla en más hogares y multitud de formas diferentes, de hecho, raro es el ámbito en el que no está ya presente. El IoT ha demostrado su capacidad para transformar la forma en que interactuamos con el mundo que nos rodea. La transformación de nuestros hogares en espacios conectados y conscientes ofrece una oportunidad sin precedentes para abordar multitud de problemas.

La gestión de alimentos en el hogar es una de las áreas que podría beneficiarse enormemente de estas soluciones IoT. En un mundo en el que la sostenibilidad y la eficiencia son cada vez más importantes es fundamental contar con herramientas que nos ayuden a reducir el desperdicio de alimentos y mejorar la eficiencia en su consumo.

El desperdicio de alimentos es un dilema global que tiene profundas implicaciones económicas, ambientales y éticas. Nos encontramos en un contexto en el que los recursos naturales se vuelven cada vez más limitados y la población mundial sigue creciendo. Pese a ello, según datos de la Organización de las Naciones Unidas para la Alimentación y la Agricultura (FAO), aproximadamente un tercio de todos los alimentos producidos a nivel mundial se pierden o se desperdician anualmente [1]

A nivel nacional, en España, según datos del Ministerio de Agricultura, Pesca y Alimentación, los hogares españoles desperdiciaron durante el año 2021 un total de 1.245,86 millones de kilos o litros [2]

Por todo ello, es importante buscar soluciones innovadoras que aborden este desafío y el IoT tiene mucho potencial en este ámbito.

## 2. Objetivo

Este Proyecto abarca el diseño y desarrollo de una de estas soluciones IoT por medio de un prototipo. El objetivo general de esta implementación es ayudar al usuario a llevar un seguimiento del consumo de alimentos, planificar sus compras y comidas de manera eficiente para optimizar su uso, ahorrar tiempo, dinero y contribuir a un mundo más sostenible.

Como objetivos específicos destacan:

- **El desarrollo de una aplicación específica** para su uso en el prototipo. Con la finalidad de gestionar el registro de introducción y extracción de alimentos en el frigorífico y desde la que lanzar consultas de recetas, con una interfaz adaptada para cada proceso.
- **Crear y gestionar una base de datos en la nube**, hacer uso de una base de datos alojada en la nube donde almacenar los productos y su información, de esta forma se facilita el posible acceso desde diferentes dispositivos y localizaciones.
- **Generar un sistema de alertas**, dicho sistema consiste en notificar por correo electrónico y a través de un bot de Telegram al usuario cuando la fecha de caducidad de alguno de los productos almacenados en el sistema haya pasado o esté próxima. El mensaje mostrado es personalizado para cada usuario y la situación de cada producto.
- **Propuesta de recetas con IA:** Implementar una funcionalidad que permita al usuario, seleccionando un listado de productos entre los disponibles, obtener una serie de propuestas de recetas posibles con dichos ingredientes. Todo ello realizando peticiones a una API de un servicio que use la Inteligencia Artificial.

Para ello se hace uso de una Raspberry Pi 4 model B. En esta se introducen diversas funcionalidades que permiten que la gestión de alimentos sea un proceso óptimo y automatizado.

### 3. Estado del arte

#### 3.1 Orígenes de la nevera inteligente

Aunque el concepto de frigorífico inteligente ha ganado popularidad en los últimos años, no es tan reciente como se piensa. Ya en 1998 comenzaban a publicarse noticias augurando su salida al mercado con titulares como *Internet ovens and fridges are "looming on the horizon"* [4] como se muestra en la Figura 3.1.



Figura 3.1: Noticia de 1998 sobre las primeras neveras inteligentes

El primer refrigerador inteligente que salió al mercado fue desarrollado a principios del milenio, en septiembre del año 2000 LG Electronics presentó una nevera conectada a Internet bajo el nombre de *LG Inernet Refrigerator* (Figura 3.2), un proyecto que comenzó a desarrollarse en 1997. Este modelo disponía de un módem 56k y una pantalla táctil de 15 pulgadas en una de sus puertas. Además de navegar por Internet y de tener correo electrónico, permitía descargar archivos MP3, conectar un DVD, hacerse una foto o ver la televisión. También era capaz de pedir automáticamente la compra de los alimentos que se han acabado o están a punto de caducar [3].



Figura 3.2: The LG Internet Digital DIOS (also known as R-S73CT)

A pesar de sus características pioneras, su recepción en el mercado no fue tan favorable como se esperaba. Muchos consumidores consideraron que lo que este producto podía llegar a aportar en sus hogares no justificaba su elevado precio (\$20,000). Además, la empresa de ciberseguridad rusa Kaspersky Lab advirtió que estos frigoríficos conectados a Internet podrían ser objetivos fácilmente atacables, ya fuese con virus que provocasen la apertura de la puerta en mitad de la noche, o como emisor de grandes cantidades de emails maliciosos. [5]

Todo ello hizo que el proyecto cosechase un éxito casi insignificante con relación al capital invertido para desarrollarlo.

Con el paso de los años se han ido sucediendo nuevos modelos de neveras inteligentes, incorporando nuevas funciones y mejorando sus prestaciones.

### **3.2 Neveras inteligentes en la actualidad**

A pesar del poco éxito que ha obtenido históricamente este tipo de producto, grandes marcas siguen apostando por desarrollar nuevos modelos. Esto refleja la creciente convicción de que la tecnología es capaz de agregar valor añadido a un electrodoméstico como puede ser una simple nevera.

Hoy en día, dentro del mundo desarrollado, es impensable la idea de una vivienda carente de un frigorífico, este elemento se ha convertido en un punto central del hogar. La sociedad avanza hacia un mundo cada vez más interconectado y las neveras inteligentes presentan la oportunidad de revolucionar la forma en que gestionamos nuestra vida cotidiana e interactuamos con nuestros alimentos. Para ello, estos dispositivos cuentan con funcionalidades como las siguientes:

- **Pantalla LCD.** En ocasiones cuentan tanto con este tipo de pantalla táctil con la que interactuar fácilmente como con un mando a distancia. Estas pantallas suelen colocarse en una de las puertas del frigorífico. Gracias a software desarrollados con interfaces adaptadas, permiten visualizar en esta pantalla información sobre el contenido o el estado del dispositivo, gestionar configuraciones o descargar nuevas aplicaciones.
- **Conexión a Internet.** Generalmente a través de Wi-Fi, permite mantener el dispositivo al día en cuanto actualizaciones, usar servicios externos y acceder a él de forma remota desde cualquier otro lugar.
- **Cámaras integradas.** Permiten visualizar el interior del frigorífico de forma remota y sin necesidad de abrirlo, aumentando su eficiencia energética.
- **Altavoces y micrófono integrados.** Con los que interactuar con asistentes de voz, recibir notificaciones del hogar conectado o escuchar música.
- **Control de temperatura inteligente.** Permite controlar la temperatura del frigorífico y ajustarla de forma automática en función de las necesidades del usuario.
- **Conectividad con aplicaciones móviles.** Muchos modelos tienen aplicaciones propias diseñadas especialmente para su uso con ese frigorífico, lo que facilita el proporcionar una experiencia óptima y personalizada.

Todas estas funcionalidades contribuyen a los beneficios que puede aportar uno de estos frigoríficos inteligentes frente a uno común.

### 3.3 Estudio de mercado

Existen pocas diferencias en cuanto a funcionalidades entre las opciones de neveras que se pueden encontrar actualmente en el mercado, pues la mayoría ofrecen prácticamente las mismas. Los apartados en los que más se diferencian suelen ser las especificaciones como tamaño de la pantalla, capacidad propia del refrigerador/congelador o la inclusión o no de cámara interior.

Para ver la situación actual del mercado servirá con estudiar uno de los más completos y mejor valorados, es el conocido como Side by Side, de la gama de frigoríficos *Family Hub*, de la marca Samsung [6]. Entre las funcionalidades que ofrece este dispositivo destacan:

- **View Inside.** (Figura 3.3) Con esta función es posible visualizar el contenido del frigorífico en la pantalla de inicio o desde un smartphone, en el momento de hacer la compra, por ejemplo.

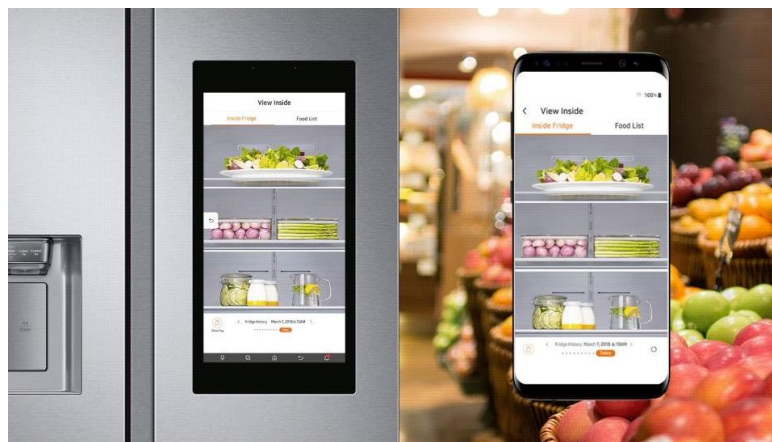


Figura 3.3: Función FamilyHub - View Inside

- **Lista de la compra.** (Figura 3.4) Esta función permite crear listas de la compra añadiendo artículos a la lista desde la pantalla, pudiendo sincronizarla con un smartphone para poder tenerla presente en el supermercado.

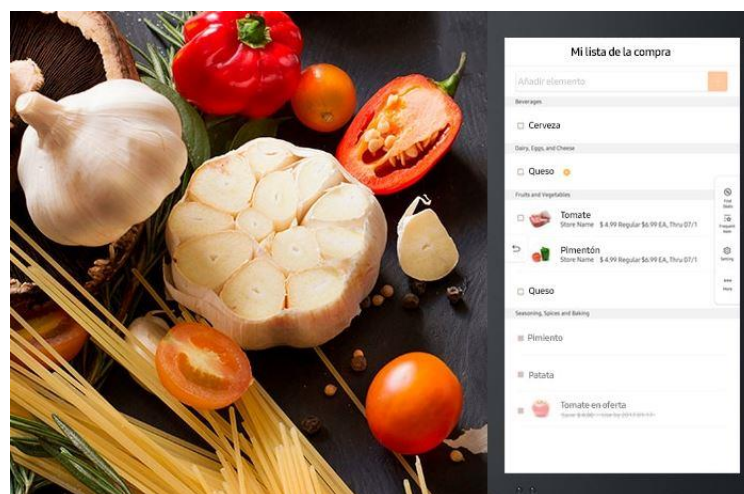


Figura 3.4: Función FamilyHub - Lista de la compra

- **Planificador de comidas.** Permite añadir menús semanales gestionándolos junto a un calendario, ayudando a prevenir la monotonía a la hora de no repetir o abusar de alimentos determinados y a llevar una dieta equilibrada y variada. Además, esta funcionalidad puede ser especialmente útil para aquellos que siguen alguna dieta especial o con restricciones alimenticias.
- **Comunicación familiar.** (Figura 3.5) Con esta función es posible interactuar con el resto de miembros que residan en la vivienda. Interacciones por medio de notas escritas a mano, dibujos, fotos o listas de cosas por hacer, sustituyendo las típicas notas escritas que terminaban en la puerta de la nevera.



Figura 3.5: Función FamilyHub - Comunicación familiar

- **Calendario.** (Figura 3.6) Para compartir y visualizar las agendas de todos los integrantes de la casa de un vistazo, pudiendo añadir o actualizar las entradas desde el propio frigorífico o desde el smartphone.

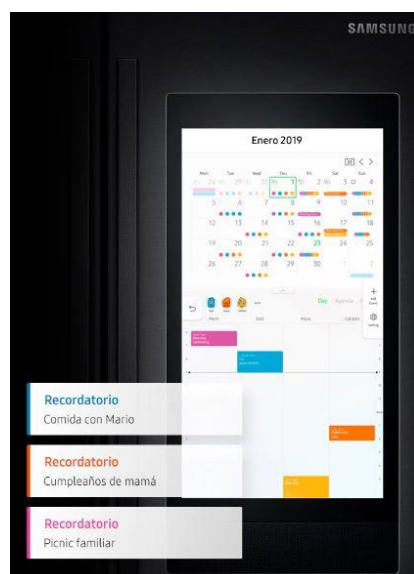


Figura 3.6: Función FamilyHub - Calendario



- **Smart View.** (Figura 3.7) Esta función permite desplegar contenido en la pantalla del frigorífico. Desde programas de TV, vídeos de recetas, pantalla del smartphone, etc. Todo ello se pudiéndose visualizar mientras cocinas.



Figura 3.7: Función FamilyHub - Smart View

- **SmartThings™.** (Figura 3.8) Con esta función se podrá conseguir una mayor domotización del resto del hogar, ya que permite controlar el resto de electrodomésticos conectados desde la pantalla de la nevera. Existe también una app para gestionarlo a través de un smartphone.



Figura 3.8: Función FamilyHub - Smart Things

## 4. Desarrollo

La pretensión de este desarrollo engloba diversas funcionalidades, similares a las vistas en el ejemplo del apartado anterior, con las principales ventajas de que, al ser un prototipo externo, es aplicable a cualquier frigorífico y su coste se ve reducido. Como se muestra en la Figura 4.1, las funcionalidades que implementa son:

- El **registro de nuevos productos** tras una compra, pudiendo elegir entre tres métodos diferentes de registro (reconocimiento de voz, entrada por teclado y escaneo de código de barras).
- La **gestión de almacenamiento**, permitiendo la visualización y eliminación de los productos ya registrados en el sistema.
- **Ideas de recetas**, que permite, tras la selección de un listado de productos presentes en el sistema, obtener recetas posibles a preparar.
- Y finalmente, **alertas / notificaciones**, que hacen saber al usuario si la fecha de caducidad de alguno de los productos presentes en el sistema ha pasado o está próxima.



*Figura 4.1: Esquema funcional del proyecto*

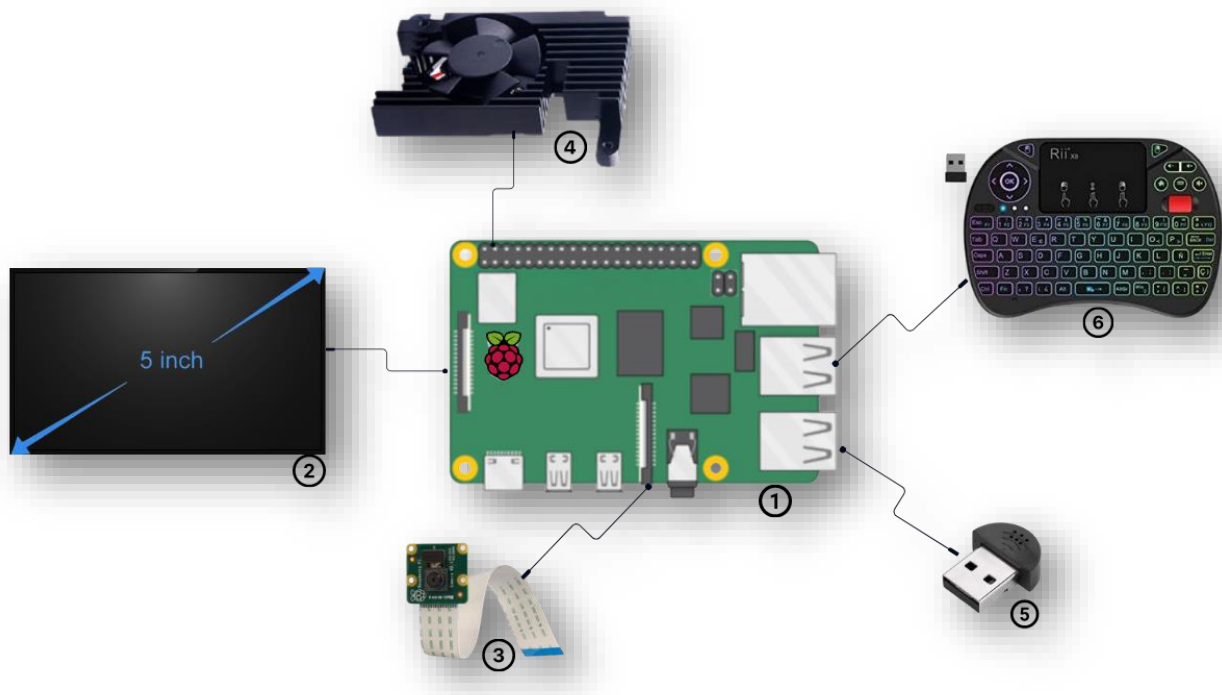
Este apartado contiene el proceso detallado que se ha seguido para desarrollar este proyecto, dividido principalmente en: **hardware** empleado, **software** empleado e **implementación** entre ambos.

### 4.1 Hardware

Para desarrollar un prototipo que añada funcionalidades propias de una nevera inteligente a una común es importante la selección e implementación de los componentes hardware. En este apartado se detallan todos los elementos que conforman la infraestructura necesaria para montar el prototipo.

Cada componente ha sido seleccionado en base a un criterio claro: La optimización de la relación entre presupuesto y funcionamiento. Este enfoque ha sido clave para garantizar que cada componente contribuya de la mejor forma posible a la arquitectura del prototipo sin permitir que su coste tenga un impacto demasiado elevado en el presupuesto global del proyecto. Al considerar cuidadosamente las capacidades técnicas de cada componente junto con las limitaciones financieras se ha logrado un equilibrio entre el eficiente rendimiento del prototipo y su coste general acumulado.

En la Figura 4.2 se observa un diagrama con los diferentes componentes que integran el prototipo.



*Figura 4.2: Diagrama del hardware del proyecto*

1. Computador de placa única SBC
2. Pantalla táctil
3. Módulo de cámara
4. Disipador de calor
5. Micrófono USB
6. Controlador a distancia

A continuación, un análisis de cada uno de los componentes empleados.

#### **4.1.1 Computador de placa única SBC**

Como elemento central del desarrollo y ‘cerebro’ del prototipo, se encuentra un computador de placa única SBC (Single Board Computer). Se trata de un dispositivo que integra los elementos esenciales de una computadora en una sola placa de circuito impreso, un ordenador que presenta una configuración compacta y simplificada, ideal para aplicaciones en las que el tamaño, el costo y la eficiencia son importantes, como lo son en este proyecto.

## Raspberry Pi 4 model B 2GB

La opción seleccionada para este componente ha sido el modelo Raspberry Pi 4 model B. Este modelo cuenta con unas especificaciones idóneas para sustentar el propósito del prototipo.

Pese a su reducido tamaño esta placa brinda una considerable potencia de procesamiento, capacidad de memoria más que suficiente, una amplia conectividad y variedad de puertos. Por otro lado, es muy compatible con otros componentes hardware y software externos. Ha sido diseñada para ser eficiente en términos de consumo energético y al tratarse de la conocida marca Raspberry, cuenta con una gran comunidad de desarrolladores y usuarios que proporcionan soporte e importantes recursos.

Las especificaciones de este modelo son las siguientes:

En un chip Broadcom BCM2711 cuenta con un **procesador** ARM Cortex-A72 de un conjunto de instrucciones de 64-bit que trabaja a una frecuencia de reloj de 1,50 GHz.

Este modelo tiene implementaciones con diferentes configuraciones de **memoria** (1GB, 2GB, 4GB o 8GB), en este caso se trata del de 2GB LPDDR4 SDRAM.

En el apartado de la **conectividad**, cuenta con una configuración Wi-Fi de 2.4 GHz y 5.0 GHz IEEE 802.11b/g/n/ac, Bluetooth 5.0, Gigabit Ethernet, 2 puertos USB 3.0 y otros 2 puertos USB 2.0. Más adelante se detallan las conexiones de las que se hace uso este prototipo.

En cuanto a **expansión** cuenta con cabezal GPIO de 40 pines, mostrados de forma detallada en el 'pinout' de la Figura 4.3.

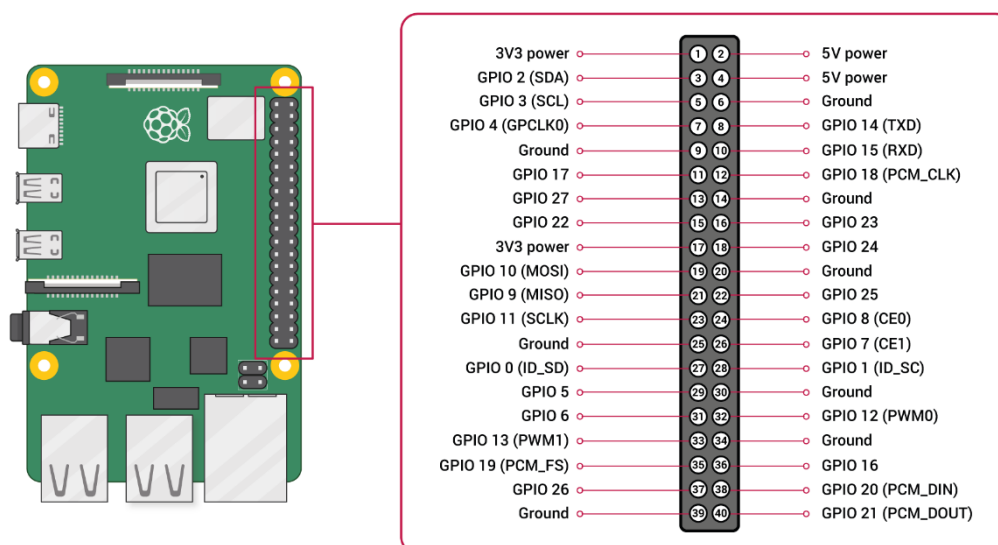
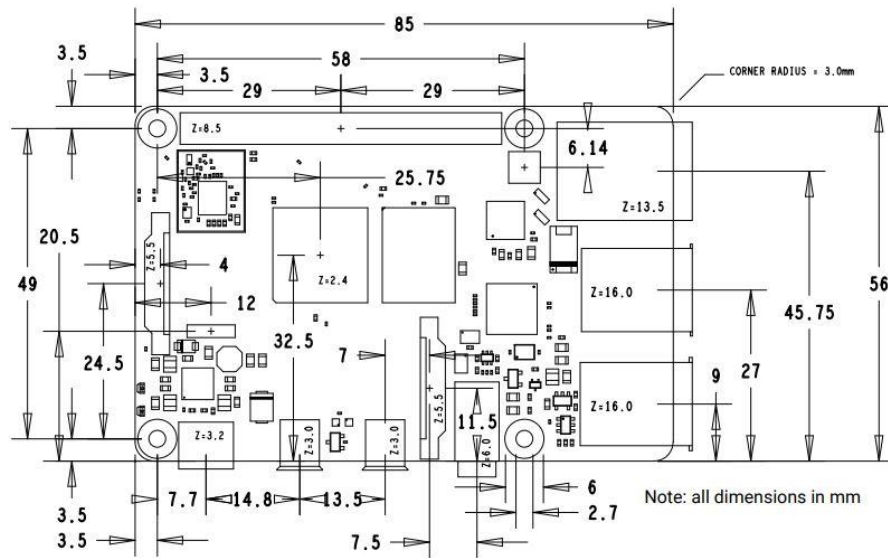


Figura 4.3: Pinout Raspberry Pi 4 Model B

Para el apartado **multimedia** cuenta con 2 puertos micro-HDMI que admiten pantallas de 4k@60Hz con HDMI 2.0, un puerto de pantalla MISI DSI, un puerto de cámara MIPI CSI. Cuenta con una ranura para micro-SD, necesaria para cargar un sistema operativo en la placa.

Para la **alimentación** es necesaria una corriente de continua de 5V y un mínimo de 3A, vía un puerto USB-C o vía cabezal GPIO.

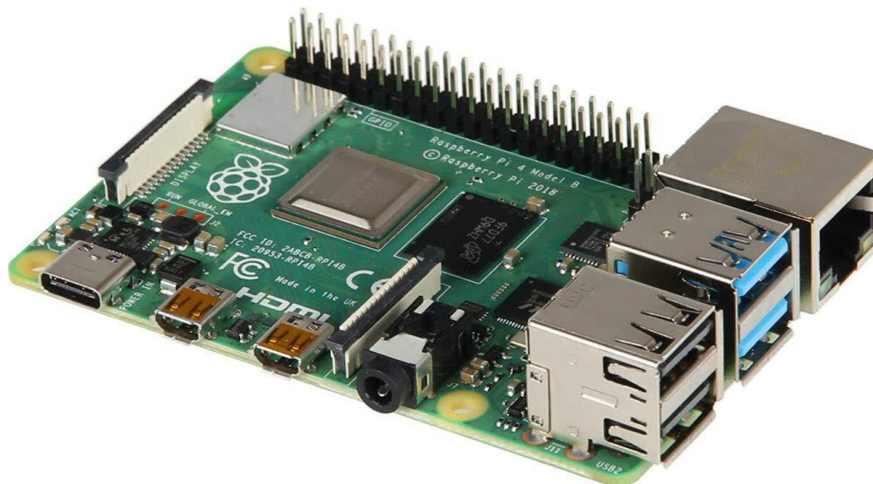
Se pueden observar al detalle las **dimensiones** de la placa en la Figura 4.4.



*Figura 4.4: Dimensiones Raspberry Pi 4 Model B*

Por último, para el **entorno**, la temperatura en la que puede operar este dispositivo se encuentra entre 0-50°C.

En la Figura 4.5 se observa una imagen real de la placa, con todas las especificaciones mencionadas anteriormente.



*Figura 4.5: Raspberry Pi 4 Model B*

### 4.1.2 Pantalla táctil 5 pulgadas

La incorporación de una pantalla táctil en el prototipo es fundamental para poder ofrecer al usuario las funcionalidades desarrolladas en él de una forma accesible e intuitiva. Gracias a ella y junto con la interfaz específica desarrollada para el proyecto, la interacción entre el usuario y el prototipo se ve reforzada.

El tamaño seleccionado de 5 pulgadas (Figura 4.6) se adapta a la perfección con la Raspberry Pi, en el siguiente apartado se puede encontrar información detallada sobre el modelo de pantalla seleccionado y cómo se lleva a cabo el montaje en la placa.

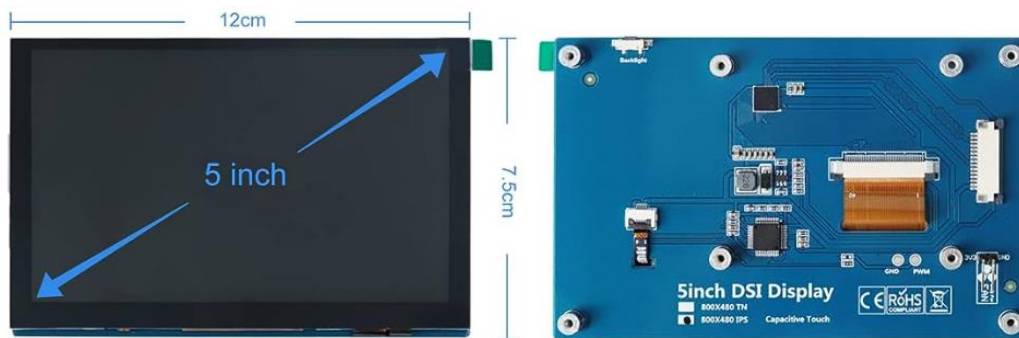


Figura 4.6: Pantalla táctil

### *Freenove 5 Inch Touchscreen Monitor*

El modelo seleccionado como pantalla para el prototipo es esta Freenove Touchscreen Monitor, la opción de 5 pulgadas. Las dimensiones del componente son de 15,2 x 14 x 3,8 cm; 191 gramos, estas se adaptan a la placa base a la perfección (Figura 4.7) sin dejar de mostrar la información de forma clara y detallada, cuenta con una resolución de 800 x 480.



Figura 4.7: Pantalla y Raspberry conectadas



Su interfaz táctil facilita la interacción con el usuario y mejora su experiencia. Ha sido diseñada y optimizada para su uso con la Raspberry Pi, por lo que garantiza compatibilidad y una configuración sencilla. Aunque la placa dispone de entradas HDMI, esta pantalla utiliza su puerto MIPI DSI para la conexión y el paso de información. El montaje es sencillo, como se muestra en la Figura 4.8.

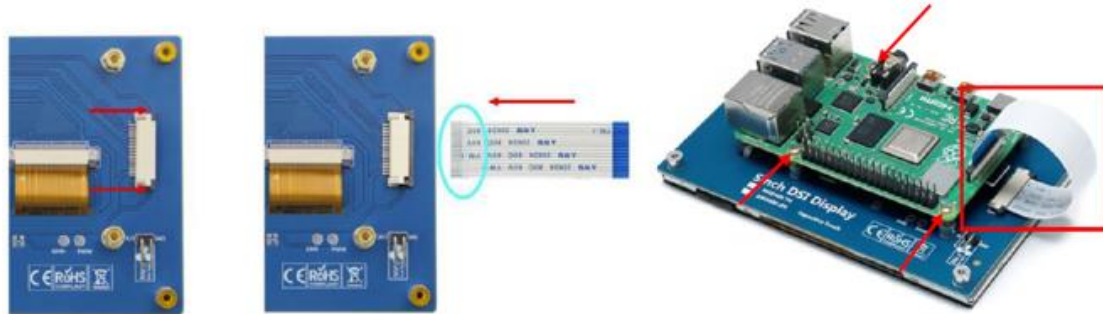


Figura 4.8: Montaje pantalla táctil

### 4.1.3 Cámara

La integración de un módulo de cámara en el prototipo se ha llevado a cabo con el propósito de habilitar uno de los tres tipos de reconocimiento de productos. Para ello, el módulo de cámara capta una imagen en la que aparece el código de barras de un producto. Con este componente permitimos que el usuario tenga una opción más donde elegir para el proceso de registro de productos en el sistema.

#### ***Pi Noir Módulo v2.1 8MP 1080p***

La selección de este modelo de cámara (Figura 4.9) se ha basado en sus características técnicas, apropiadas para su propósito en el proyecto. Es apta para captar imágenes con una resolución de 8 megapíxeles y en entornos de baja luminosidad, de muy fácil integración con la Raspberry Pi, para el montaje hace uso de su puerto MIPI CSI. Cuenta con unas dimensiones muy ajustadas (0,25 x 0,24 x 0,01 cm y 3 gramos), ayudando al diseño compacto del prototipo.

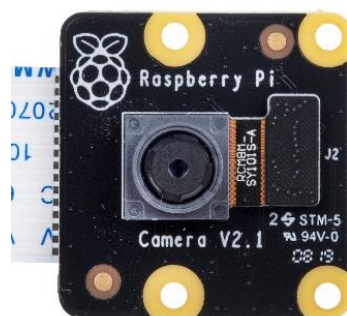


Figura 4.9: Módulo de cámara Pi Noir

#### 4.1.4 Disipador de calor

La incorporación de un disipador de calor al prototipo se ha realizado con varios propósitos. Durante el desarrollo y las pruebas de las funcionalidades que implementa el prototipo es importante controlar la temperatura del mismo, ya que son procesos que exigen mucho a los componentes. Un sobrecalentamiento puede llegar a afectar a su rendimiento y vida útil. Por lo que la integración de este componente ayuda a la estabilidad del sistema, a un mayor margen de su rendimiento y a prolongar la vida útil del proyecto en general, aparte de permitir que el prototipo funcione en un ambiente controlado, dentro de un rango seguro de funcionamiento e independientemente de las condiciones externas o el nivel de exigencia al que se exponga.

##### *GeeekPi Armor lite con ventilador PWM*

El modelo seleccionado como disipador está diseñado para la Raspberry Pi e implementa una funcionalidad diferencial que lo distingue entre la mayoría de las opciones para disipar el calor en estas placas.

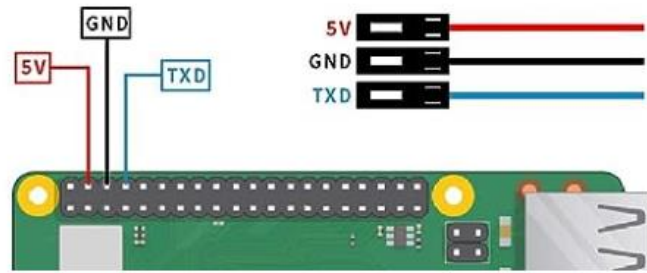
Es compatible con el control de velocidad PWM (Pulse Width Modulation), que permite controlar la velocidad mediante software y es compatible con el Sistema Operativo empleado. Funciona como interruptor, también conocido como *gate*, consiguiendo una conexión y desconexión de la energía a cada pulso. PWM convierte una señal digital en una analógica cambiando la cantidad de tiempo que se mantiene encendida o apagada en su ciclo de trabajo.[8] Sus dimensiones se adaptan perfectamente a las de la placa, se muestran en la Figura 4.10.



*Figura 4.10: Dimensiones disipador de calor*

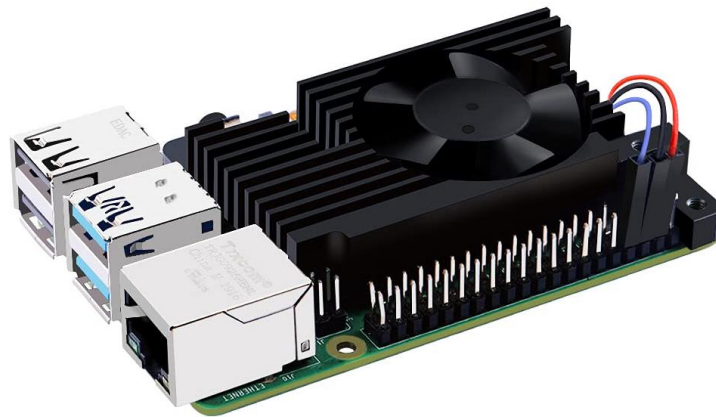
Su montaje y conexión se realiza por medio de los pines GPIO. En este caso se usa la toma de 5V, el pin 6 como tierra o GND y el 8 (GPIO 14 TXD) para transmitir los datos como se observa en la Figura 4.11.





*Figura 4.11: Conexiones disipador de calor*

Tras el montaje y conexión del disipador en la placa como se ha especificado, el resultado se muestra en la Figura 4.12.



*Figura 4.12: Placa con disipador de calor conectado*

### 4.1.5 Micrófono

Al igual que en caso de la integración del módulo de cámara, se añade un micrófono al componente para habilitar otro de los tres tipos de reconocimiento de productos, en este caso, por reconocimiento de voz.

Posiblemente esta sea la opción por la que más usuarios se decanten, ya que permite añadir productos al sistema de una forma cómoda y sencilla, por la simple mención de su nombre, lo que permite agilizar el proceso de registro y mejorar la experiencia del usuario.

#### ***Fasient USB Microphone Mini***

Es un micrófono omnidireccional de 360° con una distancia efectiva de hasta 2 metros y que soporta hasta 67 dB. A pesar de existir otras alternativas con otro tipo de conexiones, se ha seleccionado ésta aprovechando uno de los puertos USB de la placa Raspberry.

Este modelo (Figura 4.13) mantiene el diseño simple y funcional. Sus dimensiones son muy reducidas, apenas sobresaliendo del puerto USB (2,2 x 1,9 x 0,4 cm)



*Figura 4.13: Micrófono USB*

#### 4.1.6 Controlador a distancia

Con el fin de facilitar el manejo del sistema y la introducción de información, se incluye un controlador a distancia con teclado y panel táctil a modo de ratón. Este componente añade un método alternativo para la interacción con el usuario, enriqueciendo la experiencia y garantizando flexibilidad en el uso.

Además, sirve como medida cautelar ante posibles fallos en los otros dos métodos de registro de productos (reconocimiento de voz y escaneo de código de barras), este componente añade el tercer formato, a través del teclado, permitiendo escribir exactamente el nombre del producto que pretendemos registrar, así como su fecha de caducidad, además de permitir la navegación por todo el sistema.

#### ***Rii X8 Mini***

Entre las especificaciones de este modelo (Figura 4.14) se encuentra, la conexión de este componente con el prototipo se realiza a través de un receptor (dongle) en uno de los puertos USB, cuneta con un modo de transmisión inalámbrico GFSK 2.4GHz, con un rango de hasta 10m gracias a una potencia de transmisión de  $\pm 5$ db. Sus dimensiones son reducidas, 13,8 x 9,6 x 2,1 cm y un peso de 112,5g. Finalmente, cuenta con una batería recargable de ION-Litio.



*Figura 4.14: Rii X8 Mini*

## 4.2 Software

El software es un apartado esencial en cualquier proyecto tecnológico. Proporciona inteligencia, permitiendo la aplicación de una lógica y unos algoritmos determinados. Permite la automatización de procesos reduciendo la necesidad de intervención humana en gran medida. Añade adaptabilidad a los requerimientos y configuraciones de los usuarios. El software también proporciona eficiencia optimizando recursos. Un sistema con un software bien diseñado puede reducir costes y maximizar la productividad. En la Figura 4.15 se observa un diagrama del software que integra este proyecto.

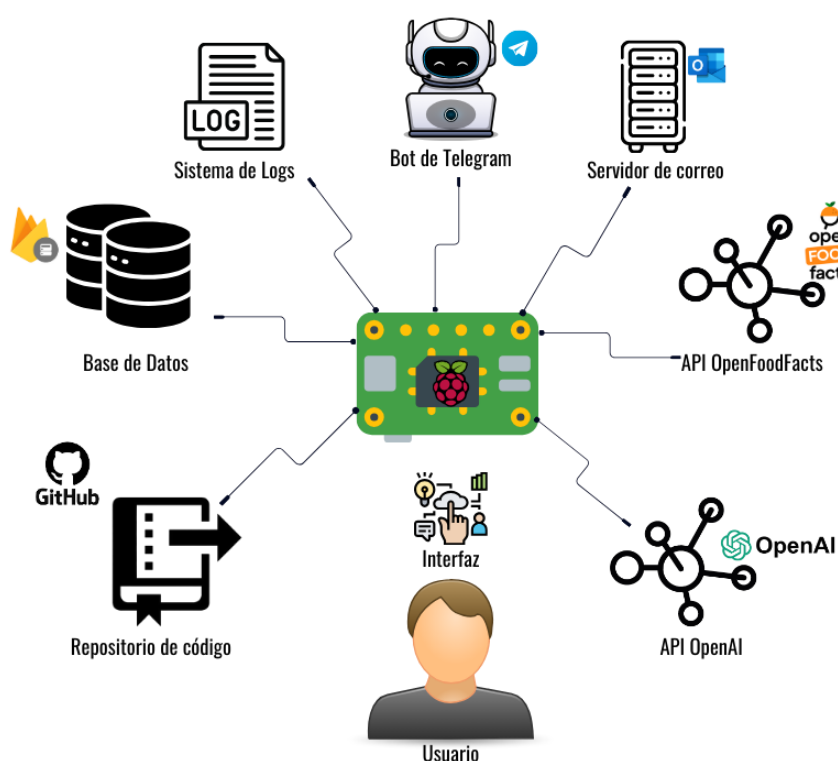


Figura 4.15: Diagrama del software del proyecto

A continuación, se presenta el software desarrollado, sus configuraciones y todas las funcionalidades que éste aporta al proyecto.

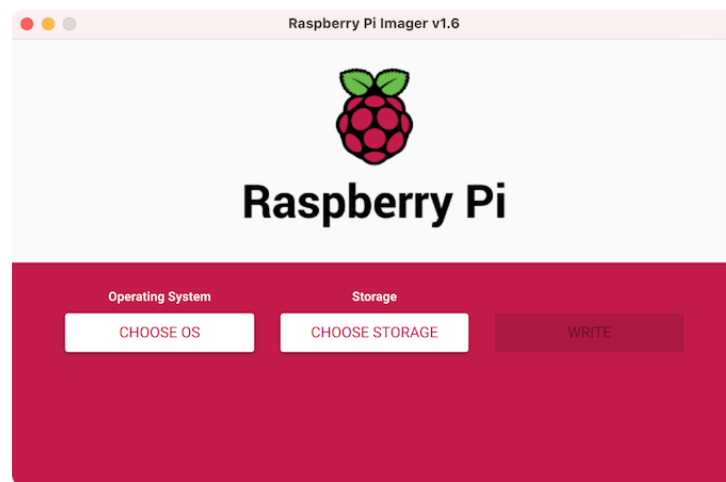
### 4.2.1 Sistema operativo

El conjunto de componentes hardware presentado anteriormente no tiene ninguna utilidad si no existe un software adaptado que coordine las funciones de cada componente y permita que trabajen en armonía hacia los objetivos del prototipo. Para esto es necesario la presencia de un sistema operativo. En este proyecto, el sistema operativo seleccionado es Raspberry Pi OS, antiguamente conocido como Raspbian.

## ***Raspberry Pi OS***

Es una distribución de Linux basado en Debian, lo que ofrece una base estable. Fue diseñada especialmente para la computadora de placa única empleada en este proyecto, la Raspberry Pi con todos sus modelos. El entorno que trae esta distribución facilita el desarrollo de nuevas aplicaciones y cuenta con una gran compatibilidad con periféricos y hardware externo.

Para su instalación y configuración en la Raspberry Pi 4 Model B empleada se ha hecho uso de la herramienta oficial de la marca; Raspberry Pi Imager (Figura 4.16), que permite grabar de forma rápida la imagen de sistemas operativos en una tarjeta microSD desde otro equipo. La versión empleada es Raspberry Pi OS 64-bit [9].



*Figura 4.16: Captura Raspberry Pi Imager*

### **4.2.2 Entorno de desarrollo**

Una vez instalado el sistema operativo en la placa, se ha configurado el entorno en el que se desarrollará la aplicación, se ha seleccionado el editor de código VSC (Visual Studio Code).

## ***Visual Studio Code***

VSC es un editor de código fuente desarrollado por Microsoft. Ofrece una amplia gama de características y extensiones que lo convierten en una herramienta muy potente para desarrollares. Entre sus características se pueden destacar la depuración integrada que permite rastrear y corregir errores y un terminal integrado que permite ejecutar comandos directamente desde el editor.

### **4.2.3 Repositorio de código**

Para un proceso de desarrollo de calidad es muy importante hacer uso de un entorno estructurado que permita llevar un seguimiento del código. Los repositorios de código añaden muchas ventajas a los proyectos.

A la hora de añadir nuevas versiones con nuevas funcionalidades es importante hacerlo de forma segura, sin permitir que ésta introduzca problemas en el funcionamiento general del proyecto, para ello, los repositorios incluyen características como un control de versiones o historial de cambios, registrando cada cambio y facilitando la identificación y reversión ante posibles modificaciones problemáticas o permitiendo comprender la evolución que sigue la aplicación.

Aunque para este proyecto, contando con un único desarrollador, no ha sido necesario la aplicación de estas características, los repositorios de código también aportan muchas ventajas a la hora de desarrollar en equipo, asegurando una colaboración efectiva.

En este proyecto, el repositorio de código empleado ha sido posiblemente el más conocido, GitHub.

## **GitHub**

GitHub es una plataforma de desarrollo colaborativo que permite alojar proyectos de software utilizando el sistema de control de versiones conocido como Git. GitHub permite mantener un repositorio en la nube con el código, pudiendo clonarlo en cualquier equipo y desde cualquier sitio siempre y cuando se tenga permisos para ello, lo que aporta seguridad al proyecto y la garantía de no perder los cambios siempre y cuando se mantenga el repositorio actualizado.

En este proyecto se ha usado GitHub por un único individuo, por lo que las características empleadas no han sido todas las que ofrece la plataforma.

Una vez enlazado el proyecto con el repositorio en GitHub, desde el terminal de Linux es sencillo realizar modificaciones en el repositorio por medio de determinados comandos, principalmente empleando el control de versiones Git mencionado anteriormente. La dirección del repositorio de GitHub empleado en el proyecto es <https://github.com/adrillb/iot-solution-tfg>.

Para añadir modificaciones a un proyecto, se trabaja en una rama específica, una rama Git es un puntero eficaz para las instantáneas de los cambios.

Para comprobar los cambios introducidos se hace uso del comando “**git status**”, que muestra los archivos modificados y los que han sido añadidos a un índice que subirán posteriormente. Si alguno de estos cambios realizados no ha sido añadido al índice, se debe usar el comando “**git add .**”, que añadirá todos los archivos modificados al índice. Cuando los cambios están listos y dentro de este índice, se debe llevar a cabo lo que se conoce como *commit* mediante el comando “**git commit**”, en este paso es muy importante añadir un comentario lo más descriptivo posible, será útil en el futuro. Es posible realizar varios commits en una misma rama. Cuando los cambios están listos para juntarse con el resto de código del repositorio, se emplea el comando “**git push**” especificando el nombre de la rama maestra y en la que se está trabajando. Tras esto se ha realizado un *merge*, es decir, las modificaciones han sido incluidas en el repositorio.

Existen casos en los que no interesa incluir determinados cambios en el repositorio general, ya sea porque son configuraciones propias para desarrollar en local o claves secretas que no interesa hacer públicas, para ello se hace uso del archivo “**.gitignore**” (Figura 4.17), que evita la subida de determinados archivos o carpetas al repositorio. En el caso de este proyecto se han incluido las siguientes líneas en este archivo.

```

adri11b@raspberrypi: ~/Desktop/TF6/Python/iot-solution
File Edit Tabs Help
adri11b@raspberrypi:~/Desktop/TF6/Python/iot-solution $ cat .gitignore
#ignore compiled python files & automatic generated files
*.pyc
__pycache__/_
*.egg-info/_
dist/_
build/_
*.egg

#ignore pictures
*.jpg
*.png
*.jpeg

#ignore logs
app.log

#ignore credentials
#Email Log info
userSettings.json
#DataBase access
FirebaseCredentials/_
#chatGPT api key
IARequest/API-KEY.json

```

*Figura 4.17: Archivo .gitignore*

En los siguientes apartados se hace referencia a estos casos específicos de archivos que no deben añadirse al repositorio.

#### 4.2.4 Base de datos

El objetivo de este proyecto reside en la capacidad de gestionar y mantener de alguna forma los productos y llevar un inventario sobre el que aplicar el resto de las funcionalidades, para esto es necesario el uso de una base de datos.

Como almacenar la información en una base de datos local carece de la eficiencia y usabilidad requeridas en la actualidad, se ha buscado una alternativa más efectiva. Con el fin de contar con la comodidad de tener acceso a esta información desde cualquier lugar y permitir un alcance global, se ha hecho uso de un modelo de base de datos en tiempo real alojado en la nube, en concreto, Firebase Realtime Database.

#### ***Firebase Realtime Database***

Firebase Realtime Database (Figura 4.18) es una base de datos NoSQL alojada en la nube en la que los datos se sincronizan en tiempo real con cada cliente manteniéndose disponibles cuando la aplicación no tiene conexión. Los archivos son almacenados en formato JSON, asociando una clave única a cada uno de los ítems.



*Figura 4.18: Logotipo Firebase*

En la Figura 4.19, se muestra una captura de ejemplo de la base de datos del proyecto:

```

1  {
2    "Database": {
3      "AlertDate": {
4        "Last_alerted_date": "08-09-2023"
5      },
6      "Products": {
7        "-NcsB7zzjWlj4maRD17i": {
8          "expiry_date": "06-11-2023",
9          "product_name": "Queso rallado"
10       },
11       "-NcsB81oMZetLr4XUkIw": {
12         "expiry_date": "12-05-2024",
13         "product_name": "Salsa de tomate"
14       },
15       "-NcsB84I01L0c7-C-OMw": {
16         "expiry_date": "01-08-2023",
17         "product_name": "Calabacín"
18       },
19       "-NdEu93VaT84q_R_H_dT": {
20         "expiry_date": "06-08-2023",
21         "product_name": "Huevos camperos"
22       },
23       "-NdKuDxNrTb391ew_KSm": {
24         "expiry_date": "06-09-2023",
25         "product_name": "Yogur natural"
26       }
27     }
28   }
29 }

```

Figura 4.19: Captura de ejemplo de base de datos del proyecto

Se observa la jerarquía y estructura de la base de datos. De una rama principal ‘Database’ cuelga una llamada ‘AlertDate’ para asegurar que no se envíe más de una alarma/notificación al usuario por día y otra llamada ‘Products’ de la que a su vez cuelgan todos los productos que se tengan almacenados. Cada uno de estos productos cuenta con una clave única que los identifica como se ha mencionado anteriormente y dentro, sus dos campos de fecha de caducidad y nombre.

#### 4.2.5 Conexión a la BBDD

En la aplicación existe una carpeta con las clases encargadas de gestionar la conexión con la base de datos y con todos los procesos CRUD (Create, Read, Update & Delete) que se lanzan a esta.

Para poder llevar a cabo la conexión entre la aplicación y la BBDD es necesario especificar la dirección de la base de datos y hacer uso de unas credenciales que garanticen su uso seguro, estas credenciales se guardan en un archivo JSON externo dentro del proyecto al que accede la aplicación cuando requiera comenzar una conexión. Este archivo JSON es uno de los que se introducen en el archivo .gitignore mencionado en el apartado anterior para garantizar la seguridad en el acceso y no compartir de forma pública las credenciales.

### 4.2.6 Lenguaje

Como lenguaje de programación en la aplicación desarrollada se ha seleccionado Python, un lenguaje poderoso y versátil, con un claro enfoque a la productividad. Es una opción muy conocida actualmente gracias a su sintaxis clara y legible, a ser multiplataforma y a su amplia biblioteca de librerías listas para usar.

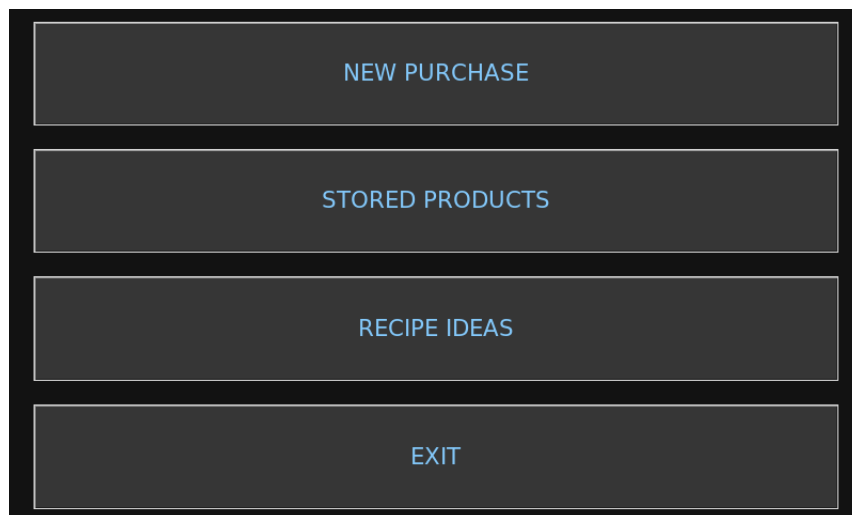
### 4.2.7 Interfaz

Para desarrollar la interfaz de la aplicación se ha utilizado una librería propia del lenguaje de programación Python llamada Tkinter. Esta librería está pensada para la creación y el desarrollo de aplicaciones de escritorio, facilita el posicionamiento y desarrollo de una interfaz gráfica.

Es una capa orientada a objetos basada en Tk, una herramienta GUI (Graphical User Interface) estándar. El montaje de una vista con Tkinter se basa en widgets jerarquizados, como raíz de la interfaz está 'tk' y sobre esta colocaremos estos widgets en función de nuestras necesidades.

Entre los widgets más utilizados se encuentran tk.Frame, tk.Label, tk.Button o tk.Text. Para cada uno de estos widgets podemos aplicar multitud de configuraciones, desde colores y tamaño hasta comandos a ejecutar tras su pulsado. También podemos gestionar la composición para determinar la posición y la relación entre widgets.

En la Figura 4.20 se muestra una captura de ejemplo de la interfaz desarrollada para esta aplicación, en este caso se trata de un menú de opciones en el que se emplean widgets de tipo tk.Button sobre un tk.Frame:



*Figura 4.20: Ejemplo de interfaz desarrollada*

Esta interfaz permite al usuario interactuar con la aplicación de forma sencilla e intuitiva, permitiendo hacer uso de sus funcionalidades.



### 4.2.8 Código y estructura

Para conseguir las diferentes funcionalidades de la aplicación, el código se divide en diferentes clases y carpetas, en la Figura 4.21 se muestra la estructura general de la aplicación para ir explicando su funcionamiento y la utilidad de cada uno de los archivos. Las capturas pertenecen al repositorio de GitHub por lo que los archivos incluidos en el .gitignore se explican aparte.

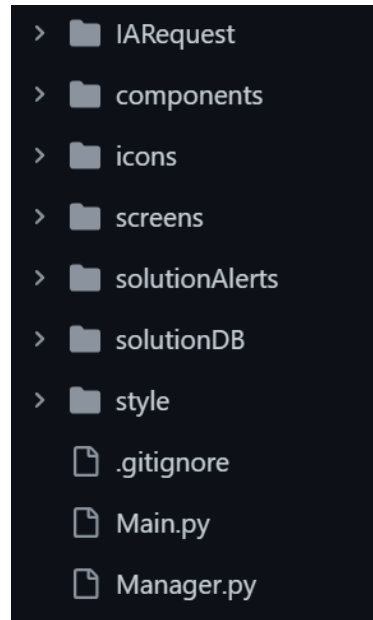


Figura 4.21: Estructura de la aplicación

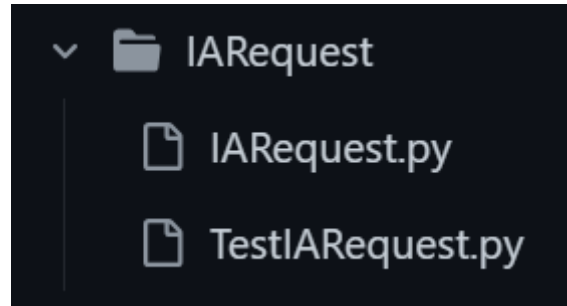
El archivo **.gitignore** ya se ha visto y explicado anteriormente en el apartado de GitHub.

Desde el **Main.py** se inicializa la ejecución de la aplicación, se comprueba si es necesario mandar alguna alerta y se crea una instancia de la clase Manager.

Desde **Manager.py** se inicializa y se configura la interfaz con todas sus pantallas y posiciones, se configura la instancia logger con la que guardar los logs de la aplicación (este logger se detallará más adelante), contiene métodos comunes como la validación de una fecha, la conversión a formato JSON o la comunicación con la clase encargada de la base de datos en el registro de nuevos productos.

## ***IAResult***

La carpeta **IAResult** (Figura 4.22) contiene archivos necesarios para cumplimentar la funcionalidad de “Recipe Ideas” y sus pruebas.



*Figura 4.22: Carpeta IAResult*

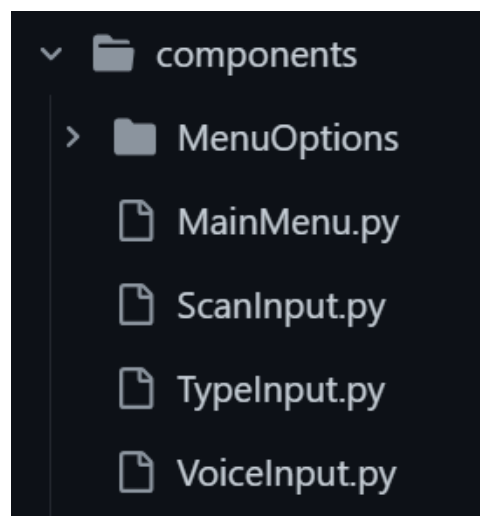
En el archivo **IAResult.py** se realiza la conexión a la API de OpenAI, para ello es necesario obtener la `apiKey` desde el fichero **API-KEY.json**, que se encuentra de forma local en el equipo que ejecuta la aplicación. Una vez obtenida puede lanzar peticiones a la API, este proceso se detalla en un apartado más adelante.

Básicamente lo que recibe esta clase es un prompt pidiendo un listado de recetas con productos seleccionados para poder generar esas “Recipe Ideas”.

**TestIAResult.py** está destinado a hacer estas peticiones directamente contra la API de OpenAI a modo de prueba, sin necesidad de lanzar toda la aplicación.

## ***Components***

A continuación, la carpeta **components** (Figura 4.23).

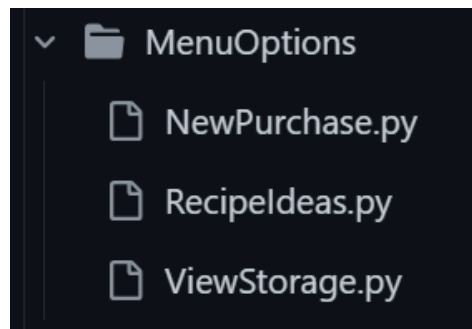


*Figura 4.23: Carpeta components*

Desde el archivo **MainMenu.py** se muestran todas las opciones del menú principal por medio de botones, se crean los diferentes tkButton con sus correspondientes configuraciones y estilos. Estas opciones son ‘NEW PURCHASE’, ‘STORED PRODUCTS’, ‘RECIPE IDEAS’ y ‘BACK’, para salir y cortar la ejecución de la aplicación.

## **MenuOptions**

La funcionalidad de las opciones principales del menú principal se encuentra en la carpeta MenuOptions (Figura 4.24).



*Figura 4.24: Carpeta components/MenuOptions*

Estos 3 archivos **NewPurchase.py**, **RecipeIdeas.py** y **ViewStorage.py** tienen una función muy sencilla, cada uno de ellos preconfigura las pantallas iniciales de la opción, aplicando estilos y posiciones.

Dentro de la primera opción de ‘NEW PURCHASE’ encontramos las siguientes opciones: Scan, Type y Voice. El resto de las opciones dentro del menú principal se detallan más adelante.

**ScanInput.py**, **TypeInput.py** y **VoiceInput.py** contienen los 3 métodos de registro de productos con los que cuenta la aplicación. Comparten estructuras muy similares, muestran su interfaz con los estilos correspondientes permitiendo el registro de productos en el sistema mediante diferentes métodos. Solicitan, de la forma en que corresponda en cada caso, el nombre del producto y, cuando lo han obtenido, se pregunta si se desea también introducir la fecha de caducidad, comprobando que se trate de una fecha válida en caso de seleccionar introducirla.

A medida que se van introduciendo productos con cualquiera de los métodos se va rellendo una lista de productos, cuando se acepta el proceso y con esta información, continúa la ejecución para registrar el o los productos en la base de datos. La interfaz de la aplicación se va modificando en función del punto en el que se encuentre el proceso para que el usuario conozca la forma en la que continuar.

## **ScanInput**

Este método escanea un código de barras y consulta información sobre él. Gracias al comando `libcamera-still` con un temporizador captura una imagen y la guarda como archivo, el cual consulta posteriormente para decodificar el número del código de barras con la librería `pyzbar`. Cuando tiene el número, lanza una consulta a la API de `OpenFoodFacts` y selecciona, entre la gran variedad de campos que esta API puede devolver (valores nutricionales, marca, ingredientes, etc.), únicamente el nombre del producto.

### ***TypeInput***

Este método es el más sencillo de los 3, solicita al usuario teclear el nombre del producto por medio del widget `tk.Entry`, éste se emplea también en caso de querer introducir también la fecha de caducidad del producto en cualquiera de los 3 métodos posibles.

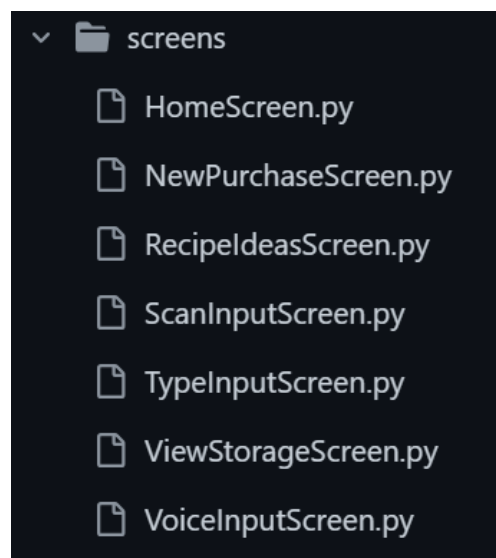
### ***VoiceInput***

Este método, posiblemente es el más cómodo de usar, solicita al usuario decir en voz alta el nombre del producto y captura el sonido a través del micrófono. Después, gracias a la librería `speech_recognition` identifica el nombre mencionado. Cuenta con un `timeout` en caso de no reconocer el producto, mostrando un mensaje de informativo en este caso.

Existen diferentes reconocedores posibles, tras varias pruebas se ha observado que el más eficaz es el de Google, aunque necesita de una muy buena conexión a Internet para poder reconocer correctamente el audio.

### ***Screens***

Las diferentes pantallas de la aplicación son preconfiguradas en los archivos ubicados en la carpeta `screens` (Figura 4.25), inicializando su título, sus botones y demás widgets con los estilos correspondientes.



*Figura 4.25: Carpeta screens*

### ***SolutionAlerts***

La carpeta `solutionAlerts` (Figura 4.26) contiene los archivos necesarios para cumplimentar con la funcionalidad de envío de correos electrónicos y mensajes con un bot de Telegram a los diferentes usuarios notificando de la proximidad de fecha de caducidad de sus productos.

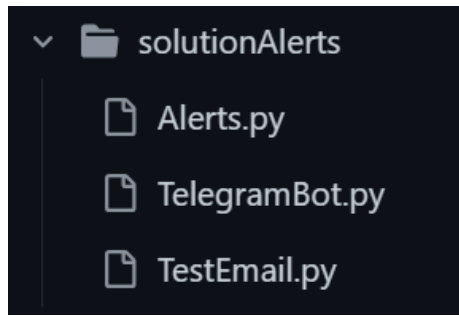


Figura 4.26: Carpeta *solutionAlerts*

Desde **Alerts.py**, antes de nada, se comprueba la fecha de la última alerta enviada, almacenada en la base de datos. Si ésta coincide con la fecha actual, termina su ejecución, de lo contrario, modifica la fecha almacenada sustituyéndola por la actual y continúa la ejecución como se explica a continuación. De este modo se evita enviar más de una alarma por día, ya que, en principio, aportarían la misma información.

Después se recorren todos los productos almacenados en el sistema comparando la fecha actual con la de caducidad de cada uno de ellos, comprobando varios casos posibles. Para que se envíe una alerta es necesario que se cumpla uno de estos casos:

- La fecha de caducidad ya ha pasado.
- La fecha de caducidad coincide con la fecha actual.
- Queda un día para que el producto caduque.
- Quedan dos días para que el producto caduque.

Para cada uno de estos casos se manda una notificación al usuario con un mensaje distinto, especificando el caso del que se trata.

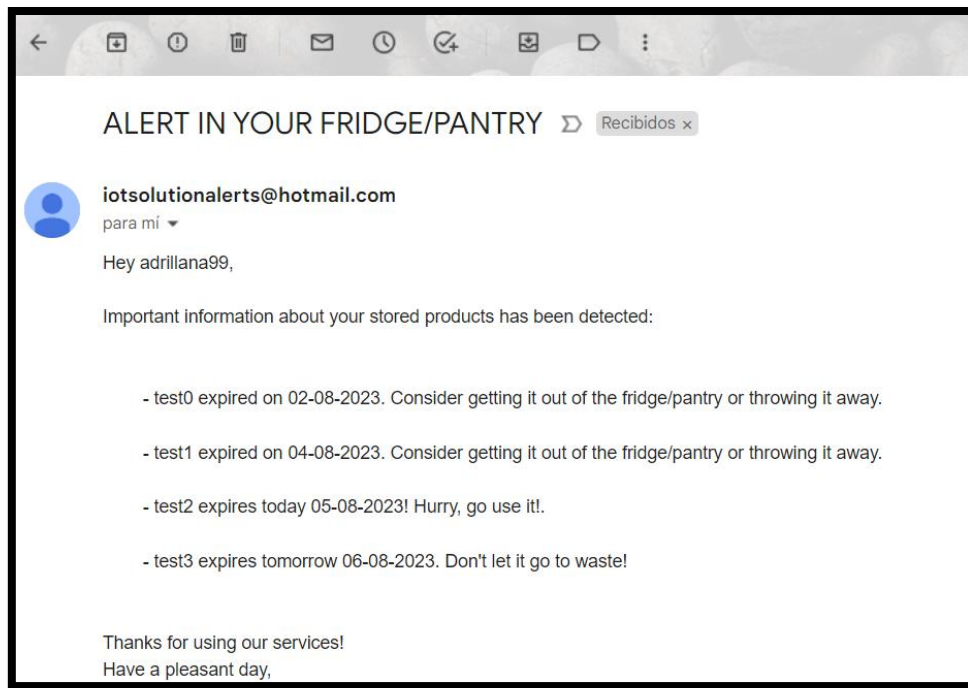
La aplicación cuenta con dos métodos a la hora de enviar estas notificaciones a los usuarios.

Para mandar **correos electrónicos** se ha creado una dirección específica para este proyecto, [iotsolutionalerts@hotmail.com](mailto:iotsolutionalerts@hotmail.com), tanto esta dirección como su contraseña están almacenadas en otro de los archivos incluidos en el ya mencionado archivo `.gitignore`, en este caso se trata de **userSettings.json**, guardado de forma local en esta misma carpeta del equipo que ejecuta la aplicación. Éste también contiene el listado de correos electrónicos destinatarios, que recibirán la alerta enviada.

También se decodifica el archivo JSON para obtener las direcciones de correo correspondientes y la contraseña del emisor, cuando tiene esta información y haciendo uso de las librerías `smtplib` y `email.mime`, establece estos valores y el servidor de correo a emplear, en este caso `'smtp-mail.outlook.com'` en el puerto 587.

Con todo esto y el mensaje ya creado, procede a enviar los correos electrónicos a cada una de las direcciones especificadas personalizando el mensaje en función del destinatario (un saludo con el nombre de la dirección de correo electrónico hasta el carácter '@').

En la Figura 4.27 se muestra un ejemplo de correo enviado con los diferentes tipos de mensaje posibles.



*Figura 4.27: Ejemplo correo de alerta enviado*

También se ha creado un **bot de Telegram** (@IotAlerts\_bot) mediante el cual el usuario puede recibir las notificaciones de alerta. Una vez creado el mensaje, se hace uso del archivo **TelegramBot.py**. Desde este archivo se decodifica el archivo JSON **userSettingsTelegram.json** (también incluido en .gitignore) para obtener el TOKEN único que identifica al bot y un listado de CHAT\_IDs, los cuales corresponden a los usuarios que recibirán la notificación. Con esta información, se recorre el listado de IDs y se procede a cada uno de los envíos. Para todo esto se ha instalado la biblioteca python-telegram-bot.

En la Figura 4.28 se observa un ejemplo de alerta que se envía a través de este bot.

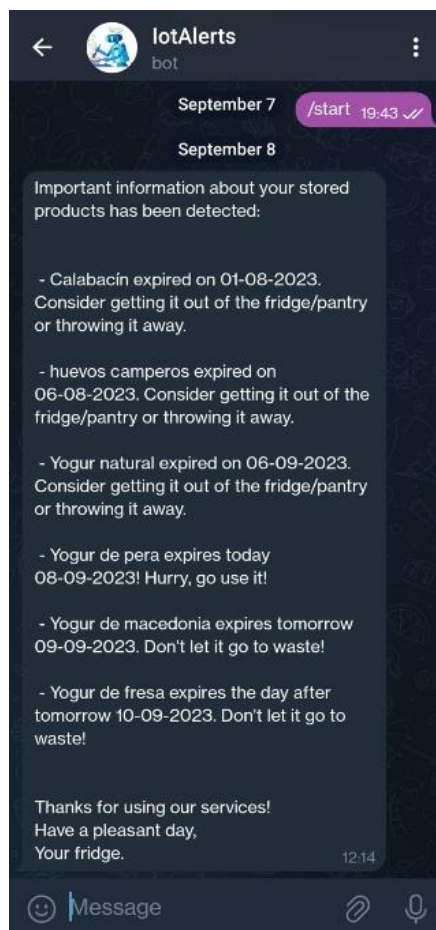


Figura 4.28: Captura bot de Telegram

Desde **TestEmail.py** se han realizado pruebas de estas funcionalidades con ejemplos sin necesidad de ejecutar la aplicación completa.

### ***SolutionDB***

La carpeta **solutionAlerts** (Figura 4.29) contiene los archivos necesarios para cumplimentar con la funcionalidad de conectarse a la base de datos y realizar las operaciones CRUD desde la aplicación. Estas funcionalidades se implementan en **DataBase.py** y se han explicado previamente de forma detalla en el apartado ‘Conexión a la BBDD’.

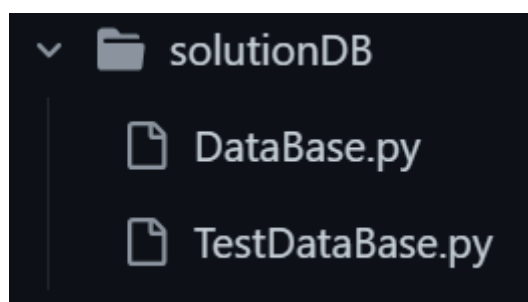


Figura 4.29: Carpeta **solutionDB**

Desde **TestDataBase.py** han realizado pruebas de esta funcionalidad sin necesidad de ejecutar la aplicación completa.

## Style

Para gestionar los estilos desde una única ubicación se hace uso de la carpeta style (Figura 4.30).

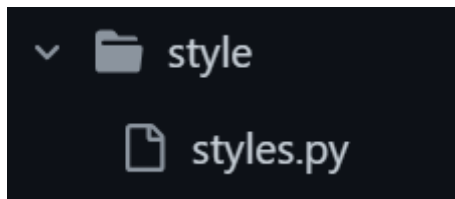


Figura 4.30: Carpeta styles

El uso de **styles.py** ayuda a centralizar la aplicación de estilos a los elementos de la interfaz, ya sea tamaño color, posición y resto de configuraciones de estilo. En él se definen configuraciones comunes a toda la aplicación.

Para el desarrollo de la interfaz se ha intentado mantener un estilo minimalista para facilitar su uso y con pequeños detalles diferenciadores. Por ejemplo, el color del fondo y el principal de los componentes es común en toda la aplicación, pero cada una de las 3 opciones principales y todo su interior, se configura un color del texto específico como se muestra en las Figuras 4.31 y 4.32, como aclaración de esta última Figura, ‘NP’ se refiere a New Purchase, ‘ST’ hace referencia a Stored Products y ‘RI’ a Recipe Ideas.

```

1
2     import tkinter as tk
3
4
5     BACKGROUND = "#121212"
6     COMPONENT = "#363636"
7
8
9     TEXT = "#84C9FB"
10    TEXT_NP = "#3DD697"
11    TEXT_ST = "#7D3C98"
12    TEXT_RI = "#FF8A33"
13
14    FONT = ("Trip Sans", 16)
15    CURSOR = "cross"
16    RELIEF = "groove"

```

Figura 4.31: Configuraciones comunes



```

STYLE_NP = {
    "bg" : COMPONENT,
    "fg" : TEXT_NP,
    "font" : FONT,
    "cursor" : CURSOR
}

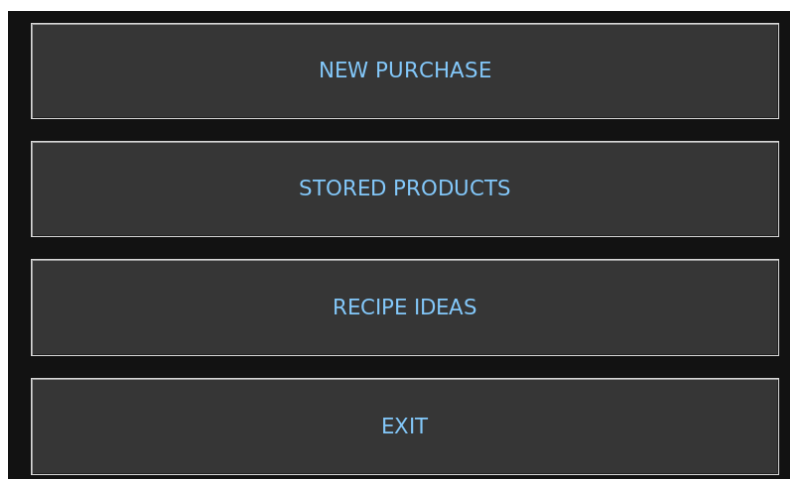
STYLE_ST = {
    "bg" : COMPONENT,
    "fg" : TEXT_ST,
    "font" : FONT,
    "cursor" : CURSOR
}

STYLE_RI = {
    "bg" : COMPONENT,
    "fg" : TEXT_RI,
    "font" : FONT,
    "cursor" : CURSOR
}

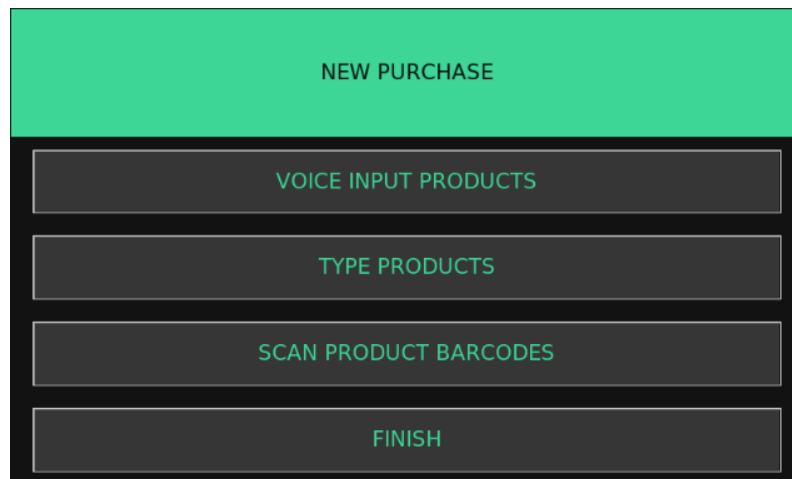
```

*Figura 4.32: Configuraciones según apartado de la aplicación*

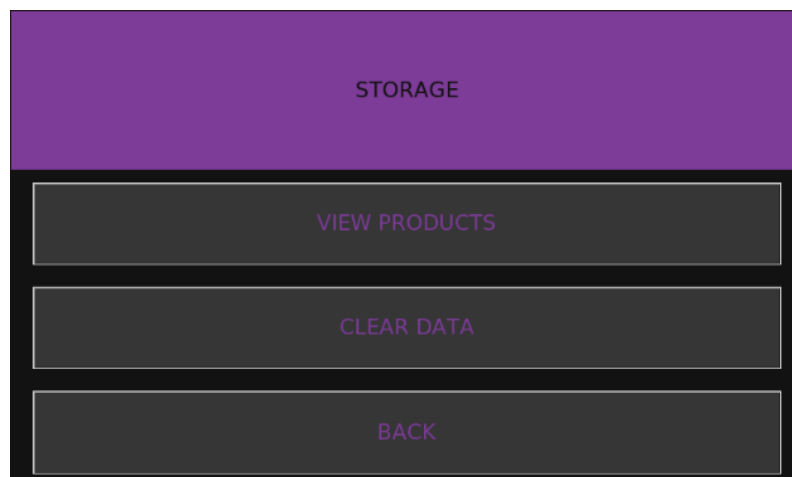
Para comprobar la aplicación de estas configuraciones, se ven las imágenes de muestra en las Figuras 4.33, 4.34, 4.35 y 4.36.



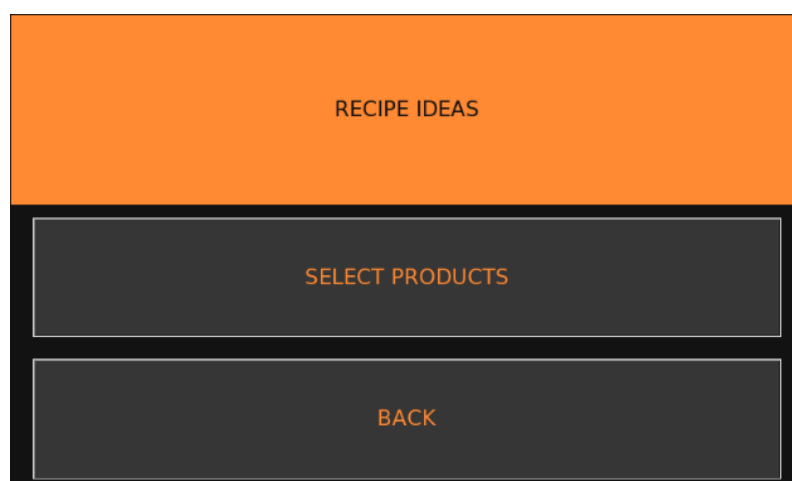
*Figura 4.33: Estilos aplicados en el menú principal*



*Figura 4.34: Estilos aplicados 'NEW PURCHASE'*



*Figura 4.35: Estilos aplicados 'STORAGE PRODUCTS'*



*Figura 4.36: Estilos aplicados 'RECIPE IDEAS'*

### 4.2.9 Servicios externos

Los servicios externos enriquecen las aplicaciones, permiten una mayor innovación y expanden sus funcionalidades.

En este proyecto, para conseguir que la aplicación implemente todas las funcionalidades propuestas, ha sido necesario hacer uso de varios de estos servicios externos con los que comunicarse y trabajar.

Se han utilizado dos APIs (Application Programming Interface), una base de datos alojada en la nube, un bot de Telegram creado desde cero y un servidor de correo con el que enviar correos electrónicos desde la aplicación.

A continuación, se detallan cada uno de estos servicios externos.

#### ***OpenAI API***

Este proyecto hace uso de la API de OpenAI (Figura 4.37), que permite hacer uso del chatbot de inteligencia artificial desarrollado en 2022 especializado en diálogo, ChatGPT.

Para generar una petición a esta API es necesario especificar el texto o prompt de consulta, el engine que debe emplear y el máximo de tokens que acepta. Esta petición equivale a una interacción con chatGPT 3.

Para poder hacer uso de esta API también es necesario crear una cuenta y generar una apiKey asociada, cada petición tiene un coste de \$0,002 por cada 1,000 tokens, pero en los primeros meses OpenAI te da un crédito de \$5 para poder hacer pruebas, esto ha sido lo empleado en este proyecto.



*Figura 4.37: Logo OpenAI*

## ***OpenFoodFacts API***

OpenFoodFacts (Figura 4.38) es una base de datos abierta y colaborativa de productos alimentarios de todo el mundo que cuenta con unos 300.000 registros, conteniendo información de cada producto como valores nutricionales, ingredientes, marca, cantidad, tipo de envase, etiquetas certificadas, países de venta y muchas otras cosas más, aunque en la actualidad de este proyecto solo se hace uso del nombre. [10]



*Figura 4.38: Logo OpenFoodFacts*

La petición realizada a esta API tiene el formato:

“<https://world.openfoodfacts.org/api/v0/product/>” + *[productID]*, sustituyendo *[productID]* por el código de barras reconocido por la aplicación. [11]

## ***BBDD en la nube***

La base de datos alojada en la nube empleada y ya explicada con detalle anteriormente también es un servicio externo del que hace uso la aplicación. Su dirección es la siguiente: <https://iot-solution-8d63c-default-rtdb.europe-west1.firebaseio.com/>

## ***Bot de Telegram***

La creación de un bot de Telegram no es un proceso demasiado complicado, únicamente es necesario tener un usuario registrado en la plataforma de mensajería. Cada bot tiene un token y cada usuario un chat\_id únicos, con estos dos valores ha sido posible utilizar la aplicación Python del proyecto para gestionar el envío de alertas/notificaciones a través del bot creado. En la Figura 4.39 se muestra la información del bot empleado en este proyecto, siendo su enlace de invitación el siguiente: [http://t.me/IotAlerts\\_bot](http://t.me/IotAlerts_bot)



Figura 4.39: Información del bot de Telegram

## SMTP Outlook Server

Como ya se ha explicado en el apartado de alertas, la aplicación hace uso de un servidor de correo electrónico smtp, este también es un servicio externo.

### 4.2.10 Uso de Logs

Los logs o registros son ‘trazas’ que va dejando la ejecución de un programa en forma de textos o mensajes, pueden ser de diferentes niveles en función de lo que se quiera registrar: errores, warnings, información, etc.

Su uso en una aplicación es muy importante ya que permite analizar y explicar el comportamiento que se ha seguido durante la ejecución. Es una forma de encontrar problemas de una manera más directa.

En este proyecto se ha implementado el uso de logs en puntos específicamente seleccionados para evaluar el comportamiento general del sistema y detectar errores de funcionamiento. Estos logs se almacenan en otro de los archivos incluidos en el .gitignore denominado app.log.

En este archivo se escriben líneas con el siguiente formato:

*“Fecha y hora - Clase ejecutora - Acción realizada”.*

A continuación, se muestra en la Figura 4.40 un ejemplo de logs de una ejecución en la que se probaba el borrado de todos los productos almacenados en el sistema a la vez.

```
2023-08-05 11:54:12,844 - Manager - Raise <class 'screens.HomeScreen.HomeScreen'>
2023-08-05 11:54:12,848 - Manager - Solution Starting ...
2023-08-05 11:54:14,943 - Manager - Raise <class 'screens.ViewStorageScreen.ViewStorageScreen'>
2023-08-05 11:54:23,894 - Manager - Deleting all products.
2023-08-05 11:54:24,145 - Manager - test0 - 02-08-2023 deleted
2023-08-05 11:54:24,306 - Manager - test1 - 04-08-2023 deleted
2023-08-05 11:54:24,547 - Manager - test2 - 05-08-2023 deleted
2023-08-05 11:54:24,756 - Manager - test3 - 06-08-2023 deleted
2023-08-05 11:54:31,694 - Manager - Solution Closing ...
```

Figura 4.40: Ejemplo de logs registrados tras una ejecución

## **5. Conclusiones**

Como conclusión final de este proyecto, se puede afirmar que se han alcanzado los objetivos inicialmente establecidos. Para ello se han seguido tanto los caminos y estrategias que se plantearon durante el análisis del proyecto como otros nuevos que han surgido a lo largo del desarrollo del mismo.

Se ha desarrollado un prototipo capaz de convertir un frigorífico común en uno con algunas de las funciones propias de uno inteligente. El principal valor añadido de este proyecto radica en la asequibilidad que ofrece a los usuarios. Los frigoríficos inteligentes que se comercializan en la actualidad tienen un coste demasiado elevado y suelen ser descartados por la mayoría del público por este motivo.

Gracias a su previsible bajo coste y compatibilidad universal con cualquier frigorífico, es posible llevar las ventajas que ofrece uno inteligente a un público más amplio.

Además, este proyecto ha sido desarrollado para ser fácilmente escalable, lo que permite la adición de nuevas funcionalidades y mejora de componentes, se detallan estas y otras ideas en el siguiente apartado de Trabajo futuro.

## **6. Trabajo futuro**

La tecnología nunca ha sido un campo estático, siempre está en constante avance, por ello, en este apartado se exploran diferentes posibilidades que permitan expandir y optimizar las funcionalidades de este proyecto.

### **6.1 Aprovechamiento de servicios externos**

Como ya se ha mencionado, el proyecto hace uso de servicios externos para cumplimentar algunas de sus funcionalidades y algunos de estos servicios ofrecen muchas más posibilidades de las que se hacen uso. Se proponen dos ejemplos:

- La API de OpenFoodFacts, aparte de ofrecer el nombre del producto del que se escanea el código de barras, permite obtener una gran cantidad de información acerca del producto; información nutricional, ingredientes, alérgenos, materiales de embalaje, certificados de medio ambiente, etc. Toda esta información puede ser muy útil a la hora de seguir dietas específicas para intolerantes, deportistas o para mejorar el estado de salud de cualquier persona.
- La base de datos de este proyecto solo está siendo empleada para guardar los productos con sus fechas de caducidad y la fecha de la última alerta enviada por el sistema. En esta misma base de datos y aprovechando su ya implementada comunicación con el prototipo se podrían registrar muchos más datos a medida que se hace uso de la aplicación. Por ejemplo, se podrían también almacenar los productos ya consumidos o desperdiciados para posteriormente realizar estudios de consumo.

### **6.2 Funcionalidades a largo plazo**

Con la posible comercialización a gran escala del prototipo se abriría la posibilidad de desarrollar funcionalidades algo más complejas pero muy interesantes.

Si el número de usuarios ascendiera de forma considerable, juntando la gran cantidad de datos que éstos generarían mediante el uso de la aplicación, sería posible desarrollar algoritmos y establecer patrones de conducta a la hora de consumir y reponer alimentos. Gracias a la inteligencia artificial y el machine learning, con todos estos datos se podría automatizar de una forma mucho más eficiente las compras, su frecuencia y cantidades.

### **6.3 Configuraciones del sistema**

Un usuario que haga uso de la aplicación no tiene acceso directo a configuraciones del sistema tales como el listado de direcciones de correo electrónico o usuarios de Telegram a los que se le envían las alertas. Se podría crear un apartado en la aplicación desde donde gestionar esta información y, por ejemplo, la frecuencia y tipo de alertas que envía el sistema.

## 6.4 Mejora de componentes

Perfectamente se puede considerar a este prototipo como un sistema empotrado. A la hora de desarrollador este tipo de sistemas es muy importante, en un análisis previo, la selección de los componentes que lo integrarán.

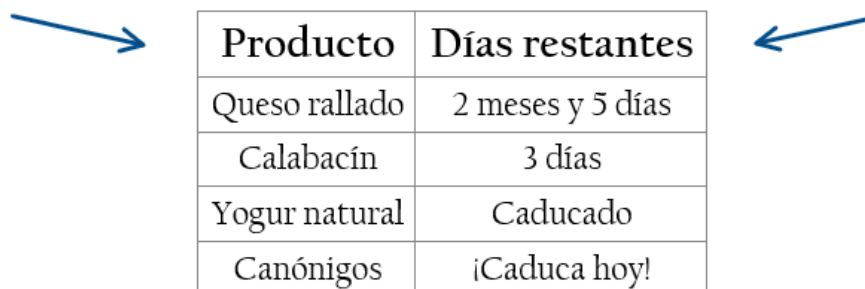
La elección de componentes de propósito específico, es decir, optimizados para una única función, aporta un mejor rendimiento frente a los componentes de propósito general, que son capaces de realizar un mayor número de tareas, pero con peores resultados.

Los componentes empleados en este prototipo han sido en su mayoría, de propósito general, pero si se planea la fabricación de un gran número de unidades, a menudo es aconsejable decantarse por la integración de componentes de propósito específico.

## 6.5 Mejoras en la interfaz

La interfaz de este proyecto ha sido desarrollada pensando en su usabilidad, pero existen varios aspectos en la parte visual que podrían añadirse para ofrecer una experiencia al usuario más satisfactoria y clara. Como ejemplos para ello se propone:

- A la hora de mostrar los productos presentes en el sistema, distinguir en columnas la información mostrada. Por ejemplo ‘nombre de producto’ y ‘días restantes’ (mostrando el número de días restantes para alcanzar la fecha de caducidad), de forma similar a como se muestra en la Figura 6.1. De esta forma se aporta más información al usuario mientras hace uso de la aplicación.



Producto	Días restantes
Queso rallado	2 meses y 5 días
Calabacín	3 días
Yogur natural	Caducado
Canónigos	¡Caduca hoy!

*Figura 6.1: Aproximación mejora tabla de productos*



- El uso de iconos en determinados botones o imágenes a lo largo de la ejecución de la aplicación. Un ejemplo de icono del que podría hacerse uso se muestra en la Figura 6.2, en sustitución del botón de borrado de un producto en la sección de ‘Stored Products’.



Figura 6.2: Icono ‘borrar’

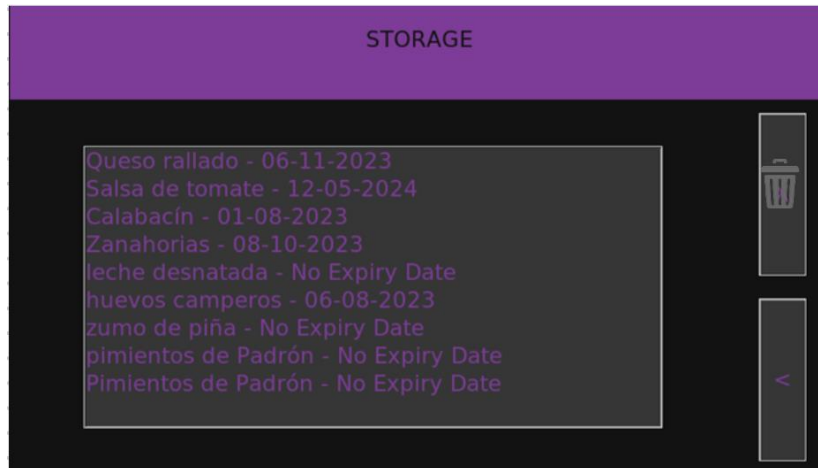


Figura 6.3: Resultado de ejemplo con icono

## Bibliografía

[1] Plataforma técnica sobre la medición y la reducción de las pérdidas y el desperdicio de alimentos

<https://www.fao.org/platform-food-loss-waste/es>

[2] Datos del desperdicio alimentario en hogares

<https://www.mapa.gob.es/es/alimentacion/temas/desperdicio/desperdicio-alimentario-hogares/>

[3] El frigorífico inteligente, con Internet, música y televisión, centro de la nueva cocina

[https://elpais.com/diario/2002/11/07/ciberpais/1036640138\\_850215.html?event\\_log=oklogin](https://elpais.com/diario/2002/11/07/ciberpais/1036640138_850215.html?event_log=oklogin)

[4] 1998: Internet fridges are "looming on the horizon"

<https://www.businessinsider.com/the-complete-history-of-internet-fridges-and-connected-refrigerators-2016-1#1998-internet-fridges-are-looming-on-the-horizon-1>

[5] Fridges to be hit by Net viruses

[https://www.theregister.com/2000/06/21/fridges\\_to\\_be\\_hit\\_by/](https://www.theregister.com/2000/06/21/fridges_to_be_hit_by/)

[6] Frigorífico Side by Side Family Hub Grafito Clasificación Energética F RS68N8941B1

<https://www.samsung.com/es/refrigerators/side-by-side/593l-metal-graphite-rs68n8941b1-ef/>

[7] Raspberry Pi 4 Tech Specs

<https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/>

[8] La PWM: ¿Qué es? ¿Cómo puedo utilizarla?

<https://www.digikey.es/es/blog/pulse-width-modulation>

[9] Raspberry Pi OS

<https://www.raspberrypi.com/software/>

[10] Yuka - Help

<https://help.yuka.io/l/es/article/5a4z64amnk-constitucion-base-de-datos>

[11] GitHub – Repositorio OpenFoodFacts

<https://github.com/openfoodfacts/openfoodfacts-laravel>

[12] Repositorio de código desarrollado

<https://github.com/adriillb/iot-solution-tfg>

[13] Bot de Telegram creado

[http://t.me/IotAlerts\\_bot](http://t.me/IotAlerts_bot)

[14] Database empleada

<https://iot-solution-8d63c-default-rtdb.europe-west1.firebaseio.com/app/>

**Anexos**

# 1. Anexo I – Coste del proyecto

En este apartado se estudian los diferentes costes que ha conllevado el desarrollo de este proyecto.

## 1.1 Coste de material

En primer lugar, se detalla el coste de material (Tabla 3) de este proyecto, que se segmenta en dos apartados, los costes de cada uno de los componentes hardware que conforman el prototipo y los costes del software desarrollado que implementa. El coste de Hardware se muestra en la Tabla 1 y el coste de Software en la Tabla 2.

DESCRIPCIÓN	NÚMERO	COSTE UNIDAD	TOTAL
Raspberry Pi 4 Model B	1	85€	85€
Pantalla	1	43,95€	43,95€
Módulo de cámara	1	34,70€	34,70€
Micrófono USB	1	10,10€	10,10€
Controlador a distancia	1	23,99€	23,99€
Disipador de calor	1	11,99€	11,99€
<b>TOTAL</b>			<b>209,03€</b>

Tabla 1. Coste Hardware

DESCRIPCIÓN	NÚMERO	COSTE UNIDAD	TOTAL
Raspberry Pi OS	1	0€	0€
GitHub	1	0€	0€
Visual Studio Code	1	0€	0€
Python			
Realtime Firebase Database	1	0€	0€
Servidor de correo Outlook	1	0€	0€
API OpenAI	1	0€	0€
API OpenFoodFacts	1	0€	0€
VIM	1	0€	0€
JSON file format	1	0€	0€
<b>TOTAL</b>			<b>0€</b>

Tabla 2. Coste Software

DESCRIPCIÓN	TOTAL
Coste Hardware	209,03€
Coste Software	0€
<b>Total</b>	<b>209,03€</b>

Tabla 3. Coste material

## 1.2 Coste de personal

A continuación, se muestra el coste de personal desglosado en diferentes tareas y las horas que ha supuesto cada una, se visualiza en la Tabla 4,

CONCEPTO	HORAS	COSTE/HORA	COSTE TOTAL
Análisis de requisitos	20	30€	600€
Diseño del prototipo	10	30€	300€
Integración/montaje del prototipo	5	30€	150€
Diseño del software a implementar	10	30€	300€
Desarrollo de interfaz de usuario	15	30€	450€
Desarrollo métodos de registro de productos	75	30€	2.250€
Conexión y gestión de la BBDD	10	30€	300€
Desarrollo sistema de alertas	15	30€	450€
Desarrollo sistema de logs	5	30€	150€
Gestión de llamadas a APIs	20	30€	600€
Creación y gestión del repositorio de código	10	30€	300€
Pruebas y validación	15	30€	450€
Desarrollo de documentación	70	30€	2.100€
Gestión del proyecto	10	30€	300€
<b>Total</b>	<b>290</b>		<b>8700€</b>

Tabla 4. Coste de personal

### 1.3 Costes generales

Los costes generales de este proyecto se detallan en la Tabla 5. En estos costes se incluyen gastos variables como equipos informáticos, electricidad, Internet, material de oficina, desplazamientos, etc. Se estima esta cantidad en un 15% de los costes personal y material del proyecto.

Costes generales	1336,36€
------------------	----------

*Tabla 5. Costes Generales*

### 1.4 Coste total

El coste total del proyecto se muestra y detalla en la Tabla 6.

<b>DESCRIPCIÓN</b>	<b>TOTAL</b>
Costes Material	209,03€
Costes Personal	8700€
Costes generales	1336,36€
<b>TOTAL</b>	<b>10245,39€</b>

*Tabla 6. Coste total*

El coste total del proyecto asciende a la cantidad de DIEZ MIL DOSCIENTOS CUARENTA Y CINCO CON TREINTA Y NUEVE CÉNTIMOS.

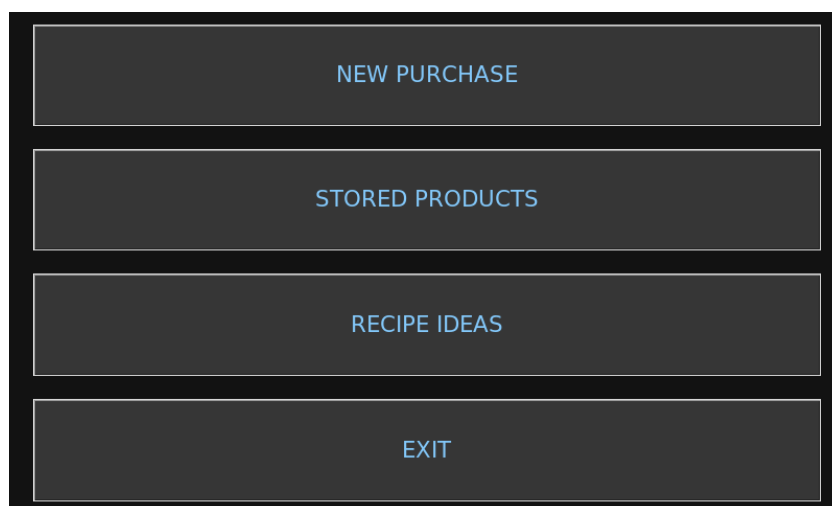
## 2. Anexo II - Manual de usuario

Aunque se trata de una aplicación sencilla, se incluye un manual de usuario para que todos los pasos y funcionalidades estén claro antes de comenzar a usarlo. Para navegar por la aplicación podremos hacer uso de la pantalla táctil o del controlador a distancia, el cual incluye un teclado con multitud de opciones y un panel táctil a modo de ratón.

En este manual se analizan todas las pantallas de la interfaz y opciones posibles en cada una de ellas.

### 2.1 Pantalla principal

La pantalla principal de la aplicación se muestra en la Figura 8.1.



*Figura 2.1: Pantalla principal de la aplicación*

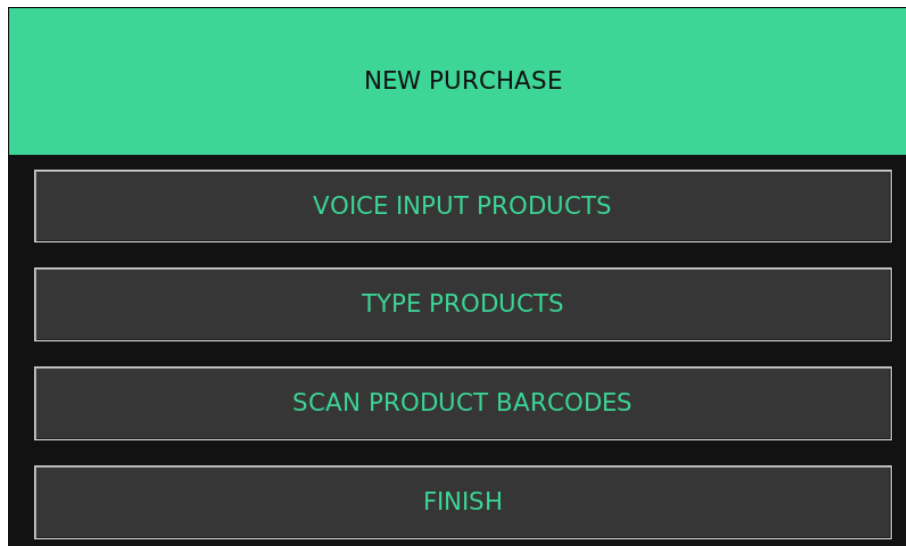
Desde esta pantalla el usuario tiene la posibilidad de acceder a tres opciones diferenciadas o cerrar la aplicación. Estas tres primeras opciones son:

- **NEW PURCHASE** – Desde este apartado se añaden productos al sistema tras cada compra, insertando nombre de producto y, de forma voluntaria, fecha de caducidad. Para ello la aplicación ofrece tres métodos de registro diferenciados, se detallan más adelante. Desde este apartado también se podrá retroceder a la pantalla principal.
- **STORED PRODUCTS** – En esta sección, el usuario puede visualizar todos los productos presentes en el sistema, con su fecha de caducidad si disponen de ella. En este apartado también es posible eliminar productos de forma individual uno a uno o si se prefiere, en su totalidad. Desde este apartado también se podrá retroceder a la pantalla principal.
- **RECIPE IDEAS** – En esta opción, el usuario puede seleccionar un listado de productos dentro de los que se encuentren presentes en el sistema y, cuando haya finalizado la selección, pedir 5 ideas de recetas que incorporen los elementos seleccionados. Desde este apartado también se podrá retroceder a la pantalla principal.

La elección del botón **EXIT**, terminará la ejecución de la aplicación.

## 2.2 NEW PURCHASE

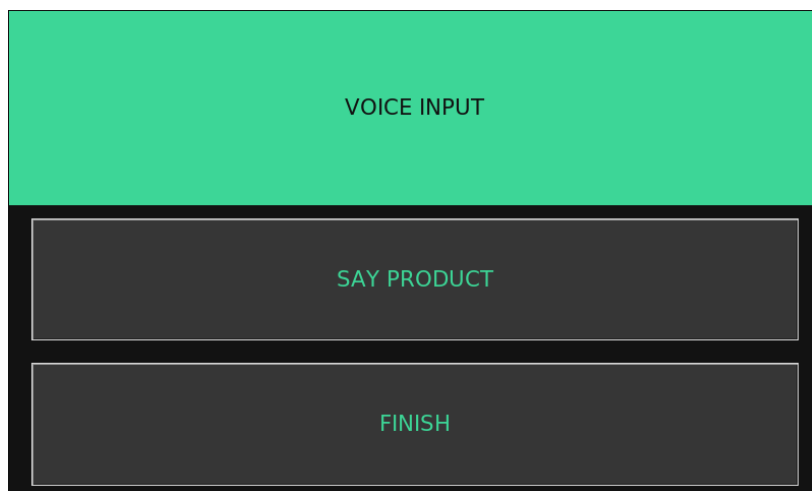
La pantalla principal de esta opción muestra las siguientes opciones como métodos de registro de productos en el sistema (Figura 8.2).



*Figura 2.2: Pantalla principal de 'NEW PURCHASE'*

Al pulsar sobre la opción **FINISH**, la aplicación volverá al menú principal.

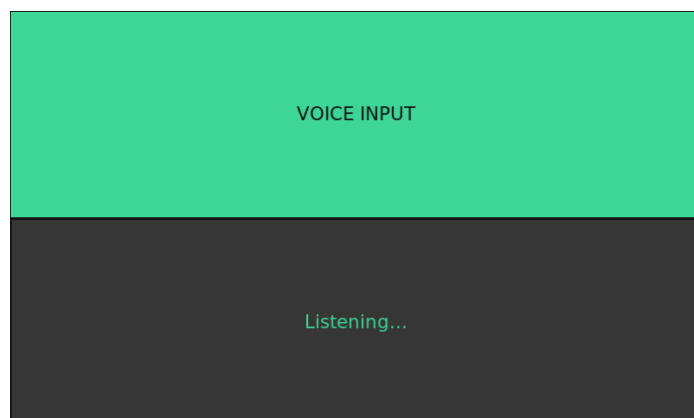
**Voice Input Products** – Al pulsar esta opción la aplicación se prepara para la inserción de productos mediante reconocimiento de voz y muestra la pantalla de la Figura 8.3.



*Figura 2.3: Preparación para el reconocimiento de voz*

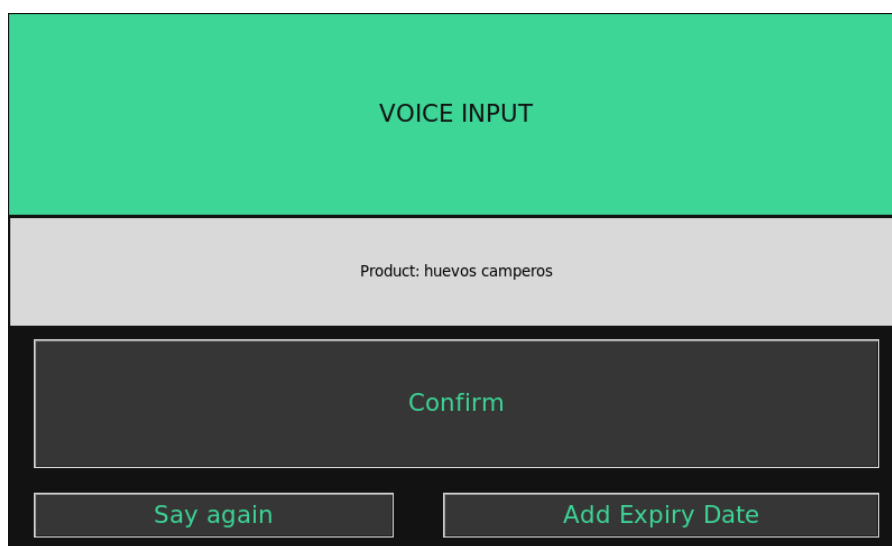


Cuando el usuario está listo, pulsa en la opción **SAY PRODUCT** y la aplicación muestra la pantalla de la Figura 8.4.



*Figura 2.4: Pantalla mientras la aplicación escucha*

Mientras se muestra esta pantalla, la aplicación está en modo de escucha hasta que detecte y reconozca el producto mencionado, cuando lo haga mostrará la pantalla de confirmación de la Figura 8.5.



*Figura 2.5: Pantalla de confirmación de reconocimiento de voz*

Desde aquí se presentan tres opciones posibles:

- **Say again**, que reiniciará el proceso de escucha en caso de que lo que la aplicación haya reconocido sea erróneo o el usuario desee repetirlo.
- **Confirm**, que añadirá el producto a la lista de productos con una fecha de caducidad por defecto '00-00-0000' y se continuará el proceso de registro de productos.
- **Add Expiry Date**, que parará a mostrar la pantalla de la Figura 8.6 en la que introducir la fecha de caducidad del producto:

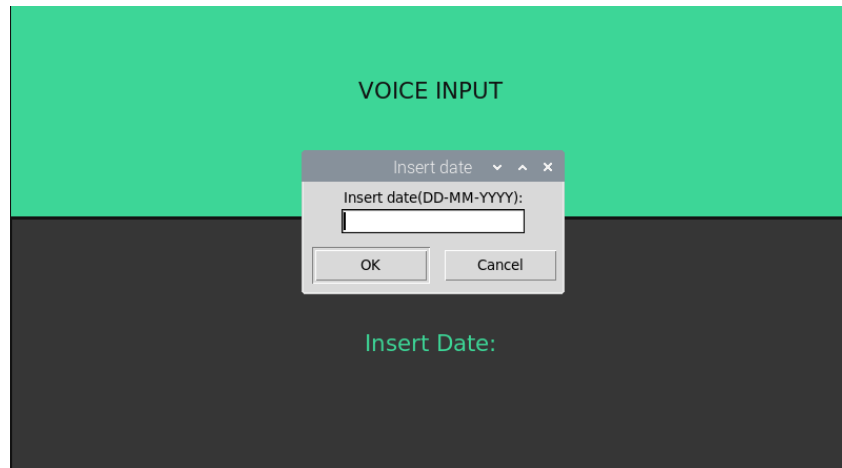


Figura 2.6: Pantalla insertar fecha

La aplicación validará la fecha introducida, por lo que en caso de introducir una fecha incorrecta como la mostrada en la Figura 8.7 se mostrará un mensaje de error (Figura 8.8), solicitando que se introduzca una fecha válida:

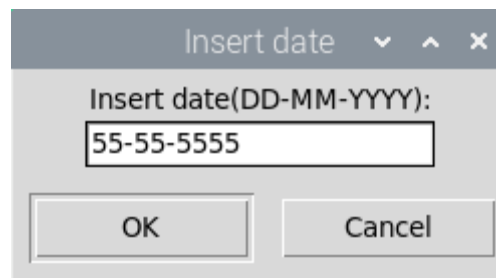


Figura 2.7: Ejemplo fecha incorrecta

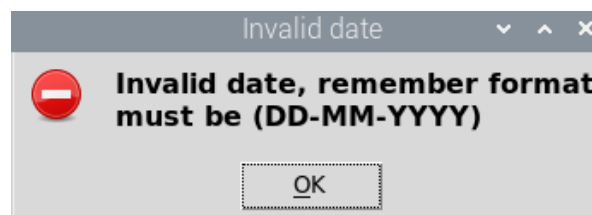


Figura 2.8: Mensaje fecha incorrecta

Cuando se introduzca una fecha válida, se añadirá el producto con su fecha de caducidad a la lista de productos y continuará el proceso de registro.

Si la aplicación no es capaz de reconocer ningún resultado, procede a mostrar el mensaje de aviso de la Figura 8.9.

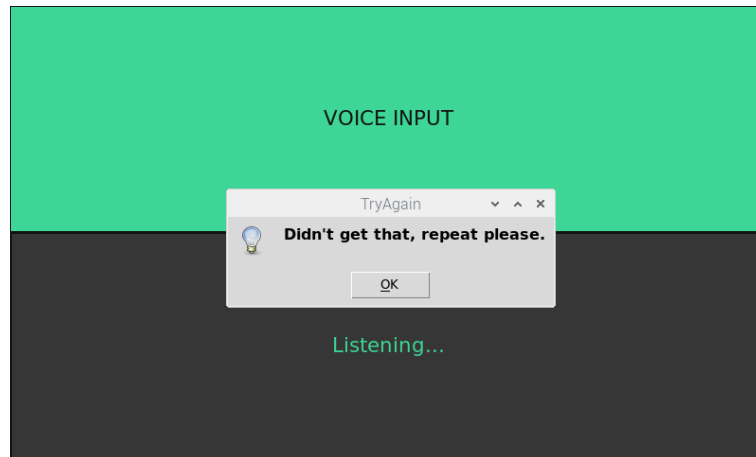


Figura 2.9: Mensaje fallo en el reconocimiento de voz

En el momento que el usuario finalice la inserción de productos debe pulsa la opción **FINISH**, en este momento la aplicación registrará la información introducida en la base de datos y mostrará el siguiente mensaje informativo (Figura 8.10).

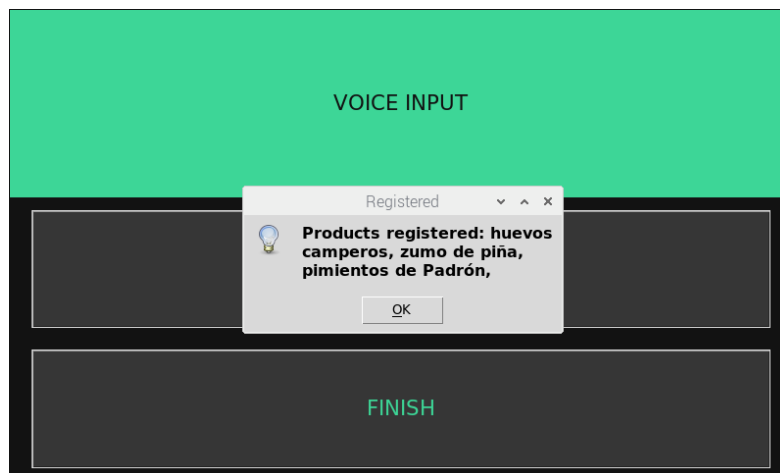
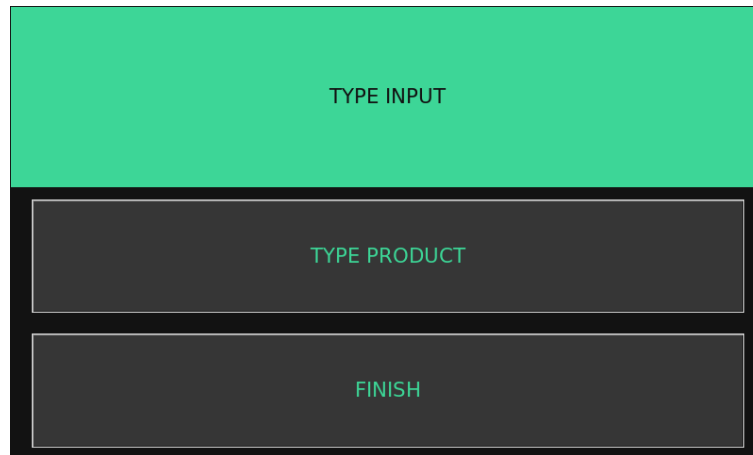


Figura 2.10: Mensaje productos registrados

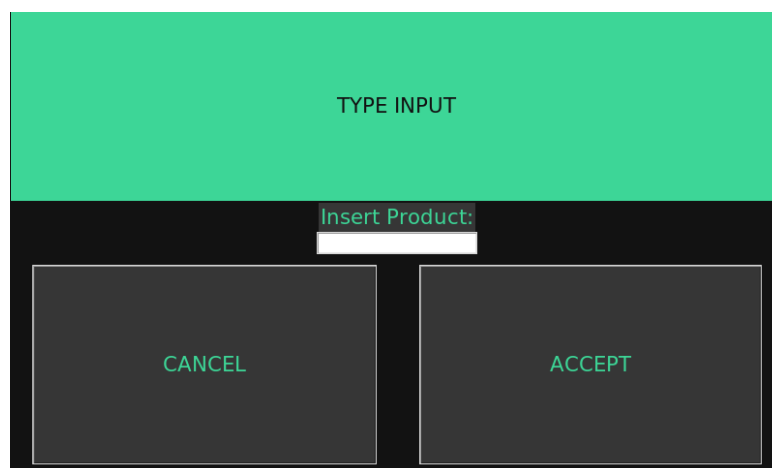
Tras esto, se volverá al menú principal de la aplicación, también si no se ha introducido ningún producto.

**Type Products** – Al pulsar esta opción, la aplicación se prepara para la inserción de productos mediante el teclado y mostrará la pantalla de la Figura 8.11.



*Figura 2.11: Preparación para la inserción por teclado*

Cuando el usuario pulsa **TYPE PRODUCT**, se muestra la siguiente pantalla (Figura 8.12) y el sistema está listo para registrar productos.



*Figura 2.12: Inserción por teclado*

Una vez introducido el nombre del producto, se muestra la siguiente pantalla (Figura 8.13) con diferentes opciones donde escoger.

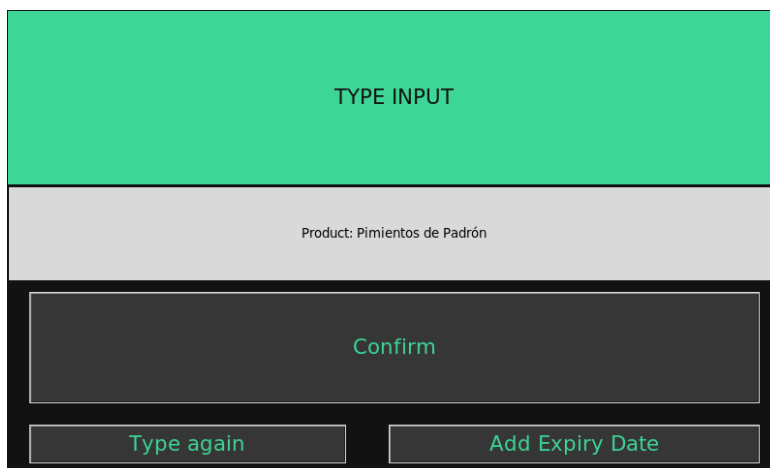


Figura 2.13: Pantalla de confirmación entrada teclado

En esta pantalla el usuario puede seleccionar **Type again** para reiniciar el proceso y volver a introducir el nombre del producto. La opción **Confirm** añade el producto a la lista de productos y continúa el proceso de inserción. La opción **Add Expiry Date** inicia el proceso de la misma forma que en el primer método de reconocimiento de voz.

Al igual que en el método anterior, cuando se finalice el proceso, la aplicación registrará en la base de datos la información introducida por el usuario.

**Scan Product Barcodes** – Al pulsar esta opción, la aplicación se prepara para la inserción de productos mediante el escaneo de sus códigos de barras y mostrará la siguiente pantalla (Figura 8.14).

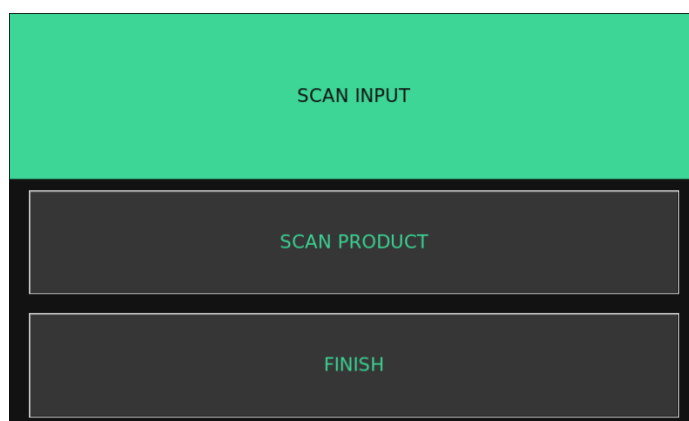
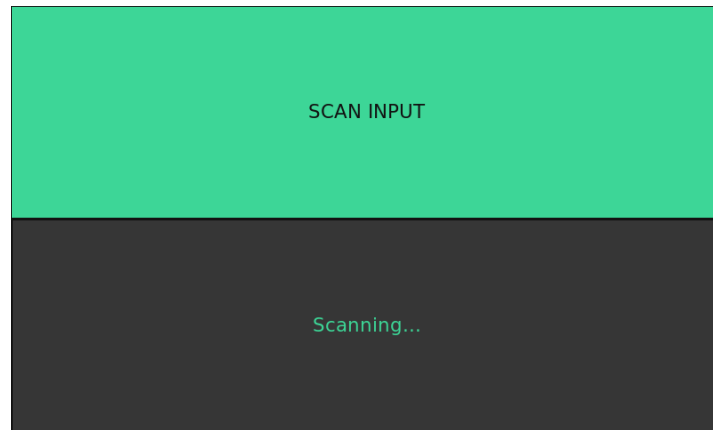


Figura 2.14: Preparación para el escaneo de productos

Cuando el usuario está listo, pulsa en la opción **SCAN PRODUCT** y la aplicación muestra la siguiente pantalla (Figura 8.15).



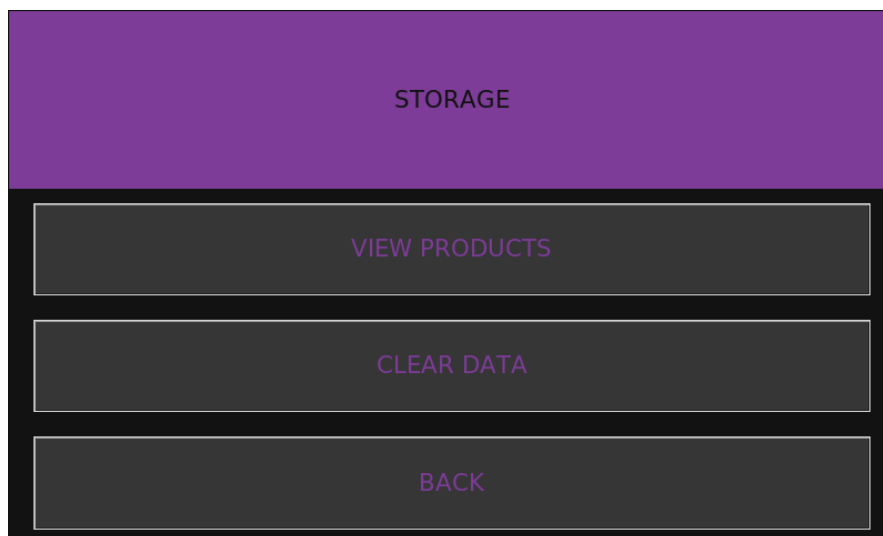
*Figura 2.15: Pantalla mientras la aplicación escanea*

Mientras se muestra esta pantalla, la aplicación está en modo de escaneo hasta que detecte y reconozca un código de barras.

Al igual que en los métodos anteriores, en este punto el usuario puede tomar 3 caminos, **Scan again** para reiniciar el proceso, **Confirm** para añadir a la lista de productos o **Add Expiry Date** para registrar el producto con fecha de caducidad.

## 2.3 STORED PRODUCTS

La pantalla principal de esta sección (Figura 8.16) muestra tres opciones, visualizar los productos almacenados en el sistema, borrar todos los productos o retroceder al menú principal de la aplicación:



*Figura 2.16: Pantalla principal de 'STORED PRODUCTS'*

Si el usuario selecciona la opción **View Products** se mostrará la siguiente pantalla (Figura 8.17).

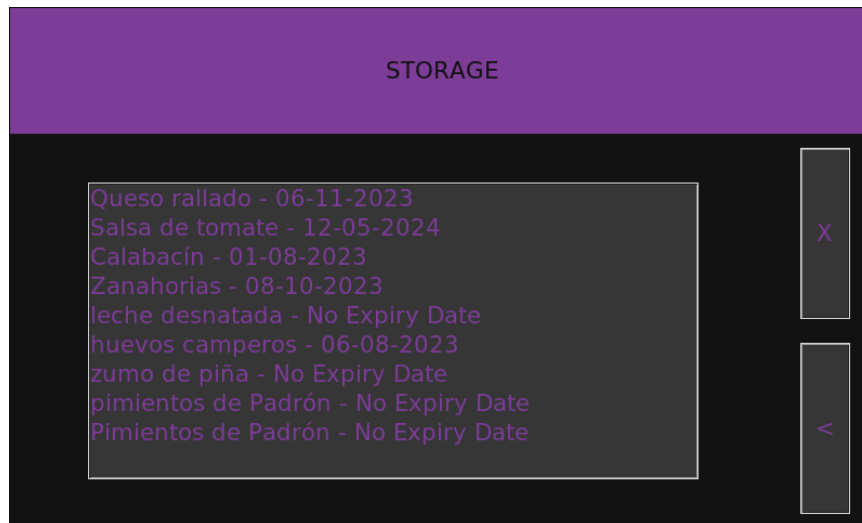


Figura 2.17: Visualización de productos almacenados

En esta pantalla se visualizan los productos almacenados en el sistema con su fecha de caducidad correspondiente, junto a este listado hay dos botones, el de abajo '<' retrocede al paso anterior, la pantalla principal de la esta sección.

El botón de arriba **X** tiene la función de eliminar un producto de forma individual de la base de datos, para ello, el usuario debe seleccionar uno de los productos en el listado como aparece en la Figura 8.18.

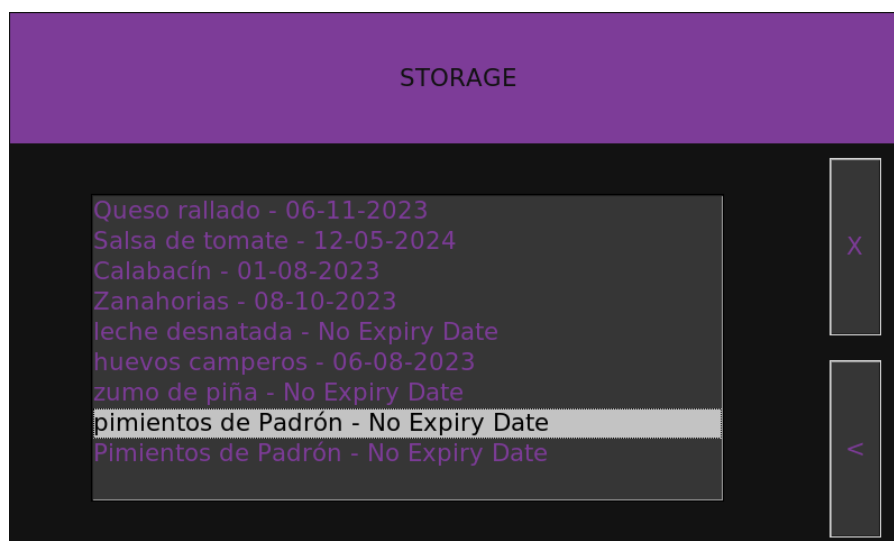


Figura 2.18: Selección de un producto

Una vez seleccionado, pulsar en dicho botón, provocando la eliminación del producto de la base de datos y la visualización del siguiente mensaje (Figura 8.19).

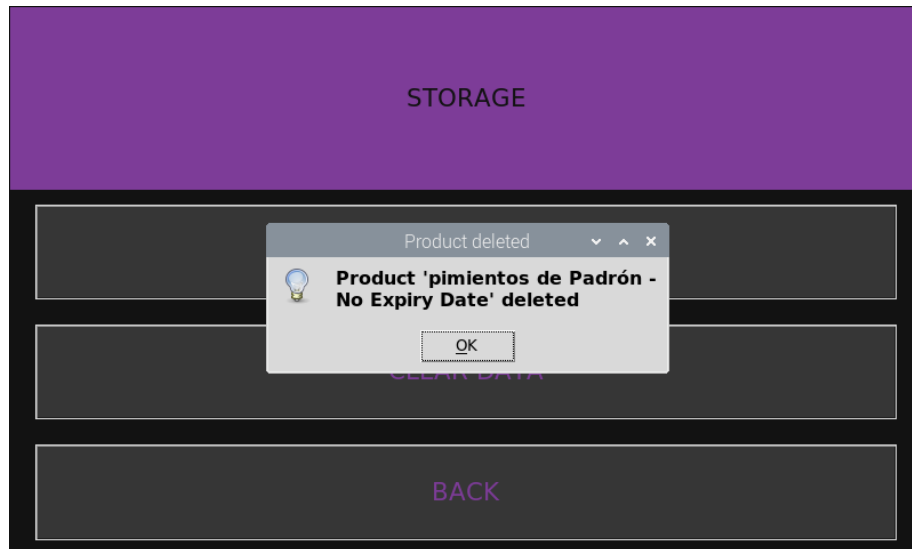


Figura 2.19: Mensaje informativo de borrado de producto

Por otro lado, si el usuario selecciona la opción de **Clear Data**, la aplicación advierte de que se si se acepta se borrará todos los productos del sistema, sin posibilidad de recuperarlos, mostrando el siguiente mensaje (Figura 8.20).

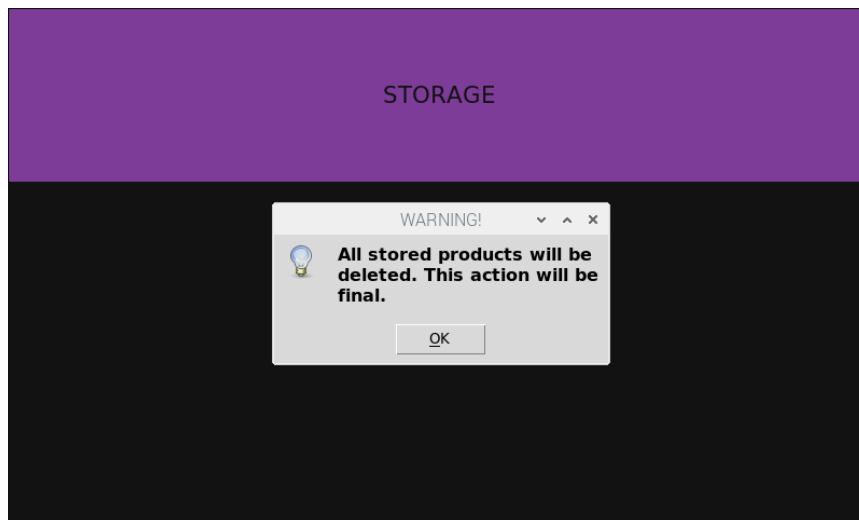


Figura 2.20: Mensaje de advertencia antes de borrar todos los productos

Tras ese mensaje de advertencia se pasa a la siguiente pantalla, donde el usuario decide si continuar con el proceso de borrado o volver a la pantalla principal de la sección (Figura 8.21).



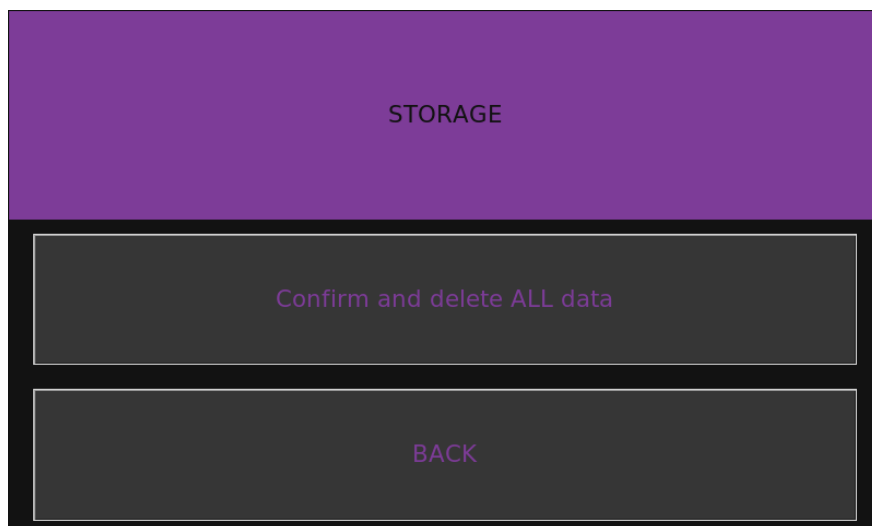


Figura 2.21: Pantalla para eliminar todos los productos

## 2.4 RECIPE IDEAS

La pantalla principal de esta sección (Figura 8.22) muestra dos opciones, retroceder al menú inicial de la aplicación o pasar a la siguiente pantalla en la que obtener ideas de recetas:

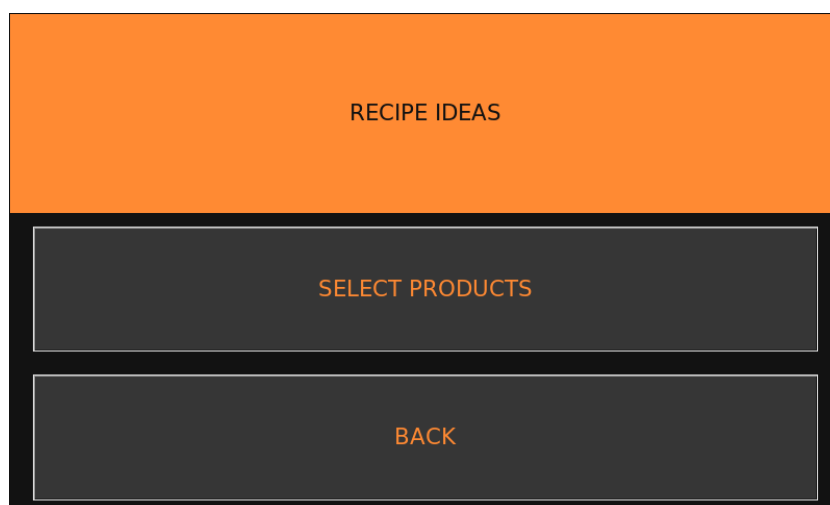


Figura 2.22: Pantalla principal de 'RECIPE IDEAS'

Cuando el usuario selecciona **Select Products**, la aplicación pasa a mostrar, de manera similar a la sección anterior, el listado de productos almacenados en el sistema. De este listado puede seleccionar una sublista con la que pretenda preparar alguna comida como se muestra en la Figura 8.23.

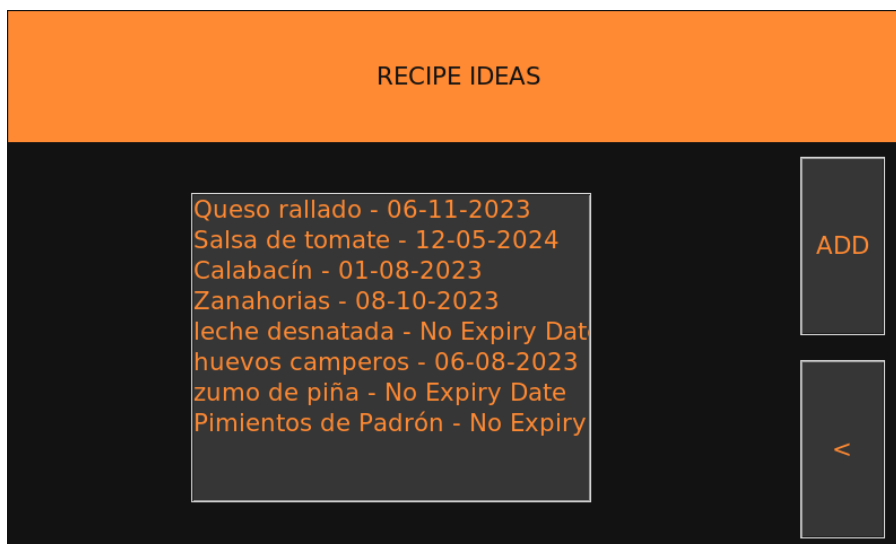


Figura 2.23: Visualización listado productos

La selección de cada producto pasa por primero hacer click sobre el producto que se desee y después sobre el botón **ADD**, dando como resultado la siguiente captura (Figura 8.24):

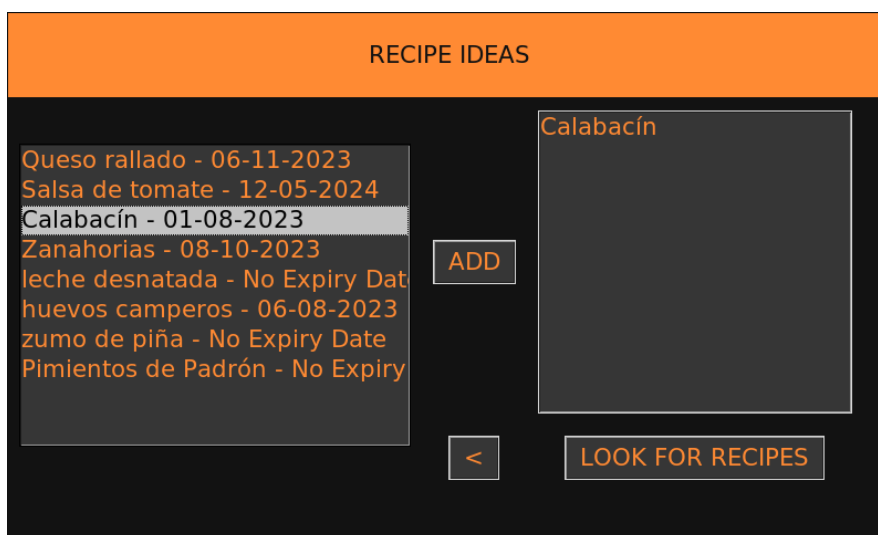


Figura 2.24: Modo de selección

Se observa cómo aparece un nuevo botón. Tras repetir este proceso para todos los productos deseados y finalizar la selección, el usuario selecciona **LOOK FOR RECIPES**, obteniendo un listado de cinco recetas posibles con los productos seleccionados. Se muestra a continuación el resultado para la siguiente selección de productos (Figura 8.24).

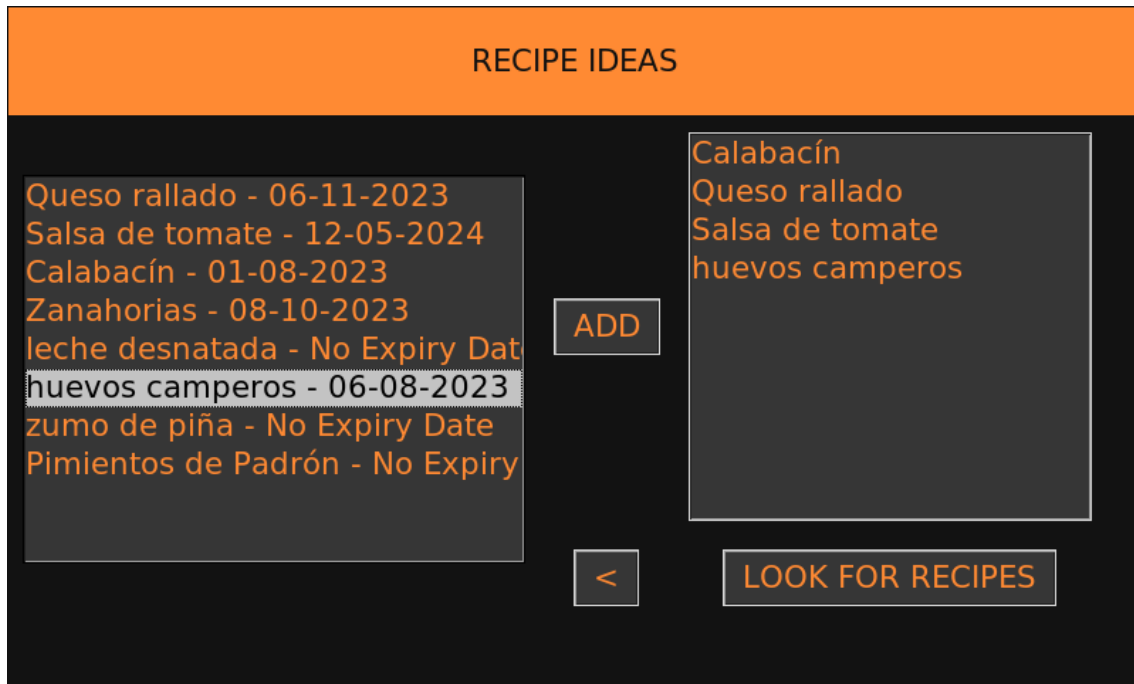


Figura 2.25: Selección listado de productos para ideas de recetas

Se muestran las ideas de recetas generadas en dos capturas (Figuras 8.25 y 8.26), para poder visualizarlas el usuario tiene que hacer scroll.

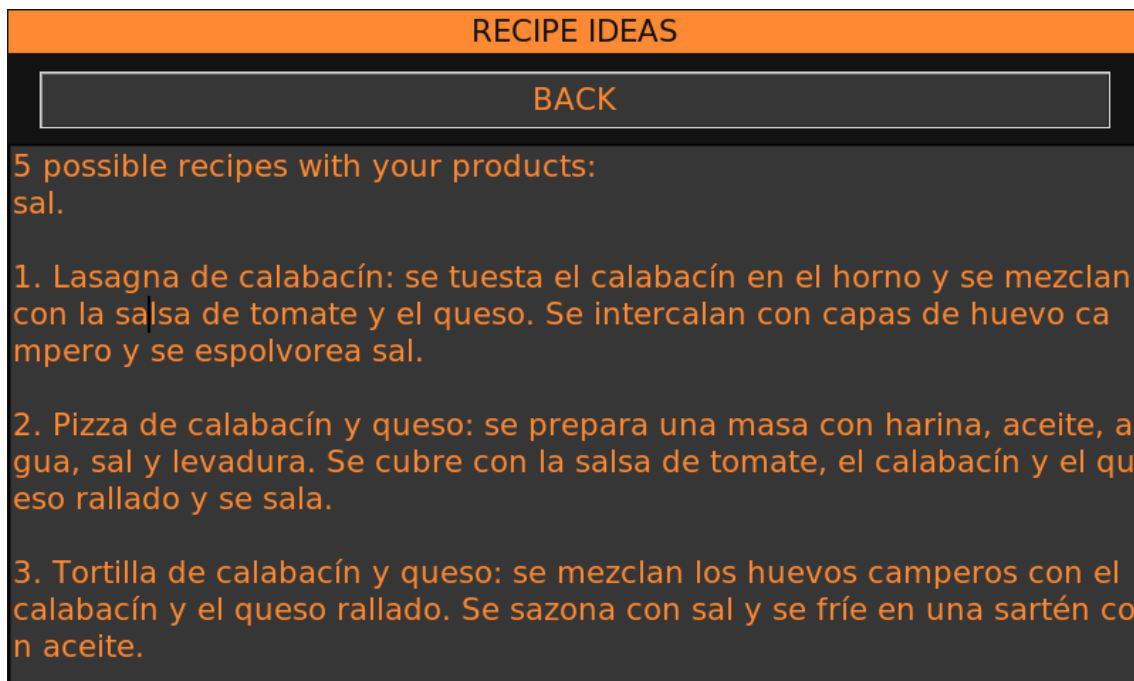


Figura 2.26: Resultado ideas de recetas 1/2

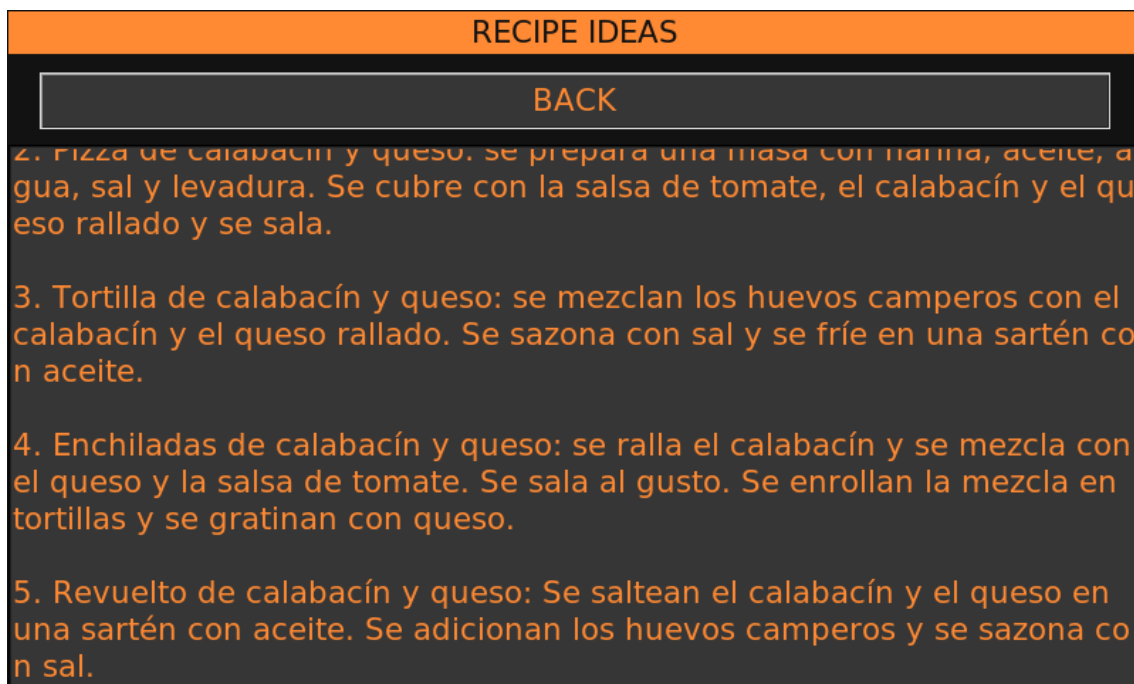


Figura 2.27: Resultado ideas de recetas 2/2

Universidad de Alcalá  
Escuela Politécnica Superior



ESCUELA POLITECNICA  
SUPERIOR



Universidad  
de Alcalá