Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Desarrollo de una aplicación web full-stack para la planificación de menús

Development of a full-stack web application for menu planning

Iván García González

La Laguna, 17 de junio de 2024

D. **Eduardo Manuel Segredo González**, profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor.

Dª. **Gara Miranda Valladares**, profesora Titular de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutora.

C E R T I F I C A (N)

Que la presente memoria titulada:

"*Desarrollo de una aplicación web full-stack para la planificación de menús*"

ha sido realizada bajo su dirección por D. **Iván García González**.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 17 de junio de 2024

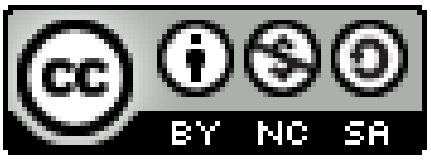
Agradecimientos

Me gustaría agradecer a mi familia y amigos, especialmente a mis padres y a mi hermana Ruth, por todo el apoyo y ayuda recibida, no solo durante esta última etapa, sino desde el principio. Su constante ánimo y confianza en mí han sido fundamentales para alcanzar todos mis logros.

Agradezco también a todas las personas, compañeros, profesores y amigos con los que he compartido momentos a lo largo de esta etapa de mi vida, momentos que recordaré para siempre.

Finalmente, quiero expresar mi gratitud a mi tutor, Eduardo Manuel Segredo González, y a mi cotutora, Gara Miranda Valladares, por la oportunidad de realizar este proyecto. Agradezco su constante apoyo, comprensión, respaldo y guía durante todo el proceso.

Licencia



Resumen

La alimentación saludable y variada es fundamental para el bienestar de las personas. En muchas ocasiones, la falta de tiempo y una planificación inadecuada lleva a la elección de comidas rápidas y poco saludables, especialmente entre aquellos con horarios ocupados. Este Trabajo de Fin de Grado presenta el desarrollo de una aplicación web full stack diseñada para facilitar la planificación de menús, ayudando a los usuarios a mantener una dieta equilibrada y nutritiva de manera sencilla y eficiente.

La aplicación web desarrollada permite gestionar una base de datos de ingredientes y recetas, ofreciendo sugerencias de menús basadas en diversas restricciones y preferencias del usuario. Cada menú incluye información nutricional detallada y estimaciones de costos, lo que permite a los usuarios tomar decisiones informadas sobre su alimentación. El sistema desarrollado incluye una API REST para la gestión de operaciones CRUD (crear, leer, actualizar, eliminar) sobre usuarios, ingredientes, recetas y menús, y se integra con el frontend para ofrecer una experiencia de usuario intuitiva. Permite al usuario no solo la visualización de ingredientes y recetas con diferentes métodos de filtrado y búsqueda, sino que también ofrece la creación y modificación de ingredientes y recetas, lo que enriquece la base de datos de la aplicación. Además, permite la gestión de los propios usuarios y los menús generados y guardados. Las tecnologías utilizadas en el desarrollo incluyen principalmente herramientas y frameworks basados en NodeJS, asegurando un rendimiento óptimo y una interacción fluida.

Como resultado de este Trabajo de Fin de Grado, se ha desarrollado una aplicación web totalmente funcional para la gestión de alimentos, recetas y menús. Entre sus puntos fuertes se destacan la facilidad de uso, la intuitividad y una interfaz agradable para el usuario. Además, no solo ofrece información detallada sobre las recetas, sino que también proporciona datos detallados sobre los ingredientes individuales.

Asimismo, se identifican varias líneas potenciales de trabajo futuro, como la integración de APIs externas para la consulta de precios en tiempo real y la incorporación de un resolutor para la optimización del problema de planificación de menús.

Palabras clave: planificación de menús, nutrición, alimentos y recetas, PLAMESA, aplicación web.

Abstract

A healthy and varied diet is essential for people's well-being. Often, the lack of time and inadequate planning leads to the choice of quick and unhealthy meals, especially among those with busy schedules. This final degree project presents the development of a full stack web application designed to facilitate menu planning, helping users maintain a balanced and nutritious diet in a simple and efficient way.

The developed web application allows managing a database of ingredients and recipes, offering menu suggestions based on various user restrictions and preferences. Each menu includes detailed nutritional information and cost estimates, enabling users to make informed decisions about their diet. The developed system includes a REST API for managing CRUD (create, read, update, delete) operations on users, ingredients, recipes, and menus, and it integrates with the frontend to offer an intuitive user experience. It allows users not only to view ingredients and recipes with different filtering and search methods but also to create and modify ingredients and recipes, enriching the application's database. Additionally, it enables the management of users and the generated and saved menus. The technologies used in the development mainly include NodeJS-based tools and frameworks, ensuring optimal performance and smooth interaction.

As a result of this final degree project, a fully functional web application has been developed for the management of food, recipes, and menus. Among its strengths are ease of use, intuitiveness, and a user-friendly interface. Moreover, it not only offers detailed information about recipes but also provides exhaustive data about individual ingredients.

Several potential future work lines have also been identified, such as the integration of external APIs for real-time price consultation and the incorporation of a solver for the optimization of the menu planning problem.

Keywords: menu planning, nutrition, food and recipes, PLAMESA, web application.

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

La malnutrición es una preocupación generalizada en la sociedad actual. Lograr mantener una alimentación variada y nutritiva de manera constante a lo largo del tiempo puede ser todo un desafío; después de algunos días, es común caer en patrones repetitivos y enfrentar dificultades para mantener una alimentación saludable y equilibrada. Es por esta razón que la elaboración de una planificación alimenticia se convierte en fundamental.

En la etapa infantil la nutrición desempeña un papel crucial y, en la mayoría de los casos, resulta difícil tener un control total sobre los alimentos que consumen los niños. Durante la infancia, la diversidad en la dieta es esencial, y dado que una parte significativa de las comidas se lleva a cabo en los comedores escolares, la planificación de menús se convierte en un aspecto fundamental. No obstante, en la actualidad, el diseño y la revisión de estos menús se realizan de manera manual, lo que puede resultar en un proceso de trabajo tedioso, y que hoy en día deja un margen de mejora considerable.

Este déficit en la planificación se relaciona en los adultos con la frecuente situación de consumir lo primero que está disponible, resultado de la falta de tiempo y planificación. Este hábito podría transformarse mediante la organización de nuestras comidas para un periodo determinado, reduciendo así los momentos de indecisión que nos llevan a optar por alternativas rápidas, generalmente poco saludables y repetitivas. Esta dinámica es especialmente común entre estudiantes universitarios, quienes, debido a la escasez de tiempo, creatividad o motivación, tienden a consumir los mismos alimentos durante varios días consecutivos. La implementación de un menú planificado podría romper con este ciclo repetitivo en la elección de alimentos.

1.1. Antecedentes

En vista de esto, se plantea el desarrollo de una aplicación inspirada en el enfoque de SCHOOLTHY [53]. Esta aplicación tiene como objetivo ofrecer planes de alimentación saludables y equilibrados, que no solo cumplan con los aportes energéticos y nutricionales recomendados, sino que también tengan un costo mínimo y una amplia variedad de platos y grupos de alimentos. SCHOOLTHY brinda una perspectiva ventajosa en la creación de planes alimenticios adaptados a diversos requisitos nutricionales y dietéticos, especialmente en entornos escolares. El principal problema de esta herramienta, no obstante, es que se trata de una herramienta de escritorio, lo que dificulta su difusión y uso por parte de potenciales usuarios interesados en su funcionalidad.

SCHOOLTHY implementa la Computación Evolutiva (EC) para abordar un novedoso sistema multiobjetivo, el Problema de Planificación de Menú (MPP), denominado Problema de Planificación de Menú Multiobjetivo (MMPP). Esta formulación se distingue de otras variantes del MPP previamente propuestas, ya que se enfoca exclusivamente en la creación de planes de alimentación para comedores escolares, destinados a un grupo específico de niños en edad escolar durante el almuerzo.

En términos de objetivos de optimización, SCHOOLTHY busca simultáneamente maximizar la variedad de platos y grupos de alimentos incluidos en el plan de alimentación, mientras minimiza los costos asociados.

En este contexto, los aspectos clave considerados para el diseño de los menús de SCHOOLTHY se basan en lo siguiente: los planes están destinados únicamente al almuerzo, compuestos por un entrante, plato principal y postre; los grupos de edad abarcados son de 4 a 13 años, sin distinción de género. Dado el tamaño de estos grupos infantiles, no se toman en cuenta las preferencias individuales de los usuarios por alimentos específicos. Es relevante señalar, no obstante, que se podrían considerar posibles alérgenos, enfermedades o incompatibilidades alimentarias que pudieran restringir el consumo de ciertos alimentos.

1.2. Objetivos del proyecto

En este trabajo, se recoge el desarrollo de una aplicación web full stack destinada a la planificación de menús. La aplicación utilizará una base de datos que incluye diversos ingredientes y platos, con el propósito de ofrecer a los usuarios planes alimenticios variados. Cada plan contendrá información detallada sobre los valores nutricionales de cada plato y del conjunto del menú, además de mostrar el costo asociado a los menús sugeridos.

Los usuarios tendrán la capacidad de seleccionar fechas, períodos y aplicar filtros según ingredientes, considerando alergias o la reducción de ciertos elementos. Asimismo, podrán introducir y guardar nuevos ingredientes y platos en el planificador, revisar planes previos, y descargar listas de compras que incluyan los ingredientes necesarios para la preparación de los menús. Adicionalmente tendrá otras opciones como la recomendación de platos a partir de una selección específica de ingredientes.

Se dividen a continuación los objetivos principales a conseguir en el proyecto:

- La implementación de una API REST que posibilite la creación, consulta y gestión de datos relacionados con ingredientes, platos y menús. Esta API estará abierta para su acceso por parte de cualquier sistema o usuario, permitiendo su integración en otros proyectos.
- El diseño y desarrollo de un *frontend* que haga uso eficiente de la API mencionada anteriormente, cumpliendo con los requisitos detallados en la Sección 1.2.1.
- Ofrecer a los usuarios la capacidad de personalizar de manera sencilla las características de los menús, adaptando los planes alimenticios a sus preferencias individuales.

- Generar propuestas de planes alimenticios variados y nutritivos que se ajusten a las preferencias de los usuarios.

1.2.1. Objetivos específicos

Para lograr el desarrollo de cualquier producto de software, es crucial llevar a cabo un estudio y evaluación de los requisitos esenciales que se pretenden alcanzar. De acuerdo con el objetivo final del proyecto, los requisitos que el sistema debe cumplir pueden resumirse en los siguientes puntos:

- Generar recomendaciones de menús
- Permitir la incorporación de restricciones alimenticias al recomendador
- Tener en cuenta el perfil del usuario al que está destinado el plan
- Utilizar algoritmos de optimización, teniendo en cuenta restricciones y perfil de usuario, y priorizando valores como variedad y costo
- Permitir la incorporación de nuevos platos en la aplicación
- Permitir la modificación de la cantidad de comensales en los platos
- Permitir la modificación de los platos dentro del plan propuesto
- Generar una lista de la compra a partir del menú propuesto
- Mostrar información nutricional sobre los platos y el menú
- Mostrar información sobre el costo de los platos y del menú

1.3. Planificación

El desarrollo del proyecto, se ha desglosado en las siguientes actividades:

1. Estudio del estado del arte sobre aplicaciones relacionadas con la planificación y recomendación de menús y planes alimenticios.
2. Análisis y especificación de requisitos del sistema.
3. Diseño del sistema de información, consistente en una API REST para el *backend*, además de una aplicación web como *frontend*.
4. Implementación del sistema de información: *backend* y *frontend*.
5. Pruebas y despliegue del sistema de información mediante una metodología Test-Driven Development (TDD).
6. Difusión de los resultados: escritura de la memoria y preparación de la defensa.

Para cada tarea se ha definido un período estimado de tiempo de trabajo, que se muestra en la Figura 1.1.

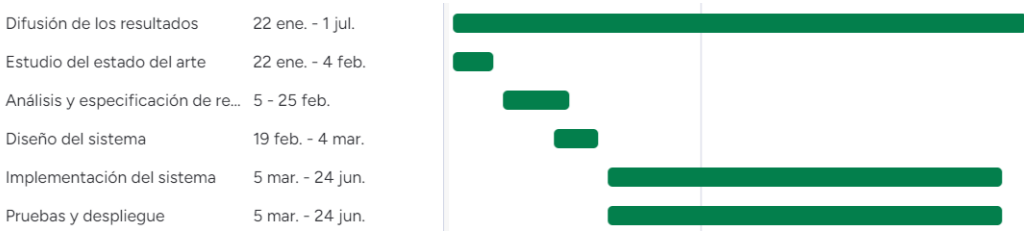


Figura 1.1: Planificación del Proyecto

A partir del trabajo realizado, se ha estructurado la memoria del proyecto de la siguiente manera:

- **Capítulo 2:** Presenta un estudio del estado del arte sobre las aplicaciones relacionadas con la planificación de menús, tanto a nivel individual como en grupos, llevando a cabo un análisis de los requisitos a implementar en la aplicación.
- **Capítulo 3:** Expone el modelo de datos diseñado, junto a las entidades definidas para este proyecto, la base de datos utilizada para almacenar los datos de la aplicación y la carga de datos realizada a la base de datos.
- **Capítulo 4:** Muestra el desarrollo tanto del *backend* como del *frontend* de la aplicación web desarrollada, PLAMESA, junto con el despliegue de las mismas y las pruebas definidas para el *backend*.
- **Capítulo 5:** Presenta un estudio de viabilidad para una posible línea futura de trabajo de este TFG para la optimización del problema de planificación de menús en un resolutor externo. En concreto, se hace el análisis para la herramienta Othimi.
- **Capítulo 6:** Detalla las conclusiones tras el desarrollo del proyecto, así como las líneas futuras del proyecto.
- **Capítulo 7:** Expone el presupuesto del proyecto.

Capítulo 2

Alimentación y nutrición

La alimentación es la ingesta de alimentos por parte de los organismos para satisfacer las necesidades nutricionales, por lo que desempeña un papel vital en el suministro de energía y el fomento del desarrollo de los organismos. Implica la elección cuidadosa de alimentos y bebidas que contribuyan al bienestar general. Un plan de alimentación saludable busca proporcionar al cuerpo la energía y los nutrientes necesarios, incluyendo proteínas, carbohidratos, grasas, vitaminas, minerales y agua, de manera equilibrada.

Por otro lado, la nutrición constituye el proceso biológico esencial en el que los organismos asimilan los alimentos sólidos y líquidos necesarios para su funcionamiento, crecimiento y mantenimiento de funciones vitales. Una buena nutrición implica adoptar hábitos alimenticios que promuevan la diversidad en la dieta, limitando el consumo de ciertos productos y regulando conscientemente las cantidades y calorías ingeridas. En el ser humano, la variabilidad en la alimentación se debe a factores como preferencias personales, edad, nivel de actividad física, recursos económicos y la disponibilidad de productos en la región de residencia. Al planificar un régimen nutricional, es esencial tener en cuenta estos factores para garantizar su eficacia y adaptación a las particularidades de cada individuo.

Es fundamental comprender la información nutricional de los productos para llevar a cabo una dieta variada, reduciendo el consumo de alimentos perjudiciales y favoreciendo aquellos que aportan beneficios. Por tanto, a continuación, analizaremos qué son los nutrientes y su importancia.

2.1. Nutrientes

Los nutrientes son elementos esenciales para el metabolismo, participando activamente en las reacciones que mantienen las funciones del organismo. Las tablas de composición de alimentos son herramientas útiles para comparar y seleccionar alimentos basándose en su contenido nutricional.

Existen dos categorías principales de nutrientes: los macronutrientes, como proteínas, carbohidratos y lípidos, y los micronutrientes, que incluyen vitaminas y minerales, presentes en cantidades más pequeñas. Aunque el agua y la fibra no son nutrientes, son esenciales en cantidades adecuadas.

Los nutrientes también se pueden clasificar según su función principal. Aquellos con función energética, como los hidratos de carbono y grasas, proporcionan energía. Los

nutrientes con función plástica, como proteínas y minerales, contribuyen a la formación y mantenimiento de estructuras corporales. Por último, los nutrientes con función reguladora, como vitaminas y minerales, participan en la regulación de las reacciones bioquímicas del organismo [30].

2.1.1. Macronutrientes

Los macronutrientes son componentes esenciales de nuestra dieta que proporcionan la energía necesaria para el funcionamiento diario del organismo. Se dividen en tres categorías principales: proteínas, carbohidratos y grasas [43].

- **Proteínas:** Tienen principalmente una función estructural, ya que componen los huesos, músculos, piel, órganos, entre otros. Son el material de construcción que forma el cuerpo y se encuentran en alimentos como carne, pescado, huevos, legumbres y productos lácteos. Se recomienda que en un adulto las proteínas representen entre el 10 % y el 35 % de la ingesta calórica diaria.
- **Carbohidratos:** Su función más importante es proporcionar energía al cuerpo. Se encuentran en alimentos como cereales, arroz, pasta, pan, frutas y verduras. Se recomienda que en un adulto los carbohidratos representen entre el 45 % y el 65 % de la ingesta calórica diaria.
- **Grasas:** Son importantes para la absorción de vitaminas, la salud cerebral y la regulación hormonal. Se encuentran en alimentos como aceite de oliva, aguacates, frutos secos y pescados grasos. Se recomienda que en un adulto los lípidos representen entre el 20 % y el 35 % de la ingesta calórica diaria.

2.1.2. Micronutrientes

Los micronutrientes, generalmente derivados de la ingesta de alimentos, son pequeñas cantidades de vitaminas y minerales requeridos por el cuerpo para la mayoría de las funciones celulares [12].

- **Vitaminas:** Las vitaminas son compuestos orgánicos esenciales para el funcionamiento normal del cuerpo, participando en procesos como el crecimiento, la digestión y la función nerviosa. Se encuentran en una amplia variedad de alimentos, como frutas, verduras, lácteos y alimentos de origen animal. Existen dos tipos principales de vitaminas: las liposolubles, presentes en alimentos grasos, como la vitamina A, D, E y K; y las hidrosolubles, como las del grupo B y la vitamina C, solubles en agua.
- **Minerales:** Los minerales son elementos inorgánicos fundamentales para diversas funciones fisiológicas en el cuerpo, como calcio, hierro, zinc y magnesio, entre otros. Se encuentran en alimentos como productos lácteos, carnes magras, frutas y verduras. Cada mineral desempeña un papel específico y esencial en el mantenimiento de la salud. Algunos, como el calcio, forman parte de las estructuras sólidas del cuerpo, como huesos, cartílagos y dientes, mientras que otros están disueltos en el organismo.

2.2. Base de datos nutricionales

Las tablas y bases de datos de composición de alimentos desempeñan un papel crucial en la evaluación del estado nutricional de una población. Estos recursos son fundamentales no solo para este propósito, sino también para el diseño de políticas nutricionales, la investigación en nutrición, la planificación de dietas y la provisión de información relevante a los consumidores.

La Organización para la Agricultura y la Alimentación (FAO) [41] coordina la Red Internacional de Sistemas de Datos Alimentarios (INFOODS) [34], que cuenta con un directorio de Bases de Datos de Composición de Alimentos (BDCA) a nivel nacional, regional e internacional.

A continuación, se proporciona un análisis de algunas de las tablas y bases de datos de composición de alimentos junto a sus características principales, que servirán para el desarrollo del modelo de datos.

- **BEDCA [31]:** La Base de Datos Española de Composición de Alimentos, publicada por la Red BEDCA, es una iniciativa colaborativa que reúne a centros de investigación públicos, instituciones gubernamentales y entidades privadas. Su principal objetivo es desarrollar y mantener una completa base de datos sobre la composición de alimentos en España. Esta base de datos se construye siguiendo los rigurosos estándares europeos establecidos por la Red de Excelencia Europea EuroFIR.

  La plataforma BEDCA ofrece una interfaz web dedicada (véase la Figura 2.1) que permite a los usuarios acceder a la información sobre la composición de alimentos almacenada en la base de datos. Sin embargo, es importante destacar que actualmente no existe una función nativa para exportar datos o acceder a ellos fuera de esta interfaz web.

Figura 2.1: Plataforma Web de BEDCA

- **EuroFIR [37]:** La Asociación EuroFIR AISBL tiene como objetivos el desarrollo, gestión, publicación y explotación de los datos de composición de alimentos, así como la promoción de la cooperación internacional y la armonización. La base de datos se actualiza regularmente con nuevas versiones de las bases de datos nacionales, lo que la mantiene bastante actualizada. Sin embargo, para poder acceder a ella es necesario registrarse y pagar una tarifa. En la Figura 2.2 podemos ver la plataforma web.

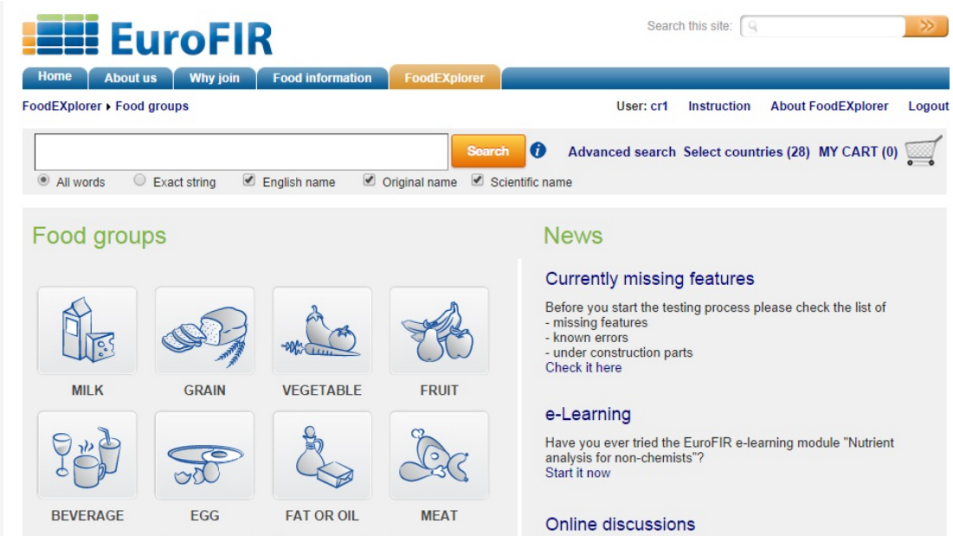


Figura 2.2: Plataforma Web de EuroFIR

- **Open Food Facts [39]:** La base de datos Open Food Facts (OPF) (Figura 2.3), es una plataforma gratuita y sin ánimo de lucro con colaboración abierta que recopila información de productos alimenticios a nivel mundial, abarcando no solo alimentos básicos, sino también una amplia variedad de productos. Utiliza la colaboración voluntaria de los usuarios y otras aplicaciones de recopilación de alimentos para obtener datos adicionales. Además de la interfaz web, ofrece una aplicación móvil que permite escanear productos para acceder a su información nutricional y también posibilita la descarga de datos almacenados en diversos formatos.

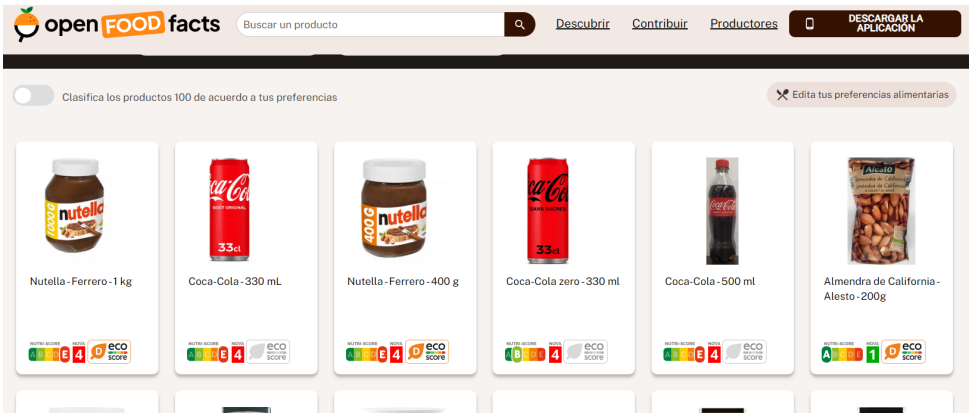


Figura 2.3: Buscador de la Base de Datos Open Food Facts

2.3. Aplicaciones relacionadas

Previo al desarrollo del proyecto, se ha llevado a cabo un estudio sobre algunas de las aplicaciones existentes, las cuales están relacionadas con los objetivos planteados para el sistema de información a desarrollar. Además, se realizó un análisis detallado de las diversas características que se espera que cumpla dicho sistema.

2.3.1. Estudios y aplicaciones de escritorio centradas en la recomendación de menús

La desnutrición y la necesidad de una alimentación de calidad son temas relevantes y actuales que afectan a todas las edades, desde la población infantil hasta la adulta y anciana. Numerosos estudios abordan este tema y proponen sistemas para la recomendación de menús adaptados a estos grupos de personas. A continuación, se exponen algunos de estos estudios y sistemas.

- **SCHOOLTHY** [53]: Aplicación de escritorio, diseñada como planificador automático de menús para comidas escolares saludables y equilibradas, sirve de apoyo en la toma de decisiones para la planificación de menús escolares durante la hora del almuerzo. Genera propuestas alimenticias que buscan minimizar tanto el costo del plan como el grado de repetición dentro del mismo.

  A partir de las especificaciones iniciales introducidas por el usuario, la aplicación genera varios planes alimenticios, proporcionando información nutricional y de costos para cada uno de ellos.

- **AMPAC** [47]: *Automated Menu Planning Algorithm for Children*, estudio que se centra en la automatización de un sistema de recomendación alimenticia utilizando el algoritmo ID3 para la población infantil en India.

  El algoritmo ID3 se implementa para tomar decisiones apropiadas entre los alimentos disponibles, produciendo una toma de decisiones adecuada sobre qué alimento asignar a un niño en particular. Considera factores como sus preferencias, la disponibilidad de alimentos, contenido nutricional, entre otros.

- **DwM** [29]: *Dealing with Malnutrition*, sistema de planificación de comidas diseñado para ayudar a personas mayores. Se presenta como un enfoque innovador basado en IA, con un planificador que busca ayudar a los usuarios en la modificación de sus hábitos alimenticios proporcionando recomendaciones centradas en el usuario. Lo distintivo de este sistema está en su capacidad para considerar no solo restricciones dietéticas y valores nutricionales, sino también factores como el tiempo de preparación, la dificultad en la preparación, la disponibilidad de ingredientes, el costo y los gustos del usuario según la clasificación de las recetas. Al integrar estos factores, DwM ofrece recomendaciones más apropiadas al usuario, abordando no solo sus restricciones alimenticias, sino también sus preferencias y circunstancias específicas.

- **DWBIDS** [48]: *Development of a Web Based Intelligent Dietetic System*, sistema dietético inteligente que contribuye a la creación de una dieta equilibrada según el horario de trabajo, índice de masa corporal (IMC) y cualquier dolencia específica que pueda tener el individuo. Consiste en un robot (bot) implementado con información

sobre los planes de control de peso humano, actuando como asesor dietético similar a un dietista experto.

2.3.2. Aplicaciones centradas en la recomendación de menús

Actualmente, existen diversas herramientas que ofrecen recomendaciones de menús o platos basados en indicaciones y restricciones específicas, aplicaciones que suelen enfocarse en promover hábitos saludables y un estilo de vida equilibrado.

A continuación, se presentan diversas aplicaciones centradas en la recomendación de menús:

- **Mealime** [49]: Aplicación móvil (véase la Figura 2.4) que permite a los usuarios planificar comidas saludables mediante la provisión de recetas variadas y nutritivas. Aunque la aplicación se centra en la personalización del plan a través de la selección de platos por parte del usuario, también posibilita la generación automática de un menú alimenticio, considerando tanto restricciones alimenticias como ingredientes a evitar. Sin embargo, no tiene en cuenta el perfil del usuario, como sus edades o niveles de actividad física. Al recomendar los planes, muestra varias planes con diferentes comidas en cada uno y permite realizar modificaciones de la comidas recomendadas, así como seleccionar el número de platos por menú, sin embargo, no establece un régimen específico por días, simplemente ofrece una variedad de platos para que el usuario organice a su gusto, lo que resulta en la falta de un menú planificado.

  Para cada plato, se proporciona información sobre los ingredientes necesarios, ajustable hasta un máximo de 6 comensales, junto con la receta, que incluye una función de paso a paso. En la opción de pago, se ofrece información nutricional detallada.

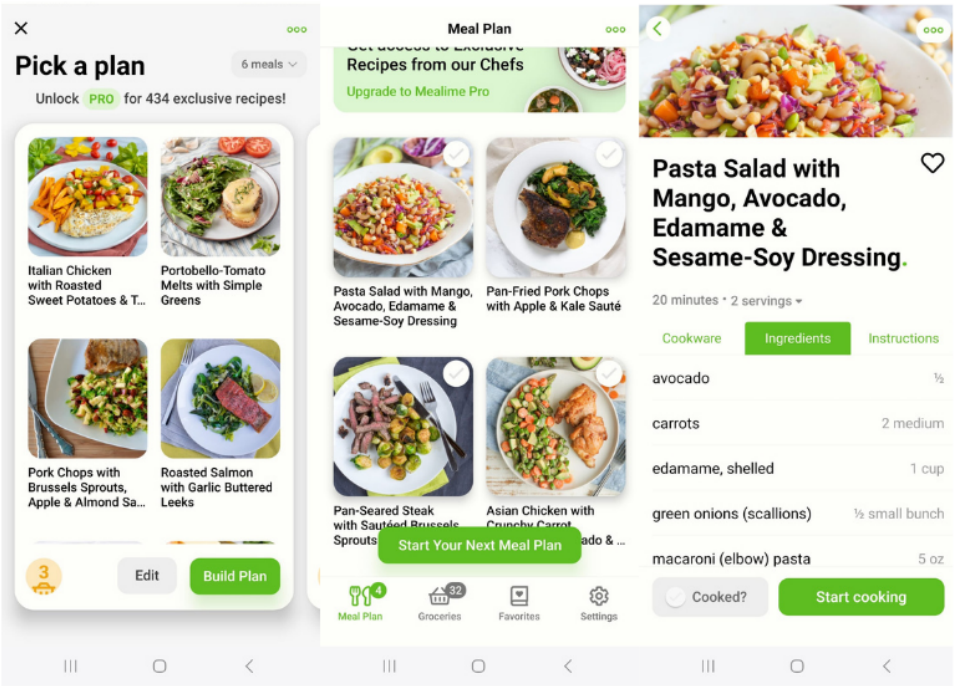


Figura 2.4: Aplicación Movil Mealime

En la sección de pago, también se encuentran otras funcionalidades, como la posibilidad de agregar nuevos platos al recomendador y acceder a un mayor número de recetas. Otra característica destacada de la aplicación es la generación de una lista de compras que incluye todos los ingredientes necesarios para los planes alimenticios creados, pero carece de información sobre el costo de los ingredientes o del plan. Tanto la lista de la compra como el plan y las recetas se pueden exportar en formato PDF.

- **Consum planificador** [33]: La página web perteneciente a la cooperativa de consumo Consum ofrece un servicio web de planificación semanal, que se muestra en la Figura 2.5, y que se adapta a las necesidades de los usuarios. Este planificador es personalizable según el número de comensales y los requisitos dietéticos específicos. Sin embargo, no tiene en cuenta el perfil individual de los usuario. Proporciona una selección diaria de platos y alimentos para cada momento del día, incluyendo desayuno, comida, merienda y cena.

  Permite cambiar los platos recomendados de manera individual y, para cada uno de ellos, incluye información detallada sobre los ingredientes necesarios, la receta y datos nutricionales básicos, como los macronutrientes. Aunque al principio se puede seleccionar el número de comensales, las recetas tienen un número fijo de comensales. Además, esta plataforma permite exportar en formato PDF tanto la receta de un plato como el menú recomendado, junto con una lista de la compra que incluye los ingredientes necesarios. Sin embargo, no proporciona información sobre el costo de los productos ni de los planes recomendados.

Figura 2.5: Plataforma web planificador de menú Consum

- **Spoonacular** [54]: Aplicación web y móvil (véase la Figura 2.6) que ofrece una planificación semanal bastante visual y completa, presentando tres platos por día. Estos pueden generarse automáticamente o seleccionarse de un conjunto de platos recomendados, permitiendo al usuario modificar algún plato específico en los menús propuestos. Para esta recomendación, la aplicación tiene en cuenta el tipo de dieta, intolerancias, el objetivo de calorías por día y los ingredientes a excluir, pero no considera el perfil del usuario, como su edad o peso.

  Para cada plato, la aplicación proporciona los ingredientes ajustables según el número de comensales deseado, las instrucciones de la receta y una información detallada sobre los aspectos nutricionales, incluyendo una estimación de costos

tanto por plato como por menú. Además, genera una lista de la compra basada en los menús propuestos, la cual puede exportarse. Sin embargo, los planes y las recetas no son exportables, lo que limita la portabilidad de la información generada.

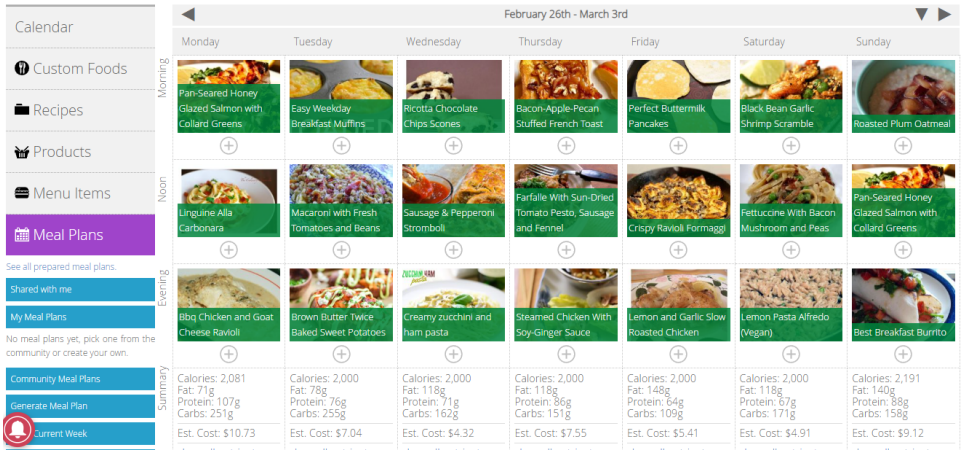


Figura 2.6: Plataforma Web Spoonacular

- **Eat This Much** [46]: La aplicación móvil y web, mostrada en la figura 2.7, ofrece la generación de un menú diario basado en un objetivo de calorías específico y un tipo de dieta seleccionados. Cuenta con una pestaña de ayuda para calcular el número ideal de calorías basado en el perfil del usuario, que incluye información como sexo, peso, altura, edad y nivel de actividad física. Dentro del objetivo de calorías, la aplicación permite personalizar el porcentaje de macronutrientes correspondientes al total. Sin embargo, al planificar, se basa unicamente en un objetivo de calorías final para recomendar los platos.

  Para cada plato del menú generado incluye una lista detallada de ingredientes, pasos de preparación e información nutricional. La opción de pago ofrece funcionalidades adicionales como la generación de un menú semanal, la creación de una lista de la compra y la posibilidad de exportar planes y recetas, entre otras opciones.

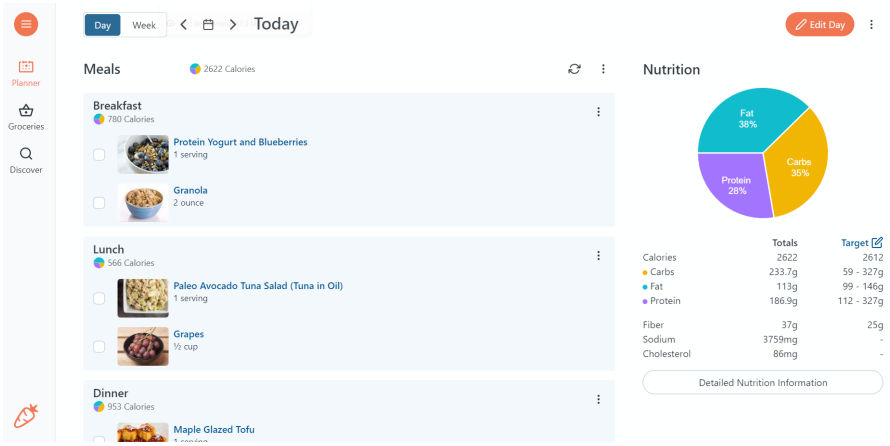


Figura 2.7: Plataforma Web Eat This Much

- **MyRealFood** [51]: Una aplicación web y móvil de origen español, que se destaca por su planificador nutricional altamente personalizado, dedicado a mejorar el estilo de vida de los usuarios. Ofrece en sus opciones de pago un planificador nutricional

adaptado al estilo de vida y al perfil de cada usuario, teniendo en cuenta parámetros como la edad, peso, nivel de actividad física, así como las restricciones alimenticias y los ingredientes a reducir.

Esta aplicación cuenta con una base de datos completamente independiente, creada por expertos nutricionistas. Entre otras funcionalidades, permite el escaneo gratuito de alimentos, mostrando su puntuación, información nutricional e incluso sugiere alternativas más saludables.

- **Ekilu** [36]: Aplicación móvil para la planificación y personalización de menús. Entre sus funciones destaca la recomendación de platos basada en ingredientes disponibles e introducidos por el usuario.
- **Menutech** [50], aplicación web de diseño y personalización de menús que incorpora inteligencia artificial, en la versión de pago, para la revisión de los menús y la recomendación de platos.
- **Recetas Calendario** [45], aplicacion movil para la planificacion de comidas, que permite seleccionar de manera sencilla un plan de alimentación de entre las opciones ofrecidas, adaptándose a diversas preferencias de los usuarios.

2.3.3. Comparación de las aplicaciones relacionadas

Al realizar una comparación entre los objetivos y las funcionalidades de algunas de las aplicaciones mencionadas anteriormente, es posible destacar las características individuales de cada una.

La Tabla 2.1 presenta las funcionalidades ofrecidas por alguna de las aplicaciones descritas anteriormente. Se han seleccionado para la comparación aquellas aplicaciones que ofrecían más características y permitían realizar una evaluación detallada.

Tabla 2.1: Comparación de las características de las aplicaciones relacionadas

	SCHOOLTHY	Mealime	Consum	Spoonacular	Eat This Much	MyRealFood
Generar recomendación de menús	✓	✓	✓	✓	✓	*
Añadir restricciones al menú	✓	✓	✓	✓	✓	✓
Tiene en cuenta el perfil del usuario en la planificación	*	✗	✗	✗	✓	✓
Añadir un plato a la app	✓	✓	✗	✓	✓	✗
Algoritmo de optimización de variedad, costo y nutrientes	✓	✗	✗	✗	*	*
Modificar cantidades del menú	✗	*	✓	✓	✓	*
Modificar plan propuesto	✗	✓	✓	✓	✓	*
Lista de la compra	✗	✓	✓	✓	*	*
Info nutricional sobre los menús	✓	*	✓	✓	✓	*
Info del costo del menú	✓	✗	✗	✓	✗	✗

* Indica una funcion en la aplicacion parcial o de pago.

Como podemos observar en la tabla, la aplicación más completa es Spoonacular, ya que cumple con casi todas las funcionalidades indicadas en su versión gratuita. Sin embargo, cabe destacar que la información sobre el costo del menú podría estar obsoleta y ser mejorada, como ocurre también en SCHOOLTHY.

Dada la posible obsolescencia de los costos proporcionados por SCHOOLTHY y Spoonacular debido a los constantes cambios en los precios de los productos alimenticios, se propone la integración de una API de comparación de precios para obtener información más precisa y actualizada. Para ello, se planea utilizar la API desarrollada en el trabajo *Desarrollo de una aplicación full-stack para la geolocalización de productos alimenticios* [40].

Algunas de las aplicaciones, como Mealime, presentan dificultades de uso al modificar restricciones alimenticias para diferentes planes, y además, limitan el número de comensales a menos de 10, una restricción que también se observa en Consum.

Una de las funcionalidades que pocas aplicaciones tienen en cuenta es la consideración del perfil del usuario al generar la recomendación de menús. Las únicas aplicaciones que abordan esta cuestión son SCHOOLTHY, que se centra en una población infantil específica, y Eat This Much y My Real Food. Sin embargo, estas dos últimas solo tienen en cuenta en la optimización el número total de calorías y las macronutrientes, sin considerar la optimización de la variedad y el costo del menú.

A partir del análisis de todas las aplicaciones, se observa la falta de varios aspectos importantes. Se evidencia una notable falta de datos sobre los costes del menú, y especialmente, la carencia de información actualizada sobre estos. Asimismo, se echa en falta la opción de recomendar varios menús según el presupuesto disponible, una característica que solo se encontró en SCHOOLTHY. Otra cuestión que muchas de las aplicaciones no tuvieron en cuenta fue la adaptación de las partes nutricionales según el perfil de usuario, incluyendo aspectos como la edad, la actividad física, el peso, entre otros. Por lo tanto, los algoritmos de recomendación podrían mejorar si se optimizan considerando varias variables importantes, como el costo, el perfil del usuario y los nutrientes, y no únicamente basándose en la exclusión de alimentos con alérgenos, como ocurría en la mayoría de los casos.

Finalmente, la aplicación que se busca desarrollar sería aquella que cumpla con todos los requisitos mencionados, similar a los servicios ofrecidos por Spoonacular. Que mantenga la facilidad de uso e intuición de Consum, al mismo tiempo que permita integrar un algoritmo de optimización para la generación de menús, similar al utilizado por SCHOOLTHY. De esta manera, se pretende alcanzar un diseño simple, accesible y con la mayor cantidad de funcionalidades posible.

Capítulo 3

Modelo de datos y base de datos

Uno de los pilares fundamentales de una aplicación web son las bases de datos y los modelos de datos. Estos elementos permiten almacenar, organizar y gestionar la información de manera eficiente, lo cual es crucial para el funcionamiento y la escalabilidad de cualquier aplicación moderna. En este capítulo, se abordarán en detalle la base de datos escogida, el modelo de datos definido y la carga inicial de datos.

3.1. Base de datos

A continuación, se va a indicar la base de datos elegida para el alojamiento de la información de la aplicación web, así como el despliegue de esta base de datos en un servidor en la nube.

3.1.1. MongoDB y Mongoose

Existen varias alternativas para la gestión de bases de datos en aplicaciones web, como MySQL [18], PostgreSQL [44], SQLite [25] y MongoDB [15], cada una con sus propias ventajas y desventajas. Para este proyecto se ha decidido utilizar MongoDB como base de datos para almacenar la información de la aplicación y la persistencia de los datos, junto con Mongoose [16]. MongoDB es una base de datos NoSQL que guarda los datos en formato JSON, lo que facilita su uso e integración con aplicaciones web modernas. Se escogió MongoDB por su flexibilidad del esquema y su facilidad de uso.

Mongoose, por su parte, es una herramienta de modelado de objetos para Node.js [19] que permite definir esquemas para los documentos de MongoDB. Además, Mongoose proporciona validaciones y otras funcionalidades útiles para manejar la lógica directamente en el modelo de datos, simplificando así el desarrollo y asegurando la integridad de los datos.

3.1.2. Despliegue de la base de datos

Para el despliegue de la base de datos se ha utilizado MongoDB Atlas [14], ya que no se cuenta con infraestructura física para almacenar la base de datos y se opta por utilizar servicios en la nube. MongoDB Atlas ofrece un servicio gratuito limitado que permite crear una cuenta, registrar un clúster compartido y generar aquí la base de datos de producción con las colecciones correspondientes.

MongoDB Atlas proporciona una URL de conexión que facilita la integración de la base de datos en el *backend* a través de la API, permitiendo realizar operaciones tanto en el entorno de producción como en el de pruebas. Es por eso que se han creado dos bases de datos (véase la Figura 3.1): una en producción con los datos reales de la aplicación web y otra base de datos para la realización de las pruebas del *backend*.

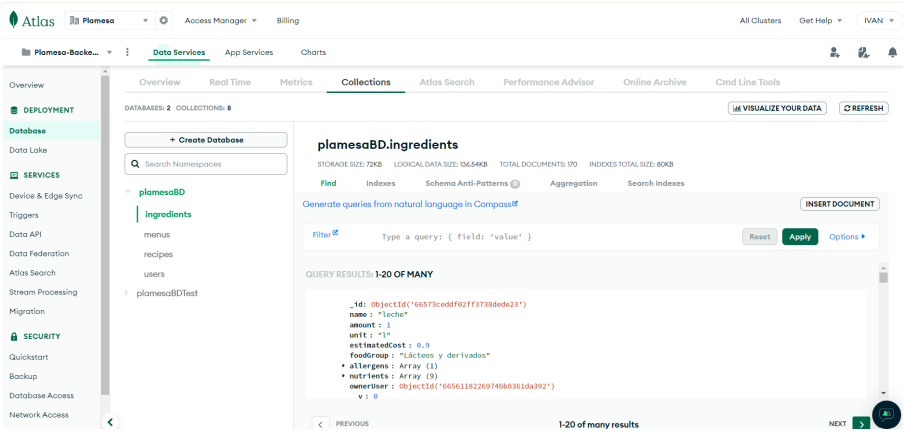


Figura 3.1: Despliegue de la base de datos en MongoDB Atlas

3.2. Entidades del modelo de datos

Para el modelo de datos de este proyecto se han definido cuatro entidades, que se muestra en la Figura 3.2, y que se detallarán a continuación.

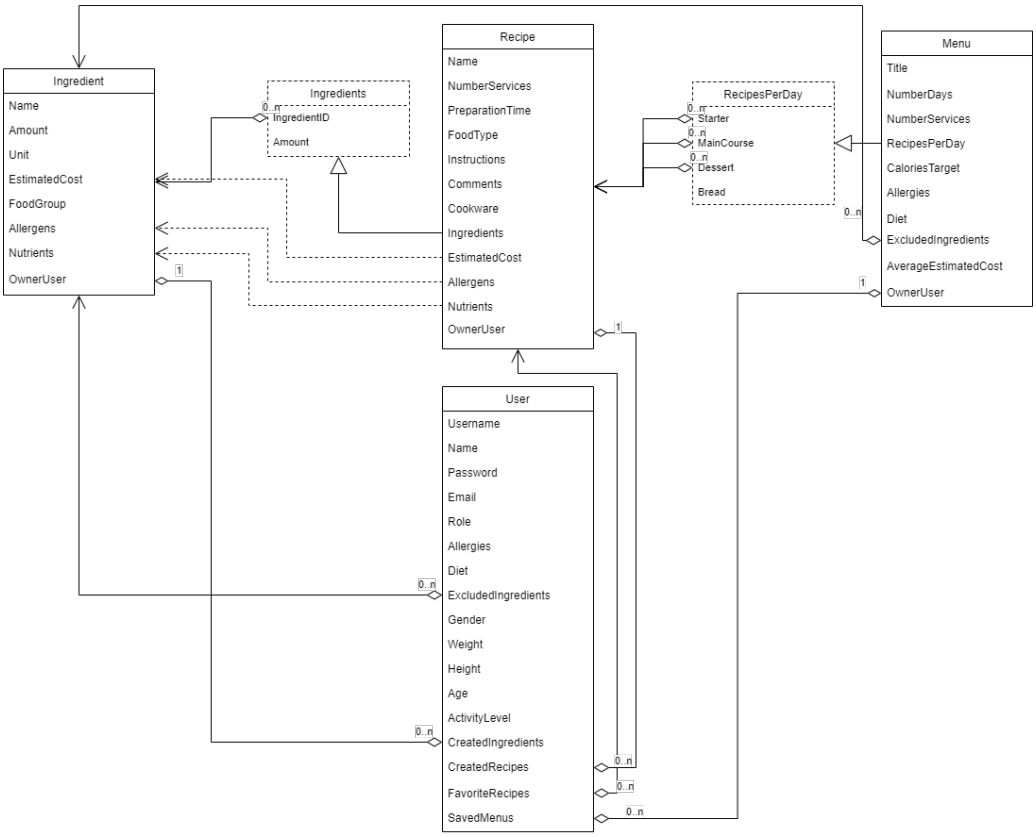


Figura 3.2: Entidades del modelo de datos

- **User:** Entidad destinada a la identificación de un usuario en la aplicación web. Con esta entidad se permitirá la autenticación de los clientes en la aplicación web, se almacenarán los datos y el perfil del usuario que se utilizarán para la optimización de los menús, y se almacenarán los ingredientes, recetas y menús guardados por el usuario.
- **Ingredient:** Esta es la entidad primaria del modelo de datos que representa un ingrediente en el sistema. Contendrá información nutricional, de alérgenos y de coste estimado.
- **Recipe:** Entidad que modela una receta. Contendrá toda la información propia de una receta, como el nombre, tiempo de preparación, tipo de comida, instrucciones e ingredientes (que harán referencia a la entidad Ingredient). Además, tendrá información de alérgenos, información nutricional y coste estimado calculado a partir de los ingredientes contenidos en la receta.
- **Menu:** Esta entidad del modelo de datos está destinada a almacenar información sobre un menú que un usuario desee guardar. Contiene las recetas que componen el menú generado, así como las restricciones utilizadas para generar el menú, incluyendo alérgenos, objetivo de calorías, dieta y coste estimado del menú.

3.2.1. Modelado de los objetos

A continuación, se definen los atributos de cada una de las entidades del modelo de datos junto con sus restricciones y requerimientos.

Modelo de usuario

El modelo de datos de usuario tiene 17 atributos.

- **Información básica del usuario**
  - **Username:** Nombre de usuario, utilizado para identificar únicamente en la aplicación web al usuario, por lo que es requerido y debe ser único. Para ello usamos el atributo *unique* que nos proporciona Mongoose.
  - **Name:** Nombre completo del usuario, es requerido y no tiene ninguna restricción.
  - **Password:** Contraseña que se utiliza para la verificación del usuario en la aplicación web, es requerida y se valida que contenga al menos 6 caracteres, 1 mayúscula y 1 número.
  - **Email:** Dirección de correo electrónico del usuario que debe ser única en la base de datos, es requerida y se valida que tenga formato de email.
  - **Role:** El rol es un atributo que se utiliza para diferenciar los usuarios regulares de un usuario administrador. Un usuario regular sería el que usaría la aplicación web como cliente y el administrador tendría más permisos y privilegios a la hora de realizar operaciones CRUD sobre todas las entidades, por lo que es un atributo requerido y solo tiene las opciones de usuario regular y administrador.

- **Información sobre el perfil del usuario.** Esta información es opcional y tiene como objetivo el cálculo del metabolismo basal y el consumo de calorías de un usuario para la optimización de la planificación de menús.
  - **Gender y Age:** Género (Masculino o Femenino) y edad del usuario.
  - **Weight y Height:** Peso y altura del usuario, ambos superiores a 0.
  - **ActivityLevel:** Nivel de actividad del usuario, comprendido entre sedentario, ligero, moderado, activo y muy activo.
- **Información sobre gustos y alergias del usuario.** Esta información también es opcional.
  - **Allergies:** Vector con los alérgenos del usuario. Estos alérgenos podrán ser los comprendidos entre los 14 alérgenos obligatorios de indicar, comentados en la Sección 3.3.2.
  - **Diet:** Tipo de dieta del usuario, comprendida entre las siguientes: Vegetariana, vegana y para diabéticos.
  - **ExcludedIngredients:** Listado de ingredientes a excluir a la hora de recomendar un menú. Estos se almacenan en forma de ID, proporcionados por la base de datos MongoDB.
- **Información sobre entidades almacenadas por el usuario.**
  - **CreatedIngredients:** Vector con los IDs referenciados a los objetos de la entidad *Ingredient*, correspondientes a los ingredientes creados por el usuario.
  - **CreatedRecipes:** Vector con los IDs referenciados a los objetos de la entidad *Recipe*, correspondientes a las recetas creadas por el usuario.
  - **FavoriteRecipes:** Vector con los IDs referenciados a los objetos de la entidad *Recipe*, correspondientes a las recetas favoritas del usuario.
  - **SavedMenus:** Vector con los IDs referenciados a los objetos de la entidad *Menu*, correspondientes a los menús generados y guardados por el usuario.

Modelo de Ingrediente

El modelo de datos de ingrediente tiene 8 atributos.

- **Name:** Nombre identificativo del ingrediente, que es obligatorio y único en la base de datos. Este se almacenará siempre en minúsculas.
- **Amount:** Cantidad del ingrediente para la cual se indican los valores del ingrediente, por defecto 100.
- **Unit:** Unidad de medida del ingrediente y de la cantidad indicada, por defecto gramos (gr).
- **EstimatedCost:** Coste estimado del ingrediente en euros para la cantidad indicada. Este es requerido y debe ser superior a 0.
- **FoodGroup:** Grupo de alimento del ingrediente a seleccionar entre los 13 grupos de alimentos definidos a partir de los grupos indicados en BEDCA [31].

- **Allergens:** Listado con los alérgenos del ingrediente, entre los 14 alérgenos definidos en la sección 3.3.2.
- **Nutrients:** Listado de nutrientes junto con su valor y unidad. Se valida que al menos se incluyan los 7 nutrientes obligatorios definidos en la sección 3.3.3, junto con su cantidad. El resto de posibles nutrientes son opcionales, y para la unidad de cada nutriente se ha creado una pequeña función que, según el nombre del nutriente, devuelve su unidad correspondiente al guardar el nutriente.
- **OwnerUser:** Identificador referenciado a la entidad *User*, correspondiente al usuario creador del ingrediente.

Modelo de Receta

El modelo de datos de receta tiene 12 atributos.

- **Name:** Nombre identificativo de la receta. Este es requerido siempre y se convierte a minúsculas al introducirlo.
- **NumberServices:** Número de porciones, es decir, personas para las que se definen las cantidades de los ingredientes. Es obligatorio y debe ser mayor que 0.
- **PreparationTime:** Tiempo de preparación en minutos de la receta. Debe ser superior a 0 y es requerido siempre.
- **FoodType:** Atributo para identificar el tipo de comida, ya sea entrante, plato principal o postre.
- **Instructions:** Vector con las instrucciones y pasos de la receta. Es requerido.
- **Comments:** Atributo opcional para indicar comentarios sobre la receta.
- **Cookware:** Listado opcional con los utensilios de cocina necesarios para la receta.
- **Ingredients:** Listado de ingredientes de la receta. Se requiere al menos un ingrediente, y se almacena el identificador que hace referencia al ingrediente, junto a la cantidad del ingrediente para la receta definida.
- **EstimatedCost:** Corresponde al coste estimado de la receta. Este se calculará automáticamente al crear la receta, a partir de los ingredientes junto a su cantidad y el precio de cada ingrediente.
- **Allergens:** Listado con los alérgenos de la receta, que se rellenan automáticamente a partir de los alérgenos de cada ingrediente referenciado en la receta.
- **Nutrients:** Listado con los nutrientes junto a su cantidad, calculados automáticamente a partir de los ingredientes incluidos en la receta y sus cantidades.
- **OwnerUser:** Identificador referenciado al usuario propietario de la receta.

Modelo de Menú

El modelo de datos de menú tiene 10 atributos.

- **Title:** Título identificativo para nombrar el menú. Por defecto será la fecha de creación del menú.
- **NumberDays:** Número de días para los que se genera el menú, debe ser mayor que 0 y es requerido.
- **NumberServices:** Número de personas para los que se genera el menú, debe ser mayor que 0 y es requerido.
- **RecipesPerDay:** Vector con referencias a las recetas que componen el menú. Cada posición del vector representa el menú de un día y estará compuesta por un identificador para una receta del tipo entrante, otra receta como plato principal, otra como postre y, si se incluye, una porción de pan en el menú. La longitud del vector debe coincidir con el número de días del menú.
- **CaloriesTarget:** Número de calorías objetivo por día y persona para las que se optimiza el menú de cada día. Se calcula a través de la fórmula y el perfil del usuario mencionado en la sección 3.3.1.
- **Allergies:** Vector con las alergias que restringen las recetas a incluir en el menú.
- **Diet:** Tipo de dieta que restringe ciertas recetas del planificador.
- **ExcludedIngredients:** Vector con referencias de ingredientes a excluir del menú.
- **AverageEstimatedCost:** Coste medio por persona y día del menú guardado, calculado a partir de los costes de cada receta del menú.
- **OwnerUser:** Identificador del usuario que guarda el menú.

3.3. Consideraciones a la hora de modelar los objetos

A la hora de modelar los objetos y definir los atributos de las entidades del modelo de datos, se han investigado ciertas restricciones y valores a tener en cuenta.

3.3.1. Perfil de usuario

A la hora de definir el perfil de un usuario, se han tenido en cuenta los datos necesarios para poder calcular el metabolismo basal y el consumo estimado de calorías diario, para así poder realizar una recomendación de menús adaptada a cada tipo de usuario.

Para este cálculo se utiliza la ecuación de Harris-Benedict [5], una ecuación empírica para estimar el metabolismo basal y el consumo estimado de calorías diario de una persona a partir del género, edad, peso, altura y nivel de actividad física:

$$\text{MB (hombres)} = (10 \times \text{peso en kg}) + (6{,}25 \times \text{altura en cm}) - (5 \times \text{edad en años}) + 5 \quad (3.1)$$

$$\text{MB (mujeres)} = (10 \times \text{peso en kg}) + (6{,}25 \times \text{altura en cm}) - (5 \times \text{edad en años}) - 161 \quad (3.2)$$

Con estos datos se puede calcular el objetivo de calorías diarias y, a partir de este objetivo total, calcular el porcentaje de calorías correspondientes a la comida, que en el caso de este proyecto es para la cual se realiza la recomendación diaria de recetas. Este porcentaje corresponde al 30 % [6] de las calorías totales diarias de una persona.

3.3.2. Alérgenos obligatorios

A la hora de definir los alérgenos de un ingrediente, se han determinado a partir de los 14 alérgenos de declaración obligatoria definidos en el Real Decreto 126/2015. Estos se muestran en la Figura 3.3, e incluyen gluten, crustáceos, huevos, pescado, cacahuetes, soja, lácteos, frutos con cáscara, apio, mostaza, sésamo, sulfitos, altramuces y moluscos [11].

Figura 3.3: Alérgenos de obligada declaración

3.3.3. Nutrientes obligatorios

Para definir los nutrientes a incluir de los diferentes ingredientes, se ha determinado que los nutrientes obligatorios a incluir en un ingrediente correspondan con el etiquetado obligatorio sobre información nutricional para la mayoría de los alimentos [1]. Los elementos a declarar de forma obligatoria son: el valor energético, las grasas, las grasas saturadas, los hidratos de carbono, los azúcares, las proteínas y la sal. Estos incluyen los macronutrientes, entre otros.

Además de los nutrientes de obligada incorporación, se da la opción de introducir información sobre otros nutrientes, correspondientes a micronutrientes, minerales y vitaminas, además de valores de fibra y colesterol, se muestran todos en la Figura 3.4.

Entre los micronutrientes se incluyen:

- **Minerales:** Calcio, hierro, potasio, magnesio, sodio, fósforo, yodo, selenio, zinc.
- **Vitaminas:** A, B6, B12, C, D, E.

Figura 3.4: Nutrientes posibles a incluir en las entidades

3.4. Carga de datos

Para la carga inicial de ingredientes y recetas en la base de datos de la aplicación web, se había pensado usar las bases de datos de alimentos analizadas en el Capítulo 2 de este proyecto. Estas tablas y bases de datos de alimentos ofrecen información sobre ingredientes y alimentos, información nutricional completa, así como información sobre grupos de comidas y alérgenos.

Entre las bases de datos disponibles, en este proyecto nos centramos en sacar la información de BEDCA [31] y OpenFoodFacts [39], ambas gratuitas. El principal problema de la primera es que, a pesar de incluir una gran cantidad de ingredientes primarios, no ofrece una manera nativa de exportar los datos, ni una API a la que realizar consultas, lo que dificulta la obtención de los datos. Además, la información nutricional de los alimentos de esta no es del todo completa, ya que no incluye información sobre todos los valores nutricionales de obligado etiquetado.

Por otra parte, OpenFoodFacts sí que incluye diferentes sistemas de exportar datos, ya sea a través de descargas de ficheros CSV, JSON, etc., o incluso a través de una API disponible gratuitamente, a la que se puede consultar individualmente diferentes ingredientes y obtener toda la información del ingrediente completa. Sin embargo, uno de los problemas de esta base de datos es la falta de ingredientes primarios, ya que resulta complicado encontrar información nutricional sobre alimentos básicos, como pueden ser frutas y verduras.

Finalmente, se investigó la forma de utilizar una inteligencia artificial para la generación de ingredientes y obtención de los datos a partir de bases de datos oficiales. Tras investigar diferentes opciones de las distintas IAs a utilizar, finalmente se optó por el uso de ChatGPT [9], en su versión gratuita, pero con las peticiones limitadas de la última versión GPT-4, su versión más actualizada y con acceso a internet para obtener información en tiempo real. Es principalmente por esto por lo que se convierte en la opción más adecuada, ya que nos permite pedirle que nos genere información sobre diferentes ingredientes, indicándole que haga uso de las bases de datos BEDCA y OpenFoodFact para la obtención de datos nutricionales. Además,de indicarle que utilice comparadores de precios online para obtener el coste de los ingredientes más actualizado.

3.4.1. Generación de ingredientes

Para la generación de los ingredientes se ha usado un prompt en el que se indicaba el formato de salida de los ingredientes así como la manera de obtener la información nutricional y de costos:

```
Buenas, quiero que generes un fichero con un vector de Json con información
    sobre X ingredientes, con la siguiente estructura:

- name: Nombre del ingrediente
- amount: cantidad del ingrediente para el que se valoran los datos
- unit: Unidad de medida del ingrediente, correspondiente al amount entre los
    siguientes [gr, l, ud, oz]
- estimatedCost: Coste estimado del ingrediente para el amount indicado
```

```
- foodGroup: grupo de comida correspondiente del ingrediente perteneciente a uno
    de los siguientes: <Listado con los FoodGroups posibles>
- allergens: vector de los alérgenos del ingrediente, posibles entre los
    siguientes: <Listado con los Allergens disponibles>
- nutrients: vector de nutrientes con su cantidad en el formato de entrada será
    { name: nombre nutriente, amount: cantidad del nutriente para la cantidad del
    ingrediente}, obligatoriamente se deben incluir 7 nutrientes para todos los
    ingredientes [Energía, Proteínas, Carbohidratos, Grasa Total, Sal, Azúcares,
    Grasas Saturadas], el resto serán opcionales pero recomendables si se pueden
    obtener, y pueden ser los siguientes: <Listado con el resto de nutrients
    disponibles>

Para obtener los valores nutricionales quiero que compares y obtengas de bases
    de datos alimentarias como Bedca y OpenFoodFact. Para obtener costes
    actualizados quiero que uses comparadores de precios y APIs de supermercados.

Muchas gracias
```

El número de ingredientes 'X' que se le pedía variaba según la petición. Inicialmente, se le pidieron 40 ingredientes que se usaran en recetas variadas y nutritivas. Después, se fueron pidiendo ingredientes en grupos de aproximadamente 10, según el grupo de alimentos.

3.4.2. Generación de recetas

Para la generación de las recetas se ha usado un prompt en el que se indicaba el formato de salida de las recetas así como el listado con la relación de los ingredientes disponibles en la base de datos y sus respectivos identificadores:

```
Buenas, quiero que generes un fichero con un vector de JSON con información
    sobre X recetas, con la siguiente estructura:

- name: Nombre de la receta
- numberServings: número de servicios, personas de la receta
- preparationTime: tiempo de preparación en minutos
- foodType: tipo de comida entre las opciones [Entrante, Plato Principal, Postre]
- instructions: vector de strings, cada posición del vector será una instrucción
- comments: string con algún comentario especial de la receta, opcional
- cookware: vector de strings con los utensilios de cocina de la receta,
    opcional, cada posición del vector será un utensilio
- ingredients: Estos son los ingredientes de la receta junto con su cantidad. Se
    definen a través de un id de tipo MongoDB. La estructura es un vector de
    esto: { ingredientID: id del ingrediente, amount: cantidad del ingrediente
    para la receta, un número sin la unidad }[]

A continuación te paso la lista de ingredientes disponibles junto con el id
    correspondiente y la unidad del ingrediente. MUY IMPORTANTE: LOS INGREDIENTES
    DISPONIBLES SON ESTOS SOLO. Recuerda mirar las unidades; por ejemplo, el agua
    y la leche están en litros, no mililitros:

leche - 66573ceddf02ff3738dede23 - L
```

```
<Resto de ingredientes>

Muchas gracias
```

El número de recetas ’X’ que se le pedía variaba según la petición. Inicialmente, se le pidieron 30 recetas que consistieran en ser variadas y nutritivas. Después, se fueron pidiendo recetas en grupos de aproximadamente 10, según el tipo de comida.

3.4.3. Script de carga

El formato de salida tanto de los ingredientes como de las recetas generadas por la inteligencia artificial es en formato JSON. Para la carga de estos JSON en la base de datos de MongoDB Atlas, se ha creado un pequeño script que utiliza tecnologías basadas en Node.js, en concreto TypeScript, y Axios para realizar peticiones HTTP a la API del *backend*, en la que se han implementado los sistemas de carga en la base de datos.

El funcionamiento de este script es muy simple. Primero, realiza una petición de inicio de sesión en el *backend* con un usuario y contraseña existentes para obtener el token de autenticación que se usará para añadir el resto de ingredientes y recetas. A continuación, se lee los ficheros JSON que almacenan todos los ingredientes y recetas y, de forma iterativa, se van realizando peticiones POST para añadir tanto los ingredientes como las recetas a la base de datos. El código de este script se encuentra en el Apéndice A

3.4.4. Costes actualizados

Para la carga y actualización de los costos de los ingredientes, se había planteado al principio del proyecto la incorporación de peticiones a una API externa desarrollada en el Trabajo de Fin de Grado *“Desarrollo de una aplicación full-stack para la geolocalización de productos alimenticios”* [40]. Sin embargo, a lo largo del desarrollo de este proyecto se investigaron otras formas de obtener precios actualizados, debido principalmente a que para la integración de la API desarrollada en el TFG anteriormente mencionado, era necesario cargar y actualizar en la API información sobre ingredientes y productos.

Inicialmente, con el uso de la inteligencia artificial en la primera carga de datos, se obtuvo un costo estimado de productos en tiempo real gracias a la comparación, por parte de la IA, con diferentes APIs y comparadores de precios. Adicionalmente, para mantener actualizada la información sobre estos costos de los ingredientes, se plantea la incorporación de peticiones a la API de Mercadona como una de las posibles líneas futuras de este proyecto.

Después de investigar las posibles formas de obtención de precios actualizados sobre ingredientes, se determinó que la API del supermercado Mercadona [32] es una opción bastante viable, ya que esta ofrece de manera gratuita información a través de peticiones HTTP sobre una gran cantidad de ingredientes y alimentos junto con su precio en tiempo real, contrastados con uno de los principales supermercados de España.

Capítulo 4

PLAMESA: el plan para tu mesa

PLAMESA, planificador de menús saludables, es el nombre que se le ha dado a la aplicación web desarrollada en este proyecto. Esta sigue una estructura de tecnologías MERN: MongoDB [15], Express [7], React [21] y NodeJS [19]. El proyecto se ha dividido en *Backend* y *Frontend*, haciendo que tanto el cliente como el servidor sean independientes uno del otro.

En el *Backend* se define una API REST destinada a actuar como intermediario entre el *frontend* y la base de datos, encargándose de la modelación y validación de los objetos, así como de las operaciones CRUD sobre la base de datos. En el *frontend* se define la interfaz web que será el intermediario entre el usuario y el *backend*. En la Figura 4.1 se muestra la estructura de la aplicación web.

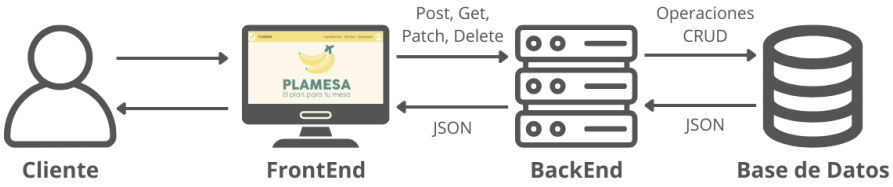


Figura 4.1: Estructura de la aplicación web

4.1. *Backend*

Como se ha mencionado, en el *backend* se utilizan tecnologías basadas en NodeJS. En concreto, para el modelado de datos se utiliza Mongoose y como framework de desarrollo se utiliza Express para instanciar el servidor, en el que se habilitan las diferentes rutas para administrar las peticiones de los clientes. Todo esto está desarrollado en lenguaje TypeScript. A continuación se muestra la estructura de directorios del *backend*:

```
| ...
|-- package.json
|-- README.md
|-- src
|   |-- app.ts
|   |-- db
|   |   |__ mongoose.ts
```

```
|   |-- index.ts
|   |-- models
|   |   |-- ingredient.ts
|   |   |-- ...
|   |__ routers
|       |-- default.routes.ts
|       |-- ingredient.routes.ts
|       |-- ...
|-- tests
|   |-- models
|   |   |-- ingredient.spec.ts
|   |   |-- ...
|   |__ routes
|       |-- ingredient.routes.spec.ts
|       |-- ...
|-- tsconfig.json
|__ typedoc.json
```

La estructura de carpetas del *backend* se puede definir principalmente por los ficheros de configuración y `package.json`, en los que se definen las dependencias y configuraciones del servidor. Además, encontramos las carpetas `src` y `tests`. En la carpeta `src` encontraremos todo el código del *backend* y en la carpeta `tests` encontraremos el codigo de las pruebas realizadas.

En la carpeta `src`, podemos encontrar sueltos los ficheros `app.ts` e `index.ts`. El primero es el encargado de la declaración de la instancia de Express e incluir en esta todas las rutas que comprenderán la API. En el fichero `index.ts`, se inicializa el servidor Express que se instanció en el fichero anterior.

4.1.1. Conexión con la base de datos

Dentro de la carpeta del código `src`, la primera subcarpeta en la que nos fijamos es la de `db`, que contiene el fichero encargado de la conexión con la base de datos a través de Mongoose. Para ello, se utiliza la URI de conexión que nos proporciona la base de datos MongoDB, tanto en entorno local como en la BD desplegada en MongoDB Atlas. Esta URI se almacena como un secreto en una variable de entorno en el fichero .env. Con esto se realiza la conexión con la base de datos y, a continuación, se exporta e importa la instancia de la conexión en el fichero `app.ts` para integrarlo en el proyecto.

4.1.2. Modelos de la API

La siguiente subcarpeta del proyecto corresponde a donde se definen los ficheros con las entidades del modelo de datos. En concreto, se definen cuatro ficheros para declarar los cuatro objetos del modelo de datos con sus atributos, restricciones, tipos y validaciones correspondientes explicadas en el capítulo anterior: *User*, *Ingredient*, *Recipe* y *Menu*.

4.1.3. JWT

Para la autenticación de los usuarios en las rutas se ha utilizado JWT [2] con el paquete `jsonwebtoken` [10]. Usamos el método `sign` para crear el token, que tendrá una hora de expiración y contendrá el ID del usuario autenticado. Usamos un secreto almacenado como variable de entorno en el archivo `.env`. Cuando se recibe una petición que necesita autenticación, se usa el método `verify` para verificar el token, obtener el ID del usuario y devolver el usuario correspondiente a ese ID para las verificaciones posteriores.

4.1.4. Rutas de la API

La siguiente carpeta a analizar corresponde con la definición de las distintas rutas que componen el *backend* y que interactúan con la base de datos. Estas reciben información en formato JSON correspondiente a los atributos de cada entidad, y se utiliza JWT para la autenticación de los usuarios en la API.

Rutas de usuario

- **POST /user:** Ruta encargada de añadir un usuario a la base de datos. Recibe la información de usuario, encripta la contraseña con el uso de bcrypt y guarda el nuevo usuario en la BD.
- **GET /user/all:** Esta ruta devuelve un JSON con la información de todos los usuarios de la aplicación, pero solo a usuarios con rol de administrador. Se utiliza JWT para autenticar al usuario que realiza la petición.
- **GET /user:** Ruta encargada de devolver la información de un usuario concreto. Se utiliza JWT para obtener el usuario que realiza la petición y devolver la información de ese usuario individualmente.
- **PATCH /user:** Esta ruta permite a un usuario cambiar su información en la BD. Se utiliza JWT para autenticar y obtener al usuario que desea cambiar su información, y se realizan los cambios que se incluyen en el *body* de la petición, siempre que sean de las opciones permitidas: información básica, gustos y alergias, y perfil del usuario. En caso de que se modifiquen las recetas favoritas o los ingredientes a excluir, se comprueba que tanto los nuevos identificadores de ingredientes como de recetas existan en la base de datos.
- **DELETE /user:** Es la ruta encargada de la eliminación de un usuario. Primero, se obtiene la información del usuario a eliminar a través de JWT y comienza el proceso de eliminación. Tanto los ingredientes como las recetas creadas por el usuario se transfieren a un usuario con rol de administrador, para asegurar la persistencia de los ingredientes y recetas en la aplicación. En el caso de los menús guardados por el usuario, como pertenecen individualmente a cada usuario, se procede a eliminar de la base de datos todos los menús cuyo propietario sea el usuario a eliminar.
- **POST /login:** Esta ruta se encarga del inicio de sesión y generación de tokens JWT. Recibe el usuario y la contraseña para autenticar, compara las contraseñas con bcrypt y, si coinciden, genera un token JWT en el que se guarda el ID del usuario autenticado. Este token tendrá una duración de 1 hora.

- **POST /calcNutrientsUser:** Esta ruta se encarga de recibir la información de perfil de un usuario y realizar los cálculos de metabolismo basal, consumo de calorías diario, calorías correspondientes a la comida y medidas de macronutrientes correspondientes.

  Para ello, se hace uso de la fórmula de Harris-Benedict [5] y el porcentaje de calorías correspondiente a la comida, definidos en el capítulo anterior. Para los porcentajes de macronutrientes se utilizan los porcentajes indicados en el capítulo dos de este proyecto, en el que se explican los macronutrientes y sus porcentajes recomendados de consumo diario.

Rutas de Ingrediente

- **POST /ingredient:** Ruta encargada de añadir un nuevo ingrediente. Se utiliza JWT para identificar al usuario que realiza la petición, quien se convierte en el propietario del ingrediente. El identificador del ingrediente creado se incluye en el vector de ingredientes creados del usuario.
- **GET /ingredient:** Ruta que devuelve todos los ingredientes de la base de datos.
- **GET /ingredient/:id:** Ruta que devuelve la información de un ingrediente concreto, indicando el identificador como parámetro en la ruta.
- **PATCH /ingredient/:id:** Ruta encargada de modificar la información de un ingrediente concreto indicado en los parámetros de la ruta. Se comprueba a través de JWT que el usuario que realiza la petición sea el propietario o un administrador. A continuación, se comprueba que los atributos a actualizar sean correctos y se validan en el modelo de datos.

  En caso de que se produzca una modificación del coste estimado o de la cantidad del ingrediente, se buscan todas las recetas que incluyan este ingrediente y se recalcula el coste estimado de la receta con el nuevo coste del ingrediente. En caso de que se modifiquen los nutrientes o sus valores, también se recorren todas las recetas con este ingrediente y se recalculan los nuevos nutrientes de la receta.

  En caso de actualizar los alérgenos del ingrediente, se buscan las recetas que incluyen el ingrediente y se actualizan los alérgenos.
- **DELETE /ingredient/:id:** Esta ruta se encarga de la eliminación de un ingrediente. El ingrediente se especifica como parámetro de la petición y, primero, se comprueba que el ingrediente exista y que el usuario indicado a través de JWT sea el propietario o un administrador. A continuación, se comprueba que no exista ninguna receta con este ingrediente.

  Si procede, se elimina el ingrediente de la lista de ingredientes a excluir de cualquier usuario y menú, y se elimina el ingrediente del listado de ingredientes creados del usuario propietario, concluyendo con la eliminación del ingrediente.

Rutas de Receta

- **POST /recipe:** Ruta que gestiona la creación de una nueva receta. Recibe el usuario con JWT y primero comprueba la existencia de los ingredientes a añadir y calcula los parámetros de la receta.

Calcula el costo estimado, los alérgenos y los nutrientes de la receta a través de los ingredientes introducidos y sus cantidades, añadiendo la receta al listado de recetas creadas por el usuario propietario.

- **GET /recipe:** Devuelve todas las recetas de la base de datos.
- **GET /recipe/:id:** Ruta que devuelve la información de una receta concreta, indicada como parámetro en la ruta.
- **PATCH /recipe/:id:** Ruta encargada de la actualización de la información de una receta. Se indica la receta como parámetro de la ruta, se comprueba que exista, que el usuario indicado con JWT sea el propietario o un administrador, y que los atributos a modificar tengan el formato y opciones correctas.

  En caso de que se modifiquen los ingredientes de la receta, se comprueban estos y se recalculan los valores de la receta. Se calculan nuevamente el coste estimado de la receta, los nutrientes y los alérgenos a partir de los nuevos ingredientes y sus cantidades.
- **DELETE /recipe/:id:** Ruta encargada de la eliminación de una receta indicada como parámetro de la ruta. Se comprueba que exista y que el usuario indicado con JWT sea el propietario o un administrador. A continuación, se comprueba que no exista ningún menú guardado que contenga esta receta.

  Si procede la eliminación, se elimina la receta de todos los listados de recetas favoritas de los usuarios y se elimina del vector de recetas creadas del usuario propietario.
- **POST /recipeSearchPerIngredients:** Esta ruta recibe ingredientes y devuelve un listado de 5 recetas que contengan el máximo número de esos ingredientes.

Rutas de Menú

- **POST /menu:** Esta es la ruta encargada de guardar un menú en la base de datos. Obtiene el usuario a través del token JWT, verifica la existencia de las recetas y los ingredientes incluidos en el menú, guarda el menú y añade el menú al listado de menús guardados del usuario propietario.
- **GET /menu:** Esta ruta devuelve un JSON con la información de todos los menús de la aplicación, pero solo a usuarios con rol de administrador. Se utiliza JWT para autenticar al usuario que realiza la petición.
- **GET /menu/:id:** Esta ruta es la encargada de devolver la información de un menú concreto indicado como parámetro de la ruta. Primero comprueba que el menú exista y que el usuario que realizó la petición a través del token JWT sea el propietario del menú o un administrador.
- **PATCH /menu/:id:** Ruta encargada de modificar los valores de un menú. Se comprueba que el usuario que realiza la petición sea el propietario o un administrador. Además, solo se permitirá modificar el título del menú.

- **DELETE /menu/:id:** Esta ruta elimina un menú de la base de datos. Primero comprueba que el usuario indicado en el token sea el propietario o un administrador y, si procede, elimina el menú y lo elimina también del listado de menús guardados del propietario.

4.1.5. Planificador

Se ha creado una ruta en el *backend* **POST /planner** para generar un menú en base a un sistema de búsqueda de recetas aleatorio. Esta ruta recibe el número de días, alergias e ingredientes a excluir y llama a una función interna que genera un plan aleatorio para el usuario.

Este algoritmo lo primero que hace es filtrar el total de recetas excluyendo aquellas que incluyan los alérgenos indicados y los ingredientes a excluir. A continuación, divide las recetas en tres conjuntos dependiendo del tipo de comida: entrante, plato principal o postre, y entra en un bucle que devuelve aleatoriamente una receta de cada tipo para cada día del menú, incluyendo estas recetas en una lista de recetas ya recomendadas para evitar repetirlas y así evitar la repetición de recetas.

4.1.6. Pruebas en el *backend*

Para las pruebas en el *backend* se han utilizado las librerías `mocha` [13] y `supertest` [26], las cuales permiten realizar pruebas sobre una aplicación de Express. Para las pruebas, como se comentó en el capítulo anterior, se usa una base de datos de pruebas almacenada en MongoDB Atlas. En total, se han realizado 91 tests, comprobando la funcionalidad y correcto funcionamiento tanto de las cuatro entidades: `ingredient`, `user`, `recipe`, `menu`, como de las rutas que las gestionan. En la Figura 4.2 se muestran algunas de las pruebas realizadas.

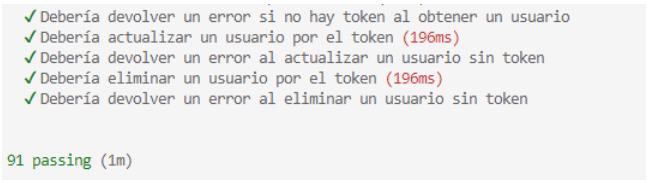


Figura 4.2: Algunas pruebas del backend

Integración continua

Además, se ha realizado la integración continua de las pruebas con GitHub Actions. Se ha definido una action que, cada vez que se realiza un `push` en el repositorio, instala las dependencias de desarrollo y ejecuta los tests. Para ello, se han definido como secretos del repositorio en GitHub la URL de conexión a la base de datos de prueba y el secreto de JWT. Esta configuración asegura que todas las modificaciones en el código sean probadas automáticamente, garantizando la estabilidad y funcionalidad del *backend*.

4.2. *Frontend*

Para el desarrollo del *frontend* se ha utilizado el framework de trabajo React [21] con el lenguaje TypeScript, debido a su amplio uso en la actualidad y las funcionalidades que ofrece.

Como biblioteca de componentes se ha usado Material-UI [17], que ofrece una gran cantidad de componentes responsivos para la creación de páginas web, y se integra muy bien con React. Los estilos de los componentes de Material-UI se han modificado con archivos CSS para que cumplan con la imagen de la aplicación web creada.

La estructura del proyecto ha sido creada al iniciar el proyecto con Vite [28] y React, destacan los archivos de configuración, el archivo `index.html`, que es el archivo inicial de la aplicación web donde se carga el resto del contenido React. El código principal se encuentra en la carpeta `src`. Dentro de `src` encontramos las siguientes subcarpetas:

- **assets:** Contiene todas las imágenes propias de la aplicación.
- **components:** Contiene los componentes reutilizables de la aplicación.
- **pages:** Contiene todas las diferentes páginas de la aplicación.
- **services:** Contiene el código de las peticiones a la API a través del endpoint.

Además, en la carpeta `src` encontramos el archivo `main.tsx`, en el que se definen las rutas de la aplicación web utilizando `react-router-dom` [22].

4.2.1. Imagen de PLAMESA

Se ha creado una paleta de colores personalizada para la página y se han diseñado las principales imágenes de la aplicación web. En concreto, se ha diseñado el logo, los grupos de comida, los tipos de comida y los alérgenos de la aplicación web, que se muestran a continuación, en las Figuras 4.3, 4.4 y 4.5.

El resto de iconos y figuras utilizadas han sido extraídas de la biblioteca de iconos que ofrece Material-UI.

Figura 4.3: Proceso diseño logo PLAMESA

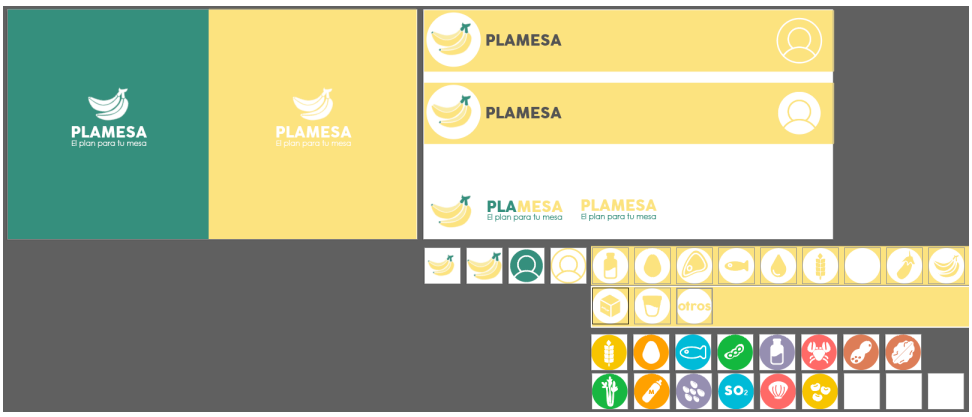


Figura 4.4: Proceso diseño alérgenos, grupos alimentos y barra de navegación PLAMESA

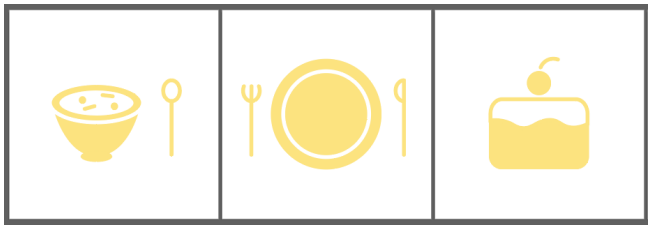


Figura 4.5: Diseño tipos de comida PLAMESA

4.2.2. Pantallas de inicio y navbar

La primera pantalla que aparecerá al abrir la aplicación web corresponde con la página de inicio (véase la Figura 4.6), que da la bienvenida al usuario. En esta, podemos observar la barra de navegación que nos acompañará en todas las pantallas y que contiene los enlaces a las principales secciones.

Para las opciones del usuario, existe un menú desplegable en el icono de usuario. Inicialmente, solo se muestra la opción para iniciar sesión, y si está iniciada, el icono cambia y se muestran las opciones del usuario.

Figura 4.6: Pantalla de inicio en pantalla grande

4.2.3. Pantallas de gestión de usuario

Inicio de sesión y registro de usuario

Las siguientes páginas a comentar corresponden a las páginas de inicio de sesión y registro de un nuevo usuario. En ambas se utiliza un formulario con los datos para autenticar a un usuario o para crearlo.

En el inicio de sesión (Figura 4.7) no se permite enviar el formulario hasta tener los campos completos y se muestra con ventanas de alerta el resultado: si la contraseña no corresponde, si el usuario no existe o si se ha iniciado sesión correctamente, todo manejando las respuestas del *backend*. En caso positivo, el token que devuelve se guarda en el local storage de la página para mantener la sesión a través de las páginas.

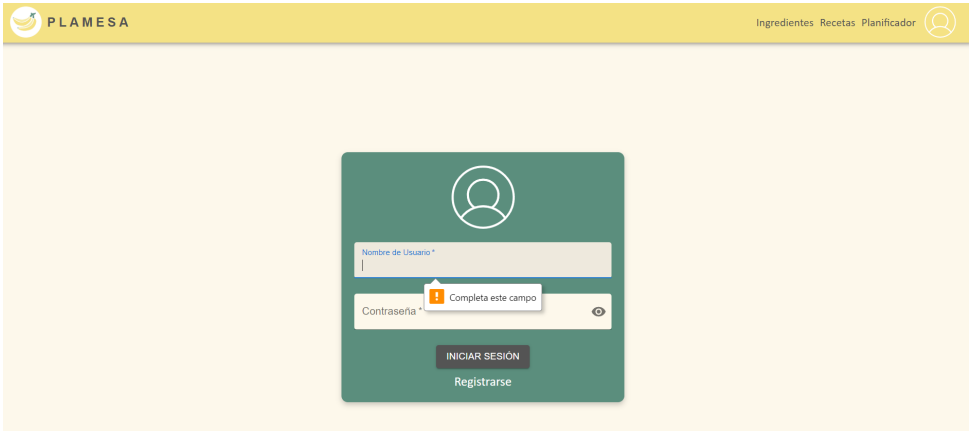


Figura 4.7: Pantalla de inicio de sesión

Para el caso de crear un usuario (véase la Figura 4.8) existe el botón de registrarse que redirige al formulario de registro, el cual obliga a introducir los datos básicos de identificación de un usuario, comprobando el formato de la contraseña y el email. También se manejan con ventanas de alerta las respuestas del *backend*, como la existencia del usuario o del email en la base de datos, o si todo ha ido bien, lo que significa que se habrá creado el usuario y se habrá iniciado sesión.

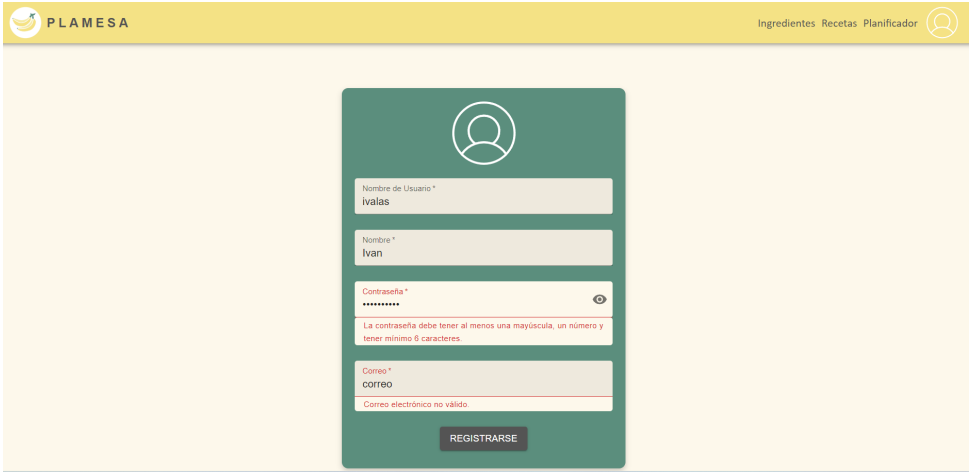


Figura 4.8: Pantalla de registro de usuario

Modificar usuario y cerrar sesión

Para manejar la información del usuario, existe la página de cuenta (Figura 4.9) donde se presenta en tres campos la información de un usuario en formato de campos de entrada. Esto permite al usuario modificar, agregar información además de visualizarla. Además, desde esta página se incluye al final el botón que permite borrar al usuario, y manda la petición al *backend*.

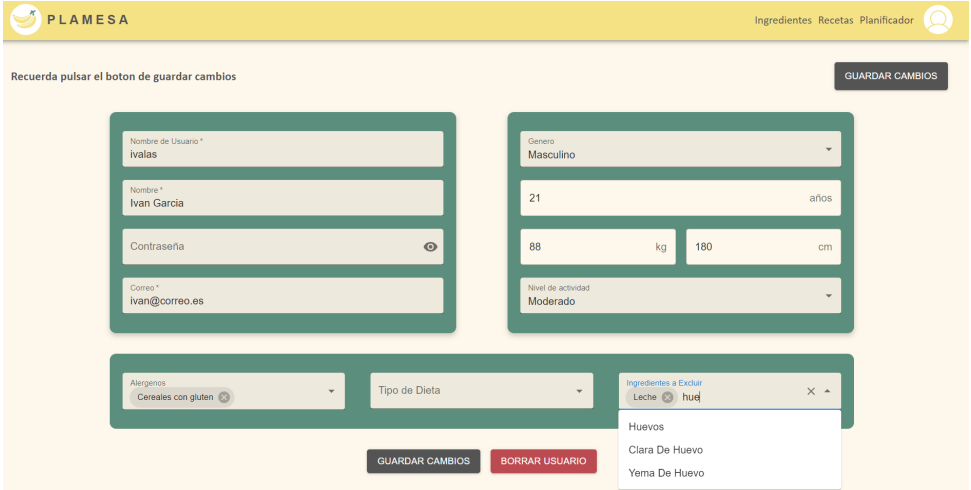


Figura 4.9: Pantalla de cuenta de usuario

Para poder acceder a esta página, en el icono de usuario, Figura 4.10, una vez iniciada la sesión, se muestran las opciones de `Cuenta`, para acceder a esta página, `mis menús` y `cerrar sesión`. Pulsando sobre esta última, se elimina el token del usuario, cerrando la sesión.

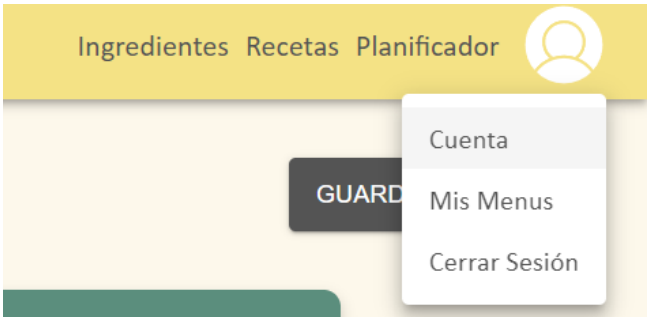


Figura 4.10: Menú desplegable de usuario

4.2.4. Pantallas de gestión de ingredientes

Visualización de ingredientes

En la sección de ingredientes, lo primero que veremos es la página en la que visualizamos todos los ingredientes ordenados alfabéticamente, Figura 4.11. Estos se muestran en formato de tarjetas, para lo cual se ha creado un componente específico. Esta tarjeta muestra la información principal y una imagen correspondiente al grupo de comida, además de un botón de `más info` que, al hacer clic, nos lleva a la página de detalles del ingrediente.

En la parte superior de la pantalla de ingredientes, encontramos un buscador por nombre de recetas y un desplegable que permite filtrar por grupo de comida, alérgenos a excluir, precio máximo y, si se ha iniciado sesión, por ingredientes creados por el usuario. En la parte superior derecha, se sitúa el botón que nos lleva a la página de crear un ingrediente.

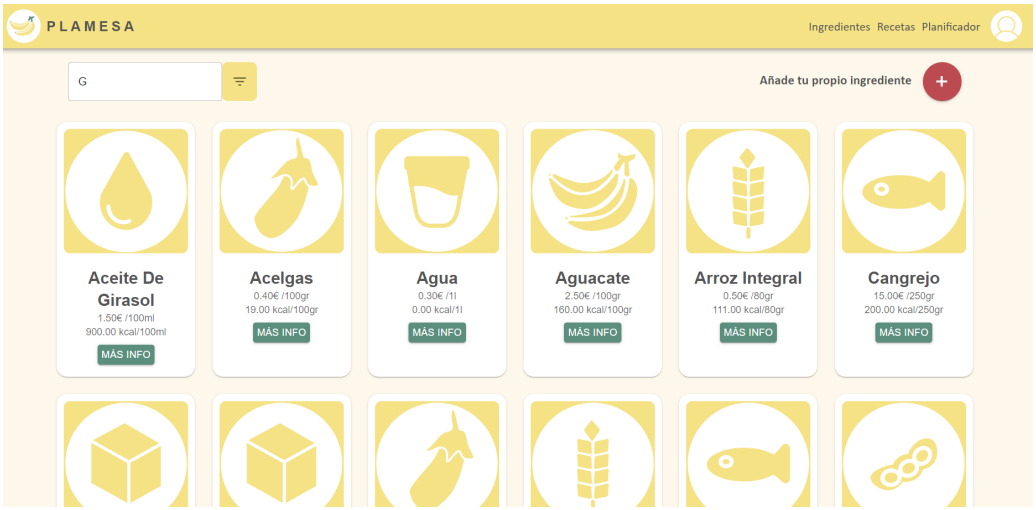


Figura 4.11: Pantalla de ingredientes

En la página de detalles de un ingrediente, Figura 4.12, se muestra toda la información del ingrediente. El campo que indica la cantidad del ingrediente es modificable, lo que permite recalcular los valores nutricionales y de costo al modificar la cantidad. Destaca también el botón situado en la esquina superior derecha que genera un PDF con la información del ingrediente para la cantidad indicada en el momento de generarlo. En caso de estar autenticado en la aplicación y ser propietario del ingrediente o tener el rol de administrador, se mostrarán dos botones en la parte baja de la página: uno lleva a la página de modificación del ingrediente y el otro permite borrar el ingrediente.

Figura 4.12: Detalles de un ingrediente

Creación y modificación de ingredientes

Para poder acceder a la pantalla de creación de ingredientes (véase la Figura 4.13), será necesario haber iniciado sesión; en caso contrario, se redirigirá al usuario a la pantalla de inicio de sesión. En la pantalla de creación de ingredientes se presentan tres secciones: la primera contiene la información básica y obligatoria del ingrediente; la segunda muestra todos los alérgenos en formato de casillas de verificación para marcar los alérgenos correspondientes; y en la tercera, situada en la parte inferior, se encuentran los campos de entrada para todos los nutrientes que un usuario puede introducir. Obligatoriamente, se deben incluir al menos los correspondientes a los macronutrientes que son de etiquetado obligatorio.

Figura 4.13: Pantalla de creación de ingredientes

La pantalla de modificación de ingredientes (Figura 4.14), tendrá el mismo formato que la de creación de ingredientes, pero mostrará inicialmente la información guardada del ingrediente, permitiendo que se modifique y se guarde posteriormente.

Figura 4.14: Pantalla de modificación de ingredientes

4.2.5. Pantallas de gestión de recetas

Visualización de recetas

En la sección de recetas, Figura 4.15, se pueden visualizar en formato de tarjetas todas las recetas de la aplicación web ordenadas alfabéticamente. Para el componente tarjeta, se muestra una imagen según el tipo de comida, el coste, las kcal y el tiempo de preparación de la receta, además de un botón para ver la receta en detalle.

En la parte superior, al igual que con los ingredientes, encontramos un buscador por nombre de recetas y un menú oculto para aplicar filtros sobre tipo de comida, alérgenos a excluir, precio máximo y, si se ha iniciado sesión, la opción para ver las recetas guardadas como favoritas y las recetas creadas. Además, al lado del icono de filtros, aparece un botón para ir a la pantalla de buscador por ingredientes. En la parte derecha podemos encontrar el botón que nos redirige a la pantalla para que el usuario cree su propia receta.

Figura 4.15: Vista de todas las recetas

Para la información detallada de cada receta (Figura 4.16), se ha creado otra pantalla que muestra toda la información de la receta, incluyendo los alérgenos, el coste estimado y la información nutricional calculada a partir de los ingredientes. Al igual que para los ingredientes, el campo que indica el número de comensales de la receta es modificable, lo que recalcula automáticamente los valores de la receta. Además, en el apartado de los ingredientes, si el usuario pincha sobre uno, se le redirigirá a la información del ingrediente concreto calculado para la cantidad de ese ingrediente en la receta.

Al igual que para los ingredientes, en la parte superior derecha se muestra un icono que genera la receta en formato PDF con los valores actuales. Además, en caso de que se haya iniciado sesión, al lado de este icono se muestra otro icono con forma de corazón para que el usuario pueda marcar esa receta como favorita o desmarcarla.

En la parte inferior, en caso de que el usuario sea el propietario de la receta o tenga rol de administrador, se mostrarán dos botones: uno para llevar a la página de modificación de la receta y otro para borrar la receta.

Figura 4.16: Detalles de una receta

Creación y modificación de recetas

En la pantalla de creación de recetas (véase la Figura 4.17), encontraremos cinco campos, siendo los tres primeros la información básica y obligatoria de la receta. En el primero tendremos varios campos de entrada para incluir la información básica de una receta, como nombre, tipo de comida, etc. El siguiente campo está destinado a la inclusión de los ingredientes de la receta. Para ello se ha dispuesto un campo de *autocomplete* que permite visualizar y buscar entre todos los ingredientes y seleccionar los que sean necesarios. Al ser seleccionados, se mostrarán debajo del campo de búsqueda con un nuevo campo de entrada para indicar la cantidad del ingrediente. El tercer campo corresponde al de entrada de las instrucciones de la receta que se incluirán una por línea en el campo descrito.

Los dos últimos campos corresponden a las opciones no obligatorias de comentarios y utensilios de cocina. Cuando el usuario termine de rellenar toda la información y pulse para crear la receta, esta información se mandará al *backend* y este devolverá la receta con todos los valores calculados.

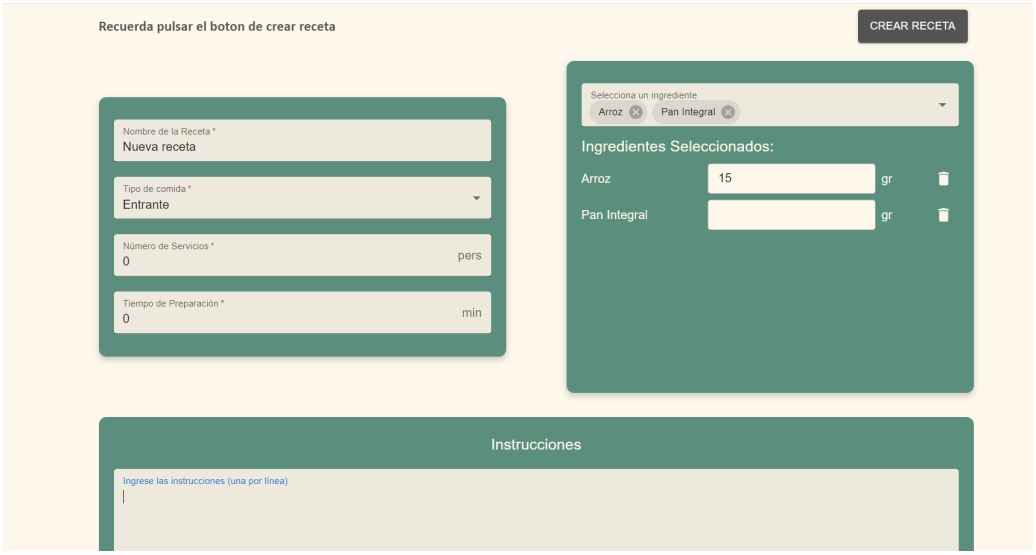


Figura 4.17: Pantalla de creación de recetas

En el caso de la pantalla de modificación de recetas (Figura 4.18), tendrá el mismo formato que la de creación de recetas, pero esta mostrará inicialmente la información guardada de la receta, permitiendo que se modifique y guarde posteriormente.

Figura 4.18: Pantalla de modificación de recetas

Recomendación de recetas por ingrediente

La pantalla de recomendación de recetas por ingredientes, Figura 4.19, se basa en un campo de entrada y búsqueda entre los ingredientes disponibles, permitiendo que el usuario seleccione los que desee. Luego, al hacer clic en el botón de búsqueda, se llama a la ruta del *backend* que devuelve 5 recetas que se muestran en formato de tarjetas.

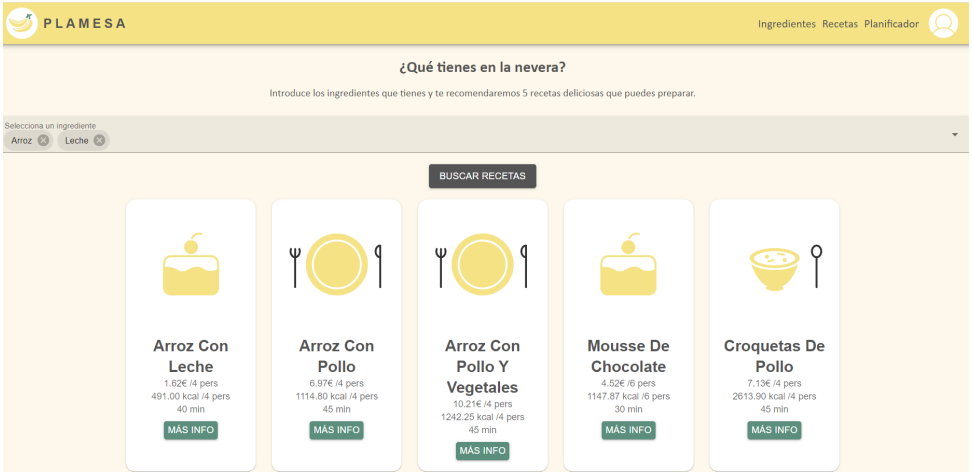


Figura 4.19: Pantalla de recomendación de recetas por ingredientes

4.2.6. Pantallas de gestión de menús

Visualización y generación de menús

La pantalla que se abre al pulsar sobre `Mis menús` (véase la Figura 4.20), nos muestra todos los menús guardados del usuario en formato de tarjetas, que presentan el título del menú, número de días, comensales, y precio y costo por día por persona. Además, hay un botón que nos lleva a los detalles del menú. En la parte superior de la pantalla,

disponemos de un buscador a la izquierda y, a la derecha, un botón que nos lleva al formulario de generación de menús.

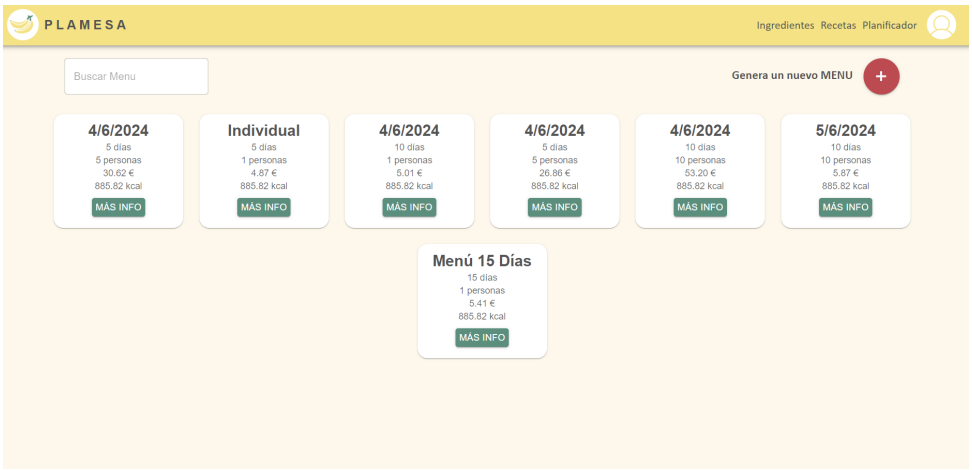


Figura 4.20: Menús guardados del usuario

El formulario de generación de un nuevo menú, Figura 4.21, se divide en tres campos. En el primero, se solicita el número de días y servicios del menú. Debajo de estos campos de entrada, existe un botón que, en caso de que el usuario haya iniciado sesión y registrado los datos del perfil en el apartado de cuenta, al pulsarlo, rellenará el resto de los campos con la información almacenada del usuario. Esta información es modificable por si se quiere añadir o cambiar algo para el menú a generar. Una vez completados los campos obligatorios, se podrá enviar el formulario, lo que nos llevará a otra página con la información del menú generado.

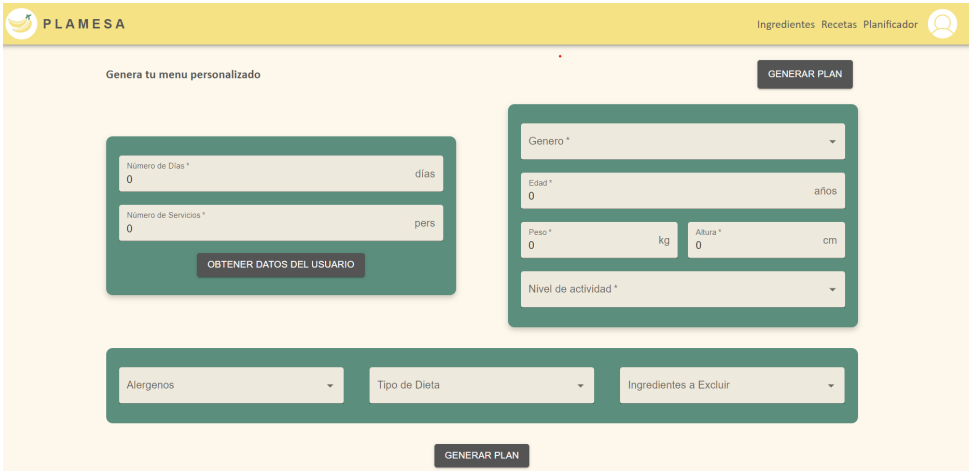


Figura 4.21: Formulario de creación menú

Visualización de un menú concreto

Una vez enviado el formulario de generación de menú, se muestra en columnas por día la selección de recetas generadas para el usuario concreto. Para cada columna se mostrarán 3 recetas, correspondientes a entrante, plato principal y postre, en formato de tarjetas con la información para el número de comensales descrito en el formulario de

generación. En caso de pulsar para ver la información de cada receta, se abrirá la pantalla correspondiente con los cálculos para el número de comensales que corresponda.

En la parte superior de la pantalla del menú (Figura 4.22), se puede visualizar el campo con el título del menú, que inicialmente corresponde a la fecha de generación y es modificable. En la parte derecha, encontramos 3 iconos: el primero para guardar el menú, el segundo genera un PDF con la información del menú, y el último redirige a la pantalla que muestra la lista de la compra para el menú correspondiente.

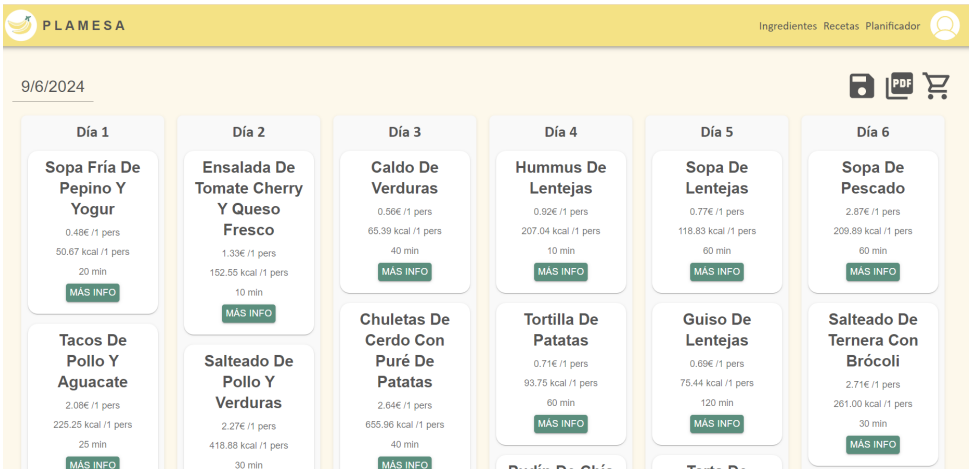


Figura 4.22: Parte superior del menú

En la parte inferior de cada día (Figura 4.23), se muestra la información media del menú por persona: costo, valores de energía y macronutrientes. En la parte inferior de la pantalla encontraremos un campo con toda la información del menú generado: número de días, servicios, kcal y costo medio por día, y el resto de la información utilizada en la generación del menú.

Debajo de este campo se muestra un botón para guardar el menú y, en caso de que se haya guardado el menú, al lado de este botón aparecerá otro que permite eliminar el menú correspondiente.

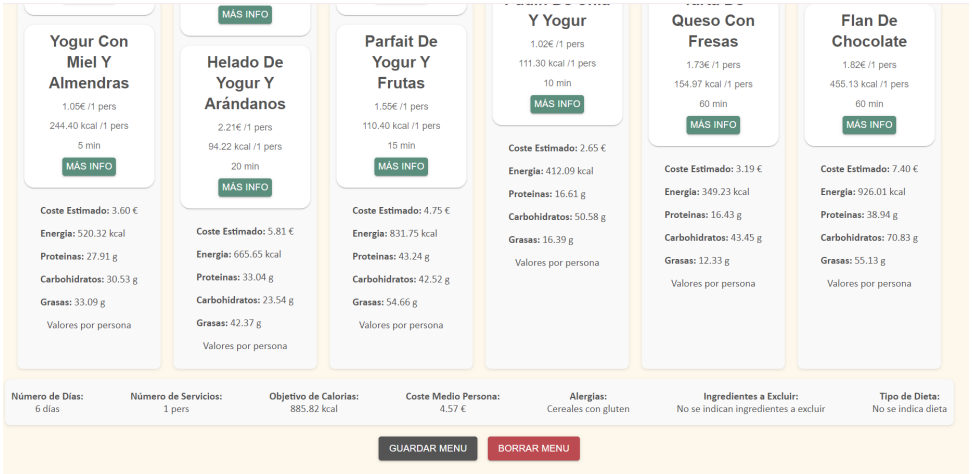


Figura 4.23: Parte inferior del menú

Lista de la compra

Si pinchamos sobre el icono de carrito de cualquier menú, se mostrará una pantalla (véase la Figura 4.24), organizada por semanas con la lista de los ingredientes y cantidades necesarios para poder realizar el menú correspondiente. Esta se muestra en formato de *checkbox*, lo que permite al usuario ir marcando los ingredientes.

Además, en la esquina superior derecha, se dispone de un botón para generar un PDF con la lista de la compra.

Figura 4.24: Lista de la compra

4.2.7. Usabilidad

Durante el desarrollo del *frontend* se ha buscado un diseño *responsive* en todo momento. Inicialmente, gracias al uso mayoritario de componentes de MaterialUI, que aportan un diseño responsivo, además de las modificaciones propias para asegurar que la aplicación web se pueda visualizar en dispositivos más pequeños. En las Figuras 4.25, 4.26 y 4.27 se pueden ver algunas de las pantallas en dispositivos pequeños.

Figura 4.25: Pantalla de receta movil

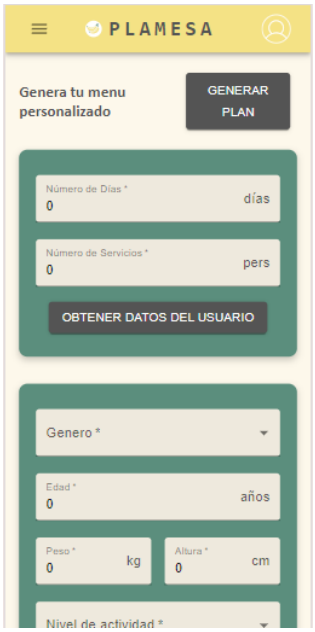


Figura 4.26: Pantalla de planificador movil

Figura 4.27: Pantalla de menú movil

4.3. Despliegue de PLAMESA

Para el despliegue de la aplicación web, al tener cliente y servidor independientes, se han utilizado diferentes plataformas.

En el *backend*, entre las diferentes posibles opciones como Railway [20], Heroku [4], Firebase [8], se ha decidido realizar el despliegue en la plataforma Render [3], que en su versión gratuita ofrece un despliegue continuo con cada `push` en el repositorio, resultando muy simple. Esto se logra al enlazar el repositorio de GitHub e indicar las variables de entorno, URI de conexión a la base de datos y el secreto de JWT. Una de las desventajas de la versión gratuita de esta plataforma es que tras periodos de 15 minutos de inactividad se suspende el despliegue y queda suspendido hasta que recibe una petición al *endpoint*, volviendo a levantar el servicio, lo que puede tardar alrededor de 1 minuto.

Para el *frontend*, entre las diferentes opciones como Vercel [27], Firebase [8], Heroku [4], se ha decidido usar Netlify [24] por su gran facilidad de uso y opciones en su versión gratuita. Esta también ofrece despliegue continuo a través de GitHub, por lo que con cada `push` al repositorio se realizará el despliegue, instalando de nuevo las dependencias y lanzando la aplicación con las variables de entorno definidas en la plataforma, correspondientes al *endpoint* de la API desplegada en Render.

Finalmente, queda desplegada la aplicación web completamente en las plataformas Netlify y Render. Debido a la suspensión tras inactividad del *backend*, cuando se navega por la aplicación web es probable que, al realizar una acción que tenga que interactuar con el *backend*, esta no funcione hasta pasado un minuto que se levante el *backend*, haciendo que la aplicación funcione al completo. Los enlaces de acceso son los siguientes:

APP WEB: `https://plamesa.netlify.app/`
API REST: `https://plamesa-backend.onrender.com`

Capítulo 5

Optimización del problema de planificación de menús

Para la optimización del problema de la planificación de menús, se ha realizado un análisis sobre el uso de un resolutor externo para la generación de planes alimenticios optimizados para el usuario, dejando el sistema preparado para implementarlo en el futuro. La aplicación web desarrollada será la encargada de gestionar todos los ingredientes, recetas, usuarios y menús, integrando una base de datos amplia y robusta. En cuanto a la generación de los menús, la aplicación obtendrá los parámetros de optimización y restricciones a tener en cuenta, junto con el listado de recetas, y se los pasará a un resolutor externo para que este genere la propuesta de planificación. El resolutor devolverá la planificación generada a la aplicación, que se encargará de gestionar y mostrar toda la información al usuario.

5.1. Othimi

En este proyecto, se ha optado por el uso de la aplicación web *Othimi*, desarrollada como proyecto de Trabajo de Fin de Grado [42], como resolutor del problema de planificación de menús. En este capítulo se plantea el problema, así como la definición teórica en *Othimi* para una futura integración, realizando así un estudio de viabilidad entre *Othimi* y *PLAMESA*, para posibles líneas futuras de este Trabajo de Fin de Grado.

La actual versión de Othimi es fruto de varios Trabajos de Fin de Grado desarrollados por estudiantes del Grado en Ingeniería Informática de la Universidad de La Laguna:

- Prodef: meta-modelado de problemas de optimización combinatoria (2020)
  Andrés Calimero García Pérez [52].
- Prodef-GUI: Interfaz gráfica para el modelado de problemas (2021)
  Daniel González Expósito [38].
- Prodef: Diseño, implementación y experimentación con nuevos resolutores (2022)
  Miguel Angel Ordoñez Morales [55].
- Prodef-Algorithm: Interfaz para el modelado de meta-heurísticas (2022)
  Daniel del Castillo de la Rosa [35].

- Prodef-SaaS: Despliegue y puesta en marcha de un servicio para la resolución de problemas de optimización (2022)

  Ángel Tornero Hernández [56].

- Prodef: Unificación e integración de módulos (2023)

  Alejandro Lugo Fumero [42].

Othimi es una herramienta que permite a los usuarios definir problemas de forma abstracta, para que luego un componente llamado resolutor sea capaz de recibir esta definición y computar una solución para el problema. Othimi es una solución web que se caracteriza por su interfaz gráfica, que permite el modelado de problemas y algoritmos sin necesidad de programar o realizar configuraciones complejas. Esta interfaz está desarrollada utilizando la biblioteca Blockly, que implementa una interfaz gráfica basada en bloques, permitiendo la construcción de problemas y algoritmos de manera intuitiva, similar a unir piezas de un rompecabezas.

Esta aplicación permite a los usuarios definir problemas de manera sencilla y eficaz mediante componentes gráficos. Se pueden definir todos los aspectos de un problema, incluyendo datos de entrada, variables, objetivos y restricciones. Podemos ver dicha interfaz en la Figura 5.1.

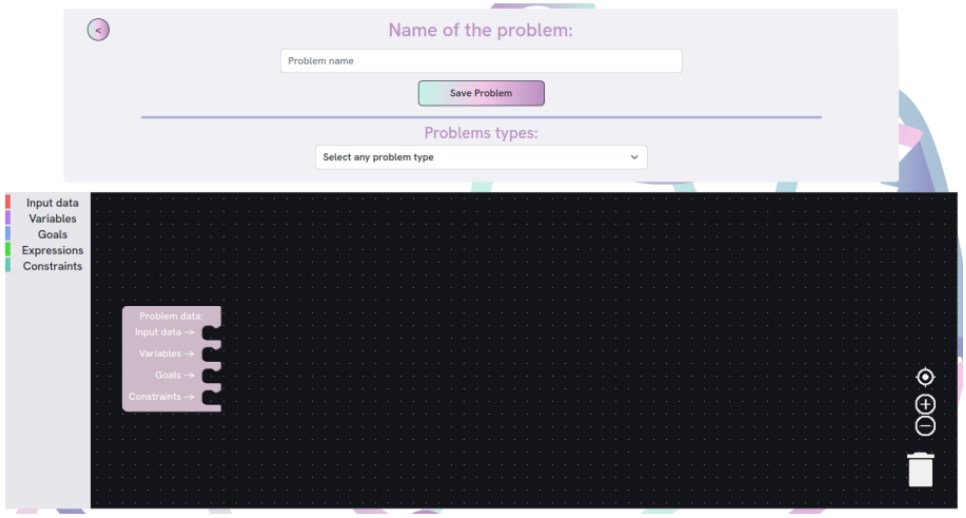


Figura 5.1: Interfaz de Othimi para la definición de problemas

Además, permite la definición de instancias a partir de parámetros en campos de entrada, la creación de tablas o la importación a través de archivos CSV. Véase en la Figura 5.2, la interfaz para la definición de instancias.

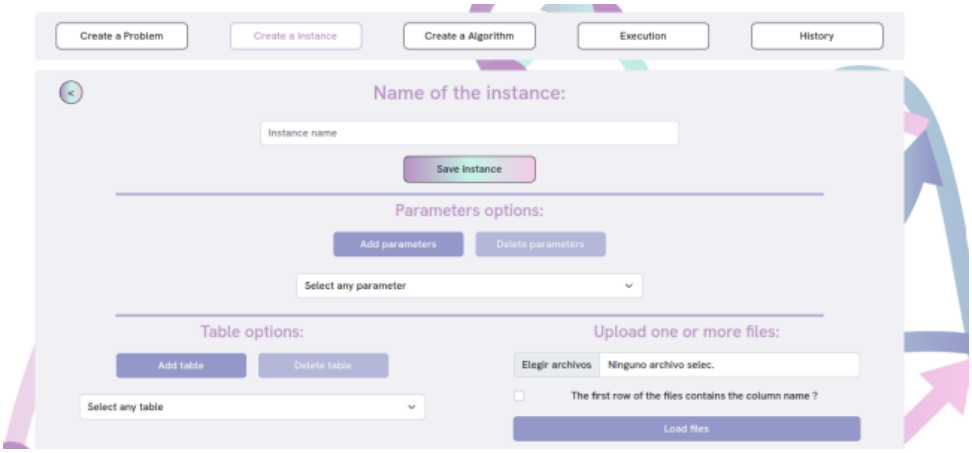


Figura 5.2: Interfaz de Othimi para la definición de instancias

5.2. Formulación multiobjetivo del problema de planificación del menú

Actualmente, la aplicación web PLAMESA ofrece un generador de menús simple, en el que se filtran las recetas según las restricciones del usuario en cuanto a alérgenos e ingredientes a excluir. Se genera una propuesta que prioriza la variedad de las recetas, evitando la repetición consecutiva de las mismas. Con esto, se pretende acercar a la formulación multiobjetivo que presenta SCHOOLTHY, en la que se busca maximizar la variedad de alimentos y minimizar los costos.

Es por ello que la formulación multiobjetivo planteada para el problema de planificación de menús en este proyecto se basa en la definición utilizada en SCHOOLTHY. En esta formulación, se consideran dos funciones objetivo a optimizar. La primera es el coste del plan de alimentación, que debe ser minimizado. La segunda es el grado de repetición de platos y grupos de alimentos, que también debe ser minimizado, ya que una dieta más diversa es más saludable.

Además, la formulación considera como requisitos fundamentales la calidad nutricional del plan alimentario en términos de aportes energéticos y macronutrientes (grasas, hidratos de carbono y proteínas).

5.2.1. Codificación de la solución

Una solución puede representarse mediante un vector de valores enteros, donde cada valor representa un entrante, un plato principal o un postre para un número determinado de días. Este vector estará formado por tripletas de platos y tendrá tantas tripletas como el número de días del menú. Cada tripleta contendrá un entrante, un plato principal y un postre, véase la Figura 5.3. Este vector se denota como I, cuya longitud es $|I| = 3{\cdot}n$, siendo n el número de días del plan de comidas. Cada valor entero en el vector corresponde al identificador de un plato específico en la base de datos.

Para ello, es necesario crear un nuevo atributo para el modelo de receta que incluya un ID único, diferente del actual ID de MongoDB. Este ID debe ser único y servirá para identificar cada receta. Los IDs serán asignados de forma consecutiva entre los diferentes

modelos. Por lo tanto, para los entrantes se asignarán desde 1 hasta l_{st}, para los platos principales desde $l_{st}+1$ hasta $l_{st}+l_{mc}$, y para los postres desde $l_{st}+l_{mc}+1$ hasta $l_{st}+l_{mc}+l_{ds}$.

El tamaño del espacio de soluciones S está dado por la Ecuación (5.1):

$$|S| = (l_{st} \cdot l_{mc} \cdot l_{ds})^n \tag{5.1}$$

Donde l_{st}, l_{mc} y l_{ds} representan el número de entrantes, platos principales y postres en la base de datos, respectivamente.

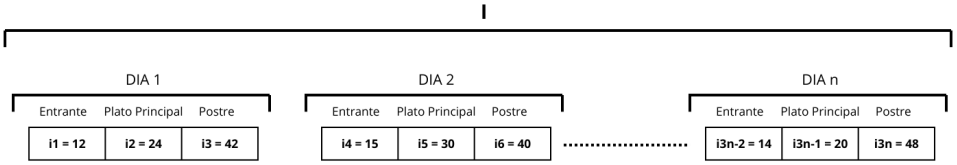


Figura 5.3: Ejemplo de codificación de la solución

5.2.2. Restricciones

Cada plan de comidas debe cumplir con una cantidad recomendada de energía y macronutrientes definida por H:

$$\forall h \in H : n \cdot r_{min,h} \cdot r_h \leq in(I,h) \leq n \cdot r_{max,h} \cdot r_h \tag{5.2}$$

Donde $in(I,h)$ es la ingesta total del nutriente h en el plan de comidas I, r_h es la ingesta diaria del nutriente h durante el almuerzo, y $r_{min,h}$ y $r_{max,h}$ son las cantidades mínimas y máximas permitidas del nutriente h para n días.

5.2.3. Funciones objetivo

Las funciones objetivo son las siguientes:

Minimización de la repetición de platos y grupos de alimentos

La función que modela el grado de repetición de platos y grupos de alimentos se calcula como:

$$R = \sum_{j=1}^{n} \left(v_{MCj} + \frac{p_{st}}{d_{stj}} + \frac{p_{mc}}{d_{mcj}} + \frac{p_{ds}}{d_{dsj}} + v_{FGj} \right) \tag{5.3}$$

Donde v_{MCj} representa la compatibilidad entre los platos de un día j, p_{st}, p_{mc} y p_{ds} son constantes de penalización por repetición de platos, definidas en la Tabla 5.1, d_{stj}, d_{mcj} y d_{dsj} son los días desde la última aparición de un plato en particular, y v_{FGj} es el valor de penalización por repetición de grupos de alimentos en los cinco días anteriores a j.

Los valores de penalización para la repetición de grupos de alimentos y platos específicos en un plan de comidas se establecieron mediante un estudio preliminar realizado por SCHOOLTHY. Las penalizaciones están diseñadas para garantizar la calidad y variedad del plan, evitando repeticiones frecuentes. Estas penalizaciones afectan la función objetivo, permitiendo al decisor ajustar sus preferencias. Las penalizaciones se aplican tanto a la

repetición de grupos de alimentos específicos (p1–p13) como a la repetición del mismo grupo en días consecutivos (p14–p18) y a la repetición de platos específicos (p19–p21).

Penalización	**Descripción**	**Valor**
$p1$	Penalización por repetir grupo de alimentos "otros"	0.1
$p2$	Penalización por repetir grupo de alimentos "carne"	3
$p3$	Penalización por repetir grupo de alimentos "cereal"	0.3
$p4$	Penalización por repetir grupo de alimentos "fruta"	0.1
$p5$	Penalización por repetir grupo de alimentos "lácteos"	0.3
$p6$	Penalización por repetir grupo de alimentos "huevos"	0.3
$p7$	Penalización por repetir grupo de alimentos "legumbres"	0.3
$p8$	Penalización por repetir grupo de alimentos "mariscos"	2
$p9$	Penalización por repetir grupo de alimentos "pasta"	1.5
$p10$	Penalización por repetir grupo de alimentos "pescado"	0.5
$p11$	Penalización por repetir grupo de alimentos "vegetales"	0.1
$p12$	Penalización por repetir grupo de alimentos "grasas"	3
$p13$	Penalización por repetir grupo de alimentos "azucares"	4
$p14$	Penalización por repetir el mismo grupo de alimentos un día anterior	3
$p15$	Penalización por repetir el mismo grupo de alimentos dos días anteriores	2.5
$p16$	Penalización por repetir el mismo grupo de alimentos tres días anteriores	1.8
$p17$	Penalización por repetir el mismo grupo de alimentos cuatro días anteriores	1
$p18$	Penalización por repetir el mismo grupo de alimentos cinco días anteriores	0.2
$p19 = pst$	Penalización por repetir un entrante	8
$p20 = pmc$	Penalización por repetir un plato principal	10
$p21 = pds$	Penalización por repetir un postre	2

Tabla 5.1: Tipos de penalizaciones definidas para calcular la primera función objetivo.

La penalización por repetición de un grupo de alimentos g en los tres platos de un día j se calcula como:

$$v_{MCj} = \sum_{g=1}^{|G|} x_g \cdot p_g \tag{5.4}$$

Donde x_g indica el número de repeticiones del grupo de alimentos g y p_g es el valor de penalización correspondiente.

La penalización por repetición de un grupo de alimentos en los últimos T días se calcula como:

$$v_{FGj} = \sum_{d=1}^{\min(j-1,T)} \left(\sum_{g=1}^{|G|} y_{gd} \cdot p_g + z_d \cdot p_{|G|+d} \right) \tag{5.5}$$

Donde $y_{gd} = 1$ si el grupo de alimentos g se repitió el día $j - d$, $z_d = 1$ si cualquier grupo de alimentos se repitió el día $j - d$, la ventana de tiempo $T = 7$, dado que los planes de

alimentación suelen diseñarse para una semana, 7 días y $|G|$ es el número de grupos de alimentos

Minimización del coste total

El coste total de un plan de comidas se calcula como:

$$C = \sum_{j=1}^{n} (c_{stj} + c_{mcj} + c_{dsj}) \tag{5.6}$$

Donde c_{stj}, c_{mcj} y c_{dsj} son los costes del entrante, plato principal y postre, respectivamente, para el día j. El coste de un plato específico se obtiene de la suma de los costes de los ingredientes necesarios para prepararlo.

5.3. Definición del problema en Othimi

Para definir este problema en Othimi, es necesario especificar los datos del problema utilizando la interfaz gráfica basada en bloques. En esta interfaz, se definen los datos de entrada, variables, objetivos y restricciones para cada problema.

Datos de entrada

Los datos de entrada que se pueden definir en Othimi incluyen parámetros y tablas:

- Parámetro con el número de días del plan de comidas (n).
- Tablas con los platos de la base de datos, divididas en tres categorías según el tipo de comida: entrante, plato principal y postre, con la información detallada de cada plato.
- Tabla con los requerimientos nutricionales (H), que incluyen los nutrientes y sus cantidades mínimas ($r_{min,h}$) y máximas ($r_{max,h}$) recomendadas, así como la ingesta diaria (r_h) para cada nutriente. Donde $H = \{\text{Energía, proteínas, carbohidratos, grasas totales}\}$.
- Tabla con las penalizaciones (p_i), definidas en la Tabla 5.1.
- Parámetro con la ventana de tiempo ($T = 7$), que representa el número de días a considerar para evaluar la repetición de grupos de alimentos.

Variables

Othimi tiene cuatro tipos de variables: número, lista, lista permutada y matriz.

Para este problema específico, se definirá una lista en función del vector de solución I de longitud $3 \cdot n$, donde cada entero representa el ID de un plato en la base de datos.

Objetivos

- **Minimización de la repetición de platos y grupos de alimentos**: Minimizar la función R, que mide la repetición de platos y grupos de alimentos en el plan de comidas. Definir la expresión R, como se indica en la ecuación 5.3, utilizando los bloques de expresión que proporciona Othimi.

- **Minimización del coste total**: Minimizar la función C, que calcula el coste total del plan de comidas. Definir la expresión C, como se indica en la ecuación 5.6, utilizando los bloques de expresión que proporciona Othimi.

Restricciones

- **Requerimientos nutricionales**: Cada plan de comidas debe cumplir con las cantidades mínimas y máximas de nutrientes recomendadas para n días:

$$\forall h \in H : n \cdot r_{min,h} \cdot r_h \leq in(I,h) \leq n \cdot r_{max,h} \cdot r_h \tag{5.7}$$

5.4. Definición de las instancias del problema

Para la definición de las instancias de un problema en Othimi, existe una sección en la aplicación donde se pueden definir las instancias como parámetros o tablas. Estas pueden ser ingresadas manualmente en la interfaz o importadas en formato CSV. Para definir correctamente el problema, se requieren las siguientes instancias:

- Parámetro con el número de días (n) para el plan, obtenido de PLAMESA tras completar el formulario de generación de un menú.
- Parámetro con la ventana de tiempo (T), con un valor de 7 días.
- Tabla con las constantes de penalizaciones específicas para la repetición de grupos de alimentos y platos, así como para la compatibilidad entre platos, definidas en la Tabla 5.1.
- Tabla con los requerimientos nutricionales. Cada fila representará un nutriente y cada columna indicará la cantidad diaria del nutriente, así como los valores mínimos y máximos. Los valores correspondientes para cada menú se obtendrán de PLAMESA una vez completado el formulario de generación de un menú. El orden de definición de los nutrientes será: Energía, proteínas, carbohidratos y grasas totales.
- Base de datos de platos. Existirán tres archivos según el tipo de comida: entrante, plato principal y postre, los cuales se extraerán de la aplicación web PLAMESA. Durante esta extracción, se realizará un filtrado de los platos según las restricciones de alérgenos, dieta e ingredientes a excluir del menú, indicados en el generador de menús. Por ejemplo, si se indica una alergia al gluten, se eliminarán todas las recetas que contengan gluten antes de enviarlas al resolvedor. El formato de los archivos será el siguiente:

  <ID>, <nombre>, <cantidad>, <coste>, <cantidad de nutriente>, <grupos de alimento>

  - **ID**: identificador del plato.
  - **Nombre**: nombre del plato.
  - **Cantidad**: cantidad de servicios del plato.
  - **Coste**: coste estimado del plato.

- **Cantidad de nutrientes**: cantidad de cada nutriente presente en el plato en el siguiente orden: {Energía, proteínas, carbohidratos, grasas totales}.
- **Grupos de alimento**: grupo de alimentos de los ingredientes que componen el plato. Se indicará con números enteros el número de repeticiones de cada grupo de comida según los grupos de alimentos de sus ingredientes, definidos en el siguiente orden: {otros, carne, cereal, fruta, lácteos, huevos, legumbres, mariscos, pasta, pescado, vegetales, grasas, azúcares}.

5.5. Definición del algoritmo de optimización

Para la planificación de menús, se puede utilizar un Algoritmo Evolutivo (EA) debido a su eficacia en la optimización de problemas complejos con grandes espacios de búsqueda. Los EAs son algoritmos de optimización inspirados en el proceso de selección natural y se utilizan para encontrar soluciones aproximadas a problemas de optimización.

Para resolver la formulación multiobjetivo del problema de optimización de menús, que tiene como funciones objetivo minimizar costes y maximizar la variedad, SCHOOLTHY utiliza un algoritmo Non-dominated Sorting Genetic Algorithm II (NSGA-II). Sin embargo, este tipo de algoritmo multiobjetivo no es posible de definir en Othimi. Por ello, en el caso del problema que definimos en este capítulo, se deberá utilizar un algoritmo evolutivo simple con inicialización aleatoria. En este algoritmo, se definirá una ponderación para cada una de las funciones objetivo. Como no se desea priorizar una función sobre la otra, se utilizará una ponderación del 50 % para cada una de las funciones objetivo.

Capítulo 6

Conclusiones y líneas futuras

Durante el desarrollo de este Trabajo de Fin de Grado, se ha llevado a cabo la creación de una aplicación web *fullstack*, implementando un *backend* robusto que ofrece una API REST y un *frontend* intuitivo y completo. La aplicación web final permite la gestión de cuatro entidades relacionadas entre sí.

Como resultado de este trabajo, se ha logrado una aplicación web que sirve como fuente de datos sobre ingredientes y recetas para cualquier persona. La aplicación ofrece una variedad de ingredientes con información básica y nutricional detallada, además de una base de datos con una amplia variedad de recetas nutritivas, también con información detallada. Adicionalmente, a diferencia de muchas aplicaciones actuales en el mercado, esta aplicación proporciona información estimada sobre los precios de las recetas e ingredientes. Todo esto hace que la aplicación sea útil para una amplia gama de usuarios, desde el usuario común que desee obtener información sobre ingredientes y recetas, hasta cocineros o encargados de la planificación y generación de menús y dietas que busquen ideas de recetas e ingredientes para incluir.

Además, en la versión actual de PLAMESA, se ofrece una primera aproximación a la generación de menús a partir de restricciones. Esto permite que cualquier usuario pueda obtener un menú para un número determinado de días, basado en restricciones de alérgenos e ingredientes a excluir, evitando la repetición consecutiva de recetas.

Al analizar los objetivos y requisitos definidos inicialmente en comparación con los resultados obtenidos, se puede observar que los objetivos principales se han logrado. Sin embargo, no se han cumplido todos los requisitos específicos. Aunque la mayoría se han alcanzado, algunos, como la utilización de algoritmos de optimización para la generación de menús y la modificación de platos en los menús propuestos, no se han podido implementar. Estos últimos requisitos no se han alcanzado en parte debido a su complejidad y a la falta de tiempo para su implementación. Entre otras problemáticas, la parte del *frontend* llevó más tiempo del esperado, principalmente debido a que se utilizó un *framework* con el que no había trabajado previamente, lo que requirió un tiempo adicional para dominarlo.

Personalmente, este proyecto me ha ayudado a tener un mayor entendimiento sobre el desarrollo completo desde cero de una aplicación web *fullstack*. He aprendido mucho sobre el desarrollo de APIs REST, con sus correspondientes problemáticas para la autenticación de usuarios, realización de tests, interrelación entre las entidades y sus consecuentes modificaciones. Y sobre todo, he aprendido en el desarrollo del *frontend*

con un *framework* de desarrollo bastante potente con el que no había trabajado antes, como es React. Además de todo lo aprendido en relación con las tecnologías y desarrollo de aplicaciones web, he conseguido un entendimiento mayor sobre la alimentación y la nutrición.

De esta manera, existen varias mejoras y líneas futuras de desarrollo que se pueden realizar. Algunas de estas son:

- Integración de PLAMESA con un resolvedor externo como Othimi para la optimización del problema de planificación de menús.
- Incorporación de peticiones a APIs externas con información actualizada sobre los costes de ingredientes, para mantener la información de PLAMESA lo más actualizada posible.
- Carga de más ingredientes y recetas en la base de datos para mejorar la recomendación y generación de menús.
- Inserción de imágenes propias para cada ingrediente y receta.
- Realización de tests en el frontend.

Capítulo 7

Summary and Conclusions

During the development of this final degree project, a fullstack web application was created, implementing a robust backend that provides a REST API and a complete and intuitive frontend. The final web application allows for the management of four interrelated entities.

As a result of this project, a web application has been developed that serves as a data source on ingredients and recipes for anyone. The application offers a variety of ingredients with basic and detailed nutritional information, as well as a database with a wide variety of nutritious recipes, also with detailed information. Additionally, unlike many current applications on the market, this application provides estimated information on the prices of recipes and ingredients. All of this makes the application useful for a wide range of users, from the common user who wants to obtain information about ingredients and recipes to cooks or those in charge of planning and generating menus and diets who seek ideas for recipes and ingredients to include.

Furthermore, the current version of PLAMESA offers a first approach to menu generation based on restrictions. This allows any user to obtain a menu for a determined number of days, based on allergen restrictions and ingredients to exclude, avoiding consecutive repetition of recipes.

By analyzing the initially defined objectives and requirements compared to the results obtained, it can be observed that the main objectives have been achieved. However, not all specific requirements have been met. Although most have been achieved, some, such as the use of optimization algorithms for menu generation and the modification of dishes in the proposed menus, have not been implemented. These latter requirements have not been met partly due to their complexity and the lack of time for their implementation. Among other issues, the frontend part took longer than expected, mainly because a framework was used with which I had not previously worked, requiring additional time to master it.

Personally, this project has helped me gain a greater understanding of the complete development from scratch of a fullstack web application. I have learned a lot about REST API development, with its corresponding issues for user authentication, testing, interrelation between entities, and their consequent modifications. And above all, I have learned in the development of the frontend with a quite powerful development framework with which I had not worked before, such as React. In addition to everything learned

in relation to web technologies and application development, I have gained a greater understanding of food and nutrition.

Thus, there are several improvements and future lines of development that can be made. Some of these are:

- Integration of PLAMESA with an external solver like Othimi for the optimization of the menu planning problem.
- Incorporation of requests to external APIs with updated information on ingredient costs, to keep PLAMESA's information as up-to-date as possible.
- Loading more ingredients and recipes into the database to improve recommendation and menu generation.
- Adding custom images for each ingredient and recipe.
- Conducting tests on the frontend.

Capítulo 8

Presupuesto

En este capítulo se expone el presupuesto para la ejecución del trabajo realizado, así como para el mantenimiento y despliegue de la aplicación web desarrollada. Para estimar el precio por hora de trabajo, se ha consultado una plataforma de comparación de sueldos [23], y se ha determinado un precio de 11€ la hora para un desarrollador junior en España.

A continuación, se expone la tabla con los costes del trabajo realizado.

Concepto	**Horas**	**Coste Total (€)**
Análisis del estado del arte y herramientas	40	440
Desarrollo backend	140	1540
Desarrollo frontend	160	1760
Diseño de marca e imagen corporativa	20	220
Total	**360**	**3960**

Tabla 8.1: Costes del trabajo realizado

En esta otra tabla se muestran los costes para el mantenimiento y despliegue de la aplicación web durante el periodo de un año.

Concepto	**Coste Total (€)**
Despliegue backend (Render)	228
Despliegue frontend (Netlify)	228
Despliegue base de datos (MongoDB Atlas)	684
Compra de dominio	20
Total	**1160**

Tabla 8.2: Costes de mantenimiento y despliegue de la aplicación web

El total de los costes de desarrollo y mantenimiento de la aplicación web durante un año sería de 5120€.

Apéndice A

Script de carga de datos

```
import path from 'path';
import axios from 'axios';
import { promises as fs } from 'fs';

const USER_NAME = process.env.USER_NAME;
const USER_PASSWORD = process.env.USER_PASSWORD;
const directoryPath = path.join(__dirname, '../data');
const apiEndpointLogin = 'http://localhost:3000/login';
const apiEndpointIngredient = 'http://localhost:3000/ingredient';

(async () => {
  if (USER_NAME && USER_PASSWORD) {
    try {
      const token = await logIn(USER_NAME, USER_PASSWORD, apiEndpointLogin);
      console.log('Login exitoso. Token:', token);

      await readAndSendJsonIngredients(directoryPath, apiEndpointIngredient, token);
    } catch (error) {
      console.error('Error:', error);
    }
  }
})();

export async function logIn(username: string, password: string, apiUrl: string): Promise<
    string> {
  try {
    const response = await axios.post(apiUrl, {
      username: username,
      password: password
    });
    const token = response.data.token;
    return token;
  } catch (error) {
    console.error('Error during login:', error);
    throw error;
  }
}

export async function readAndSendJsonIngredients(directory: string, apiUrl: string, token
    : string) {
  try {
    const filePath = path.join(directory, 'ingredients.json');
```

```
    const fileContent = await fs.readFile(filePath, 'utf-8');
    const jsonData: IngredientInterface[] = JSON.parse(fileContent);

    for (let i = 0; i < jsonData.length; i++) {
      const ingredient = jsonData[i];

      try {
        const response = await axios.post(
          apiUrl, // URL de la solicitud
          ingredient,   // El cuerpo de la solicitud
          {
            headers: {
              'Authorization': `Bearer ${token}`, // Cabecera de autorización
              'Content-Type': 'application/json', // Enviando datos en formato JSON
            },
          }
        );

        const outputFilePath = path.join(directory, 'ingredientsOut.txt');
        const responseData = response.data;
        const id = responseData._id;
        const name = responseData.name;
        const unit = responseData.unit;

        await fs.appendFile(outputFilePath, `${name} - ${id} - ${unit}\n`);
      } catch (error) {
        const errorFilePath = path.join(directory, 'errors.txt');
        const errorMessage = `Name: ${ingredient.name}, Status: ${error.response?.status
    || 'N/A'}, Error: ${error.response?.data.message || error.message}\n`;
        await fs.appendFile(errorFilePath, errorMessage);
      }
    }
  } catch (error) {
    console.error('Error:', error);
  }
}
```

Bibliografía

[1] Aesan - agencia española de seguridad alimentaria y nutrición. Accedido: 31-Abr-2024. [En línea]. Disponible: https://www.aesan.gob.es/AECOSAN/web/seguridad_alimentaria/subdetalle/nutricional.htm.

[2] CICS transaction server for z/OS 6.x. Accedido: 8-May-2024. [En línea]. Disponible: https://www.ibm.com/docs/es/cics-ts/6.x?topic=cics-json-web-token-jwt.

[3] Cloud application hosting for developers | render. Accedido: 2-Jun-2024. [En línea]. Disponible: https://dashboard.render.com/.

[4] Cloud application platform | heroku. Accedido: 3-Jun-2024. [En línea]. Disponible: https://www.heroku.com/.

[5] Ecuación de harris-benedict. Accedido: 28-Abr-2024. [En línea]. Disponible: https://es.wikipedia.org/w/index.php?title=Ecuaci%C3%B3n_de_Harris-Benedict&oldid=156568641/.

[6] Estilos de vida saludable - distribución diaria. Accedido: 28-Abr-2024. [En línea]. Disponible: https://estilosdevidasaludable.sanidad.gob.es/alimentacionSaludable/queSabemos/enLaPractica/distribuir/diario/home.htm.

[7] Express - infraestructura de aplicaciones web node.js. Accedido: 30-Mar-2024. [En línea]. Disponible: https://expressjs.com/es/.

[8] Firebase | google's mobile and web app development platform. Accedido: 30-May-2024. [En línea]. Disponible: https://firebase.google.com/?hl=es-419.

[9] Introducing ChatGPT. Accedido: 20-May-2024. [En línea]. Disponible: https://openai.com/index/chatgpt/.

[10] jsonwebtoken. Accedido: 10-Mar-2024. [En línea]. Disponible: https://www.npmjs.com/package/jsonwebtoken.

[11] Los 14 alérgenos principales. Accedido: 5-Abr-2024. [En línea]. Disponible: https://curso-alergenos.com/lecciones/los-14-alergenos-principales/.

[12] Micronutrients: Types, functions, benefits and more. Accedido: 30-Feb-2024. [En línea]. Disponible: https://www.healthline.com/nutrition/micronutrients.

[13] Mocha - the fun, simple, flexible JavaScript test framework. Accedido: 15-Mar-2024. [En línea]. Disponible: https://mochajs.org/.

[14] MongoDB atlas database | multi-cloud database service | MongoDB. Accedido: 15-Abr-2024. [En línea]. Disponible: https://www.mongodb.com/products/platform/atlas-database.

[15] MongoDB: The developer data platform | MongoDB. Accedido: 10-Abr-2024. [En línea]. Disponible: https://www.mongodb.com/.

[16] Mongoose ODM v8.4.1. Accedido: 12-Abr-2024. [En línea]. Disponible: https://mongoosejs.com/.

[17] MUI: The react component library you always wanted. Accedido: 3-May-2024. [En línea]. Disponible: https://mui.com/.

[18] MySQL. Accedido: 15-Mar-2024. [En línea]. Disponible: https://www.mysql.com/.

[19] Node.js — run JavaScript everywhere. Accedido: 30-Mar-2024. [En línea]. Disponible: https://nodejs.org/en.

[20] Railway. Accedido: 3-Jun-2024. [En línea]. Disponible: https://railway.app.

[21] React. Accedido: 8-Mar-2024. [En línea]. Disponible: https://es.react.dev/.

[22] React router dom. Accedido: 15-Mar-2024. [En línea]. Disponible: https://reactrouter.com/en/main.

[23] Salario para programador junior en españa - salario medio. Accedido: 14-Jun-2024. [En línea]. Disponible: https://es.talent.com/salary.

[24] Scale & ship faster with a composable web architecture | netlify. Accedido: 3-Jun-2024. [En línea]. Disponible: https://www.netlify.com/.

[25] SQLite home page. Accedido: 15-Mar-2024. [En línea]. Disponible: https://sqlite.org/.

[26] supertest. Accedido: 15-Mar-2024. [En línea]. Disponible: https://www.npmjs.com/package/supertest.

[27] Vercel. Accedido: 30-May-2024. [En línea]. Disponible: https://vercel.com.

[28] Vite. Accedido: 8-Mar-2024. [En línea]. Disponible: https://vitejs.dev.

[29] Johan Aberg. Dealing with malnutrition: A meal planning system for elderly. pages 1–7. Accedido: 22-Feb-2024. [En línea]. Disponible: https://www.researchgate.net/publication/221250510_Dealing_with_Malnutrition_A_Meal_Planning_System_for_Elderly.

[30] Central Lechera Asturiana. Nutrientes: qué son, tipos y funciones | central lechera asturiana. Accedido: 8-Mar-2024. [En línea]. Disponible: https://www.centrallecheraasturiana.es/nutricionysalud/nutricion/nutrientes/.

[31] BEDCA. Base de Datos BEDCA. Accedido: 28-Feb-2024. [En línea]. Disponible: https://www.bedca.net/bdpub/.

[32] Antonio Blanco. Trasteando la API del mercadona. Accedido: 5-May-2024. [En línea]. Disponible: https://medium.com/@ablancodev/trasteando-la-api-del-mercadona-cff067abc002.

[33] Consum. El planificador de menús de consum. Accedido: 24-Feb-2024. [En línea]. Disponible: https://planificador.consum.es/planificador/estilo_vida=4/tipo_dieta=2.

[34] Red Internacional de Sistemas de Datos Alimentarios. INFOODS: Tablas y bases de datos. Accedido: 9-Mar-2024. [En línea]. Disponible: https://www.fao.org/infoods/infoods/tablas-y-bases-de-datos/es/.

[35] Daniel del Castillo de la Rosa. Prodef-algorithm: Interfaz para el modelado de metaheurísticas. Technical report, Universidad de La Laguna, 2022.

[36] Ekilu. Recetas de cocina y vida saludable. Accedido: 16-Feb-2024. [En línea]. Disponible: https://ekilu.com/es.

[37] EuroFIR. FoodEXplorer » EuroFIR. Accedido: 8-Mar-2024. [En línea]. Disponible: https://www.eurofir.org/our-tools/foodexplorer/.

[38] Daniel González Expósito. Prodef-gui: Interfaz gráfica para el modelado de problemas. Technical report, Universidad de La Laguna, 2021.

[39] Open Food Facts. Base de datos open food facts. Accedido: 9-Mar-2024. [En línea]. Disponible: https://es.openfoodfacts.org.

[40] Tanausú Falcón Casanova. Desarrollo de una aplicación full-stack para la geolocalización de productos alimenticios. Accedido: 15-Feb-2024. [En línea]. Disponible: https://riull.ull.es/xmlui/handle/915/33723.

[41] FAO. Inicio | organización de las naciones unidas para la alimentación y la agricultura. Accedido: 9-Mar-2024. [En línea]. Disponible: https://www.fao.org/home/es.

[42] Alejandro Lugo Fumero. Prodef: Unificación e integración de módulos. Technical report, Universidad de La Laguna, 2023.

[43] Ismael Galancho. Macronutrientes: qué son y cuál es el porcentaje recomendado. Accedido: 8-Mar-2024. [En línea]. Disponible: https://ismaelgalancho.com/macronutrientes-que-son-y-cual-es-el-porcentaje-recomendado-en-tu-dieta/.

[44] PostgreSQL Global Development Group. PostgreSQL. Accedido: 15-Mar-2024. [En línea]. Disponible: https://www.postgresql.org/.

[45] Harmonic-Soft. Recetas calendario - aplicaciones en google play. Accedido: 17-Feb-2024. [En línea]. Disponible: https://play.google.com/store/apps/details?id=me.lwwd.mealplan&hl=es.

[46] Eat This Much Inc. Eat this much, your personal diet assistant. Accedido: 18-Mar-2024. [En línea]. Disponible: https://www.eatthismuch.com.

[47] Ashvini Kale and Nisha Auti. Automated menu planning algorithm for children: Food recommendation by dietary management system using ID3 for indian food database. 50:197–202. Accedido: 23-Feb-2024. [En línea]. Disponible: https://www.sciencedirect.com/science/article/pii/S1877050915005712.

[48] Aderonke Kayode. Development of a web based intelligent dietetic system. Accedido: 28-Feb-2024. [En línea]. Disponible: https://www.academia.edu/109880606/Development_of_a_Web_Based_Intelligent_Dietetic_System.

[49] Mealime. Mealime - meal planning app for healthy eating. Accedido: 16-Feb-2024. [En línea]. Disponible: https://www.mealime.com/.

[50] Menutech. Menutech app. Accedido: 25-Feb-2024. [En línea]. Disponible: https://app.menutech.com/dashboard.

[51] MyRealFood. MyRealFood: recetas y planes nutricionales. Accedido: 18-Mar-2024. [En línea]. Disponible: https://www.myrealfood.app/es.

[52] Andrés Calimero García Pérez. Prodef: Meta-modelado de problemas de optimización combinatoria. Technical report, Universidad de La Laguna, 2020.

[53] Eduardo Segredo, Gara Miranda, Juan Manuel Ramos, Coromoto León, and Casiano Rodríguez-León. Schoolthy: Automatic menu planner for healthy and balanced school meals. *IEEE Access*, 8:113200–113218, 2020. Accedido: 22-Feb-2024. [En línea]. Disponible: https://ieeexplore.ieee.org/document/9119403.

[54] Spoonacular. Free meal planner, food tracker, and recipe saver. Accedido: 24-Feb-2024. [En línea]. Disponible: https://spoonacular.com/.

[55] Miguel Ángel Ordoñez Morales. Prodef: Diseño, implementación y experimentación con nuevos resolutores. Technical report, Universidad de La Laguna, 2022.

[56] Ángel Tornero Hernández. Prodef-saas: Despliegue y puesta en marcha de un servicio para la resolución de problemas de optimización. Technical report, Universidad de La Laguna, 2022.