UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

GRADO EN INGENIERÍA DE SOFTWARE

FastDiet: App móvil para crear dietas personalizadas y generar listas de la compra automatizadas

FastDiet: Mobile app to create personalized diets and generate automated shopping lists

Realizado por
Pablo Pardo Fernández

Tutorizado por
Eduardo Guzmán De los Riscos

Departamento
Lenguajes y Ciencias de la Computación
UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE 2025

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA DE SOFTWARE

FastDiet: App móvil para crear dietas personalizadas y generar listas de la compra automatizadas

FastDiet: Mobile app to create personalized diets and generate automated shopping lists

Realizado por
Pablo Pardo Fernández

Tutorizado por
Eduardo Guzmán De los Riscos

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE DE 2025

Fecha defensa: septiembre de 2025

Resumen

En la sociedad actual, la planificación de una alimentación saludable, personalizada y variada supone un desafío, condicionado por la falta de tiempo, los objetivos nutricionales específicos y las preferencias o restricciones dietéticas. La gestión manual de estas variables a menudo deriva en dietas monótonas o desequilibradas, y en procesos de compra poco eficientes.

Este Trabajo de Fin de Grado aborda este reto mediante el diseño y desarrollo de FastDiet, una aplicación móvil multiplataforma que simplifica y automatiza por completo la planificación de dietas. La aplicación permite generar menús semanales de forma automática, adaptados al perfil de cada usuario, considerando sus preferencias, objetivos y restricciones. A partir de estos menús, se elabora automáticamente una lista de la compra que agrega todos los ingredientes necesarios, ajustando las cantidades y organizándolos por secciones de supermercado, facilitando el proceso de compra y evitando gastos innecesarios o el desperdicio de alimentos.

El proyecto se ha llevado a cabo con un enfoque de desarrollo full-stack. El frontend se ha implementado con React Native, Expo y TypeScript garantizando la compatibilidad con iOS y Android. El backend se ha desarrollado en Python con FastAPI y empleando MySQL como sistema de gestión de base de datos.

El resultado es una solución práctica y accesible que no solo reduce la carga mental asociada a la planificación de comidas, sino que también fomenta hábitos de alimentación saludables y equilibrados, contribuyendo a mejorar la experiencia diaria de los usuarios en torno a su nutrición y organización personal.

Palabras clave: nutrición personalizada, listas de la compra, recetas, Python, React Native.

Abstract

In modern society, planning a healthy, personalized, and varied diet presents a significant challenge, driven by factors such as lack of time, specific nutritional goals, and dietary preferences or restrictions. Manually managing these variables often leads to monotonous or unbalanced diets and inefficient shopping processes.

This Final Degree Project addresses this challenge through the design and development of FastDiet, a cross-platform mobile application that simplifies and automates diet planning. The application allows the automatic generation of weekly menus tailored to each user's profile, considering their preferences, goals and restrictions. Based on these menus, a shopping list is automatically generated, which aggregates all necessary ingredients, adjusts the quantities, and organizes them by supermarket aisles. This streamlines the shopping process, preventing unnecessary expenses and food waste.

The project has been developed with a full-stack approach. The frontend was implemented using React Native, Expo and TypeScript, ensuring compatibility with both iOS and Android platforms. The backend was built in Python with FastAPI and using MySQL as the database management system.

The result is a practical and accessible solution that not only reduces the mental load associated with meal planning, but also encourages healthy and balanced eating habits, contributing to an improved daily experience for users regarding their nutrition and personal organization.

Keywords: personalized nutrition, shopping lists, recipes, Python, React Native.

Índice

1

Introducción

1.1. Motivación

En la sociedad actual, la alimentación saludable y personalizada se ha convertido en una prioridad para un número creciente de personas. Factores como la falta de tiempo, la dificultad para planificar menús equilibrados y la necesidad de tener en cuenta intolerancias, preferencias culinarias u objetivos, hacen que mantener una dieta adecuada sea un reto diario.

A pesar de la amplia disponibilidad de información y recetas en Internet, la mayoría de las personas carecen de herramientas que centralicen y automaticen la planificación de menús, la selección de ingredientes y la generación de listas de la compra adaptadas a sus necesidades. Esto conlleva, en muchos casos, a dietas repetitivas, desequilibradas o poco adaptadas a los objetivos de cada individuo, ya sea perder peso, ganar masa muscular o simplemente mantener hábitos saludables.

En paralelo, el auge de los dispositivos móviles y el desarrollo de aplicaciones multiplataforma han permitido crear soluciones accesibles y personalizadas, capaces de procesar grandes volúmenes de información y ofrecer recomendaciones instantáneas. Estas tecnologías, combinadas con el uso de APIs especializadas y bases de datos optimizadas, facilitan la creación de experiencias interactivas y adaptadas a cada usuario.

En este contexto, surge la idea de FastDiet: una aplicación móvil diseñada para generar menús personalizados y listas de la compra automáticas, teniendo en cuenta las preferencias alimenticias, restricciones dietéticas y objetivos nutricionales de cada usuario.

El objetivo no es solo automatizar la creación de dietas, sino también enriquecer la experiencia del usuario, permitiéndole descubrir nuevas recetas, adaptar su menú de forma flexible y optimizar el proceso de compra mediante la generación automática de una lista de la compra categorizada y portable. En definitiva, se pretende reducir la carga mental asociada a la pregunta "¿qué cómo hoy?", y facilitar que los usuarios tengan un control de su alimentación de una manera sencilla, intuitiva y adaptada a su estilo de vida.

1.2. Objetivos

El objetivo principal de este Trabajo de Fin de Grado es el diseño, desarrollo e implementación de una aplicación móvil multiplataforma, disponible para dispositivos iOS y Android, orientada a la creación de menús personalizados y la generación automática de listas de la compra, teniendo en cuenta las preferencias, restricciones y objetivos nutricionales de cada usuario.

El desarrollo se ha realizado con un enfoque full-stack, en el que el frontend se ha implementado con **React Native** empleando **Expo** y **TypeScript.** En el

backend se ha utilizado **Python** con **FastAPI**, utilizando **MySQL** como base de datos y **SQLAlchemy** junto con **Alembic** para la gestión de modelos y migraciones.

Entre las funcionalidades clave que se han implementado destacan:

- **Generación de menús personalizados:** planificación automática de recetas para la semana, adaptadas al perfil y preferencias del usuario, utilizando la base de datos propia y la API de Spoonacular.

- **Gestión flexible de recetas:** posibilidad de añadir comidas extra al menú, sustituir recetas asignadas por otras de nuestras sugerencias o por recetas creadas por el usuario.

- **Generación de listas de la compra categorizadas:** agrupación automática de ingredientes utilizados en el menú, clasificados en secciones de supermercado, y exportación a PDF para que el usuario pueda compartirla o guardarla.

- **Gestión de usuario y autenticación segura:** registro, inicio de sesión con credenciales o Google, y administración de preferencias y datos personales.

1.3. Metodología

Para el desarrollo de FastDiet, desde su concepción hasta la implementación final se siguió una metodología iterativa e incremental. Este enfoque, inspirado en los principios ágiles y adaptado a un entorno de trabajo individual, permitió estructurar el proyecto en fases manejables, facilitando la incorporación de mejoras y la adaptación a posibles cambios durante el proceso.

Las principales características de la metodología aplicada fueron:

- **División en Fases:**

  El ciclo de vida del proyecto se dividió en las fases clásicas de la ingeniería de software: análisis de requisitos, diseño, implementación, pruebas y despliegue.

- **Desarrollo Incremental:**

  La fase de implementación se organizó en iteraciones o "sprints", cada uno enfocado en desarrollar un conjunto coherente de funcionalidades. Esto aseguró que al final de cada ciclo se dispusiera de una versión funcional y ampliada de la aplicación.

- **Gestión de tareas con Trello:**

  Para la planificación y el seguimiento del trabajo, se utilizó un tablero Kanban en Trello, lo que proporcionó una visión clara del progreso de las tareas a través de los estados "Pendiente", "En Progreso" y "Finalizado" ("To Do", "Doing", "Done").

- **Control de Versiones con Git:**

  Se utilizó Git para llevar un seguimiento riguroso de los cambios en el código, gestionar el desarrollo de nuevas funcionalidades en ramas aisladas y garantizar la integridad del proyecto.

- **Evaluación continua:**

  Al final de cada iteración se revisaron los avances y se realizaron pruebas funcionales, lo que permitió detectar errores tempranamente y validar que los requisitos se habían implementado correctamente.

Gracias a esta metodología, el proyecto pudo evolucionar de manera ordenada, adaptándose a las necesidades y garantizando la calidad en la implementación de cada componente y del producto final.

1.4. Estructura de la memoria

La memoria sigue un orden que permite al lector comprender progresivamente el proyecto. Comienza presentando el contexto y las tecnologías empleadas, para después avanzar a través de las distintas fases que han conformado el desarrollo de la aplicación, desde el análisis inicial hasta las conclusiones finales.

1. **Introducción**

   En este apartado se ofrece al usuario una visión general de la memoria: la motivación, los objetivos, la metodología empleada y la organización de los diferentes apartados que conforman el documento.

2. **Tecnologías, librerías y herramientas**

   Se describen las tecnologías, librerías y herramientas que desempeñaron un papel importante en la planificación e implementación del proyecto.

3. **Especificación y Análisis**

   Se describen los requisitos funcionales y no funcionales de la aplicación, también se definen los principales casos de uso.

4. **Diseño**

   Se explican las elecciones tomadas respecto a la arquitectura y la organización del proyecto.

5. Implementación, Pruebas y Despliegue

Se explica cómo han sido desarrollados los requisitos funcionales por iteraciones, explicando la lógica implementada para cumplirlos, además de las estrategias de pruebas y despliegue de la aplicación.

6. Conclusiones

Se describen las conclusiones finales del proyecto, explicando los objetivos cumplidos, dificultades encontradas y posibles ampliaciones.

2

Tecnologías, librerías y herramientas

2.1. Tecnologías empleadas

En este capítulo se describen las tecnologías que han tenido un papel clave en el desarrollo de FastDiet. La elección de cada una de ellas se ha basado en su nivel de desarrollo y adaptación a los requisitos del proyecto.

2.1.1. React Native

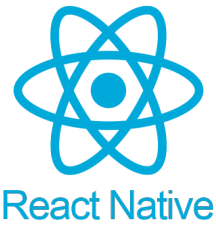


Figura 1. Logo React Native

React Native es un framework de código abierto pensado para el desarrollo de aplicaciones móviles nativas utilizando JavaScript o TypeScript junto a React.

Una de sus principales ventajas es que posibilita desarrollar para iOS y Android desde un único código base, optimizando tiempos y esfuerzos.

2.1.2. Expo

Figura 2. Logo Expo

Expo ofrece un conjunto de servicios y herramientas que apoyan el desarrollo con React Native, ofreciendo utilidades para la construcción, depuración y despliegue de aplicaciones móviles.

La aplicación Expo Go ha resultado especialmente útil para previsualizar y probar la aplicación en dispositivos físicos sin necesidad de compilaciones completas, acelerando las iteraciones de desarrollo.

2.1.3. TypeScript

Figura 3. Logo TypeScript

TypeScript es un lenguaje de programación open source creado por Microsoft, diseñado como una extensión de JavaScript. Su característica fundamental es que añade un sistema de tipado estático opcional al lenguaje.

La utilización de TypeScript en el frontend del proyecto ha sido una decisión clave para aumentar la robustez del código, facilitar la detección de errores en tiempo de desarrollo y mejorar la mantenibilidad.

2.1.4. Python

Figura 4. Logo Python

Python es un lenguaje de programación interpretado, conocido por su legibilidad y capacidad para usarse en una amplia variedad de proyectos, además de estar respaldado por una gran comunidad.

En FastDiet ha sido utilizado en el backend, implementado con FastAPI, para gestionar la lógica de negocio, la conexión con la base de datos y la integración con APIs externas.

2.1.5. MySQL

Figura 5. Logo MySQL

MySQL es gestor de bases de datos relacional, de carácter abierto y ampliamente utilizado en el ámbito del desarrollo de software.

En este proyecto se utiliza para almacenar y organizar la información de usuarios, menús, recetas, etc., garantizando integridad de datos y escalabilidad.

La aplicación de MySQL Workbench, ha resultado práctica para inspeccionar el estado y el contenido de la base de datos.

2.2. Librerías, Herramientas y Servicios utilizados

En esta sección se presentan las librerías y servicios externos más destacados que han sido empleados en el desarrollo del proyecto, tanto en la parte del backend como en el frontend.

2.2.1. Backend (Python)

a) FastAPI:

Se trata de un framework moderno y de alto rendimiento para la creación de APIs en Python, basado en tipado estático y ofrece compatibilidad con OpenAPI, lo que permite generar automáticamente documentación interactiva (Swagger UI). En este proyecto se ha utilizado para implementar la API principal, gestionando la comunicación entre el cliente y la base de datos.

b) SQLAlchemy:

Biblioteca ORM (Object Relational Mapping) que facilita la interacción con bases de datos relacionales desde Python. Permite trabajar con modelos de datos como objetos, evitando la escritura directa de consultas SQL.

c) Alembic:

Se trata de una herramienta de migración diseñada para trabajar con SQLAlchemy. Se ha utilizado para administrar y mantener el versionado de las modificaciones en la estructura de la base de datos durante el desarrollo.

d) Pydantic:

Biblioteca para la validación y serialización de datos en Python. Se ha utilizado para asegurar la integridad y el formato de los datos recibidos y enviados por la API.

e) Google Cloud Translate API:

Servicio de Google Cloud para traducción automática. Se ha utilizado para traducir recetas e ingredientes entre inglés y español.

f) Google Cloud Storage:

Servicio de Google Cloud para almacenar archivos en un bucket, usado en FastDiet para almacenar la imágenes de las recetas creadas por usuarios.

g) Spoonacular API:

API externa utilizada para obtener recetas, información nutricional, datos de ingredientes y ayudar a la generación de la lista de la compra.

h) SlowAPI:

Biblioteca que implementa un middleware de *rate limiting* en FastAPI, protegiendo la API de un uso excesivo o abusivo.

i) Passlib (bcrypt):

Biblioteca de gestión segura de contraseñas que permite aplicar *hashing* y verificación de contraseñas. Se ha configurado con el algoritmo *bcrypt* para mayor seguridad.

j) Logging:

Módulo de Python, que permite registrar eventos y mensajes relevantes durante la ejecución de la aplicación, facilitando el monitoreo y depuración del sistema.

k) AppScheduler:

Biblioteca para la programación de tareas en segundo plano. En este proyecto se utiliza para la eliminación automática de usuarios no verificados cada día.

2.2.2. Frontend (React Native + Expo)

a) Axios:

Biblioteca para la realización de peticiones HTTP, utilizada para la comunicación con la API.

b) Expo Secure Store:

Módulo de Expo para almacenar datos sensibles de forma segura en el dispositivo.

c) Expo Print y Expo Sharing:

Módulos de Expo para generar documentos imprimibles y compartirlos directamente desde la aplicación, utilizados para generar y compartir un archivo PDF de la lista de la compra de la semana

d) Expo Image Picker:

Módulo que permite seleccionar o capturar imágenes desde el dispositivo, usado para añadir fotos a las recetas creadas por el usuario.

e) Expo Localization:

Módulo que obtiene la configuración regional del dispositivo, utilizado junto con i18next para la traducción de la interfaz.

f) i18next:

Biblioteca de internacionalización que permite traducir la interfaz a múltiples idiomas. En este proyecto se ha usado para ofrecer la aplicación en español e inglés.

g) Lottie React Native:

Biblioteca para mostrar animaciones en formato JSON, utilizada en las pantallas de carga al generar el menú o la lista de la compra.

2.2.2. Otras Herramientas

a) Git:

Figura 7. Logo Git

Git es una herramienta de control de versiones distribuido, muy extendida en el desarrollo de software.

Facilita llevar un registro de las modificaciones en el código, coordinar la colaboración entre varios desarrolladores y recuperar versiones anteriores cuando se requiere.

Ha sido la herramienta fundamental para gestionar versiones y cambios durante todo el ciclo de vida del desarrollo de FastDiet.

b) Visual Studio Code

Figura 6. Logo VS Code

Visual Studio Code (VS Code) es un editor de código fuente ligero y extensible, creado por Microsoft.

En este proyecto, se ha utilizado como editor principal para todo el código, tanto del frontend como del backend, aprovechando extensiones específicas para React Native y Python.

3

Especificación y Análisis

3.1. Roles de usuario

En FastDiet existe un único rol de usuario: el **usuario registrado**, que puede utilizar sin restricciones todas las funciones disponibles en la aplicación. Cualquier persona que se descargue la aplicación y complete el proceso de registro pertenecerá a este grupo.

No hay roles administrativos ni jerarquías de permisos, ya que la aplicación está orientada a un uso personal.

3.2. Requisitos funcionales

En este apartado se detallan los distintos requisitos funcionales (RF) planteados para cubrir todas las funcionalidades previstas. Los requisitos con identificadores y descripciones se pueden consultar en la Tabla 1.

Identificador del requisito	**Nombre**	**Descripción**
RF1	Registro de usuario	El usuario podrá registrarse proporcionando correo electrónico y contraseña, o usando la autenticación con Google. Durante el proceso de completar el registro, se solicitarán datos como nombre de usuario, nombre, peso, altura, edad, sexo, nivel de actividad física, objetivo, tipo de dieta, tipos de cocina preferidos o intolerancias alimentarias.
RF2	Inicio de sesión	El usuario podrá iniciar sesión escribiendo su correo electrónico o nombre de usuario y su contraseña, o usar el inicio de sesión con Google.
RF3	Recuperación de contraseña	El sistema ofrecerá una opción para que los usuarios que hayan olvidado su contraseña puedan restablecerla. El proceso implicará verificar un código enviado al correo electrónico del usuario registrado. Una vez verificado, el usuario podrá establecer una nueva contraseña.

Identificador del requisito	**Nombre**	**Descripción**
RF4	Generación de menú semanal personalizado	El usuario podrá generar un menú semanal personalizado para desayuno, almuerzo y cena, de lunes a viernes, adaptado a sus preferencias y restricciones.
RF5	Personalización y edición del menú semanal	El usuario podrá cambiar las comidas asignadas al menú por sugerencias o por recetas creadas por él mismo. Podrá asignar/eliminar las comidas de cualquier día de la semana.
RF6	Añadir otro tipo de comidas extra al menú semanal	El usuario podrá añadir a cada día comidas o bebidas adicionales, seleccionando entre diferentes categorías como Aperitivos, Ensaladas, Postres, Bebidas o Snacks, ya sean seleccionados de nuestras sugerencias o de recetas creadas por el usuario.
RF7	Visualización detallada de recetas	El usuario podrá consultar la información completa de una receta, incluyendo ingredientes, información nutricional, instrucciones de preparación y equipo necesario.

Identificador del requisito	**Nombre**	**Descripción**
RF8	Gestión de recetas propias	El usuario podrá crear, editar y eliminar recetas propias, que podrá asignar al menú semanal.
RF9	Generación de lista de la compra	El usuario podrá generar una lista de la compra con los ingredientes necesarios para su menú semanal, agrupados por secciones de supermercado.
RF10	Exportación y compartición de la lista de la compra	El usuario podrá exportar la lista de la compra a PDF y compartirla con otras personas.
RF11	Gestión del perfil	El usuario podrá ver y editar sus datos personales y preferencias, cambiar su contraseña.
RF12	Cálculo de calorías objetivo	El sistema calculará automáticamente el objetivo de calorías diarias del usuario en base a los datos introducidos.
RF13	Cerrar de sesión	El usuario podrá cerrar su sesión activa en la aplicación de forma segura.
RF14	Eliminación de la cuenta	El usuario tendrá la opción de eliminar su cuenta y todos sus datos asociados de forma permanente.

Tabla 1. Requisitos Funcionales

3.3. Requisitos no funcionales

En esta sección se describen los distintos requisitos no funcionales (RNF), que definen los criterios de calidad y las restricciones bajo las cuales el sistema debe operar. Los requisitos con identificadores, tipos y descripciones se pueden consultar en la Tabla 2.

Id	**Tipo de Requisito No Funcional**	**Nombre**	**Descripción**
RNF1	Internacionalización	Múltiples idiomas	La aplicación detecta el idioma de preferencia del usuario (inglés o español).
RNF2	Usabilidad	Interfaz intuitiva	La aplicación debe ser fácil de usar, con un flujo claro para generar menús o listas de la compra.
RNF4	Compatibilidad	Compatibilidad multiplataforma	La aplicación ser funcional y compatible con dispositivos Android e iOS.
RNF5	Seguridad	Protección de datos	Los datos personales y de preferencias del usuario serán tratados y almacenados de forma segura.
RNF6	Mantenibilidad	Código modular	El código debe estar estructurado de forma modular para facilitar su mantenimiento y evolución.

Id	Tipo de Requisito No Funcional	Nombre	Descripción
RNF7	Fiabilidad	Integridad de datos	La información de menús, recetas e ingredientes debe almacenarse sin pérdidas ni inconsistencias.

Tabla 2. Requisitos No Funcionales

3.4. Casos de uso

A continuación, se detallan los casos de uso más representativos que describen la interacción del usuario con las funcionalidades clave del sistema, especificadas previamente en los requisitos funcionales.

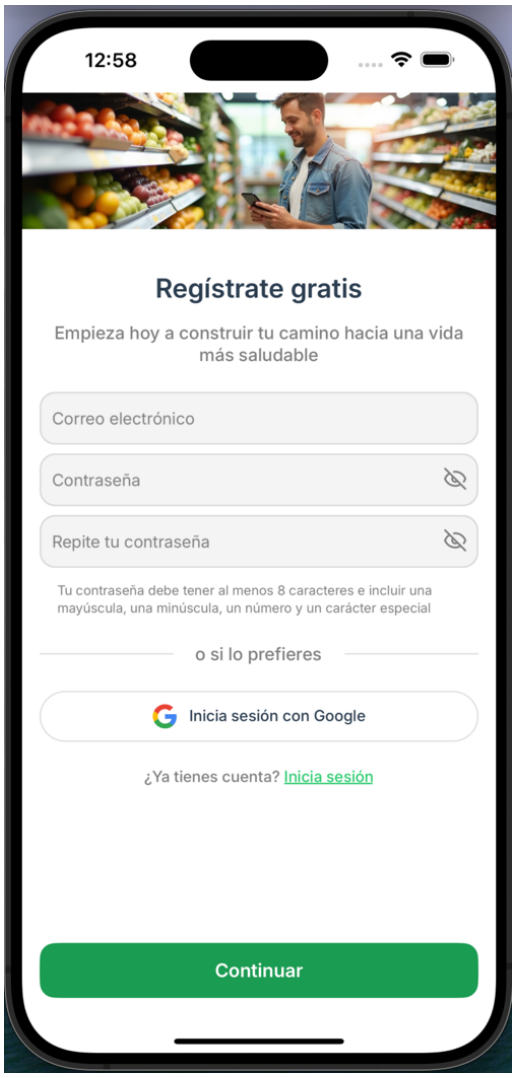
Título	Registro de usuario (RF1)
Descripción	Permite al usuario registrarse introduciendo correo electrónico y contraseña, o usando la autenticación con Google. Durante el proceso de registro, podrá completar su perfil con datos como nombre de usuario, peso, altura, edad, sexo, nivel de actividad física, objetivo, tipo de dieta, tipos de cocina preferidos o intolerancias alimentarias.
Pre-condición	El usuario no está identificado en la app o no está verificado y está en la pantalla de inicio de sesión.
Post-condición	Se añade un usuario verificado, con la información completa de su perfil y de sus preferencias.

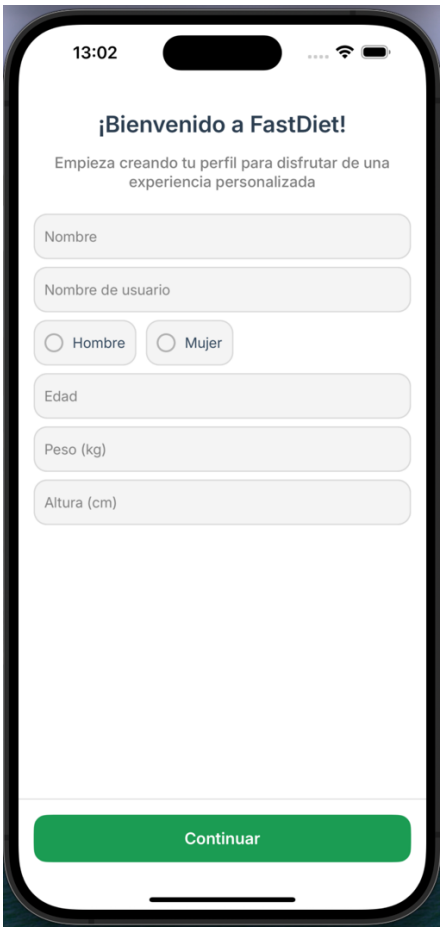
Escenario Principal
1. El usuario selecciona la opción de "¿No tienes una cuenta? Regístrate gratis". 2. El sistema redirige al usuario a la primera página de registro, el usuario escribe un correo electrónico válido, una contraseña, que cumpla la política de seguridad y su confirmación. A continuación, pulsa "Continuar". 3. El usuario es redirigido a una página de verificación e introduce el código de 6 dígitos recibido en su correo y pulsa "Verificar código". 4. El sistema valida el código y redirige al usuario al flujo de completar el perfil y las preferencias alimenticias. El usuario completa en cada pantalla la información solicitada y pulsando "Continuar" para ir completando cada paso del flujo. 5. Cuando el usuario ha completado correctamente todos los pasos del flujo de registro, el sistema crea su sesión, muestra un mensaje de éxito y redirige al usuario a la pantalla principal de la aplicación.
Escenario alternativo
A1: 2.b: El usuario inserta un correo ya existente que está verificado, o introduce una contraseña que no sigue la política de seguridad. Se muestra el correspondiente mensaje de error y el usuario permanece en la pantalla de registro hasta introducir datos válidos. A2: 3.b: El usuario inserta un código inválido o expirado. El sistema muestra un mensaje de error, el usuario permanece en la pantalla de verificación con la opción de solicitar un nuevo código.

A3:

4.b: El usuario escribe un nombre de usuario que ya está en uso o completa algún campo con datos inválidos como peso o altura negativos. El sistema muestra un mensaje de error, y el usuario permanece en la misma pantalla hasta que introduzca datos válidos y pueda continuar con el flujo de registro.

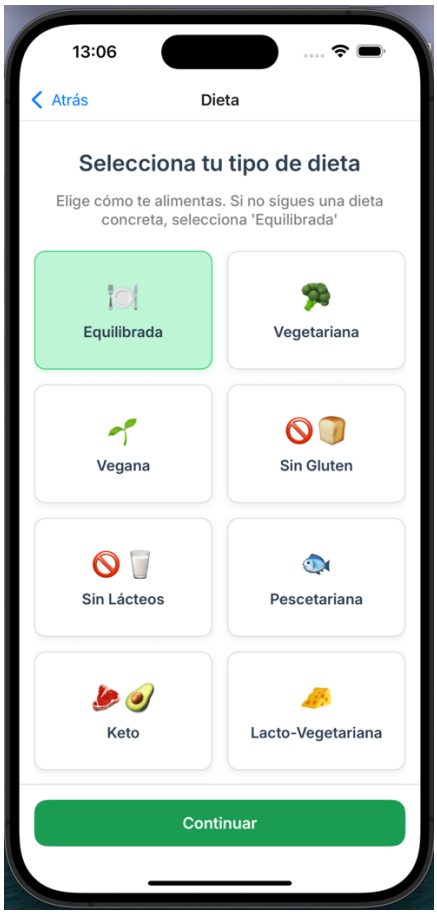
Interfaces implicadas en el caso de uso




13:02
¡Bienvenido a FastDiet!
Empieza creando tu perfil para disfrutar de una experiencia personalizada
Nombre
Nombre de usuario
Hombre
Mujer
Edad
Peso (kg)
Altura (cm)
Continuar


13:04
Atrás
Actividad
Elige tu nivel de actividad física
Esto nos ayudará a calcular mejor tus necesidades calóricas diarias
Sedentario
Poco o ningún ejercicio físico, trabajo de oficina, vida mayormente inactiva.
Actividad ligera
Ejercicio ligero 1-3 días por semana o trabajo de baja intensidad física.
Actividad moderada
Ejercicio moderado 3-5 días por semana o un trabajo algo activo.
Actividad alta
Ejercicio intenso 6-7 días por semana o trabajo físicamente exigente.
Actividad muy alta
Entrenamientos muy intensos, atletas profesionales o dos sesiones de ejercicio al día.
Continuar


13:05
Atrás
Objetivo
¿Cuál es tu objetivo?
Elige tu meta para ajustar tu plan nutricional
Perder peso
Reducir la grasa corporal manteniendo masa muscular.
Mantener peso
Conservar tu peso actual de forma saludable.
Ganar masa muscular
Aumentar peso con enfoque en masa muscular.
Continuar


13:06
Atrás
Dieta
Selecciona tu tipo de dieta
Elige cómo te alimentas. Si no sigues una dieta concreta, selecciona 'Equilibrada'
Equilibrada
Vegetariana
Vegana
Sin Gluten
Sin Lácteos
Pescetariana
Keto
Lacto-Vegetariana
Continuar

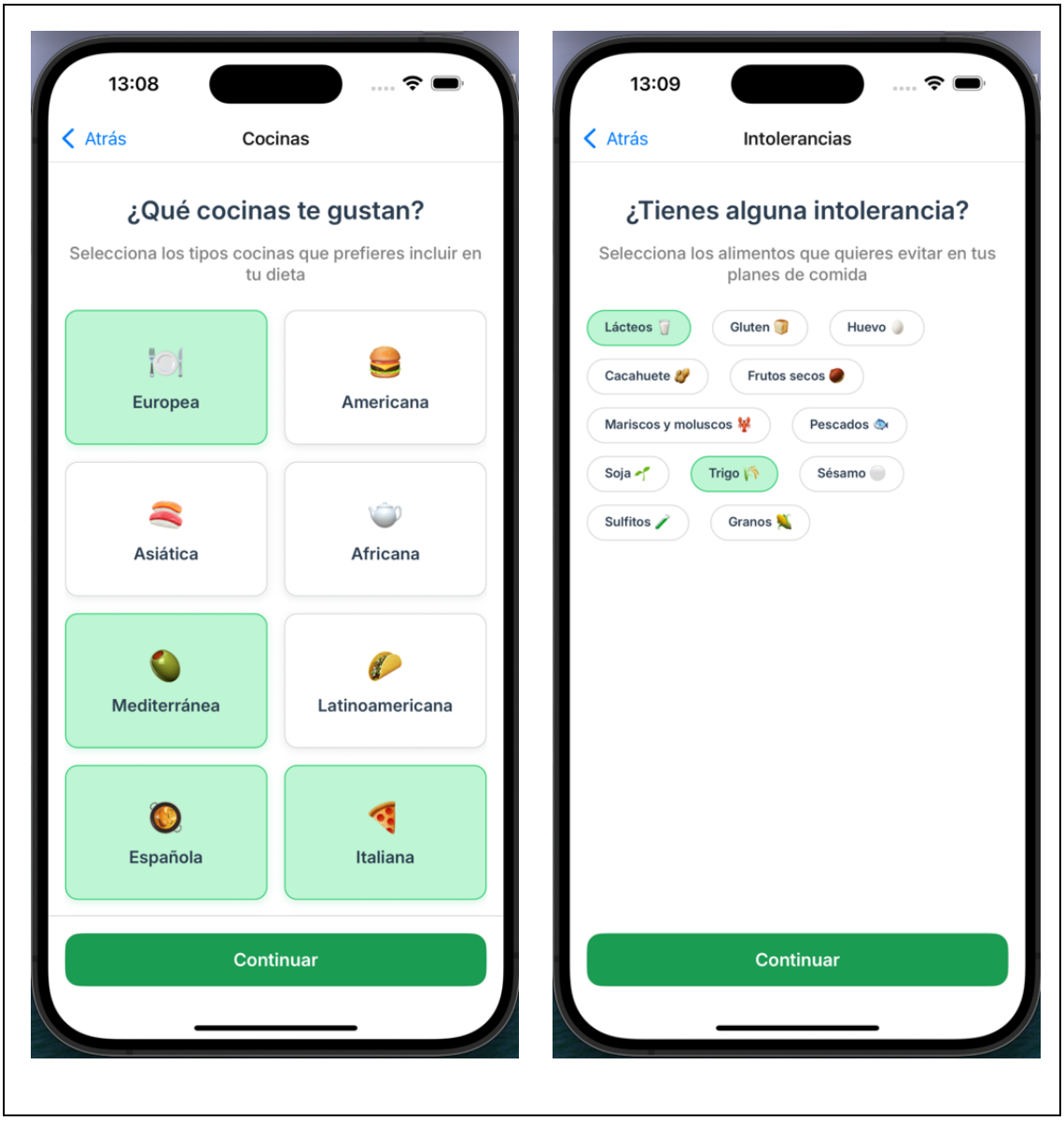


Tabla 3. Caso de Uso 1: Registro de usuario

Título	Inicio de sesión (RF2)
Descripción	Permite al usuario iniciar sesión introduciendo su correo electrónico o nombre de usuario y su contraseña, o usar el inicio de sesión con Google.
Pre-condición	El usuario está registrado y verificado pero no tiene una sesión activa.
Post-condición	El sistema crea una sesión para el usuario, El usuario es autenticado, autorizado y redirigido a la página principal.
Escenario Principal	
1. El usuario abre la aplicación y se encuentra en la pantalla de Inicio de sesión. 2. El usuario introduce el nombre de usuario o el correo electrónico y contraseña correctos y pulsa “Continuar”. 3. El sistema valida las credenciales con la base de datos y crea la sesión. El usuario es redirigido a la página principal de la aplicación.	
Escenario alternativo	
A1: 2.b: El usuario introduce un correo electrónico/nombre de usuario y/o contraseña inválidos. 3.b: El sistema muestra un mensaje de error. El usuario permanece en la página de inicio de sesión.	

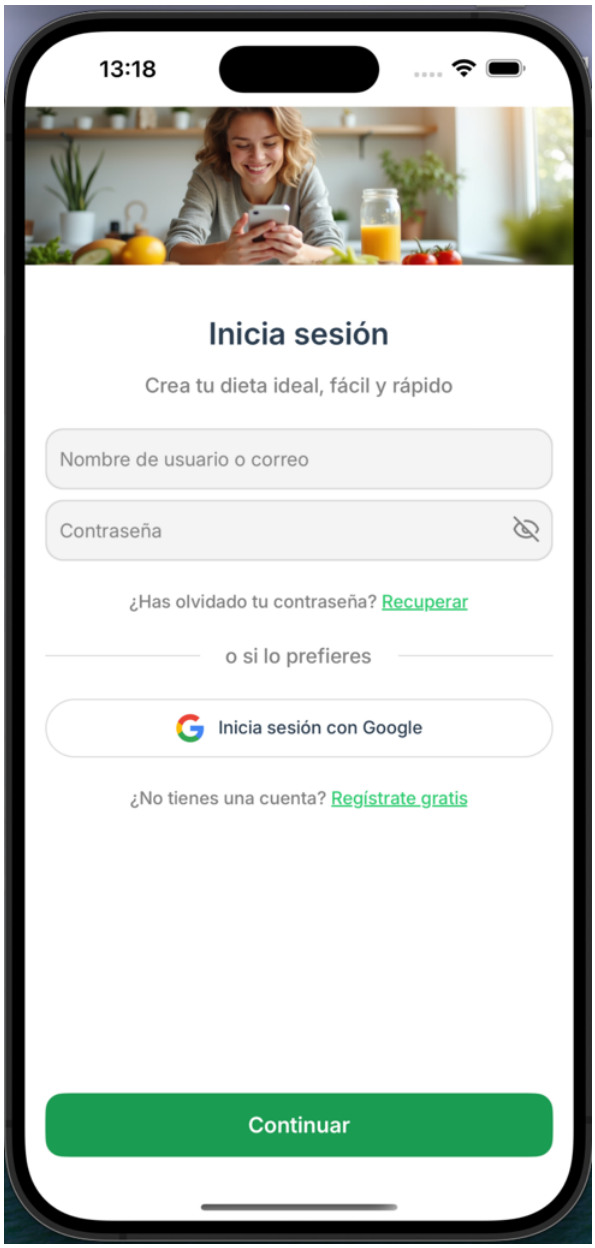
Interfaces implicadas en el caso de uso


Tabla 4. Caso de Uso 2: Inicio de sesión

Título	Recuperación de contraseña (RF3)
Descripción	Permite a los usuarios que hayan olvidado su contraseña establecer una nueva. El proceso implicará verificar un código enviado al correo electrónico del usuario registrado. Una vez verificado, el usuario podrá establecer una nueva contraseña.
Pre-condición	El usuario está registrado y verificado pero no está identificado en la aplicación y no recuerda su contraseña.
Post-condición	La contraseña del usuario se actualiza de forma segura en la base de datos. El usuario es redirigido a la página de inicio de sesión para que acceda con sus nuevas credenciales.
Escenario Principal	
1. En la página de inicio de sesión, el usuario selecciona la opción de "¿Has olvidado tu contraseña? Recuperar". 2. El usuario es redirigido a la primera página del flujo de reestablecer la contraseña, introduce el correo electrónico de su cuenta y pulsa "Enviar código". 3. El usuario es redirigido a una página de verificación e introduce el código de 6 dígitos recibido en el correo introducido anteriormente, y pulsa "Verificar código".	

4. El sistema valida el código y el usuario es redirigido a una pantalla para establecer su nueva contraseña, debe introducir una nueva contraseña siguiendo las políticas de seguridad, confirmarla y pulsar “Actualizar contraseña”. 5. El sistema actualiza la contraseña en la base de datos, guardando el hash generado, se muestra un mensaje de éxito y el usuario vuelve a la página de inicio de sesión.
Escenario alternativo
A1: 2.b: El usuario introduce un correo inválido o que no está registrado. El sistema muestra un mensaje de error. A2: 3.b: El usuario inserta un código inválido o expirado. El sistema muestra un mensaje de error, el usuario permanece en la pantalla de verificación con la opción de solicitar un nuevo código. A3: 4.b: El usuario escribe una contraseña que no cumple con las reglas de seguridad (minúsculas, mayúsculas, números, caracteres especiales). El sistema muestra un mensaje de error, y el usuario permanece en la misma pantalla hasta que introduzca una contraseña válida.
Interfaces implicadas en el caso de uso

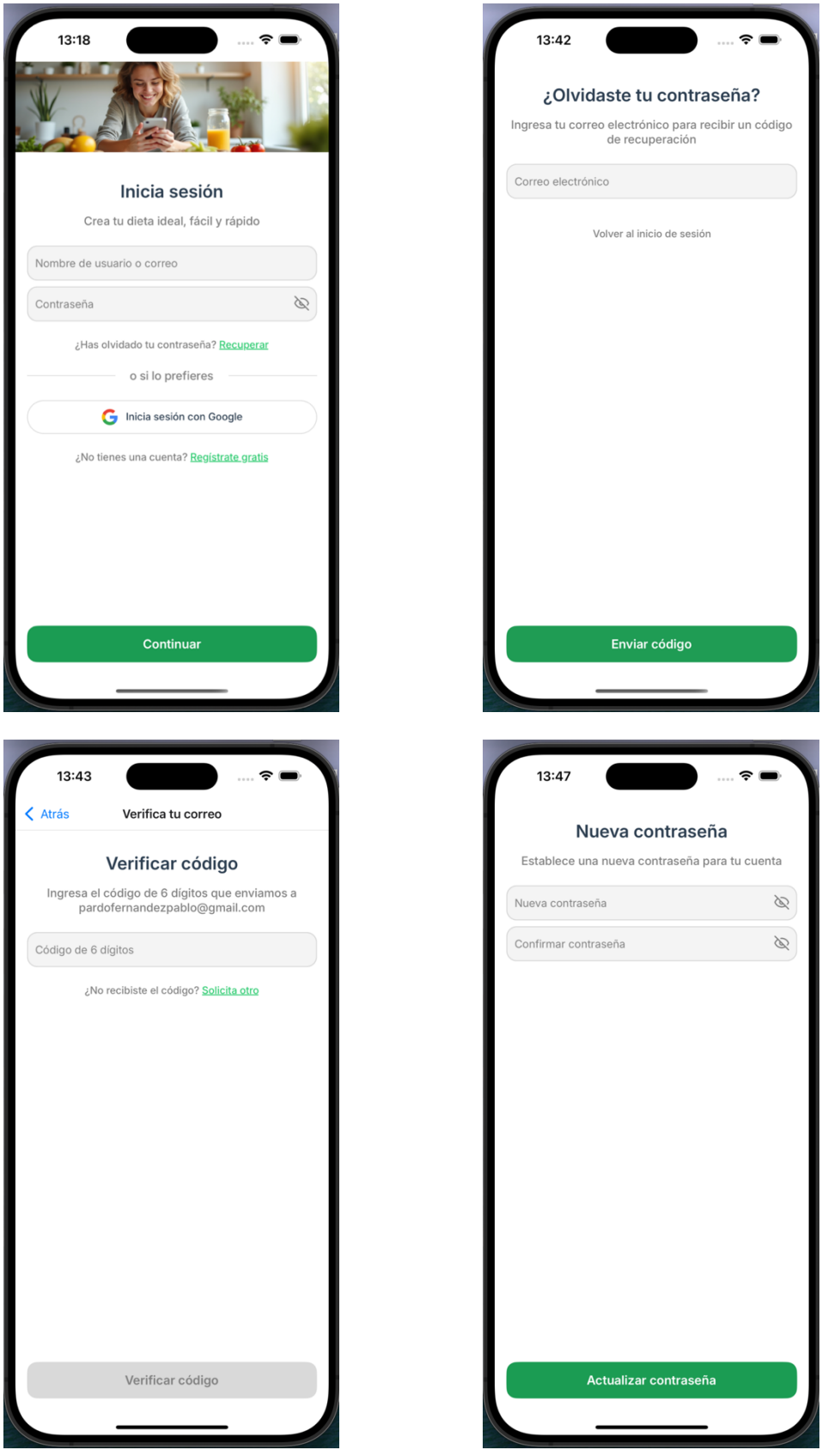


Tabla 5. Caso de Uso 3: Recuperación de contraseña

Título	Gestión del Perfil (RF11)
Descripción	Un usuario autenticado puede ver y editar sus datos personales y preferencias, incluso cambiar su contraseña.
Pre-condición	El usuario ha iniciado sesión en la app y se está en la página de “Perfil”.
Post-condición	La información específica modificada por el usuario se actualiza correctamente en la base de datos. Si se han modificado datos que afectan al cálculo de calorías (peso, altura, edad, sexo, nivel de actividad, objetivo), el sistema recalcula y actualiza el objetivo calórico diario del usuario.
Escenario Principal	
1. En la pantalla de “Perfil” el usuario selecciona la sección de la información que desea ver y modificar (ej. “Mi Plan” -> “Nivel de actividad”). 2. El usuario es redirigido a una pantalla específica que muestra la información actual y el resto de opciones que puede elegir o campos que pueda modificar. 3. El usuario realiza las modificaciones deseadas y pulsa el botón de confirmación “Guardar”. 4. El sistema valida los nuevos datos y los actualiza en la base de datos. A continuación, muestra un mensaje de éxito y devuelve al usuario a la pantalla principal de “Perfil”, donde se refleja la información actualizada.	

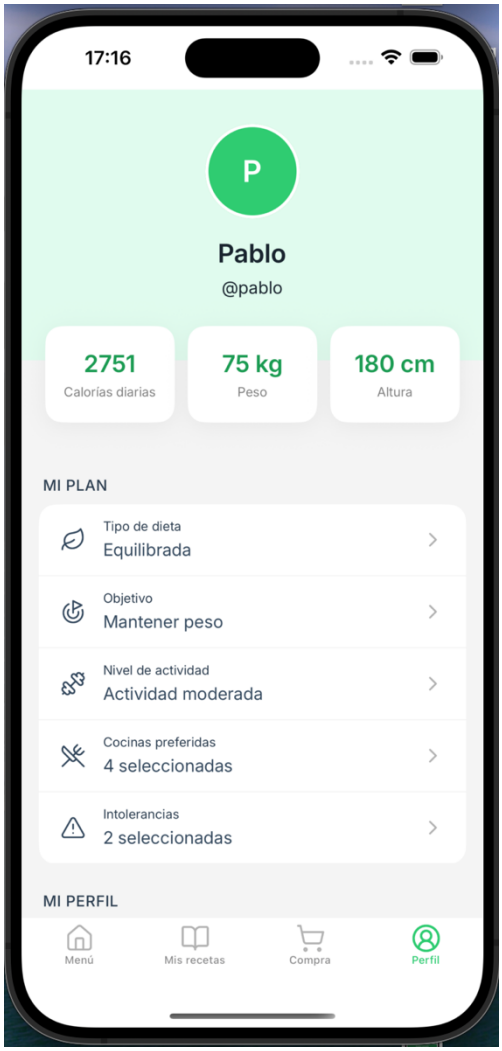
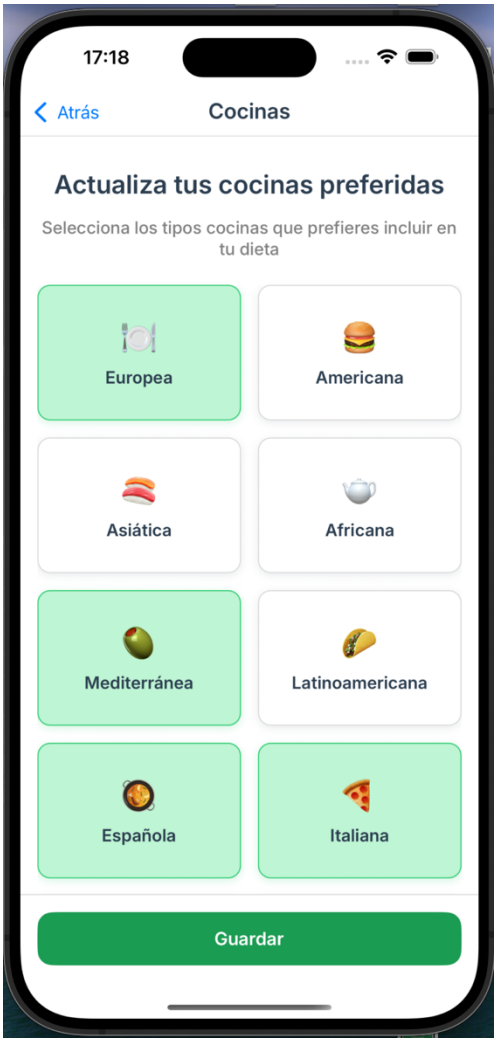
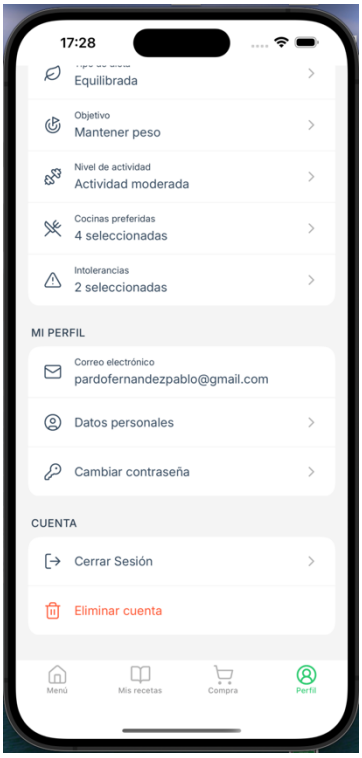
Escenario alternativo
A1: 3.b: El usuario decide no guardar los cambios y vuelve atrás. El sistema descarta las cambios y devuelve al usuario a la pantalla de “Perfil”. A2: 4.b: Los datos introducidos son inválidos. El sistema detecta que un dato es inválido (ej. un peso no numérico, un nombre de usuario ya en uso, etc.) y muestra un mensaje de error específico junto al campo incorrecto. El usuario permanece en la pantalla de edición para corregir la información.
Interfaces implicadas en el caso de uso
 

Tabla 6. Caso de Uso 4: Gestión del perfil

Título	Cerrar Sesión (RF13)
Descripción	El usuario podrá cerrar su sesión activa de forma segura.
Pre-condición	El usuario ha iniciado sesión en la app y está en la página de “Perfil”.
Post-condición	La sesión del usuario es invalidada. El usuario es redirigido a la página de inicio de sesión. Todos sus datos permanecen guardados en el sistema.
Escenario Principal	
1. El usuario selecciona la opción “Cerrar Sesión” en la sección “Cuenta”. 2. El sistema muestra un diálogo de confirmación para prevenir un cierre de sesión accidental. 3. El usuario confirma la acción, pulsando “Aceptar”. El sistema elimina los tokens de sesión del almacenamiento seguro del dispositivo y redirige al usuario a la pantalla de inicio de sesión.	
Escenario alternativo	
A1: 3.b: El usuario selecciona “Cancelar” en el diálogo (Alert). El sistema cierra el diálogo y devuelve al usuario a la pantalla de “Perfil” con su sesión activa.	
Interfaces implicadas en el caso de uso	



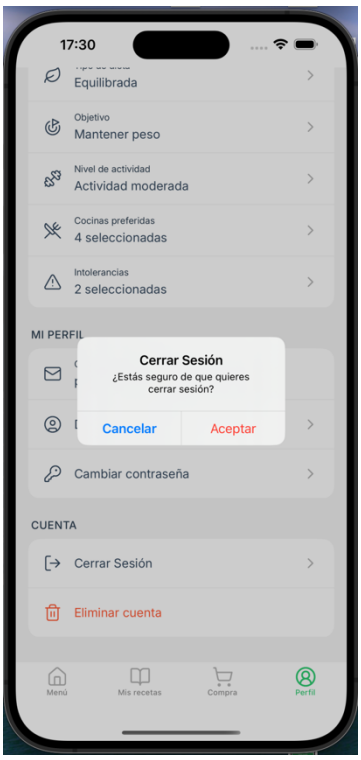


Tabla 7. Caso de Uso 5: Cerrar sesión

Título	Eliminar Cuenta (RF14)
Descripción	Permite al usuario eliminar de forma permanente su cuenta y toda la información asociada a ella.
Pre-condición	El usuario ha iniciado sesión en la app y se está en la página de “Perfil”.
Post-condición	Todos los datos del usuario (perfil, preferencias, menús, recetas propias, etc.) son eliminados permanentemente de la base de datos del sistema. La sesión del usuario es invalidada y es redirigido a la página de inicio de sesión.
Escenario Principal	

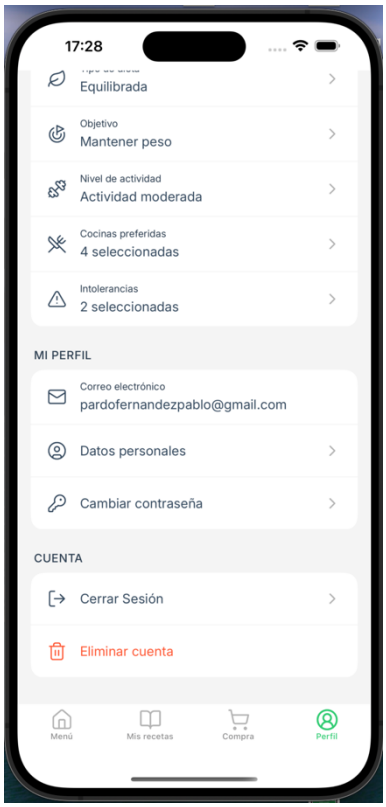
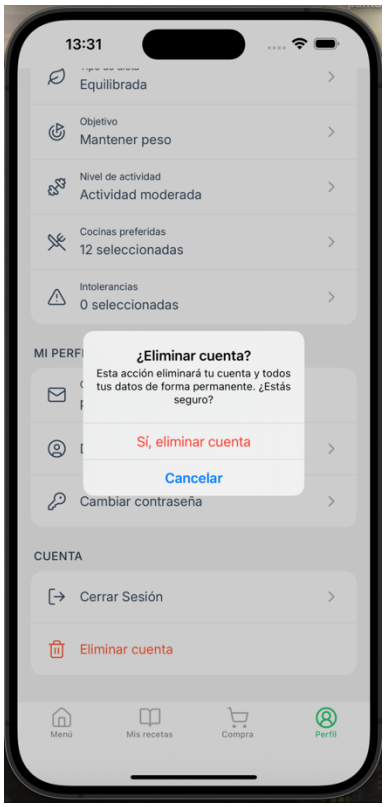
1. En la pantalla de “Perfil” el usuario selecciona la opción “Eliminar Cuenta” en la sección “Cuenta”. 2. El sistema muestra un diálogo de confirmación, advirtiendo al usuario que la acción es irreversible y resultará en la pérdida de todos sus datos. 3. El usuario confirma la acción, pulsando “Sí, eliminar cuenta”. El sistema elimina todos los datos y el usuario es redirigido a la página de inicio de sesión
Escenario alternativo
A1: 3.b: El usuario selecciona “Cancelar” en el diálogo de confirmación. El sistema cierra el diálogo y devuelve al usuario a la pantalla de “Perfil” con su sesión activa.
Interfaces implicadas en el caso de uso
 

Tabla 8. Caso de Uso 6: Eliminar cuenta

Título	Generación de Menú Semanal Personalizado (RF4)
Descripción	Permite a un usuario autenticado generar un plan de comidas (desayuno, almuerzo, cena) para cinco días de la semana. El menú se calcula automáticamente basándose en las preferencias, objetivos y restricciones definidas en su perfil.
Pre-condición	El usuario ha iniciado sesión y ha completado su perfil en el flujo de registro con toda la información necesaria (dieta, intolerancias, etc.). El usuario se encuentra en la página principal y no tiene un menú activo.
Post-condición	Se genera un nuevo menú semanal, se persiste en la base de datos asociado al usuario y se muestra en la interfaz de la aplicación.
Escenario Principal	
1. El usuario está en la pantalla principal sin menú activo, y pulsa el botón "Genera tu menú semanal". 2. El sistema procesa la petición, y consulta la base de datos y/o la API externa (Spoonacular) para encontrar recetas adecuadas para sus preferencias. Construye el menú para los 5 días, lo guarda y lo devuelve a la aplicación. 3. La aplicación recibe los datos y renderiza el menú semanal en una vista general de la semana, y una pantalla específica para cada día mostrando cada tipo de comida del día seleccionado.	
Escenario alternativo	

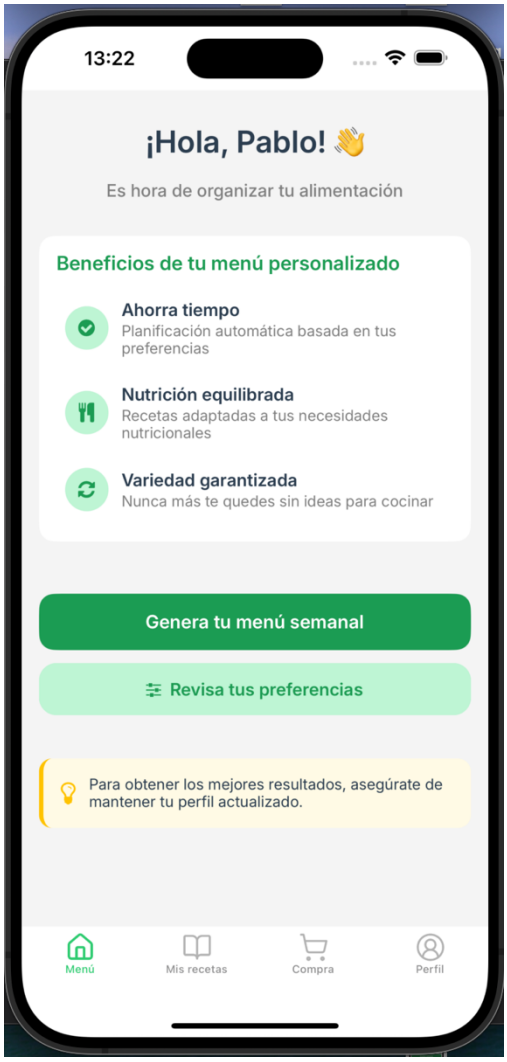
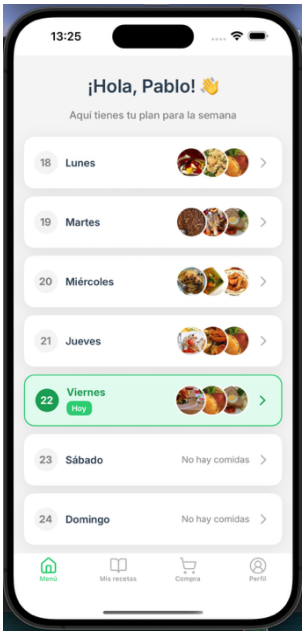
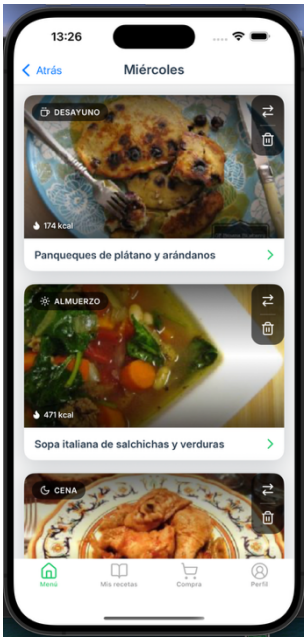
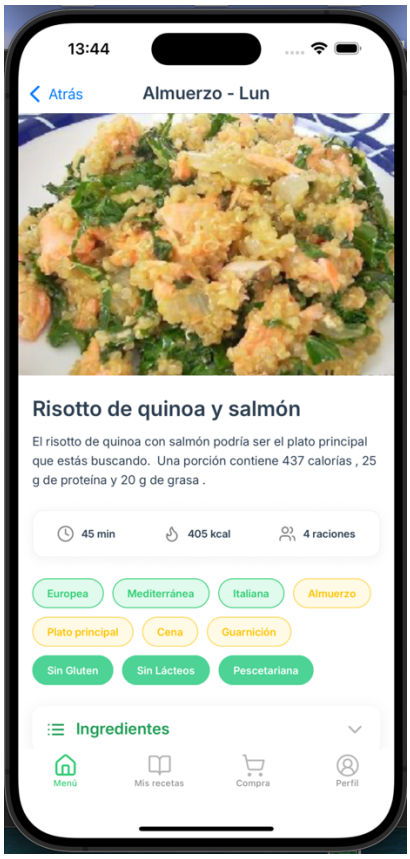
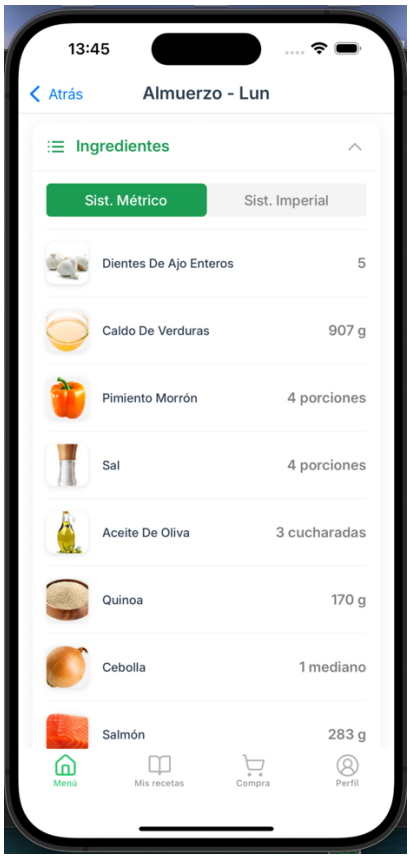
A1: 2.b: No se encuentran suficientes recetas que cumplan con los criterios del usuario. El servidor devuelve un error específico. La aplicación muestra un mensaje al usuario informándole de la situación y sugiriéndole que revise y flexibilice sus preferencias.
Interfaces implicadas en el caso de uso
  

Tabla 9. Caso de Uso 7: Generación de menú semanal personalizado

Título	Visualización Detallada de la receta (RF7)
Descripción	Permite al usuario consultar la información completa de una receta específica que forme parte de su menú semanal, incluyendo ingredientes, instrucciones de preparación, información nutricional y equipo necesario.
Pre-condición	El usuario tiene un menú semanal generado y visible en la pantalla.
Post-condición	El usuario visualiza toda la información detallada de la receta seleccionada.
Escenario Principal	
1. En la página principal donde el usuario está viendo la vista general de su menú semanal, el usuario selecciona un día de la semana. 2. El usuario es redirigido a una página específica para ese día y puede ver una vista previa de las comidas asignadas para ese día (desayuno, almuerzo, cena). A continuación, el usuario pulsa una de las comidas. 3. El usuario es redirigido a una pantalla específica de la comida seleccionada, donde puede ver toda la información detallada, como la lista de ingredientes necesarios con sus cantidades, las instrucciones de preparación, la información nutricional, el equipo necesario y otros datos como el tiempo de preparación o para cuántas raciones está adaptada la receta.	
Escenario alternativo	
A1: 2.b: El usuario ha seleccionado un día con ninguna comida asignada y en la pantalla específica del día no aparece ninguna opción para poder visualizar su información detallada.	

Interfaces implicadas en el caso de uso





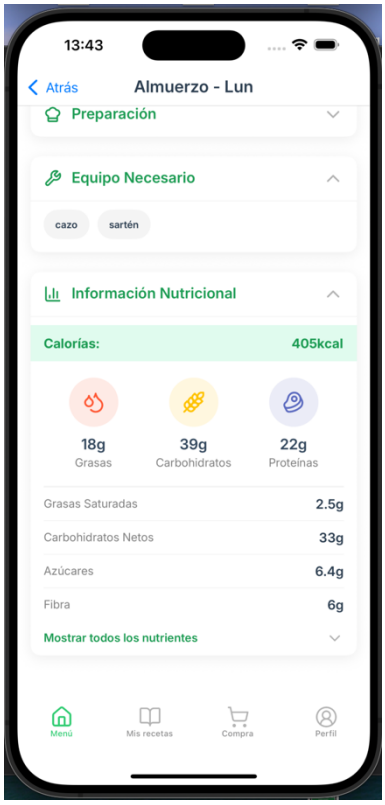


Tabla 10. Caso de Uso 8: Visualización detallada de la receta

Título	Personalización y Edición del Menú (RF5)
Descripción	Permite al usuario modificar su menú semanal generado, cambiando comidas específicas asignadas por otras sugerencias generadas por el sistema. También puede eliminar comidas del menú y asignar nuevas comidas en días que no tenga comidas asignadas.
Pre-condición	El usuario tiene un menú semanal generado y visible en la pantalla.
Post-condición	El menú del usuario se actualiza en la base de datos y los cambios se reflejan inmediatamente en la interfaz.
Escenario Principal	
1. En la pantalla principal donde el usuario está viendo la vista general de su menú semanal, el usuario selecciona un día de la semana. 2. El usuario es redirigido a una página específica para ese día y puede ver una vista previa de las comidas asignadas para ese día (desayuno, almuerzo, cena). A continuación, el usuario selecciona la opción de reemplazar de una de las comidas. 3. El sistema muestra una lista de recetas alternativas, que son compatibles con las preferencias del usuario. El usuario selecciona una nueva receta de la lista. 4. El sistema reemplaza la comida original por la seleccionada, actualiza el menú en la base de datos y refresca la pantalla del menú semanal.	
Escenario alternativo	
A1: Eliminación de comida	

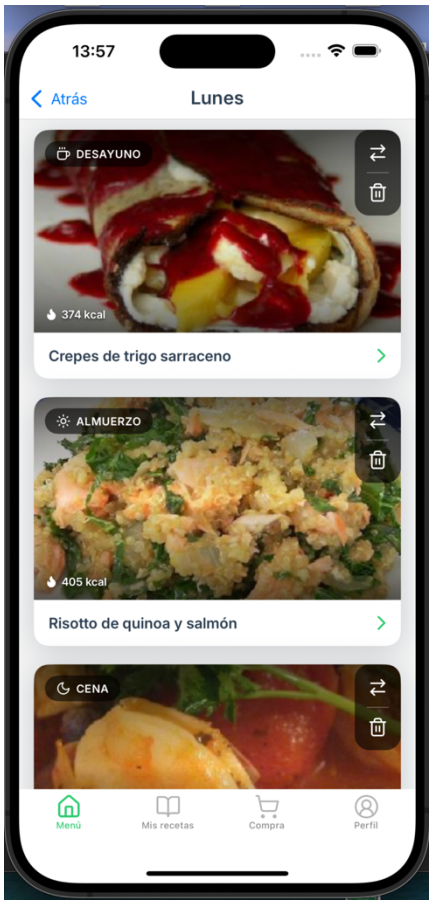
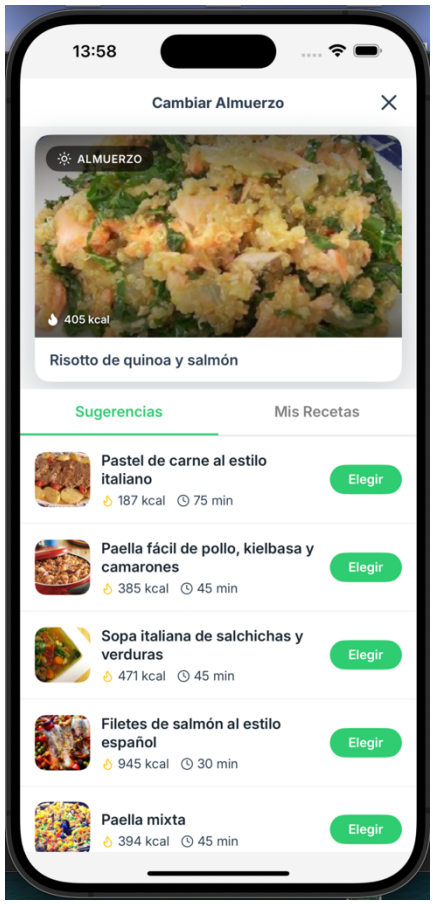
2.b El usuario selecciona la opción de eliminar la comida. 3.b: El sistema muestra un diálogo de confirmación. 4.b: El usuario confirma la acción. 5.b: El sistema elimina la comida de ese día y actualiza la base de datos y la interfaz. A2: No hay sugerencias disponibles 3.b: El sistema no encuentra alternativas adecuadas para la comida seleccionada y muestra un mensaje informativo.
Interfaces implicadas en el caso de uso
 

Tabla 11. Caso de Uso 9: Personalización y edición del menú

Título	Gestión de recetas propias (RF8)
Descripción	Permite a un usuario crear, visualizar, editar y eliminar sus propias recetas personalizadas.
Pre-condición	El usuario ha iniciado sesión en la app y está en la página de “Mis Recetas”.
Post-condición	La colección de recetas del usuario se actualiza en la base de datos (se añade una nueva, se modifica o se elimina una existente). Los cambios son visibles en la sección de “Mis Recetas”.
Escenario Principal	
1. En “Mis Recetas” el usuario ve la lista de sus recetas creadas o un estado vacío con un botón para añadir una receta nueva. A continuación, el usuario pulsa ese botón de añadir. 2. El usuario es redirigido a una pantalla de creación con un formulario vacío donde completa los campos requeridos, añade opcionalmente ingredientes, pasos de preparación y una imagen, y selecciona “Guardar Receta”. 3. El sistema valida y guarda los datos, muestra un mensaje de éxito y redirige al usuario al listado de “Mis Recetas, ya con la nueva receta.	
Escenario alternativo	
A1: Editar una receta existente 1.b: El usuario pulsa la opción de editar de una receta de su lista. 2.b: El sistema muestra el formulario precargado con los datos de esa receta y se continúa con los mismos pasos que en el flujo de creación.	

A2: Eliminar una receta

1.b: El usuario pulsa la opción de eliminar de una receta de su lista.

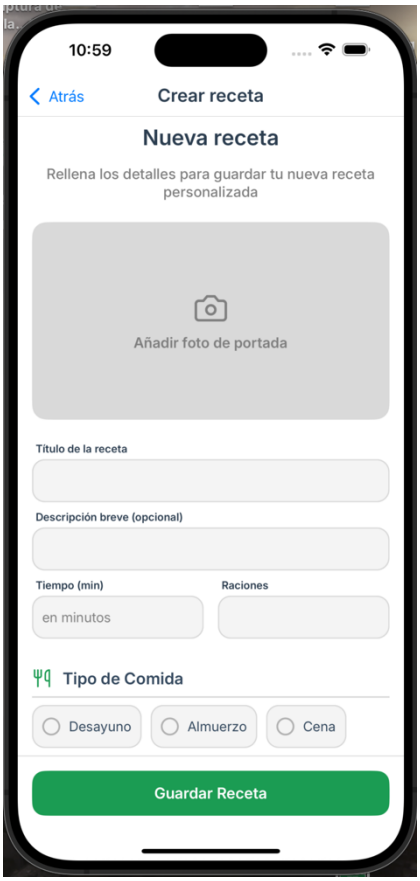
2.b: El sistema muestra un diálogo de confirmación. Si la receta está en uso en el menú, se muestra una advertencia adicional.

3.b: El usuario confirma la eliminación. El sistema borra la receta de la base de datos y actualiza el listado.

A3: Datos del formulario inválidos

4.b: El sistema detecta que faltan campos obligatorios o que hay datos introducidos que son inválidos y muestra un mensaje de error. El usuario permanece en el formulario para corregir los datos.

Interfaces implicadas en el caso de uso



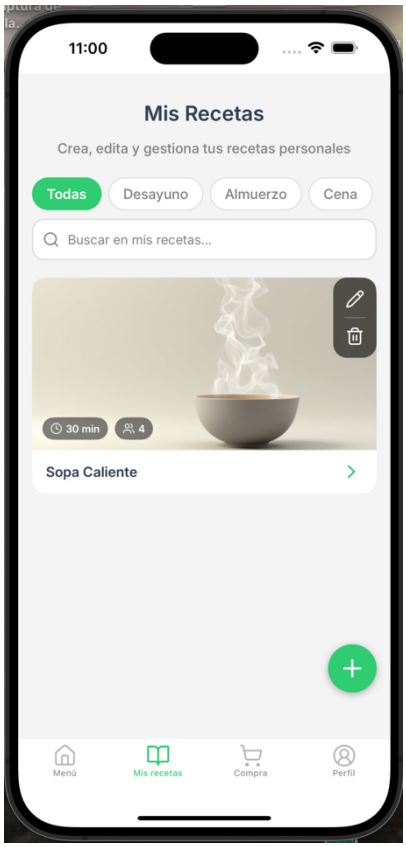


Tabla 12. Caso de Uso 10: Gestión de recetas propias

Título	Añadir Comidas Extra al Menú (RF6)
Descripción	Permite al usuario añadir comidas o bebidas adicionales (Aperitivos, Postres, Snacks, etc.) a cualquier día de su menú semanal, seleccionándolas de una lista de sugerencias o de sus propias recetas.
Pre-condición	El usuario tiene un menú semanal generado y visible en la pantalla.
Post-condición	Se añade una nueva comida a un día específico del menú del usuario. El cambio se persiste en la base de datos y se refleja en la interfaz.
Escenario Principal	

1. En la página principal donde el usuario está viendo la vista general de su menú semanal, el usuario selecciona un día de la semana. 2. El usuario es redirigido a una página específica para ese día y debajo de las tarjetas de las comidas principales hay una tarjeta vacía para "Añadir comida o bebida extra". El usuario pulsa esta opción. 3. El sistema presenta un modal con las categorías de comidas extra disponibles (Snack, Postre, etc.). El usuario selecciona una de estas categorías. 4. El sistema abre el modal de sugerencias mostrando recetas del sistema o propias del usuario que pertenecen a esa categoría. El usuario selecciona una de estas recetas sugeridas. 5. El sistema añade la receta seleccionada como una nueva comida extra en el día correspondiente, actualiza el menú y cierra el modal.
Escenario alternativo
A1: 3.b/4.b: El usuario cancela la selección, cerrando el modal de las categorías o de las sugerencias. El menú permanece sin cambios. A2: No hay sugerencias disponibles 4.b: El sistema no encuentra opciones adecuadas para la comida seleccionada y muestra un mensaje informativo.
Interfaces implicadas en el caso de uso

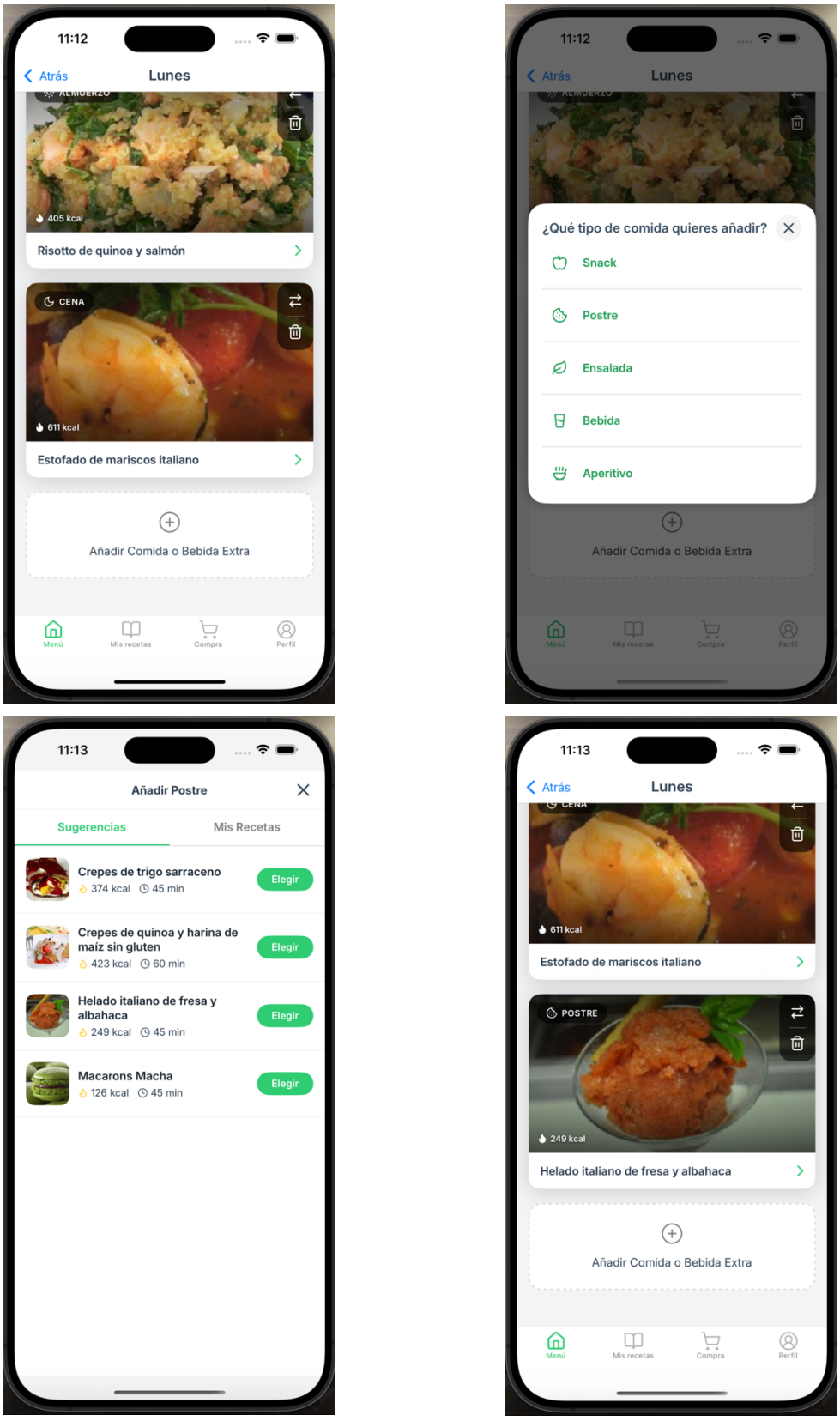


Tabla 13. Caso de Uso 11: Añadir comidas extra al menú

Título	Generación de Lista de la Compra (RF9)
Descripción	Permite al usuario generar una lista consolidada de todos los ingredientes necesarios para su menú semanal, ajustada a un número específico de raciones y agrupada por secciones de supermercado.
Pre-condición	El usuario ha iniciado sesión y tiene un menú semanal activo.
Post-condición	Se genera y muestra una lista de la compra con todos los ingredientes, cantidades y categorías.
Escenario Principal	
1. El usuario navega a la pestaña “Compra” y pulsa la opción “Generar lista de la compra”. 2. El sistema presenta un modal solicitando el número de raciones para las que se debe calcular la lista. 3. El usuario introduce el número de raciones y confirma. 4. El sistema envía la petición al servidor, que se encarga de generar la lista de la compra. Posteriormente, la aplicación se la muestra al usuario, organizada por secciones de supermercado desplegables y el usuario puede interactuar con la lista, marcando los ingredientes que ya ha comprado.	
Escenario alternativo	
A1: 3.b: El usuario cierra el modal del número de raciones y cancela la generación de la lista.	

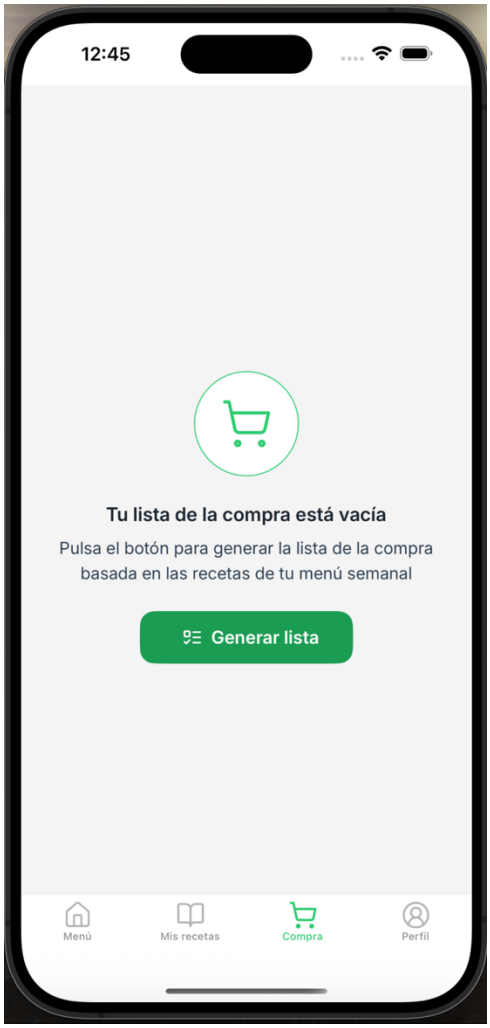
A2: Regenerar la lista

1b: El usuario ya tiene una lista generada pero ha hecho cambios en su menú. Selecciona la opción de actualizar y se sigue con el mismo flujo principal.

A3: Menú semanal vacío

2b: El sistema comprueba que tiene un menú vacío y no muestra el modal de raciones. Muestra un mensaje informativo y se cancela la generación de la lista.

Interfaces implicadas en el caso de uso



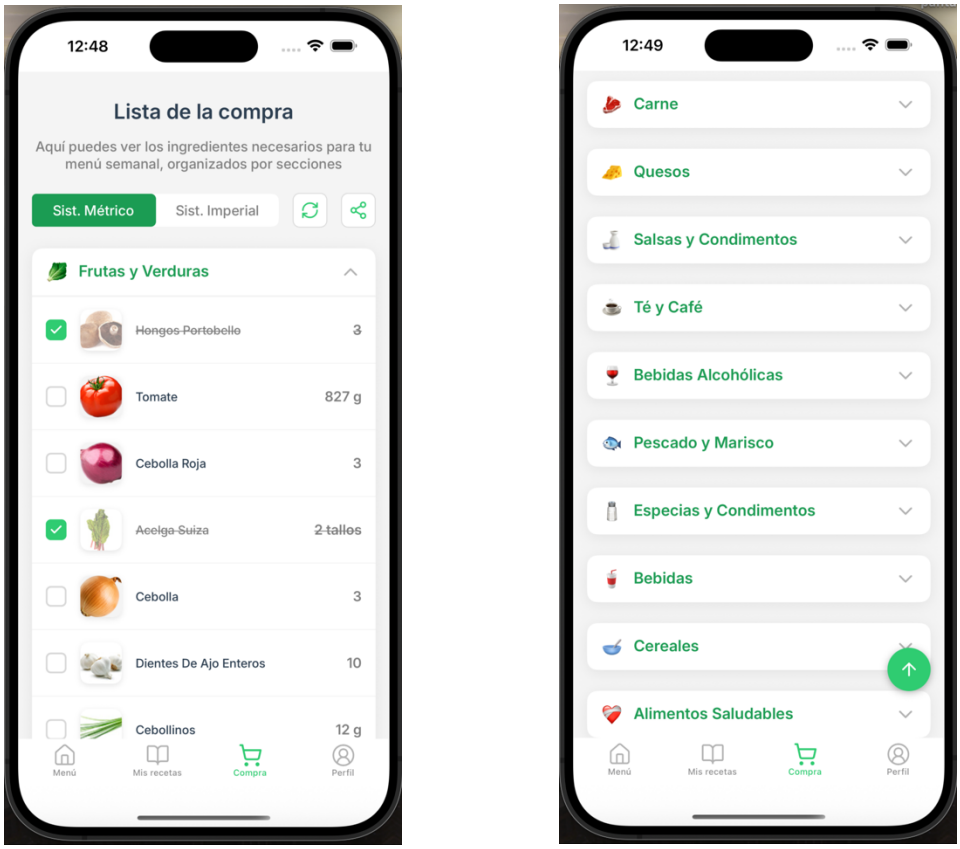


Tabla 14. Caso de Uso 12: Generación de lista de la compra

Título	Exportar y Compartir Lista de la Compra (RF10)
Descripción	Permite al usuario exportar su lista de la compra generada a un archivo PDF y compartirlo a través de las aplicaciones de su dispositivo móvil.
Pre-condición	El usuario ha generado previamente su lista de la compra.
Post-condición	Se genera un archivo PDF con la lista de la compra y se abre la interfaz nativa de compartición del sistema operativo (iOS/Android).
Escenario Principal	

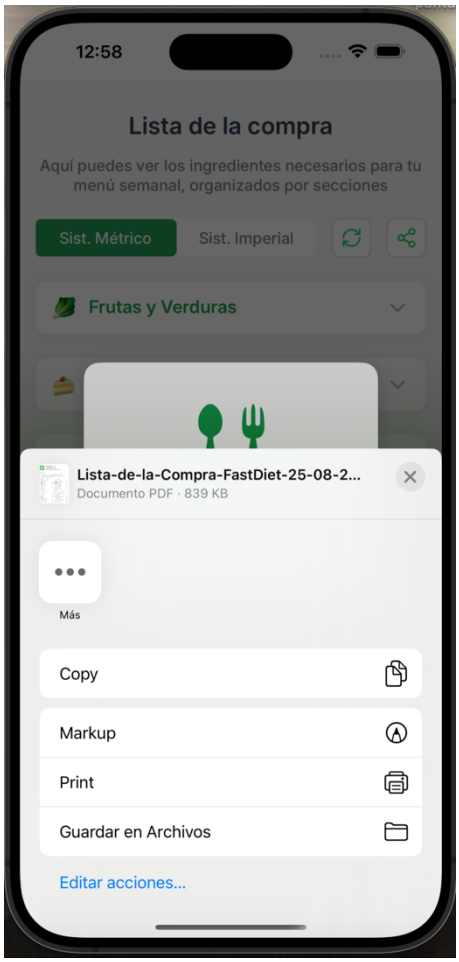
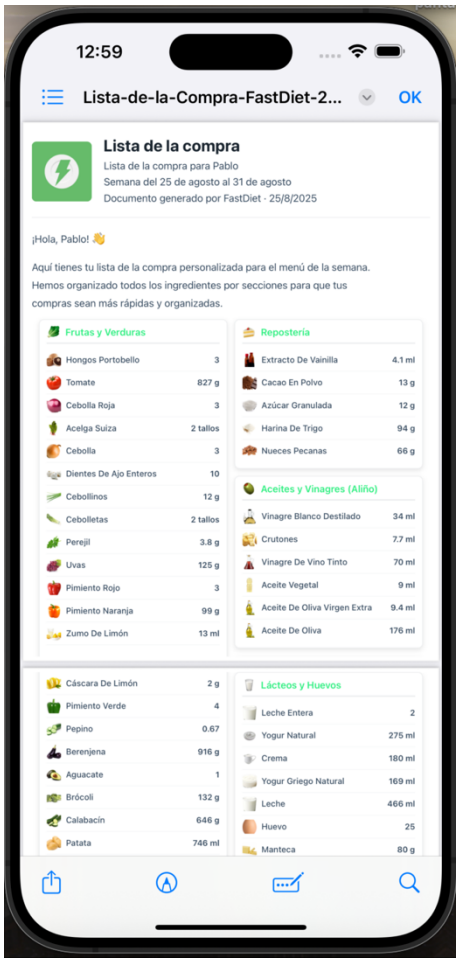
1. El usuario se encuentra en “Compra” con su lista generada y selecciona el botón de compartir. 2. El sistema genera dinámicamente un documento HTML con la lista de la compra formateada y convierte ese HTML en un archivo PDF. 3. El sistema invoca la función de compartición nativa del dispositivo, adjuntando el PDF generado. 4. El usuario elige la app o contacto con el que desea compartir el archivo.
Interfaces implicadas en el caso de uso
 

Tabla 15. Caso de Uso 13: Exportar y compartir lista de la compra

4

Diseño

En este capítulo se documentan las decisiones estratégicas tomadas durante la fase de diseño de FastDiet, abarcando la estructura de la base de datos, la arquitectura general del sistema, la especificación del servidor y la organización interna de la aplicación móvil. Los criterios que han guiado las decisiones son: coherencia del dominio, mantenibilidad, rendimiento y escalabilidad progresiva.

4.1. Base de datos

4.1.1. Elección de un modelo relacional y MySQL

Para el almacenamiento de datos en FastDiet se ha optado por un sistema de base de datos relacional en lugar de una base de datos no relacional. Esta decisión se ha basado en las siguientes consideraciones:

- **Estructura de datos fuertemente relacionada:** el dominio de la aplicación presenta entidades claramente definidas (usuario, recetas, ingredientes, menús, etc.) que mantienen relaciones 1:1, 1:N y N:M. Un modelo

relacional es ideal para representar estas entidades y conexiones de forma coherente.

- **Necesidad de integridad referencial:** es fundamental garantizar que, por ejemplo, una receta asociada a un menú o un ingrediente de una receta siempre correspondan a registros válidos.

- **Consultas complejas:** muchas operaciones requieren combinaciones de datos mediante joins, como recuperar un menú semanal junto a sus recetas e ingredientes o recopilar los ingredientes de varias recetas para generar la lista de la compra. Las bases de datos relacionales están optimizadas para este tipo de consultas.

- **Normalización de datos:** la eliminación de redundancia y la consistencia en la actualización de datos son esenciales para evitar errores y duplicidad de información.

Dentro de las bases de datos relacionales, se ha optado por MySQL por las siguientes razones:

- **Compatibilidad y soporte:** excelente integración con SQLAlchemy y herramientas de migración como Alembic, facilitando la gestión y los cambios en el diseño de la base de datos durante el desarrollo.

- **Estabilidad y rendimiento:** MySQL es un Sistema Gestor de Bases de Datos ampliamente utilizado y probado en producción durante décadas, y está comprobado que tiene un alto rendimiento en operaciones de lectura que involucran múltiples tablas relacionadas, algo frecuente en FastDiet.

4.1.2. Modelo lógico de datos

En la Figura 8 se presenta el diagrama de base de datos diseñado para la aplicación FastDiet, generado con MySQL Workbench. El modelo refleja las entidades principales de la aplicación, así como sus relaciones.

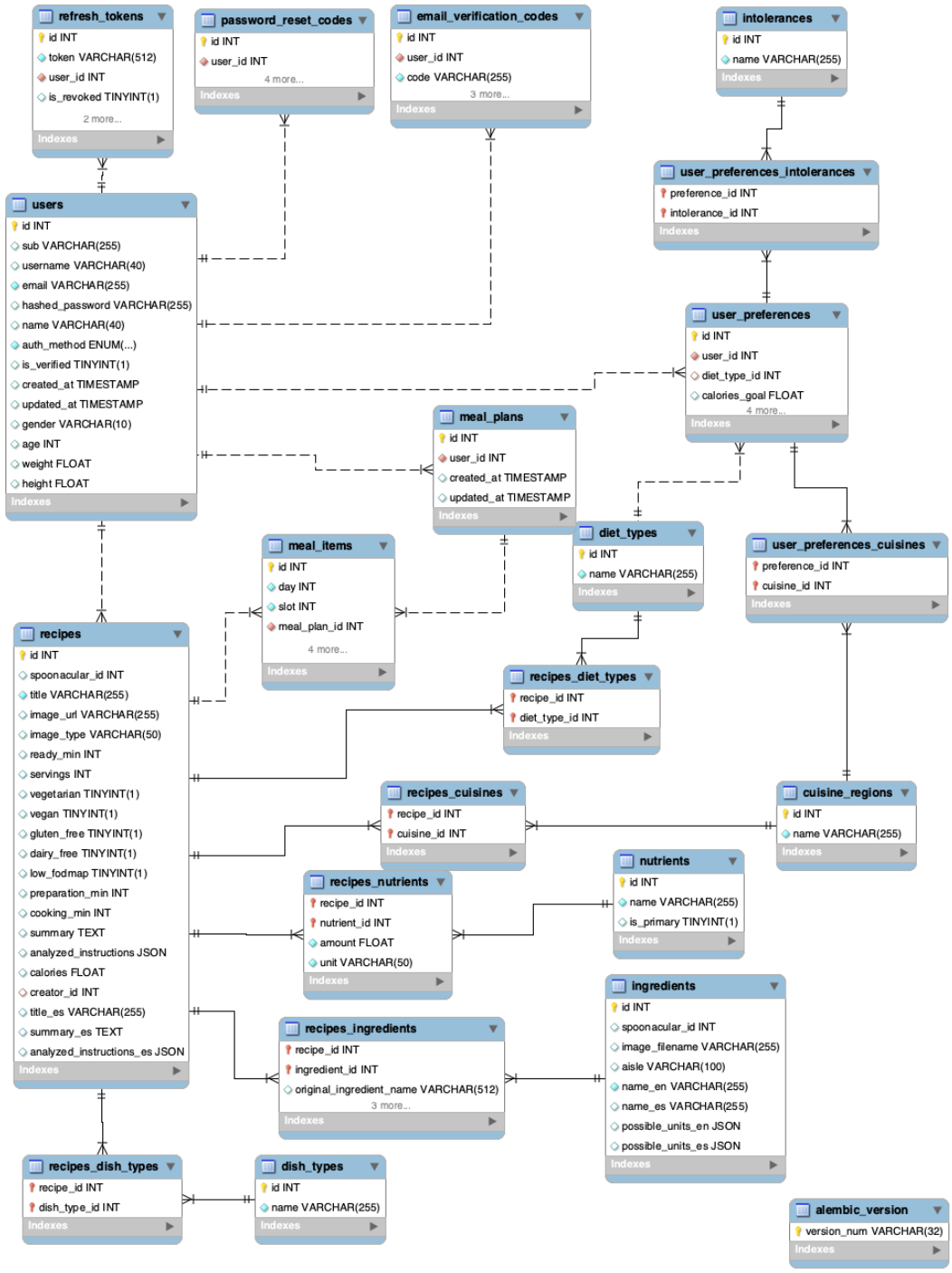


Figura 8. Diseño de la base de datos

A continuación, se describen las tablas y relaciones más relevantes:

- **users:** contiene la información principal de cada usuario registrado, incluyendo credenciales y datos personales básicos. Mantiene relaciones con tablas de verificación de correo, restablecimiento de contraseña, planes de comida, recetas y preferencias.

- **user_preferences:** contiene información más detallada de las preferencias del usuario, a través de sus atributos y relaciones podemos conocer información como dieta seleccionada, tipos de cocina preferidos, intolerancias, nivel de actividad, objetivo o las calorías diarias necesarias.

- **meal_plans y meal_items:** meal_plans agrupa un conjunto de comidas planificadas para un usuario, organizadas en días. La tabla meal_items representa cada comida dentro del plan semanal, incluyendo su relación con la receta correspondiente (recipes).

- **recipes:** tabla central que almacena la información de cada receta, con atributos como calorías, tiempo de preparación, número de raciones o instrucciones de preparación. Se relaciona con tablas auxiliares para tipos de plato (recipes_dish_types), nutrientes (recipes_nutrients), dietas (recipes_diet_types), ingredientes (recipes_ingredients) y tipos de cocina (recipes_cuisines).

- **ingredients:** contiene información detallada de cada ingrediente, incluyendo nombres, unidades posibles o secciones de supermercado.

4.2. Arquitectura general

La aplicación FastDiet sigue una arquitectura cliente-servidor de múltiples componentes, en la que la aplicación móvil actúa como cliente (frontend),

comunicándose con un servidor backend que gestiona la lógica de negocio, el acceso a la base de datos y la interacción con APIs externas.

En la Figura 9 se muestra un diagrama que ilustra los componentes principales de esta arquitectura y sus interacciones:

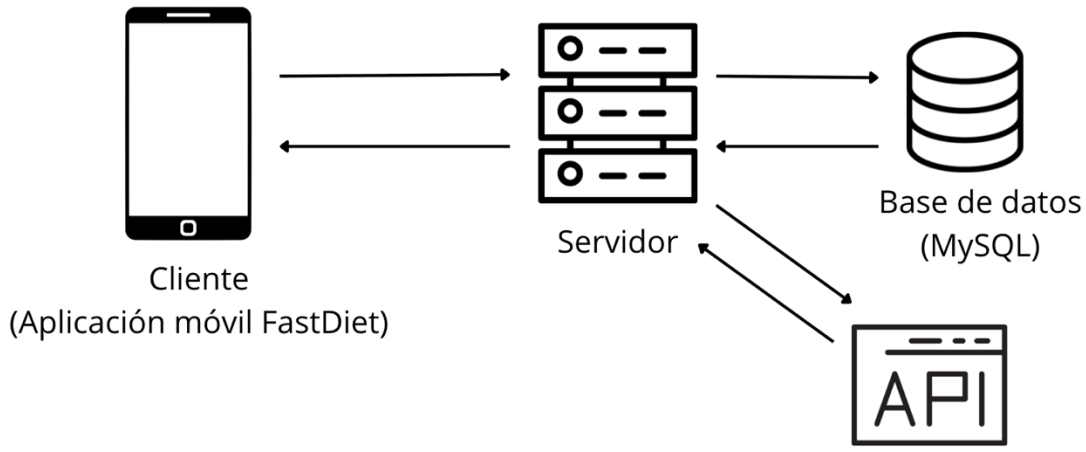


Figura 9. Arquitectura de alto nivel (Cliente-Servidor)

Descripción de los componentes:

- **Cliente – Aplicación móvil (React Native)**

  Se trata de la aplicación desarrollada con React Native que se ejecuta en el dispositivo del usuario (iOS o Android). El usuario interactúa con la aplicación, y ésta construye y envía peticiones HTTP al servidor backend.

- **Servidor Backend (FastAPI)**

  Implementa las funcionalidades y es el intermediario de todo el sistema. Sus responsabilidades son:

  - **Gestionar la lógica de negocio:** ejecuta las operaciones complejas como el cálculo de necesidades calóricas o el algoritmo de selección de recetas para generar menús o sugerencias personalizadas.

- **Proteger y gestionar los datos:** es el único componente que tiene acceso directo a la base de datos MySQL. Recibe las peticiones del cliente, las valida y realiza las operaciones de lectura o escritura correspondientes.

- **Comunicarse con servicios externos:** gestiona las llamadas a la API de Spoonacular para buscar nuevas recetas cuando no se encuentran en la base de datos, también se llama a la API para ayudar a generar la lista de la compra.

- **Gestionar la seguridad:** se encarga de la autenticación y autorización de los usuarios.

- **Base de datos (MySQL)**

  Almacena de forma persistente la información de usuarios, preferencias, recetas, ingredientes y planes de comida. La comunicación entre el backend y la base de datos se realiza mediante un ORM (SQLAlchemy), lo que facilita la abstracción y reduce el riesgo de errores de consulta.

- **API Externa – Spoonacular API**

  Fuente de datos para recetas, ingredientes y valores nutricionales. El servidor solicita información a esta API para enriquecer los planes de comida y ofrecer una mayor variedad al usuario. El servidor actúa como un intermediario que cachea los resultados de esta API en la base de datos propia para mejorar el rendimiento y reducir la dependencia del servicio externo.

La adopción de esta arquitectura cliente-servidor ofrece ventajas significativas para el proyecto:

- **Escalabilidad:** permite escalar el servidor y el cliente de forma independiente. Si la aplicación experimenta un aumento de usuarios, se pueden dedicar más recursos al servidor y a la base de datos sin necesidad de realizar ningún cambio en la aplicación móvil que los usuarios tienen instalada.

- **Centralización de la lógica y seguridad:** toda la lógica de negocio sensible y el acceso a los datos están centralizados y protegidos en el servidor.

- **Separación de responsabilidades y mantenibilidad:** el código está mucho más organizado y más fácil de mantener. El desarrollo del frontend puede centrarse en los aspectos visuales y la experiencia de usuario, mientras que el desarrollo del backend se enfoca en la eficiencia de la lógica, el rendimiento de las consultas y la seguridad de los datos.

4.3. Servidor – Arquitectura REST

El servidor de FastDiet se ha desarrollado siguiendo los principios de la arquitectura REST (*Representational State Transfer*), un estilo arquitectónico orientado a sistemas distribuidos en el que los recursos se identifican mediante URLs y se manipulan a través de métodos estándar del protocolo HTTP. En este contexto:

- **GET:** obtiene recursos del servidor (por ejemplo, obtener el menú semanal de un usuario).

- **POST:** crea nuevos recursos (por ejemplo, registrar un nuevo usuario).

- **PUT/PATCH:** modifica recursos existentes, actualización completa o parcial (por ejemplo, actualizar alguna de las preferencias alimenticias).

- **DELETE:** eliminar recursos (por ejemplo, borrar una receta guardada).

El servidor también utiliza códigos de estado HTTP para informar del resultado de cada operación:

- **2XX:** Éxito (200, 201, 204).
- **4XX:** Errores del cliente (400, 401, 404).
- **5XX:** Errores del servidor (500).

Ventajas de la arquitectura REST en FastDiet:

- **Escalabilidad:** cada petición es independiente, facilitando la distribución de carga y la replicación del backend.
- **Interoperabilidad:** el backend puede ser consumido por cualquier cliente que entienda HTTP (apps móviles, aplicaciones web, etc.).
- **Mantenibilidad:** el diseño modular permite añadir o modificar endpoints sin afectar al resto de la aplicación.

4.3.1. Patrón de diseño y organización interna

Para organizar la lógica interna del servidor se ha seguido un patrón de arquitectura por capas, una estrategia de diseño que separa las responsabilidades del software en distintas secciones lógicas. La implementación sigue una estructura inspirada en el patrón MVC (Modelo – Vista – Controlador), adaptado al ecosistema FastAPI:

- **Capa de Presentación (Controladores/Endpoints):** este es el punto de entrada de la aplicación y su principal responsabilidad es gestionar todas las

peticiones HTTP. Recibe las solicitudes, valida los datos de entrada utilizando esquemas predefinidos y orquesta la respuesta, delegando las operaciones complejas a la capa de lógica de negocio. No contiene lógica de negocio, solo se encarga de la comunicación con el cliente.

- **Capa de Lógica de Negocio (Servicios):** contiene la lógica de negocio principal y más compleja de la aplicación. Aquí residen los algoritmos, como el generador de menús personalizados o el creador de la lista de la compra. Esta capa es independiente, lo que permite que sea reutilizable y más fácil de probar de forma aislada

- **Capa de Acceso a Datos (CRUD y Modelos):** abstrae la comunicación con la base de datos. Se divide en dos componentes:
    - **Modelos:** definen la estructura de los datos, las tablas y sus relaciones utilizando un ORM como SQLAlchemy. Representan el "Modelo" de la aplicación.

    - **Funciones CRUD:** contienen las operaciones directas y reutilizables con la base de datos (Crear, Leer, Actualizar, Borrar).

- **Capas de Soporte (Infraestructura y Definición de Datos)**
    - **Definición de Datos (Esquemas):** define la estructura de los datos para la validación y serialización en las peticiones y respuestas de la API, usando herramientas como Pydantic.

    - **Infraestructura y Configuración:** centraliza la gestión de la configuración (variables de entorno), la conexión a la base de datos, la

seguridad (gestión de contraseñas, tokens JWT) y otros servicios transversales.

Esta estructura en capas bien definida, ilustrada en la Figura 10, ha sido fundamental para construir un backend robusto, organizado y fácil de mantener a lo largo del desarrollo del proyecto.

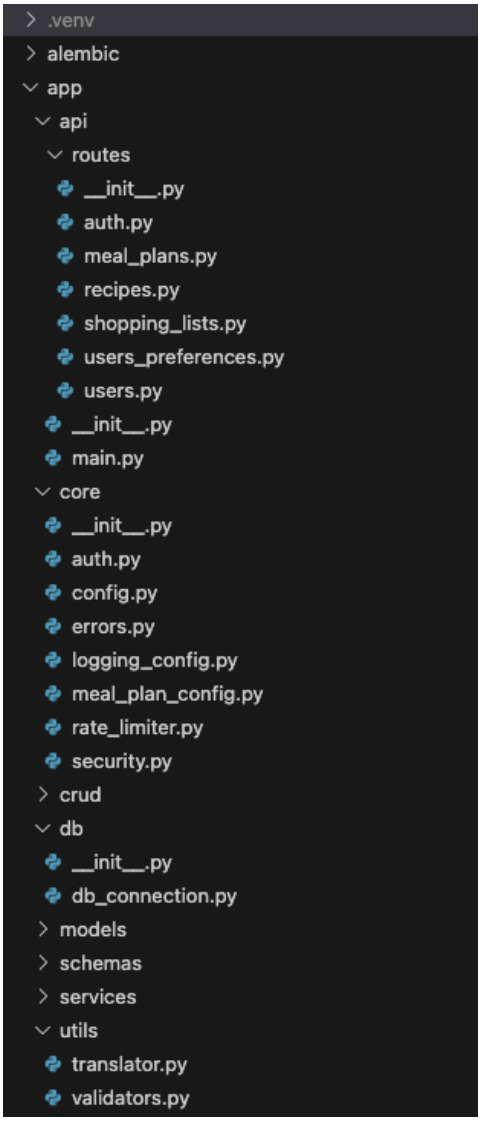


Figura 10. Estructura del proyecto del servidor

4.3. Estructura de la Aplicación móvil

La aplicación de móvil, desarrollada con React Native y Expo, sigue una arquitectura modular (Figura 11) que prioriza la organización y la separación de responsabilidades para facilitar su mantenimiento y escalabilidad. La estructura del proyecto se articula en torno a varios conceptos clave:

- **Navegación basada en ficheros:** se utiliza Expo Router para gestionar la navegación de la aplicación. Esto significa que la estructura de rutas y pantallas se define directamente por la organización de los archivos dentro de la carpeta app/. Se han definido dos grupos de rutas principales: (auth) para el flujo de autenticación y registro y (home) para las pantallas principales de la aplicación una vez que el usuario ha iniciado sesión.

- **Arquitectura por Componentes y Separación de responsabilidades:** La base del código se fundamenta en una estricta separación de responsabilidades, dividiendo la aplicación en capas lógicas:

    - **Componentes de Interfaz:** contiene elementos visuales reutilizables.

    - **Gestión de Estado Global:** se emplea un sistema de gestión de estado centralizado para compartir información crítica a través de la aplicación, como los datos del usuario autenticado o las preferencias, garantizando una única fuente de verdad.

    - **Lógica de Estado Reutilizable (Hooks):** la lógica compleja con estado se encapsula en unidades de código reutilizables (hooks), promoviendo la modularidad y evitando la duplicación de código.

- **Capa de Servicio de API Centralizada:** toda la comunicación con el backend se abstrae en una capa de servicio dedicada. Este módulo actúa como un único punto de contacto entre el frontend y el servidor, encapsulando la lógica para realizar peticiones HTTP. Esta centralización es clave para implementar funcionalidades transversales de forma eficiente mediante el uso de interceptores:

  1. **Interceptor de Petición (Request):** se encarga de adjuntar automáticamente el token de autenticación (JWT) y la cabecera del idioma actual en cada petición enviada al servidor. Esto elimina la necesidad de repetir esta lógica en cada llamada a la API.

  2. **Interceptor de Respuesta (Response):** gestiona de forma transparente la expiración de la sesión. Si una petición falla por un token expirado (error 401 Unauthorized), el interceptor intenta obtener un nuevo token de acceso usando el token de refresco. Si tiene éxito, reintenta la petición original de forma automática sin que el usuario lo note, mejorando significativamente la experiencia de uso.

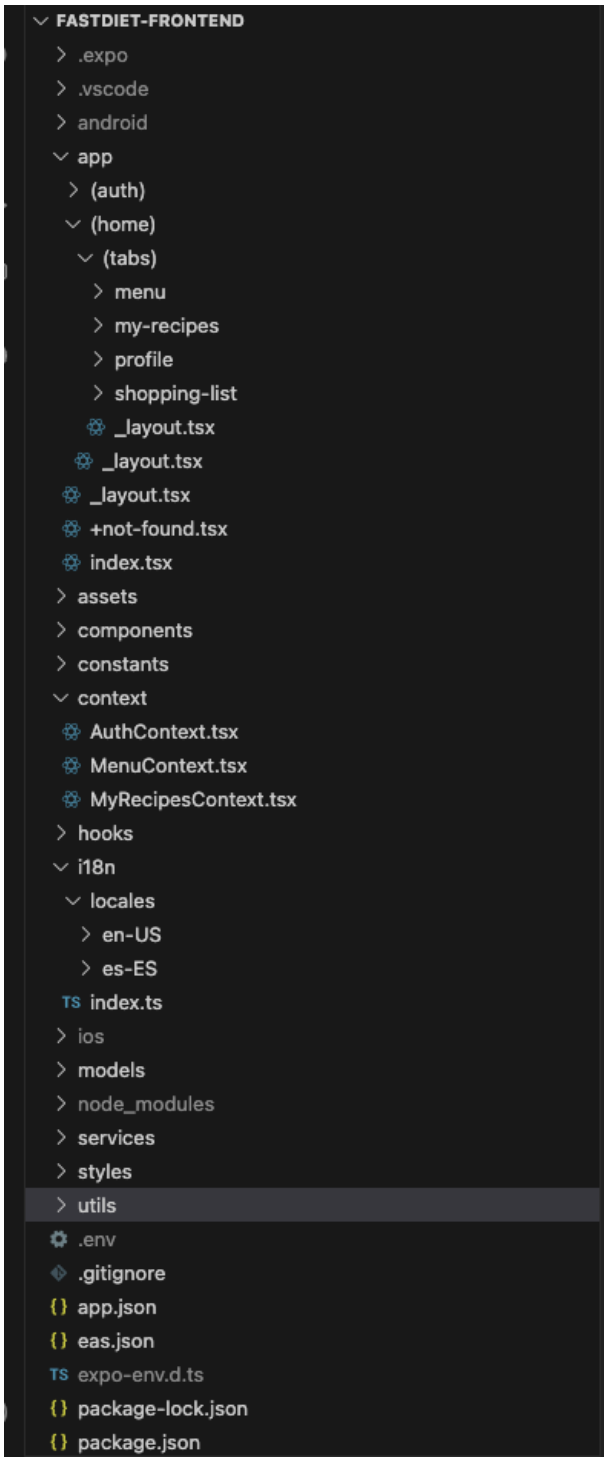


Figura 11. Estructura del proyecto de React Native

5

Implementación, Pruebas y Despliegue

En este capítulo se detalla el proceso de construcción y validación de FastDiet, materializando las decisiones de diseño expuestas en el capítulo anterior. La sección principal documenta la implementación de las funcionalidades, organizada según el enfoque iterativo definido en la metodología del proyecto. Para cada iteración, se describen en detalle los desarrollos realizados tanto en el backend como en el frontend. Finalmente, se abordarán las estrategias de pruebas y despliegue para asegurar la calidad y viabilidad del producto final.

5.1. Implementación por Iteraciones

5.1.1. Primera Iteración: Autenticación y perfil de usuario

En la primera iteración se implementó el sistema de autenticación y la gestión del perfil del usuario. Este bloque era fundamental para sentar las bases de la aplicación, ya que todas las funcionalidades posteriores dependen de la existencia de un usuario identificado con un perfil completo.

Los requisitos funcionales implementados en esta fase fueron:

- RF1 – Registro de usuario
- RF2 – Inicio de sesión
- RF3 – Recuperación de contraseña
- RF11 – Gestión del perfil
- RF12 – Cálculo de calorías objetivo
- RF13 – Cerrar sesión
- RF14 – Eliminar cuenta

5.1.1.1. Backend

En el backend se implementaron un conjunto de endpoints robustos para cubrir el ciclo de vida del usuario:

- **Sistema de Autenticación:**

  Se crearon dos mecanismos de inicio de sesión para ofrecer flexibilidad al usuario:

  1. **Inicio de sesión tradicional:** soporta la autenticación mediante correo electrónico o nombre de usuario y contraseña. El sistema valida que el usuario exista, que su cuenta esté verificada y que el hash de la contraseña proporcionada coincida con el almacenado en base de datos, utilizando el robusto algoritmo bcrypt.

2. **Inicio de sesión con Google (OAuth 2.0):** un endpoint dedicado que recibe y valida el token de identidad provisto por la API de Google para autenticar al usuario de forma segura y sin contraseñas.

En ambos casos, si la autenticación es exitosa, el sistema genera y devuelve un par de JSON Web Tokens (JWT): un access token de corta duración para autorizar peticiones a la API y un refresh token de mayor duración para renovar la sesión sin necesidad de que el usuario vuelva a introducir sus credenciales. Además, junto con los tokens se devuelve información básica del usuario.

- **Registro y recuperación de contraseña:**

  Ambos procesos comparten un mecanismo de seguridad común: la verificación por correo electrónico. Se requiere introducir un correo válido y el sistema genera un código aleatorio de 6 dígitos de un solo uso, lo almacena temporalmente con una caducidad definida y lo envía al correo del usuario. Posteriormente, un endpoint se encarga de verificar que el código introducido por el usuario es correcto para proceder:

  - **Registro:** tras la validación, el usuario es autorizado a continuar con el flujo de completar su perfil.

  - **Recuperación de contraseña:** la validación del código permite establecer una nueva contraseña que debe cumplir política de seguridad estricta (requiriendo números, mayúsculas, minúsculas y caracteres especiales).

- **Gestión y configuración del perfil:**

  El proceso de registro se organiza en varias fases para recopilar información necesaria para la generación del menú:

  1. **Información básica:** nombre, nombre de usuario, sexo, edad, peso y altura.
  2. **Nivel de actividad semanal.**
  3. **Objetivo:** perder peso, mantenerse o ganar masa muscular.
  4. **Tipo de dieta** (equilibrada, vegetariana, vegana, keto, paleo, etc.).
  5. **Cocinas preferidas** (española, mediterránea, asiática, americana, etc.).
  6. **Intolerancias alimentarias** (gluten, lácteos, huevo, frutos secos, etc.).

  Para mayor flexibilidad, se implementó un conjunto de endpoints PATCH, que permiten actualizaciones parciales y eficientes de las preferencias, tanto en el registro como en la edición posterior del perfil.

- **Cálculo de calorías objetivo:**

  Las calorías diarias recomendadas se calculan automáticamente en base a la fórmula de **Mifflin-St Jeor**, considerada una de las más precisas para estimar la Tasa Metabólica Basal (TMB) teniendo en cuenta el sexo, altura, peso y edad del usuario, y se aplican un multiplicador según el nivel de actividad física (sedentaria, ligera, moderada, alta o muy alta) y un ajuste calórico según el objetivo del usuario (perder peso, mantenerse o ganar masa muscular).

  Este cálculo no es estático, el sistema está diseñado para recalcular automáticamente este valor cada vez que el usuario modifica un dato relevante para el cálculo.

- **Gestión de cuenta:**

    - **Cerrar sesión:** invalida la sesión activa del usuario, asegurando que los tokens emitidos no puedan ser utilizados para accesos posteriores.

    - **Eliminar la cuenta:** se elimina permanentemente los datos del usuario, incluyendo menú, recetas y cualquier información asociada.

5.1.1.2. Frontend

El desarrollo en la aplicación móvil se enfocó en crear una experiencia de usuario intuitiva y completa para todas las funcionalidades de esta iteración:

- **Inicio de sesión:**

  Una pantalla que permite autenticación tradicional (correo/usuario + contraseña) o con Google.

- **Registro:**

  Una pantalla para introducir el correo y contraseña, otra para validar el código recibido, y a continuación, se inicia el flujo de completar el perfil.

- **Recuperación de contraseña:**

  Se segmentó el proceso en tres pantallas para una mayor claridad: solicitud de correo, introducción del código y establecimiento de la nueva contraseña.

- **Completar perfil (flujo de varias pantallas):**

Pantallas secuenciales para introducir información del usuario y preferencias:

1. Información básica
2. Nivel de actividad
3. Objetivo

4. Dieta
5. Cocinas preferidas
6. Intolerancias

- **Perfil de usuario:**

  Organizado en varias secciones claras para una fácil navegación y gestión del perfil por parte del usuario:

  - **Cabecera principal:** muestra de forma destacada la información más relevante como nombre, calorías diarias necesarias, peso y altura.

  - **Mi Plan:** muestra las preferencias del usuario separadas por apartados (dieta, objetivo, etc.), cada apartado da acceso a una página para ver o editar la información de forma individual.

  - **Mi Perfil:** datos personales y posibilidad de cambiar la contraseña.

  - **Cuenta:** opciones para cerrar sesión y eliminar la cuenta, ambas con confirmación requerida mediante diálogos de confirmación (Alert) para prevenir operaciones accidentales por parte del usuario.

5.1.2. Segunda Iteración: Generación y Gestión del Menú Semanal

El objetivo de esta iteración fue implementar la funcionalidad central y de mayor valor de FastDiet: la generación, visualización y personalización del menú semanal. Esta fase construye directamente sobre la base de la primera iteración, utilizando el perfil completo del usuario para ofrecer una experiencia dietética a medida. Los requisitos funcionales implementados en esta fase fueron:

- RF4 – Generación de menú semanal personalizado
- RF5 – Personalización y edición del menú semanal
- RF7 – Visualización detallada de recetas

5.1.2.1. Backend

El desarrollo en el servidor se centró en la creación de una lógica de negocio compleja para la generación de menús y los servicios de apoyo necesarios.

- **Algoritmo de Generación de Menús:**

  Se implementó un robusto algoritmo para la creación del plan de comidas, que sigue una estrategia híbrida y con mecanismos de fallback para maximizar la probabilidad de éxito:

  1. **Preparación de Parámetros:** el servicio recopila todas las preferencias del usuario (dieta, intolerancias, cocinas preferidas) y su objetivo calórico diario. Este objetivo se distribuye por comida, asignando un 25% a los desayunos, un 40% a los almuerzos y un 35% a las cenas, con un margen de tolerancia, pero intentando que las comidas seleccionadas se acerquen a esos valores.

  2. **Búsqueda en la Base de Datos Local:** el primer intento para obtener recetas candidatas (5 para cada tipo de comida) se realiza con la base de datos propia de la aplicación. Esta consulta es la más restrictiva , aplicando todos los filtros.

  3. **Búsqueda en API Externa (Spoonacular) con Fallback:** si la base de datos local no devuelve suficientes resultados, el sistema recurre a la API de Spoonacular con una estrategia de tres intentos para maximizar las posibilidades de obtener una generación del menú exitosa:

     - **Primer Intento:** se realiza una búsqueda con todos los filtros originales (tipo de comida, dieta, intolerancias, cocinas y calorías).

- **Segundo Intento (Fallback):** si el anterior no encuentra suficientes candidatas, se relaja la búsqueda eliminando el filtro de calorías. Esta decisión se basa en que el usuario podrá ajustar su ingesta calórica final mediante comidas extra (snacks, aperitivos, etc.).

- **Tercer Intento (Fallback):** si aún no hay suficientes recetas, se elimina también el filtro de cocinas preferidas. Los filtros de tipo de dieta e intolerancias se mantienen en todos los casos, ya que son requisitos de salud y preferencia innegociables.

4. **Ensamblaje del Menú y Caching:** con el conjunto de recetas obtenidas, el sistema ensambla el menú de lunes a viernes, minimizando la repetición de platos, y si es inevitable, espaciarlos lo máximo posible. Toda receta nueva obtenida de la API de Spoonacular se almacena íntegramente en la base de datos. Esta estrategia de caching reduce progresivamente la dependencia y el coste de la API externa, mejorando la velocidad de futuras generaciones de menús.

- **Detalle de Receta:**

  Gracias a la estrategia de caching, la visualización detallada de una receta es muy eficiente. Se implementó un endpoint GET /recipes/{recipe_id} que simplemente recupera la información completa de la receta (ingredientes, instrucciones, etc.) directamente desde la base de datos, evitando llamar a la API externa cada vez que el usuario quiera ver los detalles de una receta, y proporcionando así una respuesta rápida al cliente.

- **Sugerencias de Reemplazo:**

  Para la funcionalidad de cambiar platos, se creó un endpoint específico para buscar sugerencias manteniendo los filtros de dieta, cocinas e intolerancias,

pero se omite el filtro de calorías, otorgando al usuario la libertad de personalizar su menú. La lógica de búsqueda y de caching es muy similar a la de generación del menú.

5.1.2.2. Frontend

En la aplicación móvil, el desarrollo se centró en crear una interfaz clara para la visualización e interacción con el menú semanal.

- **Visualización del Menú Semanal:**

  La pantalla principal lista los días de la semana. Al seleccionar un día, se muestra una vista detallada con las comidas (desayuno, almuerzo, cena) representadas por un componente reutilizable "MealCard", que muestra una imagen de la receta, el título y alguna información de visualización rápida como las calorías de la comida.

- **Interacción y Edición del Menú:**

  Cada "MealCard" incorpora dos acciones principales representadas con iconos intuitivos:

  - **Reemplazar:** al pulsar se abre un modal mostrando las sugerencias de sustitución, manteniendo visible la comida que se pretende cambiar.

  - **Eliminar:** se muestra un diálogo de confirmación (Alert) para prevenir borrados accidentales.

  Además, cuando un espacio de comida no tiene ninguna receta asignada, se muestra una tarjeta vacía con un botón de añadir. Al pulsarla, se abre el mismo modal de sugerencias pero en modo añadir.

Tras realizar cualquier acción de manera exitosa muestra un toast de confirmación personalizado, reforzando el feedback visual para el usuario.

- **Pantalla de Detalle de Receta:**

  Accesible pulsando sobre una "MealCard". Esta pantalla está diseñada para presentar la información de forma clara y ordenada:

  - Una cabecera con la imagen, título y descripción de la receta.
  - Una sección de información rápida (tiempo de preparación, calorías, raciones, etiquetas de dietas/cocinas).
  - Secciones desplegables para "Ingredientes" (incluyendo imagen y cantidad necesaria), "Equipo Necesario", "Instrucciones de Preparación" e "Información Nutricional", evitando así la sobrecarga de información y permitiendo al usuario consultar solo lo que necesita.

- **Gestión del estado y almacenamiento local:**

  Para optimizar el rendimiento y permitir un uso sin conexión fluido, se implementó una arquitectura de estado avanzada:

  - **Contexto Global (MenuContext):** se utilizó un contexto de React para manejar el estado global del menú. Al iniciar la app, se sigue una estrategia de hidratación de estado: se carga el menú desde el almacenamiento local (AsyncStorage) para una visualización instantánea, mientras en segundo plano se sincroniza con el servidor para obtener la versión más reciente.
  - **Hook de Caching (useRecipe):** para la visualización de recetas, se creó un hook personalizado que gestiona un caché local. La primera vez

que se accede a la vista detallada de una receta, se solicita al backend y guarda en el caché. En accesos posteriores, los datos se cargan instantáneamente desde el caché, eliminando peticiones de red redundantes y mejorando drásticamente la fluidez de la navegación.

5.1.3. Tercera Iteración: Recetas de Usuario y Menú Avanzado.

El objetivo de esta iteración fue expandir significativamente las capacidades de la aplicación, proporcionando al usuario dos funcionalidades: permitirle crear y gestionar sus propias recetas e introduciendo una mayor flexibilidad en la personalización de su menú semanal. Esta fase transforma la aplicación de un simple generador de dietas a una plataforma más completa de gestión y organización de recetas y planes nutricionales.

Los requisitos funcionales implementados en esta fase fueron:

- RF6 – Añadir otro tipo de comidas extra al menú semanal
- RF8 – Gestión de recetas propias

5.1.3.1 Backend

El desarrollo en el servidor se centró en construir un módulo completo de gestión de recetas de usuario y en la implementación de una arquitectura escalable para la subida de imágenes.

- **Gestión de Recetas Propias (CRUD):**

Conjunto de endpoints para gestionar el ciclo de vida de las recetas de usuario:

  - **POST /my-recipes:** creación de una nueva receta con validaciones en el servidor.

  - **GET /my-recipes:** listado de todas las recetas creadas por el usuario.

- **PUT /my-recipes/{recipe_id}:** edición completa de una receta existente.

- **DELETE /my-recipes/{recipe_id}:** eliminación de una receta, incluyendo la lógica para desvincularla del menú semanal del usuario si estuviera asignada.

- **Arquitectura de Subida de Imágenes (Google Cloud Storage):**
  Para la gestión de imágenes de recetas, se implementó una arquitectura moderna y eficiente que evita que los archivos pasen por el servidor de la aplicación, mejorando el rendimiento y la escalabilidad. El flujo se basa en URLs formadas (Signed URLs):

  1. **Solicitud de URL (POST /generate-upload-url):** cuando el cliente necesita subir una imagen, primero solicita una URL de subida segura a este endpoint.

  2. **Generación de URL Firmada:** el backend se comunica con Google Cloud Storage (GCS) y genera una URL firmada de corta duración. Esta URL otorga un permiso temporal y específico para que el cliente puede escribir (PUT) un objeto en un bucket de GCS en una ruta única y segura (ej. user-{id}/recipes/{uuid}.ext).

  3. **Respuesta al Cliente:** el servidor devuelve al cliente la URL firmada para la subida (signed_url) y la URL pública final del archivo (public_url).

  4. **Subida y Almacenamiento:** el cliente sube la imagen directamente a GCS usando la signed_url. Posteriormente, al guardar la receta, envía la public_url al backend, que es la que se almacena en la base de datos.

Esta arquitectura desacopla el almacenamiento de archivos del servidor principal, una práctica recomendada en sistemas en la nube.

- **Soporte para Comidas Adicionales:**

  La lógica de sugerencias de recetas de la iteración anterior se amplió para aceptar nuevos tipos de comida como "Snack", "Bebida", etc., permitiendo al sistema filtrar y devolver recetas adecuadas para estas categorías.

5.1.3.2. Frontend

El desarrollo móvil se centró en una nueva sección completa para la gestión de recetas propias y en la mejora de la interfaz del menú semanal.

- **Módulo "Mis Recetas":**
  - **Navegación y Listado:** nueva pestaña en la barra de navegación principal. Esta pantalla lista las recetas del usuario e incluye componentes de búsqueda por título y filtros por tipo de comida.

  - **Formulario de Creación/Edición:** se desarrolló un formulario completo que permite al usuario introducir todos los detalles de una receta: título, descripción, raciones, tipo de comida, etc. Así como añadir dinámicamente la lista de ingredientes y pasos de preparación.

  - **Selección de Imagen:** se implementó un componente reutilizable (ImagePicker) que utiliza la librería expo-image-picker para permitir al usuario seleccionar una imagen desde la galería o tomar una nueva foto con la cámara.

- **Lógica de Eliminación Segura:** el sistema presenta un diálogo de confirmación (Alert). Si la receta está asignada en el menú, un segundo diálogo advierte que la acción también la eliminará de su planificación.

- **Integración de Comidas Adicionales en el Menú:**
  - En la vista de cada día del menú, debajo de las comidas principales se añadió una tarjeta para "Añadir comida o bebida extra".

  - Al pulsarla, se abre un primer modal que permite al usuario seleccionar la categoría de la comida extra (Snack, Postre, Bebida, Ensalada o Aperitivo).

  - Una vez seleccionada, se abre el modal de sugerencias, ya filtrado para dicha categoría. Se permite añadir más de una comida extra. Estas nuevas tarjeta de comida tienen la misma funcionalidad de reemplazo, visualización y eliminación que las principales.

- **Integración de Recetas Propias en el Menú:**

  Para cerrar el ciclo funcional, el modal de sugerencias (tanto para comidas principales como para extras) fue rediseñado para incluir dos pestañas:

  1. **Sugerencias:** muestra las recetas recomendadas por el sistema (de la base de datos o de Spoonacular).

  2. **Mis Recetas:** muestra un listado de las recetas propias del usuario, filtradas por el tipo de comida correspondiente, permitiéndole asignarlas directamente a su menú semanal.

5.1.4. Cuarta Iteración: Generación y Exportación de la Lista de la Compra

Esta última iteración de desarrollo se centró en completar el ciclo de vida de planificación de comidas, proporcionando al usuario herramientas útiles para seguir y gestionar su dieta fácilmente: la generación de una lista de la compra automática y la capacidad de exportarla y compartirla fácilmente. Los requisitos funcionales implementados en esta fase fueron:

- RF9 – Generación de la lista de la compra
- RF10 – Exportación y compartición de la lista de la compra

5.1.4.1. Backend

El desarrollo en el servidor se enfocó en implementar una lógica de negocio avanzada para generar una lista de la compra consolidada, precisa y organizada.

- **Algoritmo de Generación de la Lista de la Compra:**

  Se implementó un endpoint GET /shopping-list/me que orquesta un proceso de varios pasos para construir la lista:

  1. **Agregación y Escalado de Ingredientes:** el sistema recupera el menú semanal del usuario, itera sobre todas las recetas del plan y suma las cantidades de cada ingrediente. En este paso, aplica un factor de escalado basado en las raciones indicadas por el usuario frente a las raciones por defecto de cada receta. Este proceso consolida ingredientes idénticos presentes en diferentes recetas.

2. **Partición Inteligente de Ingredientes:** una vez agregados, los ingredientes se dividen en dos grupos:
    - Procesables por la API de Spoonacular (con ID y unidades reconocidas).
    - No procesables (ingredientes personalizados creados por el usuario con nombres o unidades no estandarizadas).

3. **Procesamiento y Clasificación:**
    - La lista de ingredientes procesables se envían a un endpoint de Spoonacular POST /mealplanner/shopping-list/compute que devuelve los ingredientes agrupados por secciones de supermercado (ej. Carne, Frutas y Verduras, etc.), además de realizar conversiones de unidades.

    - Los ingredientes no procesables por la API de Spoonacular se agrupan en una categoría genérica (Ingredientes Propios).

4. **Enriquecimiento y Consolidación Final:** el sistema enriquece la respuesta de la API con datos de la base de datos local (imágenes de ingredientes, nombres traducidos según el idioma del usuario, etc.). El resultado final es una estructura de datos unificada, organizada por secciones de supermercado y lista para ser enviada al cliente.

5.1.4.2. Frontend

El desarrollo en la aplicación móvil se centró en crear una experiencia de usuario completa y funcional, desde la generación de la lista hasta su uso práctico en el supermercado.

- **Nuevo Módulo "Compra":** se añadió una nueva pestaña en la barra de navegación principal. Al entrar por primera vez, la pantalla muestra un estado vacío con una llamada a la acción clara para generar la lista. Al pulsarla, se abre un modal solicitando el número de raciones deseadas para calcular las cantidades proporcionales.

- **Visualización Interactiva:** la lista se presenta en una vista organizada por secciones de supermercado. Cada sección es un componente desplegable que contiene los ingredientes correspondientes. Cada ítem de la lista incluye la imagen del ingrediente, su nombre y la cantidad necesaria, además de una casilla de verificación que permite al usuario tachar los productos que ya tiene o que va añadiendo a su cesta.

- **Actualización de la Lista:** la interfaz incluye un botón de refresco, que permite al usuario regenerar la lista tras realizar cambios en el menú semanal, volviendo a solicitar el número de raciones antes de recalcular.

- **Generación de PDF y Compartición Nativa:**

  Se implementó una funcionalidad avanzada para generar y compartir un PDF de la lista de la compra utilizando las librerías expo-print, expo-sharing y expo-file-system.

  - **Generación Dinámica de HTML:** al pulsar el botón de compartir, la aplicación genera dinámicamente un documento HTML en tiempo de ejecución que incluye:
    - Un encabezado profesional con el logo de FastDiet, el nombre del usuario, el rango de fechas del menú y la fecha de generación.
    - Una maquetación inteligente a dos columnas para distribuir las secciones y sus ingredientes optimizando el uso del espacio en una página A4 y mejorando la legibilidad.

- **Conversión y Compartición:**
  - expo-print se encarga de convertir el HTML en un archivo PDF.
  - expo-file-system gestiona el archivo resultante, permitiendo renombrarlo con un formato estandarizado (ej. Lista-Compra-FastDiet-24-08-2025.pdf).
  - Finalmente, expo-sharing invoca la interfaz de compartición nativa del sistema operativo (iOS/Android), permitiendo al usuario enviar el PDF a través de cualquier aplicación compatible (WhatsApp, Email, AirDrop, etc.).

5.2. Pruebas

Para garantizar la calidad y robustez del software desarrollado, se implementó una estrategia de pruebas multinivel para el backend, complementada con pruebas funcionales manuales para la aplicación móvil.

El objetivo fue verificar la lógica de negocio de forma aislada con pruebas unitarias, asegurar la correcta integración y comunicación entre los componentes del sistema a través de la API con pruebas de integración y, finalmente, validar la experiencia final del usuario.

5.2.1. Pruebas del Backend

Se utilizó el framework Pytest para la creación y ejecución de un conjunto de tests automatizados. Para lograr el aislamiento de componentes, se usó la librería unittest.mock, una herramienta clave para la creación de dobles de prueba.

Todas las pruebas se estructuraron siguiendo el patrón Arrange-Act-Assert (AAA), que aporta claridad y consistencia al proceso:

1. **Arrange (Organizar):** en esta fase se establecen todas las precondiciones necesarias para la prueba. Eso incluye la inicialización de objetos, la definición de datos de entrada y algunas veces la configuración de dobles de prueba (mocks) para simular y controla el comportamiento de dependencias externas (como la base de datos o APIs externas).

2. **Act (Actuar):** se ejecuta la unidad de código que se está probando.

3. **Assert (Verificar):** se comprueba que el resultado de la acción es el esperado. Las aserciones pueden validar un resultado obtenido, si una función fue llamada con los argumentos correctos o si se lanzó una excepción específica.

- **Pruebas Unitarias:** se centraron en verificar la lógica de negocio de las funciones de servicio con aislamiento de la base de datos y de las APIs externas, esto se logró con MagicMock.

  Se realizaron pruebas sobre las áreas más críticas del sistema:

  1. **Servicio de Gestión de Usuarios:** se validaron las funciones de autenticación y registro, cubriendo todos los escenarios posibles. Por ejemplo, para el caso de registro se probaron varios escenarios como el registro exitoso, registro erróneo con email ya existente y verificado, y registro con email ya existente no verificado.

     También, se probó que la política de seguridad de contraseñas (longitud, uso de mayúsculas, minúsculas, números y caracteres especiales) se aplica correctamente.

2. **Lógica de Generación de Menús:** se crearon tests específicos para la clase MealPlanGenerator con el fin de validar su complejo algoritmo. Los tests clave fueron:

    - **Escenario de Éxito desde Base de Datos:** se probó que, con suficientes recetas en la base de datos, el plan se genera con éxito sin realizar ninguna llamada a la API externa. Se validó que el mock del servicio Spoonacular search_recipes no fuera llamado.

    - **Escenario de Fallback a la API:** se probó que, si la base de datos no tiene suficientes resultados, el generador recurre a la API de Spoonacular para completar el menú.

    - **Escenario de Fracaso Controlado:** se validó que, si ni la base de datos ni la API externa pueden encontrar suficientes recetas para cumplir los mínimos viables, el sistema lanza una excepción controlada MealPlanGeneratorError con un código de error específico, evitando un fallo inesperado.

- **Pruebas de Integración:** se diseñaron para verificar el comportamiento de los endpoints de la API, asegurando que las distintas partes del sistema interactúan correctamente. Se utilizó el cliente de pruebas (TestClient) de FastAPI para simular peticiones HTTP y validar las respuestas completas (código de estado, cabeceras y cuerpo JSON).

  Se probó el correcto funcionamiento de endpoints públicos como /register, /login o los endpoints de envío de códigos de verificación, y endpoints protegidos que requieren autenticación como /meals_plans/generate o

/shopping-list/me. Por ejemplo, para el caso del endpoint de generación de la lista de la compra se han probado los siguientes escenarios:

- **Escenario de Éxito:** procesa menús con ingredientes reconocidos y no reconocidos por Spoonacular, enviando a la API externa los primeros y gestionando localmente el resto. Se verificó que la respuesta final fusionara correctamente ambos, con traducciones y datos enriquecidos.

- **Caso de Ausencia de Datos:** se probó que se responde con un error 404 controlado cuando un usuario sin menú intenta generar una lista.

- **Prueba de Resiliencia ante Fallo Externo:** se simuló un fallo en la API de Spoonacular. Se verificó que el endpoint gestiona la excepción de forma robusta, no interrumpe su ejecución y devuelve a usuario una lista de la compra parcial.

5.2.2. Pruebas de la Aplicación Móvil (Frontend)

Las pruebas del cliente se realizaron mediante pruebas funcionales manuales, ejecutando la aplicación en emuladores de iOS y Android, así como en dispositivos físicos. El objetivo era garantizar la funcionalidad, usabilidad y consistencia visual.

- **Pruebas Funcionales Manuales:** se ejecutaron los flujos de usuario definidos en los casos de uso (sección 3.4), desde el registro y la generación de menús hasta la consulta de recetas y la exportación de la lista de la compra, asegurando que cada funcionalidad operaba según lo especificado.

- **Pruebas de Usabilidad e Interfaz (UI/UX):** se realizaron revisiones de la experiencia de usuario, prestando especial atención a la

claridad de la navegación, la retroalimentación del sistema (indicadores de carga, mensajes de error, estados vacíos) y la consistencia del diseño en todas las pantallas.

- **Pruebas de Compatibilidad:** se aseguró el correcto funcionamiento y renderizado visual de la aplicación en distintas versiones de los sistemas operativos iOS y Android para asegurar una experiencia del usuario consistente entre plataformas.

5.3. Despliegue

El despliegue del sistema se ha diseñado considerando una arquitectura moderna, segura y escalable, contemplando tanto el backend desarrollado con FastAPI como la aplicación móvil con React Native. La arquitectura propuesta está completamente preparada para una puesta en producción eficiente, siguiendo el procedimiento que se indica a continuación:

5.3.1. Backend

Para el despliegue de la API, se ha optado por una arquitectura serverless sobre Google Cloud Platform, lo que elimina la necesidad de gestionar servidores manualmente y permite un escalado dinámico basado en la demanda.

- **Containerización con Docker:** se define un archivo Dockerfile que crea una imagen ligera y portable conteniendo la aplicación y todas sus dependencias.

- **Google Cloud Run:** aquí se desplegaría la imagen de Docker. Se eligió este servicio por sus ventajas clave:
  - **Escalado automático:** escala automáticamente el número de contenedores en función del tráfico, pudiendo incluso escalar a cero si no hay peticiones, lo que optimiza enormemente los costes.

    - **Alta disponibilidad:** al ser un servicio gestionado por Google ofrece alta disponibilidad y tolerancia a fallos de forma nativa.

    - **Seguridad Integrada:** garantiza que todas las comunicaciones con la API se realicen a través de HTTPS.

- **Google Cloud SQL:** a través de este servicio se gestionaría la base de datos MySQL, simplificando la configuración y mantenimiento, con parches de seguridad y copias de seguridad automáticas.

- **Google Cloud Storage:** ya se usa en el desarrollo para almacenar imágenes de las recetas creadas por usuarios, aportando escalabilidad y durabilidad.

- **Gestión de Secretos:** en el desarrollo local se han guardado las variables de entorno en un archivo .env. En el despliegue en producción, la información sensible (como credenciales de la base de datos o claves de APIs externas) se gestionaría mediante Google Secret Manager o mediante variables de entorno configuradas directamente en Cloud Run, evitando que se queden expuestas.

5.3.2. Frontend

Para la aplicación móvil, el despliegue se debe realizar a través de las plataformas de distribución oficiales, asegurando un alcance global y un proceso de actualización estandarizado para los usuarios.

- **Compilación con Expo Application Services (EAS):** se debe utilizar el servicio EAS build para compilar el código de React Native en los binarios requeridos por cada plataforma (iOS y Android).

- **Publicación en las Tiendas Oficiales:** una vez compiladas, las aplicaciones se deben subir y publicar en Google Play Store (Android) y Apple App Store (iOS).

6

Conclusiones y Trabajos Futuros

En esta sección se presenta una reflexión final sobre el desarrollo del proyecto, evaluando el grado de cumplimiento de los objetivos establecidos, las principales dificultades encontradas durante el proceso y las posibles líneas de ampliación y mejora que darían continuidad al trabajo realizado.

6.1. Objetivos Cumplidos y Conclusiones Finales

El objetivo principal de este Trabajo de Fin de Grado era el diseño y la implementación de una herramienta integral y funcional para la planificación de dietas personalizadas, que abarcara desde la generación de un menú semanal hasta la creación automática de la lista de la compra. Tras la finalización del proyecto, se puede afirmar que dicho objetivo se ha cumplido con éxito. El resultado es FastDiet, una aplicación móvil multiplataforma completamente operativa que da

respuesta a todos los requisitos funcionales y no funcionales definidos en las fases iniciales de análisis.

FastDiet ofrece un conjunto completo de funcionalidades que permiten al usuario cubrir todo el ciclo de uso de la herramienta:

- **Autenticación y Gestión de Usuarios:** un sistema de acceso seguro que incluye registro con verificación por correo, inicio de sesión tradicional o con Google, recuperación de contraseña, cierre de sesión y eliminación de la cuenta y todos los datos asociados del usuario.

- **Gestión de Perfil y Preferencias:** introducción y edición de información personal (edad, peso, altura, sexo), nivel de actividad física, objetivos, tipo de dieta, cocinas preferidas e intolerancias alimentarias.

- **Cálculo Calórico Automatizado:** se calcula de forma automática las calorías diarias recomendadas basándose en los datos del perfil, asegurando que el plan nutricional esté alineado con los objetivos del usuario.

- **Generación de Menús Semanales Personalizados:** elaboración de planes adaptados a las preferencias y restricciones del usuario. El sistema ofrece una alta flexibilidad, permitiendo añadir o eliminar comidas, y sustituir platos por sugerencias proporcionadas por FastDiet o incluso incorporar recetas creadas por el usuario.

- **Visualización Completa de Recetas:** cada comida del menú se presenta con información detallada, incluyendo información nutricional, pasos de preparación, ingredientes y utensilios necesarios.

- **Creación Inteligente de la Lista de la Compra:** generación de una lista consolidada y optimizada a partir del menú semanal, organizando todos los ingredientes y productos necesarios por secciones de supermercado.

- **Exportación y Portabilidad de la Lista de la Compra:** la lista de la compra puede ser exportada a un archivo PDF, que permite al usuario descargarla y compartirla fácilmente a través de otras aplicaciones de mensajería, correo electrónico o almacenamiento en la nube.

Desde un punto de vista académico y personal, este proyecto ha representado una valiosa oportunidad para poner en práctica y reforzar los conocimientos obtenidos a lo largo del grado, al mismo tiempo que implicó aprender de forma autodidacta nuevas tecnologías y herramientas para entregar un producto final de alta calidad.

6.2. Dificultades encontradas

El desarrollo de la aplicación presentó retos técnicos y de gestión que fueron superados y que constituyeron una parte fundamental del proceso de aprendizaje. De entre todos los retos que hubo que superar se resaltan los siguientes:

1. **Complejidad del algoritmo de Generación del Menú:** fue necesario diseñar una lógica avanzada que consultara primero la base de datos, y en caso de no disponer de suficientes resultados, recurriera a la API de Spoonacular con una estrategia de fallback y múltiples intentos. Además, una vez obtenidas las recetas, debían ser distribuidas de manera coherente a lo largo de la semana, minimizando repeticiones y maximizando la variedad. Esta tarea requirió un diseño cuidadoso y un proceso de depuración intensivo.

2. **Creación de la Lista de la Compra:** se identificó que no todos los ingredientes eran procesables por la API de Spoonacular. Para solventar este problema, se desarrolló una lógica que diferenciara entre ingredientes reconocibles por Spoonacular y aquellos introducidos por los usuarios, que podían tener nombres o unidades no reconocibles. Posteriormente, ambos conjuntos debían ser procesados, unificados y presentados en una única lista coherente organizada en secciones de supermercado.

3. **Gestión y almacenamiento de imágenes:** la subida de imágenes de recetas creadas por usuarios supuso una dificultad notable debido a la falta de experiencia en tratamiento de archivos. El desafío consistió en diseñar un flujo seguro y escalable para recibir, validar y almacenar esos archivos. Tras investigar diversas alternativas, se optó por la integración con Google Cloud Storage, una solución que permitió asegurar la persistencia, disponibilidad y durabilidad de las imágenes de forma profesional.

En conjunto, estas dificultades supusieron un desafío técnico significativo, pero su resolución ha sido clave para el éxito del proyecto y el proceso de aprendizaje.

6.3 Posibles Ampliaciones y Trabajos Futuros

El estado actual de la aplicación constituye una base sólida sobre la cual se pueden construir numerosas mejoras y nuevas funcionalidades. Entre las líneas de desarrollo más relevantes que se plantean destacan las siguientes:

- **Asistente inteligente experto en nutrición:** integrar un chatbot que pueda resolver dudas de los usuario sobre nutrición, ofrecer información detallada de los beneficios de ciertos alimentos o recetas, y proporcionar una experiencia de usuario más interactiva y conversacional.

- **Optimización de la lista de la compra:** integrar el sistema con APIs de supermercados para estimar el coste real de los productos, sugerir alternativas más económicas o en oferta, y permitir incluso la compra directa desde la aplicación.

- **Finalización del despliegue y publicación:** completar el proceso de despliegue de la aplicación y publicarla en Google Play Store y Apple App Store, de modo que la aplicación sea accesible por cualquier persona.

En definitiva, este Trabajo de Fin de Grado ha logrado transformar un concepto inicial en una aplicación de software completa, funcional y robusta, abordando desafíos técnicos complejos y aplicando metodologías de ingeniería de software a lo largo de todo su ciclo de vida. El proyecto no solo alcanza con las metas propuestas, sino que también establece un punto de partida sólido para posibles evoluciones y mejoras en el futuro.

Bibliografía

- FastAPI (2025). FastAPI Documentation. https://fastapi.tiangolo.com/
- Meta (2025). React Native Documentation. https://reactnative.dev/
- Expo (2025). Expo Documentation. https://docs.expo.dev/
- Microsoft (2025). TypeScript Documentation. https://www.typescriptlang.org/
- Oracle (2025). MySQL Documentation. https://dev.mysql.com/doc/
- GitHub (2025). GitHub Documentation. https://docs.github.com/
- Spoonacular (2025). Spoonacular Food and Recipe API. https://spoonacular.com/food-api
- Google (2025). Cloud Translation Documentation. https://cloud.google.com/storage/docs
- Google (2025). Cloud Storage Documentation. https://cloud.google.com/storage/docs
- SQLAlchemy (2025). SQLAlchemy Documentation. https://docs.sqlalchemy.org/
- Alembic (2025). Alembic Documentation. https://alembic.sqlalchemy.org/
- Pydantic (2025). Pydantic Documentation. https://docs.pydantic.dev/
- Passlib (2025). Passlib Documentation. https://passlib.readthedocs.io/
- Python (2025). Logging – Python Documentation. https://docs.python.org/3/library/logging.html
- APScheduler (2025). APScheduler Documentation. https://apscheduler.readthedocs.io/
- Axios (2025). Axios Documentation. https://axios-http.com/

- Expo (2025). Expo Secure Store. https://docs.expo.dev/versions/latest/sdk/securestore/
- Expo (2025). Expo Print. https://docs.expo.dev/versions/latest/sdk/print/
- Expo (2025). Expo Sharing. https://docs.expo.dev/versions/latest/sdk/sharing/
- Expo (2025). Expo ImagePicker. https://docs.expo.dev/versions/latest/sdk/imagepicker/
- Expo (2025). Expo Localization. https://docs.expo.dev/versions/latest/sdk/localization/
- i18next (2025). i18next Documentation. https://www.i18next.com/
- LottieFiles (2025). Lottie React Native. https://lottiefiles.com/
- Nutrium. (2023). Fórmula de Mifflin-St Jeor para profesionales de nutrición https://nutrium.com/blog/mifflin-st-jeor-for-nutrition-professionals/
- Pytest (2025). Pytest Documentation. https://docs.pytest.org/en/stable/
- React Nativ. Logo React Native. https://devtop.io/wp-content/uploads/2022/10/react-native-1.png
- Expo. Logo Expo. https://images.seeklogo.com/logo-png/45/1/expo-go-app-logo-png_seeklogo-457073.png
- Python. Logo Python. https://images.seeklogo.com/logo-png/33/1/python-logo-png_seeklogo-332789.png
- Oracle. Logo MySQL: https://www.logo.wine/a/logo/MySQL/MySQL-Logo.wine.svg
- Microsoft. Logo Visual Studio Code. https://images-eds-ssl.xboxlive.com/image?url=4rt9.lXDC4H_93laV1_eHHFT949fUipzkiFOBH3fAiZZUCdYojwUyX2aTonS1aIwMrx6NUIsHfUHSLzjGJFxxj7kCzMIlSC20SNjaJf9GmG15ocnF.zbBRgxMSlB7Ejh6FbgNzxLvZOoW7N3ML56fn3m5Z4MO.M8pYrCFVKIhqM-&format=source

- Git. Logo Git. https://upload.wikimedia.org/wikipedia/commons/thumb/e/e0/Git-logo.svg/2560px-Git-logo.svg.png
- Microsoft. Logo TypeScript: https://images.icon-icons.com/2415/PNG/512/typescript_original_logo_icon_146317.png

Índice de Figuras

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA