

Aplicación para la planificación de dietas

Grupo A02



Hecho por

ÁLVARO DE FRANCISCO NIEVAS, 838819@unizar.es

ANDREI DUMBRAVA LUCA, 844417@unizar.es

GUILLERMO BAJO LABORDA, 842748@unizar.es



Universidad
Zaragoza



Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Resumen

Nuestro proyecto tiene como objetivo principal la generación de planes de menús personalizados para usuarios que deseen alcanzar objetivos relacionados con su salud y nutrición. El sistema permitirá a los usuarios definir objetivos (pérdida de peso, mantenimiento, aumento de masa muscular, recomposición) y preferencias dietéticas (intolerancias, alimentos que no les gustan) para crear planes de comidas adaptados a sus necesidades específicas. Además, se considerarán datos personales, como altura, peso, edad y nivel de actividad física, para una planificación más precisa.

La necesidad que buscamos resolver con este sistema es proporcionar a los usuarios sin mucho conocimiento sobre alimentación una herramienta efectiva para planificar sus dietas de manera personalizada de acuerdo con sus objetivos de salud y preferencias alimenticias. Esto simplificará la toma de decisiones nutricionales y ayudará a los usuarios a mantener un estilo de vida más saludable.

El público destinatario de este sistema incluye a personas interesadas en mejorar su salud y bienestar a través de la nutrición adecuada.

Los datos utilizados para la planificación de los menús se obtendrán de una API que proporciona información sobre el contenido nutricional de los alimentos, así como de los usuarios que ingresarán sus datos personales y preferencias. El sistema ofrecerá planes de menús personalizados que incluirán detalles sobre las comidas a consumir y sus contenidos nutricionales.

Índice

Resumen.....	2
1. Objetivo y alcance funcional final de la aplicación.....	4
1.1. Modelo de negocio y financiación.....	4
1.2. Funcionalidades del sistema.....	5
1.3. Perfil de usuario.....	6
1.4. Información en el sistema.....	6
1.5. Otras consideraciones.....	7
2. Sistema y storyboard de la aplicación.....	8
2.1. Requisitos funcionales del sistema.....	8
2.2. Diseño de la capa vista de la aplicación y prototipado.....	9
3. Modelo de datos del sistema.....	10
3.1. Modelo entidad-relación y relacional.....	10
3.2. Elección del SGBD.....	11
3.3. Capa de acceso a datos.....	12
4. Diferencias con la versión inicial.....	13
4.1. Funcionalidad.....	13
4.2. Modelo de datos.....	14
4.3. Interfaz.....	15
5. Despliegue e instalación.....	16
5.1. Descripción del sistema.....	16
5.2. Archivos de configuración.....	17
5.3. Inicialización del sistema.....	17
6. Relativo al uso de la aplicación.....	18
6.1. Cuentas y credenciales.....	18
6.2. Autenticación.....	18
6.3. Perfil de administración.....	18
6.4. Seguridad del sistema.....	19
6.5. Otros aspectos.....	19
7. Cronograma y valoración.....	20
7.1. Cronograma de desarrollo.....	20
7.2. Dificultades encontradas.....	20
7.3. Habilidades aprendidas.....	21
7.4. Aspectos a implementar en un futuro.....	21
Anexo I: Pantallas de la aplicación.....	22
Anexo II: Sentencias SQL para la BB.DD.....	30
Bibliografía.....	32

1. Objetivo y alcance funcional final de la aplicación.

1.1. Modelo de negocio y financiación

Obtención de Capital Inicial

Para comenzar con nuestra aplicación de dietas, es esencial contar con el capital necesario para desarrollar la aplicación. Hemos considerado diversas maneras para obtener este capital inicial:

- **Inversión personal:** En la etapa inicial, utilizaremos nuestros propios recursos personales y ahorros para financiar la fase de desarrollo. Esto nos otorga un mayor control sobre el proyecto y elimina la necesidad de incurrir en deudas o ceder participación en el negocio.
- **Préstamos:** También hemos pensado en obtener préstamos de entidades financieras. Para maximizar nuestras posibilidades de aprobación, hemos elaborado un plan de negocios detallado que demuestra la viabilidad y el potencial de rentabilidad de nuestra aplicación. Los fondos obtenidos a través de préstamos se destinarán a actividades iniciales.

Monetización del sistema

Para garantizar la sostenibilidad financiera a largo plazo y ofrecer una amplia gama de planes de dieta, hemos planeado la implementación de una versión de pago que nos permitirá monetizar la aplicación.

Dado que la funcionalidad principal y diferenciadora de nuestra aplicación es la creación de planes de comidas, en la versión gratuita limitaremos el número de planes que un usuario puede tener en un momento determinado a dos planes. Si el usuario quiere tener más planes, deberá pagar una suscripción mensual. Además, en un futuro añadiríamos funcionalidades adicionales al plan premium que aumenten el valor percibido del usuario, como un generador de listas de compra, tracking del peso del usuario o una mayor personalización en los menús.

En resumen, nuestro modelo de negocio se basa en una implementación de un modelo de suscripciones. A medida que nuestra base de usuarios crece, nuestro objetivo es ofrecer un valor adicional a través de una experiencia de usuario mejorada y una mayor personalización de los planes de dieta. Esto nos permitirá alcanzar nuestros objetivos de negocio a largo plazo y ofrecer un servicio de calidad a nuestros usuarios.

1.2. Funcionalidades del sistema

El sistema de planificación de menús personalizados ofrecerá una serie de funcionalidades para satisfacer las necesidades de los usuarios. Estas funcionalidades incluirán:

Usuarios

- **Registro:** Permite a los usuarios registrarse en el sistema proporcionando información básica como nombre, correo electrónico y contraseña.
- **Autenticación:** Los usuarios registrados pueden iniciar sesión en el sistema utilizando su correo electrónico y contraseña.
- **Información:** Los usuarios pueden completar su perfil proporcionando datos personales adicionales, como altura, peso, edad y nivel de actividad física.
- **Preferencias dietéticas:** Los usuarios pueden especificar sus preferencias dietéticas, como intolerancias alimentarias y alimentos que no les gusten.

Plan de comidas

- **Objetivos:** Cada plan de comida permite asociar un objetivo de salud, como perder peso, mantenerse o aumentar la masa muscular, o recomposición.
- **Generación:** Basándose en los datos personales, objetivos y preferencias dietéticas de los usuarios, el sistema generará planes de comidas personalizados que cumplan con los criterios establecidos.

Recetario

- **Listado:** permite explorar recetas que no están incluidas en los planes del usuario.
- **Búsqueda:** podremos buscar por nombre dentro de las recetas para encontrarlas fácilmente.

Consulta de Información Nutricional

- El sistema consultará una API de información nutricional de alimentos para obtener datos precisos sobre el contenido nutricional de los alimentos.

Visualización

- **Estadísticas alimenticias:** Los usuarios podrán ver estadísticas detalladas sobre su ingesta diaria de macronutrientes (proteínas, carbohidratos y grasas) para un seguimiento efectivo de su dieta.
- **Administración:** Acceso al panel de administración que detalle información acerca del uso del sistema.

1.3. Perfil de usuario

Además del administrador, hay dos tipos de usuarios que interactúan con el sistema. Ambos pueden registrarse, autenticarse e ingresar sus datos personales, y tienen acceso al recetario y la creación de planes en base a objetivos y el recetario. La diferencia entre ambos será:

- **Usuario básico:** tiene limitaciones en su uso. Por el momento, solo puede crear un máximo de dos planes.
- **Usuario prime:** dispone de todas las funcionalidades del sistema. Puede crear planes ilimitados y en un futuro, se incorporarán más funcionalidades que aumenten el valor percibido por parte del usuario.

Además, existe el perfil del **administrador del sistema**. Tiene acceso a funcionalidades pertinentes a este tipo de usuario. Podrá ver la lista de usuarios registrados en el sistema, los planes de comidas y recetas que hay en el sistema y podrá ver las interacciones con la base de datos en tiempo real. Además podrá modificar las distintas entradas de la base de datos de forma directa.

1.4. Información en el sistema

Capturaremos información a través de dos fuentes principales, formularios y bases de datos existentes.

Formularios

Los usuarios de nuestro sistema rellenarán un formulario en el que aportarán información acerca de su físico para personalizar las comidas. Los datos recopilados incluyen la altura, peso, edad o la cantidad de actividad física que realizan semanalmente.

APIs externas

Recopilaremos información acerca de comidas a partir de bases de datos ofrecidas por APIs ya existentes, como *spoonacular*. Estos sistemas nos permiten seleccionar comidas a partir de parámetros como ingredientes, aporte calórico máximo y mínimo, la cantidad de cada macronutriente que se encuentra en el alimento...

Además, este sistema dispone de un servicio de creación de planes que aunque es limitado, nos permitirá testear nuestro modelo de negocio en el mercado.

1.5. Otras consideraciones

Además de las funcionalidades y perfiles de usuario mencionados anteriormente, el diseño del sistema de planificación de menús personalizados debe considerar una serie de aspectos adicionales para garantizar su éxito y satisfacción del usuario. Estos aspectos incluyen:

1. **Seguridad de Datos:** La seguridad de los datos es esencial. Se deben implementar medidas de seguridad robustas para proteger la información personal y nutricional de los usuarios, como la encriptación de datos, autenticación segura y protección contra amenazas cibernéticas.
2. **Accesibilidad:** El sistema debe ser accesible para todos los usuarios, incluyendo aquellos con discapacidades visuales, auditivas o motoras. Se deben seguir estándares de accesibilidad web, como WCAG, para garantizar que el sistema sea utilizable por todos.
3. **Usabilidad:** la interfaz de usuario debe ser intuitiva y fácil de usar. Esto incluye un diseño limpio y una navegación clara que permita a los usuarios encontrar rápidamente la información que necesitan y utilizar las funcionalidades sin dificultades.
4. **Diseño responsivo:** el sistema debe ser compatible con dispositivos de diferentes tamaños y resoluciones, como computadoras de escritorio y dispositivos móviles. El diseño responsivo asegura una experiencia de usuario consistente en todas las plataformas.
5. **Tamaño de letra y contraste:** el tamaño de letra y el contraste deben ser adecuados para garantizar que el contenido sea legible para todos los usuarios. Se deben evitar combinaciones de colores que dificulten la lectura.

2. Sistema y storyboard de la aplicación

2.1. Requisitos funcionales del sistema

A continuación, se detallan los requisitos funcionales del sistema. Estos requisitos describen las funciones y características esenciales que la aplicación debe tener para permitir a los usuarios crear, gestionar y seguir planes de dieta personalizados.

CATÁLOGO DE REQUISITOS		
Código	Descripción	Prioridad
RF-1	El sistema permite al usuario el inicio de sesión o la creación de una nueva cuenta.	M
RF-2	El usuario puede introducir sus datos personales (sexo, edad, peso, estatura, intolerancias, preferencias), datos sobre sus hábitos (nivel de actividad semanal) y objetivos.	M
RF-3	El usuario puede generar automáticamente nuevas dietas adaptadas a sus necesidades específicas, en función de sus datos y de sus objetivos.	M
RF-4	El usuario puede ver todas las recetas de una dieta, que contienen nombre, descripción, macronutrientes, kilocalorías e ingredientes.	M
RF-5	El usuario puede ver una lista de todos los planes de dieta que ha creado o guardado en la sección “Mis planes”.	M
RF-6	El usuario puede visualizar sus dietas creadas.	M
RF-7	El usuario puede editar o eliminar un plan de dieta existente en “Mis planes”.	M
RF-8	El usuario puede sustituir un plato/alimento de una dieta y solicitar un cambio/sugerencia de un plato distinto.	C
RF-9	El usuario puede consultar y modificar su información personal en todo momento.	M
RF-10	El usuario puede seleccionar las unidades de medida deseadas para la aplicación (kg/lb, Kcal/KJ).	W
RF-11	El usuario tiene la posibilidad de darse de baja eliminando su cuenta temporal o permanentemente.	C
RF-12	El usuario puede cerrar sesión.	M
RF-13	El sistema facilita al usuario un formulario de contacto con el administrador del sistema.	S

RF-14	Cada plan tiene asociada una lista de la compra	S
RF-15	El usuario puede ver estadísticas acerca de el plan de comidas	S

2.2. Diseño de la capa vista de la aplicación y prototipado

El diseño de la capa vista de una aplicación es un aspecto esencial en el proceso de desarrollo de software. En este contexto, la capa vista se refiere a la interfaz de usuario, la cara visible de la aplicación con la que los usuarios interactúan. En el [Anexo I](#) se encuentran los diseños de las pantallas.

Cuando los usuarios entren en nuestra web se encontrarán con un [landing](#) que expondrá las distintas características de nuestra aplicación. En la misma página se encontrará un botón para acceder a la aplicación. Este enlace nos llevará a la [página de autenticación de Google](#). Si el inicio de sesión falla, nos redirigirá a una [pantalla en la que podremos volver a intentarlo](#).

Una vez hayamos iniciado sesión, se abren dos posibles caminos. En el caso de que no tengamos ningún plan de comidas asociado a nuestra cuenta, la aplicación realizará una [encuesta](#) que le permitirá crear un plan de comidas ajustado a las necesidades y preferencias del usuario. Cada pregunta de la encuesta tiene una pantalla distinta.

Si ya hemos realizado la encuesta, la aplicación nos llevará al [dashboard principal](#) de la aplicación.

En el caso de que no tengamos ningún plan creado, la aplicación nos llevará a la pantalla de [creación de planes](#). En caso contrario, se mostrará el plan más antiguo del usuario.

En el [dashboard](#) podremos seleccionar un plan de comidas en la barra lateral izquierda y se mostrará en la pantalla. El plan de comidas consiste en una o varias recetas asociadas a un día y momento concreto, como el desayuno, comida, cena o snacks. La forma del dashboard se asemeja a un horario, haciéndolo familiar al usuario. El usuario hará scroll vertical para ver todas las comidas. Además, podremos cambiar el nombre del plan en una [pantalla de edición](#).

Al hacer clic en una [receta](#) del plan, pasamos a una pantalla en la que vemos la descripción de la misma, así como la lista de ingredientes o la información asociada a la misma (calorías, hidratos de carbono, proteínas, grasas...).

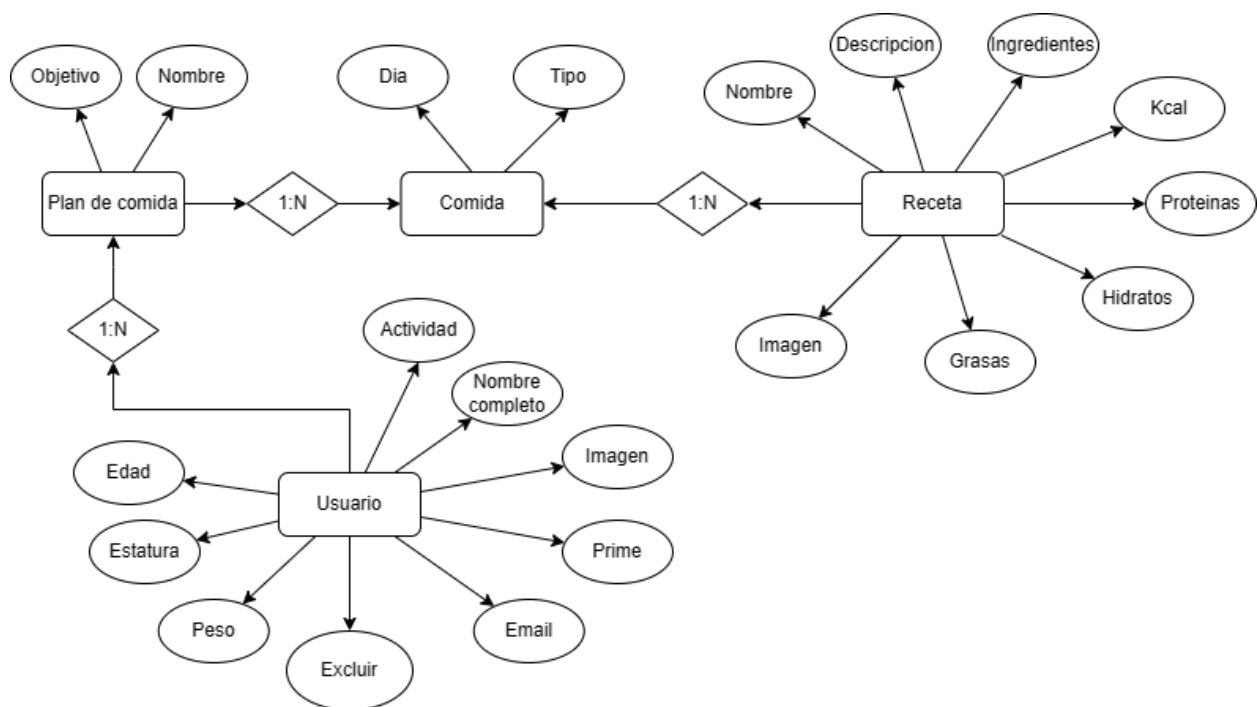
Además, en la barra lateral podemos encontrar el [enlace al recetario](#). Esta pantalla tiene una tabla paginada que contiene todas las recetas del sistema. Además, nos muestra la información nutricional de la misma, como las calorías o los hidratos de carbono. Podemos buscar, explorar la lista y hacer click en una [receta para abrir la pantalla](#) mencionada anteriormente.

Por último, desde la barra lateral podremos acceder también al [perfil del usuario](#) en el que podrá cambiar sus datos personales y el botón de cierre de sesión.

3. Modelo de datos del sistema

3.1. Modelo entidad-relación y relacional

Primero pensamos qué entidades forman nuestro modelo. En nuestra primera aproximación, llegamos a la conclusión de que eran principalmente 3: **usuario**, **plan de comidas** y **receta**. De esta forma podríamos almacenar toda la información asociada a las recetas y relacionarlas con el plan de comidas al que pertenecen. No obstante, nos dimos cuenta de que también necesitamos información acerca de cuándo se va a comer esa receta en el plan: día de la semana y momento del día. Por tanto, establecimos una cuarta entidad, **comida**. El **plan de comidas** será una colección de comidas junto con otros datos propios del mismo, como su nombre, propietario, objetivo y otros. Cada **comida** almacenará referencias a las recetas que se van a hacer en un día y momento concreto. La entidad **receta** guarda la información del plato, como una pequeña descripción, lista de ingredientes, los tipos que coinciden con esta, una imagen y atributos como las calorías, hidratos de carbono, grasas y proteínas.



Una vez planteado el esquema entidad-relación, lo pasamos a modelo relacional:



3.2. Elección del SGBD

Para el SGBD hemos elegido PostgreSQL por diversos motivos, entre ellos:

1. Por su semejanza con Oracle, el SGBD que ya utilizamos todos los miembros del equipo en la asignatura “Bases de datos” cursada el año anterior.
2. Es de código abierto, lo que nos da mucha flexibilidad al poder desplegar en cualquier entorno (VM, docker, heroku...).
3. Otra razón para elegir PostgreSQL es su amplia comunidad de usuarios y desarrolladores. Esto nos brinda acceso a una gran cantidad de recursos, documentación y ejemplos que nos serán de gran utilidad a lo largo del desarrollo.

En definitiva, lo que buscamos es que la elección de un SGBD u otro no ralentice el desarrollo de la aplicación, ya sea por tener que aprender una nueva tecnología, por tener que lidiar con problemas de integración o por falta de recursos acerca del gestor.

3.3. Capa de acceso a datos

En nuestro sistema, hemos implementado una serie de DAOs (Data Access Objects) que nos permiten abstraer eficazmente la capa de datos y proporcionar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) para diferentes entidades, incluyendo Usuario, Plan, Receta y Comida.

Para la entidad "**Usuario**", hemos desarrollado métodos que permiten crear nuevos usuarios, recuperar información de usuarios existentes y eliminar cuentas de usuario cuando sea necesario. Además, podremos actualizar perfiles de usuario, aunque únicamente para los atributos que consideramos que pueden cambiar (no se podrá cambiar el nombre, por ejemplo). Hemos añadido operaciones pertinentes al usuario como una función para contar el número de planes de comidas que tiene el usuario.

En el caso de la entidad "**Plan**", nuestros DAOs ofrecen la capacidad de crear planes personalizados, acceder a los detalles de los planes existentes, actualizar información sobre los planes y eliminar planes específicos. Para garantizar que solo el usuario puede modificar sus planes todas las operaciones deberán tener, no solo el identificador del plan, sino también el del usuario.

Para la entidad "**Receta**" hemos diseñado operaciones para obtener una lista de recetas filtradas por título ordenadas alfabéticamente y para calcular el número de páginas en base al número de recetas que mostraremos por página en el recetario. Además de estas operaciones específicas, hemos añadido operaciones tipo *getOne* y *getAll*. No consideramos la opción de eliminar recetas ya que queremos que sean persistentes en el sistema aunque se eliminen todos los planes que las contienen.

Esta abstracción de datos a través de DAOs garantiza una gestión eficiente y consistente de la información en nuestro sistema, facilitando el acceso y la manipulación de datos en todas las áreas de la aplicación.

4. Diferencias con la versión inicial

La versión final de la aplicación presenta diferencias en distintos aspectos en comparación con la versión presentada en la memoria inicial del proyecto. Estas divergencias se pueden observar en términos de funcionalidad, modelo de datos e interfaz, lo que ha resultado en mejoras significativas y algunas áreas pendientes de implementación

4.1. Funcionalidad

Al empezar el desarrollo de la aplicación tuvimos que decidir entre dos aproximaciones a la hora de generar los planes de comidas, parte esencial de nuestro sistema, crear un algoritmo propio o usar un servicio existente.

La ventaja principal del algoritmo propio es la mayor personalización de los menús. Nosotros no solo queríamos especificar no sólo un **número de calorías**, sino también los **macronutrientes diarios** (proteínas, grasas, carbohidratos, entre otros), el **número de comidas** por día, **número de platos** por comida y otros aspectos que pudieran surgir. Además, ahorraríamos costes reduciendo el número de peticiones a la API y mejoraríamos la escalabilidad futura de nuestro sistema.

Por contra, no disponemos de datos de calidad, para lo que necesitaríamos una mayor inversión inicial que nos permita hacer el número de peticiones que necesitaríamos. No obstante, los plazos y la complejidad fueron los principales factores que se tuvieron en cuenta.

Teniendo en cuenta los distintos factores, nos hemos decidido por utilizar la API externa. En esta parte del proyecto estamos desarrollando una prueba de concepto destinada a captar la atención de usuarios reales para ver cómo responde la comunidad a nuestra aplicación. Lanzar un producto mínimo viable nos permite ser ágiles en el desarrollo y construir las funcionalidades que los usuarios necesitan, no las que nosotros imaginamos que necesitan.

Aunque no fueron marcadas como requisitos fundamentales en la tabla de requisitos, inicialmente, se planeó la inclusión de una **lista de la compra**, así como **estadísticas generales** del plan de comidas, ofreciendo una visión detallada de las calorías, proteínas y otros nutrientes consumidos por semana o día.

No obstante se han implementado funcionalidades que añaden mucho valor a la aplicación pero que no se plantearon en un principio. Se ha **incluido un recetario completo** que permite a los usuarios acceder a todas las recetas del sistema y buscar recetas específicas por título. Este cambio añade una capa adicional de accesibilidad y facilita la experiencia del usuario al buscar opciones específicas de comidas.

Además, se mejoró la funcionalidad de **inicio de sesión mediante Google**. Aunque en la primera versión esta característica presentaba algunas limitaciones, en la versión final se ha optimizado y brinda una experiencia más fluida y segura para los usuarios.

4.2. Modelo de datos

Durante el proceso de evolución desde la versión inicial de la memoria hasta la versión final de la aplicación, se han introducido diversas modificaciones en el modelo de datos que han impactado en la arquitectura y funcionalidad del sistema.

En primer lugar, se ha optado por **eliminar la opción administrativa** directamente asociada al usuario. En lugar de ello, las funcionalidades administrativas se han separado de la aplicación principal. Esta decisión fue tomada para mejorar la seguridad y modularidad del sistema, asegurando un mejor control sobre las operaciones administrativas y evitando posibles riesgos de seguridad asociados con la integración directa de estas funcionalidades en el usuario común.

Además, se ha incorporado un nuevo **parámetro denominado "survey"** para rastrear si el usuario ha completado la encuesta inicial. Esta adición enriquece el conjunto de datos y proporciona información valiosa para personalizar la experiencia del usuario, permitiendo adaptar recomendaciones y ajustes basados en la información obtenida de la encuesta.

Adicionalmente, se ha realizado un cambio significativo en el modelo de datos al reemplazar el parámetro **"resumen" por el parámetro "instrucciones"**. Esta actualización consiste en transformar el enfoque del sistema, ahora proporcionando una sección específica denominada "instrucciones" que detalla los pasos necesarios para llevar a cabo una receta en lugar del resumen previamente contemplado.

Otra modificación significativa es la **gestión de contraseñas** de usuario. En la versión final, el sistema no guarda la contraseña directamente; en su lugar, esta responsabilidad recae en un servicio externo. Esta medida se ha adoptado para reforzar la seguridad del sistema y proteger la información confidencial de los usuarios.

Asimismo, se ha incluido la capacidad de almacenar la **foto de perfil** del usuario, proporcionada por Google durante el inicio de sesión. Esta mejora en la experiencia del usuario añade una capa de personalización y comodidad al permitir que los usuarios utilicen su foto de perfil de Google dentro de la aplicación.

Además, se han introducido **tipos específicos** para "actividad" y "objetivo" que no se habían considerado en la iteración inicial del modelo de datos. Estos tipos proporcionan una mayor precisión y especificidad en la representación de la información, mejorando así la calidad de los datos manejados por el sistema.

Por último, se han realizado ajustes en **valores por defecto** y **cambios en algunos identificadores** (IDs), lo que ha contribuido a una mejor coherencia y organización en la gestión de datos dentro del sistema.

4.3. Interfaz

Durante el proceso de transición desde la versión inicial de la memoria hasta la versión final de la aplicación, se han introducido cambios significativos en la interfaz, lo cual ha impactado notablemente en la experiencia del usuario al interactuar con el sistema.

En primer lugar, se ha realizado una modificación en la **presentación de los menús**. Se ha cambiado el **enfoque de desplazamiento** horizontal a la disposición de los días en horizontal en lugar de en vertical. Esta alteración permite aprovechar mejor el espacio en pantalla y facilita la navegación vertical, una forma más común y natural en la experiencia web. Este ajuste tiene como objetivo mejorar la usabilidad y la comodidad del usuario al acceder y desplazarse por los menús.

Además, se ha optado por **eliminar el porcentaje de la encuesta**, considerando que su inclusión añadía complejidad innecesaria a la interfaz. Esta simplificación ha sido implementada para optimizar la visualización de la encuesta y facilitar su comprensión por parte de los usuarios, eliminando elementos que podrían generar confusión o distraer del propósito principal de la encuesta.

Otra modificación importante ha sido la **fusión de las pantallas de perfil y ajustes**. Esta decisión ha sido tomada con el objetivo de centralizar la gestión del perfil del usuario y sus ajustes en una sola interfaz, simplificando la navegación y proporcionando una experiencia más fluida y cohesiva.

Adicionalmente, se ha llevado a cabo un cambio en el **esquema de colores** en general dentro de la interfaz. Esta actualización en la paleta de colores busca mejorar la estética visual y la coherencia del diseño en toda la aplicación, ofreciendo una experiencia más armoniosa y atractiva para los usuarios.

5. Despliegue e instalación

El despliegue de la aplicación es muy sencillo. Utilizaremos Docker y Docker Compose para construir las imágenes y lanzar los servicios. Docker es una herramienta que permite la creación, implementación y ejecución de aplicaciones mediante la contenerización, asegurando que funcionen de manera consistente en diferentes entornos. Docker Compose, por su parte, facilita la gestión de aplicaciones multi-contenedor, definiendo la configuración de los servicios en un archivo YAML para su ejecución conjunta.

5.1. Descripción del sistema

Este sistema integra Next.js tanto para el frontend como para el backend, lo cual permite un desarrollo más cohesionado y eficiente al utilizar un único framework para ambas áreas. Otro punto clave en la elección de este framework es la disponibilidad de mucha información en internet, y los componentes que lo rodean.

La aplicación se compone de varios componentes y funcionalidades clave:

- **Dashboard:** Componente principal que incluye la BarraLateral para la navegación y funcionalidades como MisPlanes, Perfil y CerrarSesión.
- **Plan:** Botones y otros mecanismos de interacción para la gestión de planes de comidas, con opciones para definir objetivos, editar y eliminar planes existentes.
- **Perfil:** Sección para la gestión de información personal y ajustes de cuenta.
- **Receta:** Componente para visualizar detalles de recetas, incluyendo título, foto, descripción, ingredientes y nutrientes.
- **Encuesta:** Un componente React que maneja el proceso de encuesta para la generación de planes personalizados.
- **InicioSesion y CrearCuenta:** Componentes para la autenticación de usuarios, con formularios y opciones correspondientes.

Cada uno de estos componentes contribuye al flujo general de la aplicación, permitiendo a los usuarios crear planes de comidas adaptados a sus necesidades físicas.

Utilizamos PostgreSQL como sistema de gestión de bases de datos y Nginx se despliega como servidor web, actuando como un proxy reverso que dirige el tráfico y facilita la entrega eficiente del contenido generado por Next.js hacia los usuarios finales. Para gestionar y administrar la base de datos PostgreSQL, se utiliza pgAdmin, una herramienta que proporciona una interfaz amigable y completa para la gestión de bases de datos.

Dentro del sistema, se definen dos redes: 'frontend' y 'backend'. La red 'frontend' se usa para conectar el servicio de Next.js y el servidor Nginx, exponiendo el frontend al puerto 3000 y redirigiendo el tráfico al

puerto 80. La red 'backend' conecta el servicio de la base de datos PostgreSQL y pgAdmin, asegurando la comunicación interna entre estos servicios.

5.2. Archivos de configuración

El archivo *package.json* define las dependencias específicas de Next.js que son esenciales para la aplicación. Estas dependencias se utilizarán durante el proceso de construcción de la imagen para asegurar que el entorno de ejecución dentro del contenedor sea coherente con los requisitos del proyecto.

En cuanto a la gestión de las variables de entorno, se utiliza un archivo denominado *.env*, ubicado en la raíz del proyecto. En el archivo *docker-compose*, se hace referencia a este fichero para cargar las variables de entorno en Next.js, permitiendo una configuración más sencilla y flexible del entorno de la aplicación.

El archivo *nginx.conf* contiene la configuración del servidor Nginx. Define aspectos como la asignación de puertos, las rutas de acceso a los recursos estáticos, la configuración de los proxys, y otras directivas relevantes para la correcta operación del servidor web en el contexto de la aplicación.

El script de inicialización de la base de datos, denominado *init.sql*, despliega los comandos necesarios para crear y configurar la base de datos al inicio del contenedor de PostgreSQL. Este archivo se ubica en una carpeta específica designada para almacenar scripts de inicialización, la cual está vinculada al contenedor de la base de datos. Al iniciarse, PostgreSQL ejecutará automáticamente este script contenido en la carpeta designada, asegurando la correcta configuración inicial de la base de datos y posiblemente creando tablas, definiciones de esquemas u otros ajustes necesarios para su funcionamiento. Este enfoque permite una gestión centralizada de los scripts de inicialización, facilitando su ejecución y mantenimiento dentro del entorno de contenedores Docker.

5.3. Inicialización del sistema

Para lanzar el sistema, el usuario deberá ejecutar el script **lauch-app.sh** en la raíz del proyecto, asegurándose de tener Docker y Docker-compose instalado. Este script baja el servicio si estaba iniciado y construye de nuevo las imágenes. Una vez se ha completado el proceso, ejecuta el sistema.

Una vez los contenedores estén en ejecución, el usuario podrá acceder a la aplicación a través de la URL **http://localhost** para interactuar con el sistema. Además, para administrar la base de datos, podrá acceder a pgAdmin mediante la URL **http://localhost:8888** utilizando las credenciales establecidas en el archivo de configuración.

6. Relativo al uso de la aplicación

6.1. Cuentas y credenciales

- **Cuentas de usuario:** En el script de inicialización se establece la cuenta smartbitesisinf@gmail.com como cuenta no prime para mostrar las limitaciones. Todas las otras cuentas con las que se inicie sesión serán prime.
- **Base de datos y pgAdmin:**
 - **Email:** `smartbitesisinf@gmail.com`
 - **Contraseña:** `i6k£ZJ5931`
 - **Contraseña de la base de datos:** `smartbites`
- **Otros:** en la aplicación se utilizan credenciales para las APIs que utilizamos, Google Auth y Spoonacular. Además, tenemos un secreto de nuestra aplicación para firmar las sesiones. Todos ellos se encuentran en el fichero `.env`.

6.2. Autenticación

El sistema de autenticación de la aplicación se realiza mediante Google Auth, asegurando un inicio de sesión seguro. El flujo de inicio de sesión comienza redirigiendo al usuario a la página de autenticación de Google, donde se le solicita iniciar sesión con su cuenta de Google. Una vez autenticado, Google redirige de vuelta a la aplicación con la información proporcionada por Google, como el nombre y correo electrónico del usuario. En caso de que sea la primera vez que el usuario inicia sesión, se crea una cuenta interna en la aplicación utilizando la información suministrada por Google y se asigna un ID interno único a esa cuenta.

La aplicación cuenta con URLs libres y protegidas. Las URLs libres de autenticación se utilizan en la promoción del producto que ofrecemos, permitiendo el acceso público a nuestro landing page. Este landing proporciona información detallada sobre las características, precios y beneficios de nuestro servicio, facilitando a los usuarios conocer más acerca de nuestra aplicación sin requerir un inicio de sesión previo.

Las URL protegidas requieren que el usuario tenga una sesión activa. Al acceder a estas rutas protegidas, se verifica la sesión del usuario y se asocia el ID del usuario a dicha sesión para autorizar el acceso.

6.3. Perfil de administración

La gestión de la información de la aplicación se realiza a través de pgAdmin, donde se puede acceder mediante una cuenta de administrador con credenciales predefinidas. Este "cuadro de mando" permite la

creación, eliminación y modificación directa de usuarios así como la información que generan, ofreciendo un control completo sobre la gestión de usuarios del sistema.

6.4. Seguridad del sistema

La seguridad es un aspecto fundamental de nuestro sistema de información. Tratamos datos personales de los usuarios, tales como su peso, edad o preferencias alimenticias y debemos tener especial cuidado.

Para acceder a las funcionalidades de la aplicación será necesaria una cuenta de usuario. Para crearla se utilizará el correo electrónico y una contraseña. Una vez autenticado, el usuario dispondrá de un JsonWebToken sellado por el servicio, con el que se identificará. El usuario utilizará su identificador (string generada utilizando el estándar UUID v4) para realizar operaciones en la aplicación una vez se haya autenticado.

Además, se cumplen protocolos de seguridad básicos como contraseñas hash y uso de herramientas que previenen las inserciones SQL. En este último aspecto hemos tenido especial cuidado, utilizando librerías como “zod” que nos permiten asegurar el formato en el servidor de los parámetros que insertamos en la base de datos. La verificación de los datos en el cliente no es garantía de seguridad.

6.5. Otros aspectos

El idioma en el que está disponible la aplicación es inglés. Aunque queríamos implementarla en español, los datos que obtenemos de la API externa están en inglés. Para mantener la coherencia en nuestra aplicación, decidimos utilizar el inglés. En un futuro, se podrían traducir los contenidos al español o cualquier otro idioma.

7. Cronograma y valoración

7.1. Cronograma de desarrollo

El desarrollo del sistema comenzó con el desarrollo de la interacción con la base de datos y la integración con la API externa con la que obtenemos información nutricional. En simultáneo, se abordó el desarrollo de la autenticación, una tarea desafiante que demandó una estrategia bien definida y su posterior implementación.

Más adelante, comenzamos a desarrollar la interfaz. Se construyeron formularios para la creación, edición y eliminación de planes de comidas y la encuesta inicial en el que el usuario rellena sus datos físicos. Además, se desarrollaron páginas específicas para la visualización de recetas y del perfil de usuario, incluyendo formularios para modificar datos personales como el peso, la actividad física y otros.

Más adelante, y para garantizar un flujo de aplicación coherente, se establecieron redirecciones contextuales. Algunos ejemplos son la redirección a la encuesta inicial si el usuario aún no la ha completado, o dirigir al usuario a la creación de un plan de comidas si no dispone de ninguno.

En mitad del desarrollo decidimos abandonar el enfoque inicial en la autenticación mediante correo y contraseña propia debido a limitaciones técnicas. Se adoptó Google Auth como alternativa dada la alta disponibilidad de cuentas de Google entre los usuarios de Internet. Se considera la posibilidad de añadir más servicios de autenticación en el futuro para ofrecer a los usuarios diferentes opciones de inicio de sesión. Esto implicó un cambio de estrategia y una refactorización completa del sistema para asegurar un inicio y cierre de sesión sin problemas.

Una vez el sistema básico funcionaba correctamente, implementamos el recetario. Este incluyó la búsqueda y paginación de recetas, además de otras funcionalidades. Al terminar esta funcionalidad, comenzamos con el despliegue de la aplicación. La adopción de Docker y Docker Compose, junto con la implementación de Nginx y PgAdmin, permitió una gestión eficiente de los contenedores y facilitó el despliegue del sistema.

7.2. Dificultades encontradas

Las dificultades encontradas durante el desarrollo incluyeron el aprendizaje simultáneo de desarrollo web, JavaScript y Next.js, con una curva de aprendizaje desafiante, y la dificultad para discernir entre componentes de servidor y cliente. La limitación de herramientas y la dificultad para diagnosticar problemas añadieron complejidad al proceso.

El diseño de la interfaz usando CSS resultó ser una tarea compleja y a veces confusa. La necesidad de esperar a que partes específicas del desarrollo estuvieran completas antes de avanzar nos dificultó la paralelización del trabajo.

7.3. Habilidades aprendidas

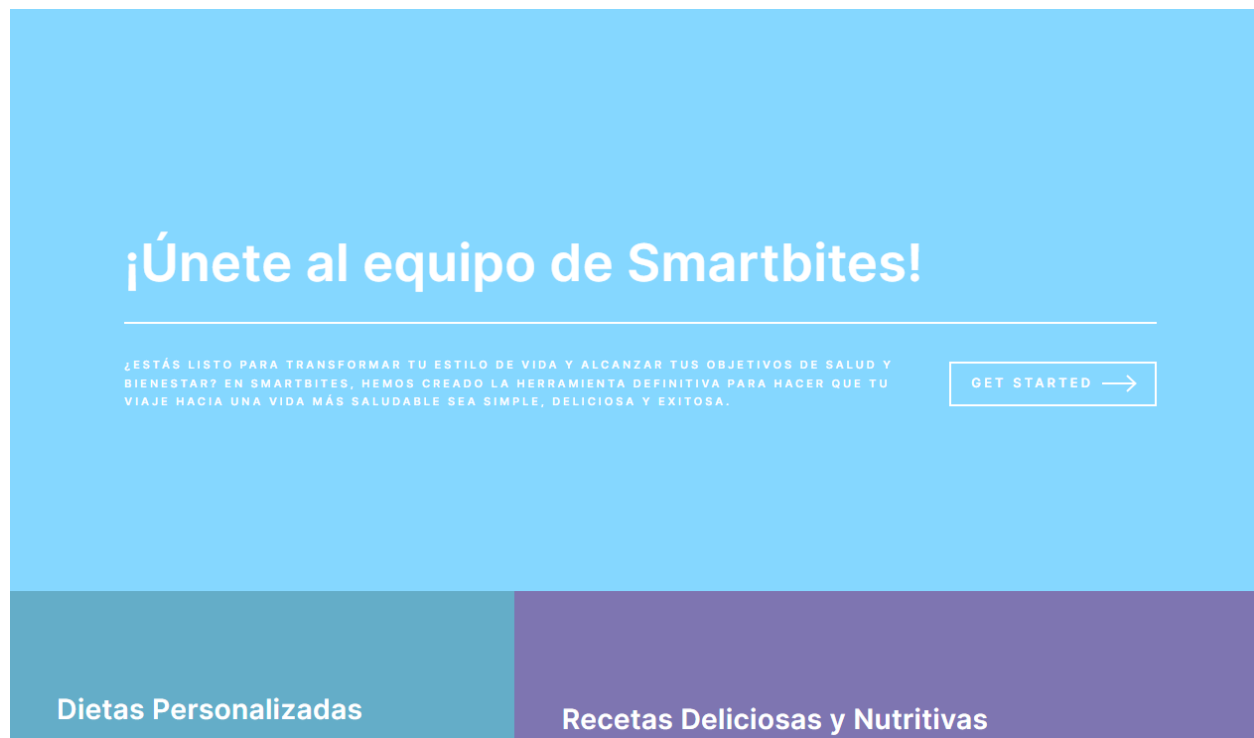
Durante el proceso de desarrollo, el equipo experimentó un notable crecimiento en habilidades técnicas, abarcando desde el desarrollo web y TypeScript hasta la interacción con bases de datos mediante herramientas ORM como Prisma, el uso de APIs externas, y el dominio de Docker y Docker Compose, entre otros conocimientos fundamentales en el desarrollo de aplicaciones modernas.

Es importante destacar que, al inicio del proyecto, solo un miembro del equipo tenía experiencia previa en tecnologías web. Los otros integrantes, si bien tenían experiencia en programación, no estaban familiarizados con tecnologías específicas del ámbito web como HTML, CSS, React y Next.js. Esto nos planteó un desafío considerable, ya que tuvimos que aprender estas tecnologías desde cero para contribuir de manera efectiva al proyecto.

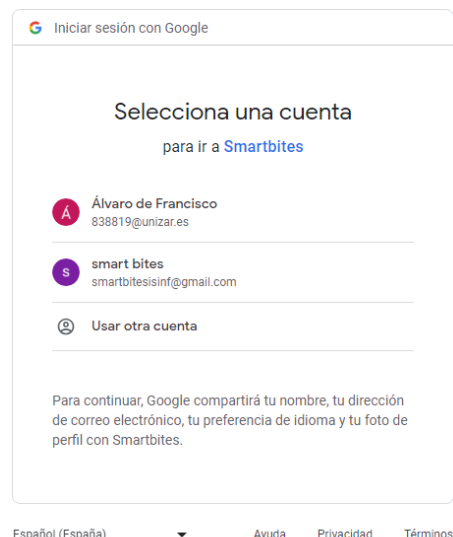
7.4. Aspectos a implementar en un futuro

A pesar de los logros, hay aspectos que no pudieron ser implementados, como una mayor personalización en los menús de comidas, herramientas para el seguimiento del peso del usuario, visualización de estadísticas sobre las macros de las comidas y filtrado por tipo de comida en el recetario. Además, se deseaba implementar un sistema de inicio de sesión utilizando contraseña propia y una pasarela para que los usuarios pasaran a un estado "prime".

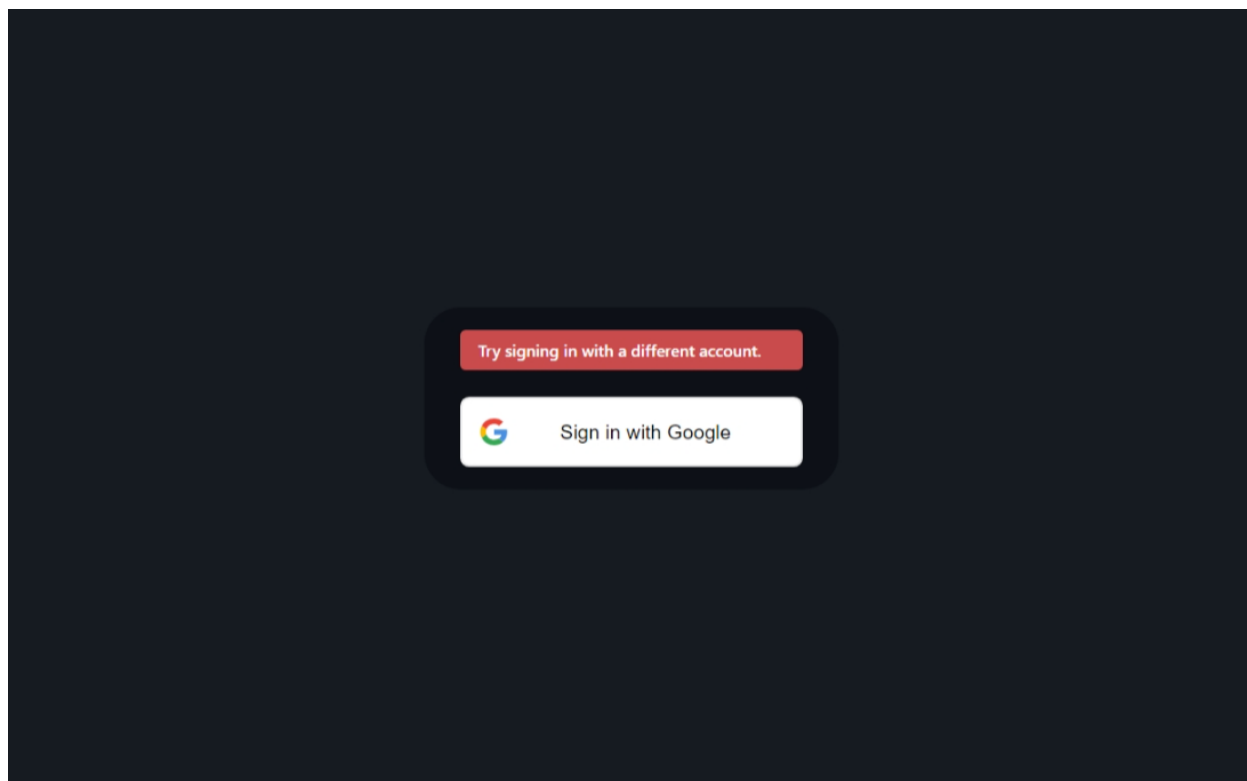
Anexo I: Pantallas de la aplicación



(1) Landing de la web



(2) Inicio de sesión con Google



(3) Fallo inicio de sesión

smartbites

1. Marque el sexo al que pertenece *

☐ Hombre

☐ Mujer

[Siguiente](#)

(4) Encuesta: selección de sexo

2. Escriba su edad: *

[Anterior](#) [Siguiente](#)

(5) Encuesta: selección de edad

3. Escriba su estatura en cm: *

[Anterior](#) [Siguiente](#)

(6) Encuesta: selección de estatura

4. Escriba su peso en kg: *

Anterior

Siguiente

(7) Encuesta: selección de peso

5. ¿Cuál es su actividad física diaria? *

Anterior

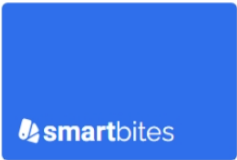
Siguiente

(8) Encuesta: selección de nivel de actividad

6. ¿Tiene alguna intolerancia o preferencia?

Anterior
Confirmar

(9) Encuesta: selección de intolerancias/preferencias



My meal plans

Plan 1

Recipe book

My profile

Sign Out

Dashboard

Create Plan +

Plan 1

Breakfast Lunch Dinner

Monday

Lemon Zucchini Muffins Crispy-Crowned Guacamole Fish Fillets Rigatoni With Sweet Sausages In Creamy Tomato Sauce

Tuesday

Chicken Porridge Panera Spicy Thai Salad No Fuss Sunday Slow-Cooker Balsamic Pot Roast

Wednesday

Granola Creamy Zucchini and Ham Pasta Crispy Chicken Burger

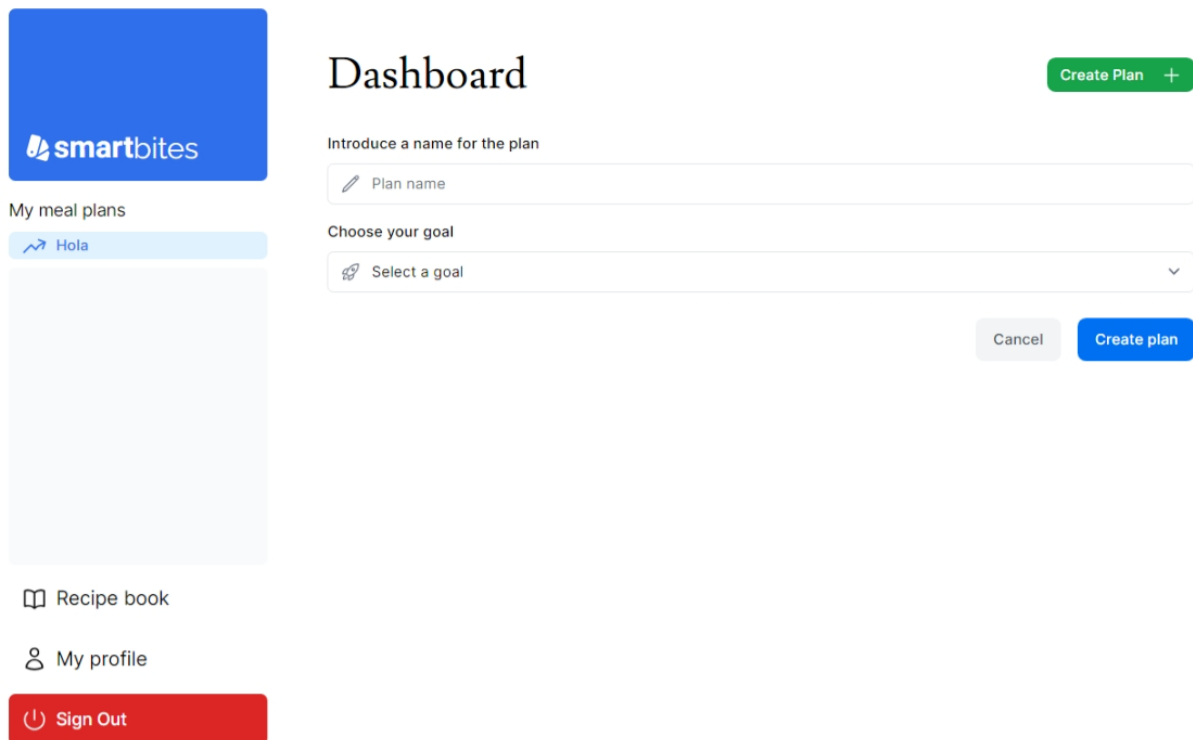
Thursday

Healthy Black Forest Baked Oatmeal Panera Spicy Thai Salad Bacon-Wrapped Meatloaf

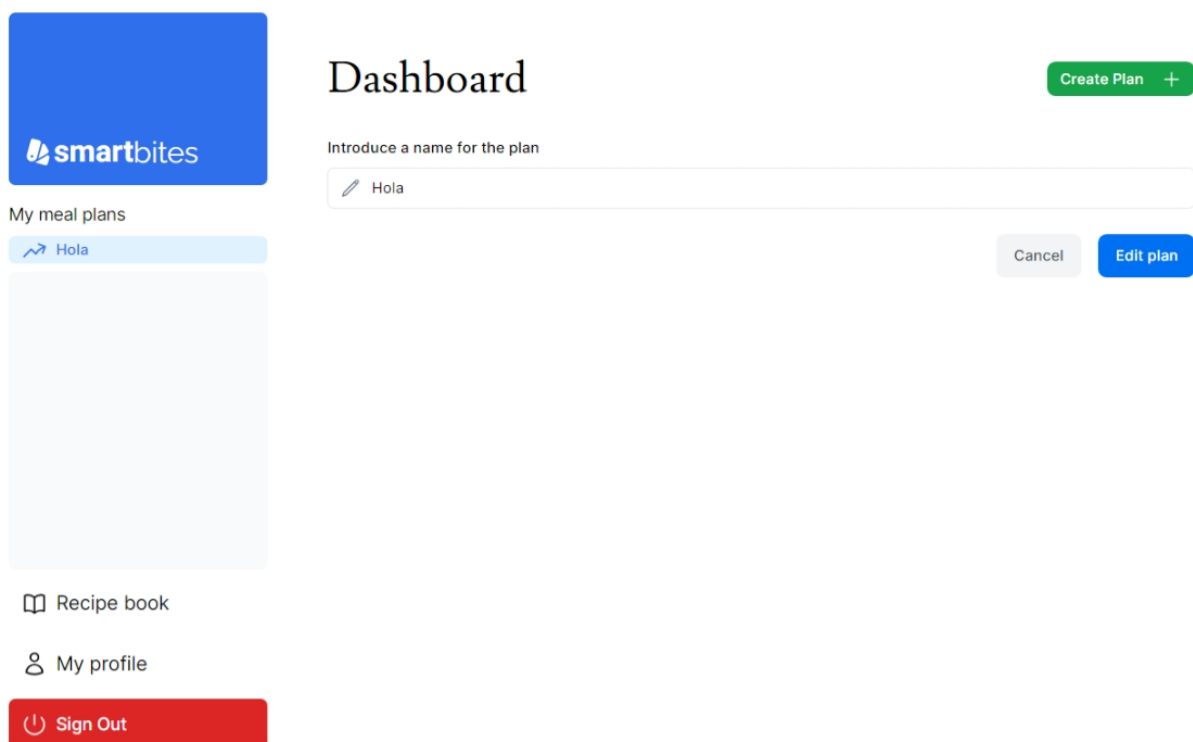
Friday

Coconut Bliss Smoothie Tuna Spaghetti With Fava Beans Tortilla Burger Loco Vaca

(10) Dashboard principal



(11) Creación de un plan



(12) Edición de un plan

My meal plans

Plan 1

Recipe book

My profile

Sign Out

Recipe

Recipe	Calories	Protein	Fat	Carbohydrates
Asparagus Eggs Benedict	779	34	57	30
Bacon-Apple-Pecan Stuffed French Toast	666	16	51	36
Bacon-Wrapped Meatloaf	701	35	55	14
Bacon Wrapped Tofu Tacos	497	27	34	17
Banana Smoothie Boost	42	N/A	N/A	8
BBQ Mac and Cheese	1236	51	27	194
Best Breakfast Burrito	960	31	49	98
Bibimbab (Korean Rice w Vegetables & Beef)	1090	48	32	149
Blueberry, Chocolate & Cocoa Superfood Pancakes - Gluten-Free/Paleo/Vegan	774	13	42	90
Blueberry Cinnamon Porridge	779	4	74	36

←

1

2

3

...

7

8

→

(13) *Recetario*

My meal plans

Plan 1

Recipe book

My profile

Sign Out

Recipe



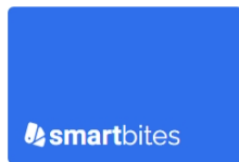
Calories	1061
Fat	57
Carbohydrates	75
Protein	78

Crispy-Crowned Guacamole Fish Fillets

Crispy-Crowned Guacamole Fish Fillets might be just the main course you are searching for. Watching your figure? This gluten free, dairy free, and pescatarian recipe has 1011 calories, 76g of protein, and 57g of fat per serving. This recipe serves 2. For \$9.9 per serving, this recipe covers 67% of your daily requirements of vitamins and minerals.

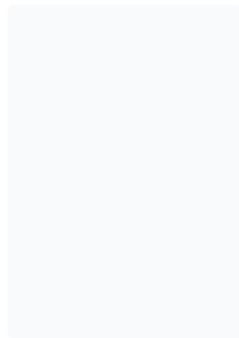
- 3 ripe avocados
- 2 bell peppers
- 2 bell peppers
- 6 cherry tomatoes
- 2 sprigs cilantro
- 650 g tilapia fish fillets (about
- 2 cloves garlic
- 0.5 onion
- 2 servings pepper
- 2 servings pepper
- 2 servings salt
- 2 servings tabasco sauce
- 60 g tortilla chips

(14) *Receta*



My meal plans

Plan 1



Recipe book

My profile

Sign Out

Mi perfil



Álvaro de Francisco

838819@unizar.es

Age: 18

Select your activity level at the moment

HIGH

Ingredients you want to exclude from your diet

Introduce the ingredients

Weight

70

kg

Height

181

cm

Save changes

(15) *Mi perfil*

Anexo II: Sentencias SQL para la BB.DD.

```
-- CreateEnum
CREATE TYPE "Activity" AS ENUM ('LIGHT', 'MODERATE', 'HIGH', 'EXTREME');

-- CreateEnum
CREATE TYPE "Goal" AS ENUM ('LOSE', 'MANTAIN', 'GAIN');

-- CreateTable
CREATE TABLE "User" (
    "id" TEXT NOT NULL,
    "email" TEXT NOT NULL,
    "fullName" TEXT NOT NULL,
    "image" TEXT,
    "prime" BOOLEAN NOT NULL DEFAULT true,
    "survey" BOOLEAN NOT NULL DEFAULT false,
    "weight" DOUBLE PRECISION,
    "height" DOUBLE PRECISION,
    "birthDate" TIMESTAMP(3),
    "activity" "Activity",
    "exclude" TEXT,
    "sex" TEXT,

    CONSTRAINT "User_pkey" PRIMARY KEY ("id")
);

-- CreateTable
CREATE TABLE "MealPlan" (
    "id" SERIAL NOT NULL,
    "name" TEXT NOT NULL,
    "goal" "Goal" NOT NULL,
    "userId" TEXT NOT NULL,

    CONSTRAINT "MealPlan_pkey" PRIMARY KEY ("id")
);

-- CreateTable
CREATE TABLE "Meal" (
    "id" SERIAL NOT NULL,
    "day" INTEGER NOT NULL,
    "type" INTEGER NOT NULL,
    "recipeId" INTEGER NOT NULL,
    "mealPlanId" INTEGER NOT NULL,
```

```

        CONSTRAINT "Meal_pkey" PRIMARY KEY ("id")
    );
-- CreateTable
CREATE TABLE "Recipe" (
    "id" SERIAL NOT NULL,
    "title" TEXT NOT NULL,
    "image" TEXT,
    "summary" TEXT NOT NULL,
    "instructions" TEXT NOT NULL,
    "calories" INTEGER,
    "fat" INTEGER,
    "carbohydrates" INTEGER,
    "protein" INTEGER,
    "ingredients" TEXT[],

    CONSTRAINT "Recipe_pkey" PRIMARY KEY ("id")
);
-- CreateIndex
CREATE UNIQUE INDEX "User_email_key" ON "User"("email");
-- CreateIndex
CREATE UNIQUE INDEX "Recipe_title_key" ON "Recipe"("title");
-- AddForeignKey
ALTER TABLE "MealPlan" ADD CONSTRAINT "MealPlan_userId_fkey" FOREIGN KEY ("userId")
REFERENCES "User"("id") ON DELETE CASCADE ON UPDATE CASCADE;
-- AddForeignKey
ALTER TABLE "Meal" ADD CONSTRAINT "Meal_recipeId_fkey" FOREIGN KEY ("recipeId")
REFERENCES "Recipe"("id") ON DELETE RESTRICT ON UPDATE CASCADE;
-- AddForeignKey
ALTER TABLE "Meal" ADD CONSTRAINT "Meal_mealPlanId_fkey" FOREIGN KEY ("mealPlanId")
REFERENCES "MealPlan"("id") ON DELETE CASCADE ON UPDATE CASCADE;

```

Bibliografía

<https://yazio.com/es> : página de ejemplo de la competencia

<https://spoonacular.com/food-api> : página a partir de la cual recibimos información acerca de la API

<https://www.postgresql.org/>: documentación del SGBD

<https://www.prisma.io/>: documentación del ORM que utilizaremos

<https://nextjs.org/>: documentación del framework utilizado

<https://www.pgadmin.org/>: documentación de la interfaz del cliente para el SGBD.

<https://es.react.dev/>: documentación de la librería de componentes que utilizaremos

<https://next-auth.js.org/>: librería de autenticación que utilizamos en la aplicación