TRABAJOS FIN DE GRADO

Universidad de Jaén

Escuela Politécnica Superior de Jaén

Generación inteligente de menús nutricionales personalizados

Estudiante: Carlos Jiménez García

Grado: Ingeniería Informática

Tutores del TFG: Manuel José Barranco García
Raciel Yera Toledo

Departamento: Informática

Fecha: 07/07/2026

Agradecimientos

Quiero agradecer el trabajo y esfuerzo de todos los profesores que me han dado clase y me han prestado su ayuda durante estos años, especialmente a los dos tutores de mi tfg, Manuel José Barranco García y Raciel Yera Toledo.

Índice

Lista de figuras

Lista de tablas

Lista de fórmulas

Capítulo 1 - Introducción

Este capítulo presenta la introducción del Trabajo de Fin de Grado. En primer lugar, se expone la motivación que ha llevado a la realización del proyecto y los objetivos que se pretende alcanzar. Se incluye también la metodología de trabajo empleada y la planificación temporal y económica del proyecto.

1. 1 Introducción al proyecto

En el contexto del e-health (uso de tecnologías digitales para mejorar, gestionar o brindar servicios de salud), la generación automáticas de menús está emergiendo actualmente como un escenario de personalización relevante, teniendo en cuenta su estrecha relación con las enfermedades no transmisibles y los conceptos de nutrición personalizada. La nutrición personalizada se ha definido como el asesoramiento sobre alimentación saludable, adaptado a la medida de un individuo en función de los datos genéticos y, alternativamente, del estado de salud personal, el estilo de vida, la ingesta de nutrientes y los datos fenotípicos. Debido al coste de la gestión de los datos genéticos, en los últimos años han aumentado intensamente los esfuerzos de investigación sobre la gestión de estos datos alternativos y se han propuesto varias soluciones computacionales con este objetivo.

Para comprender la evolución de estas tecnologías, es fundamental remontarse a los orígenes de los sistemas de recomendación, los cuales nacieron a principios de la década de 1990 como una solución técnica al creciente problema de la sobrecarga de información. El primer sistema documentado en la literatura fue Tapestry, un gestor de correo electrónico que acuñó el término "filtrado colaborativo". Poco después, el sistema GroupLens automatizó esta arquitectura para recomendar artículos de noticias en la red basándose en las calificaciones de usuarios con perfiles similares.

Sin embargo, en el ámbito específico de la alimentación, los esfuerzos de personalización matemática son aún más antiguos. Históricamente, Balintfy [9] es considerado el pionero al desarrollar los primeros algoritmos computacionales de planificación de menús mediante programación lineal, con

el objetivo de optimizar los costos hospitalarios asegurando simultáneamente el cumplimiento estricto de las restricciones nutricionales de los pacientes.

El problema de la planificación de menús se viene estudiando desde hace más de 50 años. Sin embargo, recientemente ha sido y sigue siendo un problema de investigación abierto y muy activo, centrado en añadir capacidades de personalización a los marcos de generación de menús.

De manera particular, el análisis de la literatura permite identificar dos principales vertientes de trabajos:

1- Trabajos enfocados en generar menús nutricionalmente apropiados, pero con poca o ninguna capacidad de personalización.
2- Trabajos enfocados en generar menús/alimentos centrados en las preferencias previas del usuario, pero sin considerar el valor nutricional de los alimentos recomendados.

Existen muy pocos modelos computacionales incorporando tanto las características de la dimensión 1, junto con las de la dimensión 2.

1.2 Objetivos

El objetivo principal del presente trabajo es la creación de un software que facilite la generación de menús personalizados y nutricionalmente apropiados para el usuario, y que tenga entre sus funcionalidades principales:

- Registro de los menús consumidos por parte de los usuarios.
- Carga de perfiles de usuarios con sus registros de menús, provenientes de datasets externos al sistema.
- Creación, modificación y eliminación de alimentos en el sistema, incluyendo su información nutricional asociada.
- Generación de menús personalizados a los usuarios implementando en el sistema desarrollado el modelo de generación de menús personalizados y nutricionalmente apropiados.

1.3 Planificación

Teniendo en cuenta la naturaleza del proyecto he optado por utilizar la metodología ágil Scrum para la realización del proyecto desarrollando el mismo a través de distintas iteraciones. Esta metodología nos aporta transparencia, inspección y adaptación durante el transcurso del proyecto.

Formalización y el Manifiesto Ágil

El nacimiento oficial de Scrum como un marco de trabajo estructurado para la industria del software tuvo lugar en 1995. Sutherland y Schwaber unieron sus experiencias y presentaron conjuntamente el documento *"SCRUM Software Development Process"* en la conferencia anual OOPSLA (Object-Oriented Programming, Systems, Languages & Applications) en Austin, Texas. Esta presentación formalizó las prácticas, roles y eventos que hoy conforman el núcleo del marco de trabajo.

Posteriormente, en el año 2001, Schwaber y Sutherland formaron parte del grupo de diecisiete pioneros de la industria que se reunieron en Snowbird, Utah, para redactar el **Manifiesto por el Desarrollo Ágil de Software** [1]. Aunque este manifiesto sentó las bases filosóficas para múltiples metodologías (como eXtreme Programming y Kanban), Scrum se consolidó rápidamente como la implementación práctica más extendida y estructurada de los valores ágiles.

Estandarización: La Guía de Scrum

A medida que la adopción de Scrum se globalizó, surgió la necesidad de proteger la integridad del marco de trabajo y evitar interpretaciones erróneas. Para resolver esto, Ken Schwaber y Jeff Sutherland publicaron en 2010 la primera versión de *"La Guía de Scrum" (The Scrum Guide)*. Este documento, que es revisado y actualizado periódicamente por sus creadores (con su última revisión principal en 2020), establece la definición oficial del marco de trabajo, desvinculándolo exclusivamente del software para permitir su aplicación en la resolución de cualquier tipo de problema complejo [2][Figura 1.1].

Roles

Scrum establece tres roles fundamentales:

- **Product Owner (propietario del producto):** se encarga de definir las funcionalidades del producto y establecer sus prioridades.
- **Scrum Master:** actúa como facilitador del proceso, ayudando al equipo a avanzar y eliminando los impedimentos que puedan surgir.
- **Equipo de Desarrollo:** está formado por los profesionales responsables de construir y entregar el producto.

[Figura 1.2]

Eventos

Scrum contempla varios eventos importantes: la planificación del sprint, donde se organiza el trabajo a realizar; el scrum diario, una breve reunión para coordinar actividades y planificar las siguientes 24 horas; la revisión del sprint, en la que se muestra el trabajo completado; y la retrospectiva del sprint, que sirve para analizar qué funcionó bien y qué se puede mejorar[Figura 1.3].

Artefactos

Los principales artefactos de Scrum son:

- **Product Backlog:** una lista priorizada con todos los requisitos del producto.
- **Sprint Backlog:** conjunto de elementos seleccionados para el sprint junto con el plan para desarrollarlos.
- **Incremento:** resultado de sumar todos los elementos completados durante el sprint actual y los anteriores.

[Figura 1.2]

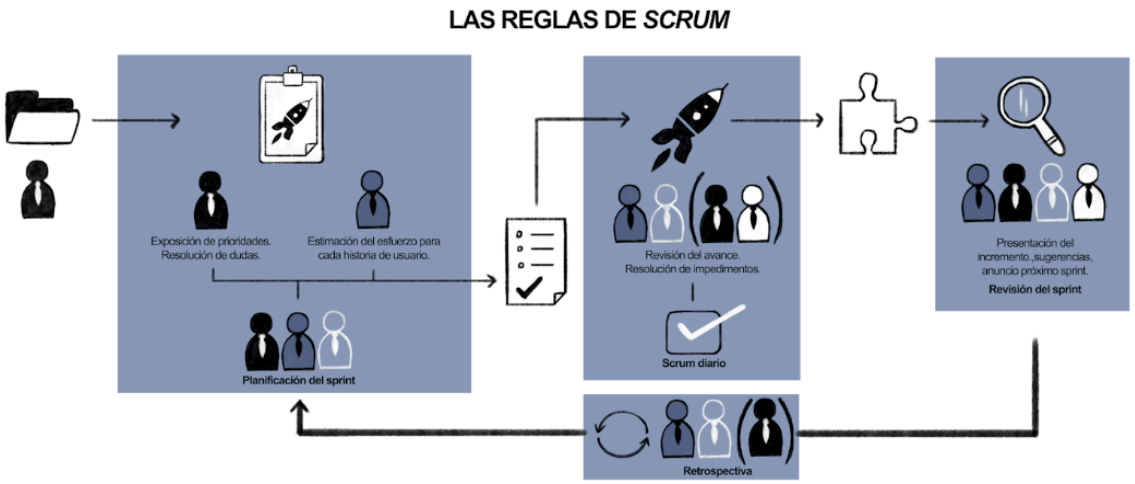


Figura 1.1: Las reglas de Scrum

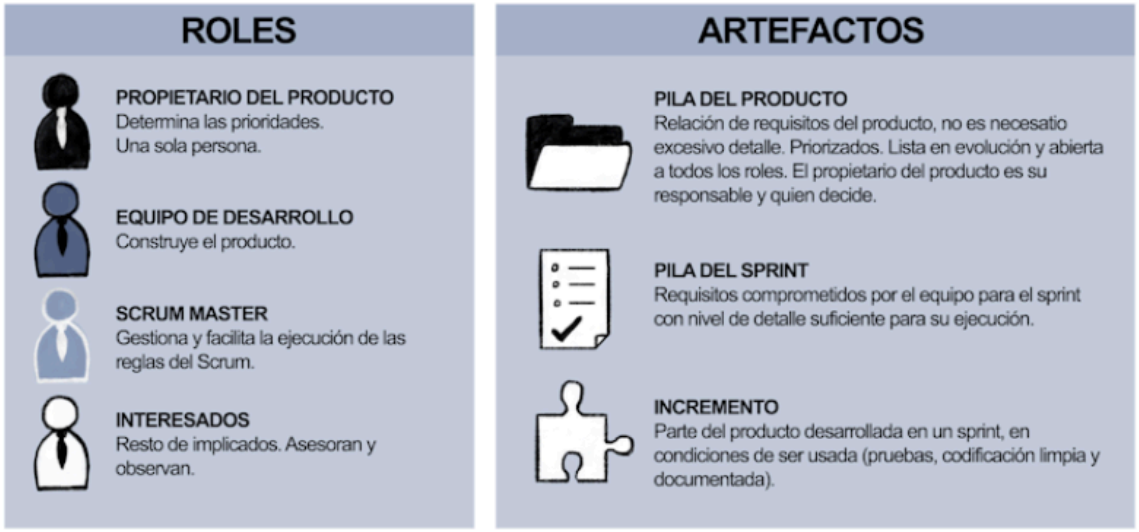


Figura 1.2: Roles y artefactos en Scrum

Figura 1.3: Eventos en Scrum

A continuación, se detalla la planificación temporal y las principales actividades realizadas en cada sprint.

- **Fase de Configuración (5 de enero - 1 de febrero).** El proyecto comenzó con el establecimiento del ecosistema de trabajo y la elección de las tecnologías fundamentales. A partir de ahí, se desarrolló un prototipo inicial que actuó como cimiento para el desarrollo futuro del proyecto. También se llevó a cabo la especificación inicial de los requisitos funcionales y no funcionales.
- **Iteración 1: Análisis y Diseño (1 de febrero- 1 de marzo).** Tras diseñar la interfaz del sistema, se puso en marcha una implementación inicial con el fin de contrastar el diseño y recoger impresiones valiosas. Basándose en estos resultados, se procedió a refinar el *backlog*, reorganizando las tareas y prioridades de cara a los siguientes ciclos de desarrollo.
- **Iteración 2: Desarrollo y Prototipado (1 de marzo- 4 de mayo).** Se dio inicio al desarrollo del backend, estableciendo la conexión con el frontend y construyendo la infraestructura básica para el módulo de generación de menús. Además se realizó una integración inicial entre frontend y backend para validar que los componentes del sistema funcionarán correctamente juntos.
- **Iteración 3: Integración y Finalización (4 de mayo- 30 de junio).** En la última iteración se completó la integración de todos los componentes del sistema. Se llevaron a cabo pruebas exhaustivas de usabilidad para identificar posibles errores y mejorar la experiencia de usuario. Durante este período también se elaboró toda la documentación necesaria, incluyendo manuales técnicos y funcionales, para dejar constancia del desarrollo realizado. Finalmente, se realizaron ajustes finales y se preparó el proyecto para la entrega final

PRESUPUESTO

Para calcular el presupuesto se utilizará la fórmula de la cuota anual (ver Ecuación 1.1)[3]

$$CA = \frac{VA - VR}{VU} \quad (1.1)$$

Donde:

- **CA:** Cuota anual.
- **VA:** Valor anual.
- **VR:** Valor residual.
- **VU:** Vida útil en años.

La amortización se calcula atendiendo al método de amortización lineal [4]:

$$A = \frac{CA - TU}{12} \quad (1.2)$$

Donde:

- **A:** Amortización.
- **CA:** Cuota Anual.
- **TU:** Tiempo de uso en meses.

Costes de hardware [Tabla 1.1]

El hardware necesario para el desarrollo de la aplicación es exclusivamente un ordenador personal que permita la instalación del software requerido para el desarrollo del código. La vida útil en ordenadores portátiles esperada es de entre 3 y 5 años, aunque esta puede variar.

Se considera que el valor residual del portátil después de los 5 años es de 0 €.

Esto resulta en una cuota anual de amortización de 205,57 €, al tener el equipo un uso de 6 meses, su amortización es de 102,78 €.

Hardware	**Precio de adquisición**	**Número de equipos**	**Amortización**
Dell Inc. Latitude 5450	1027’83 €	1	102,78 €

Tabla 1.1 Costes de Hardware

Costes de software

El software necesario para la realización del proyecto es el siguiente:

- Sistema operativo: licencias de los sistemas operativos de los ordenadores en el proyecto.
- Herramientas de diseño.
- Herramientas para el desarrollo de la documentación.
- Herramientas de desarrollo de software.

En este proyecto, el coste de software debido al uso de herramientas y sistemas gratuitos de código abierto y servicios en línea, a excepción de la licencia de Windows 11 cuyo coste es de 145 € . Windows 11, es la versión más reciente del sistema operativo de Microsoft, el cuál, ofrece un entorno robusto y seguro, lo que permite realizar tareas de desarrollo [5].
Para el diseño y creación de diagramas, se han utilizado herramientas gratuitas que facilitan la creación de diagramas UML y de flujo de forma eficiente como la aplicación para la creación de presentaciones de google o mermaid.

Asimismo, Google docs, una plataforma en línea para la creación de documentos. Para el desarrollo del software se ha utilizado IntelligentJ de JetBrains, este es un proveedor de software de vanguardia especializado en la creación de herramientas de desarrollo inteligentes [6]. Gracias a la disponibilidad de estas herramientas, el proyecto los gatos software del proyecto son de 145 € en total.

Costes del personal

El equipo de desarrollo se compone de dos miembros, un analista del proyecto, y un desarrollador de software [Tabla 1.2]. La jornada laboral será de 20 horas semanales, ajustándose así a la Ley del Estatuto de los Trabajadores. Las tablas salariales publicadas en el Boletín Oficial del Estado [BOE-A-2009-5688] son las siguientes, calculadas de manera proporcional (al 50 % de una jornada completa) en base a las retribuciones mínimas por convenio:

Perfil Profesional	Categoría de Referencia (BOE)	Salario Anual (Jornada Completa)	Salario Anual (20 h/sem)	Salario Mensual (14 pagas, 20 h/sem)
Analista del Proyecto	Área de Programación y Proyectos (BOE-A-2009-5688)	28.000 €	**14.000 €**	1.000 €
Desarrollador de Software	Programador Senior (BOE-A-2025-18060)	24.780 €	**12.390 €**	885 €

Tabla 1.2 Costes de personal

En consecuencia, el **coste salarial bruto anual** directo para este equipo de desarrollo asciende a **26.390 €**.

1.4 Estructura de la memoria

El presente proyecto está dividido en varios capítulos, cada uno de los cuales cubre un aspecto fundamental del desarrollo del “Sistema de generación inteligente de menús nutricionales personalizados”. A continuación, se ofrece una descripción detallada de cada capítulo y su contenido:

En el **capítulo 2,** se abordará el panorama actual de los sistemas de recomendación y parte de la historia de los mismos, la cuál nos ha llevado a este punto. Así mismo, se presentarán los tipos de sistemas de recomendación existentes.

El **capítulo 3** se dedicará al análisis y diseño del sistema. Aquí se establecerán los requisitos técnicos y funcionales del sistema de generación inteligente de menús nutricionales personalizados, así como la planificación de su arquitectura. Este capítulo incluirá el diseño de la base de datos, la estructura del frontend y el backend, y la interacción entre los distintos componentes del sistema.

El **capítulo 4** estará enfocado al desarrollo del proyecto, donde se detalla la implementación completa del Sistema de generación inteligente de menús nutricionales personalizados. En esta sección se describen las tecnologías utilizadas, las librerías, frameworks y los lenguajes de programación empleados. Además, se incluirán las pruebas realizadas para garantizar la funcionalidad del sistema, y cómo se integraron estas con el flujo de desarrollo.

Finalmente, el **capítulo 5** estará destinado a las conclusiones y trabajos futuros. En esta parte se resumen los resultados obtenidos, se reflexiona sobre los retos superados durante el desarrollo del proyecto, y se exponen posibles mejoras o ampliaciones que podrían realizarse en el futuro para optimizar el rendimiento o ampliar las funcionalidades del sistema. El documento incluirá además el anexo 1 donde se detallarán las configuraciones necesarias para poner en marcha el sistema, facilitando así a los usuarios y desarrolladores interesados una guía para la correcta instalación y despliegue de la plataforma; el anexo 2 que incluirá el manual de usuario.

Capítulo 2 - Contexto y marco teórico

La finalidad de de este capítulo es que el lector pueda adquirir una visión general acerca de lo que son los sistemas de recomendación y el panorama actual e histórico de los sistemas de recomendación de alimentos. [7]

2.1 Contexto histórico de los sistemas de recomendación

El desarrollo de los sistemas de recomendación comenzó a principios de la década de 1990 como una respuesta al creciente volumen de información disponible, que dificultaba a los usuarios encontrar el contenido más relevante. Uno de los primeros antecedentes recogidos en la literatura fue Tapestry [11], un sistema de gestión de correo electrónico que introdujo el concepto de filtrado colaborativo. Poco después, GroupLens [12] amplió esta idea al implementar un sistema capaz de recomendar artículos de noticias de forma automática utilizando las valoraciones de usuarios con intereses y comportamientos similares.

En el campo de la alimentación, sin embargo, la aplicación de métodos computacionales para personalizar recomendaciones tiene un origen anterior. El trabajo de Balintfy [9] marcó un hito al proponer los primeros algoritmos de planificación de menús basados en programación lineal. Estos algoritmos estaban orientados a reducir los costes en hospitales sin comprometer el cumplimiento de los requerimientos nutricionales y las restricciones dietéticas de los pacientes.

2.2 Tipos de Sistemas de Recomendación Existentes

Atendiendo a la clasificación estandarizada en la teoría computacional [13], [14], podemos dividir los sistemas de recomendación en las siguientes arquitecturas principales, según cómo enlazan y llevan a cabo el procesamiento de la información:

- **Filtrado Colaborativo (*Collaborative Filtering*):** Sugiere ítems basándose en las preferencias pasadas y la similitud entre usuarios. Funciona bajo la premisa de que si el Usuario A y el Usuario B tienen un historial de consumo similar, el sistema recomendará al Usuario A los alimentos o recetas que fueron evaluados positivamente por el Usuario B.
- **Basados en Contenido (*Content-Based*):** Analizan las características físicas o semánticas (metadatos) de los ítems que le han gustado al usuario en el pasado para sugerir ítems análogos. En el contexto alimentario, el algoritmo pondera variables intrínsecas como los ingredientes, el perfil de macronutrientes o el método de cocción.
- **Basados en Conocimiento (*Knowledge-Based*):** Utilizan reglas y conocimiento explícito sobre el dominio y las necesidades del usuario, sin depender de un gran volumen de historial de consumo previo. Son fundamentales en entornos clínicos, donde una recomendación no debe basarse solo en gustos, sino en restricciones médicas inquebrantables (por ejemplo, el bloqueo de sodio para usuarios hipertensos).
- **Híbridos (*Hybrid Systems*):** Combinan dos o más de los métodos anteriores para mitigar sus debilidades individuales (como el problema del "arranque en frío" en usuarios nuevos). En el diseño de sistemas de recomendación dietética, el enfoque híbrido es el estándar de la industria, ya que permite equilibrar las preferencias de sabor (colaborativo/contenido) con la rigurosidad clínica impuesta por los expertos médicos (conocimiento).

2.3 Panorama actual y paradigma “Food Computing”

Hoy en día, la investigación ha evolucionado hacia un paradigma interdisciplinario conocido como *Food Computing*. Según la literatura reciente [15], el panorama actual se caracteriza por la adquisición y el análisis de datos heterogéneos provenientes de múltiples fuentes como sensores biométricos, historiales clínicos, bases de datos de composición de alimentos y patrones de comportamiento para la percepción y recomendación integral de la dieta.

El mayor desafío científico de los sistemas de recomendación de alimentos modernos radica en la transición de un modelo predictivo a uno prescriptivo. Como señalan las investigaciones sobre nutrición algorítmica [16], los sistemas actuales deben ir más allá de simplemente sugerir "lo que al usuario le gustaría comer" para priorizar "lo que el usuario debería comer" en pro de su salud, minimizando al mismo tiempo la frustración o el rechazo del plato. Esta convergencia tecnológica, impulsada por la integración de algoritmos de inteligencia artificial (AI) y dispositivos del Internet de las Cosas (IoT), constituye la base operativa de la Nutrición Personalizada actual.

En este sentido, la capacidad de procesar grandes volúmenes de información mediante algoritmos de aprendizaje automático (*Machine Learning*) y procesamiento de lenguaje natural ha ampliado radicalmente las fronteras del *Food Computing*. Las aplicaciones contemporáneas no solo analizan el perfil nutricional crudo, sino que son capaces de interpretar el contexto semántico de las recetas, extrayendo relaciones complejas entre métodos de cocción, sinergias de ingredientes y su impacto metabólico. Esto permite una granularidad sin precedentes en la caracterización de los alimentos, superando ampliamente las limitaciones metodológicas de las bases de datos estáticas tradicionales.

Adicionalmente, un aspecto crucial en este nuevo paradigma es la explicabilidad de los modelos (*Explainable AI* o XAI) y la construcción de la confianza del usuario. Para que un sistema prescriptivo sea verdaderamente efectivo y se integre en la vida diaria, el individuo debe comprender el motivo clínico detrás de una recomendación. Por ello, la investigación actual subraya la necesidad de diseñar arquitecturas que traduzcan las complejas decisiones matemáticas en justificaciones nutricionales claras y comprensibles, fomentando así la educación nutricional continua y empoderando al paciente en la gestión autónoma de su salud.

2.4 Contexto Global y Relevancia Sistémica

A nivel global, el aumento de las enfermedades no transmisibles (ENT) —como la diabetes, el cáncer y las patologías cardiovasculares— constituye una de las mayores amenazas para la sostenibilidad de los sistemas de salud. Según datos de la Organización Mundial de la Salud (OMS), las ENT representan el 63% de las muertes mundiales, un impacto que es prevenible en gran medida mediante intervenciones nutricionales efectivas [8]. Ante el agotamiento de los enfoques genéricos de "talla única", la Nutrición Personalizada surge como un imperativo estratégico. En el contexto de este análisis, se define como el asesoramiento dietético adaptado a las características biológicas (genotipo y fenotipo), estado de salud, estilo de vida y preferencias psicológicas del individuo.

Las investigaciones previas han fallado sistemáticamente debido a una baja profundidad en la integración de principios nutricionales, la omisión de las preferencias del usuario y una falta de capacidad de generalización en sus modelos semánticos. Esta brecha justifica la implementación de un enfoque híbrido que logre el equilibrio entre el rigor médico y la satisfacción del usuario.

Desde una perspectiva económica y de salud pública, la carga financiera derivada de los tratamientos crónicos para las enfermedades no transmisibles resulta cada vez más insostenible para los sistemas sanitarios modernos. La transición de una medicina reactiva —centrada de forma exclusiva en tratar la afección una vez manifestada— hacia un modelo preventivo fundamentado en la intervención dietética se presenta como una solución altamente costo-efectiva. En este marco, los sistemas inteligentes de *e-health* actúan como un vehículo democratizador, permitiendo escalar un asesoramiento nutricional con rigor clínico a grandes sectores de la población sin saturar la infraestructura médica presencial.

Por otro lado, la relevancia sistémica de estas tecnologías abarca también la dimensión sociológica y psicológica del consumo de alimentos. La alimentación no es únicamente un requerimiento biológico, sino un acto profundamente arraigado en la cultura, el estilo de vida y el entorno emocional del individuo.

Por consiguiente, un sistema algorítmico que imponga dietas clínicamente perfectas, pero culturalmente alienantes o monótonas, está abocado al abandono. El verdadero éxito de una arquitectura tecnológica avanzada radica en su capacidad para conciliar la exactitud de la ciencia metabólica con la flexibilidad necesaria para respetar la identidad culinaria del usuario, garantizando de este modo la máxima adherencia terapéutica a largo plazo.

2.5 Arquitectura General del Sistema y Flujo de Información

Para garantizar la escalabilidad y la precisión en entornos clínicos de gran escala, Yera et al. [10] de modo general propone una arquitectura modular de tres capas. Esta estructura permite un procesamiento de información desacoplado que transforma datos brutos en planes de acción personalizados.

El ecosistema se articula mediante el siguiente flujo de información:

1. **Dataset de Perfiles de Usuario:** Almacena la caracterización dinámica del individuo, integrando historial de consumo y restricciones médicas.
2. **Capa de Sistemas Inteligentes (Motor Estratégico):**
    - Contexto Nutricional: Filtrado inicial de exclusión basado en preferencias.
    - Modelos de Corto Plazo: Optimización diaria de menús.
    - Ajuste de Largo Plazo: Historial alimenticio del usuario y adición de nuevos alimentos al sistema de forma interactiva.
3. **Interfaz de Usuario:** Canal para la visualización del plan y modificación de los datos pertinentes.

2.6 Modelo de Optimización y Generación de Menús Diarios

El núcleo del marco de trabajo desarrollado en [10], resuelve la complejidad de equilibrar los requisitos biológicos estrictos con las preferencias psicológicas del usuario, minimizando la monotonía alimentaria.

PREPARACIÓN INICIAL DE LOS DATOS

Los pasos iniciales necesarios para preparar los datos para la generación de las recomendaciones se basan en dos objetivos: 1) la construcción de los perfiles de los alimentos, y 2) la definición de las plantillas de menú que se completarán con los alimentos.

1) Construcción de los perfiles de los alimentos. La definición del perfil de los alimentos se construye tomando como base dos tablas que indican la composición de los alimentos. Estas tablas proporcionadas por Wander contienen información nutricional de más de 600 alimentos, relacionada con la cantidad de calorías y más de 20 macronutrientes y micronutrientes diferentes. Las tablas mencionadas organizan los alimentos en 12 grupos, que son leches, huevos, carnes, pescados, leguminosas, frutos secos oleaginosos, aceites, cereales, postres, verduras, frutas y bebidas. Además, las tablas reflejan la cantidad de calorías, macronutrientes y micronutrientes en 100 g de cada alimento.

Alimento	Kilocalorías	Proteínas	Carbohidratos	Lípidos	Colesterol	Hierro	Calcio	...
Chuleta de cerdo (60 g)	198	9	0	18	43.2	1.5	4.8	...
Conejo (125 g)	202.5	27.5	0	10	81.25	1.25	25	...
Arroz blanco (130 g)	460.2	9.88	100.1	2.21	0	1.04	13	...
...	...	...	...	...	...	...	...	...

Tabla 2.3 Composición de alimentos

Para adecuar estos datos a la generación de recomendaciones, un nutricionista determinó porciones razonables para cada alimento según su tipo y características ; y, por tanto, calculó la cantidad de macro y micronutrientes pertenecientes a cada porción. La Tabla 2.3 presenta un fragmento de estos datos finales, que es la fuente a utilizar en los perfiles de los alimentos.

De esta manera, los perfiles de los alimentos se compondrán de la cantidad de nutrientes que han sido considerados como características clave para caracterizarlos. Estos nutrientes son proteínas, lípidos, carbohidratos, colesterol, sodio y grasas saturadas ; dejando para futuros trabajos el uso de un perfil de alimentos que considere nutrientes adicionales. Las kilocalorías también se descartan porque su valor se puede calcular a través de los valores de carbohidratos, proteínas y lípidos.

Además, en el presente trabajo este contexto será tratado como una tabla de decisión, donde los alimentos a consumir son las alternativas y las calorías y nutrientes son los criterios de decisión. La Tabla 2.4 formaliza la notación que se utilizará en el resto del documento para referirse a los componentes del perfil del alimento.

Término	**Nutriente**
pro_k	Cantidad de proteínas del alimento k
lip_k	Cantidad de lípidos del alimento k
cb_k	Cantidad de carbohidratos del alimento k
ch_k	Cantidad de colesterol del alimento k
sod_k	Cantidad de sodio del alimento k
sat_k	Cantidad de grasas saturadas del alimento k

Tabla 2.4 Criterio para la caracterización de alimentos

2) Definición de las plantillas de menú Por otro lado, también es necesario como dato inicial la definición de plantillas de menú que se utilizarán en la recomendación. Una plantilla de menú sigue el esquema común de una comida diaria típica. Esta plantilla de menú está compuesta por un desayuno, un almuerzo y una cena. No consideraremos los tentempiés o *snacks*, aunque la propuesta podría extenderse fácilmente para hacerles frente.

Con el fin de facilitar la definición de la plantilla, agrupamos los perfiles de alimentos en nuevos grupos de acuerdo con su nutriente principal asociado y las características relacionadas (Tabla 2.5). A partir de estos grupos, la Tabla 2.6 muestra la plantilla propuesta para un plan de comidas diario. Específicamente, los valores de los parámetros se propondrán más adelante en la sección del caso de estudio. El objetivo final de la propuesta es llenar esta plantilla considerando tanto criterios nutricionales como criterios de preferencias.

Nombre del grupo	**Composición del grupo**
Grupo G1 (Leche)	Leche, yogures
Grupo G2 (Cereales de desayuno)	Algunos cereales (ej. pan, trigo)
Grupo G3 (Fuentes de proteínas)	Huevos, carne, pescado
Grupo G4 (Fuentes de carbohidratos)	Algunos cereales (ej. arroz), legumbres
Grupo G5 (Verduras)	Verduras
Grupo G6 (Frutas)	Frutas

Tabla 2.5 Grupos de alimentos para la generación de menús

Comida	**Composición**
Desayuno	n_{G1} alimentos del grupo G1 (Leche, yogures)
	n_{G2} alimentos del grupo G2 (Cereales de desayuno)
	n_{G6} alimentos del grupo G6 (Frutas)
Almuerzo	n^{l}_{G3} alimentos del grupo G_3 (Proteínas)
	n^{l}_{G4} alimentos del grupo G4 (Carbohidratos)
	n^{l}_{G5} alimentos del grupo G5(Verduras)
	n_{G6} alimentos del grupo G6 (Frutas)
Cena	n^{d}_{G3} alimentos del grupo G3 (Proteínas)
	n^{d}_{G4} alimentos del grupo G4 (Carbohidratos)
	n^{d}_{G5} alimentos del grupo G5 (Verduras)
	n_{G6} alimentos del grupo G6 (Frutas)

Tabla 2.6 Plantilla para un menú

MODELO DE RECOMENDACIÓN DE MENÚS BASADO EN OPTIMIZACIÓN

Aquí se introduce un enfoque que toma como entrada los alimentos clasificados como apropiados en la sección anterior, para rellenar la plantilla de menú. El objetivo del enfoque es proporcionar recomendaciones de alimentos que sean nutricionalmente apropiadas y que también coinciden con las preferencias actuales del usuario.

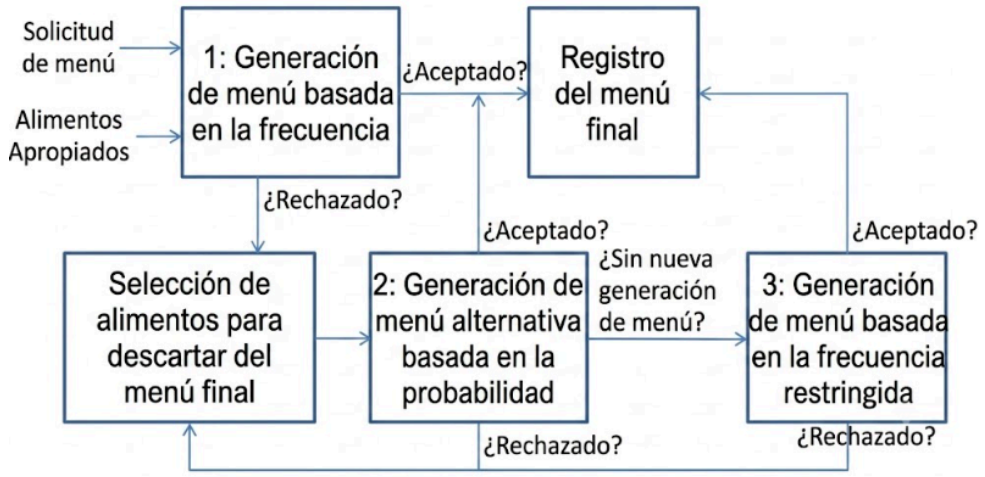


Figura 2.1 Esquema general aproximado de la recomendación de menús

La Figura 2.1 presenta una visión general del enfoque para la recomendación de menús. Este enfoque recibe como entrada la solicitud del menú y la lista de alimentos prefiltrados, y se compone de tres fases principales: la generación de menús basada en la frecuencia (paso 1), el refinamiento probabilístico del menú (paso 2), y la generación de menús basada en frecuencia restringida (paso 3). Aunque cada fase sigue un principio de funcionamiento diferente para la generación del menú, en todos los casos esta tarea se abordará como un problema de optimización centrado en rellenar las plantillas de menú predefinidas diarias, proporcionar los nutrientes diarios necesarios al usuario, y maximizar las preferencias del usuario sobre el menú final recomendado.

Para alcanzarlo, formulamos un escenario de optimización que considera el menú generado como un vector.

En ambos escenarios de plan de comidas diario, se adoptará el siguiente modelo de optimización, cuya segunda ecuación toma como base un esquema tradicional de planificación de dietas propuesto por Anderson y Earle [21]. Más allá de este trabajo, nuestra propuesta se centra en:

$$
\begin{aligned}
&Maximize \sum_{k \in A} w_k f_k \\
&s.t. \\
&|\sum_{j}(nt_{kj} * f_k) - b_j| \leq \alpha, \quad for\ each\ nutrient\ 1,\ 2,\ 3, \ldots, J \\
&\sum_{k \in G_a} f_k = n_{G_a}, \\
&for\ each\ n_{G_a} \in \{n_{G_1}, n_{G_2}, n^l_{G_3}, n^l_{G_4}, n^l_{G_5}, n^d_{G_3}, n^d_{G_4}, n^d_{G_5}, n_{G_6}, \},
\end{aligned}
$$

(2.1)

- Maximizar la suma de preferencias de todos los alimentos incluidos en el plan .
- Verificar que los nutrientes del plan generado están muy cerca de los nutrientes requeridos para el perfil de usuario actual . Este objetivo se verifica asegurando que para cada nutriente, la diferencia absoluta entre la cantidad requerida y la cantidad final, está siempre por debajo de un umbral alpha. Esto se basa en el hecho de que se deben evitar tanto los menús que están por debajo como por encima del nutriente requerido. Sin embargo, también remarcamos que es improbable que un menú generado coincida exactamente con los nutrientes requeridos de un perfil de usuario. Por lo tanto, este parámetro alpha es necesario para gestionar esa desviación mínima esperada de los menús aún apropiados.
- Garantizar que el plan generado cumple con las plantillas de menú presentadas . Este objetivo se constata asegurando que, para cada categoría de alimentos, la cantidad de alimentos incluidos en el menú coincide con la cantidad predefinida en las plantillas.

2.6.1 Función Objetivo y Restricciones del Modelo

El modelo maximiza la preferencia del usuario sujeta a una **Plantilla de Menú** (Desayuno, Almuerzo, Cena) y restricciones nutricionales de "verdad absoluta" (Ground Truth). Ejemplo:

1. Distribución Energética por Comida: 15% Desayuno, 45% Almuerzo y 40% Cena.
2. Distribución de Macronutrientes: 50% Carbohidratos, 20% Proteínas y 30% Lípidos.
3. Conversión Energética**:** 1g Proteína/Carb = 4 kcal; 1g Lípido = 9 kcal.

2.6.2 Cálculo de Requerimientos y Parámetros de Personalización

Para determinar el gasto calórico, se utilizan las fórmulas de **Harris-Benedict**, ajustadas por **Multiplicadores de Nivel de Actividad** (Table 9):

- **Hombres:** BMR = 10 * peso + 6.25 * altura - 5 * edad + 5
- **Mujeres:** BMR = 10 * peso + 6.25 * altura - 5 * edad - 161

Factores de Actividad: El BMR se multiplica por un factor de 1.2 (sedentario) hasta 1.9 (ejercicio muy fuerte) para obtener el requerimiento energético real.

La generación sigue tres fases:

1. **Generación basada en frecuencia:** Sugiere alimentos preferidos no consumidos recientemente (usando decaimiento \theta).
2. **Refinamiento probabilístico:** Utiliza probabilidad condicional P(k | m_1, m_2...) para sugerir alternativas si el usuario rechaza una opción, basándose en lo ya aceptado (agr).
3. **Frecuencia restringida:** Salvaguarda en caso de que el modelo probabilístico no encuentre solución.

Análisis de Valor: La introducción del parámetro de relajación (\alpha) permite gestionar la desviación mínima entre el plan ideal y el real, asegurando flexibilidad sin comprometer la precisión metabólica [10].

Capítulo 3 - Análisis y diseño

En este capítulo se desglosa en qué consiste la aplicación web implementada, el análisis y diseño seguidos, requisitos técnicos y funcionales del sistema, cuáles han sido las distintas metodologías de trabajo utilizadas y la arquitectura empleada.

3.1 Introducción

El presente proyecto consiste en el desarrollo de una aplicación web destinada a la recomendación inteligente de menús nutricionalmente apropiados en relación a las características individuales de cada usuario. La finalidad de esta aplicación es implementar una aplicación web que le permita al usuario saber qué comer, en qué cantidad y además tener en cuenta la preferencias del usuario, generando así menús adaptados al usuario junto a la posibilidad de crear un hábito positivo en el usuario además de prolongado en el tiempo.

La metodología de desarrollo de este proyecto ha seguido las etapas clásicas de la Ingeniería Software, definida formalmente como la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software; es decir, la aplicación de principios de ingeniería al software [17].

- **Planificación**: Se evalúa la viabilidad del proyecto, definiendo el alcance general, los recursos necesarios, los riesgos y el cronograma de trabajo.
- **Análisis**: Se recopilan y documentan los requisitos técnicos y funcionales para entender exactamente qué problemas debe resolver el sistema.
- **Diseño**: Se define la arquitectura del software, estructurando cómo se organizarán los datos, los componentes y la interfaz de usuario para cumplir con los requisitos.
- **Implementación**: Es la fase de codificación pura, donde se programa el código fuente para transformar los modelos teóricos de diseño en un producto real y funcional.

- **Pruebas e integración**: Se ejecutan validaciones para detectar errores (bugs) y se ensamblan las distintas partes del código para asegurar que todo el sistema funcione correctamente en conjunto.
- **Mantenimiento**: Llevado a cabo tras el lanzamiento del producto; consiste en actualizar aspectos de la aplicación, corregir fallos imprevistos y añadir nuevas funcionalidades a lo largo de su vida útil.

Figura 3.1

3.2. Análisis de Requisitos

Esta sección estará destinada a listar los distintos requisitos funcionales y no funcionales que tiene el sistema. Los requisitos funcionales definen las funciones específicas, los comportamientos y los servicios que el software o sistema debe proporcionar a sus usuarios o a otros sistemas [18]. Estos requisitos detallan cómo debe reaccionar el sistema ante entradas específicas y cómo debe comportarse en situaciones particulares [19].

En cambio los requisitos técnicos describen cómo debe funcionar el sistema y las limitaciones bajo las cuales debe operar. Mientras que los funcionales se

centran en el comportamiento del usuario, los técnicos se centran en la arquitectura, el entorno operativo, el rendimiento, la seguridad y las tecnologías específicas que deben utilizarse [20].

3.2.1. Requisitos Funcionales:

- **RF-01:** El sistema debe permitir acceder a la funcionalidad de la aplicación mediante nombre de usuario y contraseña.
- **RF-02:** El sistema debe permitir al usuario registrarse utilizando su nombre, apellidos, nombre de usuario, email, contraseña, edad, altura, sexo, peso y nivel de actividad.
- **RF-03:** El sistema debe permitir al usuario cambiar su contraseña introduciendo el email y la nueva contraseña.
- **RF-04:** El sistema debe permitir al usuario modificar su información personal, altura, peso, sexo, edad y nivel de actividad. Además debe permitir añadir y eliminar elementos de alimentos pertenecientes a su usuario.
- **RF-05:** El sistema debe permitir al usuario eliminar su cuenta.
- **RF-06:** El sistema de poder mostrar el menú que se generó en un día en específico indicando la fecha concreta.
- **RF-07:** El sistema debe de poder generar un menú nutricionalmente correcto adaptándose a las necesidades y preferencias del usuario en el día de hoy.
- **RF-08:** El sistema debe de implementar un buscador para poder consultar la información más relevante de cada alimento perteneciente al sistema.
- **RF-09:** El sistema debe permitir añadir un alimento a la bbdd indicando el nombre, grupo nutricional, kcal, proteínas, carbohidratos, grasas y fibra.

3.2.2. Requisitos No Funcionales (Técnicos):

- **RT-01:** El backend debe estar desarrollado en el lenguaje de programación Java 21.
- **RT-02:** El backend de la aplicación debe estar desarrollado utilizando el framework Spring Boot en su versión **3.5.7** para asegurar la compatibilidad de las dependencias y el soporte de la plataforma.
- **RT-03:** El ciclo de vida del proyecto y la gestión de sus librerías se controlará mediante Maven, heredando la configuración estructural de spring-boot-starter-parent.
- **RT-04:** Se utilizará Spring Data JPA basado en Hibernate para la interacción y transacciones con la base de datos.

3.2.3 Diagramas de Casos de Uso

A partir de los requisitos funcionales previamente definidos, se ha elaborado el diagrama de casos de uso general del sistema. Este artefacto UML ilustra las interacciones principales que el actor (el usuario final) puede llevar a cabo dentro de la plataforma. Como se observa en la figura, el sistema centraliza la experiencia en el "Usuario", permitiéndole desde la gestión administrativa de sus credenciales y datos biométricos, hasta la generación de su menú diario personalizado y la consulta del catálogo de alimentos.

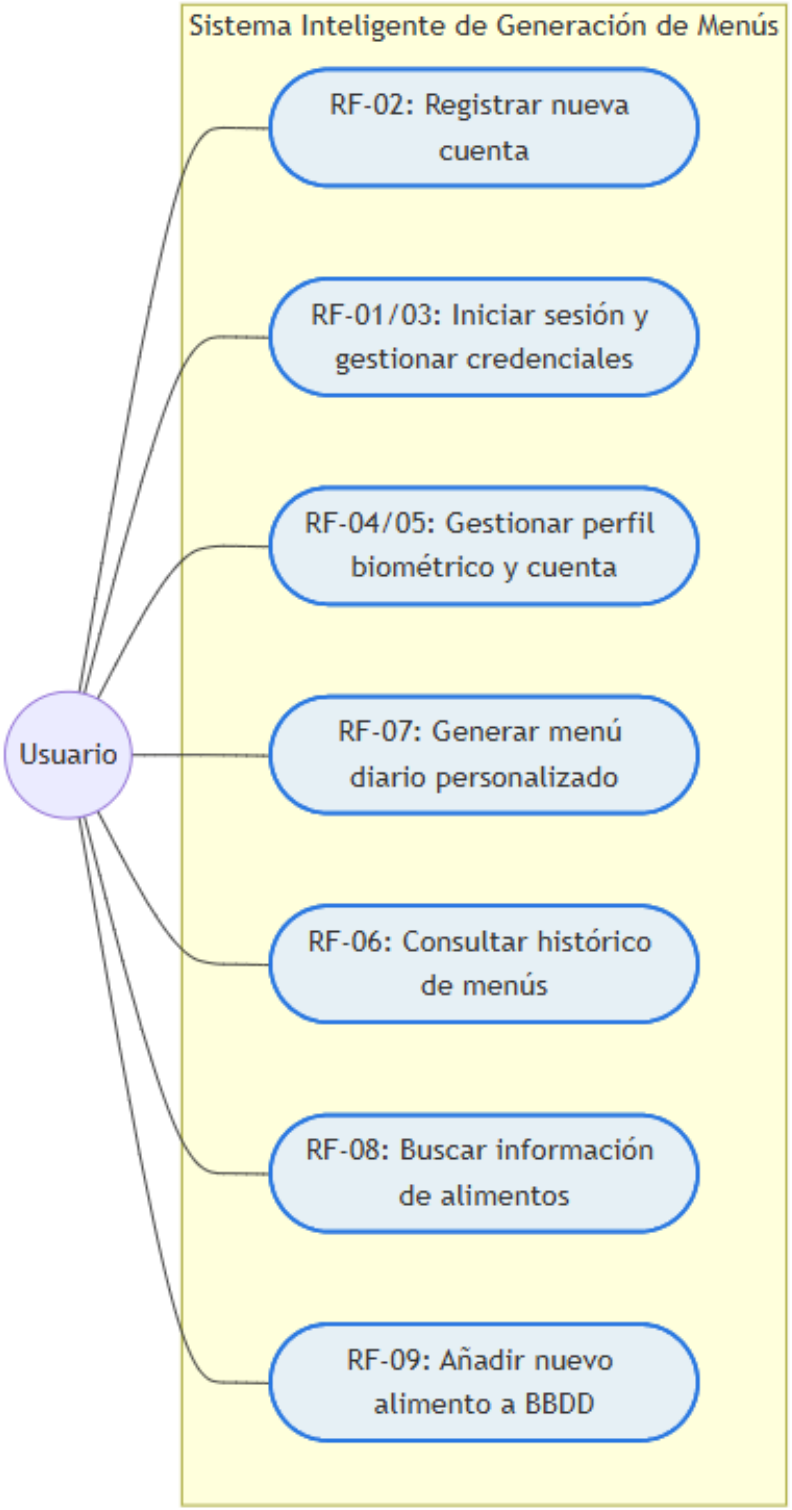

Sistema Inteligente de Generación de Menús
RF-02: Registrar nueva cuenta
RF-01/03: Iniciar sesión y gestionar credenciales
RF-04/05: Gestionar perfil biométrico y cuenta
Usuario
RF-07: Generar menú diario personalizado
RF-06: Consultar histórico de menús
RF-08: Buscar información de alimentos
RF-09: Añadir nuevo alimento a BBDD

Figura 3.2

3.3 Flujo de trabajo

Fase de Configuración (5 de enero - 1 de febrero)

- Ecosistema y elección de tecnologías: En esta etapa se seleccionó la pila tecnológica central del proyecto. Como se describe en el Capítulo 4, esto implicó establecer Java 21 y Spring Boot 3.5.7 para el servidor backend, y un entorno local de Python 3 para soportar el algoritmo de optimización matemática . Para el lado del cliente, se descartaron frameworks pesados en favor de HTML5, CSS3 y JavaScript moderno.
- Especificación de requisitos: Esta labor se tradujo en la redacción del apartado de Análisis de Requisitos del Capítulo 3, donde se listaron los requisitos funcionales y los requisitos técnicos que guiarán el desarrollo.

Iteración 1: Análisis y Diseño (1 de febrero - 1 de marzo)

- Diseño de la interfaz: Esta actividad se corresponde con la creación de los *mockups* preliminares. Se diseñaron bocetos visuales para definir la jerarquía y disposición de los componentes sin incurrir en costes de codificación, alineando los requisitos con una buena experiencia de usuario.
- Refinamiento del Backlog: Siguiendo las reglas de Scrum, el equipo reorganizó el *Product Backlog* (la lista priorizada de requisitos) basándose en las impresiones obtenidas de ese diseño inicial, preparando el terreno para la codificación pura.

Iteración 2: Desarrollo y Prototipado (1 de marzo - 4 de mayo)

- Desarrollo del Backend: Se construyó la infraestructura del servidor en Java aplicando una Arquitectura por Capas y basándose en principios de Arquitectura Hexagonal. Se programaron los Controladores REST, los Servicios (lógica de negocio) y los Repositorios para interactuar con la base de datos .
- Módulo de Generación de Menús: Se implementó el núcleo inteligente del software. Para ello, se diseñó la comunicación entre procesos (IPC) mediante ProcessBuilder, donde Spring Boot serializa los datos del

usuario en archivos JSON y delega el cálculo matemático pesado al script generador_de_menus.py .

- Integración Frontend-Backend: Se unieron las partes garantizando que el frontend en JavaScript pudiera consumir de forma asíncrona (vía peticiones Fetch) los endpoints de la API RESTful creados en el backend.

Iteración 3: Integración y Finalización (4 de mayo - 30 de junio)

- Pruebas exhaustivas: Se utilizó una colección extensa de Postman para probar los endpoints de la API de forma independiente, y se simularon tipos de casos para certificar que las restricciones del algoritmo se aplicaban de forma correcta.

Cabe remarcar que durante todo el desarrollo del proyecto se ha ido redactando la memoria de forma continua conforme se ha ido desarrollando el trabajo.

3.4. Arquitectura General del Sistema

El proyecto se ha diseñado siguiendo el paradigma arquitectónico Cliente-Servidor, estableciendo una separación estricta entre la interfaz de usuario y la lógica de negocio. Esta decisión garantiza un sistema escalable, mantenible y modular. La comunicación entre ambos extremos se realiza de forma asíncrona a través de una API RESTful, utilizando el formato JSON para el intercambio de información.

A nivel macroscópico, el sistema se divide en dos bloques principales:

3.4.1 Capa de Presentación (Frontend / Cliente)

Actúa como la interfaz directa con el usuario final. Se ha optado por un desarrollo ligero y sin dependencias externas complejas, utilizando HTML5, CSS3 y JavaScript puro.

Sus responsabilidades principales son:

- **Renderizado dinámico:** Manipulación del DOM para mostrar la información del usuario, listas de alimentos, enfermedades y menús sin necesidad de recargar la página completa, ofreciendo una experiencia fluida similar a una *Single Page Application* (SPA).
- **Gestión del estado de la sesión:** Utilización del almacenamiento local del navegador (localStorage) para persistir la información del usuario autenticado (mediante el patrón de Objetos de Transferencia de Datos o DTOs) de forma segura durante su navegación.
- **Consumo de la API:** Realización de peticiones HTTP asíncronas hacia los *endpoints* expuestos por el servidor para operaciones de lectura y escritura.

3.4.2 Capa de Lógica de Negocio y Datos (Backend / Servidor)

Desarrollado sobre el framework Spring Boot (Java), el servidor actúa como el núcleo del sistema, procesando las reglas de negocio, gestionando la seguridad y administrando el acceso a la base de datos. Para asegurar un bajo acoplamiento y una alta cohesión, el backend implementa una Arquitectura por Capas:

- **Capa de Infraestructura (Controladores REST):** Define los *endpoints* de la API. Intercepta las peticiones HTTP del cliente, delega el procesamiento a la capa de aplicación y devuelve las respuestas adecuadas (códigos de estado HTTP y objetos JSON).

- **Capa de Aplicación (Servicios):** Contiene la lógica de negocio central del sistema. Aquí se validan las reglas del recomendador de dietas, la gestión de contraseñas y las restricciones entre entidades (usuarios, enfermedades, alimentos).
- **Capa de Dominio (Entidades y DTOs):** Define los modelos de datos exactos del sistema. Para evitar la exposición directa de la estructura interna de la base de datos hacia el cliente, se hace uso del patrón DTO (*Data Transfer Object*) apoyado por librerías como *MapStruct* para el mapeo automático entre entidades y objetos de transferencia.
- **Capa de Persistencia (Repositorios):** Haciendo uso de *Spring Data JPA*, abstrae las consultas a la base de datos relacional, permitiendo operar con los objetos de dominio mediante operaciones CRUD estandarizadas.

3.4.3 Diagrama de Flujo de la Arquitectura

El ciclo de vida de una petición típica en el sistema sigue el siguiente flujo de ejecución:

1. El Cliente (Navegador) captura un evento del usuario y lanza una petición HTTP (GET, POST, PUT, DELETE, PATCH) con formato JSON.
2. El Controlador REST en Spring Boot recibe la petición, valida la entrada (@Valid) y mapea el JSON entrante a un DTO de entrada (*InputDto*).
3. El Controlador inyecta el DTO convertido en Entidad hacia el Servicio de Aplicación.
4. El Servicio aplica la lógica de negocio pertinente e interactúa con el Repositorio JPA.
5. El Repositorio persiste o extrae la información de la Base de Datos.
6. La respuesta viaja de vuelta: el Servicio devuelve la Entidad, el Controlador la mapea a un DTO de salida (*OutputDto*) y la envía como JSON al Cliente, quien finalmente actualiza la interfaz visual.

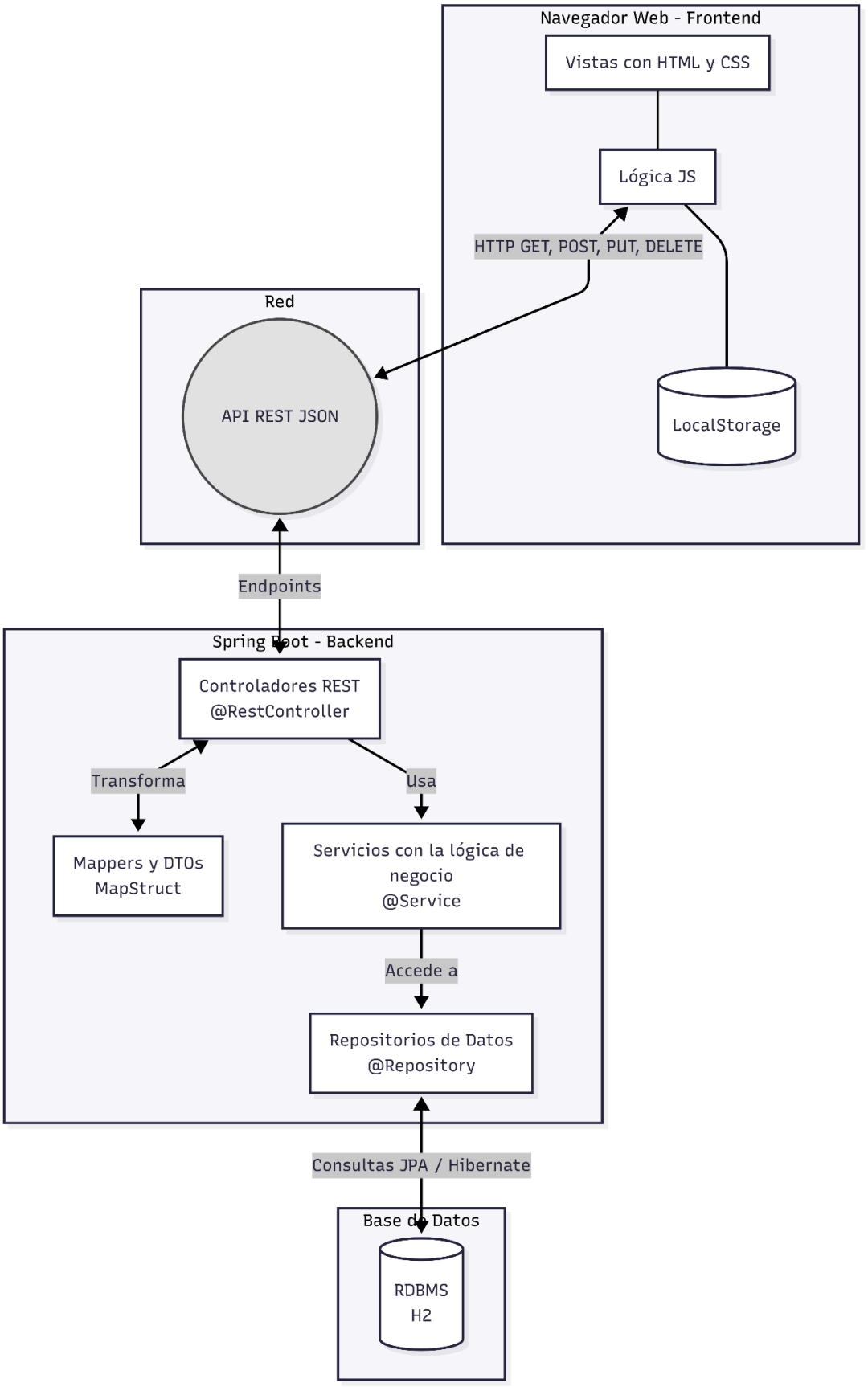


Figura 3.3 Diagrama de Flujo de la Arquitectura

3.5. Diseño de la Base de Datos

En este apartado se llevará a cabo una explicación de las distintas entidades pertenecientes al sistema junto a sus atributos. También se explicarán las relaciones que existen entre ellas y su cardinalidad. Además se mostrará un diagrama de entidad relación para su mejor entendimiento.

3.5.1 Entidades y Atributos

Definición de las principales entidades pertenecientes al proyecto junto a sus respectivos atributos.

User (Usuario): Representa a los usuarios de la plataforma y almacena los datos personales y variables fisiológicas necesarias para el cálculo de la tasa metabólica basal para la correcta generación de los menús.

- **id** (Long, Clave Primaria): Identificador único autogenerado.
- **name** (String): Nombre del usuario.
- **surname** (String): Apellidos del usuario.
- **userName** (String): Nombre de usuario único para las credenciales de acceso. Utilizado para loguearse en la aplicación web.
- **email** (String): Dirección de correo electrónico (Único). Utilizada en el cambio de contraseña del usuario.
- **password** (String): Contraseña del usuario.
- **weight** (Float): Peso corporal del individuo en kilogramos.
- **height** (Float): Altura del individuo en centímetros.
- **age** (Integer): Edad del usuario en años.
- **sex** (Boolean): Género del usuario (representado como verdadero para masculino y falso para femenino).
- **activityLevel** (Integer): Nivel de actividad física indexado del 1 al 5, crítico para los factores de ajuste metabólico.

Food (Alimento): Contiene la caracterización e información nutricional detallada por cada 100 gramos de un alimento.

- **id** (Long, Clave Primaria): Identificador único del alimento.
- **name** (String): Nombre común del alimento.
- **cod** (String): Código alfanumérico de identificación técnica.
- **groupOfFood** (Enumerado / GroupOfFood): Clasificación del grupo nutricional al que pertenece (G1 a G6, abarcando lácteos, cereales, proteínas, carbohidratos complejos, verduras y frutas).
- **Macronutrientes y Energía***:* kcal (Energía), pro (Proteínas), hc (Carbohidratos), lip (Lípidos/Grasas) y fib (Fibra).
- **Perfil Lipídico y Micronutrientes***:* Atributos específicos para el control clínico avanzado, incluyendo colesterol (cole), ácidos grasos (lipsat, lipmo, lippoli), minerales (ca, na, k, mg, fe, etc.) y vitaminas (vita, vitc, vitd, vitb12, etc.).

Menu (Menú): Entidad que agrupa la planificación alimentaria asignada a un usuario para un día en concreto.

- **id** (Long, Clave Primaria): Identificador único del menú diario.
- **date** (LocalDate): Fecha concreta del calendario para la que se planifica el menú.

Meal (Comida): Entidad que define una ingesta de un menú concreto.

- **id** (Long, Clave Primaria): Identificador de la comida.
- **typeOfMeal**: Define el momento del día en el que llevar a cabo la ingesta. Es un enumerado que contempla el desayuno, almuerzo, cena o snack. (BREAKFAST, LUNCH, DINNER o SNACK).

MealFood (Relación Comida-Alimento): Entidad intermedia implementada con la finalidad de introducir la cantidad en gramos de un alimento en una ingesta concreta.

- **id** (Long, Clave Primaria): Identificador único de la asignación.

- **recommendedGrams** (Float): Atributo propio de la relación que especifica la cantidad exacta en gramos que el usuario debe consumir de dicho alimento, calculada a medida por el script de optimización.

3.5.2 Relaciones y Cardinalidades

El comportamiento del sistema y las restricciones de integridad relacional se definen mediante los siguientes vínculos entre entidades:

- **User - Menu (1:N):** Un usuario puede registrar o tener asignados múltiples menús a lo largo del tiempo (construyendo su historial). Por el contrario, un menú pertenece obligatoria y unívocamente a un único usuario. Esta relación se implementa de manera bidireccional en el backend mediante las anotaciones @OneToMany en el dominio del usuario y @ManyToOne en el del menú.
- **User - Food (N:M):** Un usuario mantiene una colección de alimentos asociados que representan sus preferencias o exclusiones del catálogo general (por ejemplo, alimentos deshabilitados por intolerancias o gustos personales). Un mismo alimento puede ser excluido o preferido por múltiples usuarios. En el código, esta colección se gestiona eficientemente a través de un conjunto (Set<FoodSimpleEntity>) indexado en el perfil del usuario.
- **Menu - Meal (1:N):** Un menú se compone estructuralmente de varias comidas independientes distribuidas a lo largo del día. Cada toma específica está vinculada en exclusiva al menú de ese día. Se mapea en el servidor mediante una relación bidireccional @OneToMany en la entidad menú y @ManyToOne en la entidad comida.
- **Meal - MealFood (1:N)** y **Food - MealFood (1:N)**: Para solventar la relación de muchos a muchos (*Many-to-Many*) implícita entre comidas y alimentos, se ha implementado la entidad asociativa MealFood (tabla meal_food). Una comida (Meal) contiene un conjunto de registros de esta entidad intermedia (Set<MealFoodEntity>) , y cada registro de MealFood apunta a un único alimento (Food). Este diseño es fundamental, ya que permite asociar de forma unívoca a cada alimento

introducido en una comida el peso preciso en gramos (recommendedGrams) determinado por el algoritmo matemático. Además nos permite tener implementada la lógica para que en un futuro se puedan ir ajustando con mayor precisión cada toma de cada alimento.

3.5.3 Diagrama Entidad-Relación

Aquí muestro un diagrama entidad-relación con las distintas relaciones y atributos de las cinco entidades que participan en mi proyecto. Algunos atributos de la entidad FOOD han sido suprimidos para la mejor lectura del mismo.

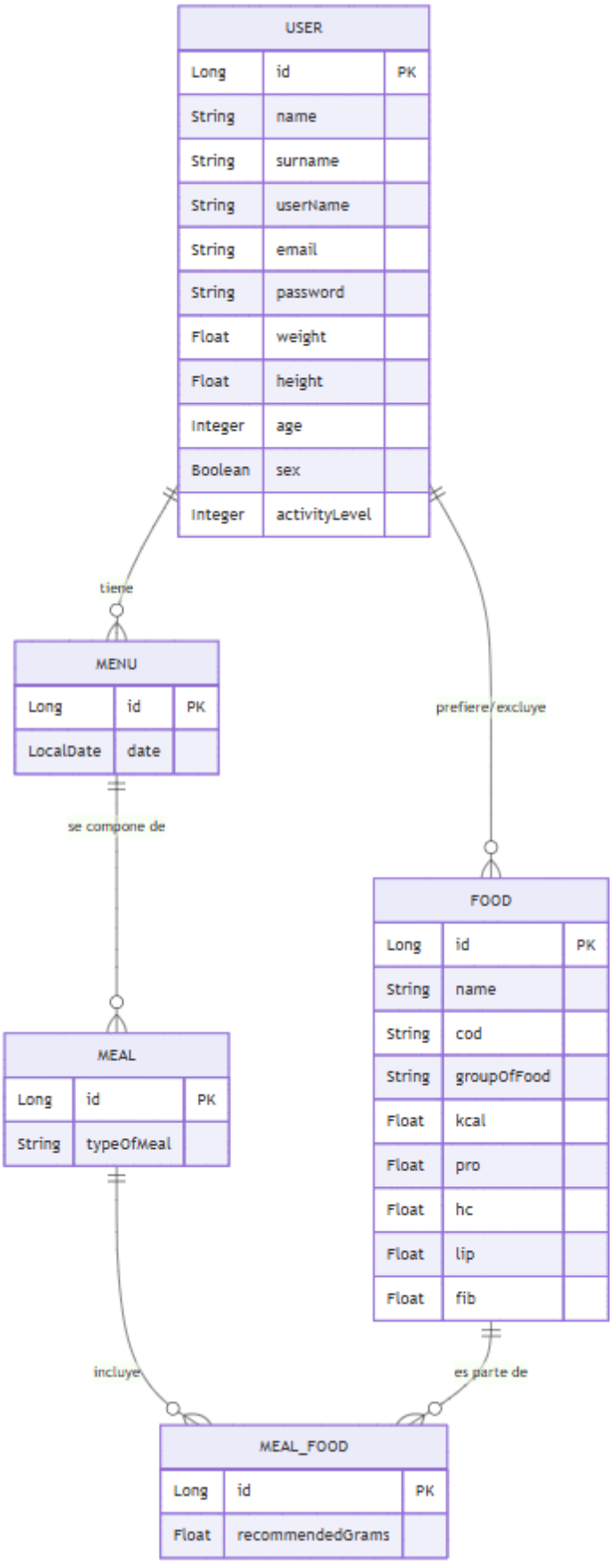


Figura 3.4 Diagrama entidad relación

3.6. Diseño del Backend y Lógica Inteligente

El servidor backend es el núcleo computacional de la aplicación, encargado de procesar la lógica de negocio y proteger el acceso a los datos. Se ha diseñado siguiendo los principios de la **Arquitectura Hexagonal (Puertos y Adaptadores)** y el diseño guiado por el dominio (DDD), dividiendo el sistema en Infraestructura, Aplicación y Dominio.

3.6.1. Estructura del Servidor y API

El servidor expone una API RESTful estructurada por dominios lógicos: Users, Food, Menus y Meals. Para asegurar un bajo acoplamiento y una alta cohesión, el backend implementa una arquitectura por capas, donde cada módulo sigue el siguiente patrón:

- **Capa de Infraestructura (Controladores REST)**: Actúan como punto de entrada de las peticiones HTTP del cliente. Controladores como CreateUserController o ReadMenuController interceptan las llamadas, reciben un DTO (Data Transfer Object) de entrada y realizan las validaciones iniciales mediante las anotaciones de Jakarta Validation (@Valid, @NotNull, etc.). Una vez validada la entrada, delegan el procesamiento a la capa de aplicación.
- **Capa de Aplicación (Servicios)**: Contienen la lógica de negocio central del sistema. Clases como GenerateMenuServiceImpl gestionan la orquestación de los procesos, como validar las reglas del recomendador de dietas, la gestión de contraseñas y las restricciones entre las distintas entidades del sistema.
- **Mapeadores y DTOs (Capa de Dominio)**: Para evitar la exposición directa de la estructura interna de la base de datos hacia el cliente, se hace uso del patrón DTO. Se utiliza la librería MapStruct (ej. UserDtoMapper, FoodJpaMapper) para el mapeo automático. Estos mapeadores convierten automáticamente las Entidades de Dominio (UserEntity) en entidades JPA (UserJpa) para la persistencia, o en DTOs (UserOutputDto) para enviarlas al cliente como objetos JSON, garantizando que datos sensibles no se expongan.

- **Capa de Persistencia (Repositorios)**: Es la encargada de interactuar de forma abstracta con la base de datos relacional. Se ha diseñado utilizando el patrón de Puertos y Adaptadores, dividiendo la responsabilidad en tres componentes clave:
    - **Interfaces de Dominio**: Definen los contratos de persistencia sin acoplarse a ninguna tecnología específica (ej. ReadMenuRepository, CreateUserRepository).
    - **Implementaciones (Adaptadores)**: Clases como ReadMenuRepositoryImpl que implementan las interfaces del dominio e inyectan las herramientas de la base de datos.
    - **Spring Data JPA**: Haciendo uso de interfaces que extienden de JpaRepository (como MenuRepositoryJpa o UserRepositoryJpa), se abstraen las consultas a la base de datos. Esto permite realizar operaciones CRUD estandarizadas y consultas personalizadas mediante métodos derivados, operando directamente con las entidades JPA (MenuJpa, UserJpa) sin necesidad de escribir sentencias SQL de forma manual.
- **Gestión de Excepciones Global**: Se ha implementado un CustomExceptionHandler mediante la anotación @ControllerAdvice. Este componente captura excepciones lanzadas en cualquier capa (como NotFoundException o EmailAlreadyExistsException) y devuelve al cliente web una respuesta HTTP estructurada con el código de error correspondiente (CustomError), mejorando la robustez y la legibilidad de los errores en el frontend.

3.6.2. Motor de Generación Inteligente de Menús

Una de las características más importantes del sistema reside en su motor de recomendación matemática, abstraído en el backend a través de la interfaz MenuOptimizerPort. Dado que el ecosistema Java no es el más eficiente de forma nativa para el cálculo algebraico avanzado y la resolución de problemas de programación lineal, se ha diseñado una arquitectura híbrida que delega el esfuerzo computacional a un módulo específico en Python.

Interoperabilidad entre Java y Python

La comunicación entre ambos entornos se logra mediante un patrón de Comunicación entre Procesos (IPC), orquestado por la clase nativa **ProcessBuilder** de Java. El flujo de ejecución consta de tres etapas fundamentales:

1. **Serialización de Entradas:** El backend en Spring Boot recopila los parámetros biométricos del usuario y el catálogo de alimentos permitidos. Empleando la librería Jackson (librería de Java que nos permite trabajar con la información en formato JSON), esta información se serializa en el disco del servidor mediante archivos temporales JSON y de texto (cómo profile.json y foods.txt utilizados por el script python para obtener la información acerca de los alimentos presentes en el sistema y la información del usuario logueado).

2. **Ejecución del Proceso:** Java instancia el proceso del sistema operativo para ejecutar el script **generador_de_menus.py**. El hilo principal de la aplicación Java queda bloqueado a la espera de la resolución matemática.

3. **Captura y Deserialización:** Al finalizar su ejecución, el script en Python no escribe un nuevo archivo, sino que imprime la estructura del menú óptimo resultante en la salida estándar (stdout) en formato JSON. Java intercepta este flujo de datos (InputStream), lo lee y mapea la información estructurada a entidades de dominio internas (MenuEntity y MealFoodEntity) para su validación final y persistencia en la base de datos relacional.

Modelo de Optimización (MILP)

El núcleo algorítmico, implementado íntegramente en Python, utiliza técnicas de Programación Lineal Entera Mixta (MILP) mediante la función **scipy.optimize.milp** de la librería científica SciPy. El cálculo del menú óptimo se ejecuta de forma secuencial (desayuno, almuerzo y cena) siguiendo cuatro fases de procesamiento:

- **Fase A - Carga de Datos y Análisis Lexicográfico:** El script procesa el catálogo de alimentos utilizando expresiones regulares para extraer y tipar dinámicamente los macronutrientes, garantizando la conversión de cadenas nulas en valores flotantes operables matricialmente.

- **Fase B - Cálculo Metabólico:** Para determinar el requerimiento energético (Target Kcal), el sistema implementa la ecuación de Mifflin-St Jeor, considerada el estándar clínico actual para estimar la Tasa Metabólica Basal (BMR):

$$BMR = 10 * peso + 6.25 * altura - 5 * edad + S \quad (3.1)$$

  (Donde la constante S toma un valor de + 5 para hombres y - 161 para mujeres).

  Este valor se escala en función del factor de actividad física del usuario. Las calorías totales obtenidas se distribuyen fijando un objetivo nutricional del **20%** en proteínas, **50%** en carbohidratos y **30%** en lípidos.

- **Fase C - Historial y Pesos de Preferencia:** Como estrategia para mitigar la fatiga dietética, el sistema calcula un peso de preferencia penalizando la frecuencia de consumo de cada alimento. Se emplea una función de decaimiento exponencial:

$$W = \left(\frac{n_k}{N}\right)\left(e^{\theta\frac{t_c - t_k}{t_c}} - 1\right) \quad (3.2)$$

Donde el peso final (W) depende de la frecuencia histórica (n_k), el tiempo actual (t_c), la última toma registrada (t_k) y el parámetro de decaimiento(θ)

- **Fase D - Resolución de Restricciones:** La función objetivo del algoritmo busca maximizar la preferencia del usuario descrita en la fase anterior, estando sujeta a una estricta matriz de restricciones algebraicas. Estas engloban fronteras nutricionales (permitiendo una desviación máxima definida por el parámetro alpha de 75.0 unidades), plantillas estructurales (exigiendo combinaciones concretas de grupos alimenticios por toma), y restricciones binarias limitadas entre 0 y 1 que garantizan la selección matemática de, como máximo, una ración exacta por alimento propuesto.

3.7. Diseño del Frontend

La capa de presentación ha sido concebida para ofrecer una experiencia rápida y sin interrupciones, simulando una *Single Page Application* (SPA), pero construida íntegramente con estándares web nativos (HTML5, CSS3 y JavaScript puro), evitando la sobrecarga de frameworks externos.

3.7.1. Arquitectura de la Interfaz

La navegación del usuario gira en torno a un archivo central (**dashboard.html**). La aplicación gestiona su estado de la siguiente manera:

- **Almacenamiento Local (Local Storage)**: Tras un inicio de sesión exitoso, el sistema guarda el DTO del usuario autenticado (currentUser) en el almacenamiento del navegador. Esto evita tener que hacer peticiones recurrentes al servidor para consultar datos básicos.

- **Navegación por Pestañas**: La interfaz se divide en múltiples *tabs* (Configuración, Historial, Menú Diario, Base de Datos de Alimentos). Mediante funciones de JavaScript (showTab()), se altera la clase active en el DOM (Document Object Model) para ocultar o mostrar secciones instantáneamente sin recargar la página.
- **Comunicación Asíncrona**: Todas las operaciones CRUD (Crear, Leer, Actualizar, Borrar) utilizan la API nativa Fetch. Las promesas de JS (async/await) permiten procesar la información JSON devuelta por Spring Boot en segundo plano y actualizar la vista dinámicamente.

3.7.2. Diseño de la Interfaz de Usuario (UI/UX)

El estilo visual aplica una hoja de estilos centralizada (style.css) basada en una paleta de colores limpia y usable, destacando el verde corporativo de la UJA. Las vistas principales incluyen:

- **Panel de Ajustes**: Permite la edición de datos biométricos en tiempo real y ofrece un buscador para que el usuario pueda seleccionar y excluir alimentos específicos de la base de datos de manera intuitiva.
- **Gestor de Menús (Daily Meals / History)**: Presenta las comidas generadas en tarjetas legibles, desglosando la información clave devuelta por el motor inteligente: el nombre del alimento, los macronutrientes aportados y, muy especialmente, la ración recomendada en gramos.
- **Sección para la visualización de la información acerca de los alimentos**: Donde se presentan todos los alimentos presentes en el sistema junto a un buscador que nos permite agilizar nuestra consulta.
- **Creación de alimentos:** Donde se presenta un formulario para la creación de alimentos que pasarán a formar parte de la bbdd del sistema.

3.7.3 Mockups previos a la implementación del frontend

De forma previa a la fase de codificación de la capa de presentación, se llevó a cabo una etapa de prototipado visual. El objetivo principal de diseñar estos *mockups* fue establecer una primera aproximación conceptual a la interfaz gráfica, definiendo la jerarquía de la información, la disposición geométrica de los componentes (como los formularios de datos biométricos, el historial de menús y el panel de configuración) y el flujo de navegación sin el coste de tiempo asociado al desarrollo directo en código. Esta planificación visual resultó fundamental para alinear los requisitos funcionales del sistema con los principios de usabilidad y experiencia de usuario (UX). A continuación, se presentan los diseños preliminares que sirvieron como plano arquitectónico para la construcción definitiva del *frontend*:

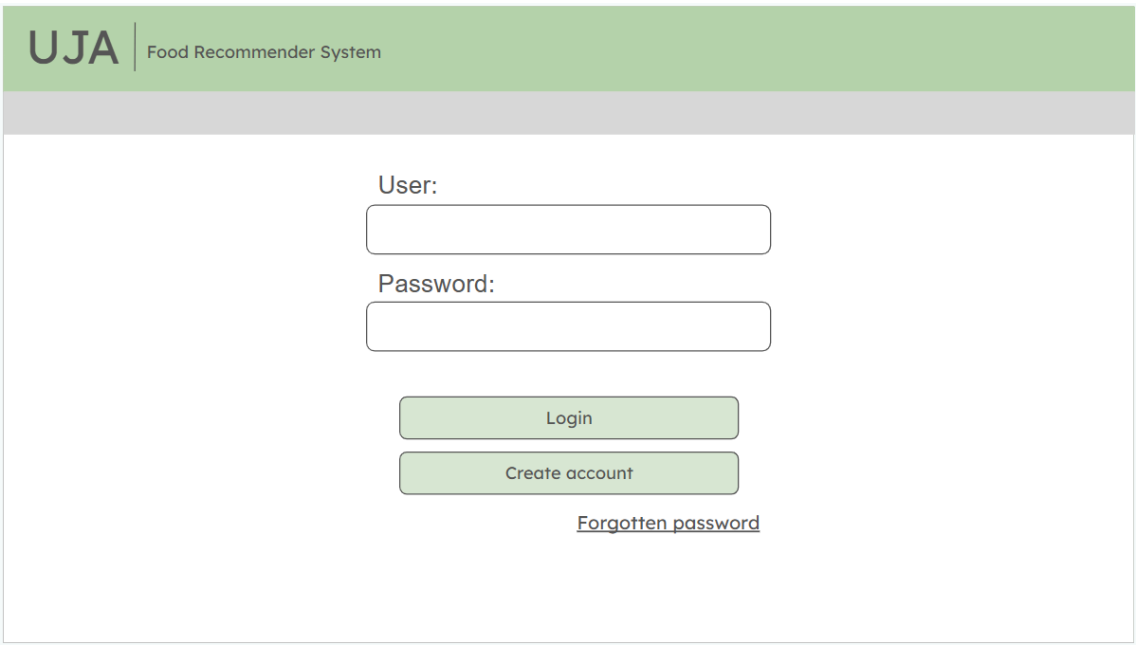


Figura 3.5 Inicio de sesión.

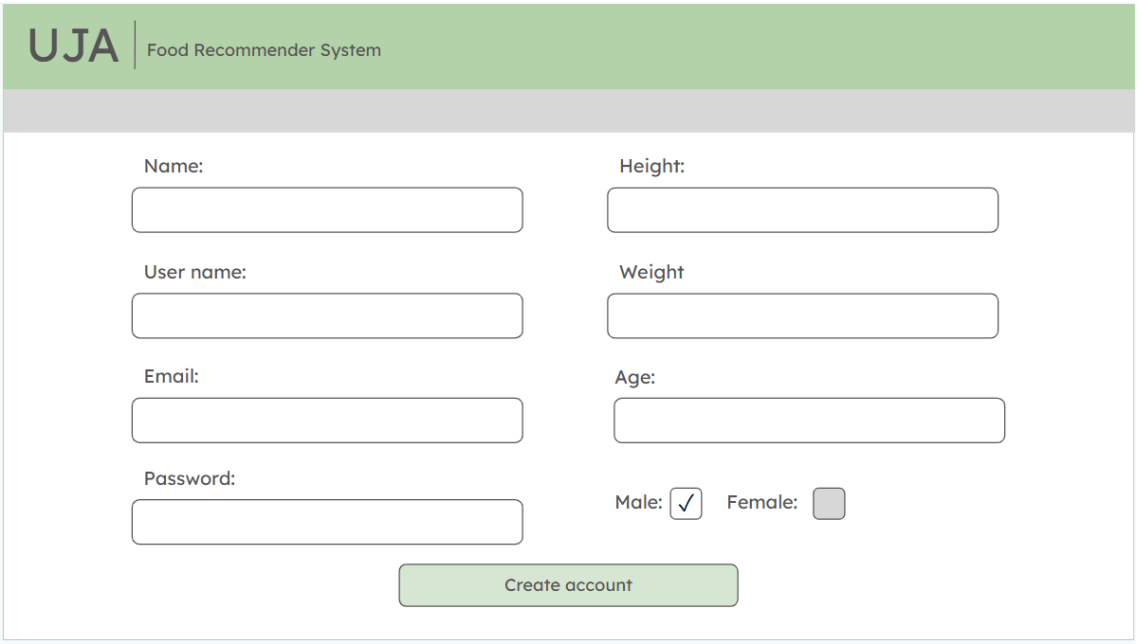


Figura 3.6 Creación de una cuenta.

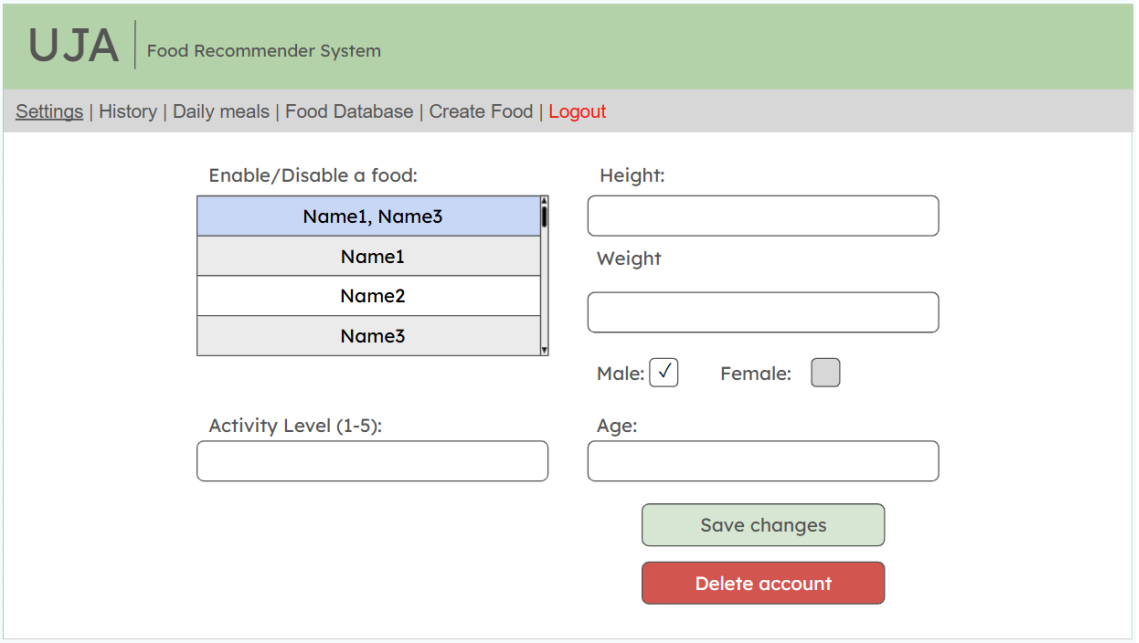


Figura 3.7 Pantalla para la visualización y modificación de los datos del usuario logueado.

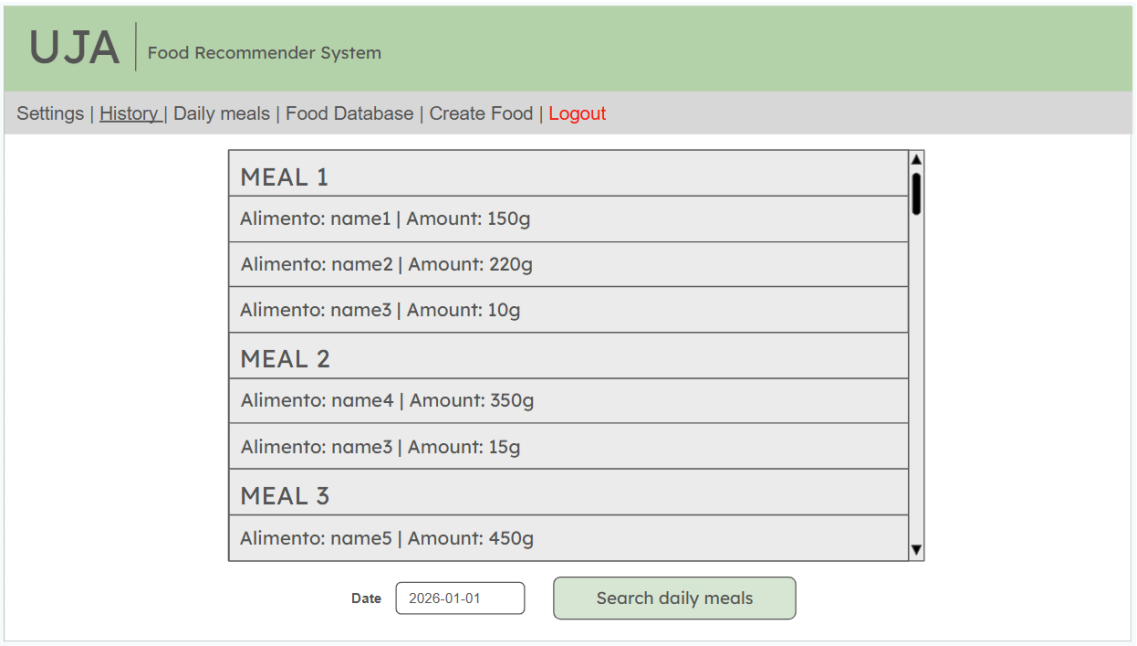


Figura 3.8 Pantalla que nos permite consultar el menú del día indicado.

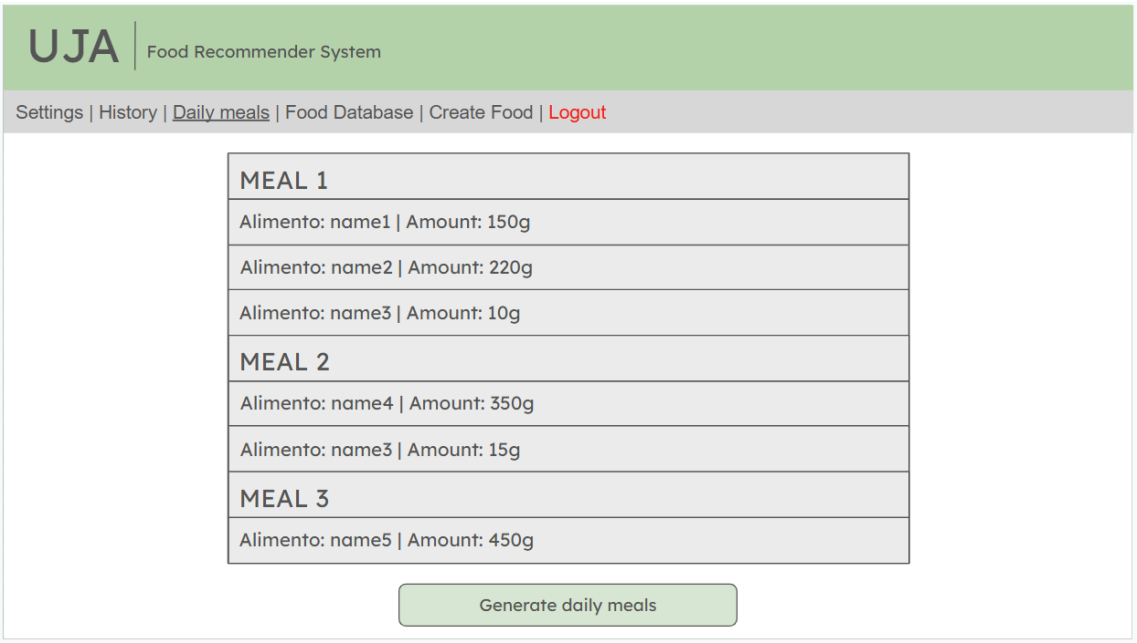


Figura 3.9 Pantalla utilizada para generar el menú diario.

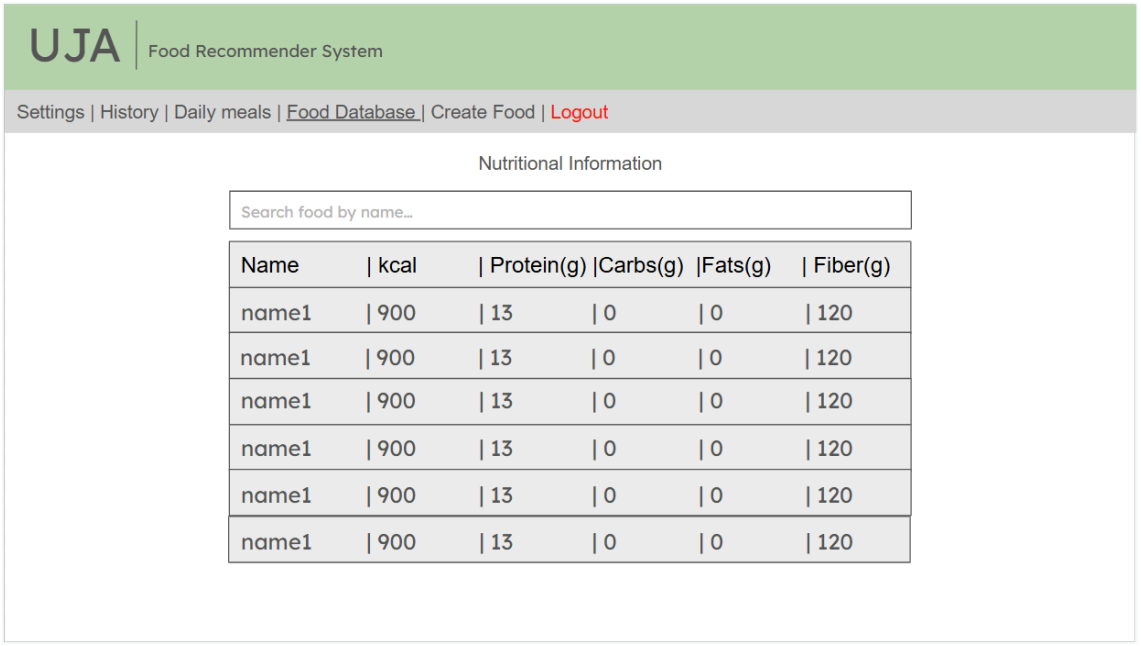


Figura 3.10 Pantalla utilizada para consultar los datos de un alimento concreto.

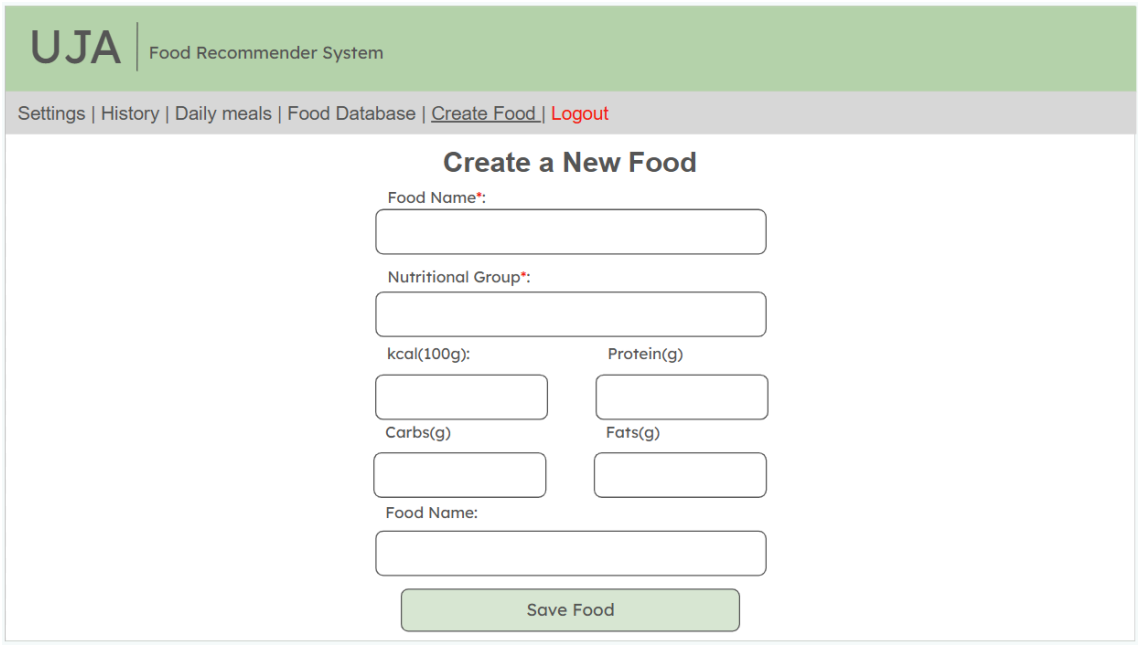


Figura 3.11 Pantalla para crear un nuevo alimento y añadirlo a los contemplados en la creación de menús.

3.8. Interacción y Dinámica del Sistema

Para comprender el funcionamiento end-to-end del sistema, es ilustrativo analizar el caso de uso central: **"Generar Menú Diario"**.

1. **Interacción del Usuario**: El usuario, desde la pestaña "Daily meals", hace clic en el botón "Generate daily meals".
2. **Petición del Cliente**: JavaScript invoca una petición HTTP POST asíncrona hacia el endpoint /menus/generate/{id}.
3. **Recepción en Backend**: El servidor Spring Boot intercepta la llamada. Verifica la existencia del usuario y acude a la capa de Aplicación (GenerateMenuUseCase).
4. **Procesamiento Inteligente**: Se invoca el adaptador de optimización. El backend extrae los alimentos, genera el contexto JSON y ejecuta el script Python. El motor matemático calcula las combinaciones exactas y devuelve el resultado.
5. **Persistencia**: La estructura devuelta (Menú, Comidas y Comidas-Alimentos con sus respectivos gramos) se guarda en la base de datos relacional mediante Spring Data JPA.
6. **Respuesta y Renderizado**: El servidor transforma las entidades guardadas en un MenuOutputDto y lo devuelve al cliente con un código HTTP 200 (OK). La función renderMenuHTML() en el Frontend procesa este JSON y construye elementos HTML (div, span) dinámicamente para mostrar al usuario la planificación del día, detallando las Kcal, Proteínas, Grasas e Hidratos de Carbono de cada ración.

Capítulo 4 Implementación y pruebas

En este capítulo se lleva a cabo una explicación del entorno de desarrollo y tecnologías utilizadas junto a las pruebas llevadas a cabo para comprobar el correcto funcionamiento del sistema y dos casos de estudio para un mejor entendimiento del mismo.

4.1. Entorno de Desarrollo y Tecnologías

La fase de implementación ha requerido un ecosistema de herramientas de desarrollo moderno, capaz de integrar múltiples lenguajes bajo un único flujo de trabajo. Para este proyecto, el entorno se configuró de la siguiente manera:

- **Backend (Java)**: Se ha utilizado el entorno de desarrollo integrado (IDE) **IntelliJ IDEA** de JetBrains. El lenguaje base ha sido **Java 21**, aprovechando sus últimas características de rendimiento. El esqueleto del proyecto se construyó mediante **Spring Boot 3.5.7**, apoyado en **Maven** para la gestión del ciclo de vida y la resolución de dependencias.
- **Base de Datos y Persistencia**: Se integró **Hibernate** a través del módulo **Spring Data JPA**. Esto permitió interactuar con la base de datos subyacente utilizando programación orientada a objetos en lugar de consultas SQL crudas.
- **Motor Matemático (Python)**: Para el algoritmo de generación, se requirió la instalación de un entorno local de Python 3, configurando gestores de paquetes como pip para incorporar librerías científicas como numpy y scipy.
- **Frontend**: Codificado mediante editores de texto ligero, empleando **HTML5, CSS3 y JavaScript moderno (ES6+)**.

4.2. Implementación del Backend (Spring Boot)

El desarrollo del servidor centró la mayor parte del esfuerzo en la estandarización del código y la seguridad de los datos a través del uso exhaustivo de Anotaciones y Mapeos.

1. **Gestión de Entidades**: Mediante la librería **Lombok**, se eliminó el código repetitivo. Anotaciones como @Getter, @Setter, @NoArgsConstructor y @AllArgsConstructor inyectan automáticamente los constructores y métodos de acceso en el momento de la compilación para entidades como UserEntity o MealEntity.
2. **Transferencia Segura de Datos (DTOs)**: Se implementó un barrera entre la base de datos y el cliente a través de los DTOs. Por ejemplo, al registrar un usuario, el cliente envía un UserInputDto. En la implementación, este DTO es sometido a validaciones estructurales (ej. @Email, @Size(min=6, max=12) en la contraseña). Posteriormente, MapStruct compila automáticamente las clases UserDtoMapper y UserJpaMapper para transformar estos DTOs de entrada en entidades del dominio, y estas, a su vez, en entidades JPA para su almacenamiento.
3. **Manejo de Operaciones Relacionales**: Las operaciones complejas, como añadir un alimento excluido a un usuario, se han resuelto eficientemente a nivel de base de datos. Por ejemplo, en UpdateUserServiceImpl, se extrae la entidad de usuario, se recupera la entidad de alimento (ReadFoodService), se añade al conjunto (Set) y se confía en la cascada de JPA para persistir la tabla intermedia.

4.3. Implementación del Frontend

La implementación de la capa web se guió por el principio de mínima fricción. Sin la utilización de Node.js ni empaquetadores como Webpack, la estructura es directa y fácilmente desplegable en cualquier servidor web (Apache, Nginx).

- **Gestión de Sesiones**: En index.html (Login), la respuesta asíncrona de autenticación desencadena la orden localStorage.setItem('currentUser', JSON.stringify(userDto)). Esto establece una sesión "stateless" (sin estado en el servidor) gobernada por el lado del cliente, desde donde dashboard.html extrae atributos como el currentUserId al cargar la ventana.
- **Interfaz Reactiva Basada en Eventos**: Para el buscador de alimentos, se implementó una función filterTable(inputId, tableId) enlazada al evento onkeyup del *input* de búsqueda. Esta función itera sobre los elementos del DOM, evaluando las coincidencias de texto e inyectando style.display = "none" para ocultar instantáneamente los elementos que no coincidan, ofreciendo una experiencia en tiempo real.

4.4. Pruebas del Sistema

Durante el desarrollo del "Sistema de generación inteligente de menús nutricionales personalizados" se han aplicado diferentes tipos de pruebas para asegurar la calidad del software y verificar que todas las funcionalidades implementadas cumplen con los requisitos funcionales previamente establecidos (RF-01 a RF-09). Las pruebas han abarcado tanto la integración de los distintos componentes como la validación funcional del sistema.

1. **Pruebas de API**: Se elaboró una extensa Colección de Postman. Esto permitió evaluar el backend de forma independiente al frontend, simulando peticiones GET, POST, PUT y DELETE con cuerpos JSON para verificar la correcta emisión de los códigos HTTP (200 OK, 400 Bad Request por fallos de validación, y 404 Not Found gestionados por el CustomExceptionHandler).
2. **Pruebas de Integración**: Se validó repetidamente la comunicación con el script de Python, ajustando el formato de salida JSON y comprobando el comportamiento del sistema ante usuarios sintéticos (ej. perfiles de edad extrema o niveles de actividad irreales) para certificar que las restricciones del algoritmo AHPSort se aplican debidamente.

Tipo de prueba	Número de pruebas diferentes	Ejemplos de lo probado	Requisitos funcionales cubiertos
Registro e inicio de sesión	4	Creación de usuario con datos válidos, registro con email ya existente, inicio de sesión correcto, y fallo por contraseña incorrecta.	RF-01, RF-02
Gestión del perfil biométrico	3	Actualización de peso/altura con valores válidos, introducción de valores negativos (ej. peso -10kg), y cambio de nivel de actividad.	RF-04
Gestión de catálogo de alimentos	3	Búsqueda de un alimento existente, exclusión de un alimento para un usuario concreto, e inserción de un nuevo alimento con macronutrientes válidos e inválidos.	RF-08, RF-09
Generación inteligente de menú	4	Generación exitosa de menú diario, fallo por restricciones calóricas imposibles, verificación de la correcta distribución de macronutrientes, y exclusión efectiva de alimentos no deseados.	RF-07
Consulta de histórico de menús	2	Búsqueda de menú mediante una fecha con registros existentes, y	RF-06

		búsqueda en una fecha sin menú generado.	
Control de acceso y seguridad	2	Intento de acceso a rutas protegidas sin sesión iniciada, y verificación de la correcta eliminación de la cuenta del usuario.	RF-05, RT-02
Total aproximado	**18 pruebas diferentes**		**RF-01 a RF-09**

Tabla 4.1 Tipos de pruebas

4.5 Casos de estudio

A continuación expondré distintos casos de estudio prácticos. Estos escenarios evalúan el comportamiento del sistema frente a perfiles de usuario contrastantes, introduciendo variaciones en sus parámetros biométricos (como edad, peso, altura y nivel de actividad física).

4.5.1 Caso 1

A. Caracterización del Usuario y Requerimientos Metabólicos

En este escenario se evalúa el comportamiento del sistema ante un usuario masculino de 35 años, con un peso corporal de 100 kg, una altura de 190 cm y un nivel de actividad física catalogado con el valor 4 (alta actividad). Para determinar el gasto energético, el sistema procesa estas variables biométricas a través de la ecuación clínica de Mifflin-St Jeor:

$$BMR = 10 * 100 + 6.25 * 190 - 5 * 35 + 5 = 2017.5 \text{ kcal/día}$$

A continuación, este metabolismo basal se ajusta mediante el factor multiplicador correspondiente a su nivel de actividad (aproximadamente 1.55 para un nivel 4), lo que sitúa su requerimiento energético real (*Target Kcal*) en

3127.12 kcal/día. Para asegurar el equilibrio metabólico, el sistema divide esta energía en la siguiente distribución de macronutrientes:

- **Carbohidratos (50%):** 1563.5 kcal → 390.8 g.
- **Proteínas (20%):** 625.4 kcal → 156.3 g.
- **Lípidos (30%):** 938.1 kcal → 104.2 g.

A nivel estructural, el algoritmo planifica estas calorías sobre la plantilla diaria base de la aplicación: 15% para el Desayuno (469 kcal), 45% para el Almuerzo (1407 kcal) y 40% para la Cena (1251 kcal).

B. Resultados obtenidos

Como se observa en la planificación generada, el motor logra cubrir la alta demanda de 3127 kcal diarias combinando inteligentemente diversas fuentes nutricionales y gramajes de precisión:

- **Desayuno:** Propone un bloque hipercalórico rápido apoyado en Lácteos (Queso Gervais, 200g) y Frutas (Kiwi y Zumo de Piña) junto con Pasta lionesa.
- **Almuerzo:** Para abarcar la ingesta principal, el modelo selecciona múltiples fuentes proteicas (Cordero chuleta 125g, Pollo corazón 125g) y carbohidratos densos (Harina de arroz 130g), apoyados por vegetales (Soja y Pimientos).
- **Cena:** Ajusta el cierre calórico con Arroz blanco (130g), Mortadela (125g) y vegetales fibrosos (Puerros, Cebolla, Judías tiernas).

Es destacable que la suma calórica total del menú recomendado se sitúa en torno a las **3163.5 kcal**. Esta cifra demuestra que el modelo de relajación del sistema opera correctamente: al aplicar el parámetro alpha = 75.0, el algoritmo de optimización permite una desviación mínima frente al objetivo teórico de 3127 kcal. Esto evita que el sistema falle por "solución infactible", logrando generar raciones lógicas y realistas (125g, 150g, 200g) que maximizan la preferencia del usuario sin vulnerar sus requerimientos fisiológicos subyacentes.

C. Evidencia Visual del Menú Generado

MEAL 1 - BREAKFAST

Food: KIWI
Recommended serving: 150g
Actual contribution: 76.5 kcal | Pro: 1.5g | Carbs: 13.6g | Fats: 0.9g

Food: QUESO Gervais natural Danone
Recommended serving: 200g
Actual contribution: 532.0 kcal | Pro: 13.8g | Carbs: 7.6g | Fats: 49.8g

Food: ZUMO DE PIÑA
Recommended serving: 150g
Actual contribution: 76.5 kcal | Pro: 0.6g | Carbs: 18.1g | Fats: 0.2g

Food: PASTA LIONESA (Choux) cocida
Recommended serving: 60g
Actual contribution: 198.0 kcal | Pro: 5.1g | Carbs: 18.4g | Fats: 12.1g

MEAL 2 - LUNCH

Food: CORDERO CHULETA
Recommended serving: 125g
Actual contribution: 281.3 kcal | Pro: 22.5g | Carbs: 0g | Fats: 21.3g

Food: HARINA DE ARROZ
Recommended serving: 130g
Actual contribution: 475.8 kcal | Pro: 8.3g | Carbs: 104.1g | Fats: 1.0g

Food: SOJA, brotes
Recommended serving: 130g
Actual contribution: 65.0 kcal | Pro: 7.2g | Carbs: 6.1g | Fats: 1.3g

Food: POLLO CORAZÓN
Recommended serving: 125g
Actual contribution: 165.0 kcal | Pro: 21.6g | Carbs: 0g | Fats: 7.3g

Food: PIMIENTOS DEL PIQUILLO
Recommended serving: 200g
Actual contribution: 44.0 kcal | Pro: 2.4g | Carbs: 7.6g | Fats: 0.4g

Food: ZUMO DE PIÑA
Recommended serving: 150g
Actual contribution: 76.5 kcal | Pro: 0.6g | Carbs: 18.1g | Fats: 0.2g

MEAL 3 - DINNER

Food: ARROZ BLANCO
Recommended serving: 130g
Actual contribution: 460.2 kcal | Pro: 9.9g | Carbs: 100.1g | Fats: 2.2g

Food: MORTADELA
Recommended serving: 125g
Actual contribution: 387.5 kcal | Pro: 17.5g | Carbs: 3.8g | Fats: 26.3g

Food: KIWI
Recommended serving: 150g
Actual contribution: 76.5 kcal | Pro: 1.5g | Carbs: 13.6g | Fats: 0.9g

Food: PUERROS
Recommended serving: 200g
Actual contribution: 84.0 kcal | Pro: 4.0g | Carbs: 15.0g | Fats: 0.8g

Food: CEBOLLA
Recommended serving: 200g
Actual contribution: 94.0 kcal | Pro: 2.8g | Carbs: 20.0g | Fats: 0.4g

Food: JUDÍAS TIERNAS
Recommended serving: 130g
Actual contribution: 50.7 kcal | Pro: 3.1g | Carbs: 9.1g | Fats: 0.3g

Figura 4.1 Menú caso de estudio 1

4.5.2 Caso 2

A. Caracterización del Usuario y Requerimientos Metabólicos

Este segundo escenario evalúa el sistema frente a un perfil femenino joven de 18 años, con un peso de 54 kg, una altura de 150 cm y un nivel de actividad física catalogado con el valor 1 (sedentario). El backend procesa estos parámetros biométricos aplicando la variante femenina de la ecuación clínica de Mifflin-St Jeor, la cual incorpora la constante de ajuste metabólico (-161):

BMR = 10 * 54 + 6.25 * 150 - 5 * 18 - 161 = 1226.5 kcal/día

Al multiplicar esta Tasa Metabólica Basal por el factor de actividad sedentaria (1.2), el sistema establece un requerimiento energético objetivo (*Target Kcal*) de **1471.8 kcal/día**. De acuerdo con las reglas de negocio implementadas, este volumen calórico se distribuye en los siguientes objetivos de macronutrientes:

- **Carbohidratos (50%):** 735.9 kcal → 183.9 g.
- **Proteínas (20%):** 294.36 kcal → 73.6 g.
- **Lípidos (30%):** 441.54 kcal → 49.0 g.

Siguiendo la distribución estructural por ingestas, las metas teóricas se fijan en un 15% para el Desayuno (220.7 kcal), 45% para el Almuerzo (662.3 kcal) y 40% para la Cena (588.7 kcal).

B. Resultados obtenidos

El menú resultante asigna los siguientes gramajes:

- **Desayuno:** Incorpora lácteos (Yogur natural enriquecido 200g, Crema choco/nata 200g) y carbohidratos (Ganchitos 60g), complementados con fruta (Arándano 150g).
- **Almuerzo:** Estructura una comida basada en legumbres (Garbanzos secos 130g) y vegetales (Habas frescas 130g, Tomate triturado 200g), junto a fuentes proteicas y lipídicas densas (Sesos 125g, Pies de cerdo 125g), finalizando con Macedonia (150g).

- **Cena:** Plantea una ingesta rica en productos del mar (Cigala 125g, Sardina en tomate 200g) combinada con legumbres (Lentejas secas 130g), vegetales (Cebolla tierna 200g, Guisantes frescos 130g) y Uva (150g).

D. Evidencia Visual del Menú Generado

MEAL 1 - BREAKFAST

Food: YOGUR nat.enriq.azúcar Danone
Recommended serving: 200g
Actual contribution: 192.0 kcal | Pro: 6.0g | Carbs: 29.0g | Fats: 6.4g

Food: GANCHITOS
Recommended serving: 60g
Actual contribution: 302.4 kcal | Pro: 5.9g | Carbs: 34.1g | Fats: 15.7g

Food: CREMA choco/nata lig. Danone
Recommended serving: 200g
Actual contribution: 166.0 kcal | Pro: 7.8g | Carbs: 19.8g | Fats: 6.6g

Food: ARÁNDANO
Recommended serving: 150g
Actual contribution: 79.5 kcal | Pro: 1.2g | Carbs: 19.6g | Fats: 0.9g

MEAL 2 - LUNCH

Food: HABAS FRESCAS
Recommended serving: 130g
Actual contribution: 83.2 kcal | Pro: 7.0g | Carbs: 13.0g | Fats: 0.4g

Food: GARBANZOS SECOS
Recommended serving: 130g
Actual contribution: 469.3 kcal | Pro: 23.4g | Carbs: 79.3g | Fats: 6.5g

Food: SESOS
Recommended serving: 125g
Actual contribution: 141.3 kcal | Pro: 12.9g | Carbs: 0g | Fats: 10.0g

Food: TOMATE TRITURADO Conserva
Recommended serving: 200g
Actual contribution: 78.0 kcal | Pro: 4.6g | Carbs: 11.0g | Fats: 1.0g

Food: PIES DE CERDO
Recommended serving: 125g
Actual contribution: 362.5 kcal | Pro: 20.0g | Carbs: 0.6g | Fats: 31.3g

Food: MACEDONIA
Recommended serving: 150g
Actual contribution: 153.0 kcal | Pro: 0.4g | Carbs: 37.5g | Fats: 0.2g

MEAL 3 - DINNER

Food: CIGALA
Recommended serving: 125g
Actual contribution: 83.8 kcal | Pro: 18.8g | Carbs: 0g | Fats: 1.0g

Food: SARDINA EN TOMATE
Recommended serving: 200g
Actual contribution: 368.0 kcal | Pro: 33.6g | Carbs: 1.0g | Fats: 25.6g

Food: CEBOLLA TIERNA
Recommended serving: 200g
Actual contribution: 78.0 kcal | Pro: 1.8g | Carbs: 16.8g | Fats: 0.4g

Food: LENTEJAS SECAS
Recommended serving: 130g
Actual contribution: 436.8 kcal | Pro: 31.2g | Carbs: 72.8g | Fats: 2.3g

Food: GUISANTES FRESCOS
Recommended serving: 130g
Actual contribution: 119.6 kcal | Pro: 7.8g | Carbs: 20.8g | Fats: 0.5g

Food: UVA
Recommended serving: 150g
Actual contribution: 121.5 kcal | Pro: 1.5g | Carbs: 25.5g | Fats: 1.5g

Figura 4.2 Menú caso de estudio 2

Capítulo 5 - Conclusiones y trabajos futuros

En este capítulo se presentan las conclusiones que exponen cómo se han alcanzado los objetivos del proyecto, y algunos de los trabajos futuros que podrían ampliar las capacidades del sistema.

5.1. Conclusiones

El desarrollo del "Sistema de generación inteligente de menús nutricionales personalizados" supone una prueba tangible de cómo la convergencia de la Ingeniería de Software tradicional y los ecosistemas matemáticos avanzados puede resolver problemas del mundo real en el marco del concepto *Food Computing*.

Tras finalizar la implementación y las fases de prueba, se pueden extraer las siguientes conclusiones fundamentales, directamente vinculadas a los objetivos propuestos al inicio del documento:

1. **Integración Tecnológica Exitosa**: Se ha demostrado la viabilidad técnica de la interoperabilidad entre un entorno transaccional fuertemente tipado (Java/Spring Boot) y un entorno de análisis de datos heurísticos (Python) utilizando adaptadores de procesos sin que ello penalice la experiencia del usuario final en la interfaz web.
2. **Cumplimiento Funcional Integral**: La aplicación web permite satisfactoriamente que los usuarios se registren, modifiquen sus parámetros biométricos y configuren sus preferencias alimenticias. Además, los administradores (o la lógica del sistema) disponen de módulos completos para mantener un catálogo dinámico de alimentos.
3. **Generación Personalizada**: El núcleo de recomendación ha logrado el objetivo más complejo: trasladarse desde el enfoque predictivo ("lo que al usuario le apetece") hacia el enfoque prescriptivo ("lo que el usuario debe consumir" adaptado por la fórmula de Harris-Benedict), calculando no solo los platos, sino raciones en gramos para evitar tanto los déficits nutricionales como la fatiga dietética.

5.2. Trabajos Futuros

Si bien el proyecto actual conforma un Producto Mínimo Viable (MVP) robusto, el dominio de la e-health y la nutrición personalizada se encuentra en constante evolución. Se identifican las siguientes líneas de trabajo futuro para escalar las capacidades del sistema:

1. **Evolución del Frontend a un Framework Reactivo**: A medida que la complejidad del sistema crezca (por ejemplo, manejando métricas en tiempo real a lo largo de un año), será necesario migrar la capa de presentación de Vanilla JS a un framework moderno como **React** o **Angular**. Esto mejorará drásticamente la gestión del estado global y permitirá modularizar la UI en componentes reutilizables.
2. **Despliegue de una Aplicación Móvil Nativa**: Dado el carácter diario de las consultas alimentarias, sería de gran valor el desarrollo de una aplicación para Android/iOS utilizando tecnologías cruzadas como *Flutter* o *React Native*.
3. **Integración Directa con Hardware IoT (Wearables)**: Actualmente, el nivel de actividad, peso y altura son valores estáticos introducidos manualmente por el usuario. Un gran avance sería la integración de APIs de terceros (como Apple HealthKit o Google Fit) para extraer la frecuencia cardiaca o los pasos diarios de relojes inteligentes, permitiendo al backend re-calcular la ingesta calórica necesaria de manera dinámica cada 24 horas.
4. **Machine Learning y Deep Learning**: Sustituir el método heurístico/programación lineal por redes neuronales especializadas en filtrado colaborativo avanzado y procesamiento de lenguaje natural. Esto permitiría al sistema aprender progresivamente si un usuario "rechaza" habitualmente los menús recomendados, optimizando la variable de tolerancia al aburrimiento (*fatiga dietética*) con una precisión muy superior.

Anexo I

Manual de Instalación y Despliegue del Sistema

El "Sistema de generación inteligente de menús nutricionales personalizados" utiliza una arquitectura Cliente-Servidor separada en tres bloques principales: la interfaz de usuario (Frontend), la lógica de negocio (Backend en Java) y el motor de optimización matemática (Python).

A continuación, se detallan los requisitos y pasos necesarios para instalar y configurar la aplicación web.

1. Requisitos Previos del Sistema

Para el correcto funcionamiento y desarrollo de la aplicación, es necesario contar con el siguiente entorno de hardware y software:

- Un ordenador personal que permita la instalación del software requerido.
- Un sistema operativo compatible, como por ejemplo Windows 11.
- Entorno de desarrollo integrado (IDE), siendo recomendado IntelliJ IDEA de JetBrains.
- Java 21 instalado en el sistema.
- Python 3 instalado y configurado en las variables de entorno del sistema.

2. Python

El núcleo algorítmico del sistema está implementado en Python y utiliza técnicas de Programación Lineal Entera Mixta (MILP). Para que el backend pueda comunicarse con él correctamente, debes preparar el entorno local:

- Asegúrate de disponer de un entorno local de Python 3.
- Configura un gestor de paquetes como pip en tu sistema.
- Instala las librerías científicas requeridas ejecutando el siguiente comando en tu terminal: pip install numpy scipy. Este paso es estrictamente necesario para ejecutar el script de Python encargado de la generación de menús.

- Verifica que el script generador_de_menus.py se encuentre en la ruta correcta esperada por el servidor, ya que Java instancia un proceso del sistema operativo mediante ProcessBuilder para ejecutarlo.

3. Backend (Java / Spring Boot)

El servidor procesa la lógica de negocio, gestiona la seguridad y administra el acceso a la base de datos. Para poder ejecutar en local el backend es necesario clonar el proyecto desde el repositorio de gitlab al cuál se dará acceso mediante petición.

Características del proyecto:

- El backend de la aplicación debe compilarse utilizando el framework Spring Boot en su versión 3.5.7.
- El ciclo de vida del proyecto y la resolución de sus dependencias se gestiona mediante Maven.
- El proyecto hereda la configuración estructural de spring-boot-starter-parent en el archivo de configuración de Maven.
- El sistema emplea Spring Data JPA basado en Hibernate para las transacciones con la base de datos.
- La persistencia de datos se realiza sobre una base de datos relacional H2. No es necesario instalar un motor de base de datos externo complejo, ya que H2 se ejecutará según la configuración de tu entorno Spring Boot.

Todas las configuraciones están preestablecidas al clonar el proyecto por lo que para que este funcione correctamente se recomienda utilizar intelligentJ para su ejecución y establecer en “Project Structure” java 21 como se muestra en la siguiente imagen.

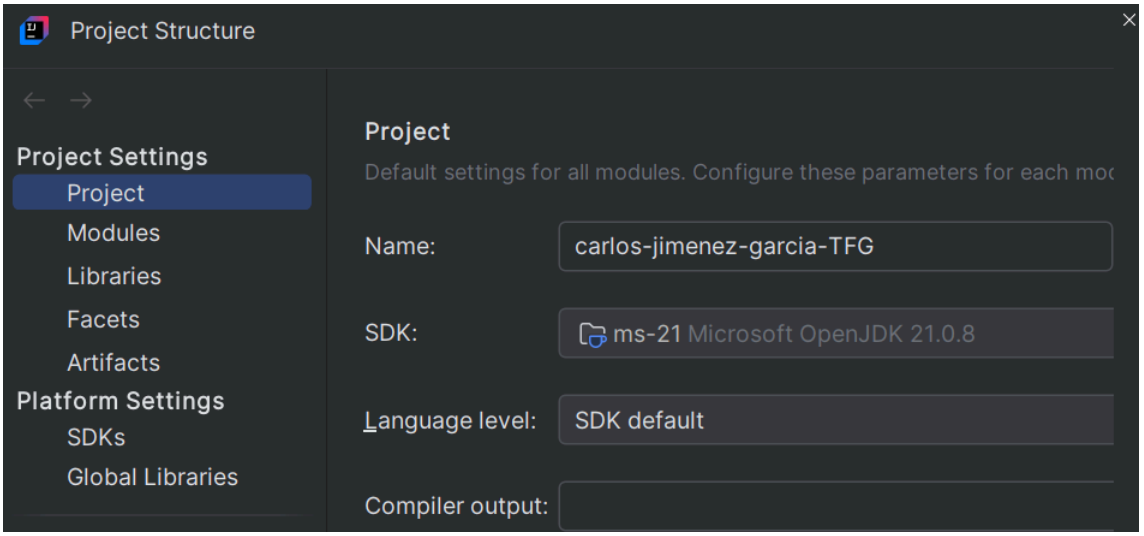


Figura Anexo I - 1

4. Frontend

La capa de presentación ha sido diseñada para ser ligera y simular una Single Page Application (SPA) sin depender de frameworks pesados ni externos como Node.js. Por lo que la ejecución del proyecto desde intelligentJ va acompañada del frontend, por lo tanto basta con acceder al puerto 8080 de nuestro ordenador desde el navegador para poder interactuar con el frontend siempre y cuando la aplicación esté ejecutándose correctamente.

Anexo II

Guía de Uso: Sistema Inteligente de Generación de Menús

A continuación, se detallan las funcionalidades principales y cómo interactuar con cada sección.

1. Acceso y Registro de Usuarios

Para comenzar a utilizar la plataforma, debes identificarte o crear una cuenta nueva.

- **Inicio de sesión:** Si ya tienes cuenta, el sistema te permite acceder introduciendo tu nombre de usuario y contraseña. También cuentas con una opción para cambiar tu contraseña introduciendo tu email y la nueva clave. Figura Anexo II - Inicio de sesión
- **Creación de una cuenta:** Si eres un usuario nuevo, puedes registrarte rellenando un formulario con tus datos personales y biométricos. El sistema requerirá: nombre, apellidos, nombre de usuario, email, contraseña, edad, altura, sexo, peso y tu nivel de actividad física (del 1 al 5). Figura Anexo II - Creación de cuenta

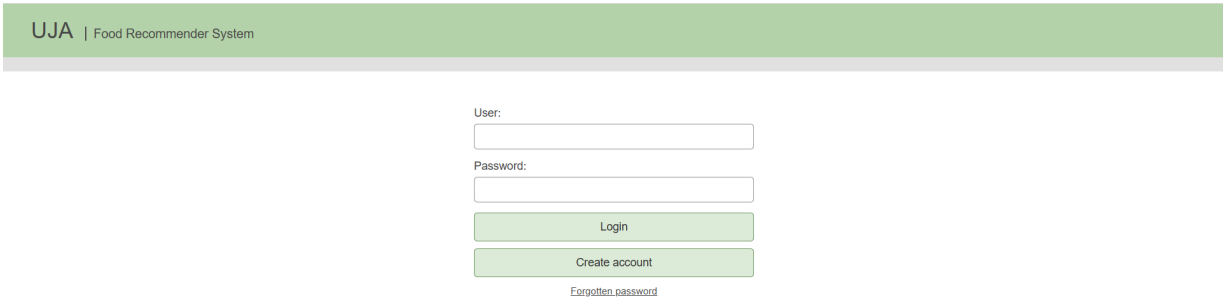


Figura Anexo II - Inicio de sesión

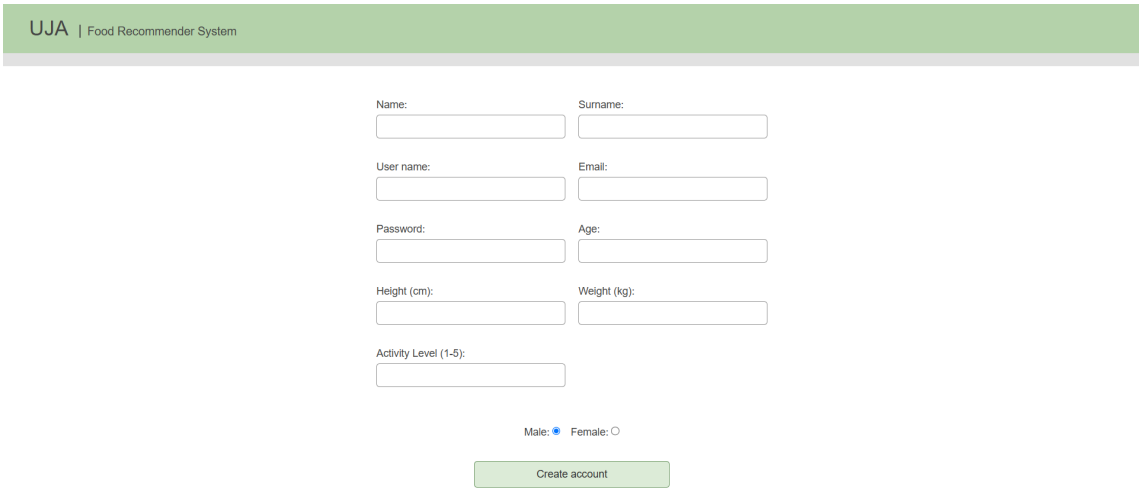


Figura Anexo II - Creación de cuenta

2. Panel de Configuración (Settings)

Una vez iniciada la sesión, podrás gestionar tu perfil y tus preferencias alimentarias desde el panel de ajustes.

- **Actualización de datos:** El sistema te permite modificar en cualquier momento tu información personal y biométrica, incluyendo altura, peso, sexo, edad y nivel de actividad. Estos datos son cruciales para el cálculo de tu tasa metabólica.
- **Preferencias de alimentos:** En esta sección dispones de un buscador para seleccionar y gestionar qué alimentos deseas habilitar o excluir de tu dieta. El sistema permite añadir y eliminar estos elementos asociados a tu perfil para personalizar las recomendaciones.
- **Gestión de la cuenta:** En caso de que lo necesites, también dispones de la opción para eliminar tu cuenta del sistema de forma definitiva.

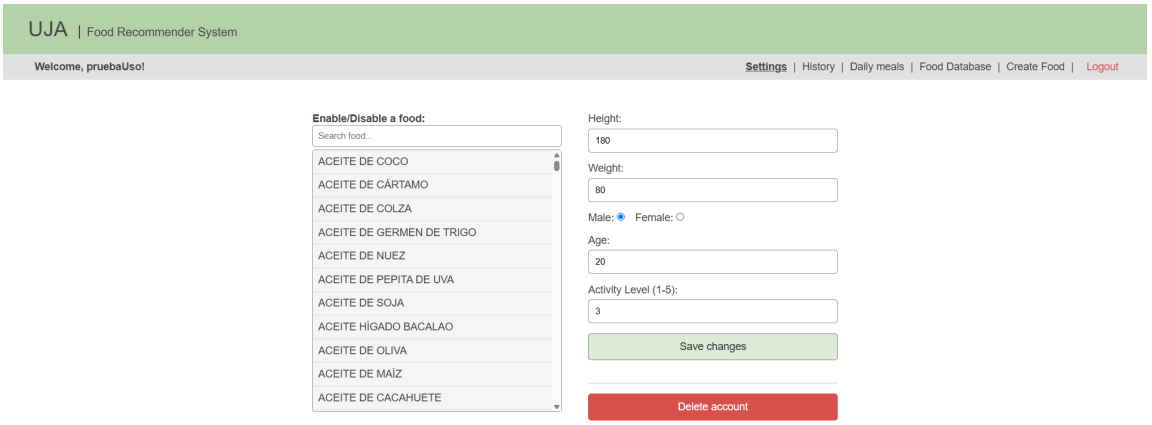


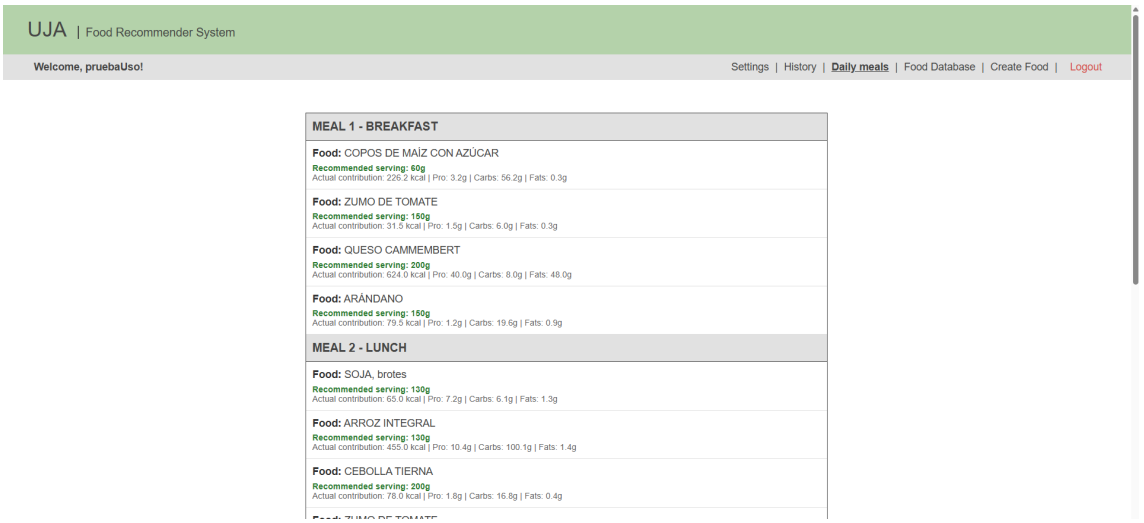
Figura Anexo II - Ajustes del usuario

3. Generación de Menús Diarios (Daily Meals)

Esta es la funcionalidad principal de la aplicación, donde se calculará tu plan alimenticio.

- **Generar nuevo menú:** Dirígete a la pestaña "Daily meals" y haz clic en el botón de generación para que el sistema cree un menú nutricionalmente correcto, adaptado a tus necesidades biológicas y preferencias para el día de hoy.
- **Visualización de raciones:** El menú generado se presentará en tarjetas legibles que desglosan la información clave de cada toma (desayuno, almuerzo, cena). Podrás ver el nombre del alimento, los macronutrientes que aporta (proteínas, grasas, carbohidratos) y la ración exacta recomendada en gramos.

Figura Anexo II - Generación de menú



Fira Anexo II - Menú generado

4. Historial de Menús (History)

Si deseas revisar tu alimentación pasada, el sistema guarda un registro de tus planes diarios.

- **Consulta por fecha:** Puedes acceder a la pestaña de historial para consultar el menú que se generó en un día específico. Solo necesitas indicar la fecha concreta en el buscador para visualizar las comidas de esa jornada.

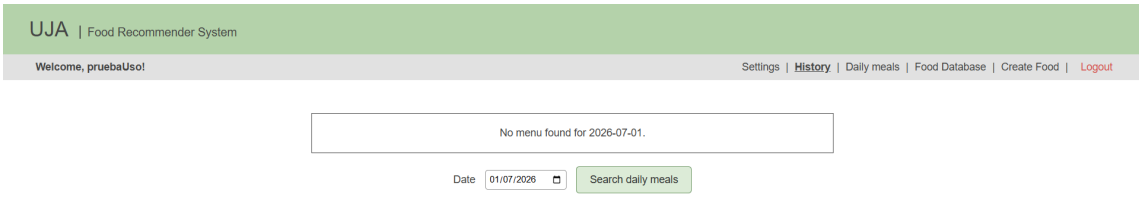


Figura Anexo II - Menú no encontrado

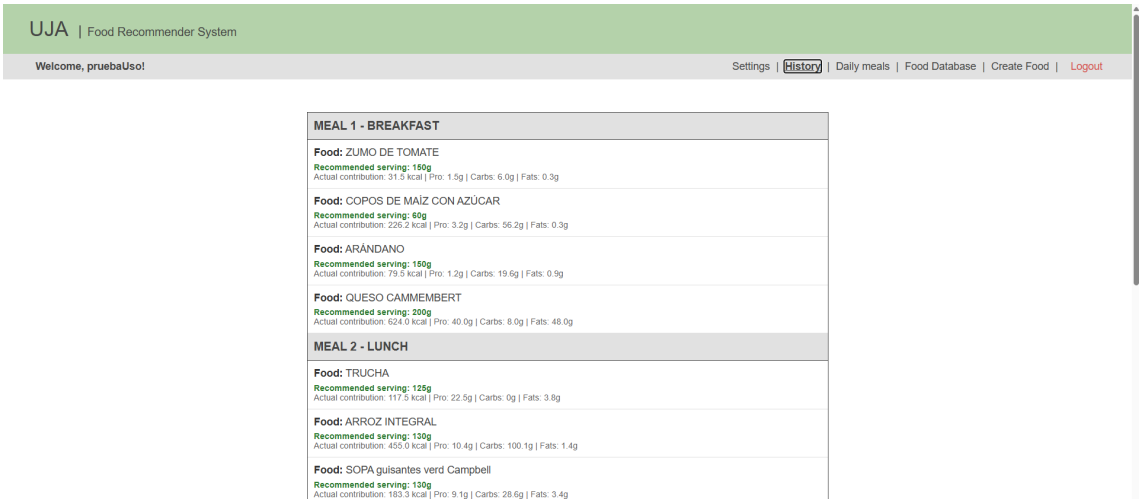


Figura Anexo II - Menú encontrado para una fecha determinada

5. Base de Datos y Creación de Alimentos (Food Database & Create Food)

La aplicación cuenta con un catálogo dinámico de alimentos que puedes consultar y ampliar.

- **Buscador de alimentos (Food Database):** El sistema implementa un buscador que te permite consultar la información nutricional más relevante de todos los alimentos contemplados en el sistema.

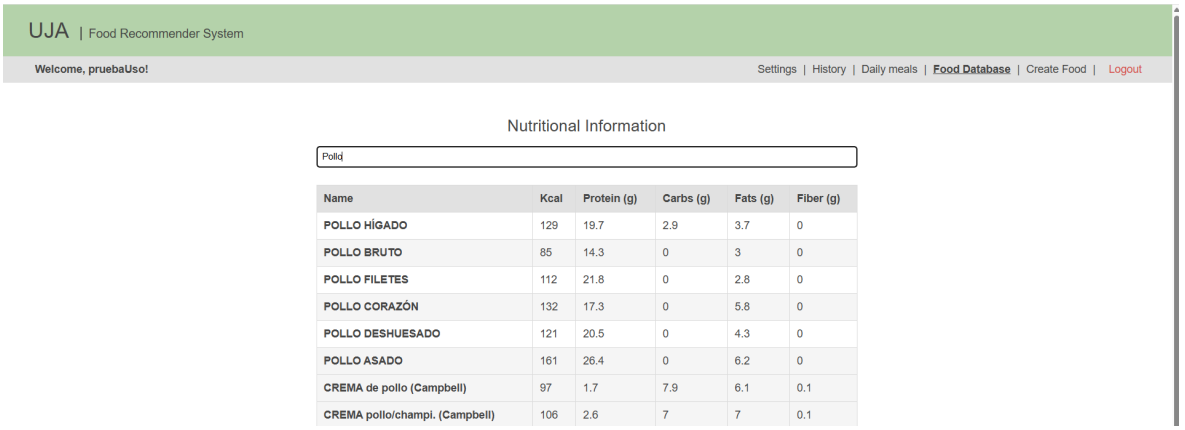


Figura Anexo II - Buscador alimentos

- **Añadir nuevos alimentos (Create Food):** Si echas en falta algún ingrediente, el sistema te permite añadir un nuevo alimento a la base de datos. Para ello, deberás completar un formulario indicando: nombre, grupo nutricional, kilocalorías (kcal), proteínas, carbohidratos, grasas y fibra.

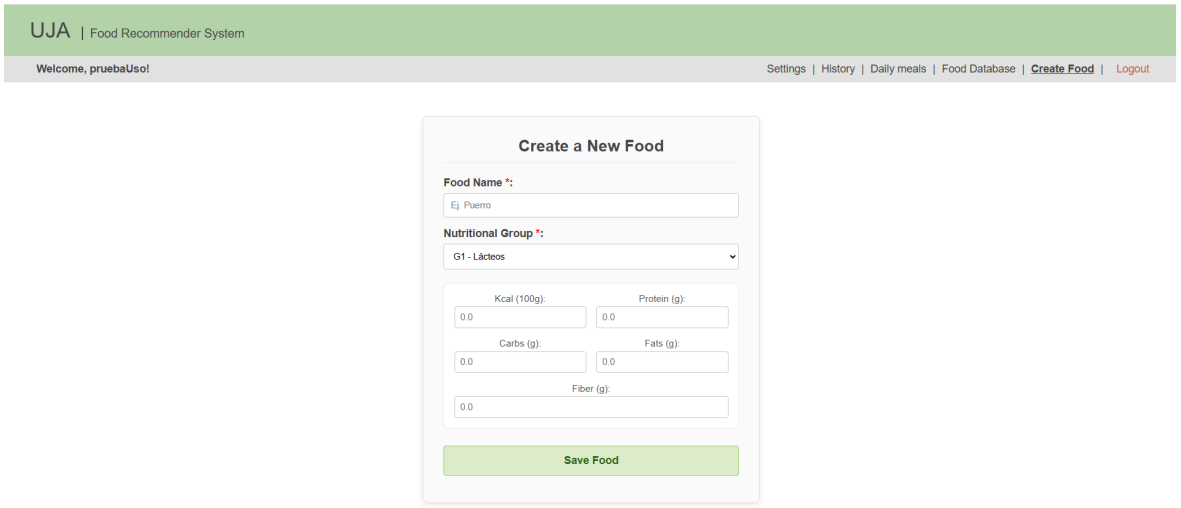


Figura Anexo II - Añadir alimento

Bibliografía

[1] K. Beck *et al.*, "Manifesto for Agile Software Development," 2001. [En línea]. Disponible en: http://agilemanifesto.org/

[2] K. Schwaber y J. Sutherland, "The Scrum Guide," 2020. [En línea]. Disponible en: https://scrumguides.org/

[3] E. Morínigo, "Depreciación, el costo invisible," *Emagister*, 2007. [En línea]. Disponible en: https://www.emagister.com

[4] V. Enciso y Á. R. Peña Cardozo, "Depreciación y amortización," 2022.

[5] Microsoft, "Windows 11," 2024. [En línea]. Disponible en: https://www.microsoft.com/es-es/windows/windows-11?r=1

[6] JetBrains, "JetBrains: Essential tools for software developers and teams." [En línea]. Disponible en: https://www.jetbrains.com/es-es/

[7] G. Adomavicius y A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734-749, Jun. 2005.

[8] G. Agapito *et al.*, "DIETOS: A dietary recommender system for chronic diseases monitoring and management," *Comput. Methods Programs Biomed.*, vol. 153, pp. 93-104, 2018.

[9] J. L. Balintfy, "Menu planning by computer," *Commun. ACM*, vol. 7, no. 4, pp. 255-259, Abr. 1964.

[10] R. Y. Toledo, A. A. Alzahrani, y L. Martínez, "A Food Recommender System Considering Nutritional Information and User Preferences," *IEEE Access*, vol. 7, pp. 96695-96711, 2019.

[11] D. Goldberg, D. Nichols, B. M. Oki, y D. Terry, "Using collaborative filtering to weave an information tapestry," *Commun. ACM*, vol. 35, no. 12, pp. 61-70, Dic. 1992.

[12] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, y J. Riedl, "GroupLens: an open architecture for collaborative filtering of netnews," en *Proc. 1994 ACM Conf. Comput. Supported Cooperative Work*, 1994, pp. 175-186.

[13] W. Min, S. Jiang, L. Liu, Y. Rui, y A. K. Jain, "A survey on food computing," *ACM Comput. Surv.*, vol. 52, no. 5, pp. 1-36, 2019.

[14] C. Trattner y D. Elsweiler, "Food recommender systems: important contributions, challenges and future research directions," *arXiv preprint arXiv:1711.02760*, 2017.

[15] IEEE Computer Society, *IEEE Standard Glossary of Software Engineering Terminology* (IEEE Std 610.12-1990), 1990.

[16] I. Sommerville, *Ingeniería de software*, 9ª ed. Madrid, España: Pearson Educación, 2011.

[17] R. S. Pressman, *Ingeniería del software: un enfoque práctico*, 7ª ed. Ciudad de México, México: McGraw-Hill Interamericana, 2010.

[18] P. Bourque y R. E. Fairley (Eds.), *Guide to the Software Engineering Body of Knowledge (SWEBOK(R))*, Versión 3.0. IEEE Computer Society, 2014.

[19] M. D. Mifflin *et al.*, "A new predictive equation for resting energy expenditure in healthy individuals," *The American Journal of Clinical Nutrition*, vol. 51, no. 2, pp. 241-247, Feb. 1990.

[20] S. Buscemi *et al.*, "Analysis of Predictive Equations for Estimating Resting Energy Expenditure in a Large Cohort of Morbidly Obese Patients," *Frontiers in Endocrinology*, vol. 9, p. 352, 2018.

[21] A. M. Anderson and M. D. Earle, ‘‘Diet planning in the third world by linear and goal programming,’’ J. Oper. Res. Soc., vol. 34, no. 1, pp. 9–16, 1983.