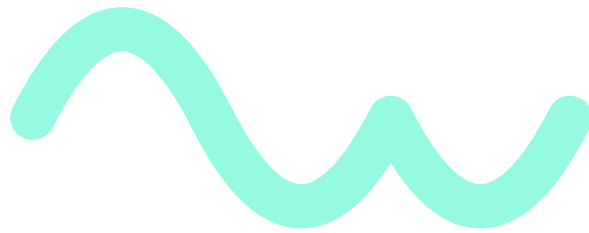


Proyecto Grado Desarrollo de Aplicaciones Web

WellnesTrack



Realizado por:
Daniel Martín Hermoso Hermoso

Tabla de contenido

1. Introducción	4
1.1. Propósito	4
1.2. Objetivos	4
1.3. Antecedentes (estado del arte)	5
1.4. Viabilidad técnica y económica del proyecto	5
Recursos hardware y software necesarios:	5
1.5. Temporalización	7
2. Análisis	8
2.1. Documentación relevante	8
2.2. Definiciones	8
2.3. Requisitos funcionales y no funcionales	8
2.4. Casos de Uso. Diagramas UML de casos de uso	10
2.5. Modelado E/R. Diagramas E/R.	11
2.6. Sketching de la interfaz.....	12
Sketching	12
Wireframing	23
3. Diseño.....	35
3.1 Arquitectura hardware y software de la solución	35
3.2 Modelado funcional de la solución. Diagramas de clase	35
3.3 Modelado de datos. Modelo relacional. Diccionario de Datos	36
3.4 Prototipado de la interfaz	37
4. Implementación	49
4.1 Requisitos de instalación y ejecución	49
Método 1 (Recomendado):	49
Método 2:	50
Ejecución una vez instalado:	51
4.2 Implementación funcional. Clases.....	51
4.3 Implementación del modelo de datos. Tablas.....	53
5. Pruebas	55
5.1. Pruebas de módulos	55
Prueba número 1:	55
Prueba número 2:	55
5.2. Pruebas de integración	56
5.3 Pruebas del sistema.....	56

5.4 Pruebas de instalación.....	57
6. Conclusiones.....	58
6.1 Grado de consecución de los objetivos inicialmente planteados	58
6.2 Dificultades encontradas.....	58
6.3 Propuestas de mejora y posibles ampliaciones	58
7. Bibliografía y recursos on-line.....	61
• Symphony	61
• Angular.....	61
• Firebase.....	61
• Base de datos	61
• Docker	61
• Despliegue.....	62
8. Glosario de términos.....	63
8.1 Informáticos	63
8.2 Problema.....	63
9. Anexos.....	64

1. Introducción

Mi proyecto se centra en desarrollar una web dedicada a poder realizar un seguimiento detallado de la alimentación y actividad física que el usuario lleve en su día a día, junto con el peso y el IMC del usuario.

1.1. Propósito

En la sociedad actual, la conciencia sobre la importancia de una alimentación saludable y la actividad física regular está en aumento. Sin embargo, muchos individuos encuentran dificultades para mantener un seguimiento preciso y sistemático de sus hábitos alimenticios y de ejercicio, lo que puede dificultar el logro de sus objetivos de bienestar personal

El proyecto responde a la necesidad de una herramienta integral y accesible que permita a los usuarios llevar un registro detallado y personalizado de su ingesta de alimentos y su actividad física. Al proporcionar una plataforma fácil de usar y altamente funcional, se busca facilitar el proceso de seguimiento de la salud y el bienestar de los usuarios

El proyecto está enfocado en los usuarios que deseen llevar un seguimiento detallado de su ingesta de alimentos, actividad física y progreso de peso.

En una hipotética extensión se podría implementar un apartado en el que los datos se lleven de forma conjunta con una persona experta en el tema ya sea un nutricionista con la posibilidad de usar la web también como una guía con la dieta que seguir o con un entrenador personal para saber que ejercicios realizar.

Fuera del alcance del proyecto quedarían aspectos médicos detallados que cada usuario pueda llegar a necesitar.

1.2. Objetivos

Las metas que se tienen como objetivo no son más que utilizar las herramientas y tecnologías usadas a lo largo de los módulos cursados para demostrar la capacidad de desarrollo y resolución de problemas que tengo como programador, así como el uso e investigación de nuevas tecnologías que se podrían llegar a usar.

Todo esto enfocado en el desarrollo de un proyecto que de verdad tiene un uso real y que soluciona un problema existente en la sociedad actual.

1.3. Antecedentes (estado del arte)

La web más similar a mi proyecto es la web [myfitnesspal](#). Es una web que cuenta con diferentes partes, desde un blog, comunidad y hasta la que trata sobre lo mismo que mi proyecto.

Es una web a mi parecer antigua, su estilo no es que incite mucho al uso ya que se ve anticuado y no es de lo más sencillo de usar, la búsqueda de los alimentos no es que sea la más sencilla y la creación de estos es tediosa, también cuenta con el apartado para hacer seguimiento de los ejercicios, aunque la búsqueda de ellos no es la mejor.

Es una web que en lo poco que la he usado en la búsqueda de este apartado no es que me haya incitado a volverla a usar, aquí es donde creo que podría hacerlo de una mejor forma en mi proyecto.

1.4. Viabilidad técnica y económica del proyecto

Recursos hardware y software necesarios:

Hardware: Se necesitará un equipo que actúe como servidor, en el cual se ejecutarán contenedores Docker. He desarrollado dos formas de desplegar el proyecto para su uso, consisten en:

- **Método 1** (Recomendada):
 - Consiste en desplegar dos contenedores docker:
 - El primero contendrá el código de la api hecha en Symfony junto con el servidor de la base de datos Xampp, iniciados automáticamente con el arranque del contendor.
 - El segundo contendrá el código de la página web hecha en Angular, web que es iniciada automáticamente con el arranque del contenedor.
- **Método 2:**
 - Consiste en desplegar un único contenedor y lanzar la página web desde el equipo:
 - El primero contendrá el código de la api hecha en Symfony junto con el servidor de la base de datos Xampp, iniciados automáticamente con el arranque del contendor.
 - La página web creada se lanzará desde el equipo del usuario mediante la terminal.

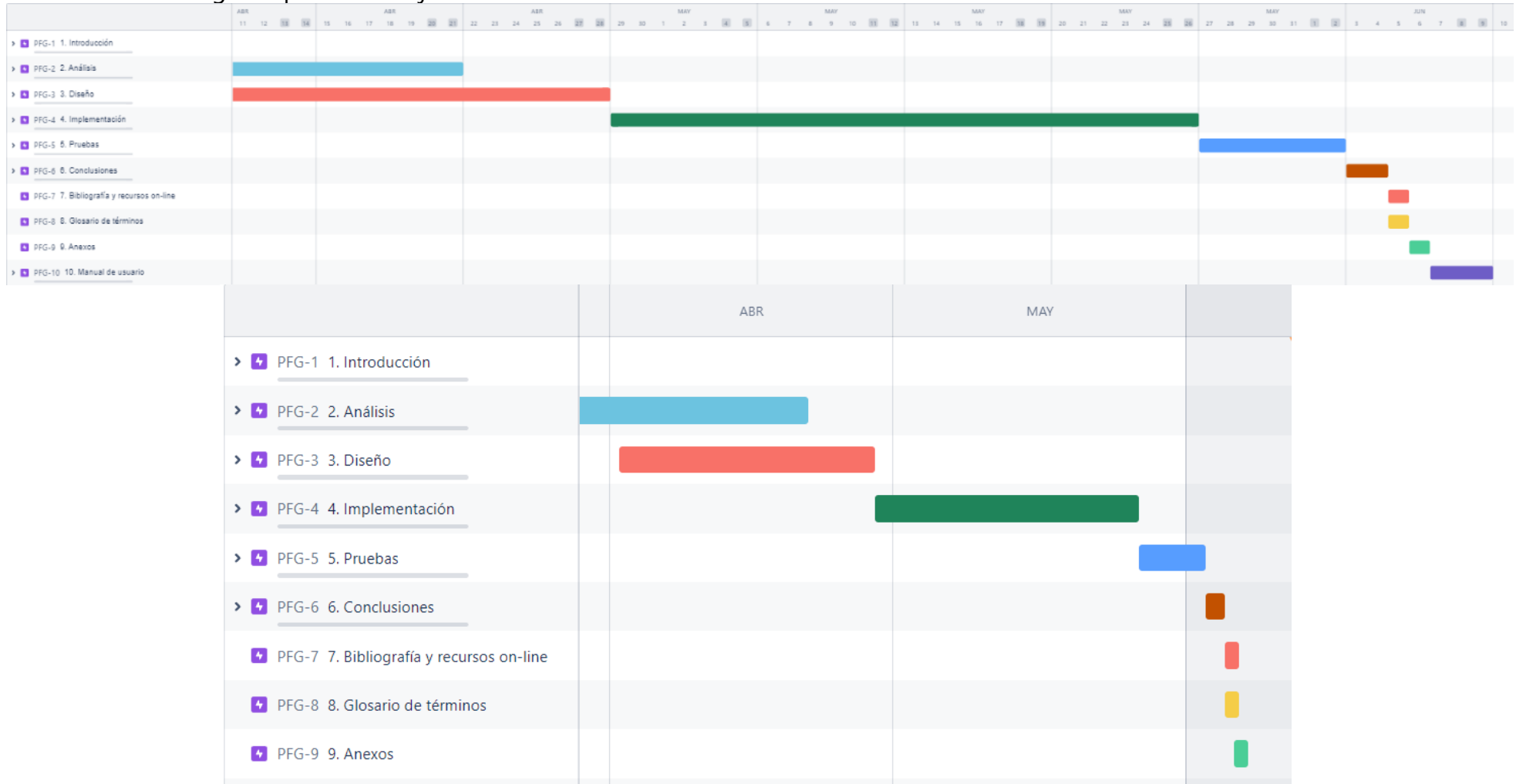
Software: Se utilizarán diversas tecnologías de desarrollo web, como HTML, CSS y JavaScript, para la interfaz de usuario. El framework de desarrollo web Angular se empleará para construir la parte front-end de la aplicación. Para el backend, se utilizará el framework Symfony para proporcionar la API que gestionará la lógica del servidor. La base de datos MySQL se utilizará para almacenar los datos del usuario, así como los datos relacionados con los alimentos, recetas y ejercicios necesarios para el funcionamiento de la aplicación. Estos datos incluirán información detallada sobre los alimentos disponibles, las recetas creadas por los usuarios y/o proporcionadas por los administradores, así como los diferentes tipos de ejercicios disponibles para realizar un seguimiento preciso de la actividad física.

Todos estos componentes se pueden ejecutar en contenedores Docker, lo que garantiza una gestión eficiente y escalable del entorno de desarrollo y producción del proyecto.

Las soluciones creadas han sido pensadas por su flexibilidad, escalabilidad y modularidad, haciendo de ellas opciones viables para el desarrollo y la implementación de la aplicación web. La arquitectura basada en contenedores Docker y las tecnologías modernas de desarrollo web ofrecen una estructura que permite una gestión más eficiente y simplificada de cada componente del sistema.

1.5. Temporalización

La planificación elegida por mi para el desarrollo del proyecto es la siguiente.
Se añaden dos imágenes para una mejor visibilidad



2. Análisis

2.1. Documentación relevante

Las definiciones han sido obtenidas en su mayoría de [Wikipedia](#)

2.2. Definiciones

- Alimento: Cualquier sustancia consumida para proporcionar apoyo nutricional a un ser vivo.
- Caloría: Nombre de una unidad de energía que se usa para definir el valor nutricional de los alimentos.
- Índice de Masa Corporal (IMC): Medida utilizada para evaluar el peso corporal en relación con la altura, que puede ayudar a determinar si una persona tiene un peso saludable, infrapeso, sobrepeso u obesidad.
- Proteínas: Son nutrientes esenciales para el cuerpo humano.
- Grasas, Carbohidratos y Azúcares: Son nutrientes que se consumen para dar energía al organismo humano.
- Vitaminas y Minerales: Son compuestos esenciales para el funcionamiento adecuado del organismo humano
- Receta: Conjunto de instrucciones para realizar sobre diferentes alimentos para realizar un plato o bebida.

2.3. Requisitos funcionales y no funcionales

- Requisitos funcionales de datos
 - Sobre los usuarios se guardará: Nombre, edad, correo, contraseña, peso, altura, objetivo, peso objetivo.
 - De los alimentos se guardará: Nombre, descripción, marca, imagen y sus valores nutricionales.
 - De las recetas se guardará: Nombre, descripción, ingredientes, imagen y sus valores nutricionales.
 - De los ejercicios se guardará: Nombre, descripción, grupo muscular, instrucciones, dificultad, calorías quemadas y enlaces.
- Requisitos funcionales de contenido
 - Los ejercicios tendrán enlaces a videos
 - Los alimentos y recetas tendrán una imagen asociada
- Requisitos funcionales de transaccionales
 - Los usuarios puede iniciar sesión introduciendo correo y contraseña.
 - Los usuarios podrán añadir alimentos o recetas a su lista de consumiciones diaria
 - Los usuarios podrán añadir ejercicios a su lista de ejercicios realizados diaria
 - Los usuarios podrán proponer alimentos, recetas y ejercicios
 - Los usuarios podrán introducir su peso a su registro de forma diaria

- Los usuarios podrán modificar su peso objetivo
 - Los administradores deberán revisar para aceptar y eliminar propuestas de alimentos, recetas y ejercicios
- Requisitos funcionales de interfaz
 - La web brindará un mecanismo a los usuarios para modificar sus datos personales
- Requisitos no funcionales
 - Los usuarios solo podrán consumir los alimentos, recetas y ejercicios que han sido aceptados por administradores
 - Los usuarios podrán proponer alimentos, recetas y ejercicios para un uso global
 - Los alimentos, recetas y ejercicios serán aceptados o rechazados únicamente por los administradores

2.4. Casos de Uso. Diagramas UML de casos de uso

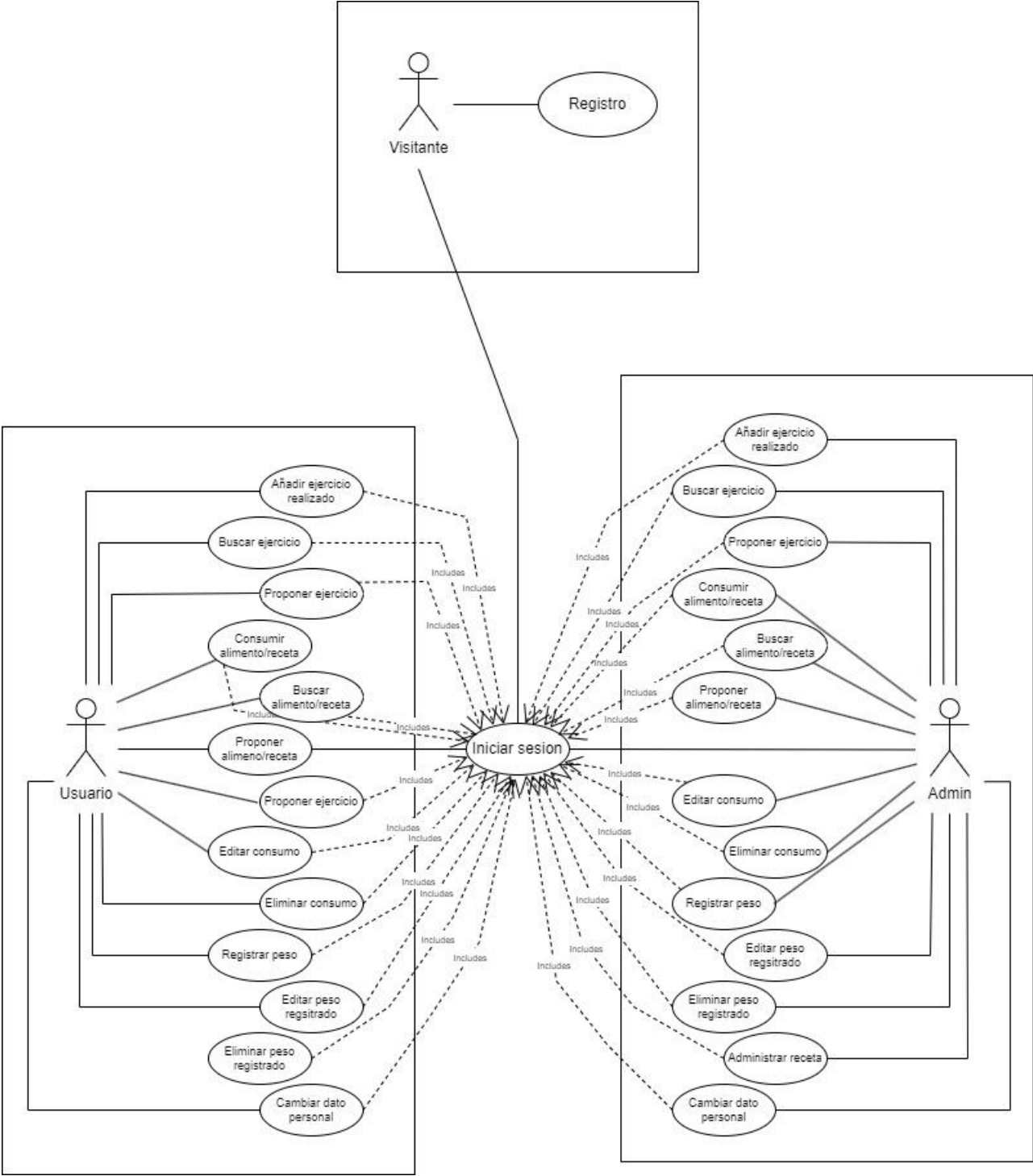


Imagen Diagrama UML

2.5. Modelado E/R. Diagramas E/R.

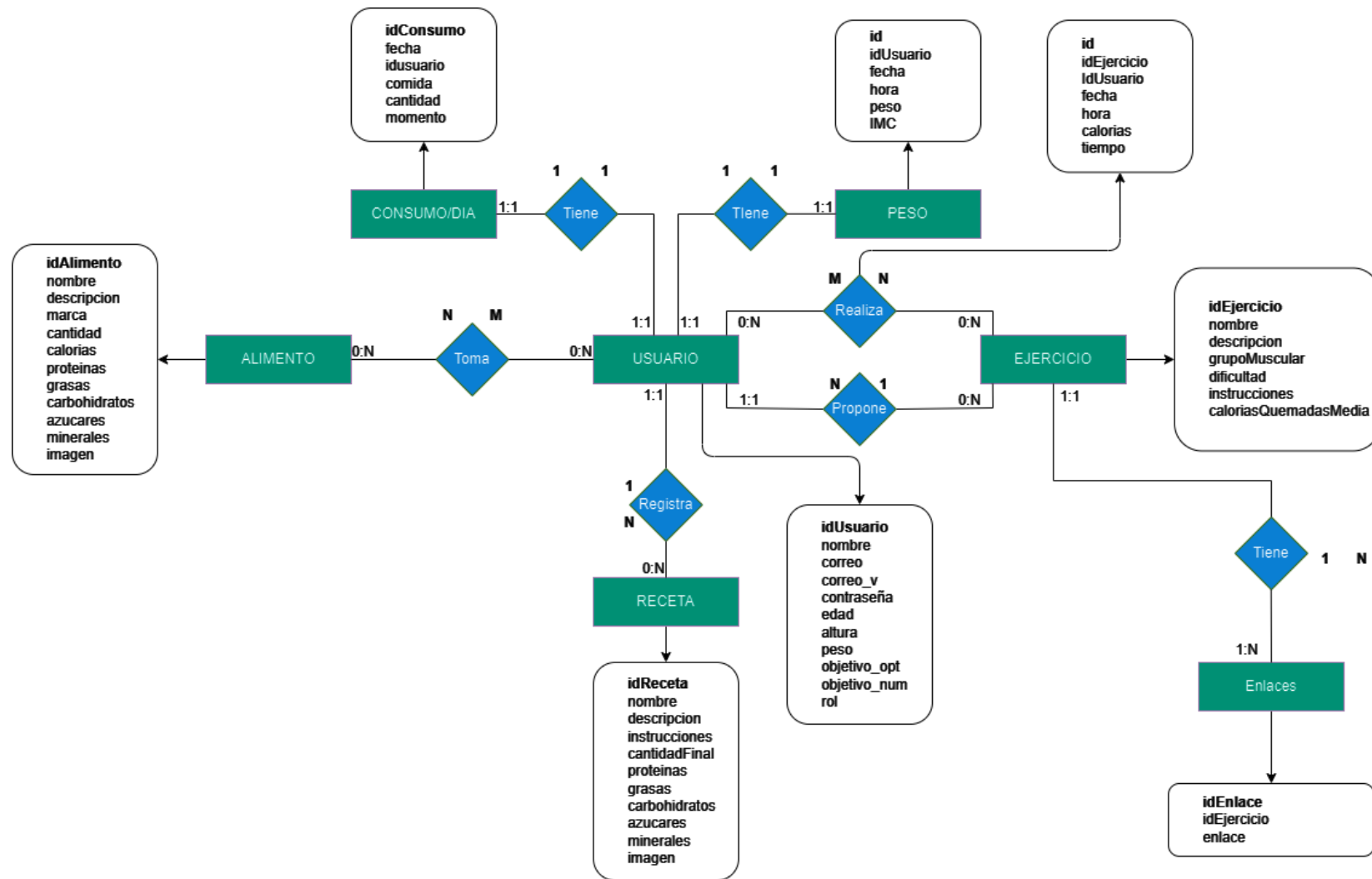


Imagen Modelo ER

2.6. Sketching de la interfaz

Sketching

Estas son las imágenes escaneadas del sketching realizado sobre la web. Todas ellas se encuentran en la carpeta de imágenes.

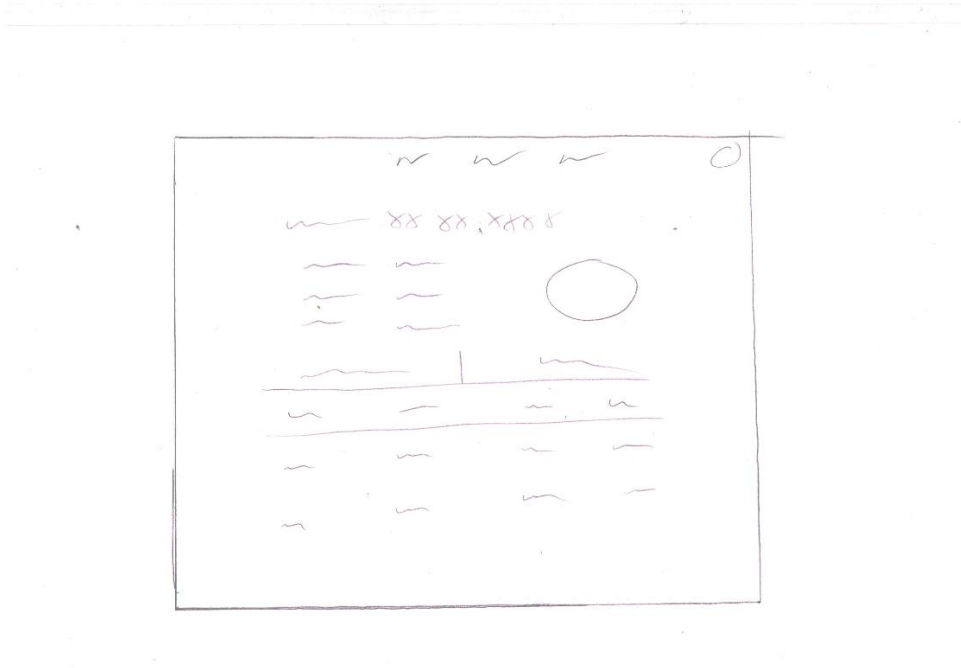


Imagen 1: Página de inicio

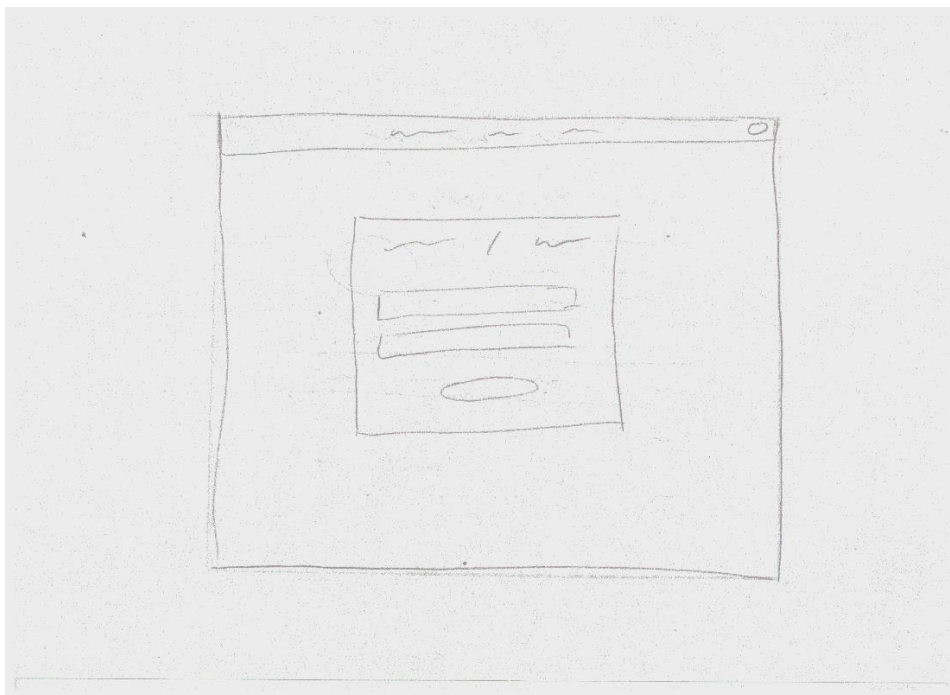


Imagen 2: Página de login

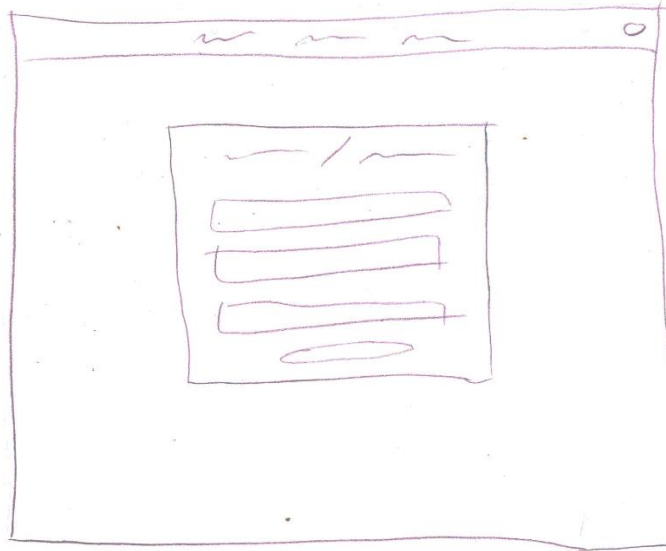


Imagen 3: Página de registro

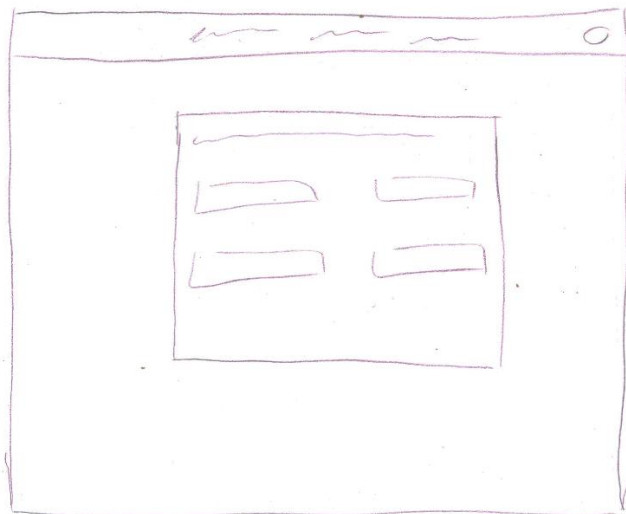


Imagen 4: Página de registro, datos personales

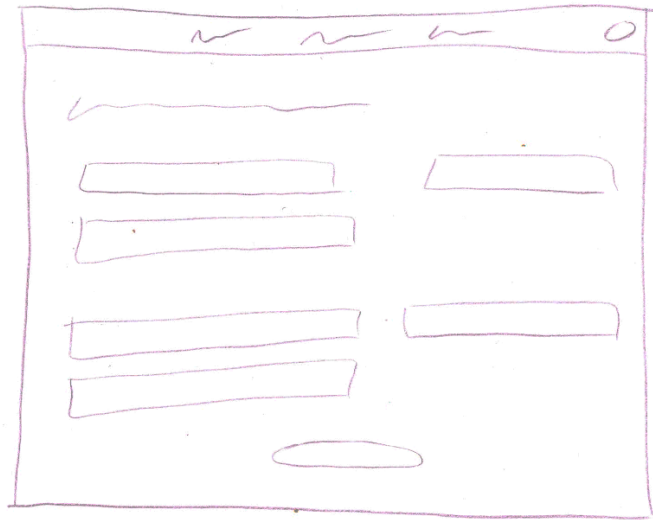


Imagen 5: Página de perfil

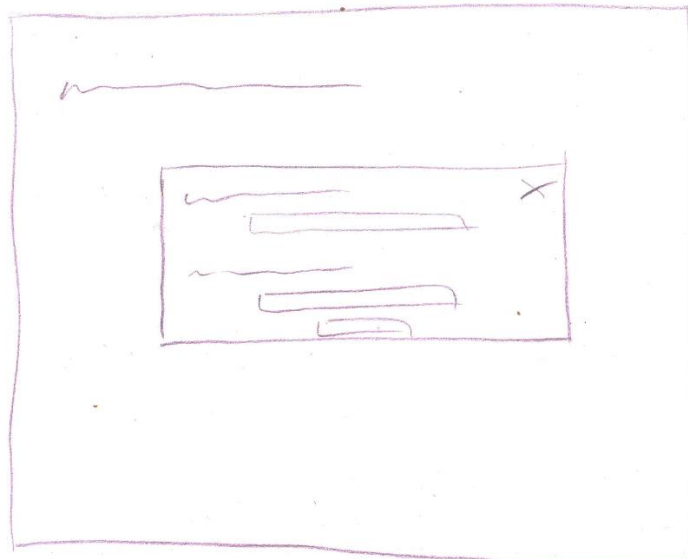


Imagen 6: Página de perfil, modificar dato

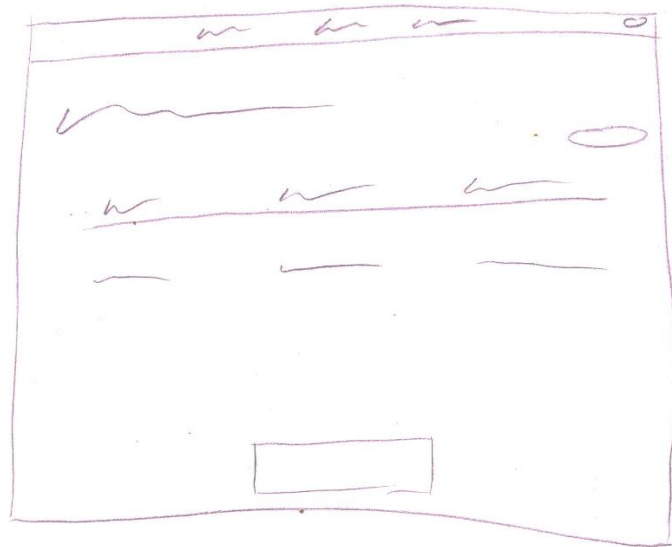


Imagen 7: Página de comidas home

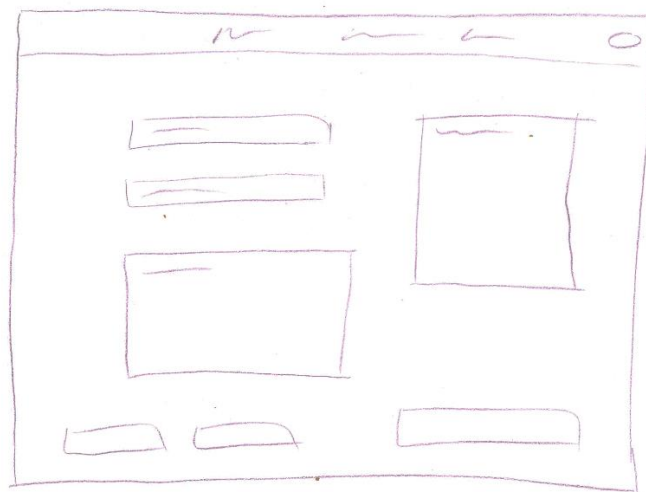


Imagen 8: Popup consumir alimento

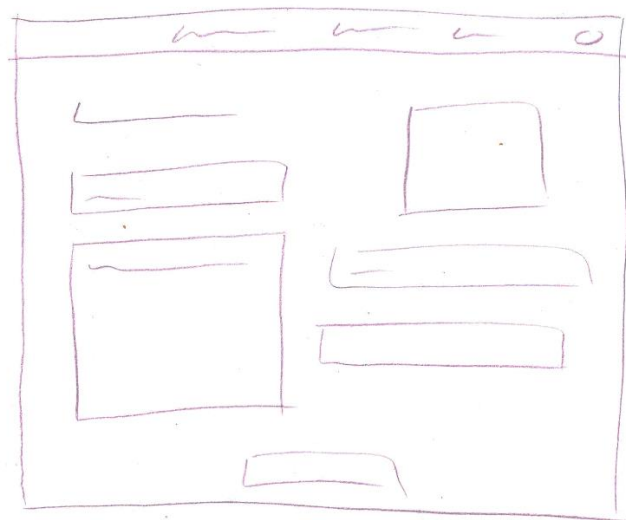


Imagen 9: Formulario proponer alimento

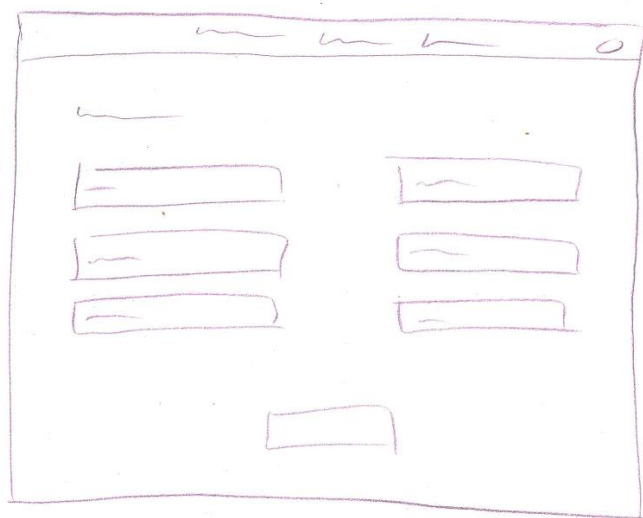


Imagen 10: Formulario proponer alimento, valores nutricionales

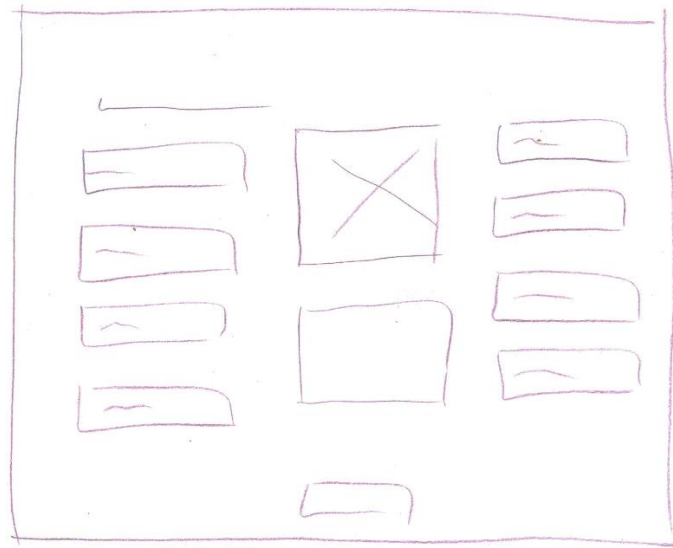


Imagen 11: Página información de alimento

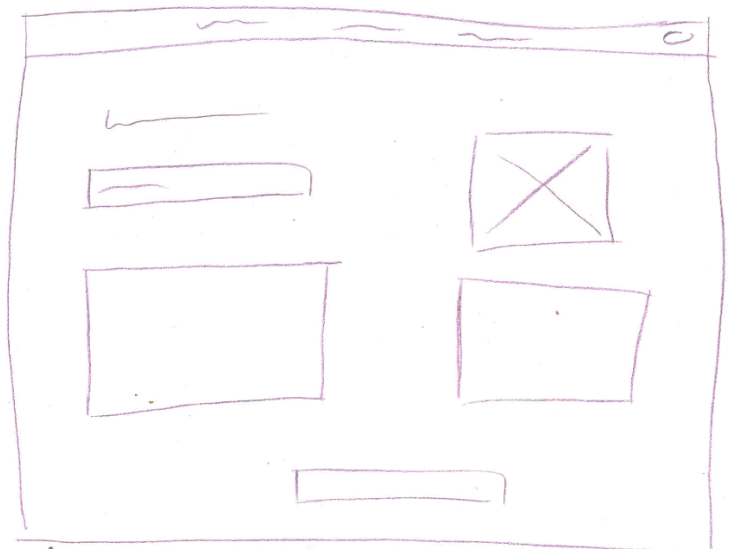


Imagen 12: Formulario agregar receta

Imagen 13: Formulario agregar receta, valores nutricionales

Imagen 14: Página información de receta



Imagen 15: Página de peso home

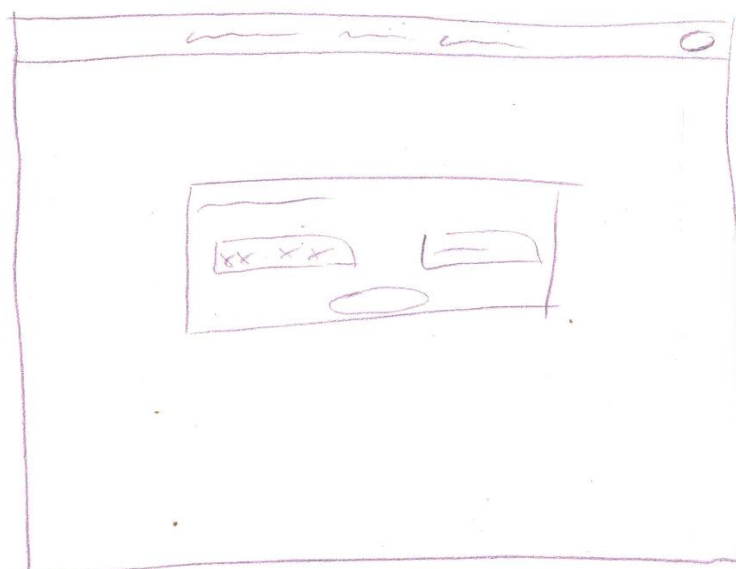


Imagen 16: Popup registrar peso

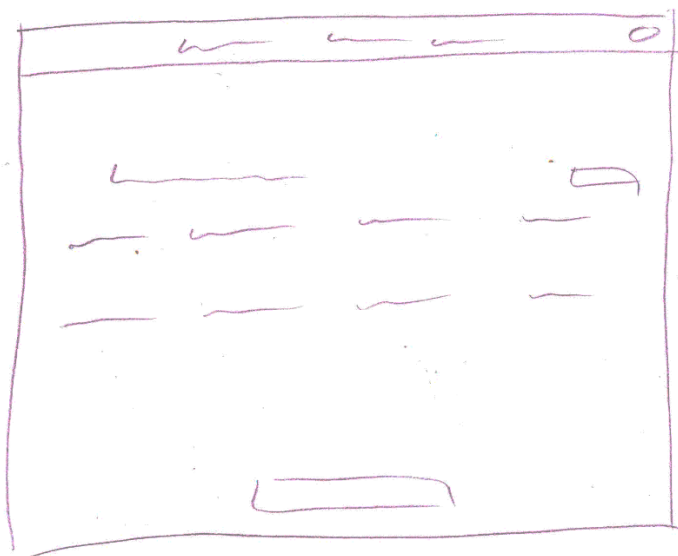


Imagen 17: Página ejercicios home

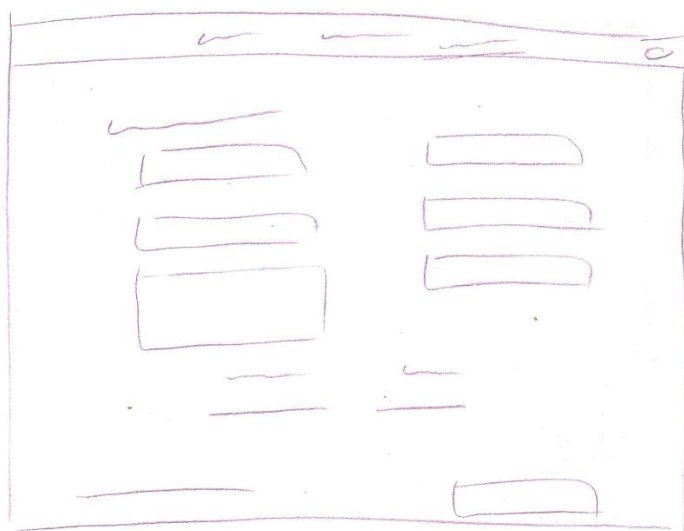


Imagen 18: Formulario agregar ejercicio realizado

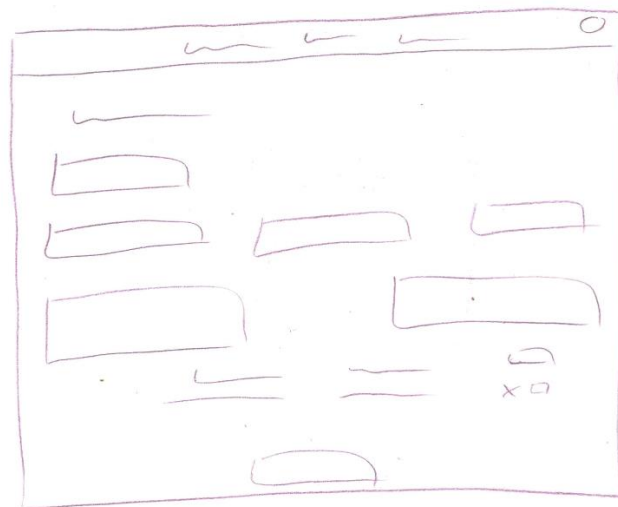


Imagen 19: Formulario proponer ejercicio

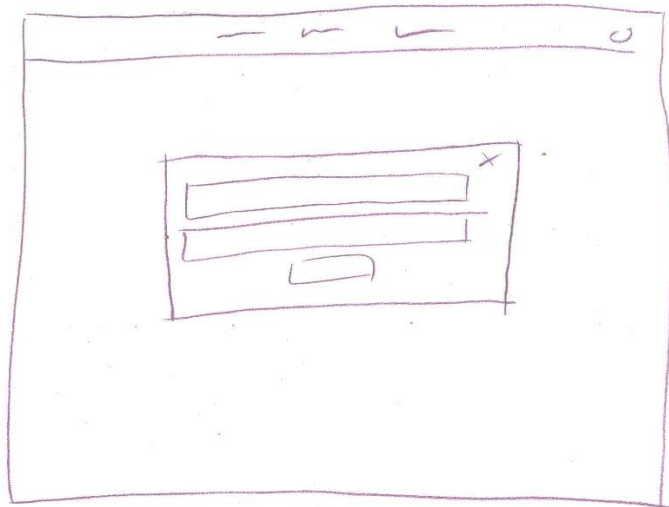


Imagen 20: Popup agregar enlace a ejercicio

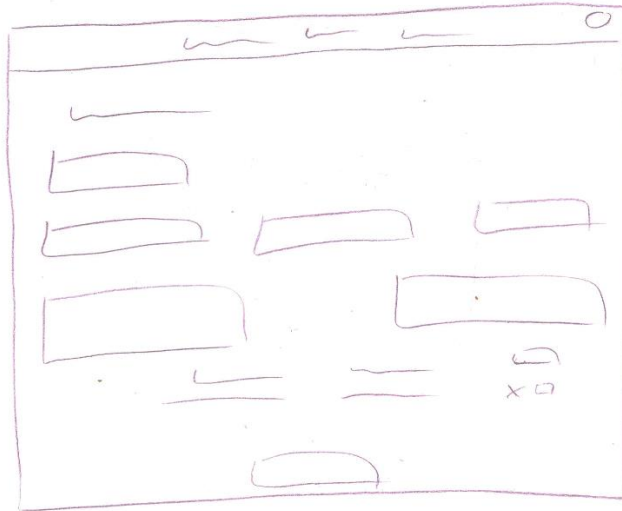


Imagen 21: Página ver información de ejercicio



Imagen 22: Página mensaje de error

Wireframing

Estas son las imágenes del Wireframing obtenidas desde Figma.



Imagen 23: Página de inicio, tabs Ejercicios

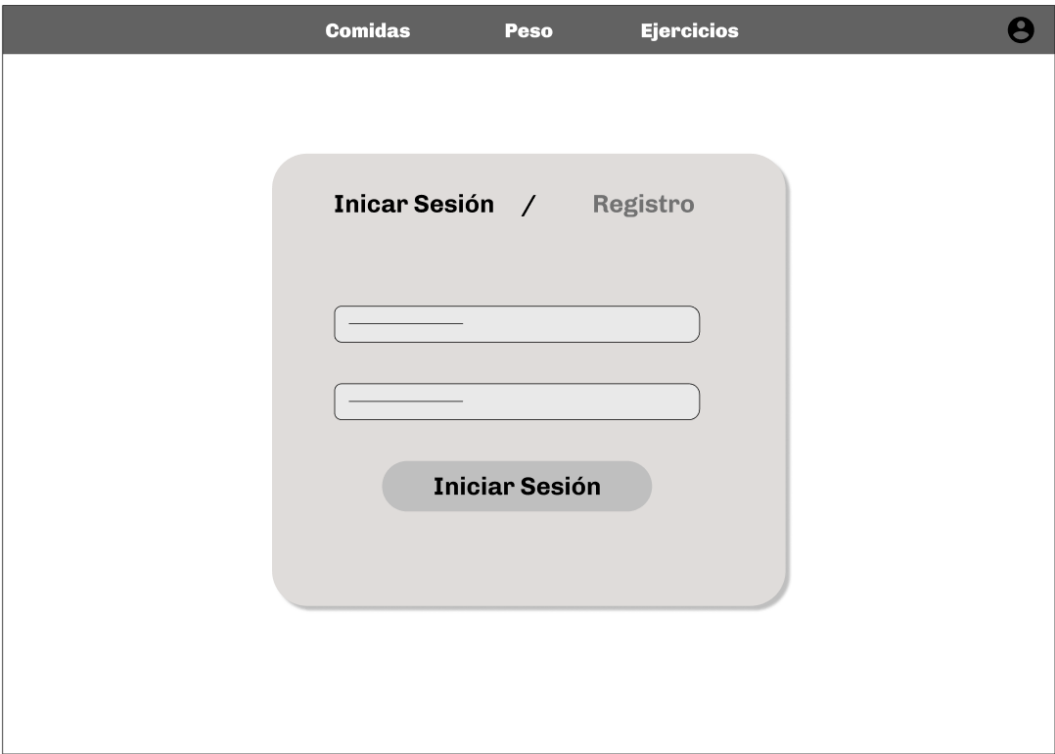
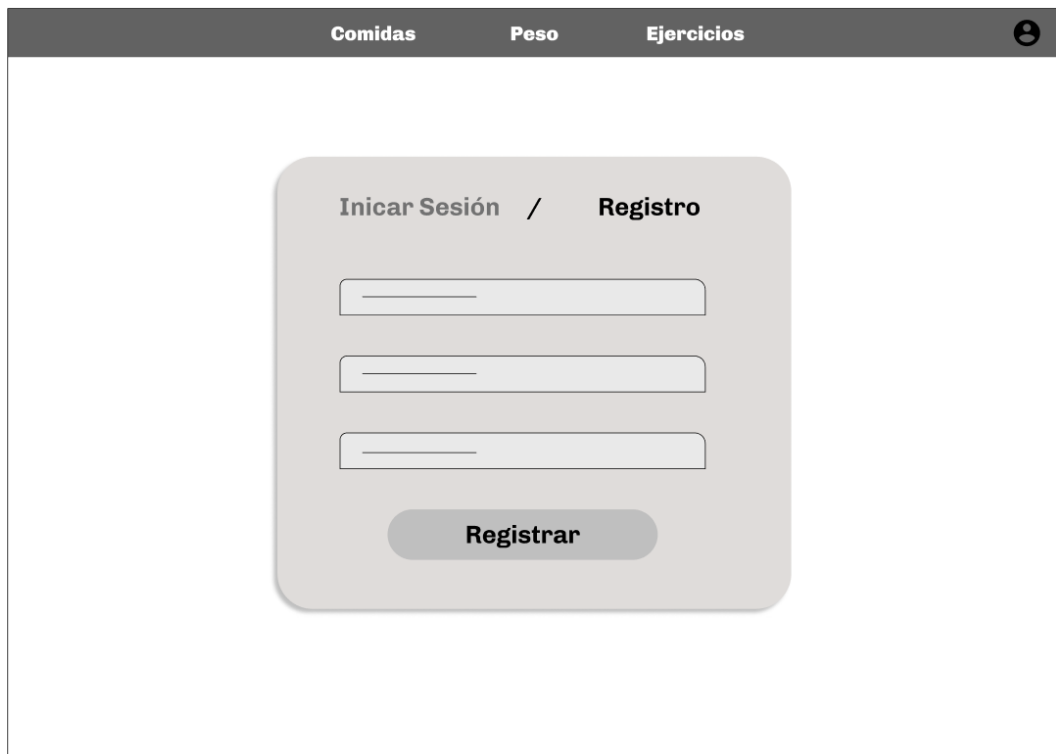


Imagen 24:Fomulario inicio de sesión



Comidas Peso Ejercicios

Inicar Sesión / Registro

Registrar

Imagen 25: Formulario de registro:



Comidas Peso Ejercicios

Rellena los siguientes datos

Registrar

Imagen 26: Formulario de registro, datos personales

[illegible]

Imagen 27: Página de perfil

Comidas

Peso

Ejercicios

Formulario cambio dato personal

Valor actual

Introduce el nuevo valor

Cancelar

Guardar

Imagen 28: Página de perfil, modificar datos

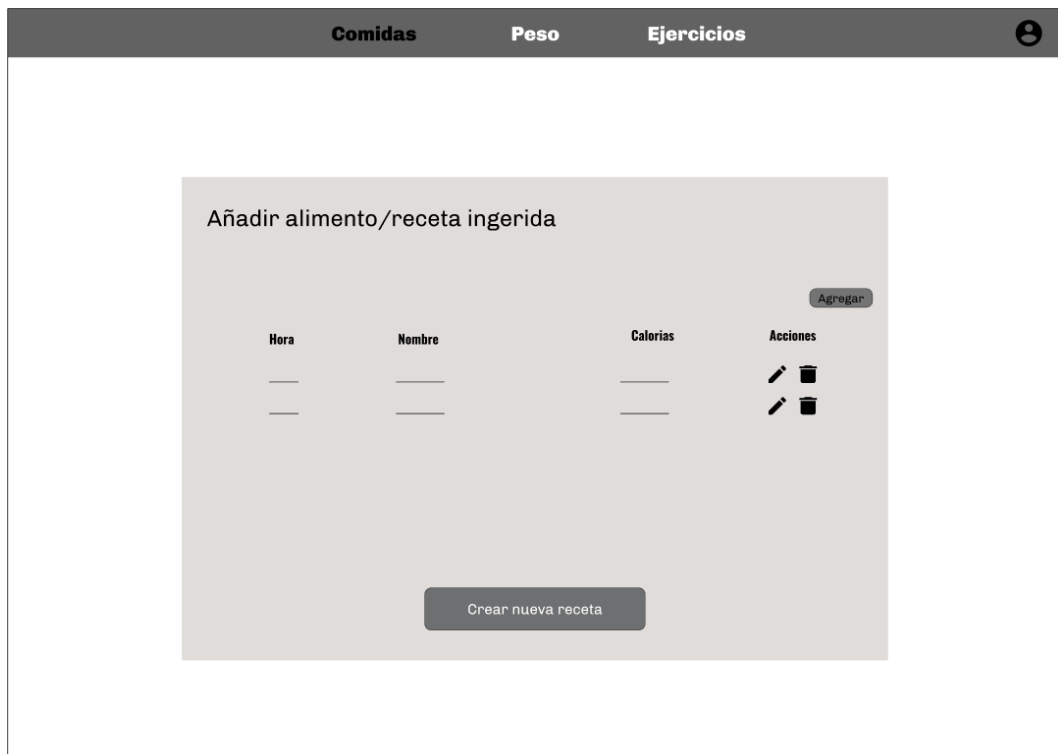


Imagen 29: Página de comidas home

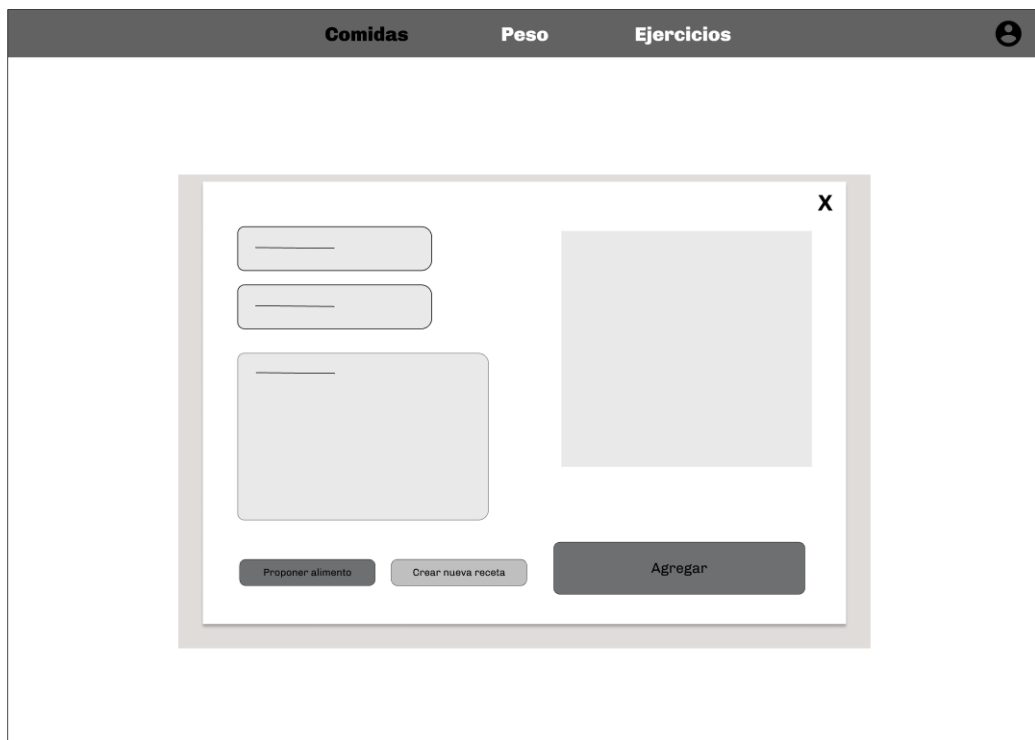


Imagen 30: Popup consumir alimento

Comidas Peso Ejercicios

Proponer Alimento

Imagen 31: Formulario proponer alimento

Comidas Peso Ejercicios

Proponer Alimento

Imagen 32: Formulario proponer alimento, valores nutricionales

Comidas Peso Ejercicios

Proponer Alimento

Formulario con dos columnas de tres campos de entrada cada una, y un botón 'Proponer' centrado al fondo.

Imagen 35: Formulario agregar receta, valores nutricionales

Comidas Peso Ejercicios

Informacion de Receta

Formulario con: tres campos de entrada a la izquierda; un placeholder de imagen con una 'X' en el centro; un campo de entrada debajo de la imagen; y una columna de seis campos de entrada a la derecha. Un botón 'Salir' está centrado al fondo.

Imagen 36: Página información de receta



Imagen 37: Pagina de peso home

The screenshot shows the 'Peso' page with a 'Nuevo registro de peso' (New weight record) popup form overlaid. The popup has a title bar with 'Nuevo registro de peso' and a close button 'X'. It contains two input fields for date and weight, a home icon, and an 'Añadir' (Add) button. The background shows the same line graph and table as in the previous image.

Imagen 38: Popup peso agregar peso



Imagen 39: Página peso home

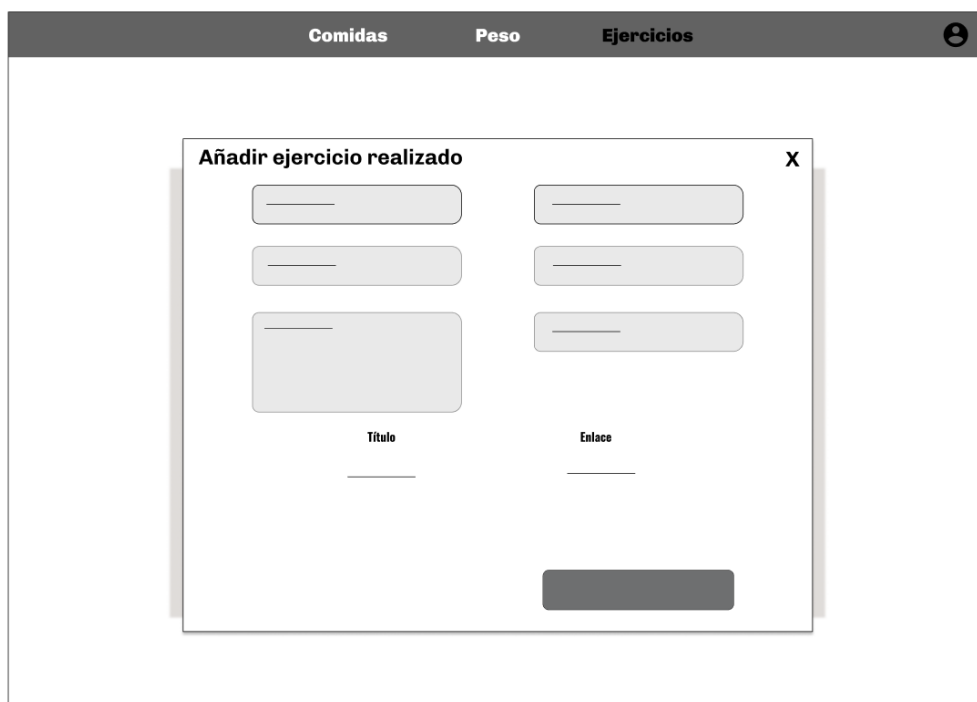


Imagen 40: Popup registrar peso

The image shows a web application interface with a dark header bar containing three tabs: 'Comidas', 'Peso', and 'Ejercicios'. The 'Ejercicios' tab is active, and a user profile icon is visible in the top right corner. Below the header, the main content area is titled 'Proponer Ejercicio'. It contains a form with several input fields: a single-line text field at the top, followed by three single-line text fields in a row, and two larger multi-line text areas. Below these fields are two labels, 'Titulo' and 'Enlace', each followed by a single-line text field. To the right of these fields is a button labeled 'Agregar'. Below the 'Agregar' button are two sets of icons, each consisting of a pencil and a trash can. At the bottom of the form is a large button labeled 'Enviar'.

Imagen 41: Pagina ejercicio home

This image shows the same 'Proponer Ejercicio' form as in the previous image, but with a modal popup open in the center. The modal has a title bar with a close button (X) in the top right corner. Inside the modal, there are two single-line text input fields stacked vertically, and a button labeled 'Agregar' at the bottom. The background form is slightly blurred, showing the same fields and buttons as before.

Pagina 42: Popup añadir enlace ejercicio



Imagen 43: Mensaje agradecimiento

The screenshot shows a web application interface with a dark header bar containing three tabs: 'Comidas', 'Peso', and 'Ejercicios'. A user profile icon is visible in the top right corner. The main content area is titled 'Información Ejercicio'. It features a form with several input fields. The form is organized into two columns. The left column has three input fields, and the right column has two input fields. Below the input fields, there are labels 'Titulo' and 'Enlace' with corresponding input fields. At the bottom of the form, there is a 'Guardar' (Save) button.

Imagen 44: Página información ejercicio



Imagen 45: Página mensaje de error

3. Diseño

3.1 Arquitectura hardware y software de la solución

- Hardware:
 - Equipo local o servidor
- Software:
 - Opción 1:
 - Docker:
 - Contenedor 1: Maneja servidor Symfony y base de datos Xampp
 - Contenedor 2: Maneja página en Angular
 - Opción 2:
 - Docker:
 - Contenedor único: Maneja servidor Symfony y base de datos Xampp
 - Angular:
 - Página en Angular

3.2 Modelado funcional de la solución. Diagramas de clase

En la siguiente imagen muestro como ha quedado el diagrama de clase final.

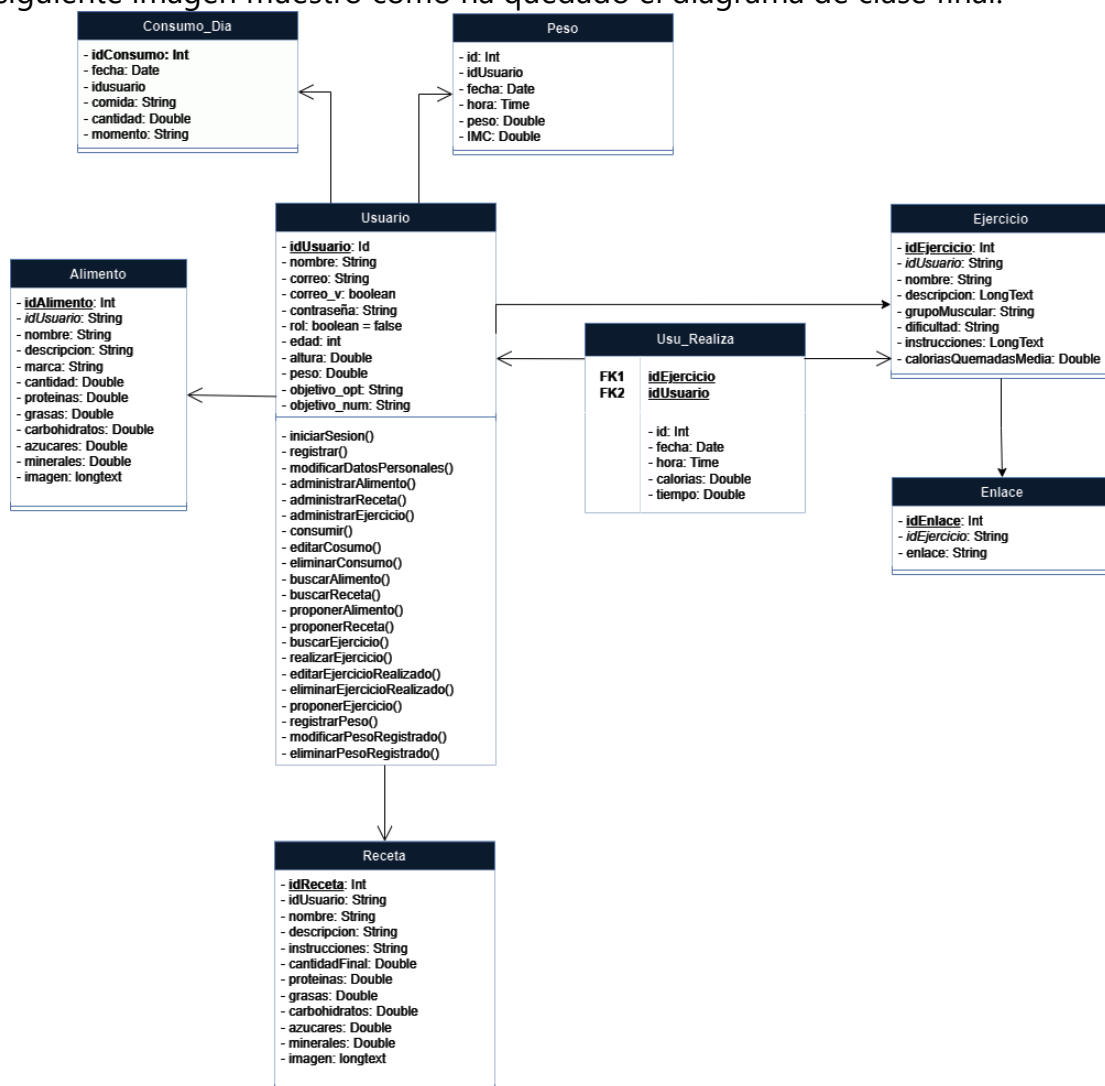


Imagen 46: Diagrama de clase

3.3 Modelado de datos. Modelo relacional. Diccionario de Datos

En la siguiente imagen muestro como ha quedado el modelo relacional de mi base de datos.

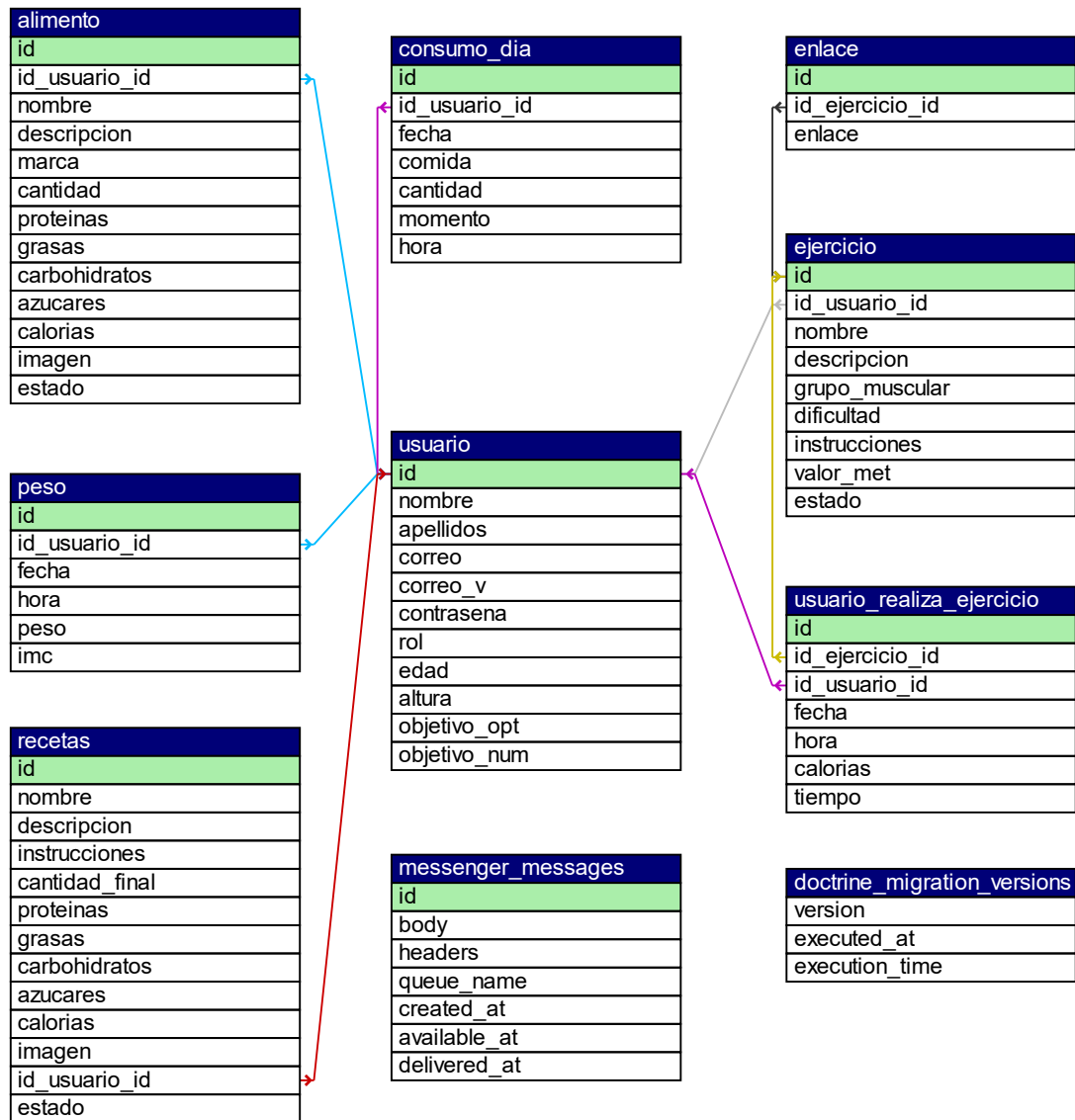


Imagen 47: Estructura Base de datos

3.4 Prototipado de la interfaz

Estas son las imágenes del prototipado de la web realizado en Figma.



Imagen 48: Página de inicio

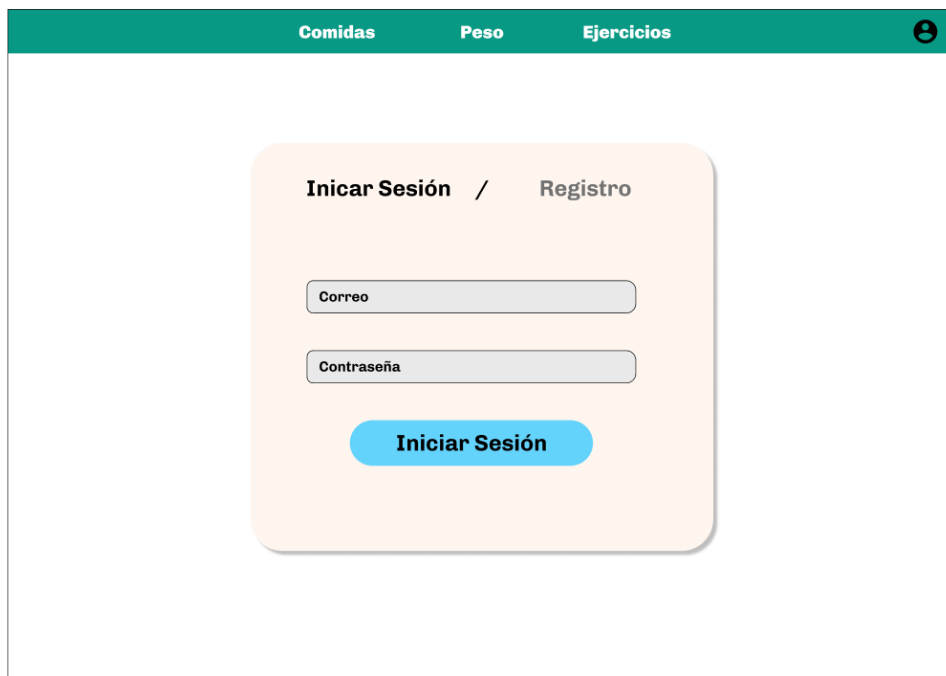



Imagen 49: Pagina de login

Comidas Peso Ejercicios 

Iniciar Sesión / Registro


Correo

Contraseña

Repetir contraseña

Registrar

Imagen 50: Pagina de registro

Comidas Peso Ejercicios 

Rellena los siguientes datos

Nombre Apellidos

Altura Peso

Registrar

Imagen 51: Pagina de registro, datos personales

The screenshot shows a web application interface with a teal header bar containing the navigation links "Comidas", "Peso", and "Ejercicios", along with a user profile icon. The main content area is titled "Estos son tus datos personales". It contains several input fields for personal data: "Nombre", "Apellidos", "Correo", "Altura", "Peso", and "Objetivo". Each of these fields has a small pencil icon to its right, indicating an edit function. At the bottom of the form area is a blue button labeled "Cerrar Sesión".

Imagen 52: Pagina de perfil

The screenshot displays a form titled "Formulario cambio dato personal" within the same teal-headered application. The form is designed for updating a specific piece of user data. It features two input fields: the first is labeled "Valor actual" and contains the text "PropiedadOld"; the second is labeled "Introduce el nuevo valor" and contains the text "PropiedadNuevo". Below these fields are two buttons: a teal "Cancelar" button and a blue "Guardar" button.

Imagen 53: Pagina de perfil, modificar dato

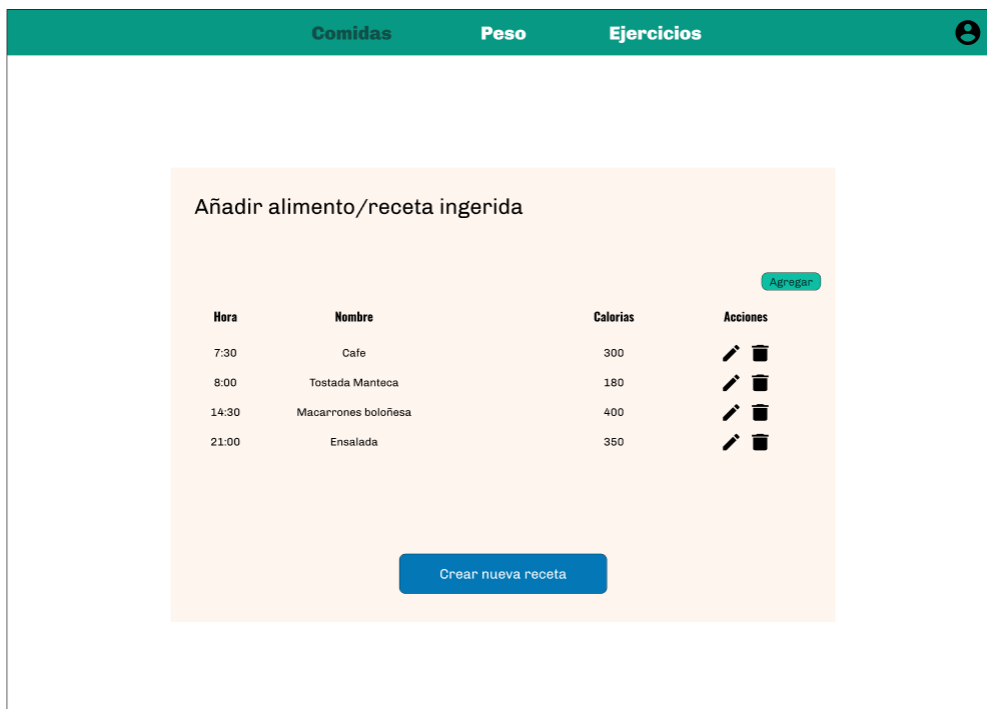


Imagen 54: Pagina de comidas home

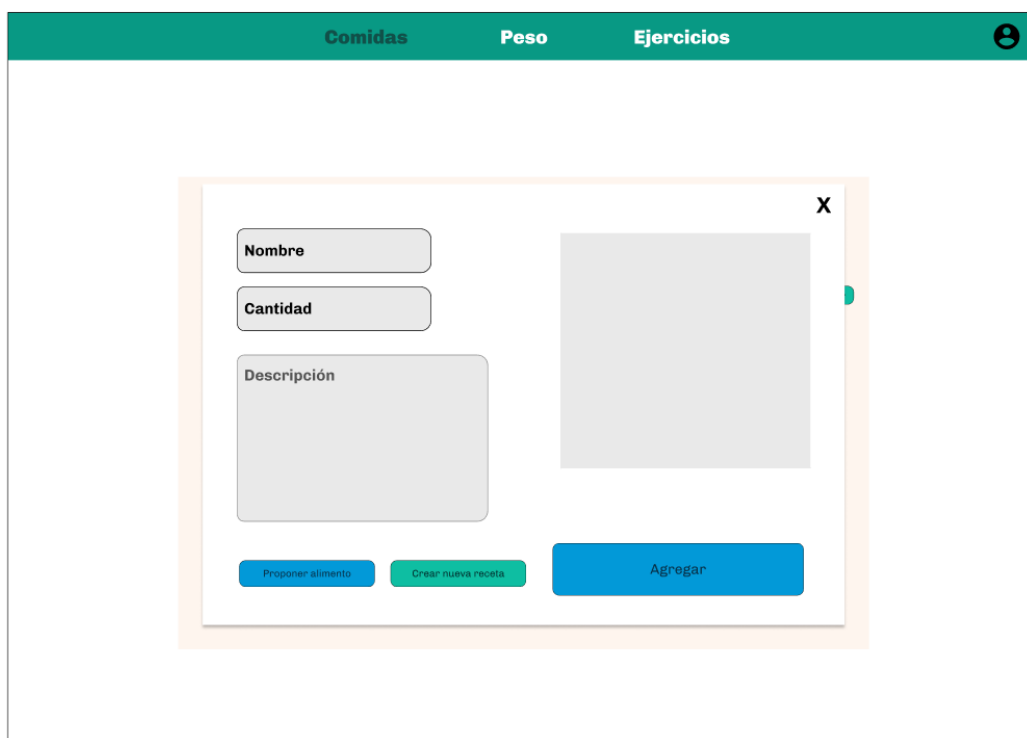


Imagen 55: Popup agregar alimento consumido

Formulario de Proponer Alimento. El formulario está dividido en dos columnas. La columna izquierda contiene un campo de texto para 'Nombre' y un campo de texto grande para 'Descripción'. La columna derecha contiene un campo de imagen con una 'X' diagonal, un campo de texto para 'Cantidad' y un campo de texto para 'Marca'. En la parte inferior del formulario hay un botón azul que dice 'Siguiente'.

Imagen 56: Formulario proponer alimento

Formulario de Proponer Alimento, datos personales. El formulario está dividido en dos columnas. La columna izquierda contiene tres campos de texto para 'Calorías', 'Proteínas' y 'Azúcares'. La columna derecha contiene tres campos de texto para 'Grasas', 'Vitaminas' y 'Carbohidratos'. En la parte inferior del formulario hay un botón azul que dice 'Proponer'.

Imagen 57: Formulario proponer alimento, datos personales

Comidas
Peso
Ejercicios

Nombre

Marca

Cantidad

Calorias

Grasas

Descripción

Salir

Vitaminas

Proteinas

Azucares

Carbohidratos

Imagen 58: Pagina información de alimento

Comidas
Peso
Ejercicios

Nombre

Instrucciones

Descripción

Siguiente

Imagen 59: Formulario agregar receta

Proponer Alimento

Calorías	Grasas
Proteínas	Vitaminas
Azúcares	Minerales
Carbohidratos	

Proponer

Imagen 60: Formulario agregar receta, datos nutricionales

Información de Receta

Nombre		Calorías
Cantidad final		Minerales
Instrucciones		Grasas
		Vitaminas
	Proteínas	
	Azúcares	
	Carbohidratos	

Descripción

Salir

Imagen 61: Página información de receta

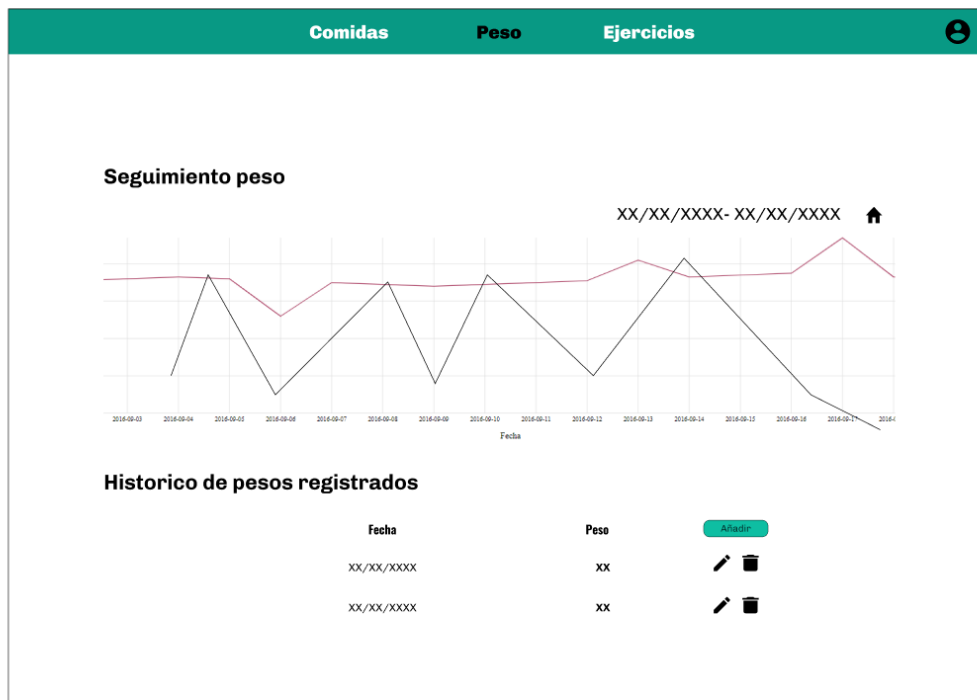


Imagen 62: Pagina de peso home

The screenshot shows the 'Peso' page with a 'Nuevo registro de peso' (New weight record) popup form overlaid. The form has a title bar with 'Nuevo registro de peso' and a close button 'X'. It contains two input fields: 'Fecha' (Date) and 'Peso (kg)' (Weight in kg), separated by a home icon. Below the fields is a blue 'Añadir' (Add) button. The background shows the same 'Seguimiento peso' graph and 'Historico de pesos registrados' table as in the previous image.

Imagen 63: Popup registrar peso

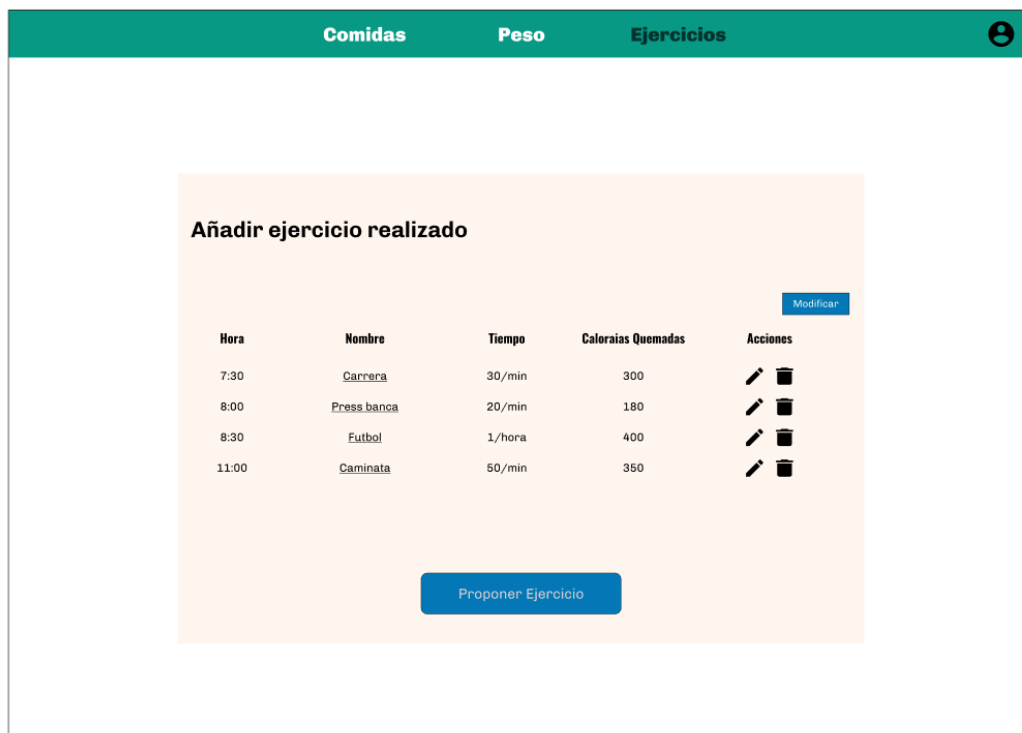


Imagen 63: Pagina de ejercicio home



Imagen 64: Popup añadir ejercicio realizado

Comidas Peso Ejercicios

Proponer Ejercicio

Nombre

Dificultad Grupo Muscular Valor MET

Descripción

Instrucciones

Título	Enlace	Agregar
Ejemplo 1	Ver video 1	
Ejemplo 2	Ver video 2	

Imagen 65: Formulario proponer ejercicio

Comidas Peso Ejercicios

Proponer Ejercicio

Nombre

Dificultad Grupo Muscular Valor MET

Descripción

Instrucciones

Título

URL

Título	Enlace	Agregar
Ejemplo 1	Ver video 1	
Ejemplo 2	Ver video 2	

Imagen 66: Popup añadir enlace



Imagen 67: Mensaje de agradecimiento

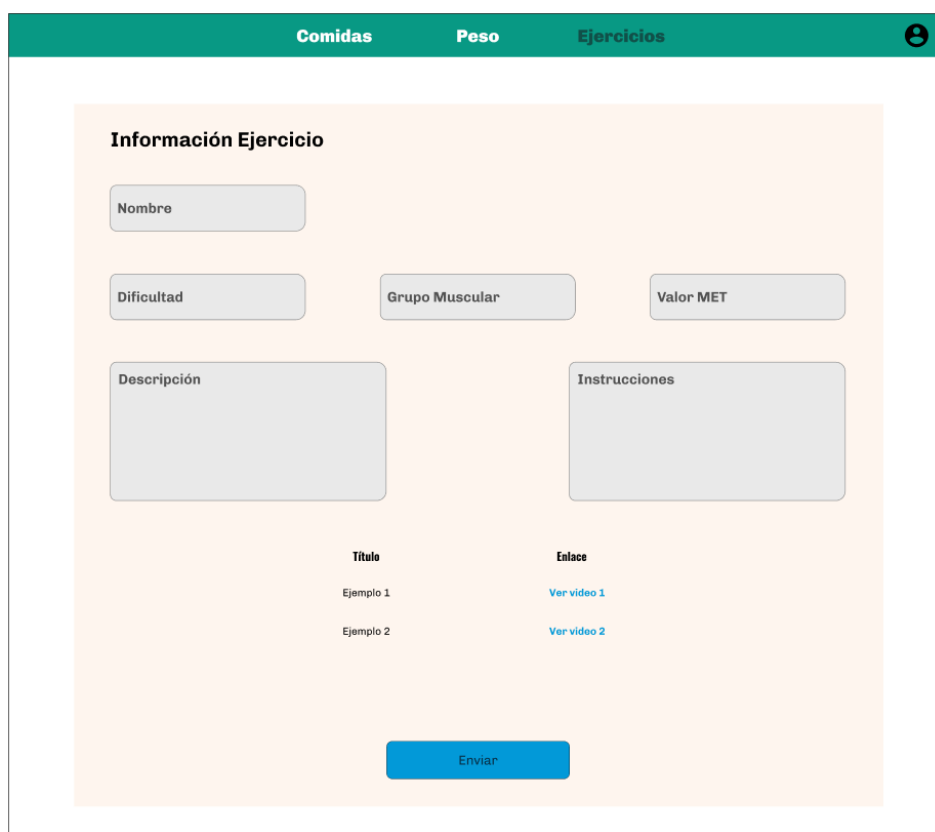


Imagen 68: Pagina información de ejercicio

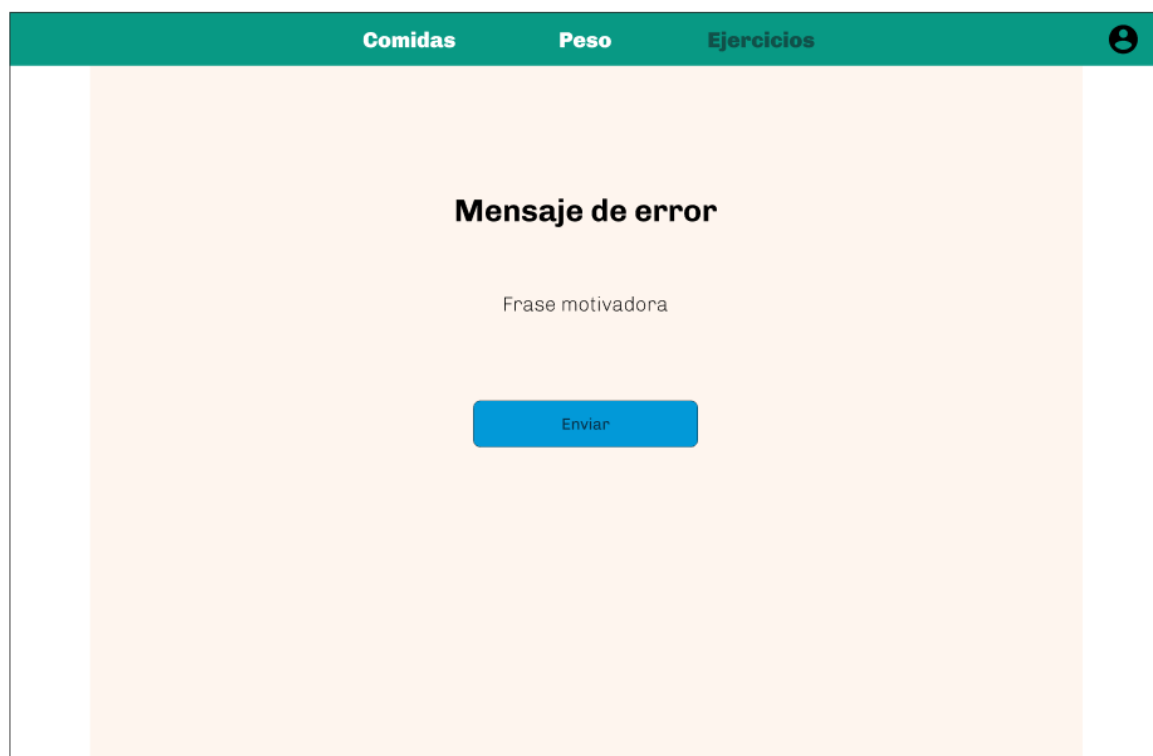


Imagen 69: 'Página mensaje de error

4. Implementación

La instalación del proyecto se ha desarrollado al completo para realizarse en un entorno Linux con Ubuntu, o un sistema Linux con el sistema de gestión de paquetes APT.

El resultado final sería la página de Angular desplegada usando la API de Symfony que gestione la base de datos poblada.

4.1 Requisitos de instalación y ejecución

Para la instalación de este proyecto no existen unos requisitos indispensables como tal, ya que proporciono la posibilidad de instalarlos mediante un script, o en caso de que ya se cuente con las dependencias instaladas, el script para lanzar el proyecto.

Los scripts de instalación, además de las instrucciones, se encuentran en el repositorio de Github:

- Wellnesstrack-completo: [WellnessTrack-completo](#)

El código fuente de ambas partes del proyecto está disponible en los siguientes repositorios de GitHub:

- API (Symfony): [WellnessTrack-Back](#)
- Página web (Angular): [WellnessTrack-Front](#)

Dependiendo del método de despliegue elegido la forma de despliegue cambiara, las instrucciones para ambas opciones son las siguientes:

Método 1 (Recomendado):

1. Base de datos y API:

- Se utiliza el contenedor Docker [hehedaniel/wellnesstrackback](#), que incluye:
 - La API desarrollada en Symfony (v5.8.19) con Doctrine, que se inicia automáticamente junto con el contenedor.
 - El servidor MySQL proporcionado por XAMPP (v8.2.12), que se inicia automáticamente junto con el contenedor.

2. Página web

- Se utiliza el contenedor Docker [hehedaniel/wellnesstrackfront](#), que ejecuta la página web desarrollada en Angular (v17.3.10). Esta página web se iniciará automáticamente al arrancar el contenedor.
- Las versiones utilizadas para el front-end son:
 - Angular CLI: v17.3.8
 - Node.js: v20.14.0
 - Npm: v10.7.0

3. Instalación:

- Ejecutar en la terminal el script "lanzarDockerProyecto.sh"
- Este script instalará el único requerimiento que será Docker
- Y también creará los contenedores y los lanzará, dando como resultado el proyecto desplegado y listo para su uso.
 - Backend
 - Acceda al backend en <http://localhost:8000>.
 - Frontend
 - Acceda al frontend en <http://localhost:4200>.
 - PhpMyAdmin
 - Administre bases de datos en <http://localhost:8080>.

Método 2:

1. Base de datos y API:

- Se utiliza el contenedor Docker [hehedaniel/wellnesstrackback](#), que incluye:
 - La API desarrollada en Symfony (v5.8.19) con Doctrine, que se inicia automáticamente junto con el contenedor.
 - El servidor MySQL proporcionado por XAMPP (v8.2.12), que se inicia automáticamente junto con el contenedor.

2. Página web:

- En este método, la página web no se ejecuta en un contenedor. En su lugar, el código de Angular se despliega directamente desde el equipo del usuario, ejecutándolo mediante la terminal.

3. Instalación:

- Ejecutar el script "instalarRequisitos.sh"
- Esto instalará todo los requisitos necesarios para la ejecución, que serán:
 - Nodejs
 - Npm
 - Angular cli
 - Php
 - Docker
 - Docker compose
 - Repositorio Wellnesstrack-front
- Este script terminará mostrando las versiones de las dependencias instaladas.
- Una vez instaladas las dependencias deberemos ejecutar el script "lanzarSistema.sh" el cual creará el contenedor para el backend y lanzará la página web.

Ejecución una vez instalado:

Estos son los pasos que debemos seguir cuando queramos lanzar el proyecto desde la segunda vez en adelante:

- API y Base de datos:
 - Iniciar el contenedor Docker [hehedaniel/wellnesstrackback](#).
 - La base de datos y el servidor Symfony se lanzará automáticamente.
- Página web (según el método elegido):
 - En el Método 1, iniciar el contenedor [hehedaniel/wellnesstrackfront](#), que cargará automáticamente la página web.
 - En el Método 2, lanzar manualmente el front-end desde la carpeta raíz del proyecto de Angular en la terminal con:



4.2 Implementación funcional. Clases

Symfony

- Controller
 - Inicio
 - Alimento
 - ConsumoDia
 - Ejercicio
 - Enlace
 - Peso
 - Recetas
 - Usuario
 - UsuarioRealizaEjercicio
- Entity
 - Alimento
 - ConsumoDia
 - Ejercicio

- Enlace
- Peso
- Recetas
- Usuario
- UsuarioRealizaEjercicio
- Repository
 - Alimento
 - ConsumoDia
 - Ejercicio
 - Enlace
 - Peso
 - Recetas
 - Usuario
 - UsuarioRealizaEjercicio
 -
- Util
 - Cbbdd
 - Respuesta

Angular

- Components
 - comidas
 - comidas-home
 - form-administrar-alimento-propuesto
 - form-administrar-receta-propuesta
 - form-consumir
 - form-editar-consumo
 - form-eliminar-consumo-diario
 - form-proponer-alimento
 - form-proponer-comida-nutrientes
 - form-proponer-receta
 - info-comida
 - proponer-comida-stepper
 - resumen-alimentos
 - switch-proponer-comida
 - tabla-comidas
 - vista-alimento
 - ejercicios
 - ejercicios-home
 - form-administrar-ejercicio-propuesto
 - form-editar-ejercicio-realizado
 - form-editar-enlace-ejercicio
 - form-eliminar-ejercicio-realizado
 - form-proponer-ejercicio
 - form-realizar-ejercicio

- info-ejercicio
 - tabla-ejercicios
- global
 - administrar-propuestas
 - home
 - pagina-error
 - spinner-mostrar
 - tabs-comida-ejercicio
 - toolbar
 - youtube-video-player
- pesos
 - form-anadir-peso
 - form-delete-peso
 - form-editar-peso
 - peso-grafica
 - pesos-home
 - tabla-pesos
- usuario
 - form-actualizar-datos-usuario
 - formulario-tabs
 - login
 - registro
 - registrodatos
 - usuario-perfil
- Models
 - alimento
 - peso
 - usuario
- Services
 - alimentos
 - auth
 - ejercicios
 - peso
 - responsive-info
 - usuario

4.3 Implementación del modelo de datos. Tablas

Tras la realización de la base de datos he obtenido las tablas:

- Usuario
- Alimentos
- Recetas
- ConsumoDia
- Ejercicios
- EnlacesEjercicios

- UsuariorealizaEjercicios
- Peso

5. Pruebas

5.1. Pruebas de módulos

En este primer punto muestro como es la respuesta que recibo por parte de la Api creada en Symfony

Prueba número 1:

Esta prueba ha consistido en la creación de una nueva receta mediante una petición POST al servidor Symfony.

POSThttp://localhost:8000/recetas/crearSend

QueryHeaders 2AuthBody 1TestsPre Run

JSONXMLTextFormForm-encodeGraphQLBinary

JSON ContentFormat

```
1 {
2   "nombre": "Macarrones con carne",
3   "descripcion": "Los macarrones con carne que me
4     ha hecho mi abuela siempre que tanto me
5     gustan",
6   "instrucciones": "Hervir los macarrones al
7     punto y hacer la carne picada con tomate,
8     echarle queso y gratinar",
9   "cantidadFinal": 600,
10  "grasas": 300,
11  "carbohidratos": 500,
12  "proteinas": 200,
13  "azucares": 0,
14  "vitaminas": 30,
15  "calorias": 1000,
16  "imagen": "imagen en base 64 no disponible",
17  "idUser": "5"
18 }
```

Status: 200 OKSize: 426 BytesTime: 25 ms

ResponseHeaders 11CookiesResultsDocs

```
1 {
2   "code": 200,
3   "respuesta": {
4     "nombre": "Macarrones con carne",
5     "descripcion": "Los macarrones con carne que me ha hecho mi abuela siempre que
6       tanto me gustan",
7     "instrucciones": "Hervir los macarrones al punto y hacer la carne picada con
8       tomate, echarle queso y gratinar",
9     "cantidadFinal": 600,
10    "proteinas": 200,
11    "grasas": 300,
12    "carbohidratos": 500,
13    "azucares": 0,
14    "vitaminas": 30,
15    "calorias": 1000,
16    "imagen": "imagen en base 64 no disponible",
17    "id": 6
18   }
19 }
```

Prueba número 2:

Esta prueba ha consistido en realizar una modificación a la receta creada anteriormente mediante una petición PUT.

PUThttp://localhost:8000/recetas/editarSend

QueryHeaders 2AuthBody 1TestsPre Run

JSONXMLTextFormForm-encodeGraphQLBinary

JSON ContentFormat

```
1 {
2   "id": "6",
3   "nombre": "Macarrones con carne y queso
4     gratinado",
5   "descripcion": "Los macarrones con carne que me
6     ha hecho mi abuela siempre que tanto me
7     gustan",
8   "instrucciones": "Hervir los macarrones al
9     punto y hacer la carne picada con tomate,
10    echarle queso y gratinar",
11  "cantidadFinal": 600,
12  "grasas": 300,
13  "carbohidratos": 500,
14  "proteinas": 200,
15  "azucares": 0,
16  "vitaminas": 30,
17  "calorias": 1000,
18  "imagen": "imagen en base 64 no disponible",
19  "idUser": "5"
20 }
```

Status: 200 OKSize: 444 BytesTime: 34 ms

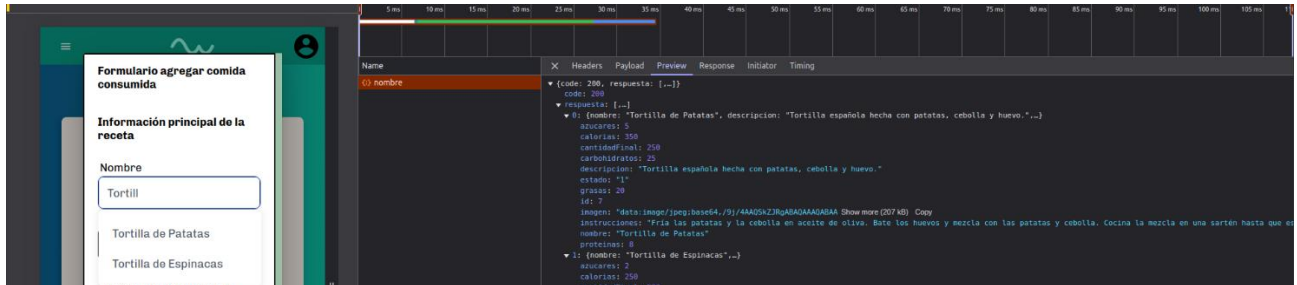
ResponseHeaders 11CookiesResultsDocs

```
1 {
2   "code": 200,
3   "respuesta": {
4     "nombre": "Macarrones con carne y queso gratinado",
5     "descripcion": "Los macarrones con carne que me ha hecho mi abuela siempre que
6       tanto me gustan",
7     "instrucciones": "Hervir los macarrones al punto y hacer la carne picada con
8       tomate, echarle queso y gratinar",
9     "cantidadFinal": 600,
10    "proteinas": 200,
11    "grasas": 300,
12    "carbohidratos": 500,
13    "azucares": 0,
14    "vitaminas": 30,
15    "calorias": 1000,
16    "imagen": "imagen en base 64 no disponible",
17    "id": 6
18   }
19 }
```

5.2. Pruebas de integración

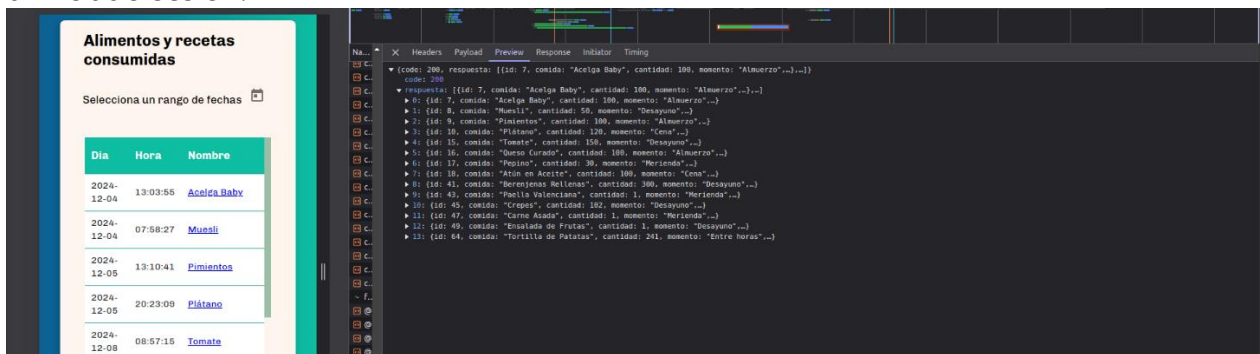
Prueba numero 1:

En esta prueba muestro como se integra la api con angular mediante la realización de peticiones a la api conforme el usuario escribe en la barra de búsqueda. Se ve como devuelve todos los alimentos que encuentra que contienen la cadena escrita por el usuario.



Prueba numero 2:

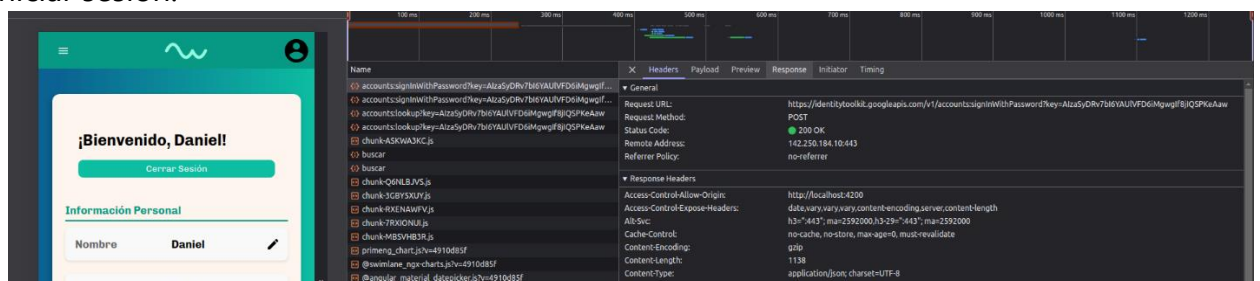
En esta prueba muestro como la api devuelve los alimentos consumidos por el usuario que ha iniciado sesión.



5.3 Pruebas del sistema

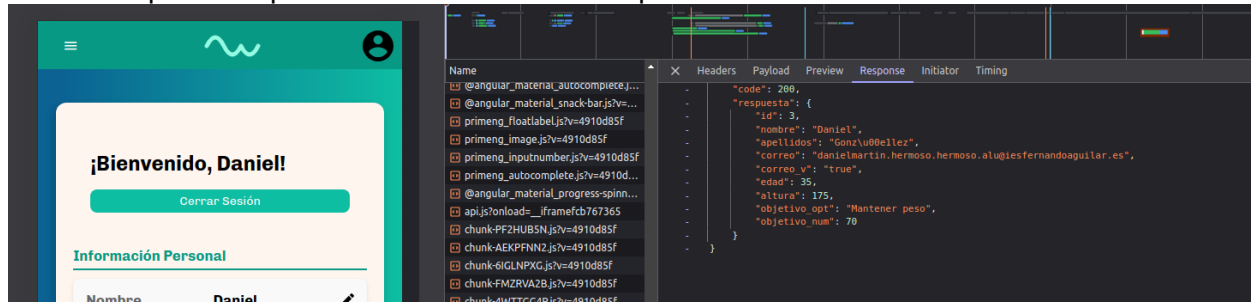
Prueba numero 1:

En la siguiente imagen muestro como a la hora de realizar la acción de iniciar sesión el sistema se comunica con Firebase para comprobar los datos con los que el usuario intenta iniciar sesión.



Prueba numero 2:

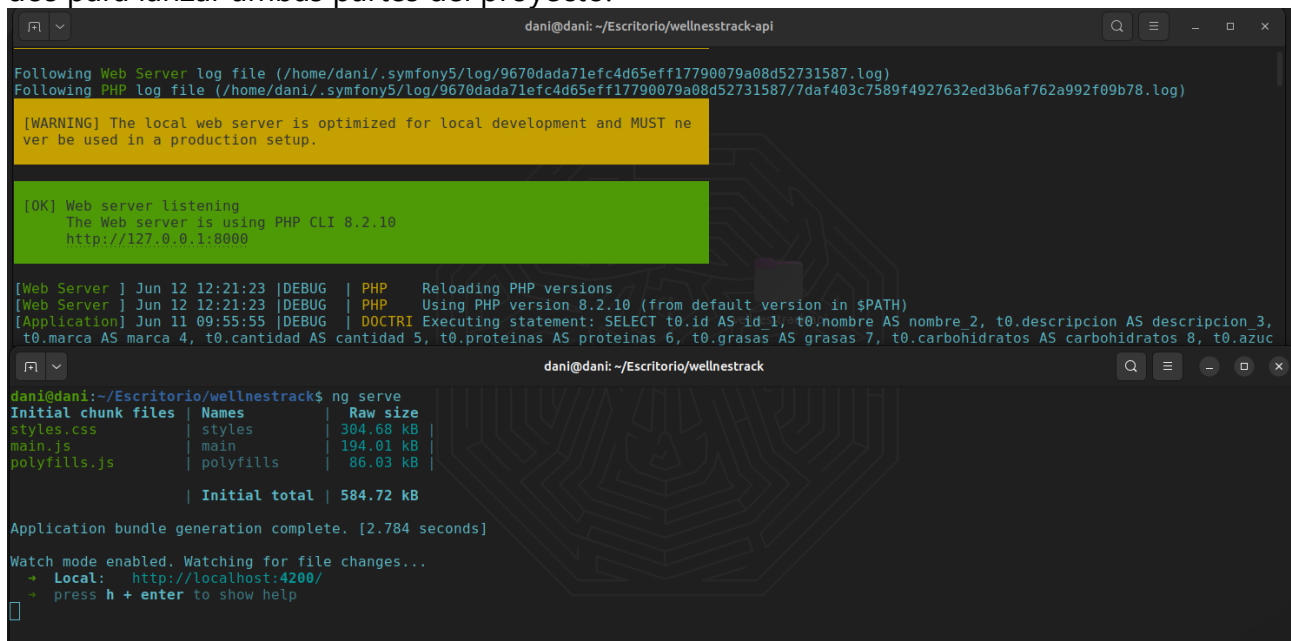
Aquí se muestra como una vez un usuario que ya ha iniciado sesión accede a su perfil se realiza una petición para mostrarles sus datos personales



5.4 Pruebas de instalación

Prueba número 1 y numero 2:

En la siguiente imagen se pueden ver las dos consolas al ejecutarse los diferentes comandos para lanzar ambas partes del proyecto.



6. Conclusiones

6.1 Grado de consecución de los objetivos inicialmente planteados

Los objetivos requeridos en el conjunto del proyecto han sido completamente conseguidos, junto a los objetivos planteados por mí que han sido mayormente conseguidos a falta de los objetivos de ampliación que no he podido conseguir su realización.

Estos últimos objetivos no he podido llegar a completarlos debido mayormente a mi falta de conocimiento sobre el cómo y donde poder realizarlos, la falta de esto he intentado suplirla con la búsqueda de información y mucha prueba y error, pero finalmente no he podido llevarlo a cabo.

La no consecución de estos objetivos de ampliación al completo no implica que diversas partes del mismo no se hayan conseguido, como puedo ser el despliegue del front en páginas como [Netlify](#) o [Github Pages](#), ya que esto sí que ha conseguido pero al ser un proyecto que involucra tanto front como back la falta de, en este caso el back, hace que al no estar completo este despliegue sea inútil, de la misma forma, el conseguir desplegar el back en plataformas como [AWS](#) ha sido satisfactorio casi al completo ya que únicamente ha faltado el poder contar con el certificado necesario para que el acceso al servidor Symphony fuera seguro.

Los objetivos planteados según lo estudiado a lo largo del curso ha sido completamente satisfactoria.

6.2 Dificultades encontradas

A lo largo de la realización del proyecto me he encontrado con diversos problemas, algunos de ellos no he sido capaz de resolverlos por mí en su momento, como puede ser la obtención de un certificado para poder realizar peticiones seguras al servidor, en cambio, otras dificultades sí que han sido resueltas conforme se iban encontrado de forma sencilla.

6.3 Propuestas de mejora y posibles ampliaciones

Desde el momento en el que planteé el proyecto tuve como meta una propuesta de mejora la cual he estado muy cerca de cumplir:

Esta consistía en el uso de una api de terceros que me proporcionaba los datos registrados en relojes y pulseras inteligentes sobre los ejercicios que los usuarios realizaban diferentes entramientos y ejercicios.

Esta propuesta de mejora la vine gestionando desde el día en el que planteé el proyecto, poco tiempo después encontré un tercero que podría proporcionarme los datos que necesitaba con el cual contacte, explique y plantee el uso que quería darle a la api que él podría

proporcionarme, con el resultado de llegar a tener la api a mi disposición para su uso durante el tiempo que establecimos.

Esta propuesta de mejora no la he llegado a cumplir debido a la falta de tiempo para el desarrollo de la parte en la que estaría.

Otra mejora que he querido introducir fue la de que el usuario tuviera la posibilidad de elegir entre un modo claro y modo oscuro a la hora de visualizar la página web.

Esta mejora no he podido llevarla a cabo debido a la falta de tiempo y mi dificultad para el diseño de interfaces teniendo que hacer uso de una de mis peores habilidades como es la elección de colores.

Otra propuesta de mejora que he querido hacer ha sido la comentada anteriormente de desplegar el proyecto en un servidor de terceros, esta no la he podido completar debido a la dificultad que esto me ha supuesto debido a mi falta de conocimientos y practica en esta parte del proceso, aquí desgloso un poco los intentos que he realizado y los problema que he tenido en ellos.

Front-end con Angular.

Netlify: El despliegue de una página web en Netlify es de los más sencillos que puede haber, solo consta de un par de pasos. El problema ha sido que, aunque llegue a desplegar la aplicación perfectamente al estar está ligada a una parte de backend las peticiones tenían que ser *'https'* lo que supuso el no poder llegar a completar el despliegue totalmente remoto, ya que si configuraba Angular para recibir peticiones desde el propio equipo (*'localhost:8000...'*) la página funciona perfectamente pero no cumple mi objetivo de un despliegue en remoto al completo.

Github Pages: Al igual que en Netlify Github Pages proporciona una forma de desplegar páginas de forma muy sencilla, solo se debe configurar de donde debe obtener los archivos y listo, tras hacerlo de forma exitosa me encontré con el mismo problema que en Netlify, las peticiones deben ser mediante *'https'*, algo que no supe solucionar ya que necesitaba la instalación de certificados para hacer un servidor Nginx seguro con la certificación pertinente.

Heroku: En Heroku he intentado desplegar la Api de Symfony, también de forma sencilla ya que es solo configurar el repositorio del que extraer el código, en este caso el problema han sido las versiones de PHP ya que Heroku usaba una versión superior de PHP, siendo este problema algo fácil de solucionar aparentemente, no fui capaz de solucionarlo.

Firebase Hosting: De igual manera que los anteriores el despliegue de una página web con Angular es también muy sencillo, en este despliegue me ocurrió lo mismo que los anteriores, las peticiones son bloqueadas al tratarse de peticiones mediante *'http'*.

AWS: En Amazon Web Service sí que fue algo más complicado gestionar todo ya que es diferente al resto, sin embargo fue la plataforma en la que más lejos llegue al llegar a tener la api de Symfony funcional junto con la base de datos en una instancia Ubuntu, aquí el

problema vino a la hora de desplegar Angular ya que al hacerlo en un primer intento la instancia dejo de funcionar al completo a mediados de la instalación, cuando lo volví a intentar la instalación se completó sin ningún problema, pero surgió otro, al estar el front y el back en instancias diferentes, y por ende, en redes diferentes, las peticiones debían ser al igual que en Netlify o en Github Pages mediante 'https' volviendo al problema comentado anteriormente. Sin embargo al estar en una plataforma con libertad para realizar la instalación que necesitara volví a intentar el despliegue pero esta vez realizando toda la instalación en una misma instancia de AWS ya que de esta forma sería replicar la instalación que tantas veces he hecho en local pero un servidor en remoto, aquí el problema vino cuando tras la instalación de Symfony y una vez casi finalizada la instalación de Angular la instancia de AWS volvió a dejar de funcionar completamente arruinando una vez más mis esperanzas de conseguir el despliegue.

DigitalOcean: En DigitalOcean lo que intente fue hacer el despliegue mediante mis imágenes Docker, el problema vino a raíz de que desde varias fuentes viese que se trataba de un despliegue gratuito este a la hora de realizar el despliegue este me indicaba que era un despliegue de pago, lo que provocó que no pudiera realizarlo.

7. Bibliografía y recursos on-line

- **Symfony**

- Crear api con Symfony. [Video](#)
- Crear documentación en PHP. [Video](#)
- Crear web con Symfony. [Video](#)
- Creación de Api en Symfony. Canal [Latte And Code](#)
 - Curso de Symfony 5, Capítulos:
 - Configuración del proyecto. [Video](#)
 - Controllers y rutas. [Video](#)
 - Servicios y container. [Video](#)
 - Base de datos e integración con Doctrine. [Video](#)
 - Formularios. [Video](#)
 - Peticiones HttpClient. [Video](#)
- Creación de login con Doctrine. [Video](#)
- Introducción a Symfony. [Video](#)
- Introducción a Doctrine y Entidades. [Video](#)
- Curso sobre Symfony – Repository . [Video](#)
- Curso sobre Symfony – Cud. [Video](#)
- Curso sobre Symfony – Registro de usuarios. [Video](#)
- Curso sobre Symfony – Controlador. [Video](#)
- Curso sobre Symfony – Métodos Mágicos. [Video](#)
- Curso sobre Symfony – Relaciones. [Video](#)
- Introducción e Instalación Symfony. [Video](#)
- Relaciones Symfony y Doctrine. [Video](#)
- Relaciones Doctrine. [Video](#)

- **Angular**

- Uso del componente *Tabs* de Angular Material. [Video](#)
- Uso del componente *Tabs* de Angular Material. [Video](#)

- **Firebase**

- Como usar Firebase. [Video](#)
- Login Angular, Firebase y Bootstrap. [Video](#)
- Email de verificación Firebase. [Video](#)

- **Base de datos**

- Relaciones MySQL. [Video](#)

- **Docker**

- Docker con Apache, PHP, MySQL y PHPMyAdmin. [Video](#)

- **Despliegue**

- Angular con servidor Nginx en AWS. [Video](#)
- Angular con servidor Nginx. [Video](#)
- Build de Angular en servidor Nginx. [Video](#)
- Angular con Nginx en Ubuntu server. [Video](#)
- Deploy de Angular con Nginx. [Video](#)
- Base de datos en Heroku. [Video](#)
- Imagen Docker en Google Cloud Run. [Video](#)
- Deploy contenedor Docker. [Video](#)
- Deploy contenedor Docker en BlueOcean. [Video](#)
- Deploy contenedor Docker en AWS. [Video](#)
- Deploy Api en cPanel. [Video](#)
- Deploy Api Symony. [Video](#)
- Deploy Api en Vercel. [Video](#)
- Deploy en Vercel. [Video](#)
- Deploy Docker. [Video](#)
- Deploy Api en Heroku. [Video](#)

8. Glosario de términos

8.1 Informáticos

- **Docker**: Plataforma de software que permite a los desarrolladores empacar, enviar y ejecutar aplicaciones en contenedores.
- **MySQL**: Sistema de gestión de bases de datos relacional, utilizado para almacenar y gestionar datos.
- **API**: Interfaz de programación de aplicaciones, que permite la comunicación entre diferentes componentes de software.
- **Symfony**: Framework de desarrollo web PHP utilizado para crear aplicaciones web y API.
- **Symfony CLI**: Herramienta de línea de comandos para trabajar con proyectos Symfony.
- **Doctrine**: ORM (Mapeo Objeto-Relacional) para PHP, utilizado en proyectos Symfony para mapear objetos a tablas de base de datos.
- **Composer**: Gestor de dependencias para PHP, utilizado para administrar las dependencias del proyecto Symfony.
- **Angular**: Framework de desarrollo de aplicaciones web desarrollado por Google, utilizado para crear aplicaciones de una sola página.
- **Angular CLI**: Herramienta de línea de comandos utilizada para iniciar, desarrollar y ejecutar proyectos en Angular.
- **Npm**: Gestor de paquetes para Node.js, utilizado para instalar y administrar paquetes de JavaScript.

8. 2 Problema

- **Alimento**: Cualquier sustancia consumida para proporcionar apoyo nutricional a un ser vivo.
- **Caloría**: Nombre de una unidad de energía que se usa para definir el valor nutricional de los alimentos.
- **Índice de Masa Corporal (IMC)**: Medida utilizada para evaluar el peso corporal en relación con la altura, que puede ayudar a determinar si una persona tiene un peso saludable, infrapeso, sobrepeso u obesidad.
- **Proteínas**: Son nutrientes esenciales para el cuerpo humano.
- **Grasas, Carbohidratos y Azúcares**: Son nutrientes que se consumen para dar energía al organismo humano.
- **Minerales**: Son compuestos esenciales para el funcionamiento adecuado del organismo humano.
- **Receta**: Conjunto de instrucciones para realizar sobre diferentes alimentos para realizar un plato o bebida.

9. Anexos

Como anexo se entregan los archivo:

- "Guía de Estilos WellnesTrack – Daniel Martin Hermoso Hermoso.pdf"
- "Manual de Usuario WellnessTrack – Daniel Martin Hermoso Hermoso.pdf"
- Código fuente
 - Wellnesstrack-Back
 - Wellnesstrack-Front
- Scripts
 - instalarRequisitos.sh
 - lanzarDockerProyect.sh
 - lanzarSistema.sh