



KCALCONTROL

Ciclo Formativo de Grado Superior
Desarrollo de Aplicaciones Web

2024-2025

AUTOR: Yoannet Díaz Valdés

TUTOR: Paula Carolina Martínez Perea

RESUMEN

Este proyecto tiene como objetivo el desarrollo de una aplicación web orientada a la gestión de la nutrición de los usuarios, permitiéndoles llevar un control de su ingesta calórica. La aplicación cuenta con un sistema de autenticación de usuarios utilizando Firebase, que permite registrar, iniciar sesión y mantener la sesión activa mediante tokens de acceso, asegurando una experiencia de usuario fluida. Además, se ha implementado una API REST en Laravel para gestionar los datos de los perfiles de cada usuarios.

La aplicación web permite a los usuarios calcular sus necesidades diarias de macronutrientes (proteínas, carbohidratos y grasas) en base a su edad, peso, sexo y objetivo personal, utilizando la fórmula de Mifflin-St Jeor. Para ello se ha diseñado una interfaz para que los usuarios puedan añadir, eliminar y editar alimentos en las tres comidas principales del día (desayuno, comida, cena, merienda). Además, se incluye un indicador visual que muestra las calorías consumidas y las que faltan para alcanzar el objetivo diario, facilitando el seguimiento del progreso.

Esta aplicación web integra, además, una API externa que permite buscar y añadir alimentos comunes, proporcionando información sobre los macronutrientes y las calorías de cada alimento. También se ha desarrollado una funcionalidad para almacenar el historial de comidas diarias de los usuarios, con una vista que permite consultar y filtrar las ingestas pasadas por fecha. Estas características permiten a los usuarios tener un control sobre su nutrición contribuyendo al objetivo que quieren alcanzar.

ABSTRACT

This project aims to develop a web application focused on managing users' nutrition, allowing them to track their caloric intake. The application features a user authentication system using Firebase, enabling users to register, log in, and maintain an active session through access tokens, ensuring a smooth user experience. Additionally, a RESTful API has been implemented in Laravel to manage the profile data of each user.

The application allows users to calculate their daily macronutrient needs (proteins, carbohydrates, and fats) based on their age, weight, gender, and personal goal, using the Mifflin-St Jeor formula. To achieve this, an interface has been designed where users can add, remove, and edit foods in the three main meals of the day (breakfast, lunch, and dinner). Moreover, a visual indicator has been included to show the calories consumed and the remaining calories needed to reach the daily goal, facilitating progress tracking.

Furthermore, this web application integrates an external API that enables users to search and add common foods, providing information on the macronutrients and calories of each food item. A feature has also been developed to store the daily meal history of users, with a view that allows them to consult and filter past intakes by date. These features give users control over their nutrition, helping them achieve their desired goals.

RESUM

Aquest projecte té com a objectiu el desenvolupament d'una aplicació web orientada a la gestió de la nutrició dels usuaris, permetent-los portar un control de la seua ingesta calòrica. L'aplicació compta amb un sistema d'autenticació d'usuaris utilitzant Firebase, que permet registrar-se, iniciar sessió i mantenir la sessió activa mitjançant tokens d'accés, assegurant una experiència d'usuari fluida. A més, s'ha implementat una API REST en Laravel per gestionar les dades dels perfils de cada usuari.

L'aplicació permet als usuaris calcular les seues necessitats diàries de macronutrients (proteïnes, carbohidrats i greixos) en funció de la seua edat, pes, sexe i objectiu personal, utilitzant la fórmula de Mifflin-St Jeor. Per a això s'ha dissenyat una interfície perquè els usuaris puguin afegir, eliminar i editar aliments en les tres menjades principals del dia (esmorzar, dinar i sopar). A més, s'inclou un indicador visual que mostra les calories consumides i les que falten per aconseguir l'objectiu diari, facilitant el seguiment del progrés.

Aquesta aplicació web integra, a més, una API externa que permet cercar i afegir aliments comuns, proporcionant informació sobre els macronutrients i les calories de cada aliment. També s'ha desenvolupat una funcionalitat per emmagatzemar l'historial de menjars diaris dels usuaris, amb una vista que permet consultar i filtrar les ingestions passades per data. Aquestes característiques permeten als usuaris tenir un control sobre la seua nutrició, contribuint a l'objectiu que volen aconseguir.

ÍNDICE

RESUMEN	1
ABSTRACT	2
RESUM	3
JUSTIFICACIÓN	6
OBJETIVOS	7
PLANIFICACIÓN DEL PROYECTO	8
Propuesta plan de trabajo inicial	8
Diagrama de Gantt del proyecto	9
Subtareas	10
HERRAMIENTAS UTILIZADAS	13
DESCRIPCIÓN DEL PROYECTO	15
Análisis de funcionalidades claves	15
Autenticación de usuarios	15
Estimación de macronutrientes	16
Gestión de alimentos	19
Importación de alimentos desde fuente externa	21
Cómputo de calorías	22
Histórico de calorías ingeridas	24
Diseño	25
Arquitectura	25
Diagrama entidad-relación de la base de datos	27
Diseño lógico	28
Implementación	30
Implementación del Frontend	30
Implementación del Backend	33

Resumen flujo de trabajo dentro de la web	35
Pruebas	36
Pruebas Funcionales	36
Pruebas de Integración	38
Limitaciones detectadas.....	38
DESPLIEGUE DE KCALCONTROL.....	39
Estructura del despliegue	39
Despliegue frontend (React)	39
Despliegue backend (Laravel API)	39
Verificación final.....	40
Resultado del Despliegue	40
AMPLIACIONES FUTURAS	42
CONCLUSIONES.....	43
BIBLIOGRAFÍA Y WEBGRAFÍA	44
ANEXOS	45

JUSTIFICACIÓN

Este proyecto surge con el motivo principal de crear una herramienta de control de calorías dirigida a personas que están en un proceso de transformación física, ya sea para perder peso, ganar masa muscular o mejorar su salud en general. Durante este tipo de procesos, es fundamental tener un control de la ingesta calórica así como los macronutrientes, para la consecución de objetivos físicos. Sin embargo, muchas veces las personas carecen de los conocimientos necesarios para administrar correctamente sus ingestas, lo que hace de esto, un proceso un poco engorroso. Y es aquí donde KcalControl ofrece una solución sencilla, accesible y personalizada.

Para el sistema de cómputo calórico y macronutrientes se ha empleado la fórmula de Mifflin-St Jeor, que es una de las fórmulas más aceptadas y precisas para estimar la Tasa Metabólica Basal (BMR). La misma fue publicada en 1990 por **Mifflin y St Jeor** en un estudio titulado: *"A new predictive equation for resting energy expenditure in healthy individuals"*.

OBJETIVOS

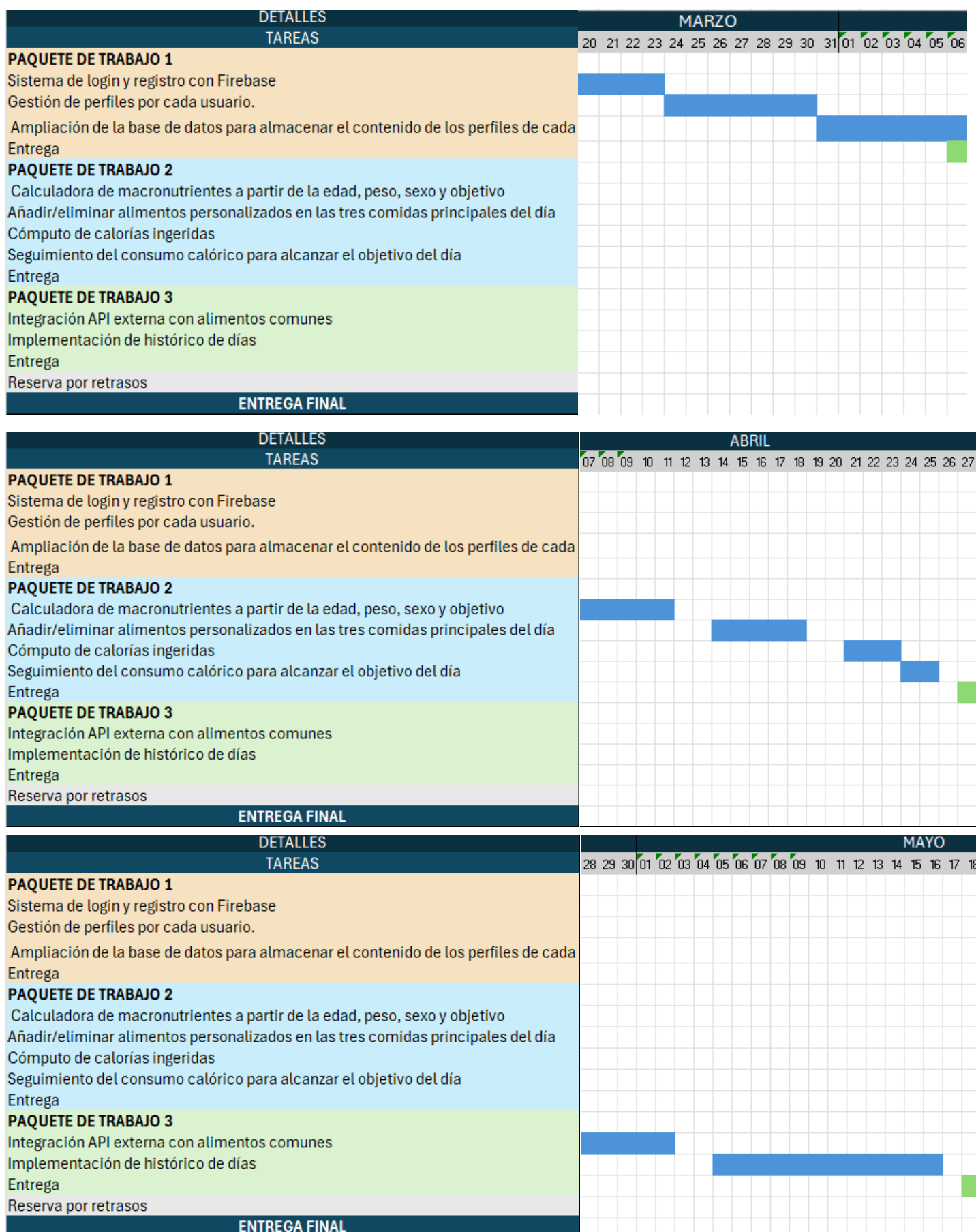
1. **Facilitar el acceso y gestión de cuentas de usuario:** El objetivo principal es permitir que los usuarios creen y gestionen sus cuentas proporcionando funcionalidades de registro, inicio de sesión y cierre de sesión.
2. **Personalizar la experiencia del usuario:** La aplicación web permitirá que cada usuario pueda crear y gestionar su propio perfil, almacenando y actualizando información básica, así como datos relacionados con su progreso y objetivos.
3. **Calcular las necesidades nutricionales individuales:** La aplicación web permitirá a los usuarios calcular sus necesidades diarias de macronutrientes (proteínas, carbohidratos y grasas) en función de su edad, peso, sexo y objetivo personal.
4. **Realizar un seguimiento de las calorías ingeridas:** La aplicación web proporcionará a los usuarios una vista clara de las calorías consumidas en cada comida y el total del día. Esto les permitirá hacer ajustes en su dieta en tiempo real para asegurarse de que cumplen con sus objetivos de ingesta calórica.
5. **Consultar el historial de ingestas diarias:** Los usuarios podrán filtrar por fecha para ver cómo ha sido su evolución a lo largo del tiempo, brindándoles un panorama claro de su trayectoria nutricional.

PLANIFICACIÓN DEL PROYECTO

Propuesta plan de trabajo inicial

Tareas	Fecha de inicio	Fecha de fin
Paquete de Trabajo 1		
PT1.1. Sistema de login y registro con Firebase	20/03/2025	23/03/2025
PT1.2. Gestión de perfiles por cada usuario	24/03/2025	30/03/2025
PT1.3. Ampliación de la base de datos para almacenar el contenido de los perfiles de cada usuario	31/03/2025	06/04/2025
Entrega parcial		06/04/2025
Paquete de Trabajo 2		
PT2.1. Calculadora de macronutrientes a partir de la edad, peso, sexo y objetivo	07/04/2025	11/04/2025
PT2.2. Añadir/eliminar alimentos personalizados en las tres comidas principales del día	14/04/2025	18/04/2025
PT2.3. Cómputo de calorías ingeridas	21/04/2025	23/04/2025
PT2.4. Seguimiento del consumo calórico para alcanzar el objetivo del día	24/04/2025	25/04/2025
Entrega parcial		25/04/2025
Paquete de Trabajo 3		
PT3.1. Integración API externa con alimentos comunes	28/04/2025	02/05/2025
PT3.2. Implementación de histórico de días	05/05/2025	16/05/2025
Entrega parcial		16/05/2025
Reserva por atrasos	16/05/2025	29/05/2025
Entrega Final		30/05/2025

Diagrama de Gantt del proyecto



Subtareas

Paquete de trabajo 1

PT1.1. Sistema de login y registro con Firebase

- Configurar un proyecto en Firebase.
- Implementar la autenticación de usuario con correo electrónico y contraseña.
- Diseño e implementación de la página de Login y Registro
- Manejar el cierre de sesión.
- Gestionar los tokens de acceso para mantener la sesión activa.

PT1.2. Gestión de perfiles de usuarios.

- Implementar una API REST en Laravel con la base una base de datos en MySQL para gestionar los datos básicos de los usuarios.
- Crear rutas y controladores en Laravel para las operaciones CRUD (crear, leer, actualizar, eliminar).
- Integrar la API de Laravel con Firebase para autenticar usuarios.

PT1.3. Ampliación de la base de datos para almacenar el contenido de los perfiles de cada usuario.

- Definir las tablas necesarias para almacenar la información del usuario.
- Crear un esquema relacional de base de datos.
- Implementar migraciones y seeders en Laravel para la creación y población inicial de la base de datos.
- Crear procedimientos almacenados para optimizar consultas específicas.

Paquete de trabajo 2

PT2.1. Calculadora de macronutrientes a partir de la edad, peso, sexo y objetivo

- Definir la fórmula para el cálculo de macronutrientes (por ejemplo, fórmula de Mifflin-St Jeor).

- Diseño e implementación de la interfaz para recibir los datos del usuario (edad, peso, sexo, objetivo).
- Crear una lógica para calcular los macronutrientes (proteínas, carbohidratos y grasas).

PT2.2. Añadir/eliminar alimentos personalizados en las tres comidas principales del día

- Diseñar formularios de entrada para que los usuarios puedan agregar alimentos de forma personalizada.
- Implementar lógica para añadir, eliminar, editar alimentos personalizados en la base de datos.
- Crear una interfaz para que los usuarios puedan gestionar los alimentos por cada comida (desayuno, almuerzo, cena).

PT2.3. Cómputo de calorías ingeridas

- Diseñar una lógica para calcular las calorías totales consumidas basadas en los alimentos ingresados.
- Mostrar el total de calorías ingeridas para cada comida y el total del día.

PT2.4. Seguimiento del consumo calórico para alcanzar el objetivo del día

- Implementar indicador visual que muestre las calorías ingeridas y las que le faltan para alcanzar el objetivo.

Paquete de trabajo 3

PT3.1. Integración API externa con alimentos comunes

- Seleccionar una API externa de alimentos (por ejemplo, la API de MyFitnessPal, Nutritionix).
- Integrar la API con la aplicación web para obtener información sobre alimentos comunes.
- Crear interfaz que permita al usuario buscar alimentos comunes y añadirlos a sus comidas.
- Mostrar información detallada de los alimentos (calorías, macronutrientes, etc.) al usuario.

PT3.2. Implementación de histórico de días

- Crear una tabla para almacenar el historial de comidas diarias de los usuarios.
- Implementar una vista que permita al usuario ver un historial de días pasados.
- Añadir funcionalidad para filtrar por fecha.

HERRAMIENTAS UTILIZADAS

Para la realización de este proyecto se emplearon las siguientes tecnologías:

React: Utilizado para el desarrollo del *frontend*, proporcionando una interfaz de usuario interactiva y dinámica aportando una experiencia fluida y reactiva, ya que permite actualizar la interfaz en tiempo real sin recargar la página.

Firebase: Utilizado para la gestión de la autenticación y gestión de usuarios. Con Firebase, fue posible gestionar el registro, inicio y cierre de sesión de los usuarios.

Firebase Authentication ofrece una solución sencilla y escalable para gestionar el inicio de sesión con diversos métodos, como Google, Facebook, correo electrónico y contraseñas, etc. Además se encarga de la gestión de contraseñas de forma segura y provee características como verificación de correo, restablecimiento de contraseñas. En cuanto a la escalabilidad es automática y se maneja correctamente en situaciones de alto tráfico.

Axios: Librería para la gestión de solicitudes HTTP desde React hacia la API.

Laravel: Este *framework* de PHP, permitió crear una API REST que gestionara los datos de los usuarios permitiendo realizar operaciones CRUD (crear, leer, actualizar y eliminar) sobre los datos almacenados de los usuarios. Laravel proporcionó una estructura robusta para gestionar la lógica de negocio y la manipulación de datos en el *backend*.

MySQL: Se empleó como sistema de gestión de bases de datos, proporcionando una solución relacional para almacenar los perfiles de los usuarios, los alimentos personalizados y el historial de ingestas diarias. MySQL garantizó una base de datos sólida y escalable.

Tailwind CSS: Se utilizó para el diseño de la interfaz de usuario, brindando un enfoque de desarrollo ágil mediante clases utilitarias que permitieron diseñar rápidamente una interfaz moderna, flexible y adaptativa para todos los dispositivos, mejorando la accesibilidad y la experiencia del usuario.

Estas tecnologías trabajaron en conjunto para ofrecer una solución escalable, con un enfoque en la facilidad de uso por parte de los usuarios.

DESCRIPCIÓN DEL PROYECTO

Análisis de funcionalidades claves

La aplicación web ofrece una serie de funcionalidades clave que responden a las necesidades del usuario las cuales se describirán a continuación.

Autenticación de usuarios

La autenticación de usuarios es una parte fundamental en el control de acceso a aplicaciones web (**Figura 1**).

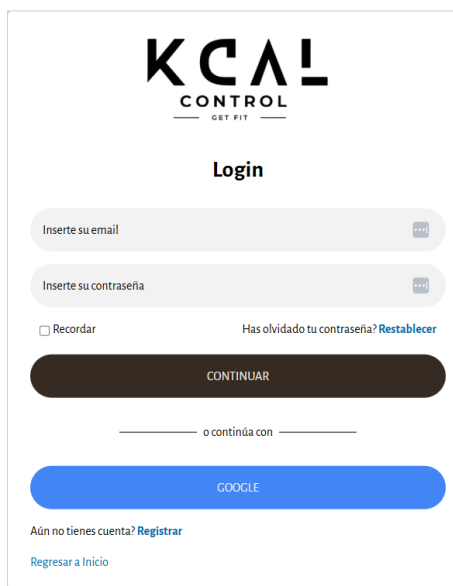
The image shows a login form for 'KCAL CONTROL'. At the top is the logo 'KCAL CONTROL' with the tagline 'GET FIT'. Below the logo is the word 'Login'. There are two input fields: 'Inserte su email' and 'Inserte su contraseña', both with password icons on the right. Below these fields is a checkbox labeled 'Recordar' and a link 'Has olvidado tu contraseña? Restablecer'. A large dark button labeled 'CONTINUAR' is below that. Underneath the button is the text 'o continúa con' followed by a blue button labeled 'GOOGLE'. At the bottom, there is a link 'Aún no tienes cuenta? Registrar' and another link 'Regresar a Inicio'.

Figura 1. Formulario *Login* para acceder a la web.

En este proyecto, se implementó el proceso de autenticación utilizando Firebase Authentication, un servicio proporcionado por Google que facilita la gestión segura de usuarios mediante diferentes métodos de acceso. En este proyecto se utilizan:

- Autenticación mediante usuario y contraseña
- Autenticación mediante una cuenta de Google

Se ha decidido optar por este método de autenticación por las ventajas que supone su utilización, entre las que se encuentran:

- Seguridad gestionada y actualizada por Google.
- Soporte para múltiples proveedores de identidad.
- Gestión automática de sesiones y tokens.

Estimación de macronutrientes

Si definimos un punto de partida en la aplicación web, sería la sección "Calculadora de macros" (**Figura 2**). En esta sección, el usuario ingresa sus datos personales, a partir de los cuales el sistema calcula y estima la cantidad de calorías diarias recomendadas.

Calculadora de Macros

Datos Personales

Completa la siguiente información para que podamos estimar tus objetivos calóricos diarios.

Inserte su Género

Seleccione una opción

Inserte su Edad

Índice de actividad

Seleccione una opción

Objetivo

Seleccione una opción

Altura (cm)

Peso Actual (Kg)

ENVIAR

Resultados

Rellena la información de la sección anterior para obtener/modificar el resultado estimado de macronutrientes necesarios por día

Distribución de macros

Proteína 32.30 g

Carbohidratos 325.64 g

Grasas 68.18 g

Calorías totales a consumir por día

2045.34 kCal

Figura 2. Interfaz para calcular la distribución de macros

Para estimar las calorías necesarias por día y su posterior distribución en macronutrientes, el usuario debe proporcionar la siguiente información:

- Género
- Edad
- Altura
- Peso
- Índice de actividad (desde sedentario hasta extremadamente activo)
- Objetivo (mantener, perder o aumentar peso)

Con estos datos se sigue un proceso que se inicia con el cómputo de la Tasa Metabólica Basal (BMR) y continúa con ajustes en dependencia del nivel de actividad que tenga el usuario y el objetivo que pretende alcanzar.

1. Cálculo de BMR

El BMR (Tasa Metabólica Basal) es la cantidad mínima de energía (en forma de calorías) que el cuerpo necesita para llevar a cabo funciones vitales en reposo, como la respiración, la circulación sanguínea, el control de la temperatura corporal, la digestión y el funcionamiento del sistema nervioso (Abreu, 2023).

Existen varias fórmulas reconocidas para calcular este parámetro, entre las cuales se incluyen:

- Ecuación de Harris-Benedict
- Ecuación de Katch-McArdle
- Ecuación de ten Haaf
- Ecuación de Cunningham

De estas cuatro ecuaciones predictivas, según el artículo "*Comparison of Predictive Equations for Resting Metabolic Rate in Healthy Nonobese and Obese Adults: A Systematic Review*", la ecuación de Mifflin-St Jeor es la que predice con mayor precisión la tasa metabólica en reposo, con un margen de error del 10% respecto a las mediciones realizadas por calorimetría indirecta, tanto en individuos no obesos como obesos. Este valor es superior al de cualquier otra ecuación y también presentó la tasa de error más baja (Frankenfield, Roth-Yousey, & Compher, 2005).

Por tal motivo, se ha decidido utilizar esta ecuación para el cómputo del BMR en este contexto. No obstante, si se busca aún más precisión, la ecuación de Mifflin-St Jeor resulta especialmente adecuada para individuos con un Índice de masa corporal (IMC) normal (es decir, con un índice entre 19 y 25), cuando no es necesario conocer el porcentaje de grasa corporal.

$$IMC = \frac{\text{peso}(Kg)}{\text{altura}(m)^2}$$

El BMR se determina teniendo en cuenta el sexo de la persona:

Sexo femenino:

$$BMR = (10 \times \text{peso en kg}) + (6.25 \times \text{altura en cm}) - (5.0 \times \text{edad en años}) - 161$$

Sexo masculino:

$$BMR = (10 \times \text{peso en kg}) + (6.25 \times \text{altura en cm}) - (5.0 \times \text{edad en años}) + 5$$

2. Ajuste según el factor de actividad adecuado. Es lo que se conoce como Gasto Energético Diario Total (TDEE)

Sedentario: Poco o ningún ejercicio, trabajo de oficina

$$TDEE = BMR \times 1.2$$

Ligeramente activo: Ejercicio físico de 1 a 3 días por semana.

$$TDEE = BMR \times 1.375$$

Moderadamente activo: Ejercicio físico de 3 a 5 días por semana.

$$TDEE = BMR \times 1.55$$

Muy activo: Ejercicio físico de 6 a 7 días por semana

$$TDEE = BMR \times 1.725$$

Extremadamente activo: Ejercicio físico dos veces al día.

$$TDEE = BMR \times 1.9$$

Este enfoque es comúnmente aceptado en la literatura de nutrición, aunque los factores exactos pueden variar ligeramente según la fuente.

Es importante tener en cuenta que estas ecuaciones son sólo una estimación y, por lo tanto, deben tenerse en cuenta las necesidades y objetivos nutricionales individuales.

Debe evaluar y reevaluar, si es necesario, los progresos del cliente y cómo se siente y, a partir de allí, hacer ajustes en el plan dietético.

3. Ajuste según objetivo (Hospitals, s.f.)

Ganar masa muscular: Añadir 250-500 calorías a tu TDEE.

Perder peso: Restar 250-500 calorías a tu TDEE.

Mantener el peso: Consumir tu TDEE calculado.

4. Distribución de macronutrientes

A partir de la información consultada en el sitio web (Besfor, 2024), se pueden recomendar las siguientes cantidades de macronutrientes:

Proteínas: Se recomienda consumir entre **1.6 y 2.2 gramos por kilogramo de peso corporal** al día para deportes de fuerza, y entre **1.2 y 1.6 gramos por kilogramo de peso corporal** al día para deportes de resistencia.

Carbohidratos: Para deportistas de resistencia, se sugieren entre **6 y 10 gramos por kilogramo de peso corporal** al día, mientras que para deportes de fuerza, la recomendación es entre **4 y 7 gramos por kilogramo de peso corporal** al día.

Grasas: Se recomienda consumir entre **1 y 1.5 gramos por kilogramo de peso corporal** al día o, en su defecto, entre el **20% y 35%** de las calorías totales diarias.

En KcalControl para estimar la distribución de las cantidades de macronutrientes se ha elegido el valor medio dentro de los rangos recomendados para cada macronutriente, con el objetivo de proporcionar una guía equilibrada y accesible para la mayoría de los usuarios. Luego los valores establecidos serían:

Proteínas: 1.9 gramos por kilogramo de peso corporal

Carbohidratos: 7 gramos por kilogramo de peso corporal

Grasas: 1.25 gramos por kilogramo de peso corporal

Gestión de alimentos

La gestión de alimentos es manejada en la web por la sección “Biblioteca de alimentos” (**Figura 3**). Este espacio de trabajo permite a los usuarios agregar, editar y eliminar alimentos dentro de una base de datos personalizada.

Cada usuario cuenta con sus propios registros de alimentos, ya que por cada alimento creado se almacena también el identificador único del usuario (UID). Esto permite mantener un control individualizado de los alimentos asociados a cada cuenta. De esta forma, cada usuario gestiona su propio catálogo de alimentos sin interferir en los registros de otros usuarios.

Biblioteca de alimentos

[AÑADIR PERSONALIZADO](#)
[IMPORTAR ALIMENTO POR CÓDIGO](#)

Descripción	Base (g)	Calorías (kcal)	Proteínas (g)	Grasas (g)	Carbohidratos (g)	Acciones
Arroz blanco cocido	100	130.00	2.00	0.00	28.00	✎ ✖
Huevo entero	100	143.00	13.00	10.00	1.10	✎ ✖
Manzana	100	52.00	0.30	0.20	14.00	✎ ✖
Lentejas cocidas	100	116.00	9.00	0.40	20.00	✎ ✖
Yogur natural	100	61.00	3.50	3.30	4.70	✎ ✖
Queso fresco	100	98.00	11.00	5.00	1.50	✎ ✖
Brócoli cocido	100	55.00	3.70	0.60	11.00	✎ ✖
Salchichas de pollo	100	186.00	11.30	13.40	4.80	✎ ✖

[1](#) [2](#)

Figura 3. Interfaz de gestión de los alimentos

Los alimentos registrados contienen los siguientes atributos:

- Descripción del alimento.
- Base en gramos sobre la cual se especifican los valores macros.
- Cantidad de calorías, grasas, proteínas, carbohidratos para la base especificada.

Los mismos son agregados desde la opción “Añadir personalizado” en la sección de “Biblioteca de alimentos” (**Figura 4**).

Figura 4. Interfaz para añadir un alimento de forma personalizada

Importación de alimentos desde fuente externa

El sistema también cuenta con acceso a la base de datos pública Open Food Facts (Open food facts, s.f.), un proyecto sin fines de lucro desarrollado por voluntarios de todo el mundo. Esta plataforma colaborativa reúne información nutricional de millones de productos alimenticios de diferentes países, con el objetivo de promover una alimentación más saludable y transparente. Todos los datos son de acceso libre y están disponibles mediante una API pública.

Open Food Facts proporciona información detallada sobre productos específicamente la información nutricional (calorías, grasas, proteínas, carbohidratos, etc.)

Dentro de la aplicación web, el usuario puede buscar e importar alimentos desde esta base de datos utilizando el código de barras presente en el envase del producto que desea añadir (**Figura 5**). Si el producto existe en la base de datos, se añadirá automáticamente a su catálogo de alimentos.

Importar alimento

Desde esta vista podrás escribir el código de barras del alimento que deseas añadir

Realice esta acción una única vez por alimento


Cada alimento añadido pasa a formar parte de su biblioteca de alimentos

Inserte el código de barras del producto deseado

ENVIAR

¿Es este el producto que buscas?

Cruesli, pepitas con mezcla de frutos secos



Propiedades Macros (por cada 100g de producto)

Proteínas: 8.5 g
Carbohidratos: 57 g
Grasas: 19 g
Calorías: 462 kcal

AÑADIR CANCELAR

Figura 5. Interfaz para importar un alimento

Cómputo de calorías

El sistema calcula automáticamente el total de calorías ingeridas por el usuario a lo largo del día en curso (**Figura 6**), en función de los alimentos registrados. El apartado “Dietario” permite al usuario visualizar e interactuar con esta información.

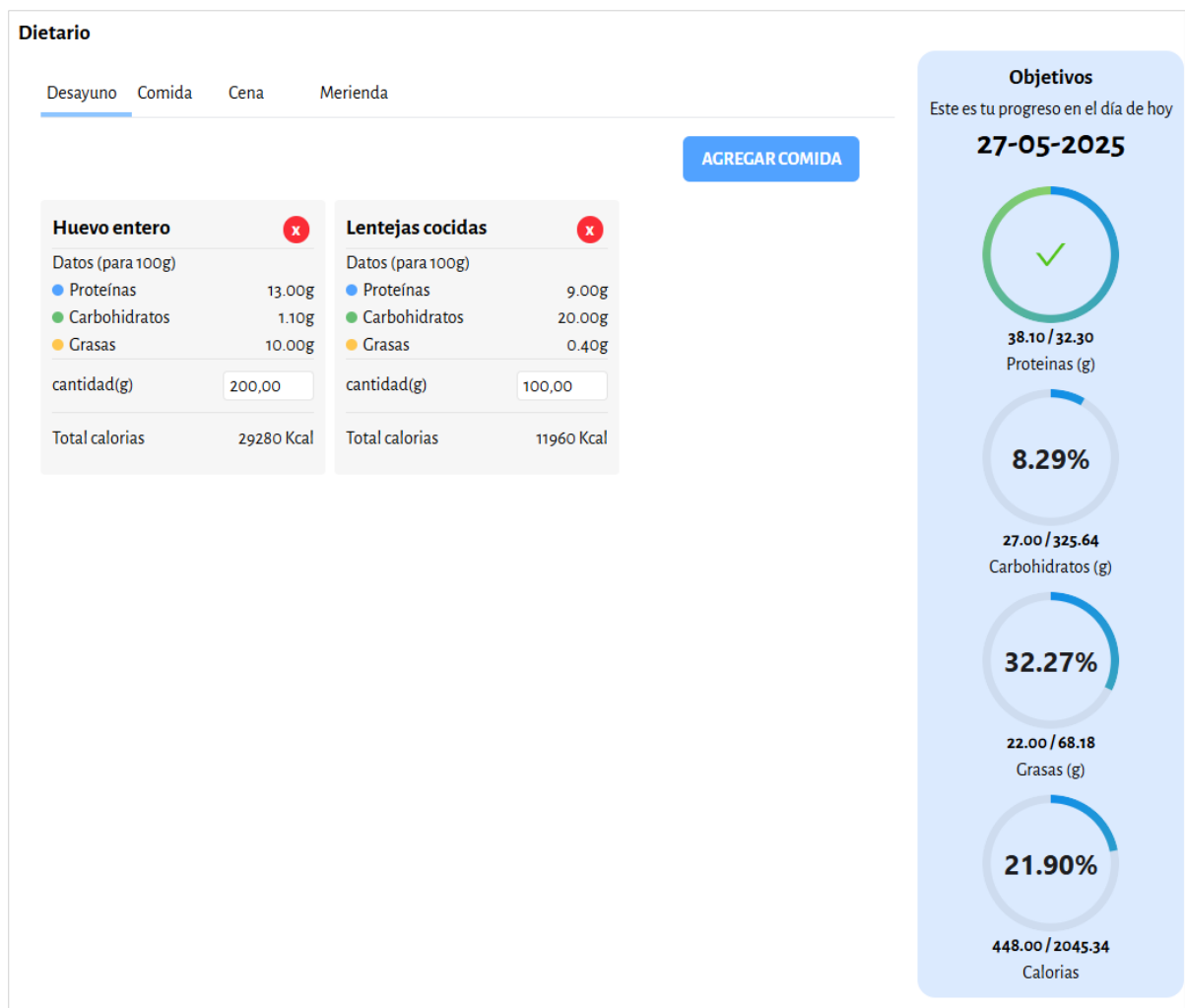


Figura 6. Interfaz para registrar alimentos por cada tipo de comida.

Desde esta sección, el usuario puede registrar los alimentos consumidos en las distintas comidas del día. Para ello, solo debe hacer clic en “Agregar comida”, lo que desplegará una lista de alimentos disponibles para seleccionar y añadir al registro correspondiente (**Figura 7**).

En este punto, también es posible importar un alimento si no se tiene en el listado, esta acción importa el alimento al sistema y además, lo registra como alimento de la comida desde donde se invocó la acción.

Añadir comida

Mis alimentos

Alimentos por Código de Barras

Descripción	Base (g)	Calorías (kcal)	Proteínas (g)	Grasas (g)	Carbohidratos (g)	Acciones
Arroz blanco cocido	100	130.00	2.00	0.00	28.00	+
Huevo entero	100	143.00	13.00	10.00	1.10	+
Manzana	100	52.00	0.30	0.20	14.00	+
Lentejas cocidas	100	116.00	9.00	0.40	20.00	+
Yogur natural	100	61.00	3.50	3.30	4.70	+

<

1

2

3

>

Figura 7. Listado de alimentos disponibles.

Desde la vista principal de “Dietario” se puede modificar la cantidad consumida que por defecto son 100g, el programa ajusta el valor calórico correspondiente basado en la nueva cantidad y actualiza el total diario.

Se incluyen indicadores para mostrar si el usuario ha alcanzado el objetivo de calorías establecido previamente, contribuyendo así a un mejor control nutricional y a la promoción de hábitos alimenticios saludables.

Histórico de calorías ingeridas

El apartado “Evolución” permite al usuario visualizar la cantidad de calorías ingeridas durante un período específico (**Figura 8**). Al seleccionar una fecha de inicio y una fecha de fin, la aplicación web recopila todos los registros disponibles dentro de ese intervalo y presenta un resumen del consumo calórico por día.

Elementos principales:

Selector de fechas: Componente calendario que permite elegir el rango de fechas deseado.

Visualización diaria: Gráfico de barras que muestra la cantidad de calorías ingeridas cada día dentro del rango seleccionado. Esta funcionalidad es útil para hacer un seguimiento del progreso alimenticio, identificar patrones de consumo y tomar decisiones más informadas sobre la dieta.

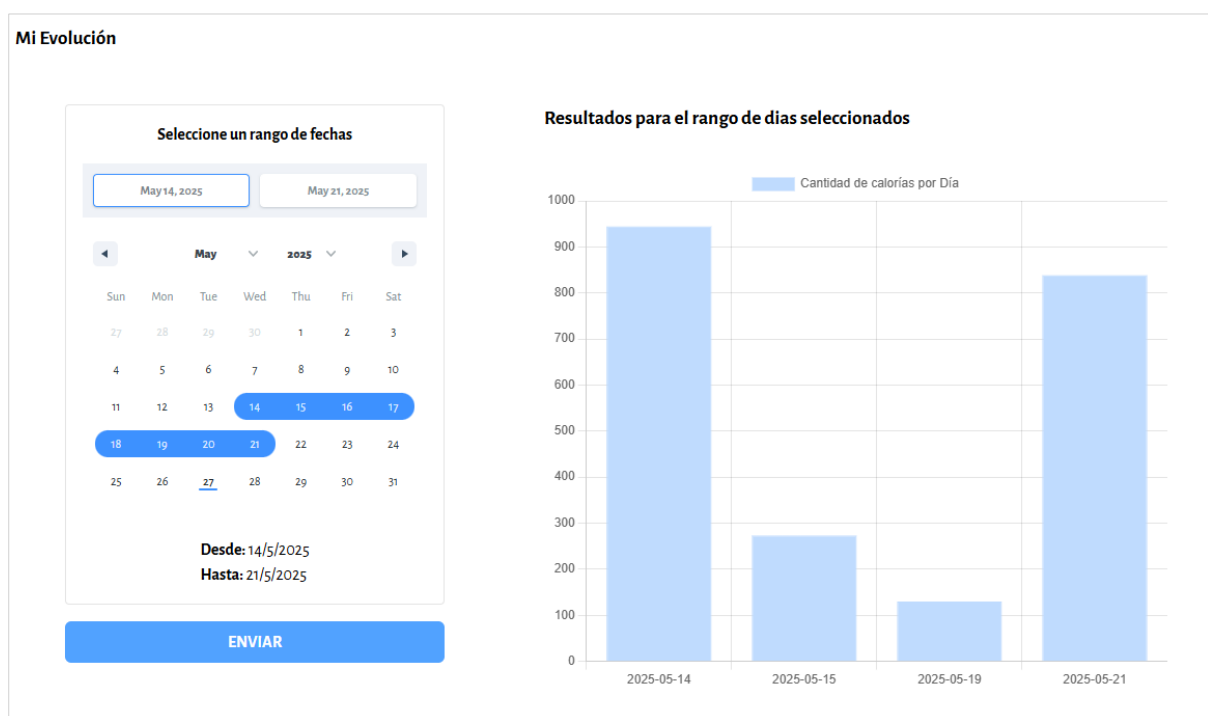


Figura 8. Interfaz que muestra histórico de calorías ingeridas para un rango de tiempo seleccionado

Diseño

Arquitectura

La arquitectura de la aplicación web sigue un modelo cliente-servidor clásico, donde el *frontend* desarrollado en React se comunica con una API REST construida en Laravel, la cual a su vez interactúa con una base de datos relacional MySQL.

❖ Frontend (React)

Se encarga de la presentación de la información y de la interacción con el usuario. Gestiona el enrutamiento, los formularios, la visualización de alimentos y gestión de los mismos por ingesta.

components/: Componentes de la interfaz como tarjetas de alimentos, formularios de ingreso, y botones de acción.

hooks/: Custom hooks que encapsulan la lógica de acceso a los contextos y simplifican su uso en los componentes.

pages/: Páginas principales como Inicio, Registro de alimentos, Historial de ingestas, etc.

context/: Gestión de los estados globales como la sesión del usuario.

Diagrama de navegación de la web

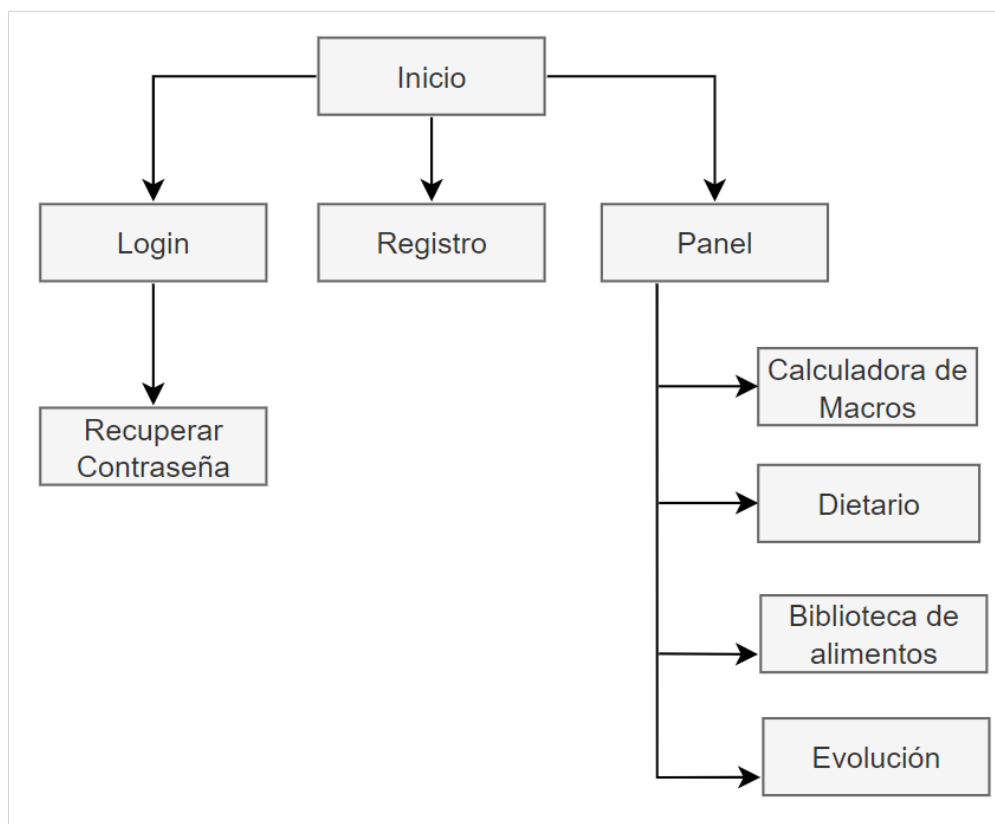


Figura 9: Diagrama de navegación de la web

❖ Backend (Laravel API)

Expone *endpoints* REST que permiten realizar operaciones de gestión de información de los usuarios, gestión de alimentos y registro de ingestas diarias.

app/Http/Controllers/: Controladores que gestionan las solicitudes del *frontend*.

routes/api.php: Archivo donde se definen las rutas de la API.

database/migrations/: Definición de las tablas de la base de datos.

app/Models/: Modelos Eloquent para interactuar con MySQL.

Diagrama entidad-relación de la base de datos

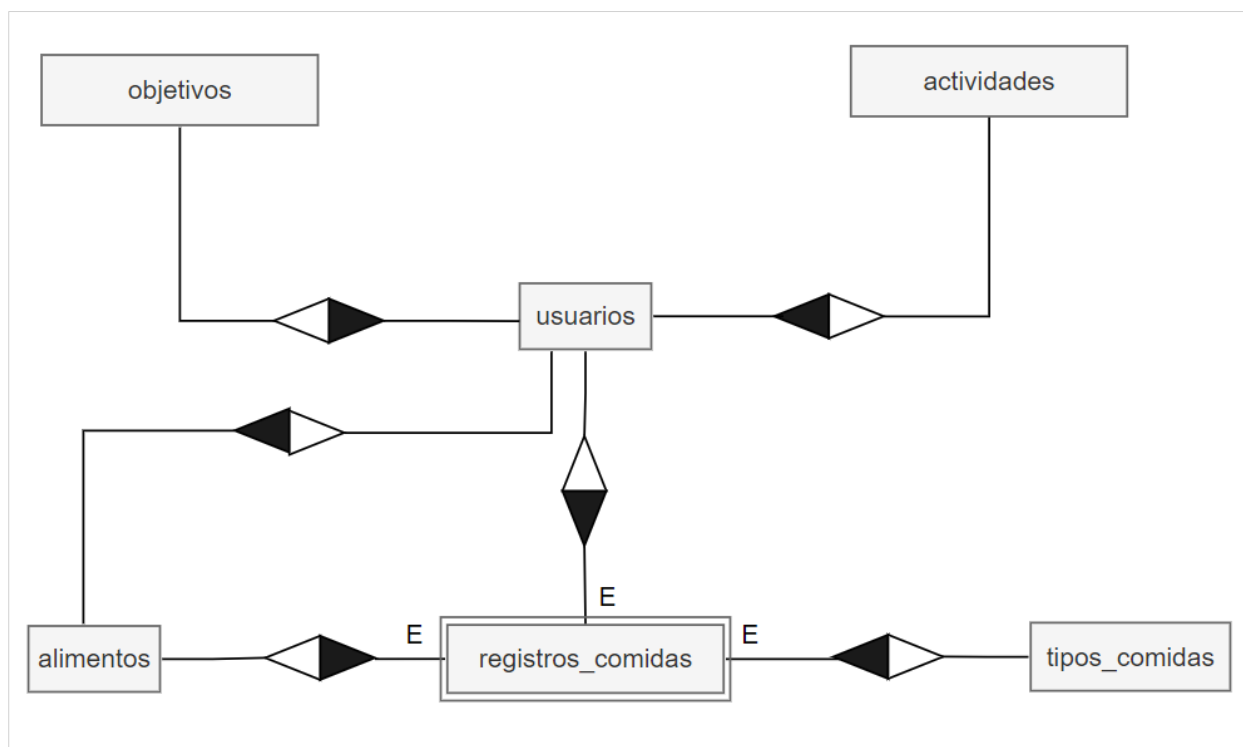


Figura 10: Diagrama entidad-relación de la base de datos

Diseño lógico

❖ Tabla: Usuarios

Campo	Tipo de dato	Descripción
uid	VARCHAR	Identificador único del usuario (clave primaria)
nombre	VARCHAR	Nombre del usuario
apellidos	VARCHAR	Apellidos del usuario
edad	INT	Edad del usuario
sexo	CHAR(1)	Sexo del usuario (por ejemplo, 'M' o 'F')
objetivo	INT	Tipo de objetivo
actividad	INT	Nivel de actividad (representado como un valor categórico)
obj_calorias	DECIMAL	Calorías diarias objetivo
obj_proteinas	DECIMAL	Gramos de proteínas diarios objetivo
obj_grasas	DECIMAL	Gramos de grasas diarios objetivo
obj_carbohidratos	DECIMAL	Gramos de carbohidratos diarios objetivo

Clave primaria: { uid }

Clave foránea : { objetivo } hace referencia a **objetivos**

Clave foránea : { actividad } hace referencia a **actividades**

❖ Tabla: tipos_comidas

Campo	Tipo de dato	Descripción
id	INT	Identificador único del tipo de comida (clave primaria)
descripción	VARCHAR	Nombre de la comida

Clave primaria: { id }

❖ Tabla: Objetivos

Campo	Tipo de dato	Descripción
id	INT	Identificador único del tipo de objetivo (clave primaria)
descripción	VARCHAR	Descripción del objetivo

Clave primaria: { id }

❖ **Tabla: Actividad**

Campo	Tipo de dato	Descripción
id	INT	Identificador único del tipo de actividad (clave primaria)
descripción	VARCHAR	Descripción de la actividad

Clave primaria: { id }

❖ **Tabla: Alimentos**

Campo	Tipo de dato	Descripción
id	VARCHAR	Identificador único del alimento (clave primaria)
uid	VARCHAR	Identificador del usuario
base	DECIMAL	Base en gramos para la cual se indican las cantidades macros
calorias	INT	Calorías contenidas en la base
proteinas	DECIMAL	Proteínas contenidas en la base
grasas	DECIMAL	Grasas contenidas en la base
carbohidratos		Carbohidratos contenidos en la base

Clave primaria: { id }

Clave foránea : { uid } hace referencia a **usuarios**

❖ **Tabla: Registros**

Campo	Tipo de dato	Descripción
id	VARCHAR	Identificador único del registro (clave primaria)
Uid	VARCHAR	Identificador del usuario
fecha	date	Fecha de registro
tipo_comida_id	INT	Tipo de comida
alimento_id	INT	Identificador del alimento
cantidad	DECIMAL	Cantidad de alimento en gramos

Clave primaria: { id }

Clave foránea : { uid } hace referencia a **usuarios**

Clave foránea : { tipo_comida_id } hace referencia a **tipos_comidas**

Clave foránea : { alimento_id } hace referencia a **alimentos**

Valor único : { uid, fecha , tipo_comida_id, alimento_id }

Implementación

En este apartado se describe cómo ha sido desarrollado el sistema KcalControl, detallando tanto la estructura y funcionamiento del *frontend*, construido con React y Firebase, como del *backend*, implementado en Laravel. Se explican los componentes principales, la organización del código, el flujo de datos entre cliente y servidor, así como los mecanismos utilizados para garantizar la autenticación, la gestión del estado, y la persistencia de datos.

Implementación del Frontend

Como es característico de React, la estructura del *frontend* se basó en componentes lo cual permitió una organización modular, facilitando la escalabilidad y el mantenimiento del proyecto.

A continuación, se describen tres aspectos clave de su implementación, que permiten comprender cómo se estructura funcionalmente la interfaz y cómo se gestionan los estados para el mantenimiento del flujo de la información dentro de la web.

Gestión del estado global con contextos en React

Para organizar de forma estructurada los distintos bloques funcionales de la aplicación web, KcalControl hace uso de la Context API de React. Esta permite compartir datos entre componentes sin necesidad de propagar *props* manualmente, evitando así el *prop drilling* (paso excesivo de *props* entre componentes intermedios que no la utilizan), lo que facilita la escalabilidad y el mantenimiento del código, especialmente en aplicaciones con múltiples estados globales interrelacionados. Es así como se distribuye la información global dentro de la app, manteniendo sincronizada la información que varios componentes utilizan.

En este proyecto se implementaron varios contextos específicos (**Figura 11**), cada uno orientado a una responsabilidad concreta dentro del flujo funcional de la web:

AuthContext: Gestiona todo el proceso de autenticación de usuarios, integrándose directamente con Firebase. Permite iniciar sesión, registrarse, cerrar sesión, iniciar sesión con Google y restablecer la contraseña. También se encarga de distribuir el usuario autenticado (*currentUser*) al resto de componentes que lo necesiten.

PersonalInfoContext: Administra la información personal del usuario, como edad, peso, altura, género, nivel de actividad y objetivo. Esta información, se obtiene y se almacena en *userData*, permitiendo que otros contextos (como *ComputoContext*) puedan operar correctamente y ofrecer funcionalidades personalizadas para cada usuario.

AlimentosContext: Encargado de gestionar los alimentos del usuario autenticado. Provee tanto el estado actual de los alimentos como las funciones necesarias para realizar operaciones CRUD (crear, leer, actualizar y eliminar) a través de peticiones via *axios* a la API del *backend*.

ComputoContext: Responsable de calcular y gestionar los valores estimados de calorías y macronutrientes que el usuario debe consumir diariamente. Toma como base los datos personales gestionados por *PersonalInfoContext* y actualiza los cálculos en función de los cambios.

RegistroComidaContext: Administra los registros de alimentos consumidos por el usuario, organizados por tipo de comida (desayuno, comida, merienda, cena). También contiene las funciones que permiten agregar, modificar o eliminar estos registros mediante comunicación con la API.

CentralContext: Este contexto cumple una función integradora y de simplificación. Centraliza y expone las variables y funciones esenciales de todos los contextos anteriores, permitiendo que los componentes del *frontend* puedan consumir todos los recursos necesarios desde un único punto. Esto reduce la complejidad a la hora de importar múltiples contextos por separado y facilita la escalabilidad, ya que las nuevas funcionalidades pueden añadirse sin modificar el acceso a los datos ya existentes.

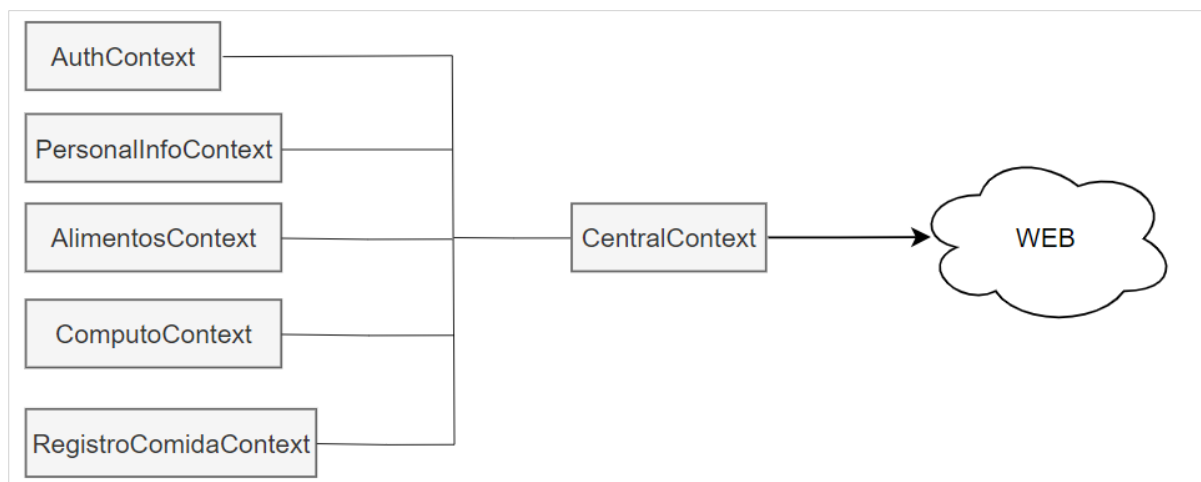


Figura 11. Estructura de Contextos en la web

Autenticación

Cuando un usuario desea crear una cuenta, proporciona su correo electrónico y una contraseña segura. Esta información se envía a Firebase utilizando el método:

```
firebase.auth().createUserWithEmailAndPassword(email, password)
```

Firebase gestiona automáticamente el almacenamiento seguro de las credenciales y la verificación del formato del correo.

Una vez registrado el usuario, para poder acceder, este insertará su correo y contraseña y la aplicación web envía esta información a Firebase a través de:

```
firebase.auth().signInWithEmailAndPassword(email, password)
```

Firebase valida las credenciales y, si son correctas, genera un **token de sesión** (JWT) que representa al usuario autenticado, de igual forma maneja de automáticamente la

persistencia de la sesión, manteniendo al usuario autenticado entre recargas de página o cierres de aplicación, hasta que este decida cerrar sesión explícitamente.

La aplicación web también gestiona el proceso de restablecimiento de contraseña mediante la función `resetPassword`, que hace uso del método `sendPasswordResetEmail` proporcionado por Firebase. Al introducir una dirección de correo válida, se envía automáticamente un enlace al usuario para que pueda establecer una nueva contraseña, sin necesidad de intervención manual por parte del administrador del sistema.

Este sistema de autenticación de firebase se integra en el contexto personalizado de React denominado ***AuthContext***, encargado de centralizar toda la lógica relacionada con el acceso de usuarios y compartirla con el resto de componentes de la aplicación. Su objetivo es que cualquier componente del *frontend* pueda acceder al usuario autenticado y ejecutar acciones como cerrar sesión, y manejar todas las funcionalidades dentro de la web.

Comunicación con la API

La comunicación entre el *frontend* desarrollado en **React** y el *backend* implementado en **Laravel** se realiza mediante la librería **Axios**, que permite realizar peticiones HTTP de forma sencilla y estructurada.

Para garantizar que cada solicitud hacia la API esté autenticada correctamente, se ha configurado un **interceptor global de Axios**. Este interceptor se activa automáticamente antes de que se envíe cualquier petición y se encarga de adjuntar el token de autenticación (proporcionado por Firebase) en el encabezado `Authorization` de la solicitud. De esta forma, se evita tener que incluir manualmente el token en cada llamada a la API, centralizando la lógica de autenticación y mejorando la seguridad y mantenibilidad del código.

Implementación del Backend

El *backend* de KcalControl ha sido desarrollado utilizando el framework Laravel, siguiendo una arquitectura REST. Su función principal es proveer una API robusta

que se comunique con el *frontend* y gestione de forma eficiente la persistencia y manipulación de los datos del usuario.

Rutas

Todas las rutas de la API están definidas en el archivo `routes/api.php`. En este archivo se declaran los distintos *endpoints* que responden a las peticiones HTTP realizadas desde el *frontend* mediante *Axios*. Cada ruta está asociada a un controlador específico que se encarga de ejecutar la lógica correspondiente a la acción solicitada (crear, obtener, modificar o eliminar un recurso).

Controladores

El sistema cuenta con varios controladores, cada uno enfocado en una tabla o entidad específica del modelo de datos. Estos controladores contienen la lógica de negocio y las funciones CRUD necesarias para manipular la base de datos. Entre ellos destacan:

UsuarioController: Gestiona las operaciones relacionadas con los usuarios (creación, actualización, obtención de datos).

AlimentoController: Contiene la lógica para crear, leer, actualizar y eliminar alimentos personalizados por el usuario.

TipoComidaController: Administra los tipos de comidas definidos por el sistema (desayuno, comida, cena, merienda).

CreacionRegistroComidaController: Controlador auxiliar para gestionar el proceso de creación conjunta de un alimento y su registro asociado en una comida.

RegistroComidaController: Encargado de gestionar los registros diarios de alimentos según el tipo de comida.

Estos controladores son responsables de validar los datos entrantes y devolver respuestas JSON estandarizadas para el consumo en el *frontend*.

Middleware CORS

Para permitir la correcta comunicación entre el *frontend* (que puede estar alojado en otro origen) y el *backend*, se ha configurado un middleware personalizado llamado

CorsMiddleware. Este middleware se encarga de modificar las cabeceras HTTP de las respuestas para permitir el acceso a la API desde la aplicación cliente. Entre las cabeceras configuradas se encuentran:

Access-Control-Allow-Origin

Access-Control-Allow-Methods

Access-Control-Allow-Headers

Esto permite realizar solicitudes seguras y controladas entre dominios distintos, algo esencial para el funcionamiento de aplicaciones web modernas con arquitectura desacoplada.

Migraciones

Laravel permitió gestionar la estructura de la base de datos mediante migraciones, que se encuentran en el directorio database/migrations. Cada archivo de migración representa una tabla de la base de datos y define su esquema (campos, tipos de datos, relaciones y restricciones). Gracias a este sistema, la base de datos puede crearse o restaurarse fácilmente con los comandos *php artisan migrate* o *php artisan migrate:fresh*.

Estas migraciones son el reflejo directo del modelo de datos de la aplicación web y permiten mantener sincronizado el entorno de desarrollo y producción, facilitando la colaboración y el control de versiones.

Resumen flujo de trabajo dentro de la web

1. El usuario inicia sesión o se registra si aún no tiene una cuenta. El inicio de sesión puede realizarse mediante usuario y contraseña, o a través de su cuenta de Google.
2. Una vez dentro, para comenzar a utilizar la aplicación web, debe introducir los datos necesarios para calcular las calorías que debe consumir diariamente: edad, peso, altura, sexo, nivel de actividad y objetivo deseado.
3. Con estos datos, el sistema permitirá al usuario registrar las comidas diarias, organizadas por grupos: desayuno, comida, cena y merienda.
4. El usuario interactúa con la interfaz desarrollada en React, realizando acciones como agregar alimentos o registrar ingestas. Este proceso puede hacerse de

forma manual, introduciendo las macronutrientes del alimento, o automáticamente, indicando el código de barras del producto.

5. React utiliza Axios para enviar solicitudes HTTP (POST, GET, PUT o DELETE) a la API desarrollada en Laravel.
6. La API recibe la solicitud, valida los datos y realiza las operaciones correspondientes sobre la base de datos MySQL.
7. Laravel responde al *frontend* con los datos actualizados o con el estado de la operación solicitada.
8. React actualiza la interfaz en función de las respuestas del *backend*.
9. El estado de la aplicación web se gestiona de forma global mediante Context y se consume en los componentes a través de custom hooks como useAuthContext, useAlimentosContext o useComputoContext, lo que favorece un flujo de datos claro, reutilizable y mantenible.

Pruebas

Para garantizar el correcto funcionamiento de la web, se llevaron a cabo diversos tipos de pruebas en el *frontend* y en el *backend*, incluyendo pruebas funcionales y de integración. En esta fase, se listan las principales pruebas de esta etapa.

Pruebas Funcionales

Durante el desarrollo de la aplicación web, se llevaron a cabo diversas pruebas funcionales con el fin de validar que cada sección respondiera correctamente ante las acciones del usuario y que los flujos definidos se cumplieran sin errores. A continuación, se detallan los principales casos evaluados:

- ✓ Calculadora de Macros

Objetivo de la prueba:

Verificar el correcto cálculo de los requerimientos calóricos diarios y su distribución en macronutrientes a partir de los datos personales ingresados.

Acciones realizadas:

- Se introdujeron distintos conjuntos de datos (sexo, edad, peso, altura, índice de actividad y objetivo) para comprobar que el resultado cambiara en consecuencia.
- Se validaron los campos del formulario para asegurar que los datos obligatorios no permitieran valores vacíos o erróneos.

Resultado esperado:

El sistema devuelve una estimación coherente y personalizada del total de calorías recomendadas por día.

- ✓ Registro de Comidas (Dietario)

Objetivo de la prueba:

Confirmar que el usuario puede registrar alimentos correctamente en cada tipo de comida del día (desayuno, almuerzo, cena, etc.).

Acciones realizadas:

- Se verificó que al hacer clic en "Agregar comida", el registro se asocie correctamente al tipo de comida correspondiente.
- Se comprobó que la importación de un alimento desde Open Food Facts, realizada desde esta sección, creara dos acciones: una en el catálogo personal del usuario (tabla de alimentos), y otra en el dietario (registro de consumo).

Resultado esperado:

Alimentos correctamente vinculados a la comida seleccionada y almacenados sin errores.

- ✓ Biblioteca de Alimentos

Objetivo de la prueba:

Validar que el usuario puede gestionar su propio catálogo de alimentos.

Acciones realizadas:

- Se probaron operaciones de creación, edición y eliminación de alimentos personalizados.
- Se verificó también que un alimento se puede importar desde Open Food Facts mediante el código de barras, y que quede correctamente registrado en la base de datos del usuario.

Resultado esperado: Las operaciones CRUD (crear, leer, actualizar y eliminar) funcionan correctamente y reflejan los cambios en la interfaz.

✓ Evolución Nutricional

Objetivo de la prueba:

Comprobar que el sistema muestra un resumen de calorías consumidas por fecha o dentro de un rango de tiempo definido.

Acciones realizadas:

- Se seleccionaron distintos intervalos de fechas con y sin registros asociados.
- Se verificó que el sistema sumara correctamente las calorías por día y mostrara un total acumulado.
- Se evaluó el comportamiento cuando no existía información: el sistema notificó al usuario que no había datos para mostrar.

Resultado esperado:

Visualización coherente de los datos disponibles y feedback claro cuando no se encuentra información.

Pruebas de Integración

Se testó la api del *backend* en Laravel, utilizando las herramientas de Postman y *console.log()* desde el *frontend* en desarrollo, con la certeza de que:

- Las llamadas a la API están siendo gestionadas correctamente.
- Los datos de alimentos, usuarios y registros se almacenan y consultan sin errores.
- Las respuestas del *backend* cumplen con el formato esperado y son interpretadas adecuadamente por la interfaz.

Limitaciones detectadas

Se observaron tiempos de espera elevados al acceder a la aplicación web pues la carga inicial intentaba obtener todos los datos del usuario directamente al montar el componente principal del Panel. Como solución se refactorizó el flujo de carga de datos, gestionando las peticiones de forma segmentada desde cada componente y utilizando estados de carga (*loading states*) para mejorar la percepción de rendimiento.

DESPLIEGUE DE KCALCONTROL

Para la publicación del sistema KcalControl se utilizó un plan de hosting proporcionado por Hostinger, el cual permite gestionar tanto el dominio principal como subdominios personalizados. El despliegue se realizó en dos entornos independientes: uno para el *frontend* y otro para el *backend*, permitiendo así una arquitectura desacoplada y más escalable.

Estructura del despliegue

Dominio principal: www.yoadiaval.com

Frontend (React) desplegado en: kcalcontrol.yoadiaval.com

Backend (Laravel API) desplegado en: apikcalcontrol.yoadiaval.com

Despliegue frontend (React)

Para la compilación del proyecto se generó una versión optimizada del frontend usando el comando:

```
npm run build
```

Subida de archivos:

El contenido de la carpeta build/ fue subido al directorio correspondiente al subdominio appkcalcontrol.yoadiaval.com mediante el administrador de archivos de hostinger

Configuración del entorno:

En el archivo **.env** del proyecto React se definió la URL de la API (REACT_APP_API_URL=<https://apikcalcontrol.yoadiaval.com>).

Despliegue backend (Laravel API)

Subida de archivos

Se utilizó el administrador de archivos de Hostinger para subir el contenido completo del proyecto Laravel en la carpeta (/apikcalcontrol) del dominio y el contenido publico al directorio correspondiente del subdominio apikcalcontrol.yoadiaval.com.

Configuración del entorno:

Se editó el archivo `.env` con los datos de conexión reales a la base de datos MySQL proporcionados por Hostinger.

Se estableció el valor de APP_URL a `https://apikcalcontrol.yoadiaval.com`.

Base de datos:

Se creó una base de datos MySQL desde el panel de Hostinger.

Se importó manualmente la base de datos desde phpMyAdmin local.

Permisos y caché:

Se ajustaron los permisos de las carpetas `storage/` y `bootstrap/cache/`.

Configuración del servidor (Apache):

Se aseguró que el subdominio apunte al directorio `public/` de Laravel.

Se modificó el archivo `.htaccess` para garantizar la redirección correcta.

CORS y seguridad:

Se configuraron los encabezados CORS en Laravel para permitir el acceso desde el dominio del frontend (`kcalcontrol.yoadiaval.com`).

Verificación final

Se realizaron pruebas funcionales para comprobar la correcta comunicación entre frontend y backend, el funcionamiento del login, el consumo de la API y la carga de datos.

Resultado del Despliegue

Con esta configuración, KcalControl queda completamente operativo en línea, separando la interfaz de usuario del servicio de API para lograr un mantenimiento más sencillo y una mejor organización del sistema. Esta estructura también permite futuras

integraciones o migraciones a otros entornos (como servidores dedicados o servicios en la nube) con mayor facilidad.

AMPLIACIONES FUTURAS

Mejoras propuestas para la aplicación web:

1. **Sistema de notificaciones:** Alertar sobre el progreso de sus objetivos. Esto mejoraría la adherencia a los objetivos y la motivación del usuario, manteniéndolos enfocados en su proceso de transformación.
2. **Menús de alimentos completos preestablecidos:** Ofrecer menús prediseñados según los objetivos nutricionales (pérdida de peso, ganancia muscular, mantenimiento) para facilitar la planificación de las comidas. Los usuarios podrán agregar menús completos a las comidas principales del día.
3. **Selección de ecuación para el cómputo de macronutrientes:** Permitir a los usuarios avanzados elegir entre diferentes ecuaciones científicas (como Mifflin-St Jeor, Harris-Benedict, Katch-McArdle) para calcular sus necesidades de macronutrientes, adaptándose a sus circunstancias. Esto ofrece mayor flexibilidad y precisión en los cálculos, ayudando a los usuarios a obtener resultados más adecuados a su perfil y objetivos.

CONCLUSIONES

La aplicación desarrollada ha conseguido ser una herramienta digital intuitiva y centrada en la personalización del control nutricional. A través de una interfaz clara la web permite a los usuarios tomar el control de su alimentación diaria y su progreso personal.

En primer lugar, la aplicación web facilita la creación de cuentas de usuario mediante un sistema de autenticación seguro basado en Firebase. Los usuarios pueden registrarse utilizando correo electrónico o iniciar sesión utilizando su cuenta de Google.

En segundo lugar, se ha puesto especial atención en **la personalización de la experiencia del usuario**, permitiendo que cada perfil almacene información clave como edad, peso, altura, sexo, nivel de actividad física y objetivos personales. Esta información es gestionada de forma centralizada y actualizable, lo cual permite que las funcionalidades del sistema se adapten a los cambios de cada usuario, fomentando un enfoque individualizado en su proceso de transformación nutricional.

Se ha conseguido además, establecer una estimación personalizada de las necesidades nutricionales gracias a la calculadora de macros integrada que permite generar automáticamente los requerimientos calóricos y de macronutrientes diarios a partir de los datos personales ingresados. Esto proporciona al usuario una base clara para planificar su alimentación.

Finalmente, se ha implementado un sistema de historial que permite consultar la evolución de la ingesta diaria en calorías a lo largo del tiempo, con filtros por fechas para ofrecer una visión longitudinal del progreso. Esto aporta al usuario un panorama claro de su adherencia y evolución nutricional, cumpliendo con el objetivo de ofrecer un seguimiento detallado y visualmente comprensible.

En conclusión, la aplicación no solo cumple con los objetivos definidos inicialmente, sino que lo hace integrando una arquitectura técnica moderna (React, Laravel, Firebase, Axios) que garantiza escalabilidad y una experiencia de usuario coherente. Esto permite ofrecer una solución digital completa para el acompañamiento nutricional individualizado.

BIBLIOGRAFÍA Y WEBGRAFÍA

- Abreu, M. (3 de mayo de 2023). *Ecuación Mifflin-St. Jeor para profesionales de la nutrición*. Obtenido de Nutrium: <https://nutrium.com/blog/es/ecuacion-mifflin-st-jeor-para-profesionales-de-la-nutricion>
- Besfor. (Noviembre de 2024). *Cómo calcular las cantidades ideales de macronutrientes según tu nivel de actividad física?* Obtenido de Besfor: <https://besfor.com/calcular-macronutrientes-segun-actividad-fisica/>
- Developer documentation for Firebase*. (s.f.). Obtenido de Firebase: <https://firebase.google.com/docs>
- Frankenfield, D., Roth-Yousey, L., & Compher, C. (2005). Comparison of predictive equations for resting metabolic rate in healthy nonobese and obese adults: a systematic review. *Journal of the Academic of Nutrition and Dietetics*, 775-789.
- Get started with Tailwind CSS*. (s.f.). Obtenido de Tailwindcss: <https://tailwindcss.com/docs/installation/using-vite>
- Hospitals, M. (s.f.). *Optimice su dieta: Calculadora de ingesta calórica*. Obtenido de Medicover Hospitals: <https://www.medicoverhospitals.in/es/fitness-health-calculators/calorie-intake-calculator>
- Laravel Documentation*. (s.f.). Obtenido de Laravel: <https://laravel.com/docs/11.x/readme>
- Learn React*. (s.f.). Obtenido de React.dev: <https://react.dev/learn>
- Mifflin, M. D., Jeor, S. T., Hill, L. A., Daugherty, S. A., & Koh, Y. O. (1990). A new predictive equation for resting energy expenditure in healthy individuals. *ScienceDirect*, 241-247.
- onlinetoolkit. (19 de Septiembre de 2024). *Calculadora Mifflin-St Jeor: Tasa Metabólica Basal*. Obtenido de Onlinetoolkit: <https://onlinetoolkit.co/es/calculadora-mifflin-st-jeor-tasa-metabolica-basal>
- Open food facts*. (s.f.). Recuperado el 2025, de <https://es.openfoodfacts.org/>

ANEXOS

I. Manual de instalación

❖ Backend

Este manual asume que tienes el proyecto Laravel completo en una carpeta local, y que deseas ejecutarlo en un entorno de desarrollo local utilizando XAMPP (Windows)

Requisitos previos

Instalar

XAMPP (incluye Apache, MySQL, PHP)

Composer (gestor de dependencias de PHP)

PHP (incluido en XAMPP; Laravel recomienda PHP 8.1 o superior)

Pasos

1. Copiar la carpeta del proyecto de Laravel en el directorio htdocs de XAMPP

2. Abrir el terminal y navegar hasta la carpeta del proyecto

```
>> cd C:\xampp\htdocs\apikcalcontrol
```

3. Ejecutar XAMP y verificar que Apache y MySQL estén activos desde el Panel de Control de XAMPP.

4. Instalar las dependencias del proyecto

```
>> composer install
```

Esto descargará las dependencias listadas en composer.json

5. Configurar archivo .env

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=
APP_DEBUG=true
APP_URL=http://localhost/apikcalcontrol/public
```

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=kcalcontrol
DB_USERNAME=root
DB_PASSWORD=
```

6. Ejecutar las migraciones

```
>> php artisan migrate
```

7. Iniciar servidor desde laravel

```
>> php artisan serve
```

❖ FrontEnd

Requisitos previos

Instalar

Node.js (versión recomendada: LTS, por ejemplo, 18.x o superior)

NPM (viene incluido con Node.js) o Yarn como gestor de paquetes.

Pasos

1. Desde la terminal, dentro de la carpeta del proyecto ejecutar

```
>> npm install
```

2. Ejecutar el proyecto en modo desarrollo

```
>> npm run dev
```

3. Crear versión para producción

```
>> npm run build
```

4. Abrir el navegador web y visitar la URL indicada en la terminal que es donde se está sirviendo el proyecto en modo desarrollo.

II. Contenidos en soporte digital

- ❖ Memoria del proyecto

IES El Grao. 2025-05 Proyecto DAW YoannetDiazValdes.pdf

- ❖ Documento con instrucciones de instalación

Instrucciones.pdf

- ❖ Carpeta con contenido *frontend* y *backend* del proyecto

PROYECTO_KCALCONTROL

El soporte digital entregado contiene el proyecto completo dentro de la carpeta **PROYECTO_KCALCONTROL**, estructurado en dos carpetas principales: **front_kcalcontrol** y **back_kcalcontrol**, correspondientes al *frontend* y *backend* de la aplicación web, respectivamente.

Estructura relevante dentro de **front_kcalcontrol**

/src/: contiene los componentes, contextos, vistas y servicios principales de la aplicación.

/public/: archivos estáticos públicos.

package.json: define las dependencias y scripts de ejecución del proyecto *frontend*.

Esta carpeta se relaciona directamente con los apartados de esta memoria dedicados a la interfaz de usuario, funcionalidades visuales y experiencia del usuario

Estructura relevante dentro de **back_kcalcontrol**

/app/, **/routes/**, **/database/**: contienen la lógica del servidor, rutas de la API y migraciones de base de datos.

.env: archivo para configurar las variables de entorno necesarias.

composer.json: archivo con las dependencias del proyecto Laravel.

Esta carpeta está vinculada a los apartados de esta memoria que abordan el modelo de datos, la estructura de la API, la lógica de negocio y las operaciones sobre la base de datos