

# **FITNESS TRACKER**

## **MEMORIA**

Realizado por Aaron Esono, Guillermo Quintanar y Hugo Pelayo  
26 de abril de 2024

# Índice

<b>Introducción.....</b>	<b>4</b>
<b>Análisis del entorno .....</b>	<b>5</b>
<b>Análisis del sistema actual.....</b>	<b>7</b>
<b>Solución propuesta .....</b>	<b>7</b>
<b>Planificación temporal del desarrollo del proyecto.....</b>	<b>8</b>
<b>Detalles de la solución adoptada .....</b>	<b>9</b>
Funcionalidad del cliente Android.....	9
Funcionalidad del cliente web.....	10
Requisitos técnicos .....	11
Seguridad .....	12
Comunicación IA.....	14
Arquitectura sistema .....	15
Plataformas y dispositivos compatibles .....	16
Aspectos de seguridad y privacidad .....	17
Interfaz de usuario y experiencia de usuario .....	19
<b>Estudio de la viabilidad del proyecto.....</b>	<b>30</b>
<b>Documentación del diseño e implementación de la solución adoptada.....</b>	<b>32</b>
Arquitectura backend .....	32
<b>Código fuente comentado .....</b>	<b>47</b>
Estructura de carpetas.....	49
<b>Manual de configuración y funcionamiento de la aplicación. ....</b>	<b>53</b>
Aplicación Android.....	53
Aplicación web .....	55
<b>Manual de usuario .....</b>	<b>56</b>
Aplicación móvil.....	56
Aplicación web .....	56
<b>Plan de formación a los Usuarios de la Aplicación .....</b>	<b>58</b>
<b>Bibliografía y fuentes de información .....</b>	<b>58</b>

Agradecimientos a todos los profesores del instituto de IES Villablanca que han ofrecido soporte para poder resolver las dudas y cuestiones que surgieron durante la realización de este proyecto.

# Introducción

Fitness tracker es una aplicación destinada a la gestión de la actividad física y el cuidado de la salud de nuestros clientes. Como tal, ofrece un entorno web en el cual los usuarios pueden seguir estudiando el progreso del ejercicio físico que realizan a lo largo del día. Paralelamente al entorno web está la plataforma de Android, donde los usuarios tienen mejor acceso a sus datos recolectados de un reloj inteligente con sistema Watch OS, a través de un entorno fácil de usar e intuitivo.

El uso del smartwatch ha experimentado una evolución significativa en los últimos años, pasando de ser un dispositivo principalmente utilizado por atletas de alto rendimiento a convertirse en una herramienta accesible y necesaria para la actividad física en general. Además de servir como una extensión del teléfono celular, los smartwatches ofrecen diversas ventajas para quienes practican ejercicio regularmente.

Una de las principales funciones de los smartwatches es la medición del ritmo cardíaco durante el ejercicio. Mediante sensores ópticos integrados, es posible monitorizar la frecuencia cardíaca de manera precisa, lo que permite ajustar la intensidad del entrenamiento y mejorar la salud cardiovascular.

Además, algunos smartwatches ofrecen funciones de control de peso, solicitando datos como estatura y peso para personalizar planes de entrenamiento que apoyen en la pérdida de peso y mejora del estado físico. Estos dispositivos también pueden actuar como entrenadores personales, proporcionando planes de entrenamiento gratuitos adaptados a las necesidades individuales de cada usuario, como sería el caso de nuestra aplicación, Fitness Tracker.

El monitoreo del sueño es otra característica importante de algunos smartwatches, que registran la calidad del descanso y proporcionan información sobre las diferentes fases del sueño, como sueño profundo, REM (del inglés Rapid Eye Movement o sueño de movimientos oculares rápido en castellano), sueño ligero e insomnio. Comprender y mejorar los hábitos de sueño es fundamental para el bienestar físico y mental.

Además, los smartwatches ofrecen análisis de datos detallados en tiempo real, mostrando información relevante como frecuencia cardíaca, pasos dados, calorías quemadas y actividad física realizada. Esta capacidad permite a los usuarios realizar un seguimiento exhaustivo de su estilo de vida y tomar decisiones informadas para mejorar su salud y bienestar.

# Análisis del entorno

Se prevé un mayor consumo en el futuro cercano de este producto, por ello creemos que es relevante invertir en su desarrollo.

## Ventas de relojes inteligentes a nivel mundial de 2016 a 2025 (en millones de unidades)

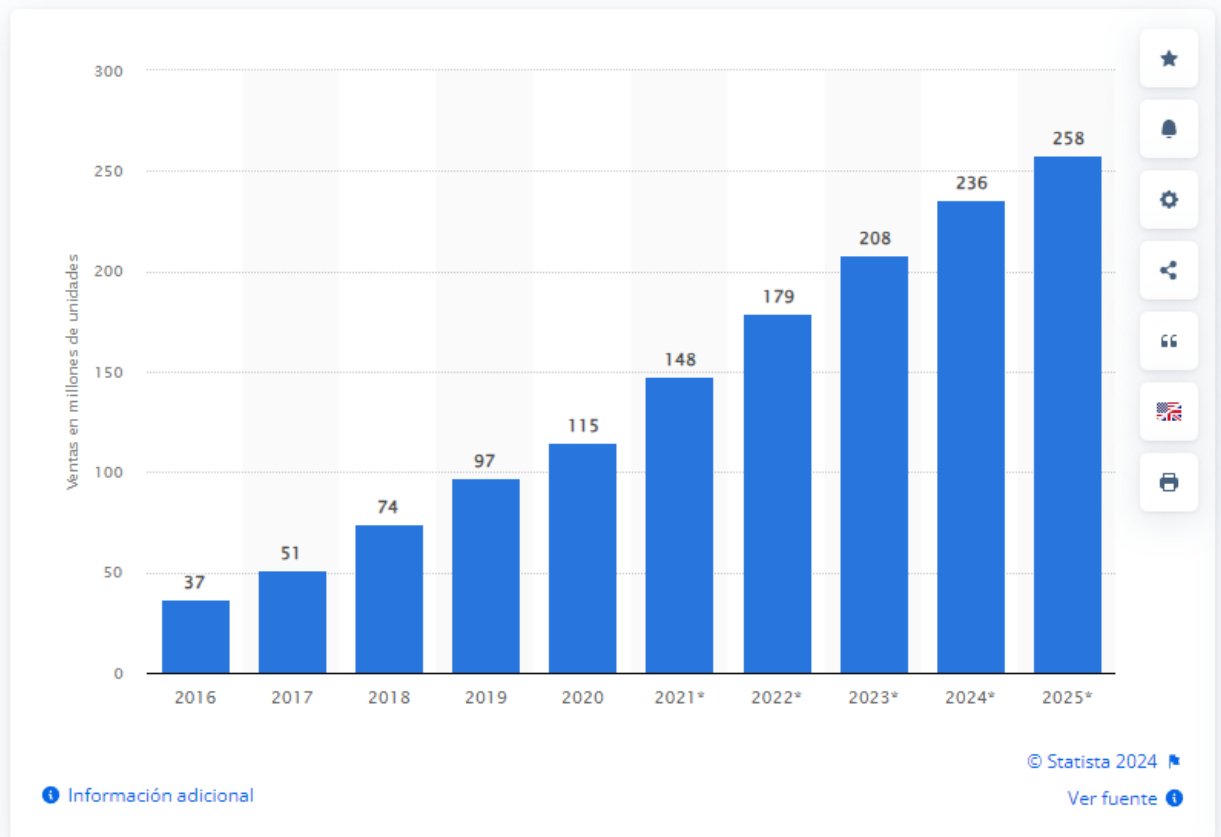


Figura 1. Ventas de relojes inteligentes

A pesar de los avances en la experiencia de usuario, persisten desafíos pendientes, como la gestión de la mensajería, donde la transcripción de voz a texto sigue siendo poco eficaz.

En el ámbito del software, hay una gran diversidad de sistemas operativos para smartwatches, incluyendo WatchOS de Apple, Tizen de Samsung, Harmony OS de Huawei, entre otros. Esta

heterogeneidad refleja la naturaleza incipiente y en evolución continua del mercado de los relojes inteligentes. Este es un aspecto importante a tener en cuenta ya que estos dispositivos son la fuente principal de datos de nuestra aplicación y dicha heterogeneidad dificulta más el desarrollo de aplicaciones que estén destinadas a trabajar con estos dispositivos.

Fitness Tracker es una solución que combina una plataforma web con una aplicación móvil para dispositivos Android. Presentamos una aplicación innovadora que redefine la forma en que los usuarios gestionan su actividad física, rendimiento y salud, al integrar una plataforma web con una aplicación móvil para dispositivos Android.

Se ha decidido desarrollar únicamente para Android en lo que respecta a dispositivos móviles, ya que para desarrollar una versión para iOS implica la necesidad de mayor mano de obra (a parte de aprender a desarrollar en un entorno nuevo) paralelamente con varios requisitos de desarrollo en la plataforma de iOS que no podemos asumir que son: la necesidad de un ordenador Mac para desarrollar aplicaciones para esta plataforma; en caso de querer distribuir la aplicación sería necesario pagar una licencia anual de 99 dólares.

Esta solución ofrece un conjunto completo de funciones básicas diseñadas para mejorar el bienestar general del usuario. Desde el seguimiento de la actividad física hasta el monitoreo del estado de salud, nuestro objetivo es proporcionar una experiencia integral que empodere a los usuarios para alcanzar sus objetivos de salud y rendimiento de manera efectiva y conveniente.

# **Análisis del sistema actual**

Hoy en día, si quieres saber cuántas calorías o saber qué rutina tienes que seguir en x tiempo para alcanzar cierta meta, es muy difícil hacerlo por tu cuenta, ya que tienes que tener registrado qué comes en todo momento, cuales son los nutrientes que contiene lo que consumes. Aparte, también tienes que monitorizar todo el ejercicio diario y lo que consume tu cuerpo simplemente por existir.

Por otro lado, también tienes que conocer tu cuerpo en cuanto a la altura, peso y otros detalles para saber a profundidad qué es lo que necesitas.

Hacer todo eso es muy complejo, ya que tienes que saber sobre nutrición y bienestar, además de perder mucho tiempo realizando todos estos cálculos diariamente.

Por ello, con esta aplicación, lo que conseguiría el usuario sería ahorrarse todos estos datos, los cuales no tendrá que hacer manualmente, y simplemente tendrá que registrar los alimentos consumidos, el ejercicio extra realizado y la cantidad de agua consumida al día. Además, podrá consultar con la inteligencia artificial rutinas y dietas que se adaptarán todo lo posible al usuario.

Con esta aplicación, el usuario podrá modificar sus datos de forma sencilla, le aparecerá la información necesaria, y podrá ver todo su progreso en la App desde que la instaló.

## **Solución propuesta**

Para afrontar el proyecto a desarrollar, necesitaremos distintas tecnologías para desarrollarlo, tanto para la parte de desarrollo web, Android, y la parte del servidor.

Para la parte del Servidor donde guardaremos los datos de los usuarios utilizaremos SpringBoot con Java, ya que nos da las suficientes ventajas para facilitar esos microservicios correctamente. Para la parte de los alimentos y la inteligencia artificial utilizaremos .net core para facilitar esas uniones con el cliente y la IA, aparte que es muy fácil de implementar la seguridad de la misma.

Para la base de datos utilizaremos Mongo, ya que no sabemos cuántos datos puede llegar a meter el usuario, aparte que habrá más escrituras que lecturas

Para la parte Android utilizaremos Jetpack compose y Kotlin, ya que jetpack compose nos permite crear aplicaciones Android más compactas y que tiene más funcionalidades como por ejemplo el manejo de los estados en la App.

Para la parte Web, utilizaremos React porque es sencillo de manejar y para aplicaciones pequeñas viene mejor.

El proyecto a realizar tendrá un pequeño coste debido a que hay que pagar una pequeña cuota para utilizar la IA y comunicar a través del backend, y para el tema de los alimentos, ya que no se puede hacer llamadas ilimitadas. Para intentar recuperar esa inversión, la opción de poder utilizar la IA será de pago para amortizar esa pérdida entre la IA y los alimentos.

## **Planificación temporal del desarrollo del proyecto**

El desarrollo del proyecto se estructura en varias fases bien definidas, cada una con tiempos estimados para asegurar un avance ordenado y eficiente. La primera fase es el Análisis de Requisitos del Producto, que abarca tanto nuestras aplicaciones web y Android como el backend necesario para soportarlas. En esta etapa inicial, se recoge toda la información relevante de los usuarios, se identifican las funcionalidades esenciales y se establecen los objetivos específicos del proyecto. Este proceso detallado nos permite tener una visión clara de las necesidades y expectativas de los usuarios, y suele estimamos que nos tomaría aproximadamente dos semanas.

Una vez completado el análisis de requisitos, se avanza hacia la fase de Diseño del Modelo de Datos. Aquí, se define la estructura de la base de datos que soportará todas las funcionalidades del sistema. Este modelo debe ser robusto y flexible para manejar eficientemente los datos de los usuarios y las operaciones del sistema. Esta fase incluye la creación de diagramas entidad-relación y la especificación de las tablas y relaciones, y generalmente se completa en un período de dos semanas.

Con el modelo de datos definido, se procede a la fase de Desarrollo del Primer Prototipo. Este prototipo inicial es crucial para visualizar cómo interactuarán los usuarios con el sistema. Se desarrolla tanto la interfaz de la aplicación web como la de Android, junto con las primeras versiones de los servicios backend. El objetivo es tener una versión funcional que permita realizar pruebas iniciales y recibir feedback. Esta fase es intensiva y suele extenderse durante cuatro semanas, permitiendo iterar sobre las primeras versiones de la interfaz y los servicios backend.



Finalmente, se llega a la fase de Despliegue y Puesta en Marcha de los Servicios en el Servidor. En esta etapa, todos los componentes del sistema se integran y se despliegan en los servidores correspondientes. Utilizamos AWS para alojar el backend, aprovechando su fiabilidad y escalabilidad, y Vercel para desplegar la aplicación web, facilitando un flujo de trabajo ágil y efectivo. Esta fase también incluye la configuración de dominios, la seguridad y la optimización del rendimiento. Se estima que esta última fase tome alrededor de dos semanas, asegurando que todo esté correctamente configurado y funcionando antes del lanzamiento oficial.

## **Detalles de la solución adoptada**

### **Funcionalidad del cliente Android**

#### **Registro de usuario**

Permite a los usuarios crear una cuenta en la aplicación proporcionando la información necesaria.

Un usuario para poder registrarse necesitará ingresar por un formulario el nombre de usuario, este es un nombre opcional con que el usuario se identifica dentro de la aplicación, se utiliza en casos de generar informes para él, este campo no es obligatorio de rellenar en el registro, en cuyo caso se identificaría con el nombre completo en la aplicación (incluyendo apellidos), más tarde puede editarse si se desea; el usuario introducirá su nombre, primer y segundo apellido; la contraseña, el correo electrónico y la fecha de nacimiento con que se puede determinar la edad más tarde si es necesario.

#### **Perfil de usuario**

Permite a los usuarios agregar y editar su información personal, donde se incluye la edad, el peso, la altura, la contraseña, el nombre de usuario, el nombre, los apellidos y el sexo, en esencia los mismos datos que en el momento de registro, sin embargo, la dirección de correo no se puede modificar y será siempre la misma.

#### **Notificaciones y recordatorios**

Envía notificaciones y recordatorios a los usuarios para mantenerlos informados sobre sus metas, progreso y otras actividades relevantes.

La aplicación de Android será capaz de generar una notificación para el usuario poder revisar las comidas que tocan en un día en concreto, por otra parte.

### **Establecimiento de objetivos**

Permite a los usuarios establecer objetivos personalizados relacionados con la actividad física, el rendimiento y la salud, como la cantidad de ejercicio semanal, la cantidad de pasos diarios, etc.

### **Monitoreo en tiempo real**

Proporciona a los usuarios la capacidad de monitorear su actividad física y rendimiento en tiempo real a través de la aplicación; entre estos datos se incluye la medición del ritmo cardíaco, el número de pasos realizados en un determinado tiempo, el tiempo de sueño, el nivel de oxígeno en sangre y el número estimado de calorías quemadas con el ejercicio físico realizado.

### **Control de objetivos establecidos**

Permite a los usuarios realizar un seguimiento del progreso hacia sus objetivos establecidos, proporcionando estadísticas y métricas relevantes para evaluar su rendimiento.

Aquí se incluye un listado de elementos por cumplir en formato ToDo List, esto es, el usuario podrá establecer objetivos de quema de calorías que automáticamente se marcarán como hechas de acuerdo al progreso que lleve el cliente.

## **Funcionalidad del cliente web**

Este cliente al igual que el cliente android, ofrece la funcionalidad de registro a inicio de sesión y la sección de perfil de usuario.

### **Control de objetivos establecidos**

Permite a los usuarios realizar un seguimiento de sus objetivos establecidos, tanto en la aplicación móvil como en la plataforma web, para garantizar una experiencia coherente y sin problemas en ambos dispositivos.

## Historial de actividades

Proporciona a los usuarios un registro detallado de todas sus actividades pasadas, incluidos entrenamientos, mediciones de salud, logros alcanzados, etc., para ayudar en la evaluación del progreso a lo largo del tiempo.

## Requisitos técnicos

Como se ha mencionado anteriormente, se va a elaborar una para la plataforma Android y una aplicación web disponible para escritorio.

Para el desarrollo de la interfaz en Android se utilizará la librería de Jetpack Compose mediante el lenguaje de programación Kotlin, ambos muy comunes para el desarrollo de interfaces en Android. Para mejorar la compatibilidad entre dispositivos Android y maximizar el número de dispositivos móviles en que se puede utilizar nuestra aplicación para Android, se va a desarrollar para la versión 8.0 de este sistema operativo con la API 26.

Para establecer conexión entre esta aplicación y la API REST se utilizará la librería de Retrofit, disponible también para el lenguaje de programación Kotlin.

La aplicación web se va a desarrollar sobre el framework de React usando JavaScript como lenguaje de programación. Se prevé la utilización de la librería de CSS Bootstrap para agilizar el desarrollo del diseño de las interfaces.

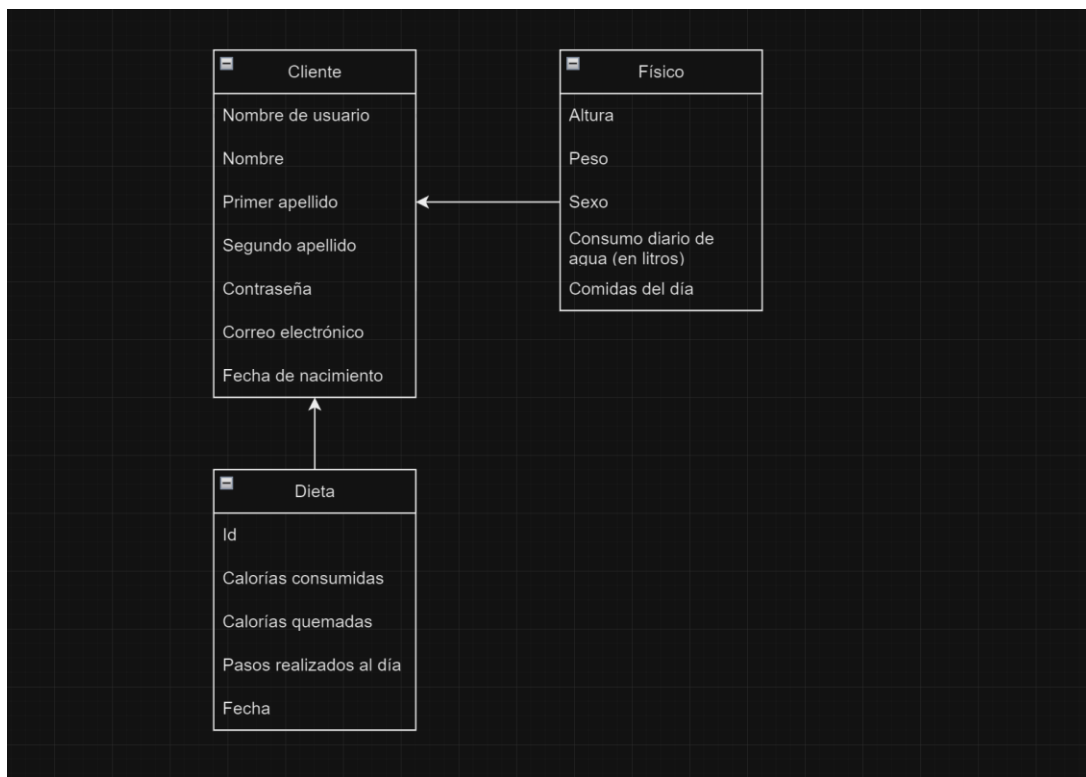
## Almacenamiento de datos

Del cliente se van a recoger datos identificativos, a citar: el nombre completo junto con los apellidos; una dirección de correo electrónico la cual será única por cuenta en nuestra base de datos, de modo que no admitimos varias cuentas con la misma dirección de correo electrónico; una contraseña para la cuenta del usuario y finalmente un nickname identificativo de la cuenta del usuario que es opcional a la hora de registrarse.

Un usuario registrado puede almacenar los siguientes datos actualmente: la altura, el peso, el sexo y la fecha de nacimiento, con que se deduce la edad. Estos datos se almacenan para mantener constancia de su metabolismo basal, el Índice de Masa Muscular o IMC, estimar consumo

recomendado de agua al día, requerimiento calórico diario para así poder estimar también el número de proteínas, grasas o carbohidratos que se han de consumir.

Por otro lado, también se pretende almacenar datos de dietas como: el número de comidas al día (desayuno, media mañana, almuerzo, merienda, cena); cantidad de agua consumida al día en un día concreto (valor estimado); ejercicio físico hecho a lo largo del día. A continuación, se muestra un esquema provisional de los datos que se van a guardar en la base de datos.



## Seguridad

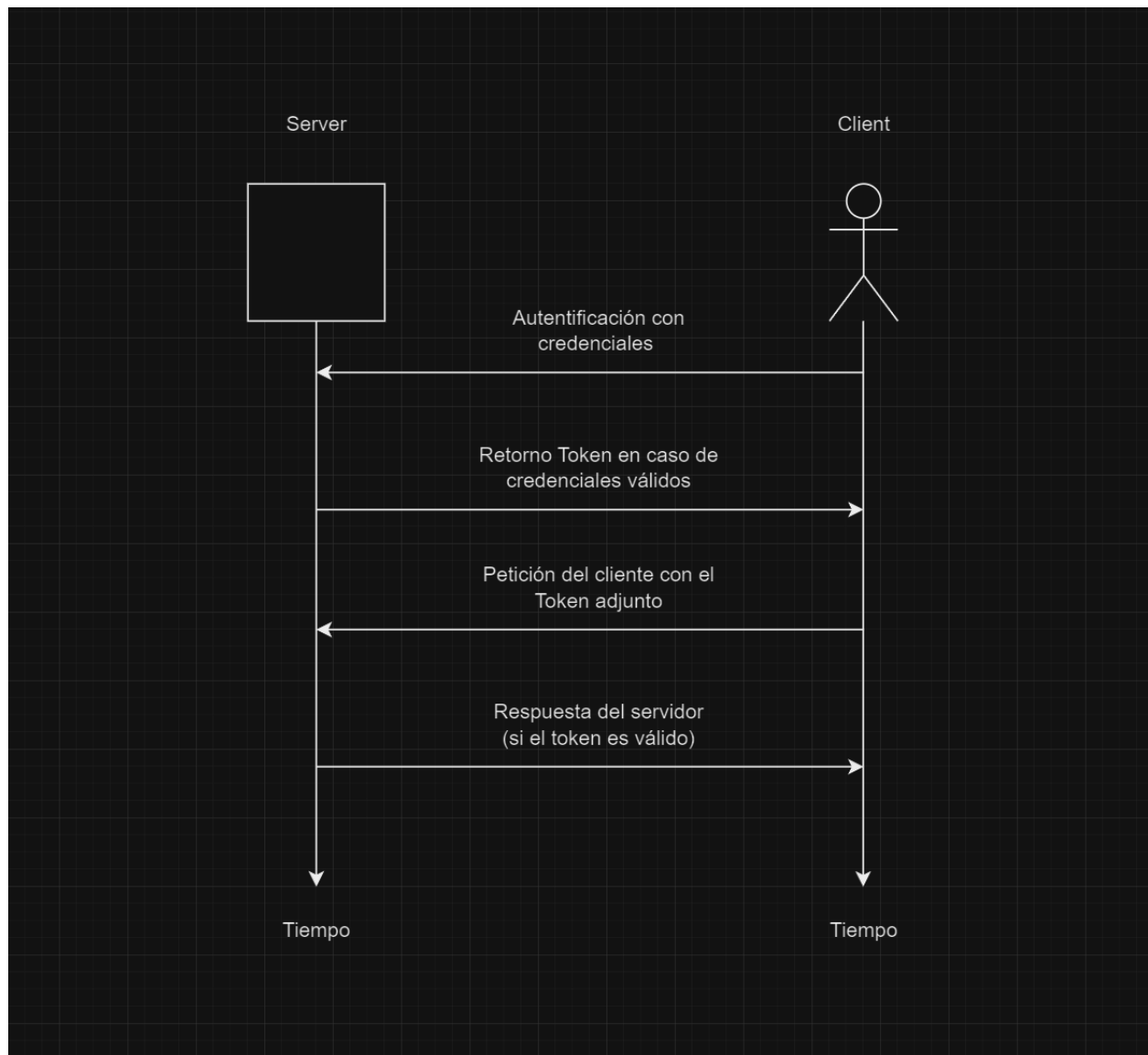
La seguridad es un aspecto importante a tener en cuenta. Este aspecto se va a trabajar tanto en el servidor como en ambos clientes, el de Android y la aplicación web.

Para garantizar la seguridad en la aplicación web, será fundamental restringir el acceso a ciertas rutas únicamente a usuarios autenticados. Esto se va a lograr mediante el uso de componentes de enrutamiento, mediante **react-router**, que permite definir rutas privadas y públicas. Las rutas privadas estarán protegidas y solo serán accesibles para usuarios que han iniciado sesión

correctamente, mientras que las rutas públicas están disponibles para todos los usuarios. Al implementar este enfoque, se puede controlar de manera efectiva qué contenido y funcionalidades son accesibles para cada tipo de usuario.

En la aplicación Android el enfoque será similar, será necesario iniciar sesión para poder acceder a las funcionalidades de la aplicación.

La comunicación con el servidor se realizará mediante validación por token. El usuario ingresa sus credenciales en alguna de las aplicaciones clientes para poder iniciar una nueva sesión, validados las credenciales, el usuario recibirá un token con que podrá comunicarse con el servidor de manera segura. A continuación, un esquema representativo de esta arquitectura:



## Comunicación IA

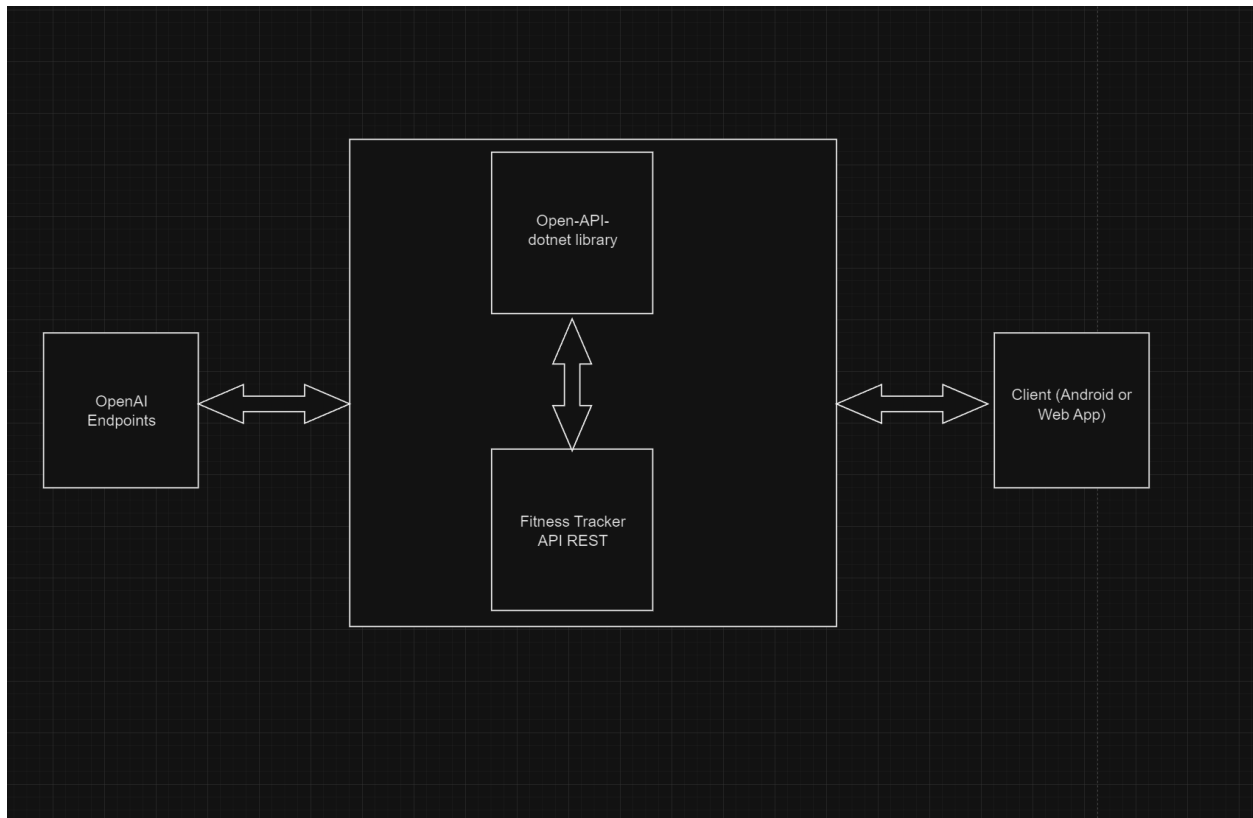
Se va a establecer un servicio que se comunica con el modelo de lenguaje GPT 4 Vision mediante peticiones HTTP. Para ello se va a desarrollar una API REST en C# utilizando ASP.NET Core junto con la librería OpenAI-API-dotnet, una librería que facilita la comunicación con los servicios de OpenAI, esta librería nos abstrae de la tarea de realizar las peticiones a los endpoints de OpenAI y nos devuelve los resultados a través de objetos con que podemos trabajar con facilidad en nuestra aplicación.

Importante destacar que este servicio está disponible para usuarios autenticados, como tal, la API REST que se comunica con OpenAI-API-dotnet también deberá validar las credenciales del usuario antes de poder realizar cualquier petición a la inteligencia artificial, para ello usamos una clave que nos ofrece OpenAI.

Para utilizar el servicio de inteligencia artificial, los usuarios tendrán que confirmar sus credenciales, esto es así ya que se pretende cambiar este comportamiento en futuro para restringir su uso a usuarios a través de un plan de pago.

Para comunicarse con las APIs de Fitness Tracker se utilizará objetos JSON o parámetros en la URI, como tal se tiene que validar estos datos. Para validar los JSON de entrada se va a usar la librería de FluentValidation de .NET para asegurar que no se llaman a estos endpoints con datos inadecuados.

A continuación, un esquema del sistema (las flechas representan el flujo de datos):

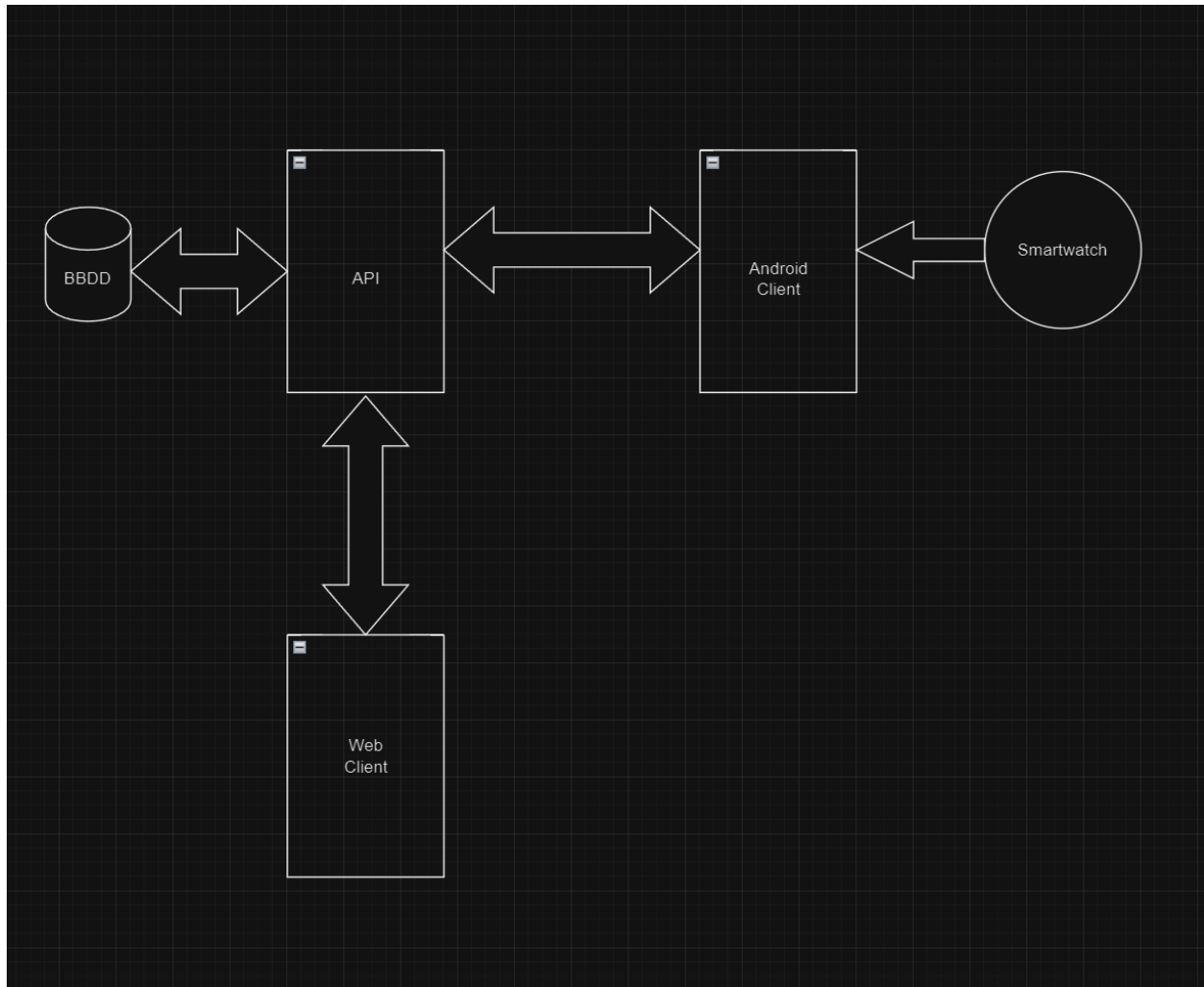


## Arquitectura sistema

Nuestro sistema consta de tres elementos principalmente: un backend que consta de una API REST sobre una base de datos mongo y una API REST con ASP.NET que sirve de enlace con los endpoints de la IA con que se va a realizar los prompts, a parte de también realizar la comunicación entre nuestro backend y otros servicios externos que sean necesarios como el de información nutricional Edamam. Nuestros clientes se conectarán a la API de ASP.NET que consta de un API Gateway que redirige las peticiones a los servicios que tenemos en nuestro sistema ejecutándose, de esta forma se simplifica donde implementar la seguridad y tenemos más control sobre qué puertos se deben exponer al exterior y cuáles no; dos clientes que interaccionan con este backend y el último componente sería nuestro reloj inteligente que sirve como fuente de datos para el cliente Android.

Por la naturaleza de los datos que se van a guardar en la base de datos, esta va a ser documental. Para integrar esta base de datos se va utilizar MongoDB.

A continuación, se muestra un esquema del sistema:



## Plataformas y dispositivos compatibles

La aplicación estaría disponible para dispositivos móviles Android, siendo esta desarrollada en Kotlin con Jetpack Compose.

También, esta aplicación estará disponible para la parte web, estado desarrollada en React.



Aparte, esta aplicación estará disponible para smartwatch. En un principio, se desarrollará para los relojes que tengan integrado Wear OS, y llevándolo a todos los demás distintos tipos de relojes en un futuro.

Para verificar que las funciones se complementan correctamente, usaremos Postman para hacer los test y comprobar que las llamadas devuelven los datos correctos, así como Swagger para ver qué parámetros se les pasa a las llamadas y qué devuelven.

Para el cliente, usaremos el emulador y varios dispositivos móviles para comprobar que todo se ve de la forma esperada.

## Aspectos de seguridad y privacidad

Garantizar la seguridad de los datos del usuario mediante técnicas de cifrado y autenticación segura.

Ya que nuestra aplicación está pensada para un entorno abierto es necesario cifrar los datos de manera que las claves de cifrado no puedan ser interceptadas por un tercero durante el envío de la misma. Por ello usaremos cifrado de tipo asimétrico.

Utilizaremos dos claves diferentes que están vinculadas entre sí matemáticamente. La app mantendrá en secreto la clave privada, mientras que la pública se compartirá entre todos los clientes de nuestro servicio. Con esto conseguiremos que los datos se muevan de manera cifrada a través de la red y solo puedan ser consultados por el cliente.

En cuanto al autenticador que usaremos tanto para nuestra aplicación web como para la aplicación móvil, usaremos la autenticación por JWT. Para realizar peticiones al backend se debe iniciar sesión en un usuario válido primero, este recibirá un token con que puede hacer peticiones a los endpoints privados.

**Cumplimiento de regulaciones de privacidad**, como GDPR, para proteger la privacidad de los usuarios y su información personal.

Nuestro principal objetivo es la seguridad y privacidad de nuestros usuarios por lo tanto seguiremos el protocolo de regulación de privacidad GDPR. Este garantiza la protección y privacidad de datos de los usuarios. Los principios básicos de GDPR son:

- **Transparencia:** Debemos tratar los datos personales dentro de la ley, de manera justa y transparente. Esto significa que precisamos siempre de notificar que estamos recolectando información y debemos, aun, especificar cuál información y cómo será utilizada.
- **Propósito delimitado:** Debemos recolectar apenas datos específicos con intenciones específicas. Jamás podemos procesar información más allá de las que contemplan las intenciones específicas e informadas al usuario.
- **Minimización de datos:** Apenas podemos recolectar datos personales adecuados y relevantes para nuestras intenciones. Esto significa que recolectar o cuestionar cualquier información no relacionada al servicio ofrecido está prohibido.
- **Precisión:** Todo dato personal que recolectamos debe ser correcto, claro y, cuando sea necesario, ser actualizado para estar en día con la información personal del usuario.
- **Eliminación de Datos:** Los datos personales de los usuarios solo deben ser mantenidos mientras son necesarios y sean útiles para el propósito original.
- **Seguridad:** Nuestra organización debe usar técnicas apropiadas y medidas de seguridad para proteger datos personales contra el procesamiento no autorizado o vaciamiento. Acceso, pérdida o alteración indebida de los datos implica en penalidades. Dependiendo del caso es recomendado, cuando no obligatorio, el uso de datos segregados, criptografados, seudonimizados o anonimizados.

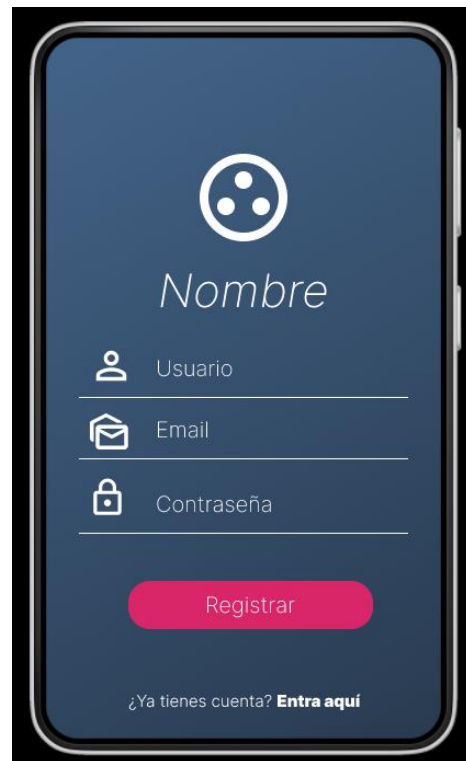
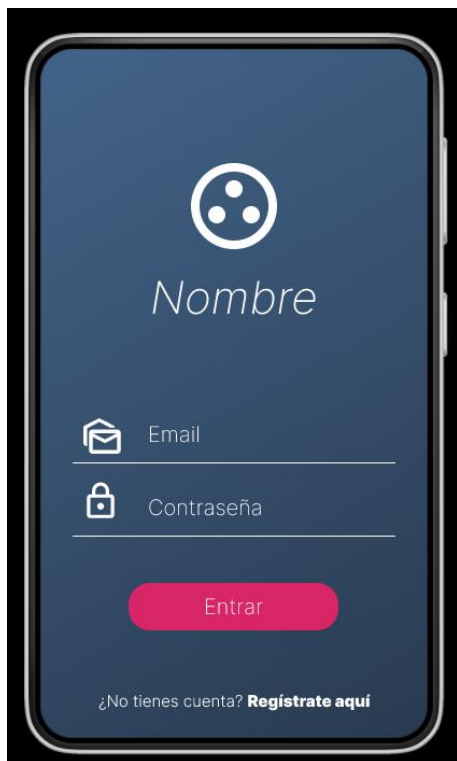
Seguiremos todos y cada uno de estos puntos para preservar la seguridad y privacidad de cada uno de nuestros usuarios.

## Interfaz de usuario y experiencia de usuario

Para la interfaz de usuario de la aplicación móvil, se hará de tal forma para que la navegación y la introducción de datos de la misma sea lo más intuitiva y sencilla posible.

Para empezar, en la parte del login, se hará una interfaz sencilla que se centre tanto en los campos de email, contraseña y el botón, y que cuando presione sobre uno de estos quede marcado para que se dé cuenta de qué está situado en uno de los campos, todo en una misma ventana y sin que tenga que scrollar en esta.

Se adjunta una foto de un prototipo de cómo quedaría el login y el register.



En la parte del menú principal, en la parte de arriba, el usuario tendrá un menú donde podrá navegar entre las distintas pantallas de forma sencilla, a parte de un menú desplegable en la parte lateral para que el usuario tenga opciones para poder personalizar algunas partes de la app, como los colores, etc.

En la primera pantalla se verían todos los datos intrínsecos del usuario, permitiéndole cambiar estos mismos cuando él quiera. También, se mostrarán los datos básicos sacados de los hablados

anteriormente, como el imc, gasto calórico diario, etc. Estos cambiarán cuando el usuario cambie sus propios datos.

La interfaz quedaría de la siguiente manera:



En la segunda pantalla estará todo lo relacionado con las consumiciones diarias, cuántas calorías, grasas, carbohidratos y proteínas recomendadas tiene que consumir el usuario. Además, debajo de estos vendrán las comidas y las dietas sugeridas para el usuario por la inteligencia artificial. Toda esta información vendrá ordenada en una columna en la que el usuario podrá scrollear para poder tener acceso a la información.

Adjunto una foto del prototipo de como quedaría.



En la última ventana, se mostrarían todos los alimentos consumidos por el usuario en el día, pudiendo ver los registros de otro día si así lo quisiera el usuario. En esta pantalla, se repartiría el desayuno, comida y cena, pudiendo el usuario cambiar los nombres o meter otra comida entre medias, y midiendo las calorías que lleva ese día junto con las que debe suplir. Esto se hace para motivar al usuario que intente hacer un poco más de ejercicio o para que consuma la cantidad de agua necesaria al día, marcándolo adecuadamente si ha conseguido la meta diaria.

Adjunto una foto del prototipo de la pantalla.



El [siguiente enlace](#) redirige a una vista previa desde el navegador de la interfaz:

La interfaz web estará creada de manera que tenga un inicio llamativo para atraer a nuevos clientes, una vez iniciada la sesión en la web el diseño será más sencillo e intuitivo para la facilidad de uso del usuario.

La web de inicio será la siguiente la cual tiene un buen diseño con algunas partes en las que el usuario podrá ver reflejado como es el funcionamiento de nuestra app y para qué es. En esta web de inicio se podrá deslizar hasta llegar al pie de página.

Adjunto imágenes de los diferentes puntos de la página de inicio.



Fitness-Tracker

[Home](#)

[App](#)

[Nuestros Clientes](#)

[Sobre Nosotros](#)

[Registrarse](#)

## Virtualiza tu cuidado personal

Fitness-Tracker provee de un seguimiento personal y de salud para tu beneficio personal en base a tus objetivos y metas

[Consigue nuestra App](#)



## Nuestros Servicios

Le brindamos las mejores opciones para usted. Ajustalo a tus necesidades de salud y metas personales. Puedes consultar con nosotros qué tipo de ejercicios son mejores para alcanzar tus objetivos.



### Busqueda Inteligente

Encuentra y conoce todo aquello que necesites gracias a nuestra IA interactiva acerca de la salud y el fitness.



### Smartwatch

Escoge cualquier smartwatch y comienza a usar la app para mejorar tu salud y alcanzar tus objetivos.



### Consulta Personalizada

Consulta gratuitamente todo aquello que necesites para alcanzar tus objetivos con nuestra IA.



### Info detallada

Consulta la información acerca de tus entrenamientos y rendimiento diario.



### Cuenta

Accede a tus registros desde cualquier lugar y dispositivo: descargándote la app e iniciando sesión o desde nuestra web.



### Tracking

Trackea tus entrenamientos, tu día a día, ¡incluso tus periodos de sueño!

[Saber más](#)



## Lideres en salud y fitness con nuevas tecnologías

Fitness-Tracker ofrece seguimiento de salud, rutinas de gimnasio y dieta de nutrición para alcanzar tus objetivos, accesible a través de dispositivos móviles y en línea para todos.

[Saber más](#)



## Descarga nuestra app para dispositivos móviles

Nuestra aplicación dedicada a la recolección de datos a través del software del smartwatch y soluciones para salud, nutrición y rutinas fitness para el beneficio personal del usuario. Comienza a alcanzar tus objetivos.

Descargar ↓



**Fitness-Tracker**

Fitness-Tracker provee de soluciones saludables para tu beneficio personal

©Fitness-Tracker 2024. All rights reserved

### Compañía

Sobre nosotros  
Encuentra entrenador  
App

### Region

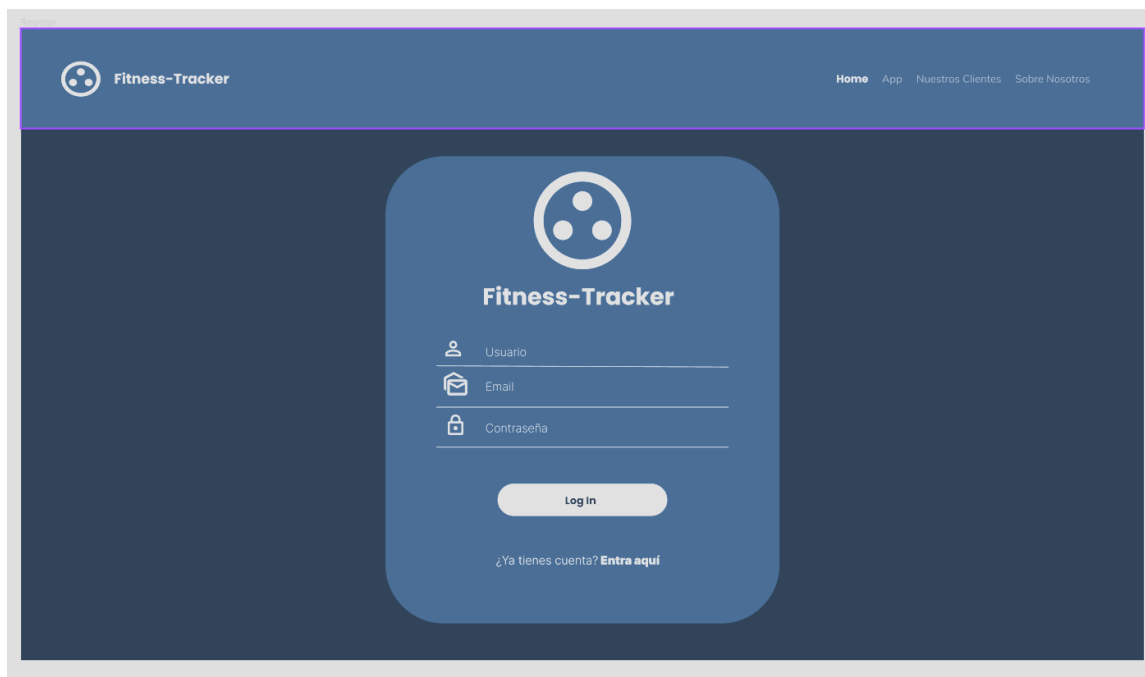
España

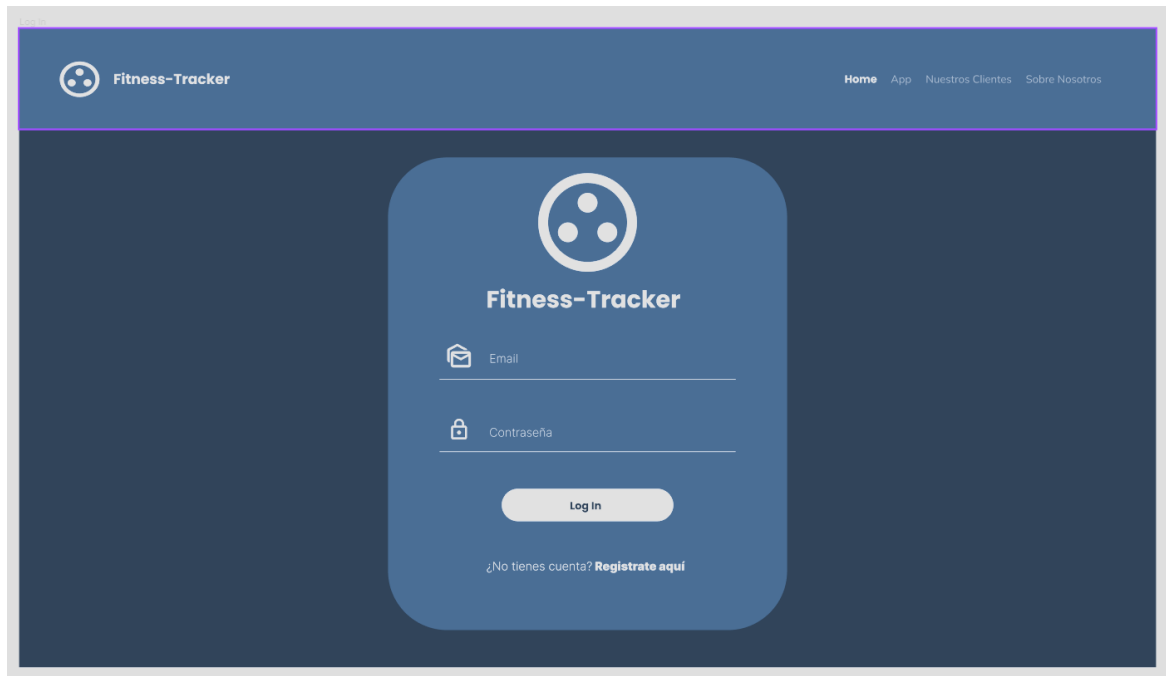
### Ayuda

Centro de ayuda  
Contacto de soporte  
Instrucciones  
Como funciona

El apartado de login y register para el usuario será una interfaz sencilla que estará compuesta de manera que se centre tanto en los campos de email, contraseña, el botón. Cuando uno de estos campos este seleccionado quedará marcado para que el usuario se dé cuenta de que está situado en uno de los campos todo en una misma ventana y que no tenga que scrolllear.

Se adjunta imagen del login y register.






Una vez iniciada la sesión en la web, en la parte de arriba el usuario tendrá una barra de navegación a parte, por donde podrá navegar entre los diferentes apartados de la aplicación web.

En el perfil el usuario podrá ver tanto sus datos personales como su imagen de perfil, objetivos y los resultados actuales. Estos irán cambiando conforme el usuario va avanzando con el uso de la app.


Adjunto apartado del perfil.

Perfil



**Fitness-Tracker**

[Home](#)
[App](#)
[Nuestros Clientes](#)
[Sobre Nosotros](#)
[Guillermo Quintanar](#)

[Inicio](#)
[Perfil](#)
[Hoy](#)
[Calorías diarias](#)



**MARISA LOPEZ SÁNCHEZ**

 mlopez43@gmail.com

### Datos Personales

- Nick.....FastMarisa33
- Nombre.....Marisa Lopez Sánchez
- Edad.....25

### Resultados

- Dato1.....Valor\_1
- Dato2.....Valor\_2
- Dato3.....Valor\_3

### Objetivos

- Dato3.....Valor\_1
- Dato2.....Valor\_2
- Dato1.....Valor\_3

En el apartado “Hoy” se mostrarían todos los alimentos consumidos por el usuario en el día, pudiendo ver los registros de otro día si así lo quisiera el usuario. Esta pantalla se repartiría entre el desayuno, comida y cena, pudiendo el usuario cambiar los nombre meter otra comida entre medias, y midiendo las calorías que lleva ese día junto con las que debe suplir. Esto se hace para motivar al usuario que intente hacer un poco más de ejercicio o para que consuma la cantidad de agua necesaria al día, marcándolo adecuadamente si ha conseguido la meta diaria.

Adjunto foto del apartado hoy.

Hoy

**Fitness-Tracker** Home App Nuestros Clientes Sobre Nosotros Guillermo Quintanar

Inicio Perfil **Hoy** Calorías diarias

**Kilocalorías totales**  
0/255 g

- Carbohidratos 0/255 g
- Grasas 0/255 g
- Proteínas 0/255 g

**Desayuno**

- Dato1.....Valor\_1
- Dato2.....Valor\_2
- Dato3.....Valor\_3

**Resultados**

- Dato1.....Valor\_1
- Dato2.....Valor\_2
- Dato3.....Valor\_3

**Datos Personales**

- Dato3.....Valor\_1
- Dato2.....Valor\_2
- Dato1.....Valor\_3

En el apartado “Calorías Diarias” el usuario podrá encontrar todo lo relacionado con las consumiciones diarias, cuántas calorías, grasas, carbohidratos y proteínas recomendadas tiene que consumir el usuario. Además debajo de estos vendrán las comidas y las dietas sugeridas para el usuario por la inteligencia artificial.

Adjunto foto del apartado “Calorías diarias”

Requerimiento calórico diario

**Fitness-Tracker**

Home App Nuestros Clientes Sobre Nosotros Guillermo Quintanar

Inicio Perfil Hoy Calorías diarias

**Requerimiento Calórico Diario**

0/255 g

- Carbohidratos 255 g
- Grasas 255 g
- Proteínas 255 g

**Dietas y ejercicios**

- Dato1.....Valor\_1
- Dato2.....Valor\_2
- Dato3.....Valor\_3

**Mis comidas**

- Dato1.....Valor\_1
- Dato2.....Valor\_2
- Dato3.....Valor\_3

**Dietas sugeridas**

- Dato3.....Valor\_1
- Dato2.....Valor\_2
- Dato1.....Valor\_3

Link al prototipo o haz clic [aquí](#):

## Estudio de la viabilidad del proyecto

El proyecto Fitness Tracker es una aplicación web diseñada para ayudar a los usuarios a monitorear y mejorar su estado físico y salud. La viabilidad del proyecto depende de diversos factores financieros, técnicos y de mercado. Este estudio analiza estos factores para evaluar la factibilidad del desarrollo y mantenimiento de la aplicación.

El análisis financiero del proyecto comienza con la consideración de los costos de desarrollo y mantenimiento. El uso de la inteligencia artificial a través de la API de OpenAI implica suscribirse a su plan de precios, como se explicó en apartados anteriores. Publicar la aplicación en Google

Play Store tiene un costo único de 25 euros. Actualmente, el hosting web se realiza en un servicio gratuito, aunque sin dominio propio. El backend se hospeda en una instancia EC2 de Amazon Web Services (AWS) con una IP elástica, y hasta ahora se ha invertido 6 euros, con un crédito gratuito de \$100 a través de AWS Academy que cubre futuros gastos. Los costos de AWS pueden aumentar dependiendo del tráfico y el uso.

En cuanto a los ingresos potenciales, se pueden implementar modelos de suscripción que ofrecen acceso a funciones premium, dietas personalizadas y rutinas de ejercicio. La publicidad dentro de la aplicación puede generar ingresos adicionales, y la venta de productos complementarios como equipos de fitness o suplementos puede ser otra fuente de ingresos, pero es una característica que para esta entrega del proyecto no se considera implementar debido al corto plazo de tiempo del que se dispone.

Desde una perspectiva técnica, la infraestructura del proyecto es robusta y escalable. AWS EC2 permite escalar fácilmente la infraestructura a medida que crece la base de usuarios. La integración con la API de OpenAI proporciona capacidades avanzadas de inteligencia artificial con el modelo de lenguaje GPT4 Turbo Vision (que es el que se está usando ahora), mejorando la personalización y recomendación de rutinas y dietas. Es necesario planificar actualizaciones regulares para mejorar la aplicación, corregir errores y agregar nuevas funcionalidades. Mantener la seguridad de los datos de los usuarios es crucial, y AWS proporciona herramientas de seguridad robustas que deben ser implementadas y monitoreadas constantemente, entre ellas reglas de entrada para sólo exponer aquellos puertos por los que queremos recibir peticiones, incluso podemos llegar a bloquear peticiones provenientes de ciertas direcciones IP mejorando así la seguridad de nuestro entorno backend.

El análisis de mercado revela que hay una demanda creciente por aplicaciones de salud y fitness. El interés en la salud y el fitness ha aumentado, especialmente con la creciente conciencia sobre la importancia de un estilo de vida saludable. Sin embargo, existe una competencia intensa en el mercado de aplicaciones de fitness. Fitness Tracker debe diferenciarse ofreciendo funcionalidades únicas y personalizadas a través del uso de inteligencia artificial. Los usuarios objetivos son jóvenes y adultos interesados en el fitness y la salud, que buscan soluciones digitales para seguir y mejorar su régimen de fitness.

# Documentación del diseño e implementación de la solución adoptada.

## Arquitectura backend

Nuestro sistema consta de tres componentes. Dos servicios REST con sus respectivos endpoints. Uno de los servicios REST es el Base que contiene la lógica de negocio relacionada con el manejo de cuentas de clientes, así como su validación al iniciar sesión; paralelo a este está el servicio REST Sistema External Services, que, entre otras APIs, incluye la de la IA y una sobre información nutricional.

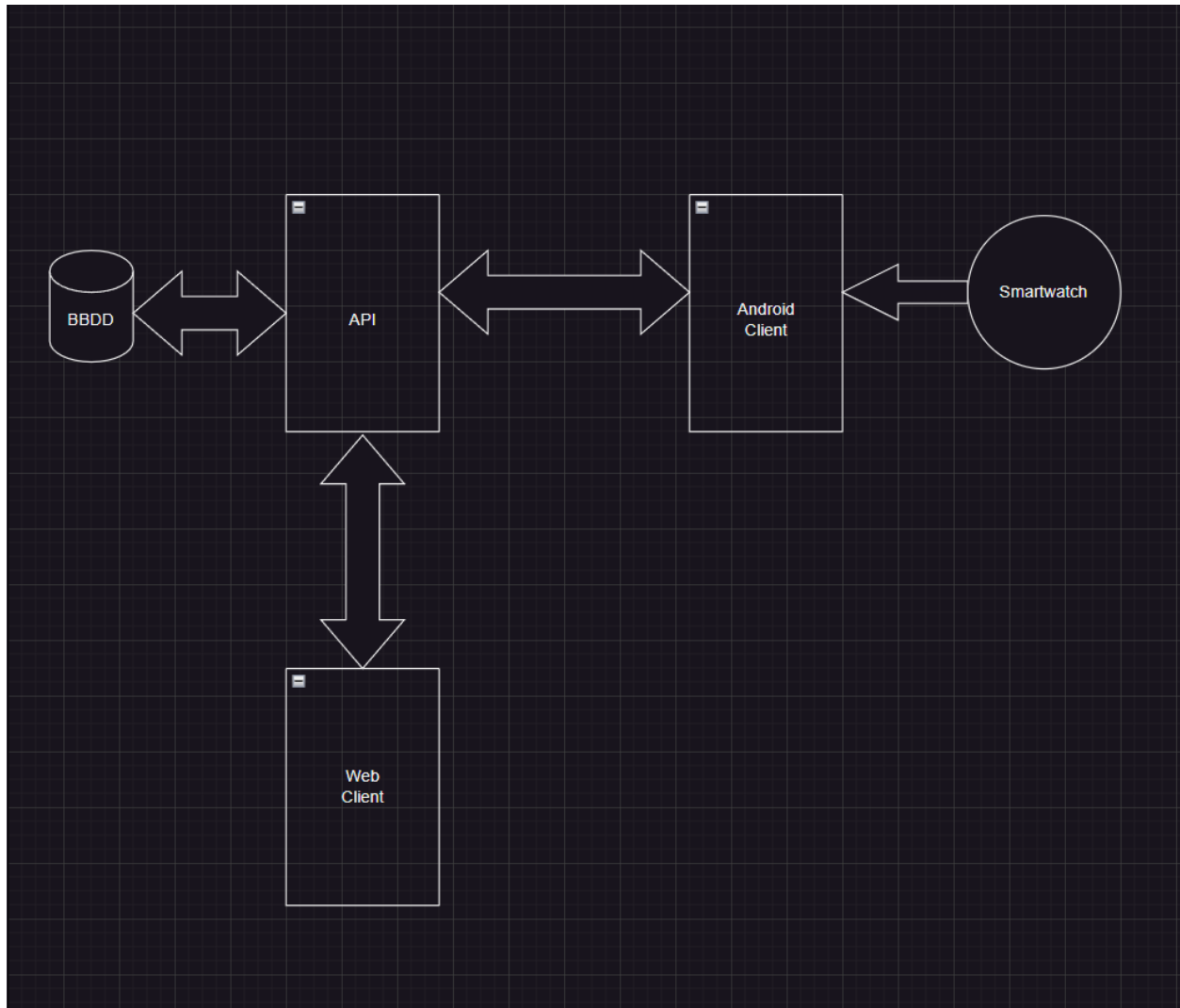
El API de la IA es una que maneja la lógica de negocio respecto la generación de itinerarios relativos a dietas o rutinas y que también puede asistir al cliente en las necesidades que este pueda tener en relación con sus rutinas de ejercicios físicos diaria, entre otros aspectos. Esta API es el centro de nuestro backend, sobre él está integrado el API Gateway, que es la pieza que se expone al exterior, es decir, todos nuestros clientes realizarán peticiones contra el API Gateway, el cual se encarga de enrutar las peticiones a cada uno de los servicios que implementa nuestro backend. Esto especialmente importante ya que nuestro servidor solo expone al exterior el puerto del api Gateway, cortando cualquier petición a otro puerto y dándonos mejor control sobre que servicios pueden realizar peticiones los clientes y cuáles no.

Por otro lado, tenemos dos clientes. Un cliente web y uno Android. Ambos podrán ser capaces de comunicarse con nuestro backend. El cliente Android por su lado consta de otra fuente de datos que es el reloj inteligente, este le proporciona datos como el número de pasos realizados al día, calorías quemadas en un período de tiempo, entre otros.

Por la naturaleza de los datos que se van a guardar en la base de datos y cómo están relacionados los modelos, estos se van a serializar a una base de datos documental, para ello se utilizará MongoDB que es un gestor de bases de datos NoSQL con muy eficiente para lecturas frecuentes de datos.

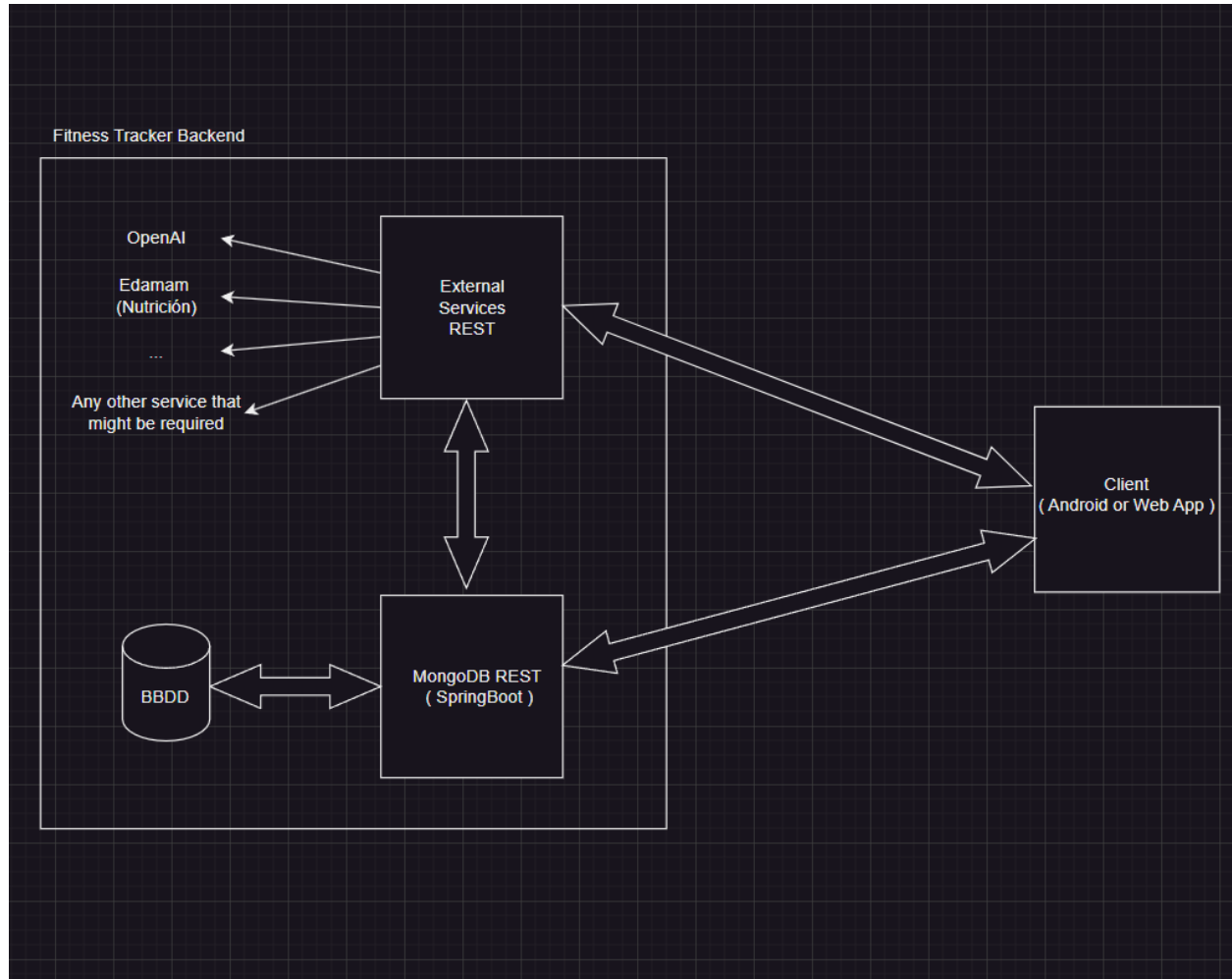


A continuación, se muestra un esquema del sistema:



El backend consta de una, por una parte, de una API REST desarrollada sobre el framework de ASP.NET Core, este es un framework que facilita la creación de un sistema a modo de pasarela entre nuestro backend y servicios externos como el de nutrición y el de OpenAI. ASP.NET Core es un framework con que es sencillo tener una API REST robusta y escalable.

Por otra parte, tenemos la API REST que gestiona la lógica de negocio de los clientes, así como almacenar datos relativos a estos, incluimos aquí las dietas, registros del reloj inteligente, entre otros. Esta API es la que tiene conexión directa con la base de datos y se desarrolla en Java utilizando el framework de Spring Boot. A continuación, un esquema ilustrativo.

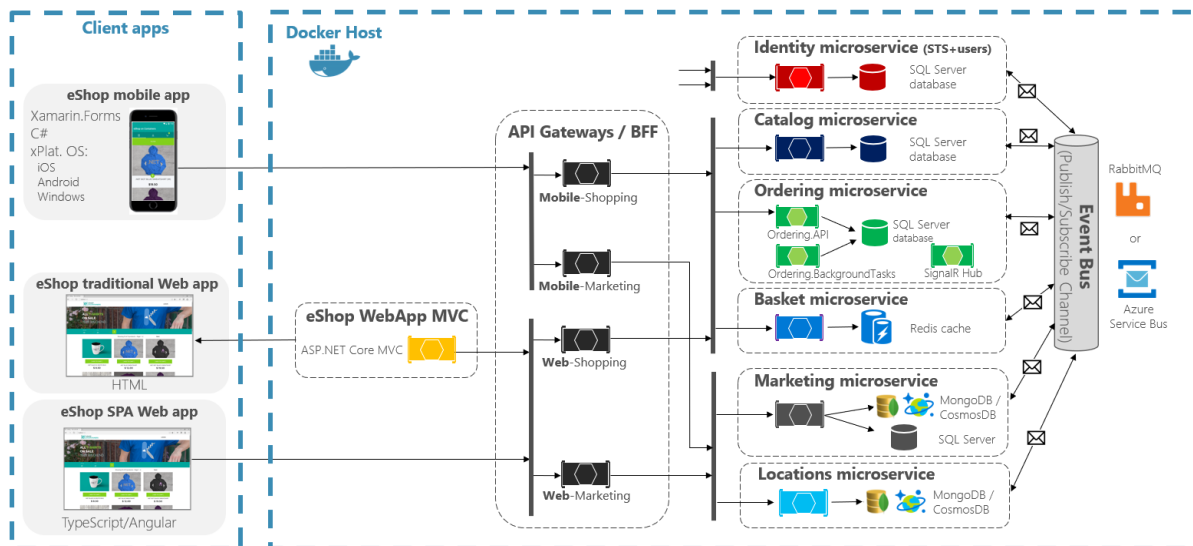


La razón detrás de esta arquitectura yace en que queremos diferenciar los servicios que son propiamente nuestros de los que son dependientes de servicios externos. MongoDB REST gestiona, como se ha mencionado anteriormente, la lógica y negocio relacionados con la manipulación de datos relativos a nuestros clientes. Paralelo a ello, External Services es un módulo que ofrece nuestro backend para ciertas funcionalidades que requieren nuestros clientes pero que no son servicios completamente dependientes de nosotros.

## API Gateway REST External Services

Una API Gateway es un componente esencial en arquitecturas de microservicios y sistemas distribuidos. Actúa como un punto de entrada para todas las solicitudes de clientes que acceden a múltiples servicios en la infraestructura. En esencia, proporciona una capa de abstracción entre los clientes y los microservicios subyacentes, simplificando la gestión de las solicitudes, mejorando la seguridad, y permitiendo funcionalidades adicionales como la autenticación, autorización, monitoreo y enrutamiento. En lugar de exponer cada servicio individualmente a través de su propia dirección IP y puerto, se expone una sola dirección IP y puerto para el API Gateway, que enruta las solicitudes a los servicios apropiados.

**eShopOnContainers reference application**  
(Development environment architecture)



Las funciones principales de una API Gateway incluyen el enrutamiento de solicitudes, autenticación y autorización, seguridad, transformación de datos, y monitoreo y análisis. Dirige las solicitudes de los clientes a los servicios correspondientes en función de la URL, el método HTTP u otros criterios. Gestiona la autenticación de los clientes y autoriza el acceso a los servicios según las políticas de seguridad establecidas. Proporciona una capa de seguridad adicional al actuar como un punto de entrada único a la infraestructura, protegiendo los servicios subyacentes de exposiciones no deseadas. Realiza transformaciones en los datos de solicitud y respuesta según sea necesario, para adaptarlos a los formatos esperados por los clientes o servicios. Recopila métricas y registra datos de tráfico para realizar análisis de rendimiento, seguimiento de errores y tomar decisiones informadas sobre la escalabilidad y la optimización de la infraestructura.

Fitness Tracker consta de una API Gateway implementada con Ocelot, una librería de terceros en .NET para integrar este tipo de sistemas en backend.

Ocelot es una herramienta de enrutamiento y API gateway de código abierto para .NET que permite a los desarrolladores crear gateways de API personalizados.

Con Ocelot, podemos implementar características como enrutamiento, autenticación, autorización, control de velocidad, balanceo de carga, almacenamiento en caché, transformación de respuestas, orquestación, monitoreo y más, de manera simple y eficiente, lo que mejora la eficiencia del despliegue y la escalabilidad de la aplicación.

Ocelot se utiliza para proporcionar un único punto de entrada central para las solicitudes a múltiples microservicios. Utiliza un archivo de configuración para definir las rutas de enrutamiento y las políticas de seguridad.

La arquitectura de microservicios ha ganado una gran popularidad en los últimos años, principalmente debido a que mejora la escalabilidad, flexibilidad y rendimiento del flujo de nuestro sistema.

Dado que las aplicaciones basadas en microservicios comprenden varios servicios diferentes, a menudo necesitamos una interfaz común o gateway para llamar a estos servicios para que podamos definir y gestionar todas las peticiones en un solo punto de entrada, en lugar de replicarlas en todos los servicios posteriores.

Aquí es precisamente donde entra en juego un API gateway, proporcionando un único punto de entrada para dirigir el tráfico a varios microservicios y un lugar central para implementar seguridad, monitoreo, entre otros aspectos.

Sin un gateway, tendríamos que implementar estas funcionalidades en cada uno de los servicios, lo que sería una tarea desalentadora.

## **Sistema inteligencia artificial**

El sistema de inteligencia artificial es un módulo tanto para la aplicación Web como el cliente Android que sirve de asistente virtual para el usuario. Permite al usuario realizar consultas a la hora de construir rutinas de ejercicios diarias o semanales.

Empezando por la inteligencia artificial que se utiliza en este módulo. Se va a utilizar el modelo de lenguaje GPT 4 Turbo con Visión, la versión de GPT capaz de aceptar imágenes y trabajar sobre éstas. Es una versión mejorada de GPT 3.5 Turbo, con la mejora de que puede procesar y generar imágenes, juntamente con la capacidad de poder procesar aún más palabras con más fluidez y precisión. Open AI expone una serie de endpoints a través de los cuales se puede realizar peticiones a estos modelos. Para ello es necesario primero una clave API que se puede generar desde su página.

Este servicio no es gratuito y está sujeto a unos límites de uso. Los límites de uso de la API son restricciones cruciales que la plataforma impone para regular el acceso y garantizar una experiencia equitativa para todos los usuarios. Estos límites, también conocidos como "Rate Limits", definen la cantidad máxima de veces que un usuario o cliente puede acceder a los servicios dentro de un período de tiempo específico.

Las razones principales para imponer estas restricciones están recogidas en la guía de desarrollo y son:

**Protección contra el Abuso:** La API se protege contra posibles abusos o malos usos que podrían sobrecargar los servidores o interrumpir el servicio. Esto se logra limitando la cantidad de solicitudes que un usuario puede realizar en un período de tiempo dado.

**Equidad de acceso:** los límites de uso garantizan que todos los usuarios tengan acceso justo a nuestros servicios. Al limitar el número de solicitudes que un usuario puede hacer, evitamos que un individuo o entidad monopolice los recursos disponibles, lo que podría ralentizar la experiencia para otros usuarios.

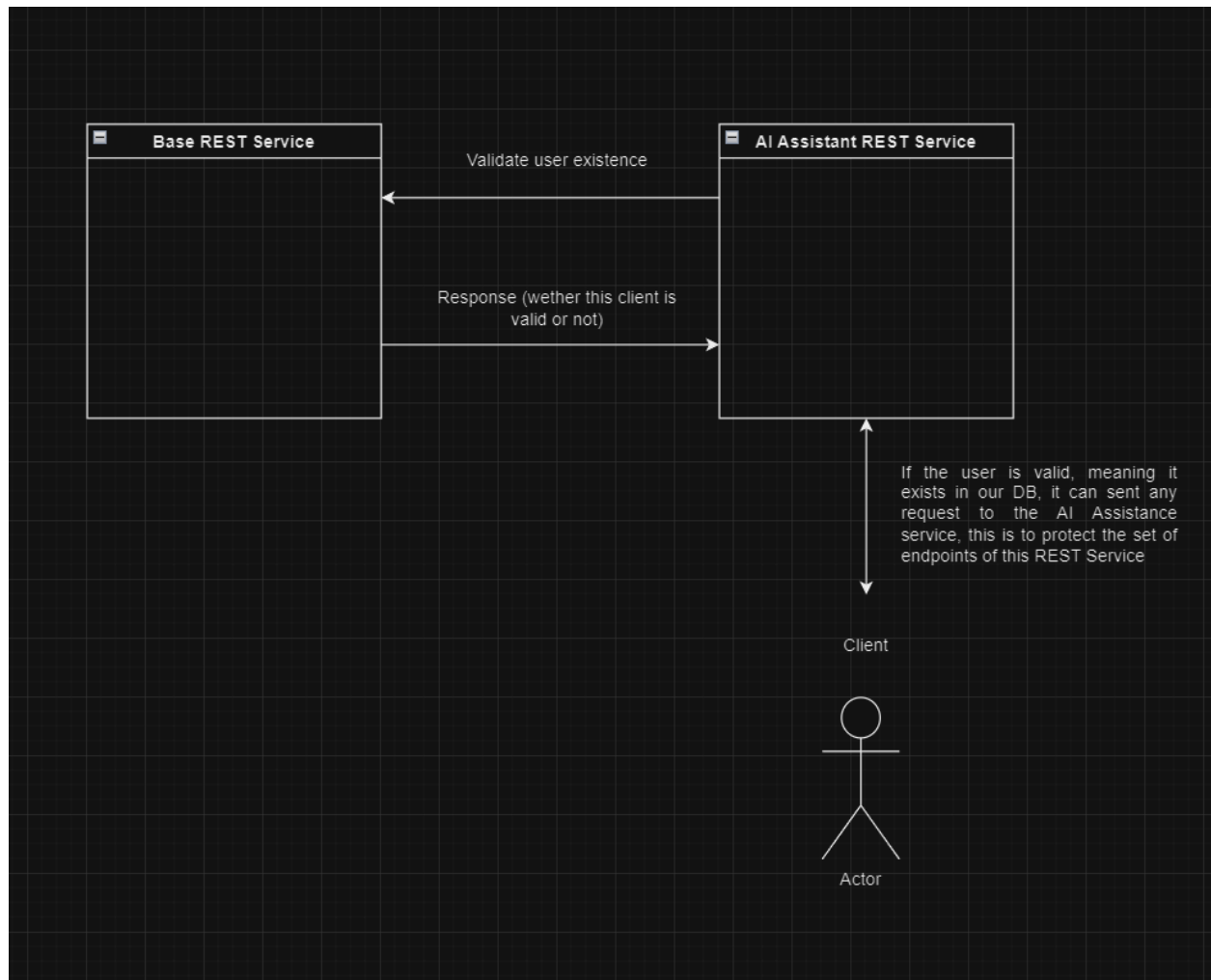
**Gestión de la carga de infraestructura:** Los límites de uso ayudan a administrar la carga total en nuestra infraestructura. Al controlar la cantidad de solicitudes que recibimos, podemos garantizar un rendimiento estable y consistente de nuestros servidores, incluso durante períodos de alta demanda.

Los límites de uso se expresan en diferentes métricas, que incluyen solicitudes por minuto (RPM o Request Per Minute), solicitudes por día (RPD o Request Per Day), tokens por minuto (TPM o Token Per Minute), tokens por día (TPD o Tokens Per Day) e imágenes por minuto (IPM o Images Per Minute). Estos límites se aplican según lo que ocurra primero en cada caso.

Es importante destacar que los límites de la API se definen tanto a nivel de organización como a nivel de proyecto, no a nivel de usuario individual. Además, los límites pueden variar según el modelo específico que se esté utilizando en la API.

La API REST que se comunicará con los servicios de OpenAI juntamente con los servicios de autenticación de usuarios se va a realizar con el lenguaje de programación C# usando ASP.NET Core, un framework que facilita mucho la creación de servicios REST.

Para realizar peticiones a la API de OpenAI se va a utilizar la librería de .NET OpenAI-API-dotnet. Una librería de código abierto que facilita la tarea de realizar peticiones contra la API de OpenAI, agilizando el trabajo de procesar las respuestas recibidas de los endpoints de Open AI en clases e interfaces más sencillas de usar.



El sistema de inteligencia artificial tendrá dos módulos: uno de asistente y otro como generador de dietas personalizadas.

Para generar las dietas se ingresarán los siguientes parámetros en un formulario a través de la aplicación web:

- Edad: Campo de texto.
- Género: Un menú desplegable que consta de Masculino o Femenino.
- Peso: Campo de texto (en kilogramos).
- Altura: Campo de texto (en centímetros)
- Nivel de Actividad Física: Menú desplegable que consta de sedentario, Moderado, Activo, muy Activo.
- Objetivo Principal: Menú desplegable que consta de Perder Peso, Mantener Peso, Ganar Masa Muscular, Mejorar Salud.

- Restricciones Alimenticias: [Casillas de verificación: Vegetariano / Vegano / Sin Gluten / Sin Lácteos / Sin Frutos Secos / Otro]
- Preferencias de Alimentos: Especificación de alimentos preferidos o no deseados, aquí el usuario podrá seleccionarlos, el backend enviará información nutricional recogida de Edamam de la cual se podrá seleccionar comidas ya con la información nutricional completa.
- Habilidad en la Cocina: Menú desplegable: Principiante, Intermedio, Avanzado, si el usuario quiere restringirse a comidas fáciles de realizar.
- Notas o Comentarios Adicionales: Área de texto para que el usuario pueda agregar cualquier información adicional relevante.

## **Arquitectura de la base de datos**

### **Dietas**

Una dieta se puede definir como el conjunto de alimentos y bebidas que una persona consume regularmente para mantener su salud y nutrición. Sin embargo, en nuestra aplicación consideramos la dieta como un plan estructurado y personalizado de ingesta de alimentos diseñado con objetivos específicos, como perder peso, ganar masa muscular, mejorar la salud cardiovascular, entre otros aspectos.

Para cada usuario se mantiene constancia de las dietas que lleva a lo largo del tiempo. Un usuario puede tener varias dietas a lo largo del tiempo en donde puede incluir todo tipo de comidas, las cuales incluso pueden repetirse entre dietas.



Las dietas se generan con la ayuda del módulo de External Services, a través del soporte de la inteligencia artificial. Luego esta se almacena en MongoDB REST que es nuestra API REST que se encarga de manipular todos los datos e información relativa al cliente.

Para generar una dieta, el usuario deberá cumplir un formulario similar al siguiente:

## **Ejercicios físicos**

El módulo de Ejercicios Físicos de nuestra aplicación complementa el aspecto dietético del plan de salud y bienestar del usuario. Este módulo proporciona un enfoque integral para ayudar a los usuarios a alcanzar sus objetivos de fitness y mejorar su calidad de vida.

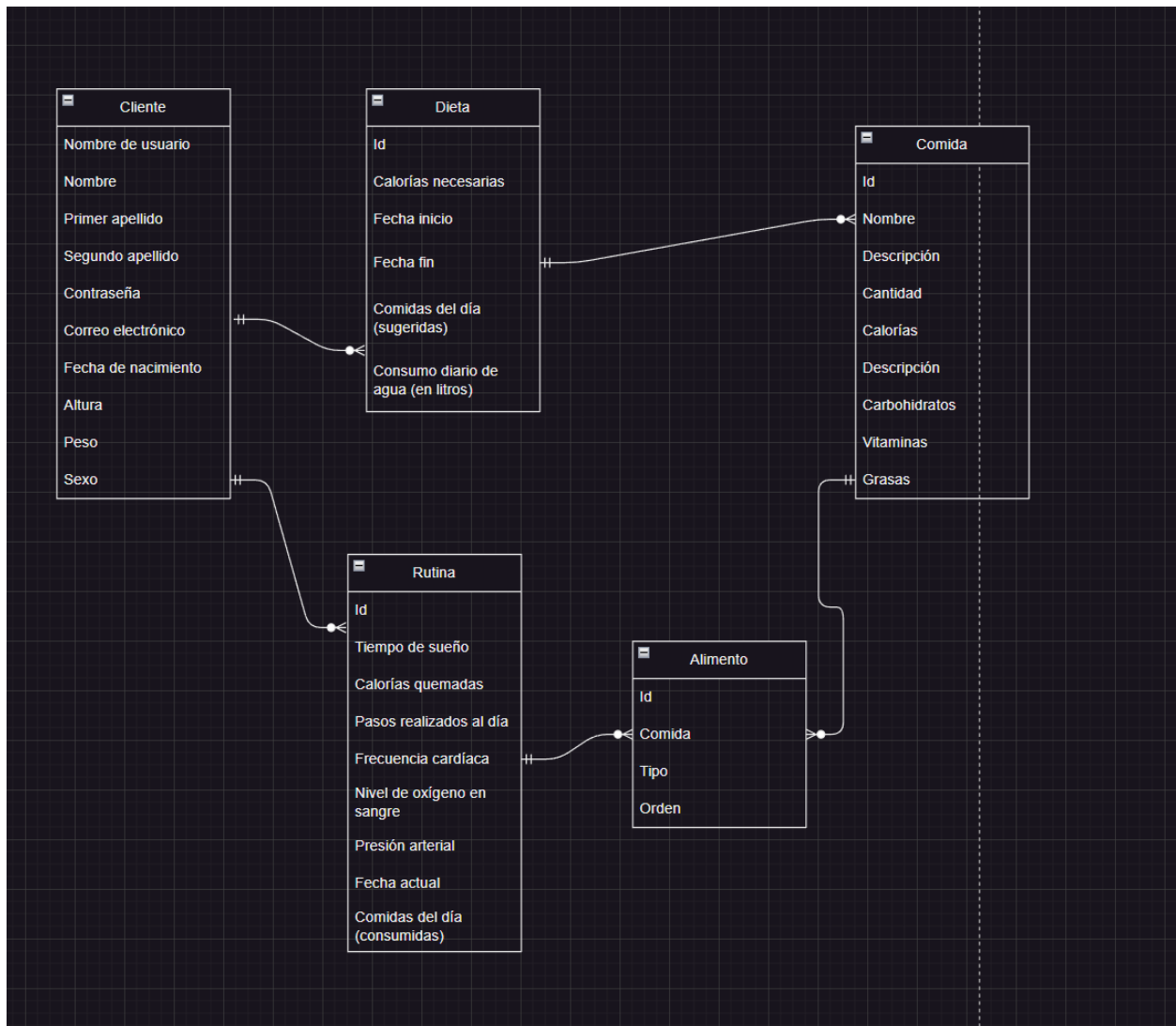
Este módulo permite a los usuarios registrar sus sesiones de ejercicio, incluyendo el tipo de actividad, la duración y la intensidad. También muestra a los usuarios un resumen de su actividad física a lo largo del tiempo, lo que les permite visualizar su progreso y establecer metas alcanzables.

## **Consejos de alimentación y ejercicio**

El módulo de Consejos de Alimentación y ejercicio proporciona a los usuarios orientación y apoyo adicional para ayudarles a adoptar hábitos de vida más saludables.

Este módulo ofrece consejos específicos de alimentación adaptados a las necesidades y objetivos de cada usuario, teniendo en cuenta factores como edad, género, nivel de actividad y preferencias alimenticias. Además, proporciona ideas y recetas de comidas nutritivas y equilibradas, ayudando a los usuarios a diversificar su dieta y disfrutar de opciones más saludables. Facilita la planificación de comidas diarias de acuerdo con los objetivos de salud y las preferencias personales de los usuarios, y proporciona recomendaciones de ejercicios específicos y estrategias de entrenamiento adaptadas a sus objetivos de fitness.

A continuación, se muestra un esquema del modelo de datos de nuestra base de datos.



En la arquitectura de la base de datos de nuestra aplicación, hemos identificado cuatro modelos principales: Cliente, Dieta, Ejercicio físico (Rutina) y Comida. Cada uno de estos modelos tiene su propio propósito y están interrelacionados de acuerdo con las funcionalidades de nuestra aplicación.

## Modelo de Cliente

El modelo de Cliente representa a los usuarios de nuestra aplicación, cada uno de los cuales tiene una cuenta registrada en nuestra base de datos, la cuenta mantiene constancia de los datos personales del cliente. A continuación se listan los datos personales del cliente que se consideran relevantes para la aplicación:

- **Nombre de usuario:** Campo opcional con que el usuario puede indicar cómo quiere ser dirigido en la aplicación, si no se especifica, se dirige a él por su nombre.
- **Nombre:** Nombres del cliente.
- **Primer apellido:** Primer apellido del cliente.
- **Segundo apellido:** Segundo apellido del cliente, no obligatorio en caso de que el cliente no lo tenga.
- **Contraseña:** Contraseña que se guarda encriptada en nuestra base de datos.
- **Correo electrónico:** Correo electrónico del cliente.
- **Fecha de nacimiento:** Fecha de nacimiento del cliente, con que se puede deducir la edad donde es necesario.
- **Altura:** Altura en centímetros del cliente.
- **Peso:** Peso en kilogramos del cliente.
- **Sexo:** Sexo del cliente. Donde puede especificar mujer u hombre. Este campo hace referencia al sexo biológico ya que se considera un factor importante para luego poder determinar las dietas.

## Modelo de Dieta

El modelo de Dietas representa los planes de dieta disponibles en nuestra aplicación. Un cliente podrá tener varias dietas a lo largo del tiempo, tantas como quiera y, obviamente podemos almacenar en nuestro servidor. Las dietas son planes de comida semanales que se ajustan a las necesidades nutricionales y alimenticias que quiere llevar el cliente en un intervalo de tiempo determinado, por ende son recomendaciones, y no necesariamente lo que el usuario ha

consumido en concreto. La dieta una vez creada se puede modificar. A continuación se listan las propiedades de una dieta:

- **Id:** Identificador único para cada comida, generado automáticamente por el sistema.
- **Calorías objetivo:** Cantidad de calorías a quemar acorde a la dieta, medidas en Kcal o Julios.
- **Fecha de inicio:** Fecha de inicio de la dieta.
- **Fecha de fin:** Fecha de fin de la dieta.
- **Comidas del día (sugeridas):** Comidas sugeridas para consumir en esta dieta.
- **Consumo de agua diario:** Consumo de agua diario mínimo recomendado.

## Modelo de Comida

El modelo de Comidas representa los alimentos específicos o recetas específicas que forman parte de cada dieta o de una rutina en concreto, es decir, se diferencia entre las comidas que se sugieren para seguir una dieta en concreto y las que el usuario ha consumido. A continuación se listan sus propiedades:

- **Id:** Identificador único para cada comida, generado automáticamente por el sistema.
- **Nombre:** Nombre de la comida (puede ser una receta completa).
- **Descripción:** Descripción de la comida.
- **Calorías:** Cantidad de calorías proporcionadas por esta comida.
- **Carbohidratos:** Carbohidratos proporcionados por esta comida.
- **Vitaminas:** Tipo de vitaminas proporcionadas por esta comida.
- **Grasas:** Valor en grasas proporcionado por esta dieta.

Cada alimento puede tener atributos como nombre, descripción, valor nutricional, etcétera. Los alimentos están relacionados con las dietas, ya que cada alimento específico solo se relaciona con una dieta en particular. Es el cliente quien puede seleccionar los alimentos dentro de una dieta, en función de sus necesidades y preferencias dietéticas.

## Modelo de Ejercicio Físico (Rutina)

El modelo de rutina representa la información recopilada de dispositivos del seguimiento de la actividad física del cliente, incluyendo datos recopilados de relojes inteligentes y las comidas consumidas al día.

- **Id:** Identificador único para rutinas, generado y asignado automáticamente por el sistema gestor de la base de datos.
- **Tiempo de sueño:** Indica el tiempo de sueño invertido para inhibir la somnolencia durante el día siguiente, representado en minutos.
- **Calorías quemadas:** Cantidad de calorías quemadas por ejercicio físico realizado durante el día, medido en kcal o Julios (J)
- **Pasos realizados al día:** Número de pasos realizados al día.
- **Frecuencia cardíaca:** Frecuencia de las pulsaciones del corazón por minuto. Medido en BPM o *Beats Per Minute*.
- **Nivel de oxígeno en sangre:** Porcentaje de oxígeno en sangre.
- **Presión arterial:** Presión con que circula la sangre por nuestro organismo. Medido en milímetros de mercurio, ya que resulta más informativo de cara al cliente.
- **Comidas del día:** Alimentos consumidos al día, ver el modelo de Alimento.

## Modelo de Alimento

Este modelo nace como consecuencia de que el usuario no necesariamente va a ceñirse a las restricciones de la dieta y acabará consumiendo algún alimento del día. Este modelo mantiene constancia de ello y siempre es parte de una rutina, no existe por sí mismo, es decir, tiene una relación de agregación con el modelo de rutina.

Los alimentos se obtienen del servicio externo de FT - Alimentos. Estos luego se serializan a nuestra base de datos cuando se genera la dieta ya que la dieta se serializa a la base de datos y esta debe mantener constancia de los alimentos que la componen. En esencia esto implica que tenemos datos duplicados y parte de ellos en una base de datos totalmente ajena a nosotros. Esto es así porque esto nos libera, como desarrolladores, de la carga de mantener actualizada una base de datos con información nutricional lo suficientemente extensa como para asistir las necesidades de un cliente potencial.

A continuación, se listan sus atributos:

- **Id:** Identificador único generado automáticamente por la base de datos para identificar los diferentes alimentos.
- **Comida:** Comida consumida en una fecha específica.
- **Tipo de comida:** Especifica el tipo de comida, que puede ser el desayuno, almuerzo, merienda o la cena.
- **Orden:** Especifica el orden de plato, que puede ser primer plato, segundo plato, tercer plato.

## Interrelaciones

Cliente y Dietas: Un cliente puede tener una relación de uno (obligatorio) a muchos (opcional) con las dietas, lo que significa que un cliente puede tener asociadas varias dietas a lo largo del tiempo, los clientes pueden obtener dietas similares, pero estas son únicas por cada uno de ellos.

Dietas y Comidas: Existe una relación de muchos (opcional) a muchos (opcional) entre las dietas y los alimentos, lo que significa que cada alimento específico está asociado con varias dietas, al igual que varias dietas pueden tener asociados alimentos varios alimentos.

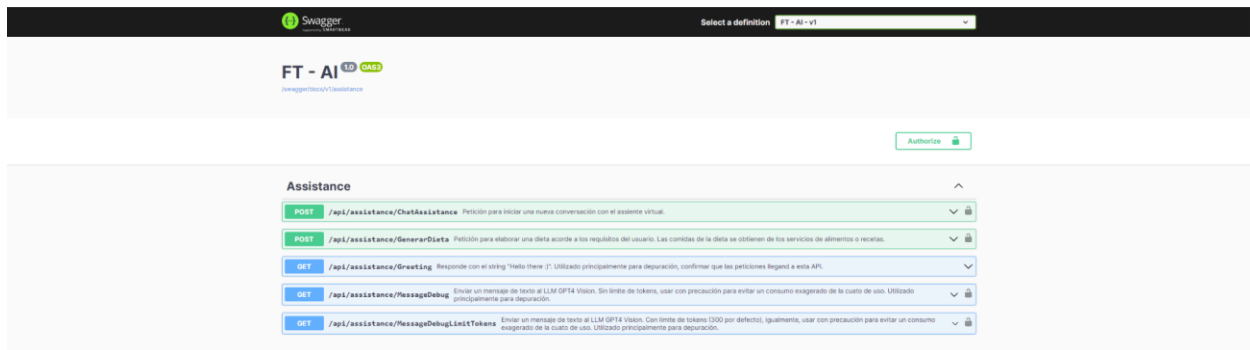
Cliente y Rutina: Un cliente tiene una relación de uno a muchos con la rutina, lo que significa que un cliente puede tener múltiples registros de datos diarios relativos a su actividad física, consumo de alimentos, entre otros aspectos.

Alimento y Comida: Como se ha explicado anteriormente, los alimentos son las comidas que el usuario ha consumido, no necesariamente los mismos que sugiere la dieta para un día en concreto. La relación es de muchos (opcional) a muchos (opcional).

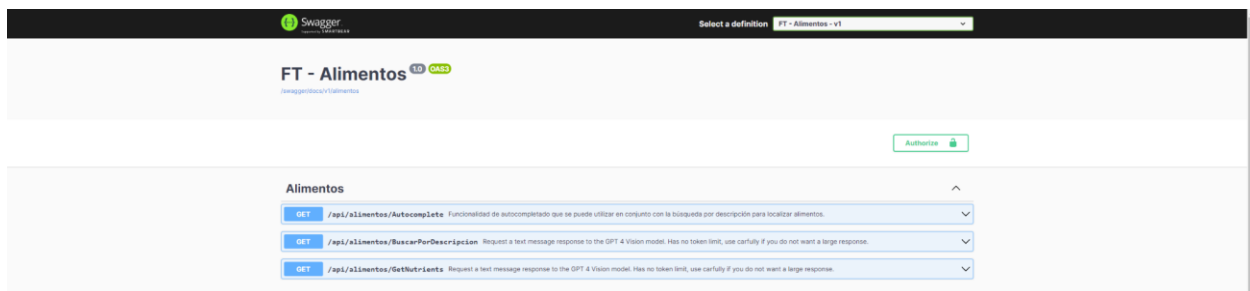
# Código fuente comentado

Como se ha mencionado anteriormente nuestro sistema consta de un backend desarrollado sobre .NET y uno de los servicios desarrollado sobre Spring Boot con Java. Los métodos que nuestra API REST expone son los siguientes:

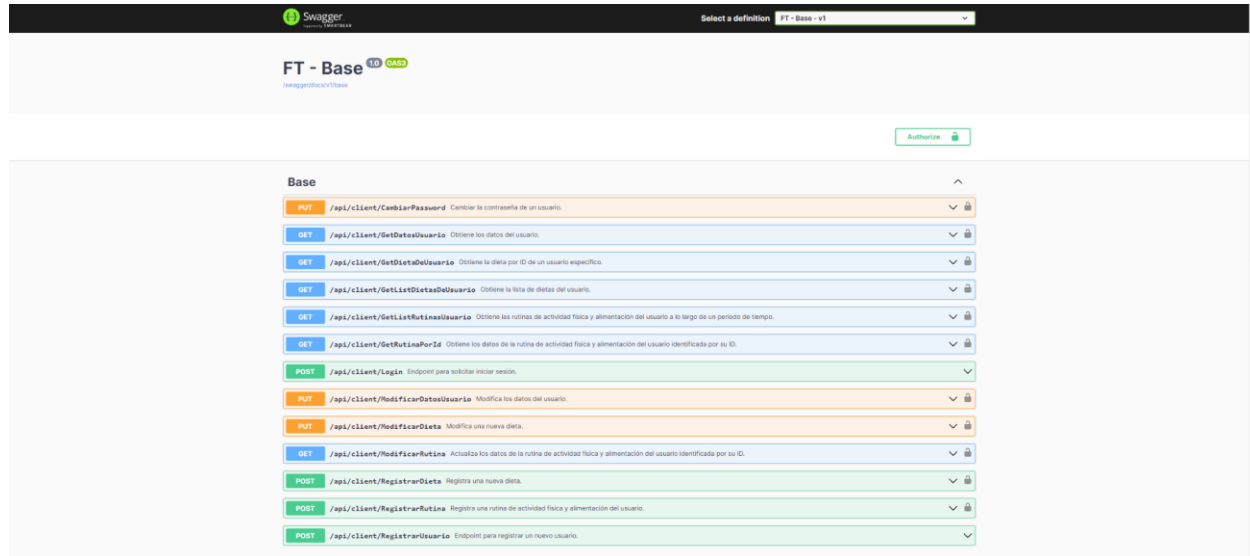
## Endpoints de inteligencia artificial.



## Endpoints de información nutricional



## Endpoints relativos al usuario



Base	
PUT	/api/client/CambiarPassword Cambiar la contraseña de un usuario.
GET	/api/client/GetDatosUsuario Obtener los datos del usuario.
GET	/api/client/GetDietasUsuario Obtener la dieta por ID de un usuario específico.
GET	/api/client/GetListaDietasUsuario Obtener la lista de dietas del usuario.
GET	/api/client/GetListaRutinasUsuario Obtener las rutinas de actividad física y alimentación del usuario a lo largo de un periodo de tiempo.
GET	/api/client/GetRutinaPorId Obtener los datos de la rutina de actividad física y alimentación del usuario identificado por su ID.
POST	/api/client/Login Endpoint para solicitar iniciar sesión.
PUT	/api/client/ModificarDatosUsuario Modifica los datos del usuario.
PUT	/api/client/ModificarDieta Modifica una nueva dieta.
GET	/api/client/ModificarRutina Actualiza los datos de la rutina de actividad física y alimentación del usuario identificado por su ID.
POST	/api/client/RegistrarDieta Registra una nueva dieta.
POST	/api/client/RegistrarRutina Registra una rutina de actividad física y alimentación del usuario.
POST	/api/client/RegistrarUsuario Endpoint para registrar un nuevo usuario.

La API REST de ASP.NET sigue la misma estructura en todos los endpoints se recibe un objeto JSON de entrada que se valida, si es válido se redirecciona la petición al servicio y si no se devuelve un Bad Request a la petición. En los servicios se realiza la lógica de negocio, así como acceder a bases de datos o servicios externos si fuera necesario.

## Ejemplo de un endpoint de backend:

```
/// Las comidas de la dieta se obtienen de los servicios de alimentos o recetas.
/// </summary>
/// <param name="model">Requisitos de la dieta</param>
/// <returns>Respuesta del modelo de vista. Ver: <see cref="ResponseGenerarDietaVM"/>.</returns>
[HttpPost("GenerarDieta")]
[Authorize] // authorize
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(ResponseGenerarDietaVM))]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
0 references
public async Task<ActionResult<ResponseGenerarDietaVM>> GenerarDieta([FromBody] RequestGenerarDieta model)
{
    var result = _validatorRequestDieta.Validate(model);
    if (result == null || !result.IsValid)
    {
        return BadRequest(result?.Errors);
    }

    var res = await _assistanceService.GenerarDieta(model);

    return Ok(res);
}
```



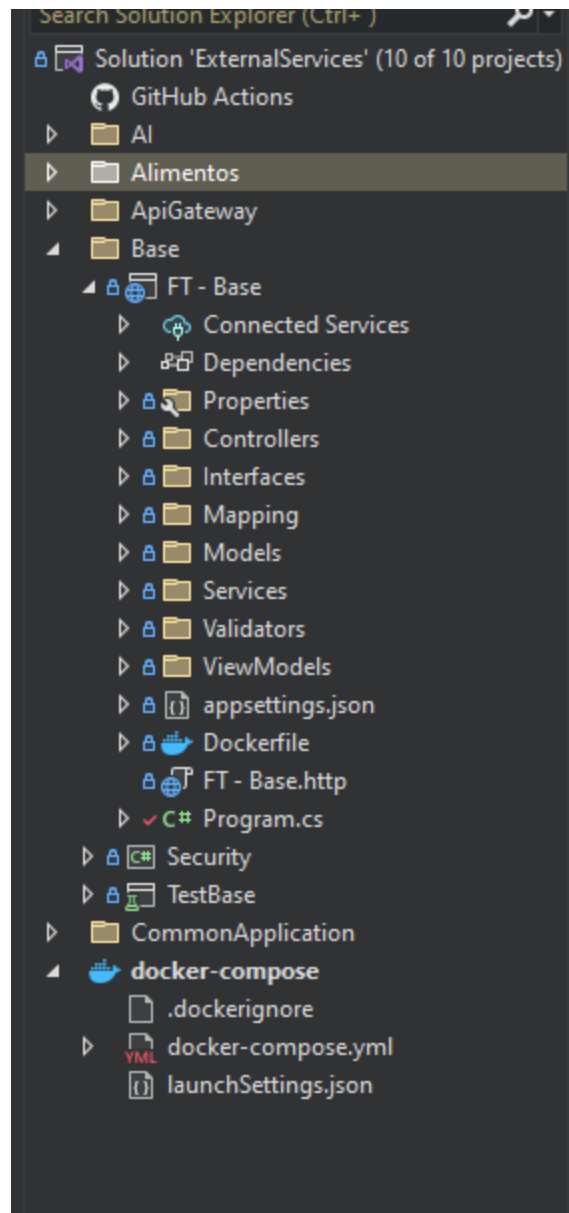
Esta estructura se repite a lo largo del resto de endpoints de nuestro backend.

## Estructura de carpetas

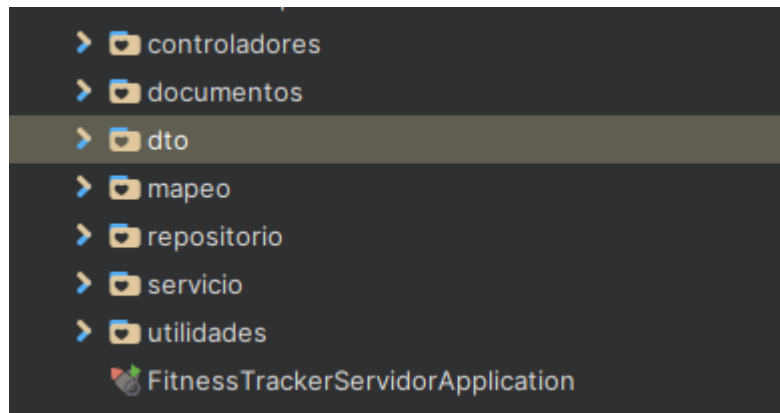
Nuestros servicios siguen la misma estructura. Tenemos varias carpetas, cada una de ellas con una funcionalidad determinada. Ciertas carpetas como la de Properties las crea el editor al crear un proyecto nuevo de ASP.NET y es allí donde se guarda información relativa al proyecto como los puertos que utiliza, protocolos de comunicación, entre otros aspectos. En la carpeta Controllers tenemos los controladores. En la carpeta Interfaces tenemos las interfaces, definiciones de funciones de uso común entre diferentes clases y que también facilitan la posibilidad de inyección de dependencias. En la carpeta Mapping tenemos los perfiles de mapeo, que son clases que nos permiten construir POJOs a partir de otros, esto es especialmente útil a la hora de deserializar objetos JSON de servicios externos para poder trabajar con ellos en nuestro backend. En la carpeta Services tenemos los servicios, son bloques de código que llevan a cabo la lógica de negocio. En la carpeta Validators tenemos los validadores, que son clases que se encargan de validar los datos que recibe nuestro backend; y, finalmente, en la carpeta ViewModels tenemos los modelos de vista que son clases que encapsulan el conjunto de datos que nosotros queremos hacer visibles al cliente exterior.

Cada servicio consta de un dockerfile, ya que, por cuestión de despliegue, construimos las imágenes para poder generar contenedores de nuestros servicios y así poder desplegar nuestra aplicación en un servidor remoto.

Por otra parte, ciertos servicios constan de los llamados Class Library, que son librerías que contienen utilidades que no son comunes a todos los microservicios pero que tienen sentido ser extraídos a un módulo aparte, en la imagen se puede ver por ejemplo el módulo de Security, que es un módulo para gestionar el sistema de autenticación con JWT.



Por otra parte, tenemos una aplicación API REST desarrollada sobre Spring Boot que tiene la siguiente estructura de carpetas:



La estructura de carpetas de una aplicación Spring Boot con MongoDB está diseñada para organizar de manera clara y eficiente las distintas responsabilidades del código, facilitando su mantenibilidad y escalabilidad. En la carpeta "controladores" se encuentran las clases responsables de manejar las solicitudes HTTP entrantes y definir los endpoints de la API REST.

La carpeta "documentos" contiene las clases que representan las entidades o modelos que se almacenarán en la base de datos MongoDB. Estas clases están anotadas con las anotaciones de Spring Data MongoDB para mapear las colecciones de la base de datos a objetos Java, definiendo los campos que serán persistidos.

En la carpeta "dto" se agrupan las clases utilizadas para transferir datos entre las distintas capas de la aplicación y hacia o desde el cliente. Los DTO encapsulan los datos enviados en las solicitudes y respuestas de la API, proporcionando una forma segura y controlada de transferir información, además de incluir validaciones y transformaciones necesarias para la comunicación con el cliente.

La carpeta "mapeo" contiene clases encargadas de convertir entre entidades de dominio y DTOs. Estos mappers centralizan la lógica de transformación de datos, asegurando que la conversión entre diferentes representaciones sea manejada de manera consistente y separada del resto del código.

En la carpeta "repositorio" se encuentran las interfaces que extienden de Spring Data MongoDB Repository. Estas interfaces proporcionan métodos CRUD y consultas personalizadas para interactuar con la base de datos MongoDB, abstrayendo la lógica de acceso a datos y facilitando la realización de operaciones con la base de datos.

La carpeta "servicio" alberga las clases que implementan la lógica de negocio de la aplicación. Los servicios son responsables de procesar la lógica compleja, realizar validaciones y coordinar las operaciones entre diferentes partes de la aplicación. Los controladores llaman a los métodos de los servicios para cumplir con las solicitudes del cliente, manteniendo una clara separación entre la lógica de negocio y la de presentación.

Por último, la carpeta "utilidades" incluye clases y métodos que proporcionan funcionalidades auxiliares y utilitarias reutilizables en diferentes partes de la aplicación. Estas utilidades pueden incluir funciones de ayuda para manipulación de cadenas, fechas y manejo de excepciones, entre otros.

La estructura de los endpoints por lo general es la misma.

```
331 エキット
332 @PostMapping("@modificarrutina")
333 ResponseEntity<ResponseModificarRutina> modificarRutina(@RequestBody RequestModificarRutina model) {
334     ResponseModificarRutina responseData = ResponseModificarRutina.builder().build();
335     ResponseEntity<ResponseModificarRutina> response;
336
337     try {
338         Boolean result = usuarioServicio.modificarRutina(model);
339         responseData.setSuccess(true);
340
341         if (result) {
342             responseData.setModifiedAt(LocalDateTime.now());
343             responseData.setResponseDescription("Rutina modificada con éxito");
344         } else {
345             responseData.setSuccess(false);
346             responseData.setResponseDescription("No se pudo modificar la rutina. Esta es inválida o el usuario no existe.");
347         }
348
349         response = new ResponseEntity<>(responseData, HttpStatus.OK);
350     } catch (Exception e) {
351         responseData.setSuccess(false);
352         responseData.setResponseDescription(e.getMessage());
353         response = new ResponseEntity<>(responseData, HttpStatus.INTERNAL_SERVER_ERROR);
354     }
355
356     return response;
357 }
```

Los endpoints siempre reciben un objeto y devuelven un objeto. Esta aplicación sólo consta de un controlador porque está sumamente enfocada sobre el usuario.

Al recibir los datos estos se redirigen a los servicios, los cuales entre otras cosas validan que existan los objetos con los ids que se suministran. Se realiza control de erros mediante bloques try y catch en caso de haber una excepción, en cuyo caso se responde con un Internal Server Error.

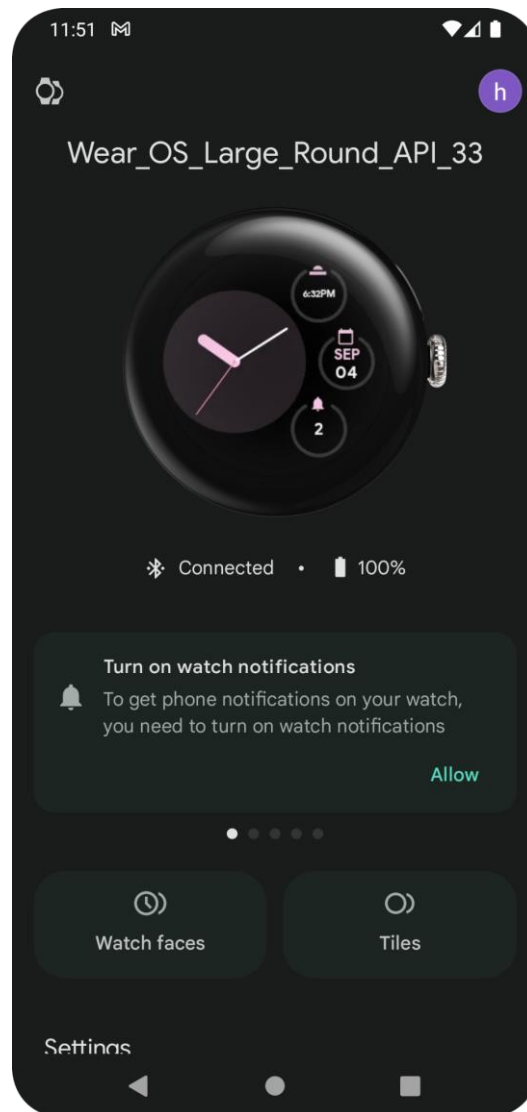
Si la excepción es de tipo `RuntimeException` se devuelve todavía un OK para facilitar el desarrollo con Retrofit en el entorno de Android.

# **Manual de configuración y funcionamiento de la aplicación.**

## **Aplicación Android**

La aplicación móvil Fitness Tracker es una herramienta diseñada para ayudar a los usuarios a monitorear y mejorar su estado físico. Esta guía proporciona instrucciones detalladas sobre cómo instalar, configurar y utilizar la aplicación para aprovechar al máximo sus funciones.

Antes de instalar la aplicación, asegúrese de que su dispositivo cumpla con los siguientes requisitos: sistema operativo Android 8.0 o superior, al menos 87 MB de espacio libre en el almacenamiento y una conexión a Internet activa para sincronizar datos y acceder a funcionalidades en línea. Es importante también tener el móvil sincronizado con el Wear OS, esto se realiza mediante conexión Bluetooth y debemos instalarnos para esto primero la aplicación de Watch, para ello vamos a la Play Store y la instalamos. Emparejados los dispositivos nos saldría una pantalla como la siguiente:



Para instalar la aplicación en un dispositivo Android, siga estos pasos. De forma preliminar se permitirá la descarga de los instaladores mediante un link a GitHub. Bajados los instaladores, podrá hacer clic sobre ellos y darle al botón de “Instalar”. Finalizada la instalación será cuestión de iniciar la aplicación tanto en el dispositivo Android como en el dispositivo del reloj inteligente.

Una vez instalada, abra la aplicación desde el menú de aplicaciones. Si es un nuevo usuario, seleccione "Registrarse" y complete el formulario con su nombre, correo electrónico y una contraseña segura. Si ya tiene una cuenta, seleccione "Iniciar sesión" e ingrese sus credenciales.

La pantalla principal muestra un resumen de su actividad diaria, incluyendo pasos, calorías quemadas, frecuencia cardíaca, requerimiento de agua diario en litros, metabolismo basal y el

IMC. Desde aquí puede acceder a diferentes secciones de la aplicación como Estadísticas, Dietas, Rutinas de Ejercicio y Cuenta.

En la sección de cuenta, puede actualizar su información personal como peso, sexo y altura.

## Aplicación web

La aplicación web Fitness Tracker es una herramienta integral diseñada para ayudar a los usuarios a monitorear y mejorar su estado físico y salud desde cualquier dispositivo con acceso a Internet. Este manual sirve de guía a través del proceso de instalación, configuración y uso de la aplicación web, asegurando que pueda aprovechar al máximo sus funcionalidades.

Asegúrese de que su navegador web esté actualizado. La aplicación es compatible con las versiones más recientes de Google Chrome, Mozilla Firefox y Microsoft Edge.

Abra su navegador preferido y visite la página oficial de la aplicación web Fitness Tracker en <https://fitness-tracker-blue.vercel.app/>.

Si es un nuevo usuario, haga clic en "Registrarse" y complete el formulario con su información personal. Si ya tiene una cuenta, haga clic en "Iniciar sesión" e ingrese sus credenciales.

Una vez iniciada la sesión, complete su perfil proporcionando información como peso, altura, edad en el apartado de perfil, este se puede modificar.

La aplicación web Fitness Tracker está diseñada para ser intuitiva y fácil de usar. Aquí se detalla el funcionamiento de sus principales características:

Al iniciar sesión, será dirigido al panel principal que muestra un resumen de su actividad diaria, información sobre su físico y la dieta que esté activa actualmente.

**Registro de Actividades:** En el menú lateral, seleccione "Actividades" para registrar nuevas actividades físicas. Complete los detalles como tipo de actividad, duración, distancia y calorías quemadas. Puede ver y editar sus registros en cualquier momento.

# Manual de usuario

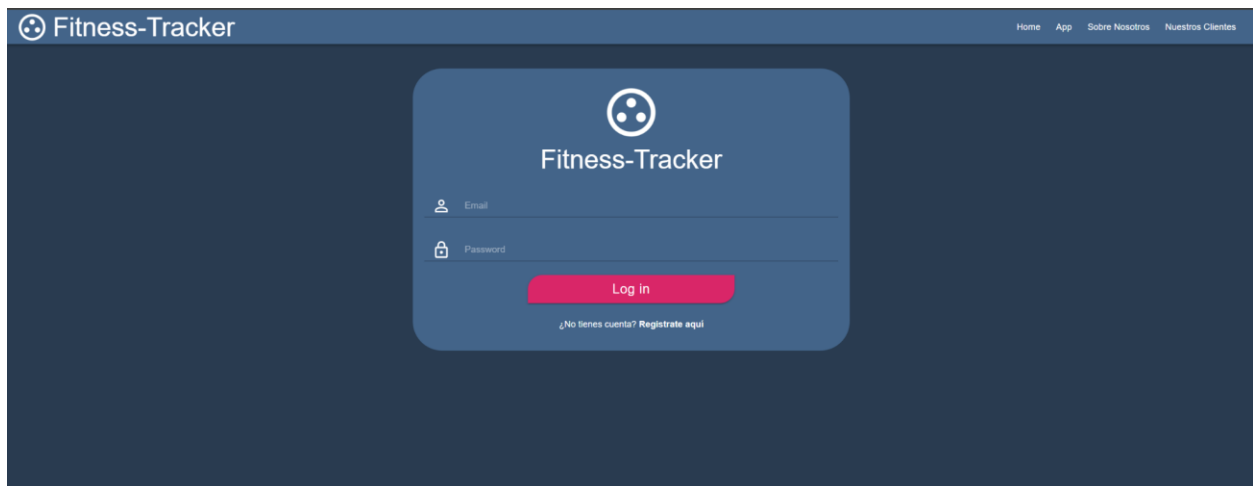
## Aplicación móvil

La aplicación móvil Fitness Tracker es una herramienta integral diseñada para ayudar a los usuarios a seguir y mejorar su estado físico y salud. Este manual de usuario sirve de guía a través de las diversas funcionalidades de la aplicación, desde la navegación básica hasta el registro y monitoreo de actividades físicas.

Al abrir la aplicación, será recibido con la pantalla principal que muestra un resumen de su actividad diaria. En la parte superior de la pantalla, encontrará una barra de navegación con accesos directos a las principales secciones de la aplicación, información del perfil, seguimiento físico diario y el seguimiento distribuido por fechas.

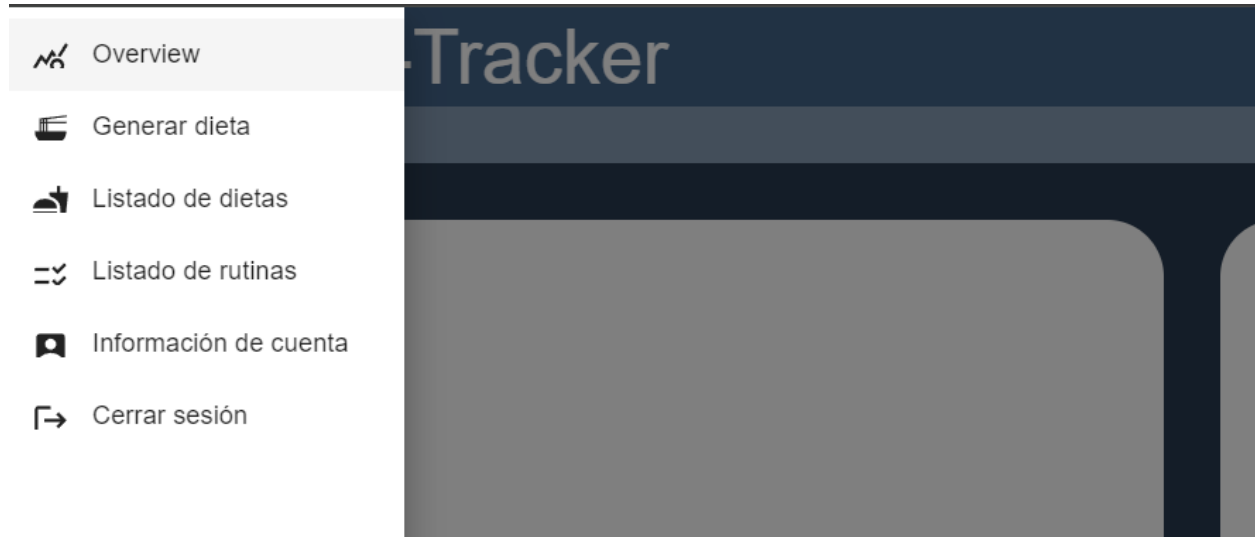
## Aplicación web

La aplicación web Fitness Tracker es una plataforma completa que le permite llevar un registro detallado de su actividad física, dieta y progreso de fitness. Este manual de usuario le guiará a través de las diversas funcionalidades de la aplicación, asegurando una experiencia de usuario eficiente y efectiva.

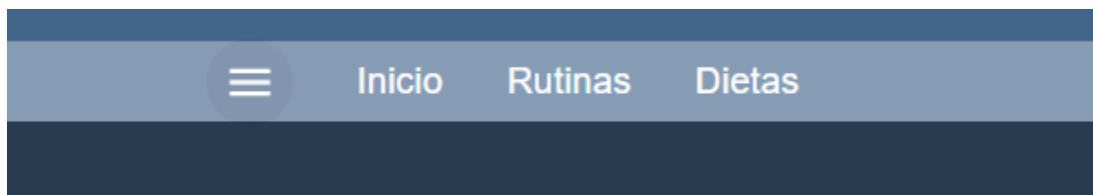




Al ingresar a la aplicación, se encontrará con el panel principal que proporciona un resumen rápido de su actividad diaria y detalles de su físico. La navegación principal se encuentra en el menú lateral izquierdo, desde donde puede acceder a las diferentes secciones de la aplicación:



- Overview: Muestra una vista general de su actividad diaria y progreso.
- Generar dieta: Permite interactuar con el módulo de inteligencia artificial para registrar nuevas dietas rellenando un formulario.
- Listado de dietas: Redirige a la pantalla del listado de dietas del usuario junto con la dieta activa actualmente.
- Listado de rutinas: Redirige a la pantalla con el listado de rutinas del usuario.
- Información de cuenta: Ofrece información del perfil del usuario.
- Cerrar sesión: Permite cerrar la sesión del usuario.
- Puede acceder a este sidebar a través de el botón de tres barras en la parte superior de la página:



De igual manera el botón de Inicio le redirige al Overview, el de Rutinas al Listado de las rutinas y el de Dietas al Listado de dietas.

# Plan de formación a los Usuarios de la Aplicación

El plan de formación a los usuarios de la aplicación Fitness Tracker está diseñado para asegurar que todos los usuarios puedan aprovechar al máximo las funcionalidades y beneficios que ofrece la aplicación.

Para garantizar un aprendizaje continuo, ofrecemos el correo [zanetty54@gmail.com](mailto:zanetty54@gmail.com) donde los usuarios pueden exponer sus preguntas.

## Bibliografía y fuentes de información

(s.f.). Obtenido de Dveloper Android:

<https://developer.android.com/training/wearables/data/data-layer>

*API Gateway*. (s.f.). Obtenido de <https://www.redhat.com/en/topics/api/what-does-an-api-gateway-do>

*AWS Academy*. (s.f.). Obtenido de <https://aws.amazon.com/training/awsacademy/>

*Cómo dockerizar una aplicación de Spring Boot*. (s.f.). Obtenido de Epam Anywhere:

<https://anywhere.epam.com/es/blog/como-dockerizar-una-aplicacion-de-spring-boot>

*Dieta y nutrición*. (s.f.). Obtenido de <https://mhanational.org/dieta-y-nutricion>

*Edamam*. (s.f.). Obtenido de <https://www.edamam.com/>

*Implement API Gateway*. (s.f.). Obtenido de <https://learn.microsoft.com/en-us/dotnet/architecture/microservices/multi-container-microservice-net-applications/implement-api-gateways-with-ocelot>

*Statista*. (2024). Obtenido de Ventas de relojes inteligentes a nivel mundial de 2016 a 2025: <https://es.statista.com/estadisticas/664393/prevision-de-las-ventas-mundiales-de-smartwatches/>

*Tutorial de Docker*. (2 de Octubre de 2021). Obtenido de Tutorial de Docker: <https://atareao.es/tutorial/docker/>