Universidad de Alcalá
Escuela Politécnica Superior

Computer Engineering Bachelor's Degree

Bachelor's Degree Final Project

IoT Solution for Home Food Management

Author: Adrián Llana Ben

Tutor: Bernardo Alarcos Alcázar

2023

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

Computer Engineering Bachelor's
Degree

Bachelor's Degree Final Project

IoT Solution for Home Food Management

Author: Adrián Llana Ben

Tutor: Bernardo Alarcos Alcázar

2023

Contents

1. Introduction ... 10
2. Purpose ... 11
3. State of art ... 12
   3.1 Smart fridges origins ... 12
   3.2 Nowadays smart fridges ... 13
   3.3 Market study ... 14
4. Development ... 17
   4.1 Hardware ... 17
      4.1.1 Single-Board Computer SBC ... 18
      4.1.2 5 inch touchscreen ... 21
      4.1.3 Camera ... 22
      4.1.4 Heat dissipater ... 23
      4.1.5 Microphone ... 24
      4.1.6 Remote control ... 25
   4.2 Software ... 25
      4.2.1 Operating system ... 26
      4.2.2 Developing enviroment ... 27
      4.2.3 Code repository ... 27
      4.2.4 Database ... 29
      4.2.5 DDBB connection ... 30
      4.2.6 Programming language ... 31
      4.2.7 Interface ... 31
      4.2.8 Structure and code ... 32
      4.2.9 External services ... 42
      4.2.10 Loggin ... 44
5. Conclusions ... 45
6. Future additions ... 45
   6.1 Using external services ... 45
   6.2 Long-term functionalities ... 46
   6.3 System configurations ... 46
   6.4 Components improvements ... 46
   6.5 Interface improvements ... 46
7. Bibliography ... 47
8. Annex ... 49
   8.1 Annex I – Project Cost ... 49
      8.1.1 Material cost ... 49

8.1.2 Personal cost 50
8.1.3 General costs 51
8.1.4 Coste total 51
8.2 Annex II – User Manual 52
8.2.1 Main menu 52
8.2.2 NEW PURCHASE 53
8.2.3 STORED PRODUCTS 60
8.2.4 RECIPE IDEAS 63

Figures

Figure 3.1: 1998 Article about first smart fridges 12
Figure 3.2: The LG Internet Digital DIOS (also known as R-S73CT)..... 12
Figure 3.3: Function FamilyHub - View Inside..... 14
Figure 3.4: Function FamilyHub – Shopping List 14
Figure 3.5: Function FamilyHub – Family Communication..... 15
Figure 3.6: Function FamilyHub - Calendar 15
Figure 3.7: Function FamilyHub - Smart View..... 16
Figure 3.8: Function FamilyHub - Smart Things 16
Figure 4.1: Project Functional Scheme 17
Figure 4.2: Project Hardware Diagram 18
Figure 4.3: Pinout Raspberry Pi 4 Model B 19
Figure 4.4: Dimensions Raspberry Pi 4 Model B 20
Figure 4.5: Raspberry Pi 4 Model B 20
Figure 4.6: Touchscreen 21
Figure 4.7: Connected Touchscreen and Raspberry 21
Figure 4.8: Touchscreen assemblent..... 22
Figure 4.9: Camera mosule Pi Noir..... 22
Figure 4.10: Heat dissipater Dimensions 23
Figure 4.11: Heat dissipater Conexions..... 23
Figure 4.12: Board and Heat dissipater connected..... 24
Figure 4.13: USB Microphone 24
Figure 4.14: Rii X8 Mini..... 25
Figure 4.15: Project Software Diagram..... 26
Figure 4.16: Screenshot Raspberry Pi Imager..... 27
Figure 4.17: .gitignore File..... 28
Figure 4.18: Firebase logo..... 29
Figure 4.19: Project Database example..... 30
Figure 4.20: Developed interface example..... 31
Figure 4.21: App structure..... 32
Figure 4.22: IARequest Folder 33
Figure 4.23: components Folder..... 33
Figure 4.24: components/MenuOptions Folder..... 34
Figure 4.25: screens Folder 35
Figure 4.26: solutionAlerts Folder 36
Figure 4.27: Alert mail sent example 37
Figure 4.28: Telegram bot screenshot 38
Figure 4.29: solutionDB Folder 38
Figure 4.30: styles Folder..... 39
Figure 4.31: Common configurations 39
Figure 4.32: Each app part configuration..... 40
Figure 4.33: Main menu style 40
Figure 4.34: ‘NEW PURCHASE’ style..... 41
Figure 4.35: ‘STORAGE PRODUCTS’ style 41
Figure 4.36: ‘RECIPE IDEAS’ style..... 41
Figure 4.37: OpenAI logo 42
Figure 4.38: OpenFoodFacts logo..... 43
Figure 4.39: Telegram bot info..... 44
Figure 4.40: Registered logs example..... 45
Figure 6.1: Product Table improvement example 46

Figure 6.2: Delete icon 47
Figure 6.3: Icon test result 47
Figure 8.1: App main menu 52
Figure 8.2: ‘NEW PURCHASE’ main screen 53
Figure 8.3: Voice reccognition preparation 53
Figure 8.4: Screen while app is listening 54
Figure8.5: Voice recognition confirmation screen 54
Figure 8.6: Insert Date screen 55
Figure 8.7: Wrong date example 55
Figure 8.8: Wrong date message example 55
Figure 8.9: Voice recognition fail message 56
Figure 8.10: Registered products message 56
Figure 8.11: Type input preparation 57
Figure 8.12: Type input 57
Figure 8.13: Confirmation screen Type input 58
Figure 8.14: Scan product preparation screen 58
Figure 8.15: Screen while app is scanning 59
Figure 8.16: ‘STORED PRODUCTS’ main screen 60
Figure 8.17: Stored products visualization 60
Figure 8.18: Porduct selection 61
Figure 8.19: Product deleted informative message 61
Figure 8.20: Warning message all product delection 62
Figure 8.21: Delete every product stored screen 62
Figure 8.22: ‘RECIPE IDEAS’ main screen 63
Figure 8.23: Product list visualization 63
Figure 8.24: Selecting mode 64
Figure 8.25: Product list selection for recipe ideas 64
Figure 8.26: Recipe ideas result 1/2 65
Figure 8.27: Recipe ideas result 2/2 65

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. Introduction

The Internet of Things (IoT) technology is no longer a novelty. In recent decades, it has undergone impressive advancements, becoming increasingly prevalent in homes in various forms. In fact, it's rare to find an area where IoT is not already present. IoT has demonstrated its ability to transform how we interact with the world around us. The evolution of our homes into connected and aware spaces presents an unprecedented opportunity to address numerous issues.

Home food management is one of the areas that could benefit immensely from these IoT solutions. In a world where sustainability and efficiency are becoming increasingly important, having tools that help reduce food waste and enhance consumption efficiency is crucial.

Food waste is a global dilemma with profound economic, environmental, and ethical implications. We find ourselves in a context where natural resources are becoming more limited, and the global population continues to grow. Despite this, according to the Food and Agriculture Organization (FAO) of the United Nations, approximately one-third of all globally produced food is lost or wasted annually [1].

On a national level, in Spain, according to data from the Ministry of Agriculture, Fisheries, and Food, Spanish households wasted a total of 1,245.86 million kilograms or liters in the year 2021 [2].

Therefore, it is important to seek innovative solutions to address this challenge, and IoT holds great potential in this field.

2. Purpose

This project encompasses the design and development of an IoT solution through a prototype. The overarching goal of this implementation is to assist users in tracking food consumption, efficiently planning their purchases and meals to optimize usage, save time, money, and contribute to a more sustainable world.

Specific objectives include:

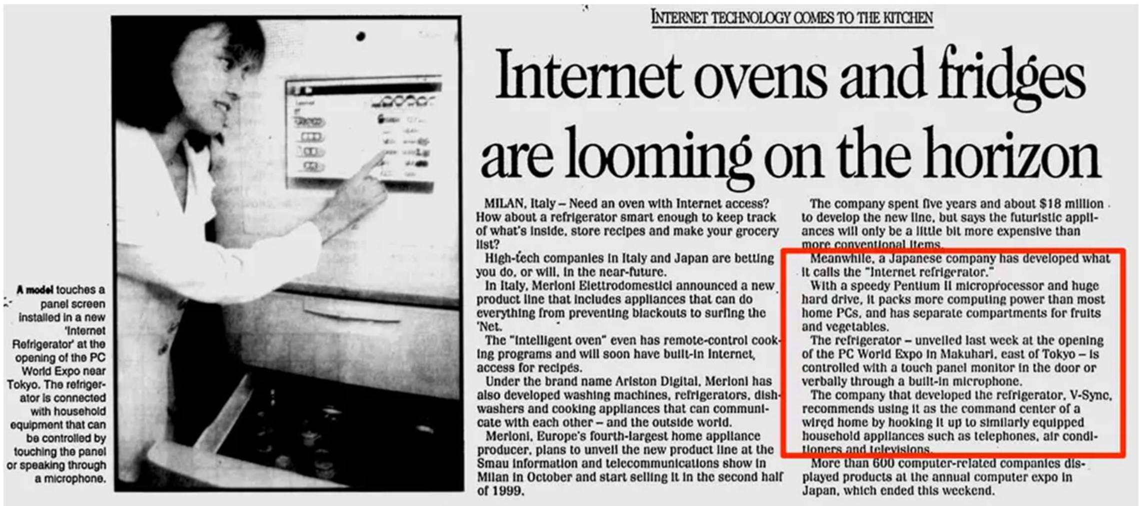
- **Development of a specific application** for use in the prototype. This application aims to manage the logging of food input and extraction from the refrigerator. It also facilitates recipe queries with an interface adapted for each process.
- **Creation and management of a cloud-based database.** Using a cloud-hosted database to store product information enables potential access from different devices and locations.
- **Implementation of an alert system.** This system notifies the user via email and through a Telegram bot when the expiration date of any stored product has passed or is approaching. The messages are personalized for each user and the status of each product.
- **AI-powered Recipe Suggestions**: Implement functionality allowing users to select a list of available products and receive recipe suggestions utilizing artificial intelligence. This involves making requests to an API from a service that uses AI.

The project utilizes a Raspberry Pi 4 Model B, integrating various features to make food management an optimal and automated process.

3. State of art

3.1 Smart fridges origins

Even though the concept of a smart refrigerator has gained popularity in recent years, it is not as recent as one might think. As early as 1998, there were reports anticipating their market debut with headlines such as '*Internet ovens and fridges are "looming on the horizon*' [4], as shown in Figure 3.1.



INTERNET TECHNOLOGY COMES TO THE KITCHEN

Internet ovens and fridges are looming on the horizon

A model touches a panel screen installed in a new 'Internet Refrigerator' at the opening of the PC World Expo near Tokyo. The refrigerator is connected with household equipment that can be controlled by touching the panel or speaking through a microphone.

MILAN, Italy – Need an oven with Internet access? How about a refrigerator smart enough to keep track of what's inside, store recipes and make your grocery list?

High-tech companies in Italy and Japan are betting you do, or will, in the near-future.

In Italy, Merloni Elettrodomestici announced a new product line that includes appliances that can do everything from preventing blackouts to surfing the 'Net.

The "intelligent oven" even has remote-control cooking programs and will soon have built-in Internet, access for recipes.

Under the brand name Ariston Digital, Merloni has also developed washing machines, refrigerators, dishwashers and cooking appliances that can communicate with each other – and the outside world.

Merloni, Europe's fourth-largest home appliance producer, plans to unveil the new product line at the Smau information and telecommunications show in Milan in October and start selling it in the second half of 1999.

The company spent five years and about $18 million to develop the new line, but says the futuristic appliances will only be a little bit more expensive than more conventional items.

Meanwhile, a Japanese company has developed what it calls the "Internet refrigerator."

With a speedy Pentium II microprocessor and huge hard drive, it packs more computing power than most home PCs, and has separate compartments for fruits and vegetables.

The refrigerator – unveiled last week at the opening of the PC World Expo in Makuhari, east of Tokyo – is controlled with a touch panel monitor in the door or verbally through a built-in microphone.

The company that developed the refrigerator, V-Sync, recommends using it as the command center of a wired home by hooking it up to similarly equipped household appliances such as telephones, air conditioners and televisions.

More than 600 computer-related companies displayed products at the annual computer expo in Japan, which ended this weekend.

Figure 3.1: 1998 Article about first smart fridges

The first smart refrigerator to hit the market was developed in the early 2000s. In September 2000, LG Electronics introduced an Internet-connected refrigerator called the LG Internet Refrigerator (Figure 3.2). This project, initiated in 1997, featured a 56k modem and a 15-inch touchscreen on one of its doors. In addition to internet browsing and email capabilities, it allowed users to download MP3 files, connect a DVD player, take a photo, or watch television. It could also automatically order groceries for items that were running out or nearing their expiration date [3].

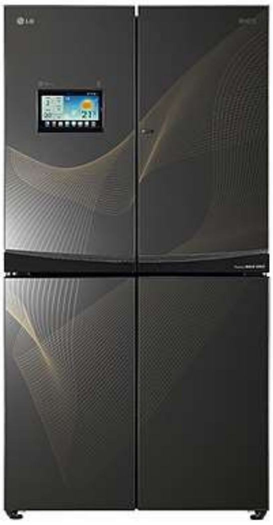


Figure 3.2: The LG Internet Digital DIOS (also known as R-S73CT)

Despite its pioneering features, the market reception for this product was not as favourable as expected. Many consumers felt that what this product could bring to their homes did not justify its high price ($20,000). Additionally, the Russian cybersecurity company, Kaspersky Lab, warned that these internet-connected refrigerators could be easily targeted, either with viruses causing the door to open in the middle of the night or serving as sources of large volumes of malicious emails [5].

All of this led to the project achieving almost negligible success relative to the capital invested in its development. Over the years, new models of smart refrigerators have been introduced, incorporating new features and improving their capabilities.

3.2 Nowadays smart fridges

Despite the historical lack of success for this type of product, major brands continue to invest in developing new models. This reflects the growing belief that technology can add value to an appliance as simple as a refrigerator.

Today, the idea of a home without a refrigerator is unthinkable, as this appliance has become a central point in households. Society is moving towards an increasingly interconnected world, and smart refrigerators present an opportunity to revolutionize how we manage our daily lives and interact with our food. To achieve this, these devices come with features such as:

- LCD Screen: Sometimes equipped with a touchscreen for easy interaction or a remote control. These screens are often placed on one of the refrigerator doors. Thanks to software with user-friendly interfaces, they allow users to view information about the contents or the device's status, manage settings, or download new applications.
- **Internet Connection**: Generally, through Wi-Fi, this feature enables keeping the device up-to-date with updates, using external services, and accessing it remotely from anywhere.
- **Integrated Cameras**: Allow remote viewing of the refrigerator's interior without the need to open it, enhancing energy efficiency.
- **Built-in Speakers and Microphone**: Enable interaction with voice assistants, receive notifications from connected homes, or listen to music.
- **Smart Temperature Control**: Allows users to control the refrigerator's temperature and adjust it automatically based on user preferences.
- **Connectivity with Mobile Apps**: Many models have dedicated applications designed specifically for use with that refrigerator, making it easier to provide an optimal and personalized experience.

All these features contribute to the benefits that a smart refrigerator can offer compared to a regular one.

3.3 Market study

There are few differences in terms of functionalities among the refrigerator options currently available in the market, as most offer practically the same features. The sections where they tend to differ the most are specifications such as screen size, refrigerator/freezer capacity, or the inclusion of an interior camera. To get an overview of the current market situation, it suffices to study one of the most comprehensive and highly-rated options, the Side by Side from the Family Hub series by Samsung [6]. Among the functionalities offered by this device are:

- **View Inside.** (Figure 3.3) With this feature, it's possible to visualize the refrigerator's content on the home screen or from a smartphone, for example, while shopping.

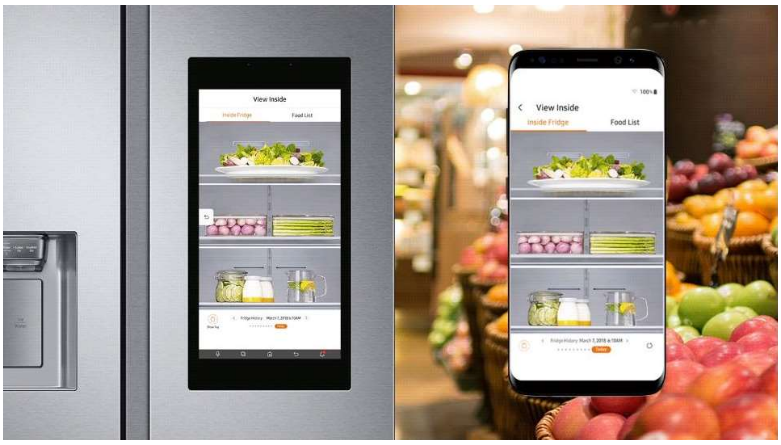


Figure 3.3: Function FamilyHub - View Inside

- **Shopping List.** (Figure 3.4) This function allows users to create shopping lists by adding items to the list from the screen. It can be synchronized with a smartphone for easy reference while at the supermarket [6].

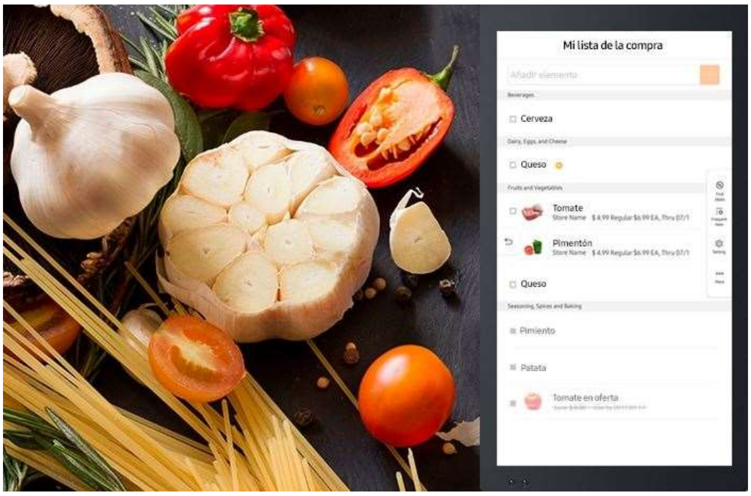


Figure 3.4: Function FamilyHub - Shopping List

- **Meal Planner:** Allows users to add weekly menus, manage them along with a calendar, and helps prevent monotony by avoiding repetition or overconsumption of certain foods.

It aids in maintaining a balanced and varied diet. This functionality can be particularly useful for those following a special diet or dietary restrictions.

- **Family Communication:** (Figure 3.5) With this function, users can interact with other members residing in the household. Interactions can include handwritten notes, drawings, photos, or to-do lists, replacing the typical notes that end up on the fridge door [6].

Figure 3.5: Function FamilyHub - Family Communication

- **Calendar:** (Figure 3.6) This feature allows users to share and view the schedules of all household members at a glance. Entries can be added or updated from the refrigerator itself or from a smartphone [6].

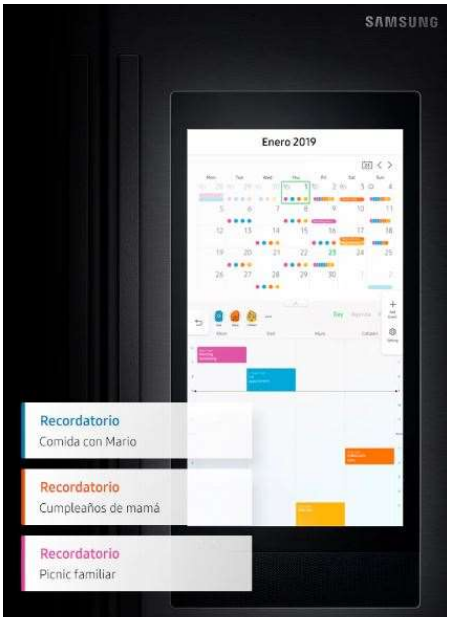


Figure 3.6: Function FamilyHub – Calendar

- **Smart View.** (Figure 3.7) This feature allows you to display content on the refrigerator screen, including TV shows, recipe videos, smartphone screen mirroring, and more. All of this can be viewed while you're in the kitchen cooking.

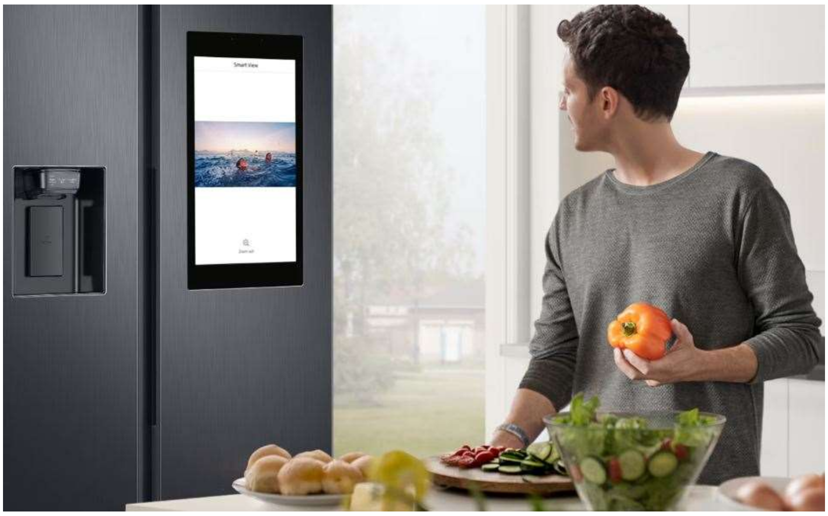


Figure 3.7: Function FamilyHub - Smart View

- **SmartThings™.** (Figure 3.8) With this feature, you can achieve greater home automation, as it allows you to control other connected appliances from the refrigerator screen. There is also an app available to manage this through a smartphone.

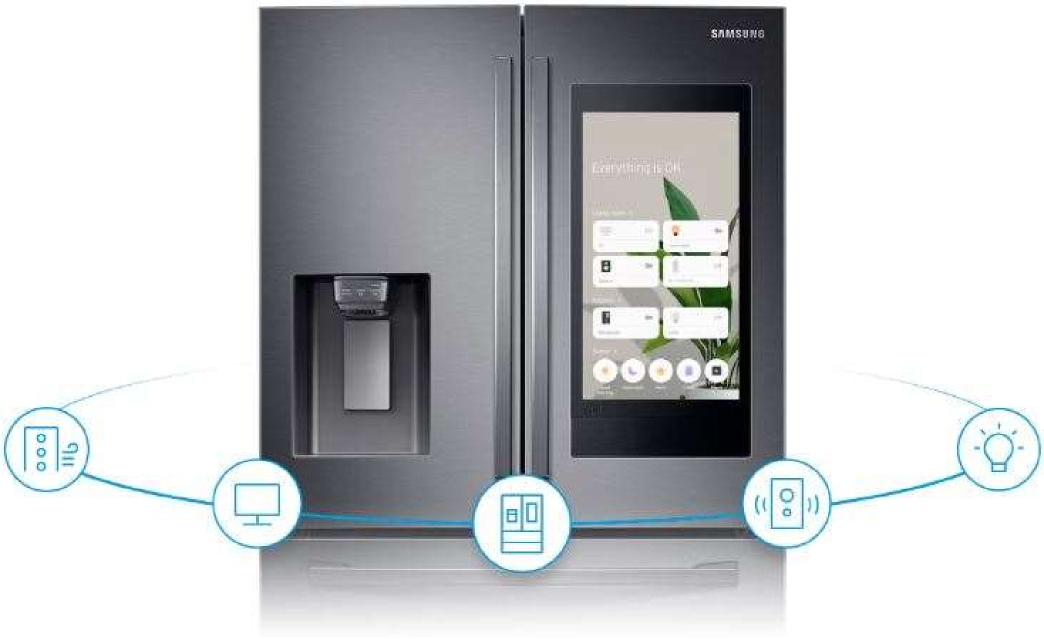


Figure 3.8: Function FamilyHub - Smart Things

4. Development

The intention of this development encompasses various functionalities, similar to those seen in the example in the previous section, with the main advantages being that, as an external prototype, it is applicable to any refrigerator, and its cost is reduced. As shown in Figure 4.1, the implemented functionalities include:

- **Registration of new products** after a purchase, with the option to choose from three different registration methods (voice recognition, keyboard input, and barcode scanning).
- **Storage management**, allowing the visualization and deletion of products already registered in the system.
- **Recipe ideas**, which, after selecting a list of products present in the system, provide possible recipes to prepare.
- Finally, **alerts/notifications**, informing the user if the expiration date of any products in the system has passed or is approaching.

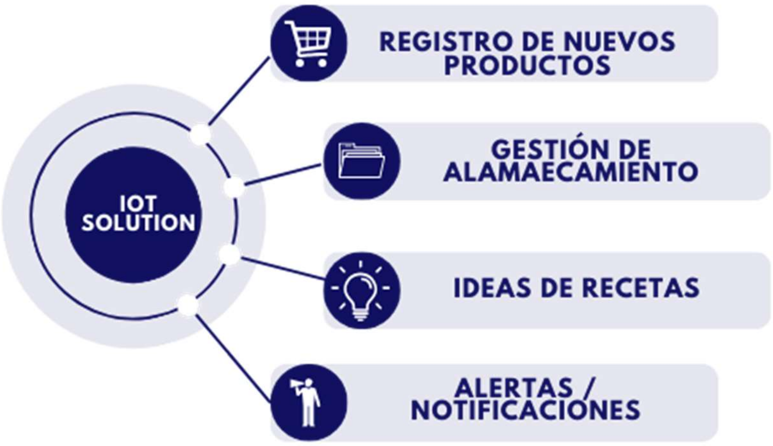


Figure 4.1: Project functional scheme

This section contains the detailed process followed to develop this project, primarily divided into: **hardware** used, **software** used, and the **implementation** between the two.

4.1 Hardware

To develop a prototype that adds smart refrigerator functionalities to a common one, the selection and implementation of hardware components are crucial. This section details all the elements that make up the infrastructure necessary to assemble the prototype.

Each component has been chosen based on a clear criterion: optimizing the relationship between budget and performance. This approach has been key to ensuring that each component contributes in the best possible way to the prototype's architecture without allowing its cost to have too high an impact on the overall project budget. By carefully considering the technical capabilities of each component along with financial limitations, a balance has been achieved between the efficient performance of the prototype and its overall cumulative cost.

In Figure 4.2, you can see a diagram with the different components that make up the prototype.

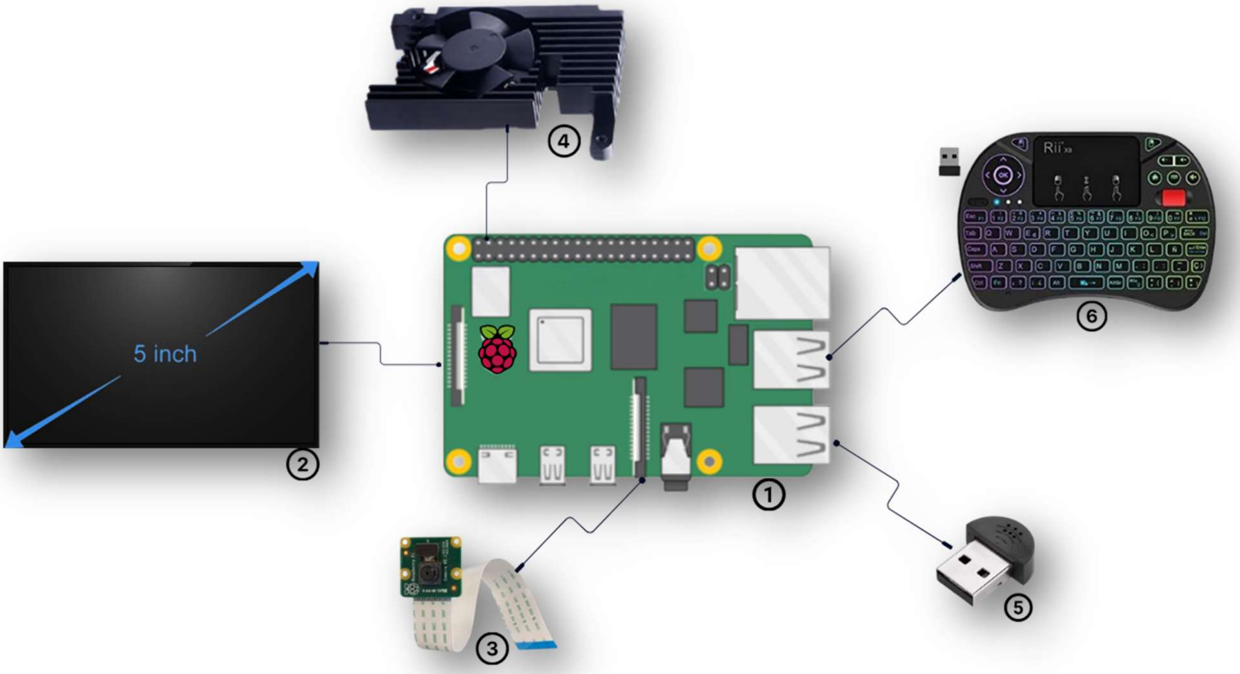


Figure 4.2: Project Hardware Diagram

1. Single-Board Computer SBC
2. Touch screen
3. Camera Module
4. Heat dissipater
5. USB Microphone
6. Remote control

Next, an analysis of each of the components used.

4.1.1 Single-Board Computer SBC

As the central element of the development and the 'brain' of the prototype, a Single Board Computer (SBC) is used. This is a device that integrates the essential elements of a computer into a single printed circuit board, providing a compact and simplified configuration. It is ideal for applications where size, cost, and efficiency are crucial, as in this project.

Raspberry Pi 4 model B 2GB

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.

The selected option for this component is the Raspberry Pi 4 model B. This model has ideal specifications to support the purpose of the prototype.

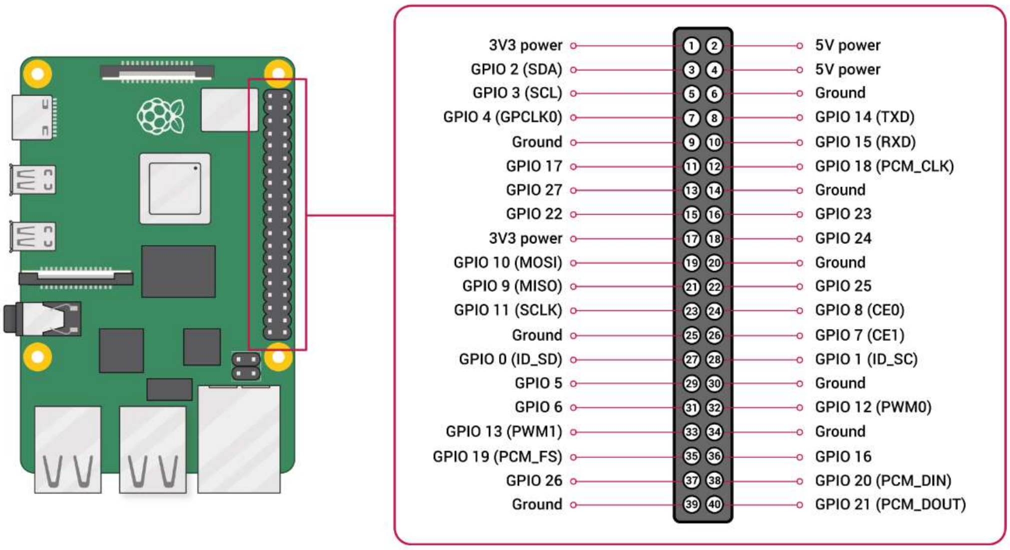
Despite its small size, this board provides considerable processing power, more than sufficient memory capacity, extensive connectivity, and a variety of ports. Additionally, it is highly compatible with other external hardware and software components. Designed to be energy-efficient, and being a well-known Raspberry brand, it has a large community of developers and users providing support and valuable resources.

The specifications of this model are as follows:

Broadcom BCM2711 chip with a 64-bit ARM Cortex-A72 **processor** running at a clock frequency of 1.50 GHz.

Different **memory** configurations are available for this model (1GB, 2GB, 4GB, or 8GB), and in this case, it is equipped with 2GB LPDDR4 SDRAM.

In terms of **connectivity**, it features a 2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac Wi-Fi setup, Bluetooth 5.0, Gigabit Ethernet, 2 USB 3.0 ports, and another 2 USB 2.0 ports. The connections used by this prototype are detailed later. For expansion, it includes a 40-pin GPIO header, shown in detail in the 'pinout' of Figure 4.3.



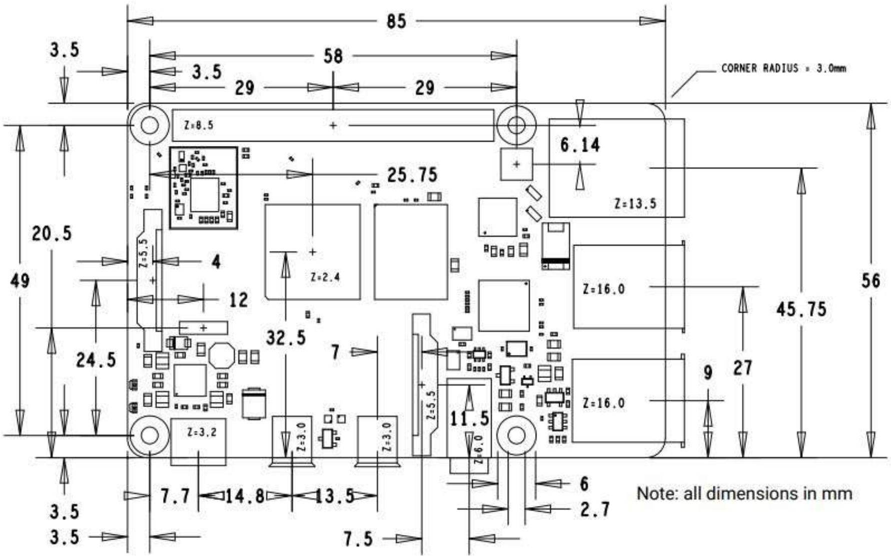
3V3 power 1 2 5V power
GPIO 2 (SDA) 3 4 5V power
GPIO 3 (SCL) 5 6 Ground
GPIO 4 (GPCLK0) 7 8 GPIO 14 (TXD)
Ground 9 10 GPIO 15 (RXD)
GPIO 17 11 12 GPIO 18 (PCM_CLK)
GPIO 27 13 14 Ground
GPIO 22 15 16 GPIO 23
3V3 power 17 18 GPIO 24
GPIO 10 (MOSI) 19 20 Ground
GPIO 9 (MISO) 21 22 GPIO 25
GPIO 11 (SCLK) 23 24 GPIO 8 (CE0)
Ground 25 26 GPIO 7 (CE1)
GPIO 0 (ID_SD) 27 28 GPIO 1 (ID_SC)
GPIO 5 29 30 Ground
GPIO 6 31 32 GPIO 12 (PWM0)
GPIO 13 (PWM1) 33 34 Ground
GPIO 19 (PCM_FS) 35 36 GPIO 16
GPIO 26 37 38 GPIO 20 (PCM_DIN)
Ground 39 40 GPIO 21 (PCM_DOUT)

Figure 4.3: Pinout Raspberry Pi 4 Model B

For **multimedia**, it has 2 micro-HDMI ports that support 4k@60Hz displays with HDMI 2.0, a MIPI DSI display port, and a MIPI CSI camera port. It also features a micro-SD card slot, necessary for loading an operating system onto the board.

For **power**, a continuous current of 5V and a minimum of 3A is required, delivered either through a USB-C port or via the GPIO header.

Detailed dimensions of the board can be seen in Figure 4.4.



Note: all dimensions in mm

Figure 4.4: Dimensions Raspberry Pi 4 Model B

Finally, regarding the **environment**, the temperature at which this device can operate ranges from 0-50°C.

A real image of the board with all the specifications mentioned above is shown in Figure 4.5.

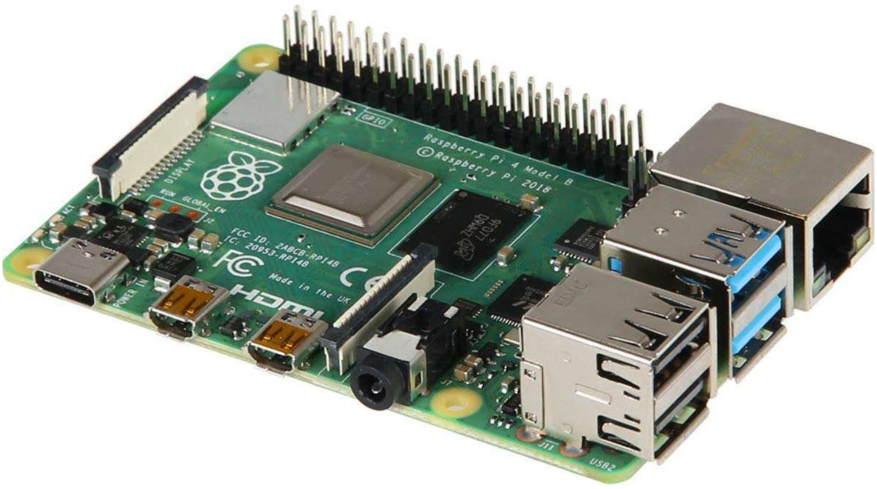


Figure 4.5: Raspberry Pi 4 Model B

4.1.2 5 inch touchscreen

The incorporation of a touchscreen in the prototype is crucial to providing the user with the developed functionalities in an accessible and intuitive way. Thanks to it, and together with the specific interface developed for the project, the interaction between the user and the prototype is strengthened.

The selected size of 5 inches (Figure 4.6) fits perfectly with the Raspberry Pi. In the next section, detailed information about the selected screen model and how the assembly on the board is carried out can be found.

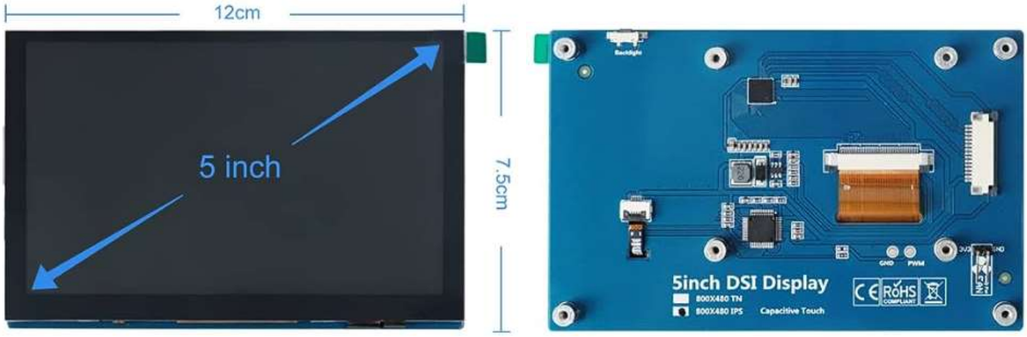


Figure 4.6: Touchscreen

Freenove 5 Inch Touchscreen Monitor

The selected model for the prototype's screen is this Freenove Touchscreen Monitor, the 5-inch option. The dimensions of the component are 15.2 x 14 x 3.8 cm; 191 grams, fitting perfectly with the baseboard (Figure 4.7) while still displaying information clearly and in detail, with a resolution of 800 x 480.

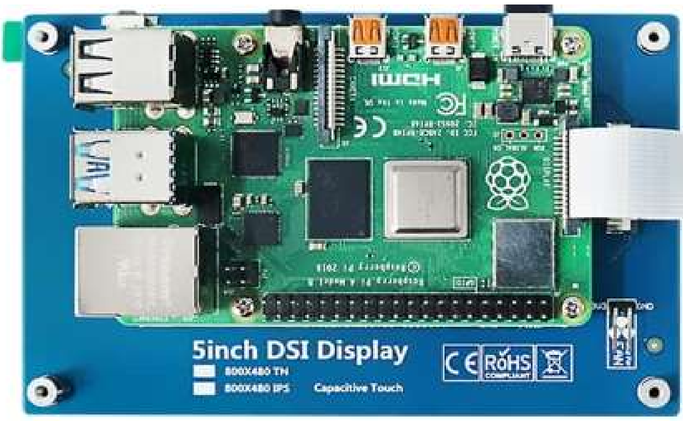


Figure 4.7:Connected touchscreen and Raspberry

Its touchscreen interface facilitates user interaction and enhances their experience. It has been designed and optimized for use with the Raspberry Pi, ensuring compatibility and easy setup.

Although the board has HDMI inputs, this screen uses its MIPI DSI port for connection and information transfer. The assembly is straightforward, as shown in Figure 4.8.

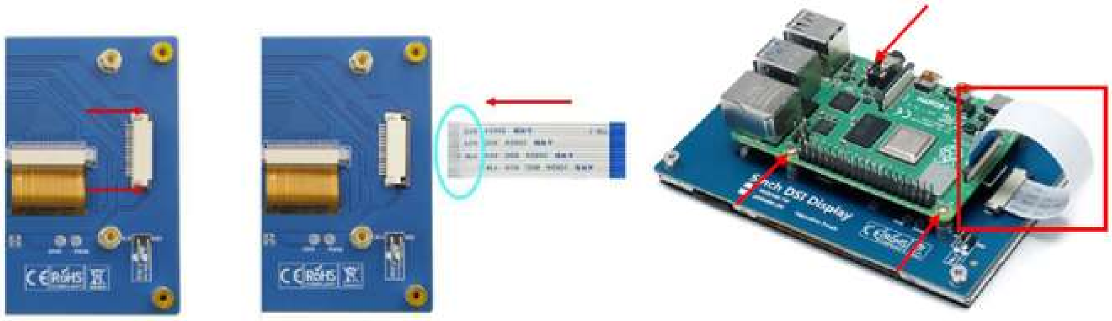


Figure 4.8: Touchscreen assemblent

4.1.3 Camera

The integration of a camera module in the prototype has been carried out with the purpose of enabling one of the three types of product recognition. For this, the camera module captures an image showing the barcode of a product. With this component, we allow the user to have one more option to choose from for the product registration process in the system.

Pi Noir Módulo v2.1 8MP 1080p

The selection of this camera model (Figure 4.9) has been based on its technical characteristics, suitable for its purpose in the project. It is capable of capturing images with a resolution of 8 megapixels and in low-light environments, very easy to integrate with the Raspberry Pi, and for mounting, it uses its MISI CSI port. It has very compact dimensions (0.25 x 0.24 x 0.01 cm and 3 grams), contributing to the compact design of the prototype.

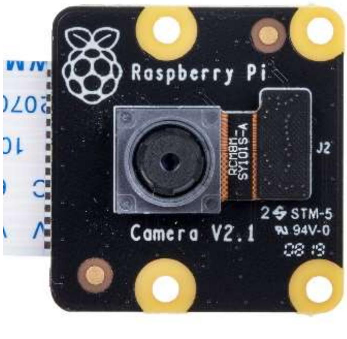


Figure 4.9: Camera Module Pi Noir

4.1.4 Heat dissipater

The incorporation of a heat sink into the prototype has been done with several purposes. During the development and testing of the prototype's implemented functionalities, it is important to control its temperature, as these processes demand a lot from the components. Overheating can affect their performance and lifespan. Therefore, the integration of this component helps with the system's stability, provides a greater performance margin, and extends the overall project's lifespan. Additionally, it allows the prototype to operate in a controlled environment, within a safe operating range, regardless of external conditions or the level of demand it is exposed to.

GeeekPi Armor lite with PWM fan

The selected model for the heat sink is designed for the Raspberry Pi and implements a differential functionality that distinguishes it from most options for dissipating heat on these boards.

It is compatible with PWM (Pulse Width Modulation) speed control, allowing for software-controlled speed adjustments compatible with the operating system used. It functions as a switch, also known as a gate, achieving a connection and disconnection of power with each pulse. PWM converts a digital signal into an analog one by changing the amount of time it stays on or off in its duty cycle. [8] Its dimensions fit perfectly with those of the board, as shown in Figure 4.10.

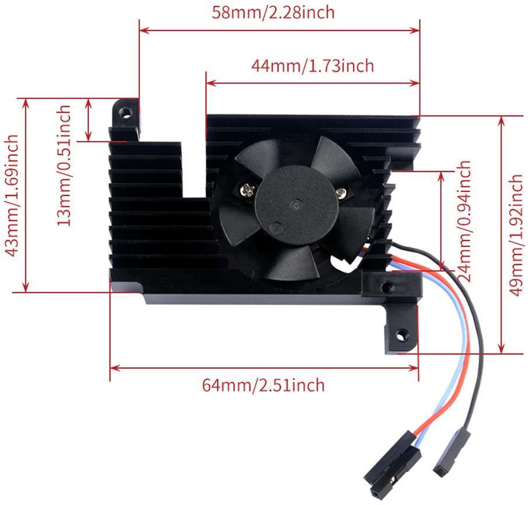


Figure 4.10: Heat dissipater dimensions

It is mounted and connected using the GPIO pins. In this case, the 5V socket, pin 6 as ground or GND, and pin 8 (GPIO 14 TXD) to transmit data are used, as shown in Figure 4.11

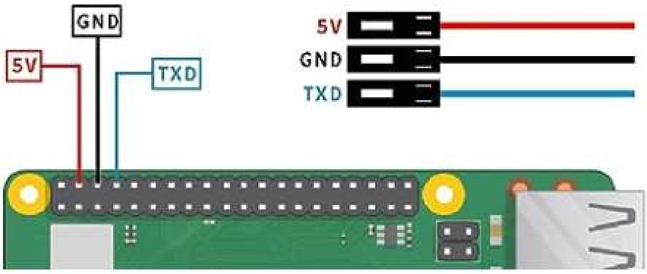


Figure 4.11: Conexiones disipador de calor

After mounting and connecting the heat sink to the board as specified, the result is shown in Figure 4.12.

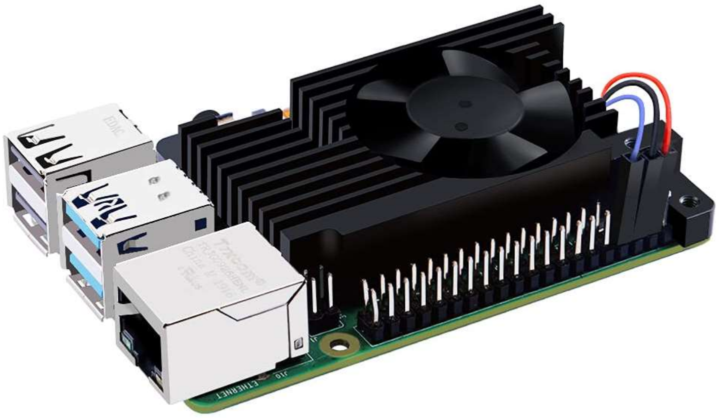


Figure 4.12: Board with dissipater connected

4.1.5 Microphone

As with the integration of the camera module, a microphone is added to enable another of the three types of product recognition, in this case, voice recognition.

This is possibly the option that most users may choose, as it allows adding products to the system in a convenient and simple way, just by mentioning their name, which speeds up the registration process and improves the user experience.

Fasient USB Microphone Mini

It is a 360º omnidirectional microphone with an effective distance of up to 2 meters and supports up to 67 dB. Despite the existence of other alternatives with different types of connections, this one has been selected, taking advantage of one of the USB ports of the Raspberry Pi board.

This model (Figure 4.13) maintains a simple and functional design. Its dimensions are very small, barely protruding from the USB port (2.2 x 1.9 x 0.4 cm).

Figure 4.13: USB Microphone

4.1.6 Remote control

To facilitate system management and information input, a remote controller with a keyboard and touchpad as a mouse is included. This component adds an alternative method for user interaction, enriching the experience and ensuring flexibility in use.

Furthermore, it serves as a precautionary measure against possible failures in the other two methods of product registration (voice recognition and barcode scanning). This component adds the third format, through the keyboard, allowing the user to precisely type the name of the product they intend to register, as well as its expiration date. Additionally, it enables navigation throughout the system

Rii X8 Mini

Among the specifications of this model (Figure 4.14), the connection of this component with the prototype is carried out through a receiver (dongle) in one of the USB ports. It features a 2.4GHz GFSK wireless transmission mode with a range of up to 10m thanks to a transmission power of ±5db. Its dimensions are compact, measuring 13.8 x 9.6 x 2.1 cm, with a weight of 112.5g. Lastly, it comes with a rechargeable Lithium-Ion battery.

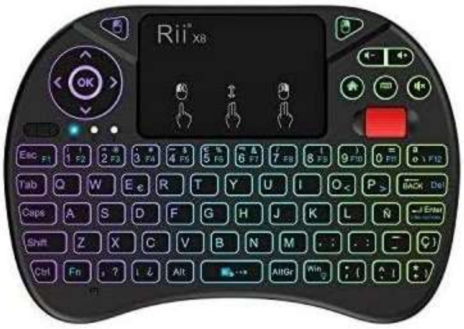


Figure 4.14: Rii X8 Mini

4.2 Software

The software is an essential aspect of any technological project. It provides intelligence, allowing the application of specific logic and algorithms. It enables process automation, reducing the need for human intervention to a large extent. It adds adaptability to user requirements and configurations. Software also provides efficiency by optimizing resources. A system with well-designed software can reduce costs and maximize productivity. Figure 4.15 shows a diagram of the software integrated into this project.

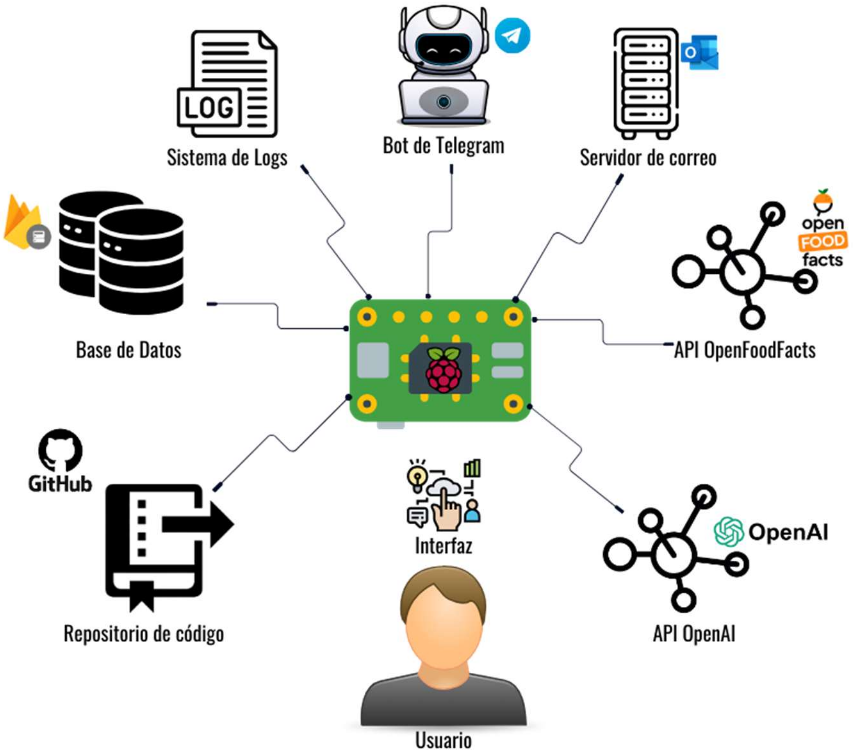


Figure 4.15: Project Software Diagram

The developed software, its configurations, and all the functionalities it brings to the project are presented below.

4.2.1 Operating system

The set of hardware components presented earlier serves no purpose without tailored software that coordinates the functions of each component and allows them to work in harmony towards the goals of the prototype. For this, the presence of an operating system is necessary. In this project, the selected operating system is Raspberry Pi OS, formerly known as Raspbian.

Raspberry Pi OS

It is a Linux distribution based on Debian, providing a stable foundation. It was designed specifically for the single-board computer used in this project, the Raspberry Pi, across all its models. The environment that comes with this distribution facilitates the development of new applications and has great compatibility with peripherals and external hardware.

For its installation and configuration on the Raspberry Pi 4 Model B used, the official tool from the brand, Raspberry Pi Imager (Figure 4.16), has been employed. This tool allows for quick image burning of operating systems onto a microSD card from another device. The version used is Raspberry Pi OS 64-bit [9].

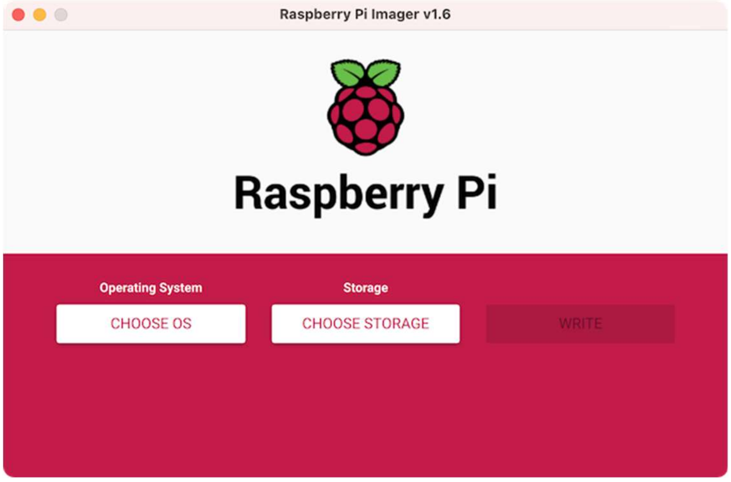


Figure 4.16: Screenshot Raspberry Pi Imager

4.2.2 Developing enviroment

Once the operating system is installed on the board, the environment in which the application will be developed has been configured, and the code editor VSC (Visual Studio Code) has been selected.

Visual Studio Code

VSC is a source code editor developed by Microsoft. It offers a wide range of features and extensions that make it a powerful tool for developers. Among its features, integrated debugging allows tracking and fixing errors, and it includes an integrated terminal that enables running commands directly from the editor.

4.2.3 Code repository

For a quality development process, it's crucial to use a structured environment that allows tracking the code. Code repositories add many advantages to projects.

When adding new versions with new features, it's important to do it securely, without allowing it to introduce problems into the general functioning of the project. For this, repositories include features such as version control or changelog, recording each change and facilitating identification and rollback in case of problematic modifications or understanding the evolution of the application.

Although for this project, with a single developer, the application of these features hasn't been necessary, code repositories also bring many advantages when developing as a team, ensuring effective collaboration.

In this project, the code repository used has been possibly the most well-known, GitHub.

GitHub

GitHub is a collaborative development platform that allows hosting software projects using the version control system known as Git. GitHub enables maintaining a repository in the cloud with your code, allowing you to clone it on any device and from any location if you have the necessary permissions. This adds security to the project and ensures you don't lose your changes as long as you keep the repository updated.

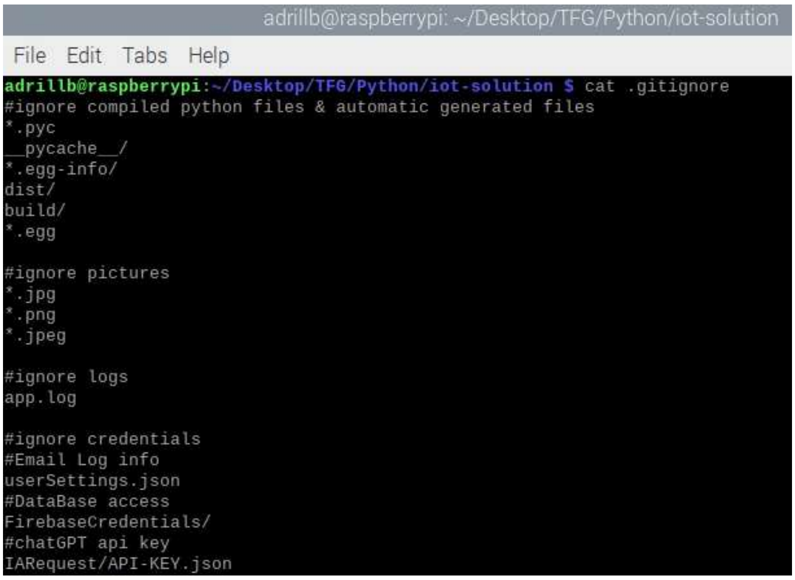
In this project, GitHub has been used by a single individual, so not all the features of the platform have been utilized.

Once the project is linked to the GitHub repository, it's straightforward to make modifications to the repository using specific commands from the Linux terminal, mainly employing the Git version control mentioned earlier. The repository's GitHub address used in the project is https://github.com/adrillb/iot-solution-tfg.

To add modifications to a project, you work on a specific branch. A Git branch is an efficient pointer to snapshots of changes.

To check the introduced changes, the "**git status**" command is used, displaying the modified files and those that have been added to an index that will be uploaded later. If any of these changes have not been added to the index, the "**git add**." command is used, adding all modified files to the index. When the changes are ready and within this index, the so-called commit must be carried out using the "**git commit**" command. In this step, it's crucial to add a comment that is as descriptive as possible, as it will be useful in the future. Multiple commits can be made in the same branch. When the changes are ready to be merged with the rest of the repository code, the "**git push**" command is used, specifying the name of the master branch and the one being worked on. After this, a merge has been performed, meaning the modifications have been included in the repository.

There are cases where certain changes are not meant to be included in the general repository, either because they are specific configurations for local development or secret keys that should not be made public. For this purpose, the "**.gitignore**" file (Figure 4.17) is used, preventing the upload of certain files or folders to the repository. In the case of this project, the following lines have been included in this file:



```
adrillb@raspberrypi:~/Desktop/TFG/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
```

Figure 4.17: .gitignore File

In the following sections, reference is made to these specific cases of files that should not be added to the repository.

4.2.4 Database

The goal of this project lies in the ability to manage and somehow maintain products, creating an inventory on which to apply the rest of the functionalities. For this, the use of a database is necessary.

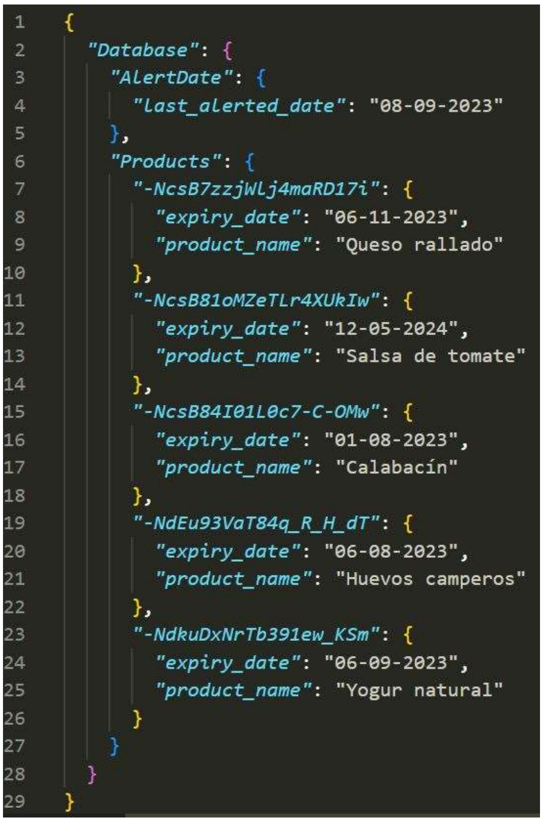
Since storing information in a local database lacks the efficiency and usability required today, a more effective alternative has been sought. In order to have the convenience of accessing this information from anywhere and allowing global reach, we have used a real-time database model hosted in the cloud, specifically, Firebase Realtime Database.

Firebase Realtime Database

Firebase Realtime Database (Figure 4.18) is a cloud-hosted NoSQL database where data is synchronized in real-time with each client, remaining available when the application is offline. Files are stored in JSON format, associating a unique key with each item.

Figure 4.18: Firebase logo

In Figure 4.19, an example screenshot of the project's database is shown:



```
{
  "Database": {
    "AlertDate": {
      "last_alerted_date": "08-09-2023"
    },
    "Products": {
      "-NcsB7zzjWLj4maRD17i": {
        "expiry_date": "06-11-2023",
        "product_name": "Queso rallado"
      },
      "-NcsB81oMZeTLr4XUkIw": {
        "expiry_date": "12-05-2024",
        "product_name": "Salsa de tomate"
      },
      "-NcsB84I01L0c7-C-OMw": {
        "expiry_date": "01-08-2023",
        "product_name": "Calabacín"
      },
      "-NdEu93VaT84q_R_H_dT": {
        "expiry_date": "06-08-2023",
        "product_name": "Huevos camperos"
      },
      "-NdkuDxNrTb391ew_KSm": {
        "expiry_date": "06-09-2023",
        "product_name": "Yogur natural"
      }
    }
  }
}
```

Figure 4.19: Project Database example screenshot

The hierarchy and structure of the database can be observed. From a main branch 'Database,' there are two branches called 'AlertDate' to ensure that the user doesn't receive more than one alarm/notification per day and another called 'Products' from which all stored products hang. Each of these products has a unique key that identifies them, as mentioned earlier, and within it, their two fields of expiration date and name.

4.2.5 DDBB connection

The application contains a folder with classes responsible for managing the connection with the database and all CRUD (Create, Read, Update & Delete) processes launched to it.

To establish the connection between the application and the database, it is necessary to specify the database's address and use credentials that ensure secure access. These credentials are stored in an external JSON file within the project, which the application accesses when it needs to initiate a connection. This JSON file is one of those included in the .gitignore file mentioned in the previous section to ensure secure access and avoid publicly sharing credentials.

4.2.6 Programming language

The programming language selected for the developed application is Python, a powerful and versatile language with a clear focus on productivity. It is widely known for its clear and readable syntax, cross-platform compatibility, and extensive library of ready-to-use modules.

4.2.7 Interface

To develop the application interface, a Python library called Tkinter has been used. This library is designed for creating and developing desktop applications, facilitating the positioning and development of a graphical interface.

Tkinter is an object-oriented layer based on Tk, a standard Graphical User Interface (GUI) tool. Building a view with Tkinter is based on hierarchically organized widgets. At the root of the interface is 'tk', and widgets are placed on it according to the application's needs.

Some of the commonly used widgets include tk.Frame, tk.Label, tk.Button, or tk.Text. Each of these widgets can be configured extensively, from colors and size to commands to be executed upon interaction. Composition can also be managed to determine the position and relationship between widgets.

In Figure 4.20, an example screenshot of the interface developed for this application is shown. In this case, it is a menu of options using widgets of type tk.Button on a tk.Frame:

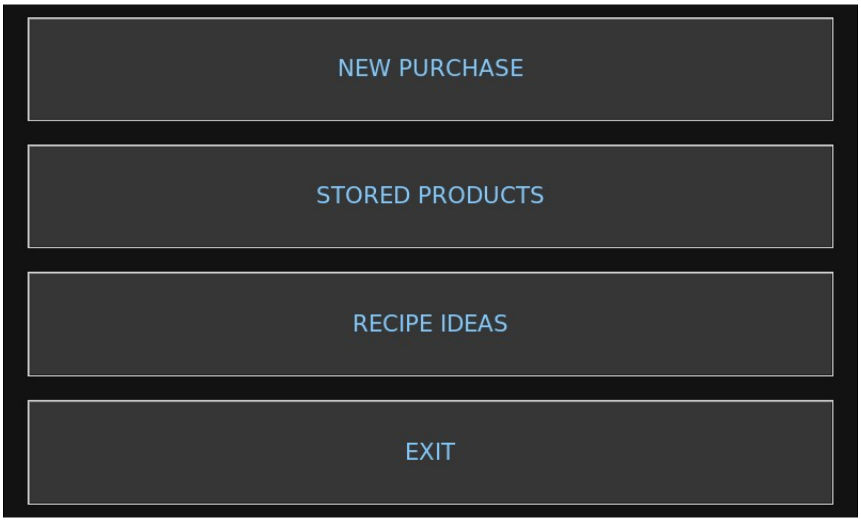
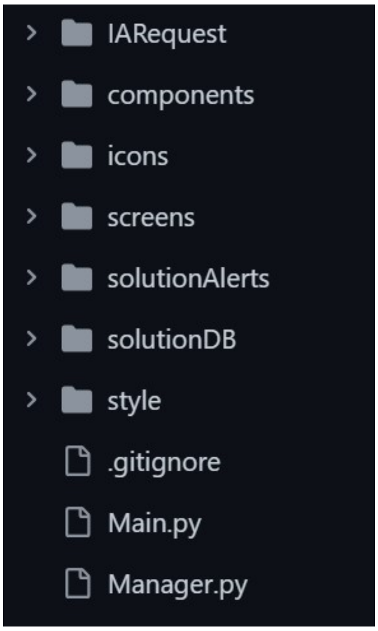


Figure 4.20: Developed Interface example

This interface allows the user to interact with the application in a simple and intuitive way, enabling them to use its functionalities.

4.2.8 Structure and code

To achieve the different functionalities of the application, the code is divided into different classes and folders. In Figure 4.21, the general structure of the application is shown to explain its operation and the purpose of each file. The screenshots belong to the GitHub repository, so the files included in the .gitignore are explained separately.



```
> IARequest
> components
> icons
> screens
> solutionAlerts
> solutionDB
> style
  .gitignore
  Main.py
  Manager.py
```

Figure 4.21: App structure

The **.gitignore** file has already been seen and explained earlier in the GitHub section.

Execution of the application is initiated from **Main**.**py**, where it checks if any alerts need to be sent and creates an instance of the Manager class.

Manager.py initializes and configures the interface with all its screens and positions, configures the logger instance for saving application logs (this logger will be detailed later), and contains common methods such as date validation, conversion to JSON format, or communication with the class responsible for the database when registering new products.

IARequest

The IARequest folder (Figure 4.22) contains files necessary to fulfil the "Recipe Ideas" functionality and its tests.

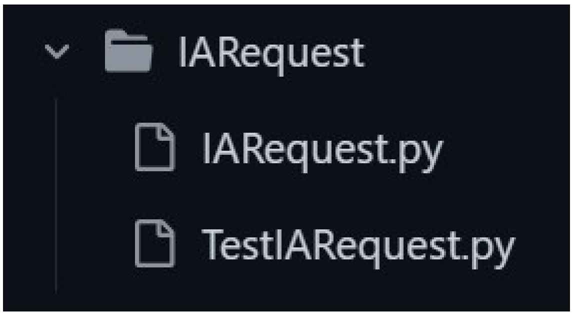


Figure 4.22: IARequest Folder

In the file **IARequest.py**, the connection to the OpenAI API is established. To do this, it is necessary to obtain the apiKey from the **API-KEY.json** file, which is located locally on the machine running the application. Once obtained, it can send requests to the API, and this process is detailed in a section later on.

Essentially, what this class receives is a prompt requesting a list of recipes with selected products to generate those "Recipe Ideas".

TestIARequest.py is intended to make these requests directly to the OpenAI API for testing purposes, without the need to run the entire application.

Components

Next, components folder (Figure 4.23).

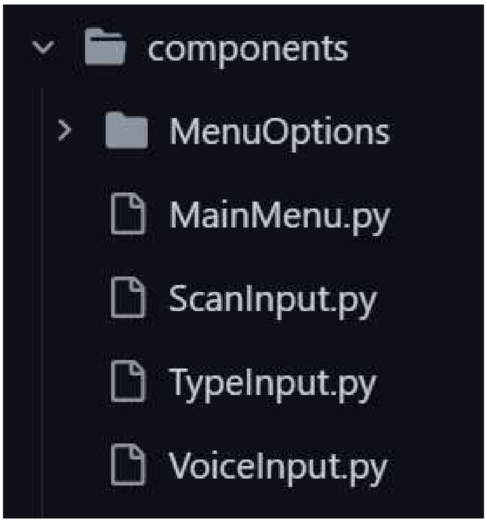
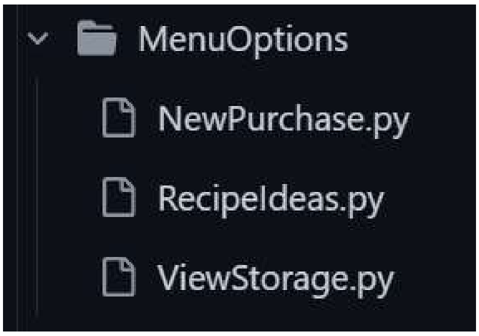


Figure 4.23: Components Folder

In the file **MainMenu.py**, all the options of the main menu are displayed through buttons. Different tkButton instances are created with their corresponding configurations and styles. These options include 'NEW PURCHASE', 'STORED PRODUCTS', 'RECIPE IDEAS,' and 'BACK' to exit and terminate the application's execution.

MenuOptions

The functionality of the main menu options is located in the MenuOptions folder (Figure 4.24).



MenuOptions
NewPurchase.py
RecipeIdeas.py
ViewStorage.py

Figure 4.24: components/MenuOptions Folder

The files **NewPurchase.py, RecipeIdeas.py, and ViewStorage.py** each have a very simple function: they preconfigure the initial screens of their respective options, applying styles and positions.

Within the 'NEW PURCHASE' option, there are three sub-options: Scan, Type, and Voice. The rest of the options in the main menu are detailed later.

ScanInput.py, TypeInput.py, and VoiceInput.py contain the three product registration methods available in the application. They share very similar structures, displaying their interfaces with the corresponding styles, allowing the user to register products in the system using different methods. They prompt the user to enter the product name and, when obtained, inquire whether they also want to enter the expiration date, checking that it is a valid date if selected.

As products are entered using any of the methods, a list of products is populated. When the user confirms the process, the application continues execution to register the product(s) in the database. The application's interface is modified based on the stage of the process, guiding the user on what to do next.

ScanInput

This method scans a barcode and queries information about it. Using the libcamera-still command with a timer, it captures an image and saves it as a file. It later retrieves this file to decode the barcode number using the pyzbar library. Once it has the number, it queries the OpenFoodFacts API and selects only the product name from the various fields that this API can return (nutritional values, brand, ingredients, etc.).

TypeInput

This method is the simplest of the three. It asks the user to type the name of the product using the tk.Entry widget, which is also used if the user wants to enter the expiration date of the product in any of the three possible methods.

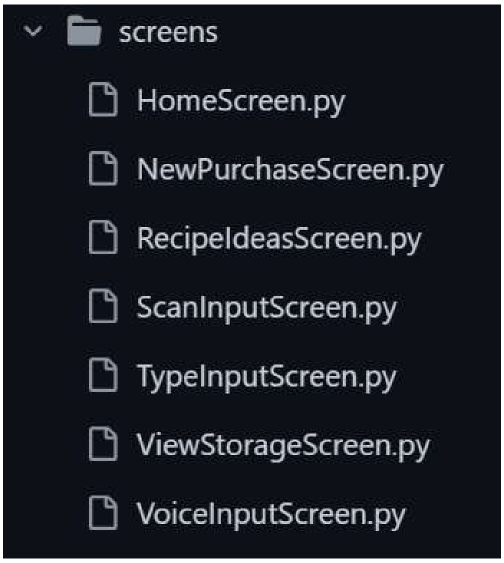
VoiceInput

This method is possibly the most convenient to use. It asks the user to say the name of the product out loud and captures the sound through the microphone. Then, thanks to the speech_recognition library, it identifies the mentioned name. It has a timeout in case it fails to recognize the product, displaying an informative message in this case.

There are different possible recognizers; after several tests, it has been observed that the most effective one is the Google recognizer, although it requires a very good internet connection to recognize audio correctly.

Screens

In the "screens" folder, different screens of the application are preconfigured in files (Figure 4.25), initializing their title, buttons, and other widgets with the corresponding styles.



screens

HomeScreen.py

NewPurchaseScreen.py

RecipeIdeasScreen.py

ScanInputScreen.py

TypeInputScreen.py

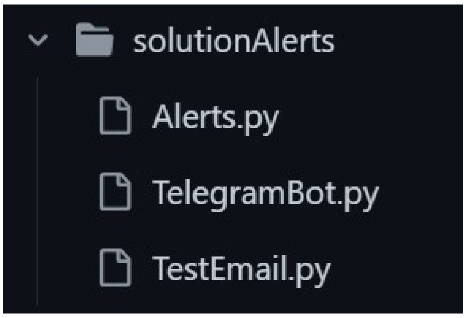
ViewStorageScreen.py

VoiceInputScreen.py

Figure 4.25: screens Folder

SolutionAlerts

The "solutionAlerts" folder (Figure 4.26) contains the necessary files to implement the functionality of sending emails and messages with a Telegram bot to users, notifying them of the approaching expiration dates of their products.



solutionAlerts
Alerts.py
TelegramBot.py
TestEmail.py

Figure 4.26: solutionAlerts Folder

From Alerts.py, first, it checks the date of the last sent alert, stored in the database. If this date matches the current date, the execution ends; otherwise, it updates the stored date with the current date and continues the execution as explained below. This ensures that no more than one alert is sent per day, as they would, in principle, provide the same information.

Then, it iterates through all the products stored in the system, comparing the current date with the expiration date of each of them, checking various possible cases. To send an alert, one of the following cases must be met:

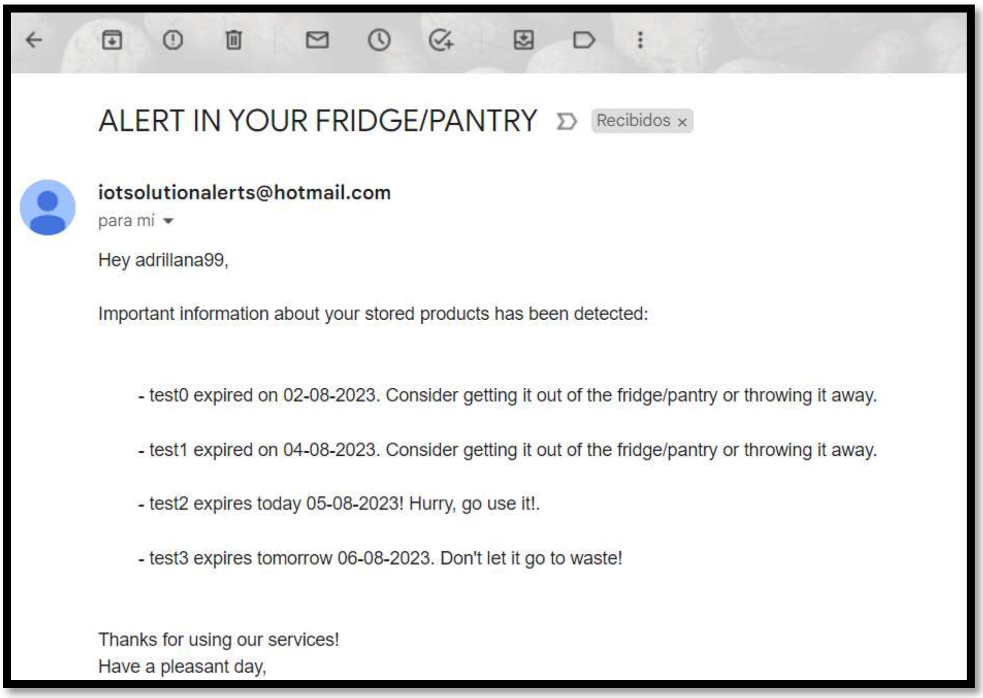
- The expiration date has already passed.
- The expiration date matches the current date.
- There is one day left until the product expires.
- There are two days left until the product expires.

For each of these cases, it sends a notification to the user with a different message, specifying the case.

The application has two methods for sending these notifications to users. To send emails, a specific address has been created for this project, iotsolutionalerts@hotmail.com. Both this address and its password are stored in another of the files included in the .gitignore file, in this case, userSettings.json, saved locally in the same folder as the team running the application. This file also contains the list of recipient email addresses that will receive the sent alert.

It decodes the JSON file to obtain the corresponding email addresses and sender's password. With this information and using the smtplib and email.mime libraries, it sets these values and the mail server to use, in this case, 'smtp-mail.outlook.com' on port 587. With all this and the already created message, it proceeds to send emails to each of the specified addresses, personalizing the message based on the recipient (a greeting with the name before the '@').

In Figure 4.27, an example of an email sent with the different types of possible messages is shown:



ALERT IN YOUR FRIDGE/PANTRY

iotsolutionalerts@hotmail.com

para mí

Hey adrillana99,

Important information about your stored products has been detected:

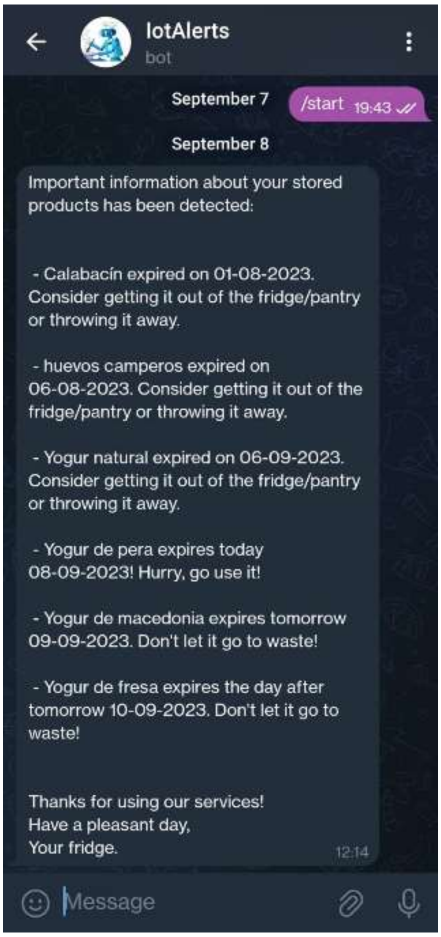
- test0 expired on 02-08-2023. Consider getting it out of the fridge/pantry or throwing it away.
- test1 expired on 04-08-2023. Consider getting it out of the fridge/pantry or throwing it away.
- test2 expires today 05-08-2023! Hurry, go use it!.
- test3 expires tomorrow 06-08-2023. Don't let it go to waste!

Thanks for using our services!
Have a pleasant day,

Figure 4.27: Alert email sent example

Additionally, a **Telegram bot** (@IotAlerts_bot) has been created through which the user can receive alert notifications. Once the message is created, the file **TelegramBot.py** is used. From this file, the **userSettingsTelegram.json** file is decoded (also included in .gitignore) to obtain the unique TOKEN that identifies the bot and a list of CHAT_IDs, which correspond to the users who will receive the notification. With this information, it iterates through the list of IDs, and each of the notifications is sent. For all this, the python-telegram-bot library has been installed.

In Figure 4.28, an example of an alert sent through this bot is observed.



September 7

/start 19:43

September 8

Important information about your stored products has been detected:

- Calabacín expired on 01-08-2023. Consider getting it out of the fridge/pantry or throwing it away.

- huevos camperos expired on 06-08-2023. Consider getting it out of the fridge/pantry or throwing it away.

- Yogur natural expired on 06-09-2023. Consider getting it out of the fridge/pantry or throwing it away.

- Yogur de pera expires today 08-09-2023! Hurry, go use it!

- Yogur de macedonia expires tomorrow 09-09-2023. Don't let it go to waste!

- Yogur de fresa expires the day after tomorrow 10-09-2023. Don't let it go to waste!

Thanks for using our services!
Have a pleasant day,
Your fridge.
12:14

Figure 4.28: Telegram bot screenshot

With TestEmail.py, tests for these functionalities have been conducted with examples without the need to run the complete application.

SolutionDB

The solutionAlerts folder (Figure 4.29) contains the files necessary to fulfill the functionality of connecting to the database and performing CRUD operations from the application. These functionalities are implemented in **DataBase**.**py** and have been previously explained in detail in the 'DDBB' Connection' section.

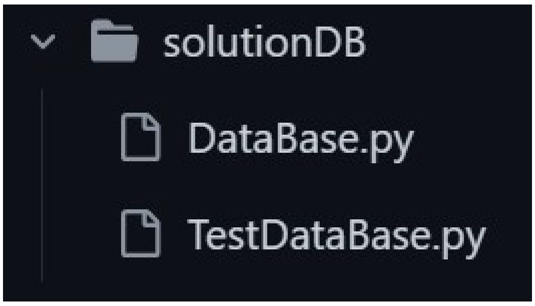


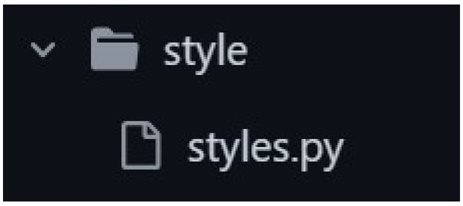
Figure 4.29: solutionDB Folder

Tests for this functionality have been conducted using **TestDataBase.py** without the need to run the entire application.

Style

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

To manage styles from a single location, the style folder (Figure 4.30) is used.

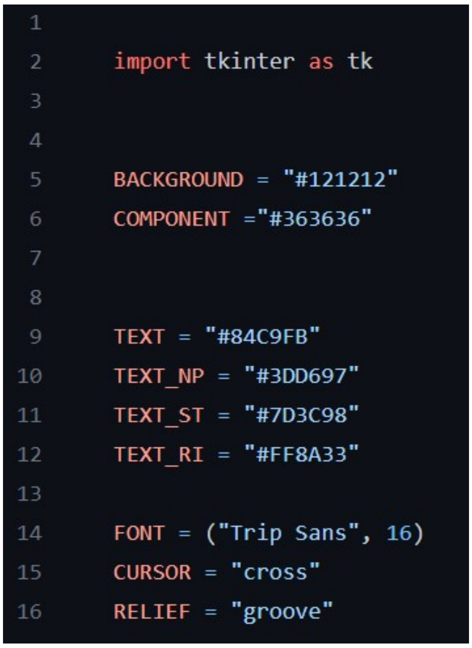


```
style
    styles.py
```

Figure 4.30: styles Folder

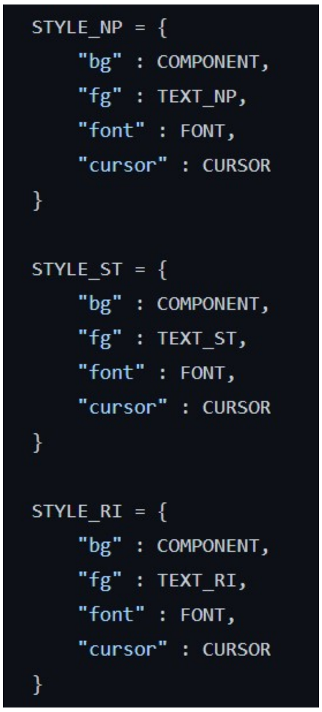
The use of **styles.py** helps centralize the application of styles to the elements of the interface, whether it's size, color, position, or other style configurations. Common configurations for the entire application are defined within it.

For the development of the interface, an attempt has been made to maintain a minimalist style to facilitate its use, with small differentiating details. For example, the background color and the main color of the components are common throughout the application, but each of the 3 main options and their entire interior is configured with a specific text color, as shown in Figures 4.31 and 4.32. As a clarification for this latter Figure, 'NP' refers to New Purchase, 'ST' refers to Stored Products, and 'RI' refers to Recipe Ideas.



```

import tkinter as tk


BACKGROUND = "#121212"
COMPONENT ="#363636"


TEXT = "#84C9FB"
TEXT_NP = "#3DD697"
TEXT_ST = "#7D3C98"
TEXT_RI = "#FF8A33"

FONT = ("Trip Sans", 16)
CURSOR = "cross"
RELIEF = "groove"
```

Figure 4.31: Common configurations



```
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
}
```

Figure 4.32: Each App part configuration

To verify the application of these configurations, sample images can be seen in Figures 4.33, 4.34, 4.35, and 4.36.

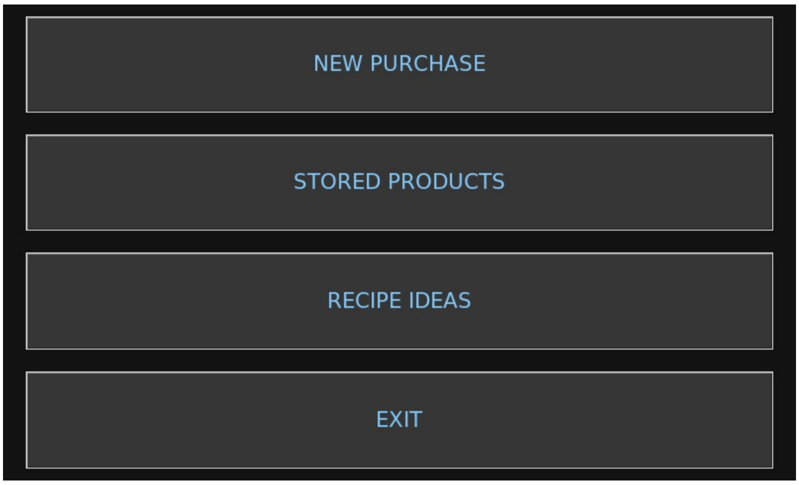


Figure 4.33: Main menu applied style

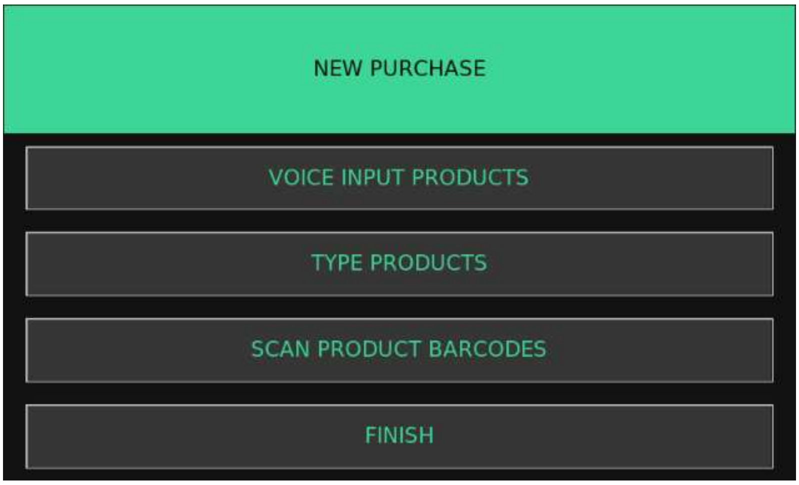


Figura 4.34: 'NEW PURCHASE' style

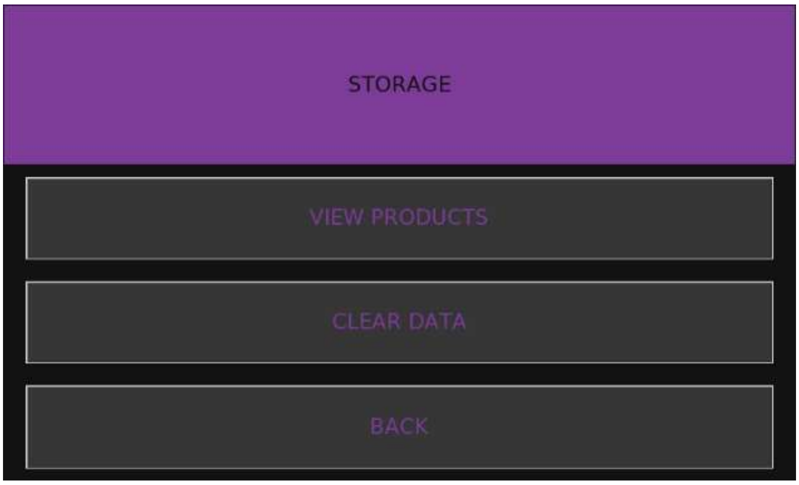


Figura 4.35: 'STORAGE PRODUCTS' style

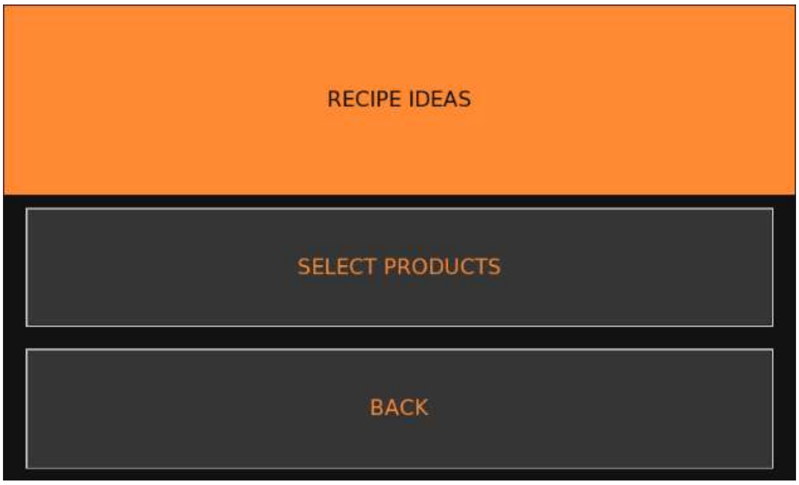


Figura 4.36: 'RECIPE IDEAS' style

4.2.9 External services

External services enrich applications, allowing for greater innovation and expanding their functionalities.

In this project, to ensure that the application implements all the proposed functionalities, it has been necessary to use several of these external services to communicate and work with. Two APIs (Application Programming Interface), a cloud-hosted database, a Telegram bot created from scratch, and a mail server were utilized.

Each of these external services is detailed below.

Openai API

This project utilizes the OpenAI API (Figure 4.37), which enables the use of the artificial intelligence chatbot developed in 2022, specialized in dialogue, ChatGPT.

To generate a request to this API, it is necessary to specify the text or prompt for the query, the engine to use, and the maximum number of tokens it accepts. This request corresponds to an interaction with ChatGPT 3.

To use this API, it is also necessary to create an account and generate an associated API key. Each request has a cost of $0.002 per 1,000 tokens. However, in the first few months, OpenAI provides a $5 credit for testing purposes, and this has been used in this project.

Figure 4.37: OpenAI logo

OpenFoodFacts API

OpenFoodFacts (Figure 4.38) is an open and collaborative database of food products from around the world, with approximately 300,000 records. It contains information about each product, including nutritional values, ingredients, brand, quantity, packaging type, certified labels, countries of sale, and many other details. However, for the current project, only the product names are utilized.[10]

Figure 4.38: OpenFoodFacts logo

The API request made has the following format:

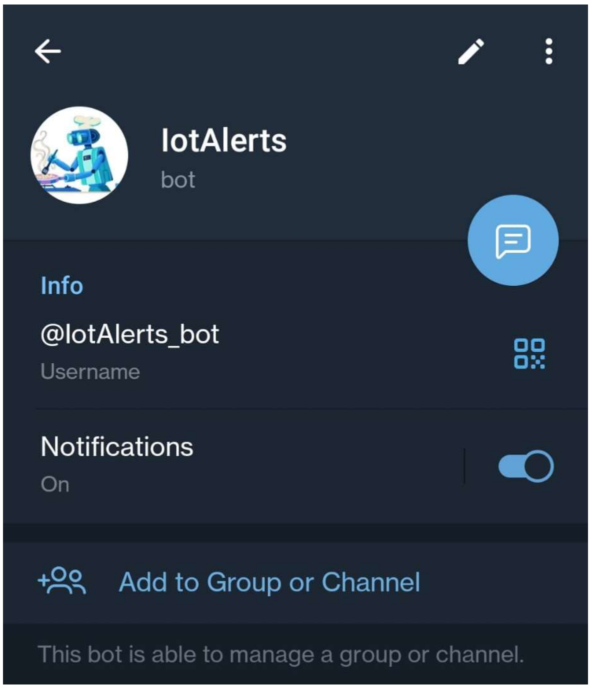
“https://world.openfoodfacts.org/api/v0/product/” + [productID], replacing [productID] with the barcode identifies by the app.[11]

Cloud Database

The cloud-hosted database used, as explained in detail earlier, is also an external service utilized by the application. Its address is as follows: https://iot-solution-8d63c-default-rtdb.europe-west1.firebasedatabase.app/

Telegram bot

Creating a Telegram bot is not a very complicated process; it only requires having a registered user on the messaging platform. Each bot has a unique token, and each user has a unique chat_id. With these two values, it has been possible to use the Python application of the project to manage the sending of alerts/notifications through the created bot. In Figure 4.39, the information of the bot used in this project is shown, and its invitation link is as follows: http://t.me/IotAlerts_bot


IotAlerts
bot

Info
@IotAlerts_bot
Username

Notifications
On

Add to Group or Channel

This bot is able to manage a group or channel.

Figure 4.39: Telegram bot Information

SMTP Outlook Server

As already explained in the alerts section, the application uses an SMTP email server, which is also an external service.

4.2.10 Loggin

Logs or records are 'traces' left by the execution of a program in the form of texts or messages. They can be of different levels depending on what you want to record: errors, warnings, information, etc.

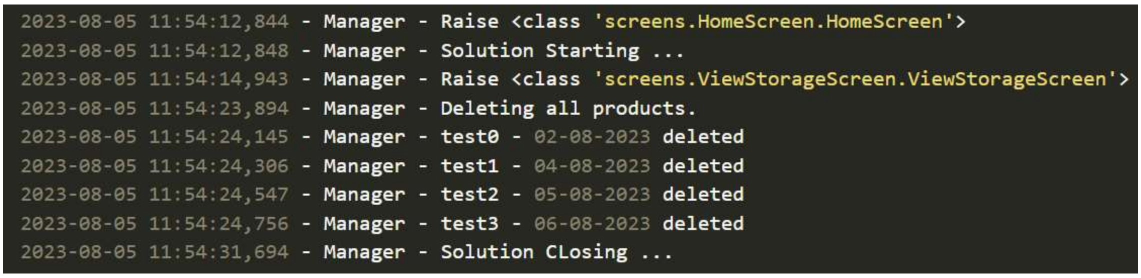
Their use in an application is crucial as it allows for analysis and explanation of the behavior followed during execution. It is a way to identify problems more directly.

In this project, the use of logs has been implemented at specifically selected points to assess the overall system behavior and detect operational errors. These logs are stored in another file included in the .gitignore called app.log.

In this file are stored logs with the following format:

"Date and time – Executed Class - Action made".

Below is an example of logs from an execution testing the deletion of all products stored in the system at once, as shown in Figure 4.40.



```
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 ...
```

Figure 4.40: Execution logs example

5. Conclusions

In conclusion, the initially set objectives of this project have been achieved. This was accomplished by following the planned paths and strategies outlined during the project analysis, as well as exploring new approaches that emerged during its development.

A prototype has been created that can transform a regular refrigerator into one with some features of a smart fridge. The main value of this project lies in its affordability for users. Current smart fridges on the market are often too expensive and are typically ruled out by the majority of consumers for this reason.

Due to its expected low cost and universal compatibility with any refrigerator, it becomes feasible to bring the advantages of a smart fridge to a broader audience. Moreover, this project has been designed to be easily scalable, allowing for the addition of new features and component improvements. Further details on these ideas and other future work are provided in the following section.

6. Future additions

Technology has never been a static field; it is always evolving. Therefore, in this section, various possibilities are explored to expand and optimize the functionalities of this project.

6.1 Using external services

As mentioned earlier, the project uses external services to complement some of its functionalities, and some of these services offer many more possibilities than those currently utilized. Two examples are proposed:

- OpenFoodFacts API: In addition to providing the name of the product scanned from the barcode, the OpenFoodFacts API allows retrieving a wealth of information about the product, including nutritional information, ingredients, allergens, packaging materials, environmental certifications, etc. All this information can be very useful for following specific diets for intolerances, athletes, or improving the overall health of individuals.
- Database Enhancement: The project's database is currently used to store products with their expiration dates and the date of the last alert sent by the system. Leveraging the existing communication with the prototype, the database could be expanded to record more data as the application is used. For instance, it could track consumed or wasted products, enabling subsequent studies on consumption patterns.

6.2 Long-term functionalities

With the potential for large-scale commercialization of the prototype, there would be an opportunity to develop somewhat more complex but highly interesting functionalities.

If the user base were to increase significantly, aggregating the substantial amount of data generated by users through the application, it would be possible to develop algorithms and establish behavioural patterns when it comes to consuming and replenishing food items. Leveraging artificial intelligence and machine learning with all this data could lead to a much more efficient automation of purchases, their frequency, and quantities.

6.3 System configurations

A user using the application does not have direct access to system settings such as the list of email addresses or Telegram users to whom alerts are sent. It would be possible to create a section in the application from where to manage this information and, for example, the frequency and type of alerts sent by the system.

6.4 Components improvements

This prototype can be perfectly considered as an embedded system. When developing such systems, the selection of components that will integrate it is crucial.

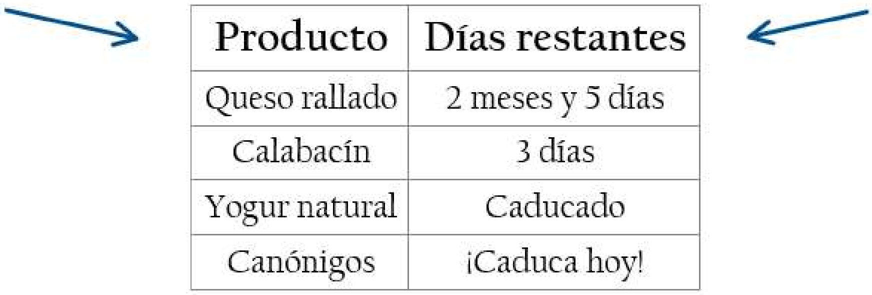
Opting for purpose-specific components, optimized for a single function, provides better performance compared to general-purpose components, which can perform a greater number of tasks but with worse results.

The components used in this prototype have mostly been of general purpose. However, if the manufacturing of a large number of units is planned, it is often advisable to lean towards integrating purpose-specific components.

6.5 Interface improvements

The interface of this project has been developed with usability in mind, but there are several visual aspects that could be added to offer a more satisfying and clear user experience. As examples, the following improvements are suggested:

- When displaying the products in the system, differentiate the information shown in columns. For example, 'product name' and 'days remaining' (showing the number of days remaining until the expiration date), similar to the format shown in Figure 6.1. This provides more information to the user while using the application.

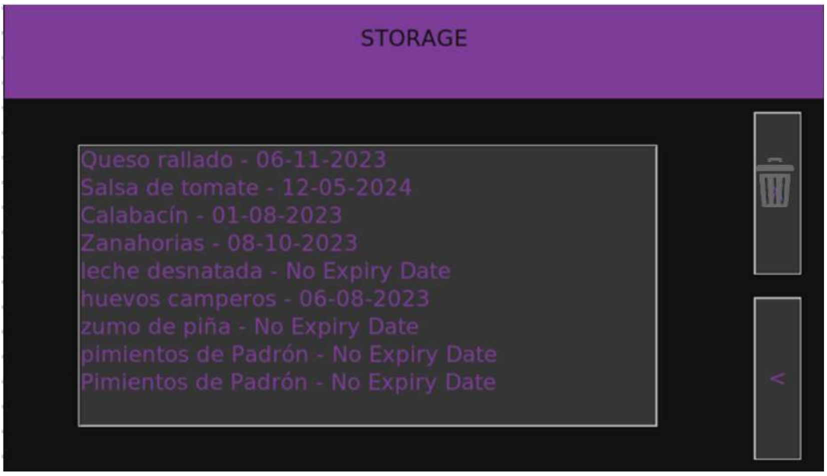


Producto	Días restantes
Queso rallado	2 meses y 5 días
Calabacín	3 días
Yogur natural	Caducado
Canónigos	¡Caduca hoy!

Figure 6.1: Product table possible improvement

- The use of icons on certain buttons or images throughout the execution of the application. An example of an icon that could be used is shown in Figure 6.2, replacing the delete button for a product in the 'Stored Products' section.

Figure 6.2: Delete icon



Queso rallado - 06-11-2023
Salsa de tomate - 12-05-2024
Calabacín - 01-08-2023
Zanahorias - 08-10-2023
leche desnatada - No Expiry Date
huevos camperos - 06-08-2023
zumo de piña - No Expiry Date
pimientos de Padrón - No Expiry Date
Pimientos de Padrón - No Expiry Date

Figure 6.3: Icon test example

7. Bibliography

[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

[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/

[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/adrillb/iot-solution-tfg

[13] Bot de Telegram creado

http://t.me/IotAlerts_bot

[14] Database empleada

https://iot-solution-8d63c-default-rtdb.europe-west1.firebasedatabase.app/

8. Annex

8.1 Annex I – Project Cost

8.1.1 Material cost

The material cost (Table 3) of this project is divided into two sections: Hardware Cost (Table 1) and Software Cost (Table 2).

DESCRIPTION	QUANTITY	INDV. COST	TOTAL
Raspberry Pi 4 Model B	1	85€	85€
Touchscreen	1	43,95€	43,95€
Camera module	1	34,70€	34,70€
USB Microphone	1	10,10€	10,10€
Remote control	1	23,99€	23,99€
Heat dissipater	1	11,99€	11,99€
TOTAL			**209,03€**

Table 1. Hardware cost

DESCRIPTION	QUANTITY	INDV. COST	TOTAL
Raspberry Pi OS	1	0€	0€
GitHub	1	0€	0€
Visual Studio Code	1	0€	0€
Python			
Realtime Firebase Database	1	0€	0€
Outlook mail Server	1	0€	0€
API OpenAI	1	0€	0€
API OpenFoodFacts	1	0€	0€
VIM	1	0€	0€
JSON file format	1	0€	0€
TOTAL			**0€**

Table 2. Software cost

DESCRIPTION	**TOTAL**
Hardware Cost	209,03€
Software Cost	0€
Total	**209,03€**

Table 3. Material cost

8.1.2 Personal cost

This project personal cost is detailed in Table 4.

CONCEPT	**HOURS**	**COST/HOUR**	**TOTAL COST**
Requirements analysis	20	30€	600€
Prototype Design	10	30€	300€
Prototype integration	5	30€	150€
Software design and development	10	30€	300€
User interface design	15	30€	450€
Input methods development	75	30€	2.250€
DDBB connection and management	10	30€	300€
Alert system development	15	30€	450€
Log system development	5	30€	150€
APIs requets management	20	30€	600€
Code repository creation and management	10	30€	300€
Testing and validation	15	30€	450€
Documentation	70	30€	2.100€
Project management	10	30€	300€
Total			**8700€**

Table 4. Personal cost

8.1.3 General costs

The overall costs of this project are detailed in Table 5. These costs include variable expenses such as computer equipment, electricity, internet, office supplies, travel, etc. This amount is estimated at 15% of the personnel and material costs of the project.

General costs	1336,36€

Table 5. General costs

8.1.4 Coste total

The total cost of the project is shown and detailed in Table 6.

DESCRIPTION	**TOTAL**
Material costs	209,03€
Personal costs	8700€
General costs	1336,36€
TOTAL	**10245,39€**

Tabla 6. Total cost

The total cost of the project amounts to the sum of TEN THOUSAND TWO HUNDRED FORTY-FIVE POINT THIRTY-NINE.

8.2 Annex II – User Manual

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.

8.2.1 Main menu

App main menu is shown in Figure 8.1.

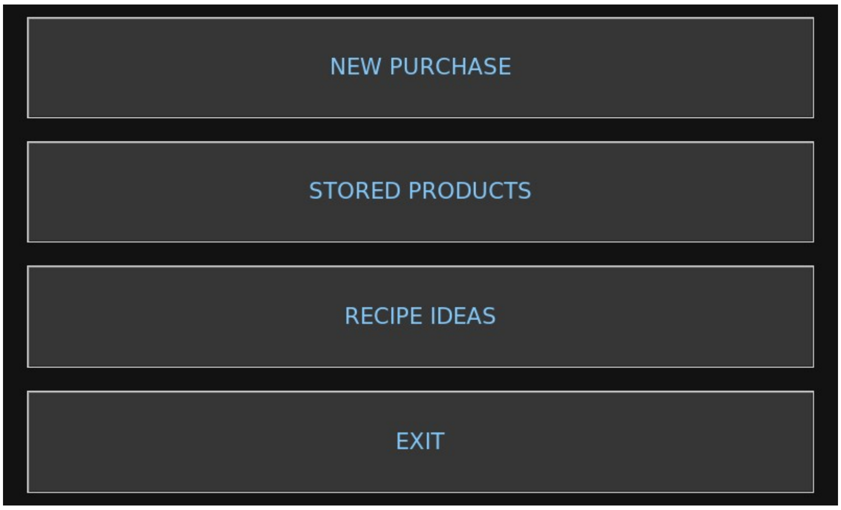


Figure 8.1:App main menu

From this screen, the user has the possibility to access three differentiated options or close the application. These first three options are:

- **NEW PURCHASE** – From this section, products are added to the system after each purchase, entering the product name and, optionally, the expiration date. The application offers three different registration methods, detailed below. From this section, you can also go back to the main screen.
- **STORED PRODUCTS** – In this section, the user can view all products present in the system, with their expiration date if available. In this section, it is also possible to delete products individually one by one or, if preferred, all of them. From this section, you can also go back to the main screen.
- **RECIPE IDEAS** – In this option, the user can select a list of products from those present in the system, and when the selection is complete, request 5 recipe ideas that incorporate the selected items. From this section, you can also go back to the main screen.

Choosing the **EXIT** button will terminate the execution of the application.

8.2.2 NEW PURCHASE

The main screen of this option displays the following options as methods for registering products in the system (Figure 8.2).

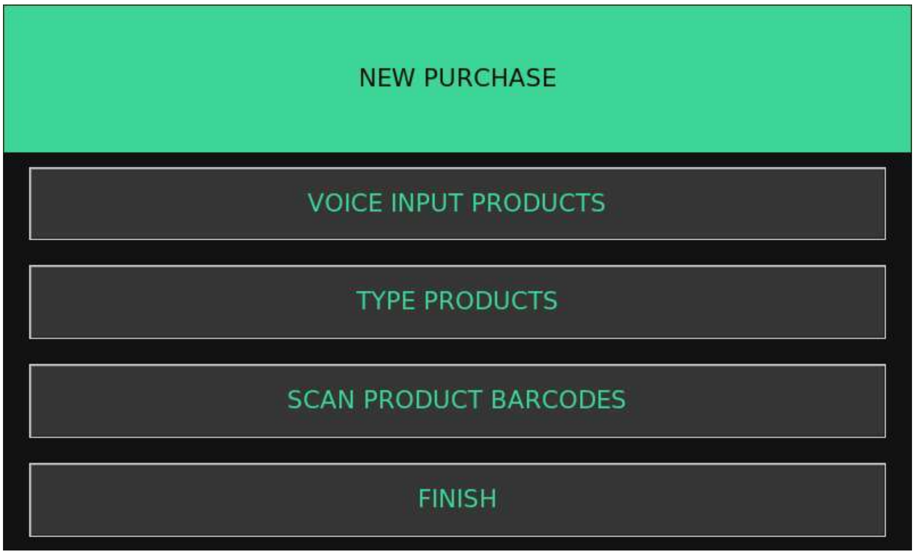


Figure 8.2: 'NEW PURCHASE' Main screen

By clicking on the FINISH option, the application will return to the main menu.

Voice Input Products - By selecting this option, the application prepares for product insertion through voice recognition and displays the screen shown in Figure 8.3.

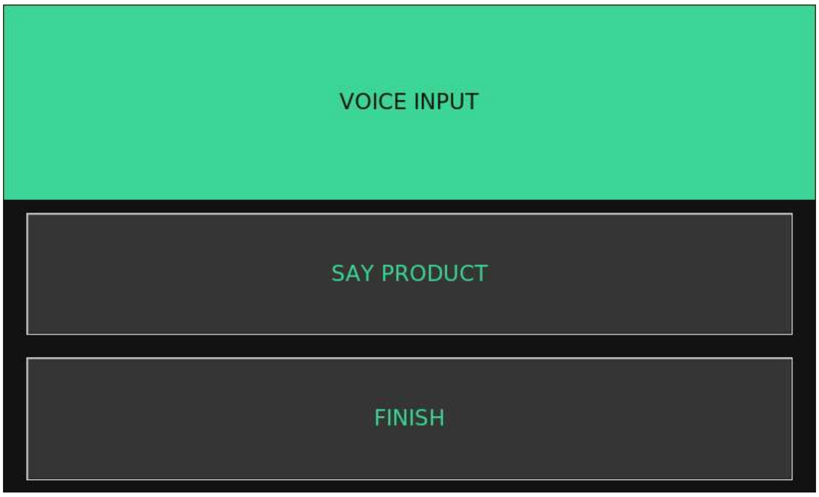


Figure 8.3: Voice recognition preparation

When the user is ready, they press the **SAY PRODUCT** option, and the application displays the screen shown in Figure 8.4:

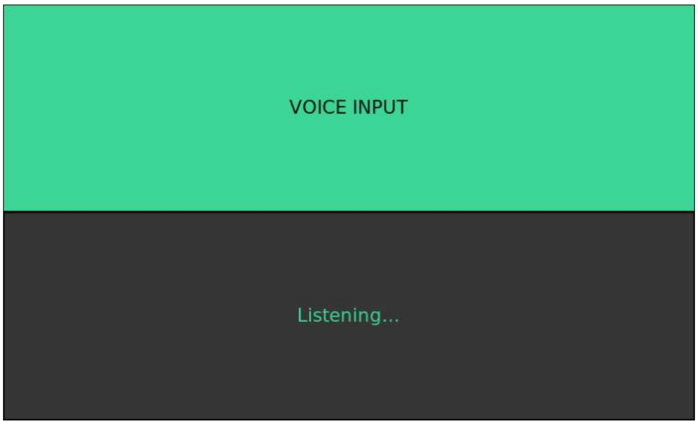


Figure 8.4: Screen while app is listening

While this screen is displayed, the application is in listening mode until it detects and recognizes the mentioned product. When it does, it will display the confirmation screen shown in Figure 8.5.

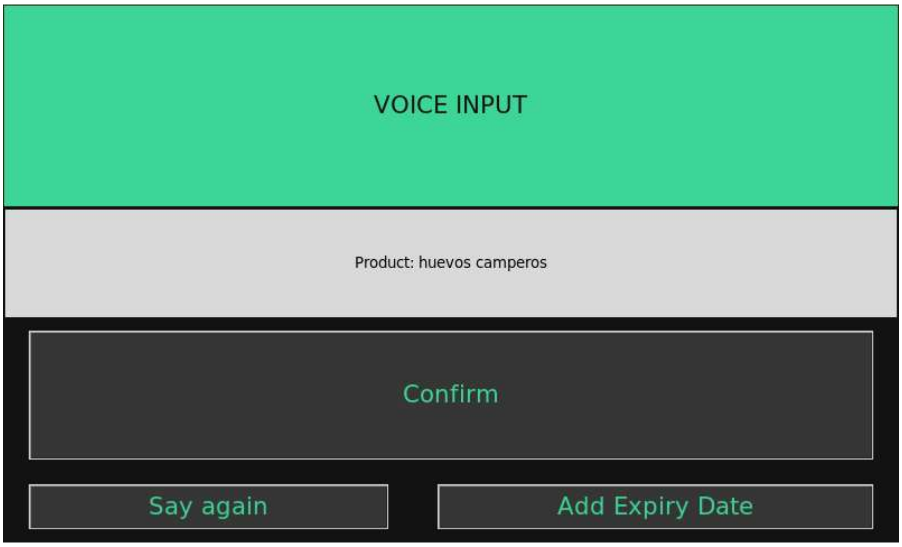
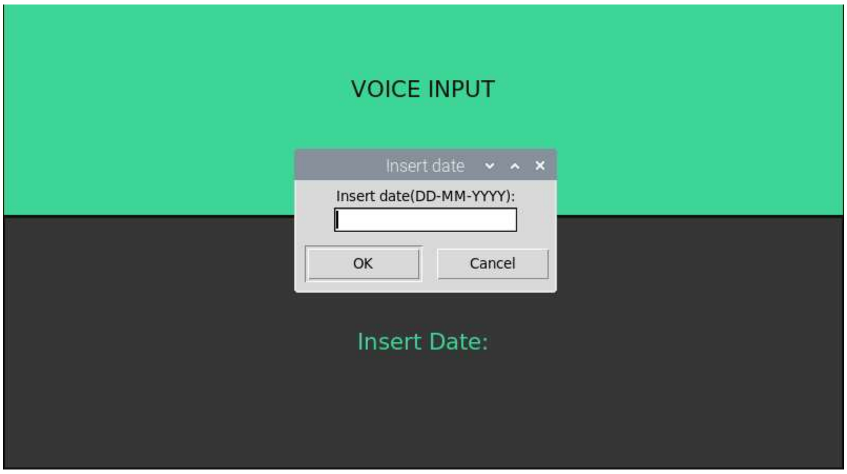


Figure 8.5: Voice recognition configuration screen

From here, three possible options are presented:

- **Say again**: This option will restart the listening process in case what the application recognized is incorrect, or the user wishes to repeat it.
- **Confirm**: This option will add the product to the list of products with a default expiration date of '00-00-0000', and the product registration process will continue.
- **Add Expiry Date**: This option will prompt the user to enter the expiration date of the product, showing the screen in Figure 8.6.



VOICE INPUT

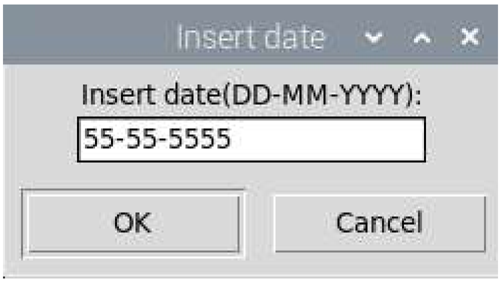
Insert date

Insert date(DD-MM-YYYY):

Insert Date:

Figure 8.6: Insert Date screen

The application will validate the entered date, so in case of entering an incorrect date, as shown in Figure 8.7, an error message will be displayed (Figure 8.8), requesting a valid date to be entered:

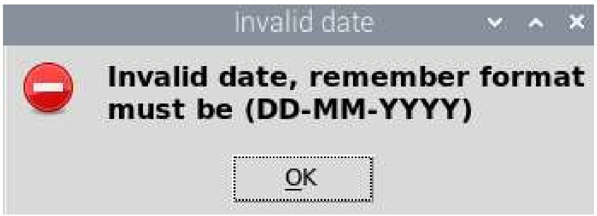


Insert date

Insert date(DD-MM-YYYY):

55-55-5555

Figure 8.7: Wrong date example



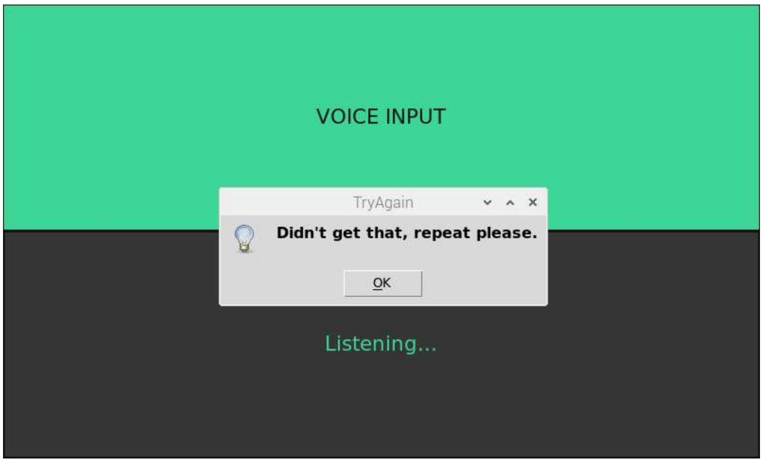
Invalid date

Invalid date, remember format must be (DD-MM-YYYY)

Figure 8.8: Wrong date message

When a valid date is entered, the product with its expiration date will be added to the list of products, and the registration process will continue.

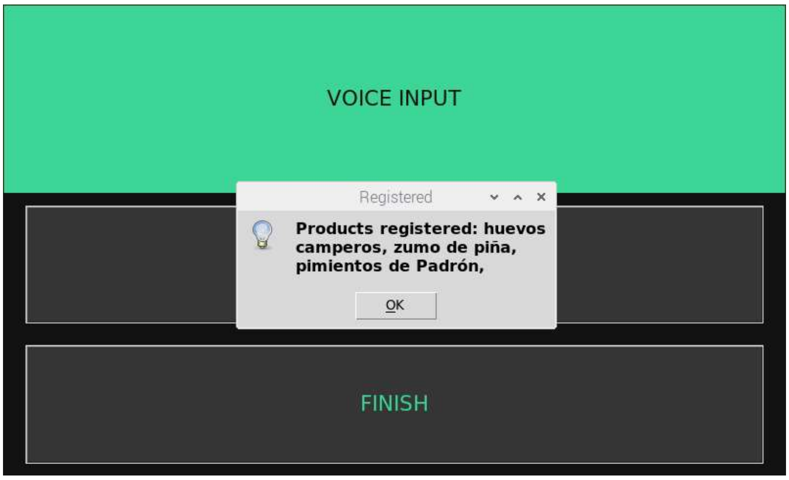
If the application is unable to recognize any result, it proceeds to display the warning message in Figure 8.9:



Didn't get that, repeat please.

Figure 8.9: Voice recognition fail message

At the moment the user finishes entering products, they must press the **FINISH** option. At this moment, the application will record the entered information in the database and display the following informative message (Figure 8.10).



Products registered: huevos camperos, zumo de piña, pimientos de Padrón,

Figure 8.10: Registered products message

After this, you will return to the main menu of the application, even if no product has been entered.

Type Products – By selecting this option, the application is ready for product entry using the keyboard and will display the screen in Figure 8.11.

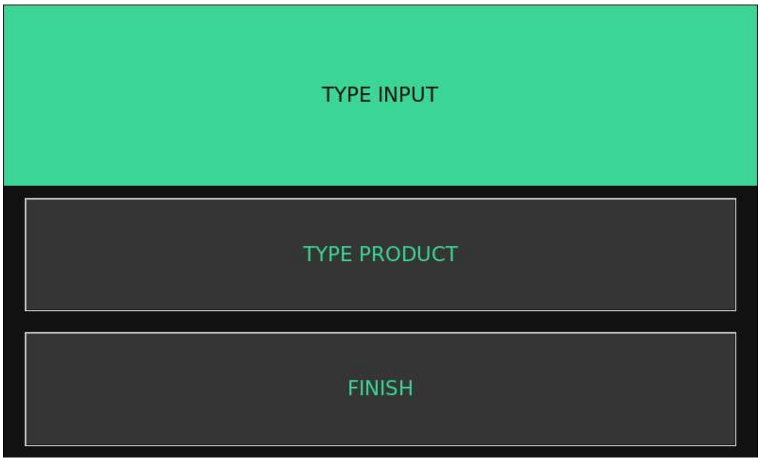


Figure 8.11: Typing input preparation

When the user clicks on **TYPE PRODUCT**, the following screen is displayed (Figure 8.12), and the system is ready to register products.

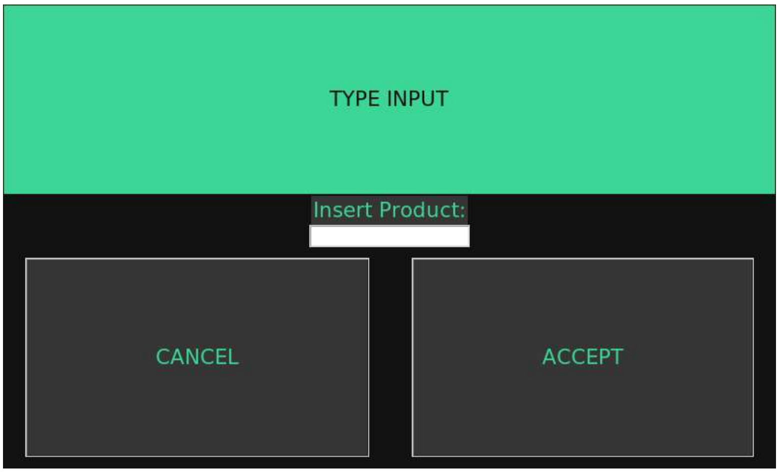
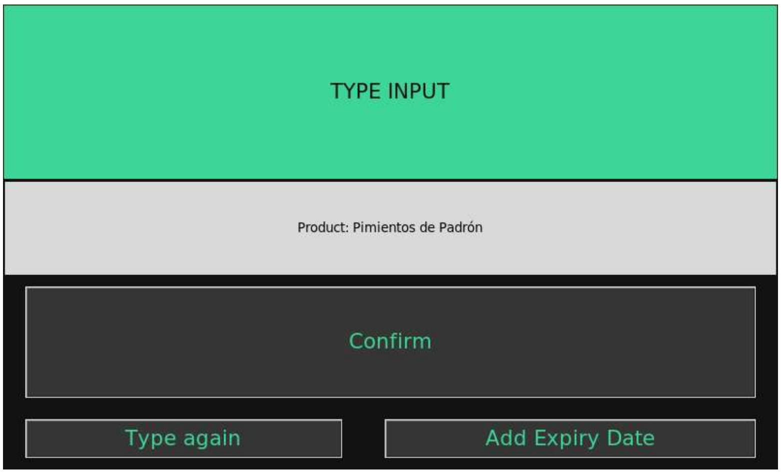


Figure 8.12: Type input screen

Once the product name is entered, the following screen is displayed (Figure 8.13) with different options to choose from:



Product: Pimientos de Padrón

Figure 8.13: Type input confirmation screen

In this screen, the user can select "**Type again**" to restart the process and enter the product name again. The "**Confirm**" option adds the product to the list and continues the insertion process. The "**Add Expiry Date**" option initiates the process in the same way as in the first voice recognition method.

Like the previous method, when the process is completed, the application will record the information entered by the user in the database.

Scan Product Barcodes – When selecting this option, the application is ready for product insertion through barcode scanning and will display the following screen (Figure 8.14).

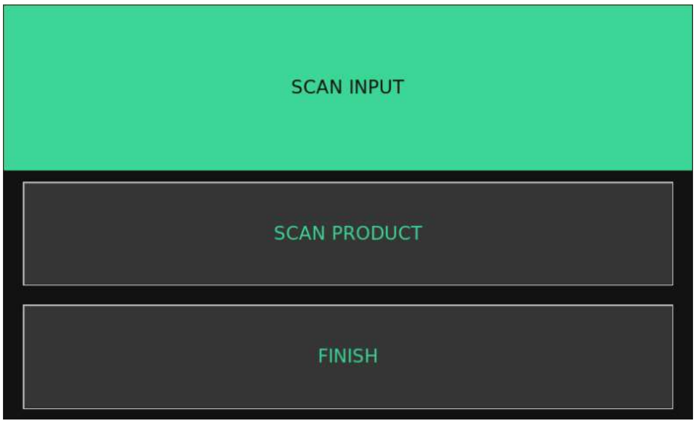
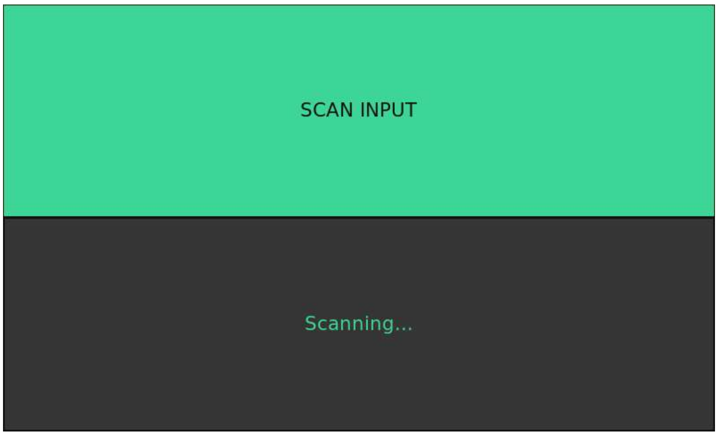


Figure 8.14: Scan input preparation

When the user is ready, they press the "**SCAN PRODUCT**" option, and the application displays the following screen (Figure 8.15).



SCAN INPUT

Scanning...

Figure 8.15: Screen while app is scanning

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.

While this screen is displayed, the application is in scanning mode until it detects and recognizes a barcode.

Similar to the previous methods, at this point, the user can choose three options: "**Scan again**" to restart the process, "**Confirm**" to add the product to the list, or "**Add Expiry Date**" to register the product with an expiration date.

8.2.3 STORED PRODUCTS

The main screen of this section (Figure 8.16) displays three options: view the products stored in the system, delete all products, or go back to the main menu of the application.

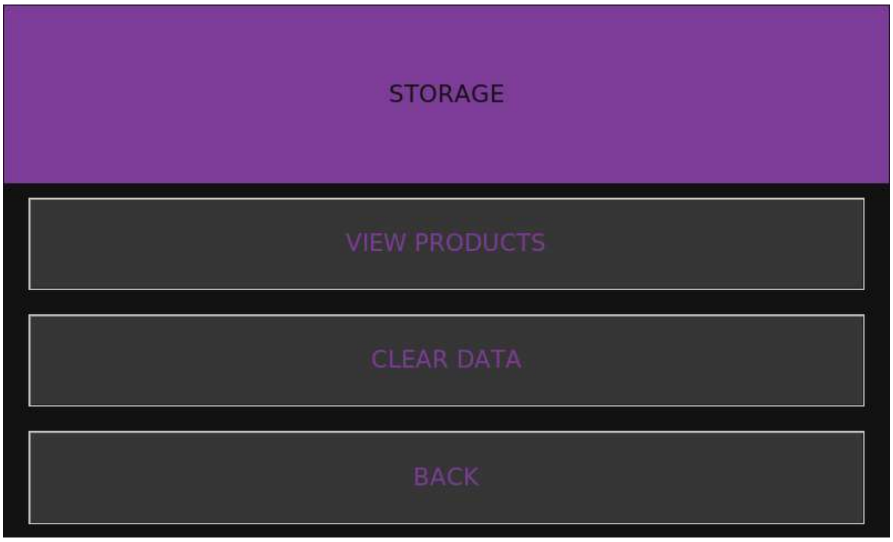
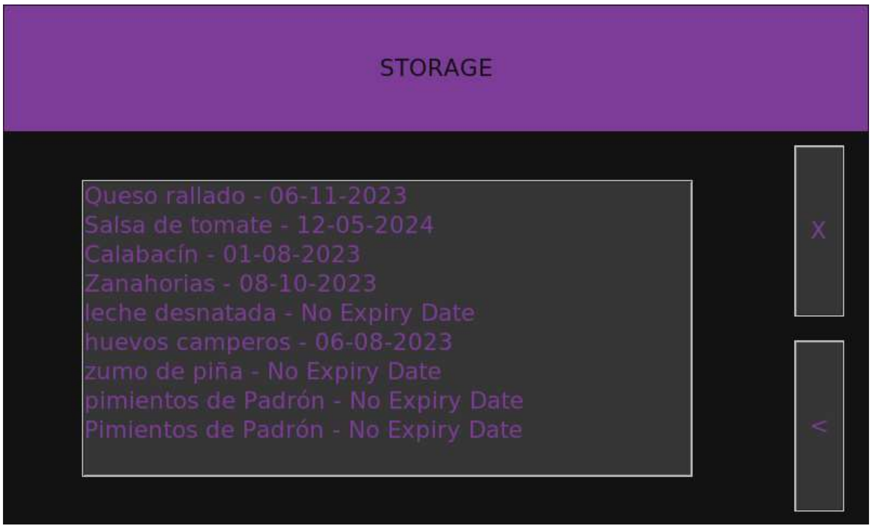


Figure 8.16: 'STORED PRODUCTS' main screen

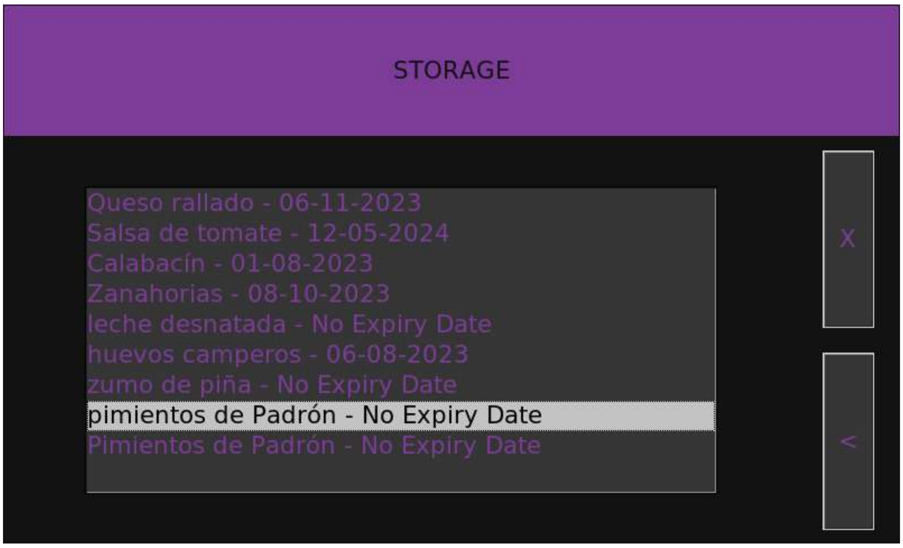
If the user selects the View Products option, the following screen will be displayed (Figure 8.17).



Queso rallado - 06-11-2023
Salsa de tomate - 12-05-2024
Calabacín - 01-08-2023
Zanahorias - 08-10-2023
leche desnatada - No Expiry Date
huevos camperos - 06-08-2023
zumo de piña - No Expiry Date
pimientos de Padrón - No Expiry Date
Pimientos de Padrón - No Expiry Date

Figure 8.17: Stored products visualization

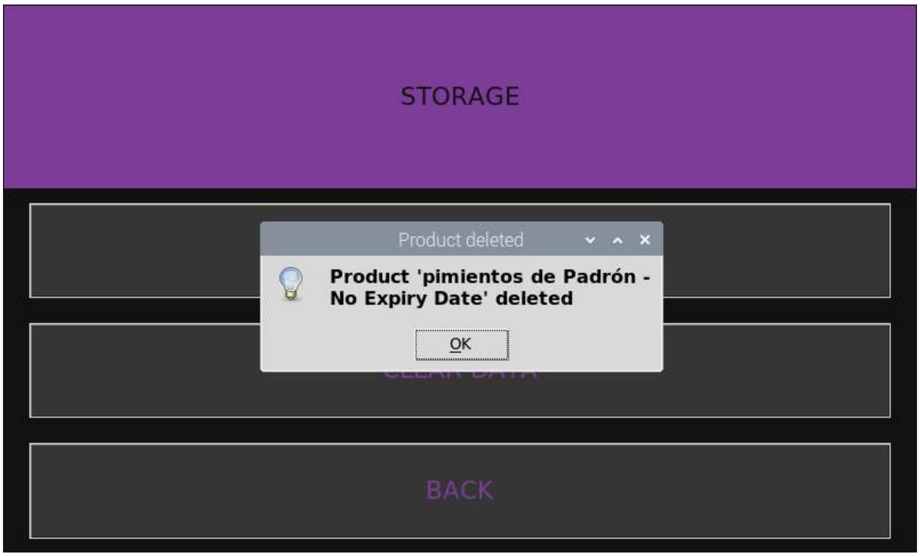
In this screen, the stored products are displayed with their corresponding expiration dates. Alongside this list, there are two buttons: the bottom one '<' goes back to the previous step, the main screen of this section. The top button '**X**' is used to delete a product individually from the database. To do this, the user must select one of the products in the list, as shown in Figure 8.18.



Queso rallado - 06-11-2023
Salsa de tomate - 12-05-2024
Calabacín - 01-08-2023
Zanahorias - 08-10-2023
leche desnatada - No Expiry Date
huevos camperos - 06-08-2023
zumo de piña - No Expiry Date
pimientos de Padrón - No Expiry Date
Pimientos de Padrón - No Expiry Date

Figure 8.18: Product selection

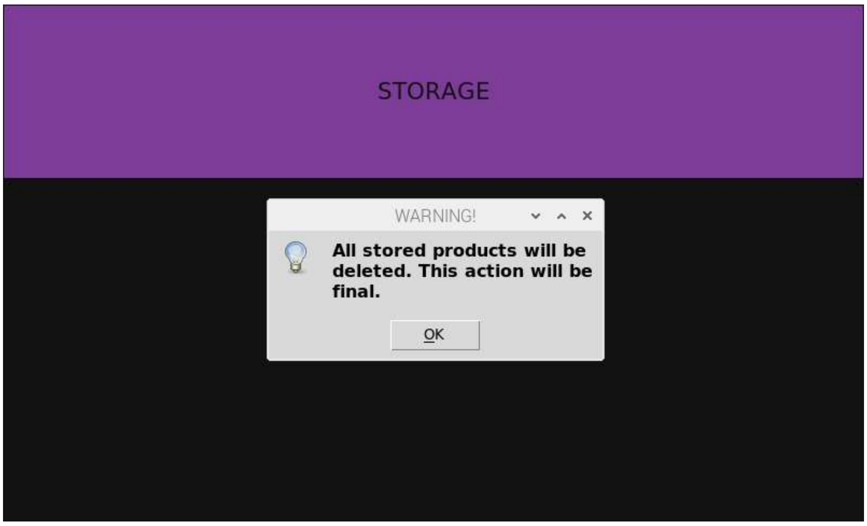
Una vez seleccionado, pulsar en dicho botón, provocando la eliminación del producto de la base de datos y la visualización del siguiente mensaje (Figura 8.19):



Product 'pimientos de Padrón - No Expiry Date' deleted

Figure 8.19: Deleted product informative message

On the other hand, if the user selects the Clear Data option, the application warns that accepting it will delete all products from the system, with no possibility of recovery, displaying the following message (Figure 8.20):



All stored products will be deleted. This action will be final.

Figure 8.20: Warning message pre-delete

After that warning message, the next screen is displayed, where the user decides whether to continue with the deletion process or return to the main screen of the section (Figure 8.21):

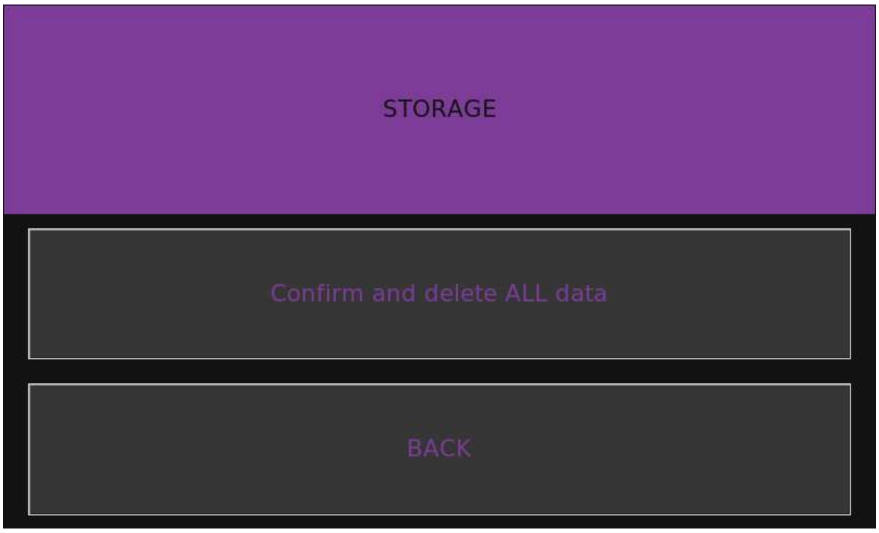


Figure 8.21: Screen to delete every stored product

8.2.4 RECIPE IDEAS

The main screen of this section (Figure 8.22) displays two options: go back to the initial menu of the application or proceed to the next screen where you can find recipe ideas:

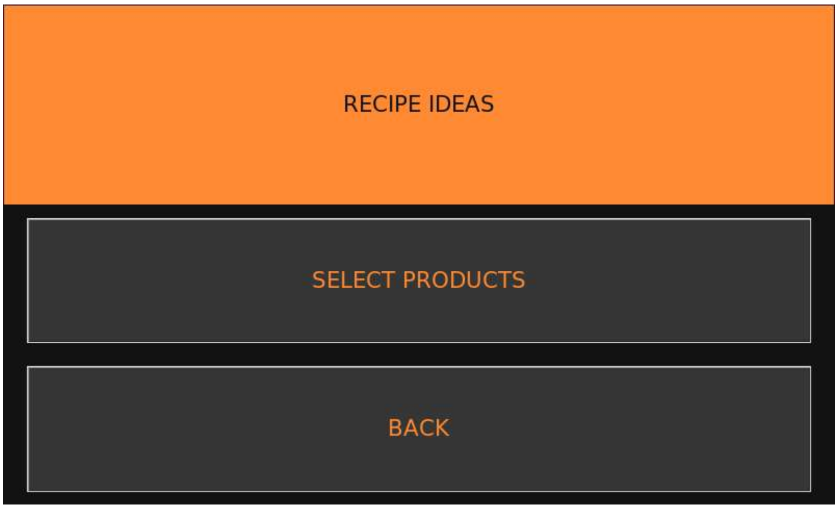
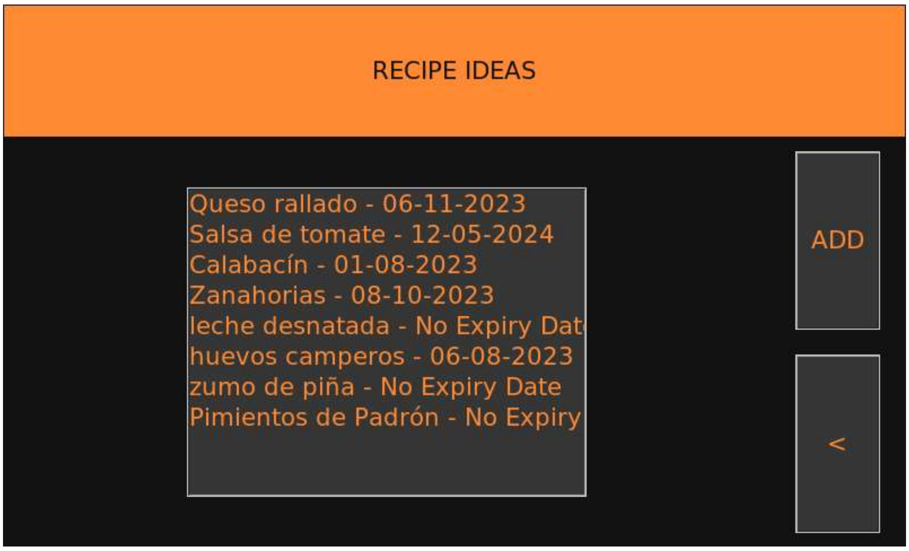


Figure 8.22: 'RECIPE IDEAS' main screen

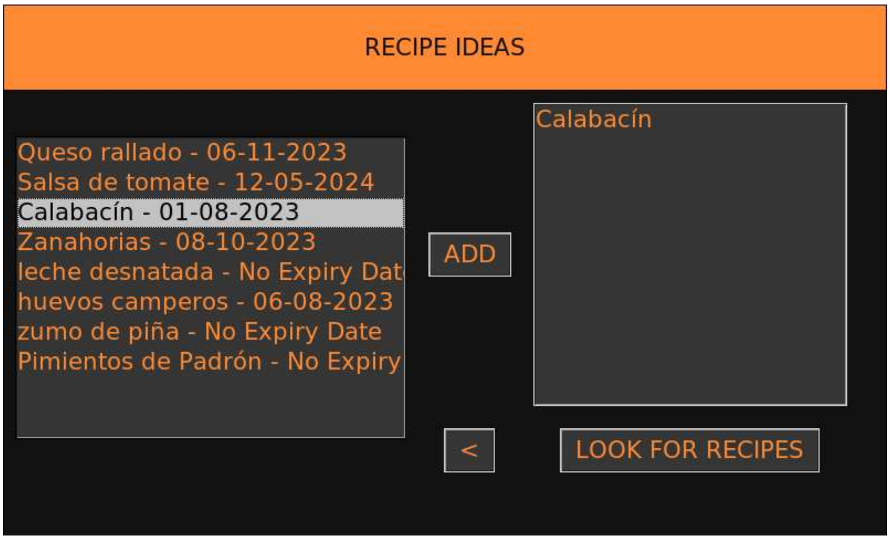
When the user selects "**Select Products**," the application transitions to display, much like the previous section, the list of products stored in the system. From this list, they can choose a sublist of items they intend to use in preparing a meal, as illustrated in Figure 8.23.



Queso rallado - 06-11-2023
Salsa de tomate - 12-05-2024
Calabacín - 01-08-2023
Zanahorias - 08-10-2023
leche desnatada - No Expiry Dat
huevos camperos - 06-08-2023
zumo de piña - No Expiry Date
Pimientos de Padrón - No Expiry

Figure 8.23: Product list visualization

Choosing each product involves first clicking on the desired item and then clicking the ADD button, resulting in the following screenshot (Figure 8.24):

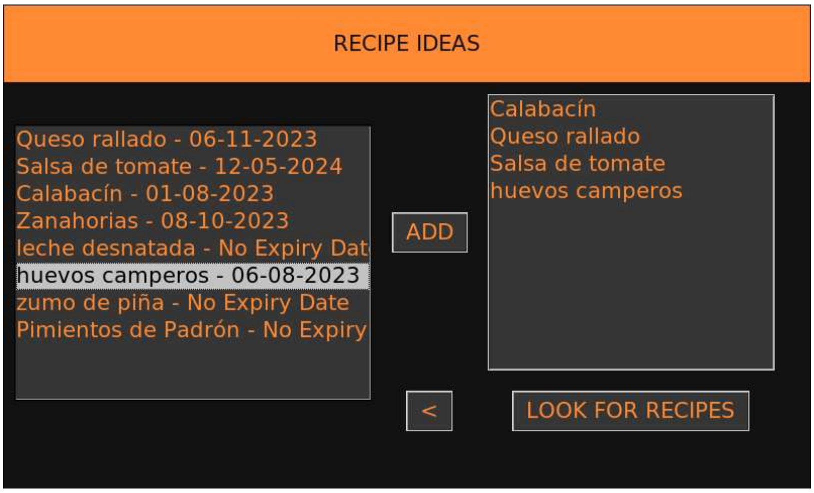


Queso rallado - 06-11-2023
Salsa de tomate - 12-05-2024
Calabacín - 01-08-2023
Zanahorias - 08-10-2023
leche desnatada - No Expiry Dat
huevos camperos - 06-08-2023
zumo de piña - No Expiry Date
Pimientos de Padrón - No Expiry

Calabacín

Figure 8.24: Selecting mode

You can see how a new button appears. After repeating this process for all the desired products and completing the selection, the user clicks on "**LOOK FOR RECIPES**," obtaining a list of five possible recipes with the selected products. The result for the following product selection is shown below (Figure 8.24).

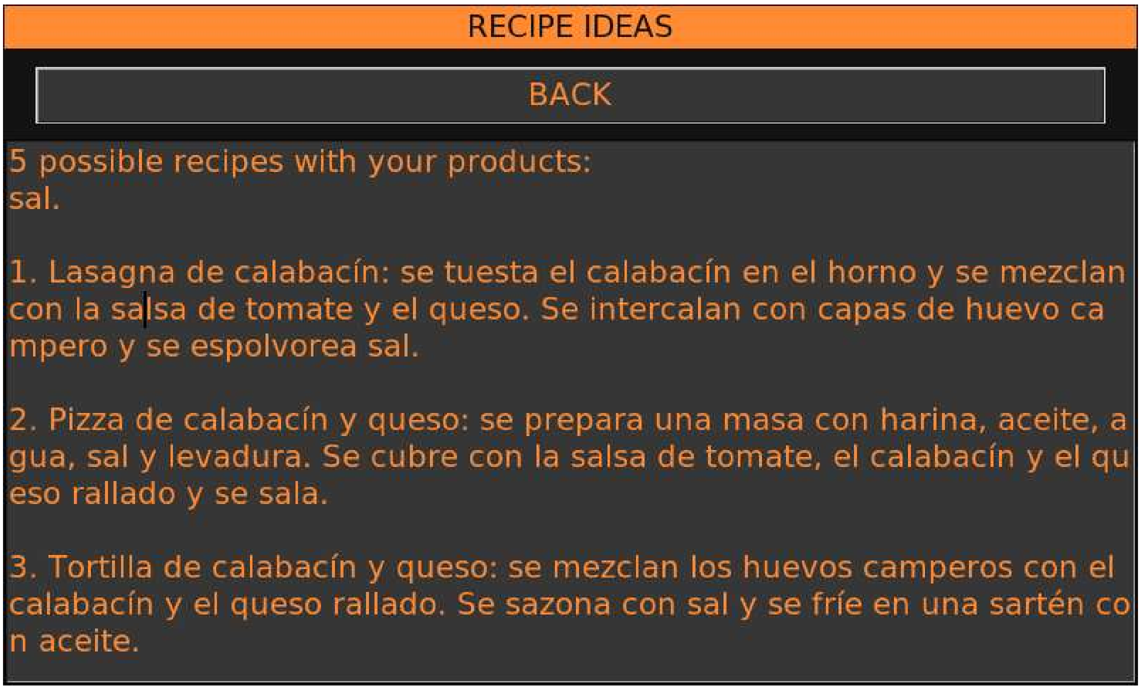


Queso rallado - 06-11-2023
Salsa de tomate - 12-05-2024
Calabacín - 01-08-2023
Zanahorias - 08-10-2023
leche desnatada - No Expiry Dat
huevos camperos - 06-08-2023
zumo de piña - No Expiry Date
Pimientos de Padrón - No Expiry

Calabacín
Queso rallado
Salsa de tomate
huevos camperos

Figure 8.25: Selection of product list for recipe ideas

The generated recipe ideas are displayed in two screenshots (Figures 8.25 and 8.26); the user can scroll to view them.



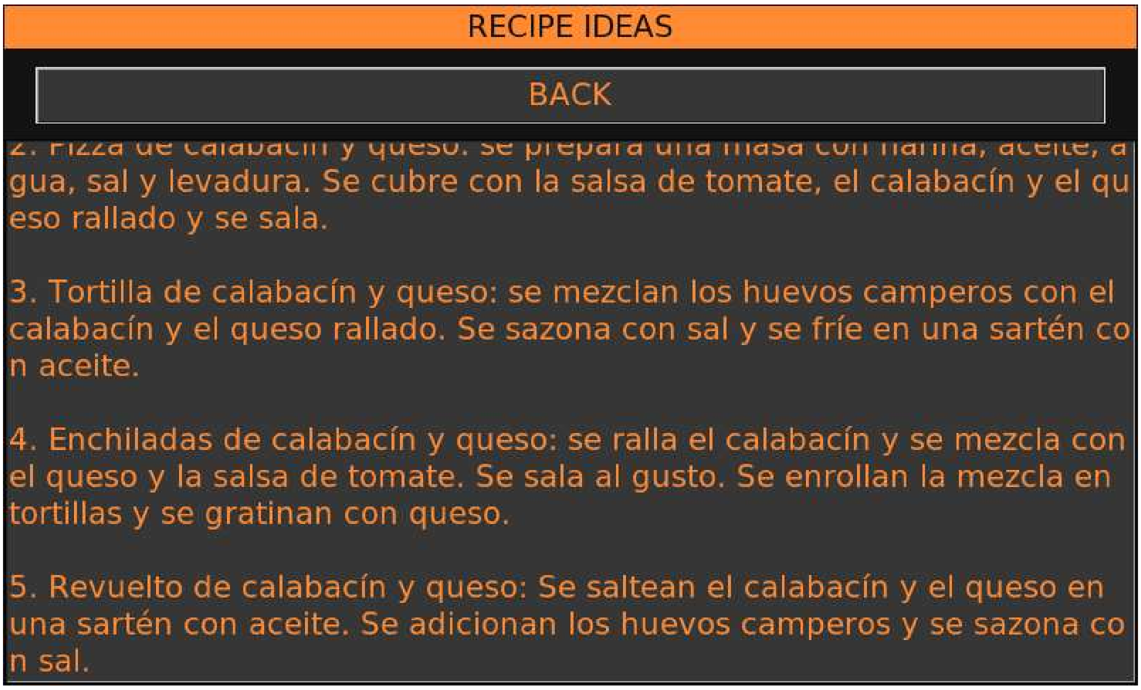
5 possible recipes with your products:
sal.

1. Lasagna de calabacín: se tuesta el calabacín en el horno y se mezclan con la salsa de tomate y el queso. Se intercalan con capas de huevo campero y se espolvorea sal.

2. Pizza de calabacín y queso: se prepara una masa con harina, aceite, agua, sal y levadura. Se cubre con la salsa de tomate, el calabacín y el queso rallado y se sala.

3. Tortilla de calabacín y queso: se mezclan los huevos camperos con el calabacín y el queso rallado. Se sazona con sal y se fríe en una sartén con aceite.

Figure 8.26: Recipe ideas result 1/2



2. Pizza de calabacín y queso: se prepara una masa con harina, aceite, agua, sal y levadura. Se cubre con la salsa de tomate, el calabacín y el queso rallado y se sala.

3. Tortilla de calabacín y queso: se mezclan los huevos camperos con el calabacín y el queso rallado. Se sazona con sal y se fríe en una sartén con aceite.

4. Enchiladas de calabacín y queso: se ralla el calabacín y se mezcla con el queso y la salsa de tomate. Se sala al gusto. Se enrollan la mezcla en tortillas y se gratinan con queso.

5. Revuelto de calabacín y queso: Se saltean el calabacín y el queso en una sartén con aceite. Se adicionan los huevos camperos y se sazona con sal.

Figure 8.27: Recipe Ideas result 2/2