IES ALBARREGAS

TÉCNICO SUPERIOR EN DESARROLLO DE APLICACIONES WEB

Departamento de Informática

PROYECTO

FitMind

Manual Técnico

Autora: Elena Ardila Delgado

Curso Académico: 2025-2026

Índice

1. Introducción **4**

2. Arquitectura de la aplicación **5**

2.1 Frontend 6

2.1.2 Tecnologías usadas 6

2.1.2 Justificación de la elección: 7

2.1.3 Entorno de desarrollo 8

2.2 Backend 8

2.2.1 Tecnologías usadas 8

2.2.2 Justificación 9

2.2.3 Entorno de desarrollo 9

3. Documentación Técnica **10**

3.1 Análisis 10

3.1.1 Roles principales 10

3.1.2 Requisitos funcionales 10

3.1.3 Requisitos no funcionales 11

3.2 Desarrollo 12

3.2.1 Base de Datos 12

3.2.2 Justificación del diseño 14

3.2.3 Diagrama de casos de uso 15

3.3 Pruebas realizadas 18

4. Proceso de despliegue **20**

4.1 Software necesario 20

4.2 Pasos para desplegar la aplicación 20

4.2.1 Configurar Supabase 20

4.2.2 Configurar el proyecto local 20

4.2.3 Despliegue en Vercel 21

5. Propuestas de mejoras **22**

6. Bibliografía **23**

1. Introducción

FitMind es una aplicación web desarrollada como proyecto final del ciclo de Técnico Superior en Desarrollo de Aplicaciones Web. Su finalidad es ofrecer a cada usuario un entorno unificado donde pueda gestionar su entrenamiento físico y su alimentación mediante rutinas y dietas semanales personalizadas, con una experiencia de uso sencilla y accesible desde cualquier dispositivo con navegador moderno.

La aplicación sigue una arquitectura front-only: toda la lógica de presentación y de interacción reside en el cliente (React + Vite + TailwindCSS) y los servicios de autenticación, base de datos y almacenamiento de ficheros se delegan en **Supabase**, que actúa como backend-as-a-service (BaaS). De este modo se simplifica la infraestructura, se reduce el número de servicios a desplegar y se facilita el mantenimiento del proyecto.

A nivel funcional, FitMind permite:

- Registro, inicio, cierre de sesión y recuperación de contraseñas de usuarios mediante Supabase Auth.
- Gestión de un perfil con datos básicos (nombre, correo), datos físicos (sexo, edad, altura, peso, objetivo) y avatar.
- Visualización de un panel de control tipo dashboard con métricas de seguimiento (peso, IMC, calorías objetivo) y acceso rápido a las últimas rutinas y dietas.
- Gestión CRUD de los planes de dieta y los entrenamientos por semanas, con posibilidad de consultar el historial.
- Un panel de administración donde el rol admin puede listar usuarios, filtrarlos (por nombre, fecha de alta, sexo, peso y altura) y activar/desactivar cuentas.

La motivación principal del proyecto es cubrir la necesidad de disponer de una herramienta ligera que ayude a los usuarios a organizar su estilo de vida saludable.

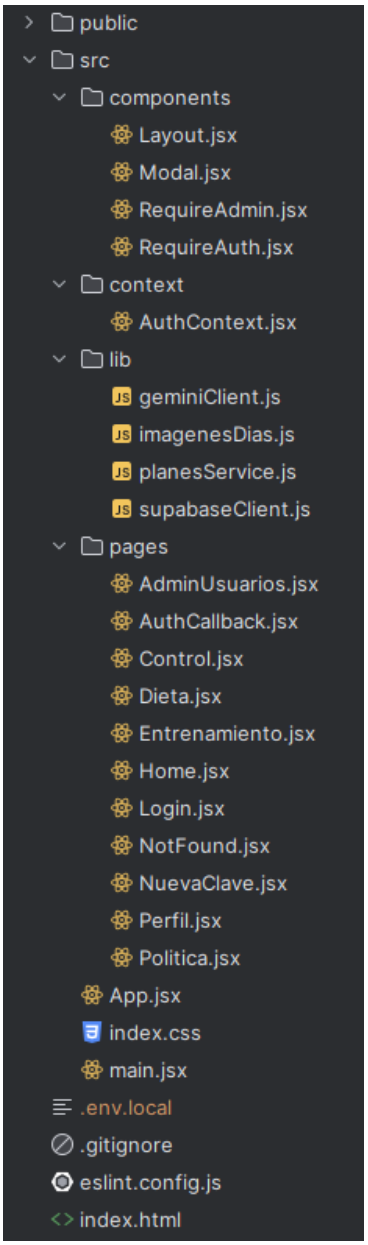
2. Arquitectura de la aplicación

La arquitectura de FitMind está basada en una **Single Page Application (SPA)** desarrollada con React. El proyecto se organiza en las siguientes capas:

- **Capa de presentación (Frontend)**: componentes React, enrutado, estilos con TailwindCSS y gestión de estado de autenticación mediante contextos.
- **Capa de servicios (BaaS)**: Supabase proporciona base de datos PostgreSQL, autenticación de usuarios, almacenamiento de ficheros y políticas de seguridad a nivel de fila (Row Level Security).
- **Servicios externos**: integración prevista con un proveedor de IA, **Gemini**, para la generación de rutinas y dietas personalizadas a partir de los datos del perfil del usuario.

La comunicación entre la aplicación y Supabase se realiza mediante la librería oficial @supabase/supabase-js, utilizando peticiones HTTP sobre HTTPS y respuestas en formato JSON.

En la siguiente imagen encontramos la estructura del proyecto:



A continuación, se detalla la arquitectura diferenciando la parte de frontend y el backend (Supabase).

2.1 Frontend

2.1.2 Tecnologías usadas

- Lenguaje principal: JavaScript con JSX para la definición de componentes.

- Framework / Librerías:
  - **React v19.1.1**: construcción de la SPA mediante componentes funcionales y hooks.
  - **Vite**: herramienta de bundling y desarrollo rápido, que proporciona recarga en caliente y una configuración mínima.
  - **React Router**: gestión de las rutas públicas y protegidas (por ejemplo, /, /login, /control, /dieta, /entrenamiento, /admin).
  - **TailwindCSS (v3.4.18)**: sistema de utilidades CSS para maquetar la interfaz, dar soporte a modo oscuro y garantizar el diseño responsive.
  - **React Icons**: librería de iconos para mejorar la usabilidad y la comunicación visual en botones, menús y tarjetas.

  - **JsPDF:** Permite generar documentos PDF desde el cliente. Se utiliza para crear las versiones imprimibles de las rutinas de entrenamiento y de las dietas semanales.
  - **Html2canvas:** Complementa a jsPDF capturando elementos del DOM como imágenes rasterizadas. Esto permite exportar paneles visuales completos (como la tabla diaria de comidas o la rutina con iconos) directamente al PDF, manteniendo el estilo visual del componente React.

2.1.2 Justificación de la elección:

- React y Vite permiten un desarrollo muy ágil, con un ecosistema maduro y abundante documentación.
- TailwindCSS simplifica la gestión de estilos al trabajar con clases utilitarias, evitando archivos CSS muy grandes y facilitando la coherencia visual.
- React Router se integra de forma natural con React y permite proteger rutas en función del estado de autenticación.

2.1.3 Entorno de desarrollo

- **Sistema operativo**: Windows 11.
- **IDE**: IntelliJ Ultimate, con extensiones para JavaScript, React, TailwindCSS e integración Git.
- **Gestor de dependencias**: npm (incluido con Node.js).
- **Navegador para pruebas**: Google Chrome, usando sus herramientas de desarrollo para depuración y análisis de rendimiento.
- **Control de versiones**: Git + GitHub, donde se almacena el repositorio público elenardila/FitMind con el código fuente del proyecto. GitHub

Este entorno se ha elegido por ser estándar en el desarrollo web moderno, disponer de abundante documentación y por integrarse bien con la plataforma de despliegue (Vercel).

2.2 Backend

En este proyecto no se ha desarrollado un backend propio con un framework tipo Spring. En su lugar se ha optado por un enfoque **Backend as a Service** utilizando Supabase.

2.2.1 Tecnologías usadas

- **Supabase**:
  - **PostgreSQL**, para almacenar datos de usuarios, perfiles, medidas y planes.
  - **Supabase Auth** para la autenticación (registro, login, logout, recuperación de contraseña) y gestión de sesiones en el cliente.
  - **Supabase Storage** para el almacenamiento de avatares de usuario.
  - **Row Level Security (RLS)** para asegurar que cada usuario solo pueda leer y modificar sus propios registros.
- **Librería de acceso**:
  - @supabase/supabase-js para conectarse desde React al proyecto de Supabase utilizando la URL del servicio y la clave anónima.
- **Servicios de IA**:

- Integración con un modelo de inteligencia artificial de Google AI (Gemini) para generar automáticamente dietas y rutinas a partir de los datos del perfil.

2.2.2 Justificación

- Supabase proporciona en un único servicio todo lo necesario (BD, Auth, Storage), lo que reduce drásticamente el tiempo de desarrollo y la complejidad de despliegue.
- Al utilizar PostgreSQL se dispone de un sistema potente y estándar, con soporte para SQL, vistas, funciones y buenas herramientas de administración.

2.2.3 Entorno de desarrollo

- **Dashboard web de Supabase** para la creación de tablas, definición de políticas RLS y ejecución de scripts SQL.
- **Vercel** como plataforma de despliegue del frontend, configurada con las variables de entorno necesarias para conectar con Supabase en producción. La aplicación se encuentra desplegada en https://fitmind-six.vercel.app.

3. Documentación Técnica

3.1 Análisis

En esta sección se resumen los principales requisitos funcionales y no funcionales que han guiado el desarrollo de FitMind.

3.1.1 Roles principales

- **Invitado**: puede acceder a la página de inicio (landing), consultar la información general de la aplicación y acceder al formulario de registro/login.
- **Usuario registrado**: dispone de perfil, panel de control y acceso a las secciones de dieta y entrenamiento, así como al historial de planes generados mediante IA.
- **Administrador**: puede acceder a un panel de administración para consultar: listado, filtrado y activación/desactivación de cuentas de usuarios.

3.1.2 Requisitos funcionales

- RF1. Un invitado puede registrarse mediante email y contraseña y confirmar su cuenta y convertirse en usuario registrado mediante Supabase Auth.
- RF2. Un usuario registrado puede iniciar y cerrar sesión de forma segura y recuperar la contraseña mediante correo electrónico.
- RF3. El usuario puede completar y editar su perfil, incluyendo datos personales y físicos (nombre, edad, sexo, altura, peso, nivel de actividad y objetivo). También puede gestionar alergias y actualizar su avatar almacenado en Supabase Storage.
- RF4. El usuario dispone de un panel de control donde se calcula y muestra su IMC, objetivo calórico diario y resumen del último plan activo.
- RF5. Desde la sección Dieta, el usuario puede generar, consultar, guardar y eliminar planes de comidas semanal.

- RF6. Desde la sección Entrenamiento, el usuario puede generar, consultar, guardar y eliminar rutinas de ejercicios semanal.
- RF7. El usuario puede descargar cualquier dieta o rutina en formato PDF.
- RF8. El administrador puede consultar la lista de usuarios, filtrarla por fecha de creación, sexo, peso y altura, resetear filtros, y activar/desactivar cuentas sin eliminarlas físicamente.

3.1.3 Requisitos no funcionales

- RNF1. Arquitectura: aplicación front-only. La aplicación puede desplegarse fácilmente en servicios de hosting estáticos y no requiere mantenimiento de servidores propios.
- RNF2. Seguridad: autenticación y control de acceso con Supabase Auth. Solo los usuarios autenticados acceden a sus datos personales y ningún usuario puede consultar o modificar información de otros.
- RNF3. Privacidad y tratamiento de datos. La aplicación muestra desde la landing un aviso legal y una política de privacidad. Los datos del usuario (edad, peso, altura, alergias, preferencias, avatar) se utilizan exclusivamente para generar sus planes. El usuario puede eliminar su cuenta, lo cual implica la desaparición del perfil y restricción de acceso a sus registros asociados.
- RNF4. Rendimiento: fluidez en la experiencia de usuario. Al tratarse de una SPA, la navegación entre páginas es inmediata. Las operaciones principales (calcular IMC, cargar rutinas, cargar dietas, actualizar perfil y generar PDF) se ejecutan en el cliente, evitando esperas prolongadas.
- RNF5. Interfaz y usabilidad: La aplicación utiliza TailwindCSS para ofrecer un diseño limpio, coherente y responsive.
- RNF6. Generación de PDF sin backend adicional. Los usuarios pueden descargar sus planes en PDF en cualquier momento, manteniendo el estilo visual de la aplicación.

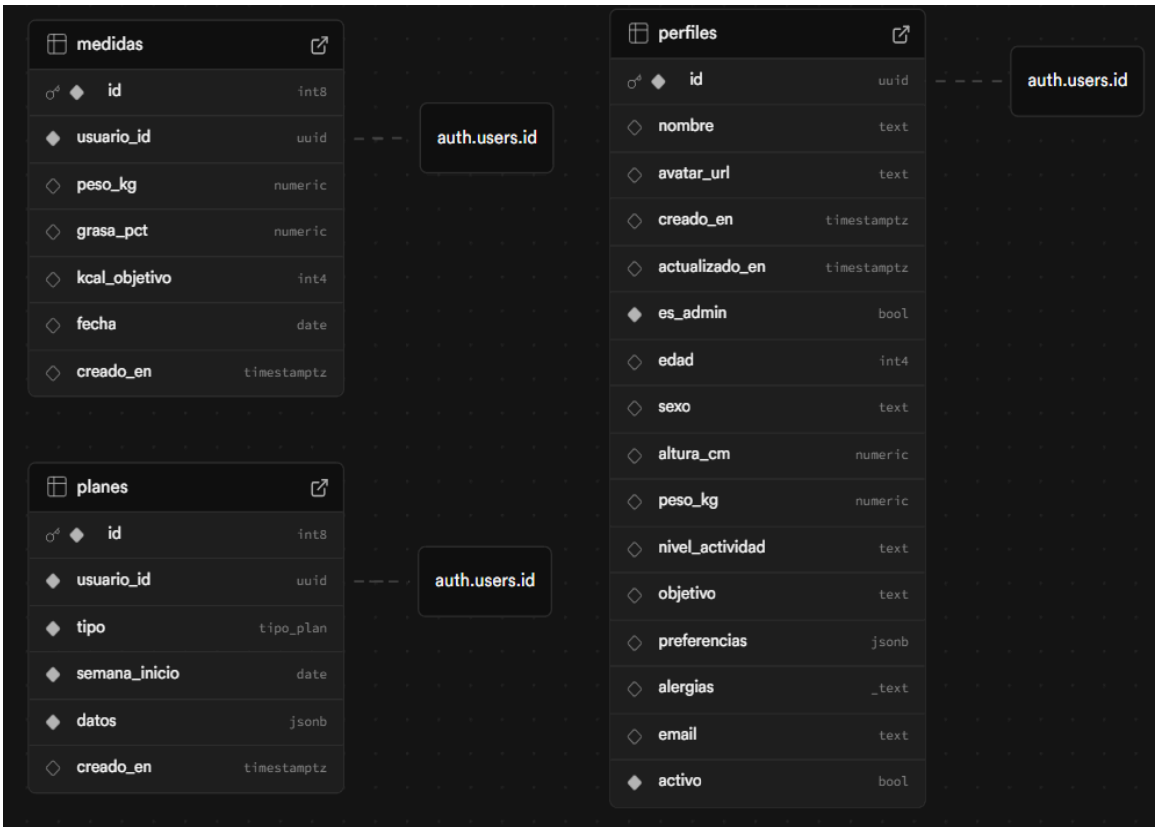
3.2 Desarrollo

3.2.1 Base de Datos

La aplicación FitMind utiliza una base de datos PostgreSQL gestionada por Supabase.

La estructura de tablas se ha diseñado teniendo en cuenta la integración directa con el sistema de autenticación Supabase Auth, lo cual influye en la forma en que se representan las relaciones dentro del modelo.

A continuación se muestra la representación generada por el propio panel de Supabase:



En esta vista se aprecia que las tablas aparecen como entidades independientes y no muestran relaciones visuales entre sí. Esto puede resultar llamativo a primera vista y dar la impresión de que falta la integridad referencial.

Sin embargo, este comportamiento es normal y esperado cuando se trabaja con Supabase Auth ya que utiliza un esquema interno llamado auth, cuya tabla principal es auth.users. Esta tabla actúa como la entidad central que contiene:

- el identificador único del usuario (UUID)

- el email
- la contraseña cifrada
- los metadatos básicos

Por tanto, las tablas de la aplicación no referencian entre sí, sino que todas ellas apuntan directamente a auth.users.id. Por este motivo, Supabase únicamente dibuja las flechas hacia la tabla interna auth.users, pero no representa las relaciones indirectas entre las tablas, lo que provoca que en la interfaz visual parezcan desconectadas.

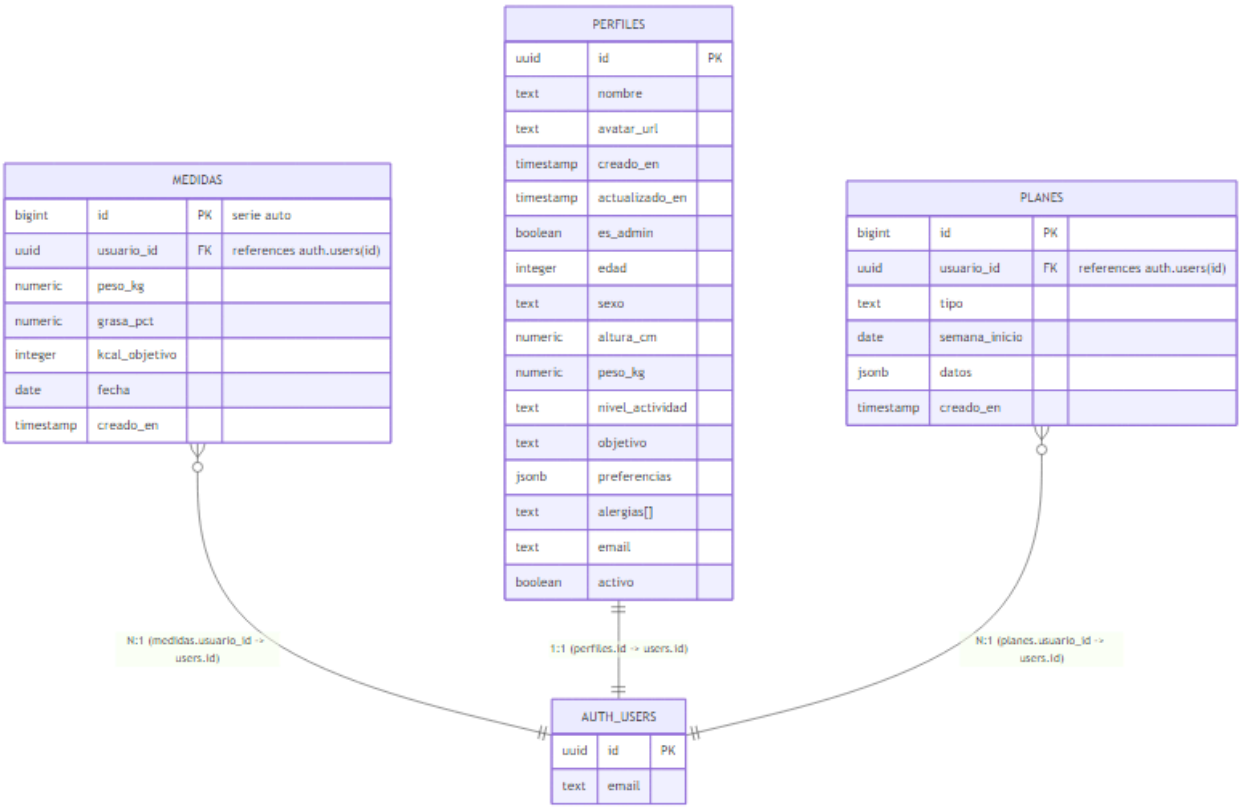
La siguiente imagen muestra la estructura real del modelo relacional. En ella se puede observar claramente que:

- La tabla perfiles tiene una relación 1:1 con auth.users (comparten el mismo UUID como clave primaria y foránea).
- Las tablas medidas y planes mantienen una relación N:1 con auth.users mediante el campo usuario_id.

Aunque el panel gráfico de Supabase no refleje las flechas entre las tablas del esquema public, el vínculo lógico sí existe: todas las tablas comparten el mismo identificador de usuario desde auth.users. Esto implica relaciones efectivas:

- perfiles → medidas: relación 1:N
- perfiles → planes: relación 1:N

La diferencia está en que dichas relaciones no se implementan mediante claves foráneas directas entre tablas propias, sino mediante un punto común centralizado (auth.users.id), que simplifica las reglas de seguridad (RLS), reduce duplicidad y evita inconsistencias.



3.2.2 Justificación del diseño

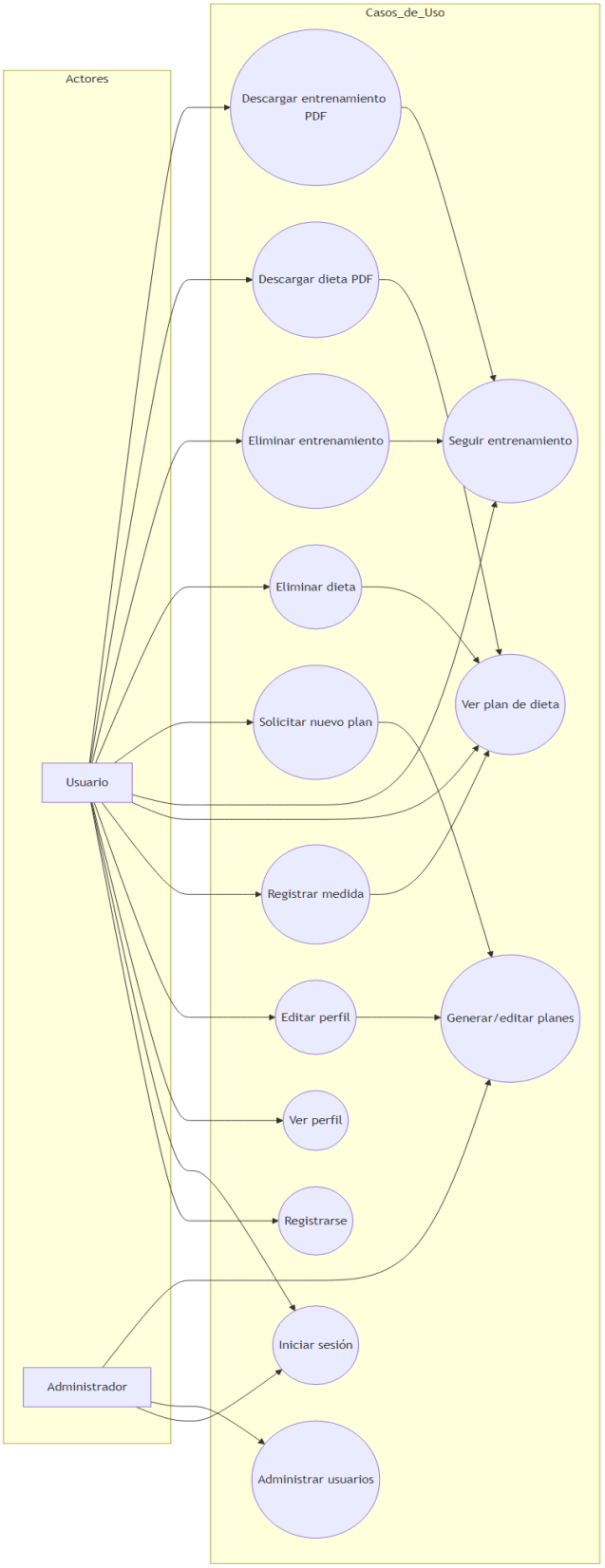
El uso de auth.users como tabla raíz ofrece ventajas importantes:

- Integración natural con Supabase Auth: un único origen de verdad para la identidad del usuario.
- Mayor seguridad: las políticas RLS pueden comparar fácilmente auth.uid() con los campos usuario_id.
- Menor redundancia: no es necesario duplicar relaciones entre tablas del esquema public.
- Evolución más sencilla: cualquier cambio en la estructura del perfil no afecta al resto de tablas.

Este enfoque es el recomendado por Supabase para aplicaciones front-only que delegan autenticación y persistencia al propio servicio.

3.2.3 Diagrama de casos de uso

La siguiente figura muestra el diagrama de Casos de Uso de la aplicación FitMind.



Los casos de uso representados recogen las funcionalidades que la aplicación implementa realmente:

- Registro e inicio de sesión: ambos actores pueden autenticarse en el sistema mediante Supabase Auth.
- Gestión de perfil: el usuario puede ver y editar sus datos personales, físicos y preferencias alimentarias.
- Registro de medidas: el usuario puede almacenar sus mediciones corporales (peso, grasa, kcal objetivo).
- Generación de planes: el usuario puede solicitar nuevos planes semanales mediante la API de IA, tanto de dieta como de entrenamiento.
- Consulta y seguimiento: incluye ver planes existentes, seguir el entrenamiento diario y consultar los detalles de cada dieta.
- Eliminación de planes: el usuario puede eliminar una dieta o un entrenamiento previo.
- Descarga en PDF: tanto la dieta como la rutina semanal pueden exportarse a PDF utilizando jsPDF + html2canvas.
- Administración de usuarios: el administrador accede a una sección específica para gestionar el ciclo de vida de los usuarios (listar, filtrar y activar/desactivar cuentas).

3.3 Pruebas realizadas

Aunque no se ha implantado un framework de pruebas automatizadas completo, se han llevado a cabo pruebas manuales sistemáticas sobre las funcionalidades principales:

➔ **Pruebas de registro y login**: Login + AuthContext.

Datos de entrada:

- Caso válido: email correcto y contraseña con longitud mínima.
- Casos inválidos: email vacío, formato incorrecto, contraseñas que no coinciden en registro, etc.

Resultado esperado:

- En el caso válido, Supabase devuelve una sesión y el usuario es redirigido al panel de control.
- En casos inválidos se muestran mensajes de error específicos y no se crea sesión.

➔ **Pruebas del panel de control (Control):** página Control y funciones de cálculo de IMC y calorías objetivo

Datos de entrada:

- Varias combinaciones de altura y peso de prueba en el perfil.

Resultado esperado:

- El IMC se calcula correctamente según la fórmula IMC = peso(kg) / altura(m)^2.
- La clasificación (bajo peso, peso ideal, sobrepeso, etc.) se actualiza en función del valor calculado.

➔ **Pruebas de gestión de planes de dieta y entrenamiento:** páginas Dieta y Entrenamiento.

Datos de entrada:

- Creación de varios planes con distintos nombres y semanas.

Resultado esperado:

- El plan se guarda en la tabla planes asociado al usuario.
- La carga del historial muestra correctamente los planes previos.
- Se mantiene la coherencia entre lo mostrado en la interfaz y lo almacenado en la base de datos.

➔ **Pruebas del panel de administración:** página Admin.

Datos de entrada:

- Base de datos con varios usuarios de prueba con distintas fechas, pesos, alturas y sexos.

Resultado esperado:

- Los filtros por fecha de creación, sexo, peso y altura funcionan correctamente.
- El botón de "resetear filtros" restaura el listado completo.
- Al activar/desactivar un usuario, el cambio se refleja en la interfaz y en la base de datos.

4. Proceso de despliegue

El despliegue de FitMind se realiza sobre dos servicios principales:

- Supabase: para la base de datos, autenticación y almacenamiento.
- Vercel: para alojar el frontend construido con React + Vite.

4.1 Software necesario

- Navegador web moderno.
- Cuenta en GitHub.
- Cuenta en Supabase.
- Cuenta en Vercel.
- Node.js y npm instalados localmente (si se desea hacer el build en local).

4.2 Pasos para desplegar la aplicación

4.2.1 Configurar Supabase

a. Crear un nuevo proyecto en Supabase.

b. Definir las tablas perfiles, medidas y planes mediante el editor SQL o el diseñador de tablas.

c. Establecer las políticas RLS necesarias para que cada usuario solo pueda acceder a sus propios registros.

d. Habilitar Supabase Auth con el proveedor de correo electrónico.

e. Obtener la URL del proyecto y la anon key, que se utilizarán desde el frontend.

4.2.2 Configurar el proyecto local

f. Clonar el repositorio desde GitHub:

```
git clone https://github.com/elenardila/FitMind.git
cd FitMind
```

g. Crear un archivo .env.local con las variables:

```
VITE_SUPABASE_URL=...
VITE_SUPABASE_ANON_KEY=...
VITE_GEMINI_KEY=...
```

h. Instalar dependencias y probar localmente:

```
npm install
npm run dev
```

4.2.3 Despliegue en Vercel

i. Iniciar sesión en Vercel y elegir la opción de “Importar proyecto desde GitHub”.

j. Seleccionar el repositorio FitMind.

k. Configurar las variables de entorno en Vercel con los mismos valores usados en local.

l. Lanzar el despliegue.

m. Una vez finalizado, Vercel proporcionará una URL pública desde la que se puede acceder a la aplicación.

Cualquier nuevo cambio que se suba a la rama principal del repositorio (main) puede volver a desplegarse de forma automática al estar integrado con Vercel.

5. Propuestas de mejoras

Durante el desarrollo del proyecto se han identificado distintas posibilidades de mejora, tanto funcionales como técnicas:

- Permitir edición manual avanzada de las rutinas y dietas generadas (añadir, mover o sustituir ejercicios y platos).
- Incluir un sistema de histórico de progreso más detallado, con gráficas de evolución del peso, IMC y calorías consumidas/objetivo.
- Añadir notificaciones (por ejemplo, recordatorios diarios de entrenamiento o de comidas) mediante correo electrónico u otro canal.
- Ampliar las funcionalidades administrador: consultar el histórico de planes de entrenamiento y dietas de los usuarios.
- Personalizar los correos electrónicos de confirmación de registro y cambio de contraseña.

6. Bibliografía

A continuación, se recogen las principales fuentes consultadas durante el desarrollo del proyecto y la elaboración de este manual:

- Documentación oficial de React: conceptos básicos, hooks y buenas prácticas en desarrollo de SPAs. https://es.react.dev/
- Documentación oficial de Vite: configuración de proyectos React, scripts de build y despliegue. https://es.vite.dev/guide/
- Documentación oficial de TailwindCSS: sistema de utilidades y diseño responsive. https://tailwindcss.com/
- Documentación oficial de Supabase: configuración de proyectos, Auth, Storage y Row Level Security. https://supabase.com/
- Documentación de Vercel para despliegue de aplicaciones React/Vite. https://vercel.com/
- Documentación de Google AI Studio. https://aistudio.google.com/app/
- Apuntes y documentación interna del módulo de Proyecto de Desarrollo de Aplicaciones Web del IES Albarregas, incluyendo la plantilla de manual técnico y ejemplos de proyectos anteriores.