2025

Proyecto MiniFit

Modelo Vistas 4+1

**Integrantes:** Dyland Martinez Cisternas

Polet Adasme Peñailillo

Fernanda Barra

**Asignatura:** CAPSTONE

**Sección:** 007v

**Docente:** Fernando Gonzalo Herrera Francesconi

**INDICE**

# Vistas 4+1

## Modelo de Vistas 4+1 aplicado a MiniFit

Es una metodología de arquitectura de software que permite describir un sistema desde cinco perspectivas complementarias. En el contexto del proyecto MiniFit, este modelo facilita una documentación clara, estructurada y comprensible tanto para desarrolladores como para usuarios y evaluadores académicos.

Las vistas aplicadas son:

• Vista de Escenarios: valida el funcionamiento del sistema mediante casos de uso reales, como el registro de alimentación diaria o la visualización de reportes por parte del apoderado.

• Vista Lógica: muestra los módulos funcionales de MiniFit, como el registro de hábitos, reportes, recomendaciones y gestión administrativa.

• Vista de Procesos: representa el comportamiento dinámico del sistema, diferenciando las acciones del apoderado y del administrador.

• Vista Física (o de Despliegue): presenta cómo se distribuyen los componentes en el entorno real, incluyendo servidores, dispositivos cliente y conexiones seguras.

• Vista de Desarrollo: describe cómo está organizado el código fuente, incluyendo la API .NET, la base de datos SQL Server y las interfaces móviles y web.

Este enfoque permite documentar MiniFit de forma integral, asegurando que cada aspecto técnico y funcional esté respaldado por una vista específica y coherente.

## Vistas de escenario (Casos de Uso)

La vista de escenario representa cómo los distintos actores interactúan con el sistema MiniFit, permitiendo visualizar los objetivos funcionales desde la perspectiva del usuario. Esta vista es clave para validar los requerimientos del proyecto y asegurar que las funcionalidades respondan a las necesidades reales de quienes utilizarán la aplicación.

MiniFit está diseñado principalmente para apoderados, quienes serán los encargados de registrar y monitorear los hábitos alimenticios y de actividad física de sus hijos. El sistema también contempla el rol de administrador, responsable de mantener la calidad y consistencia de los datos, y el rol pasivo del niño, quien se beneficia indirectamente del uso de la aplicación.

Los casos de uso incluyen acciones como iniciar sesión, registrar alimentación y actividad física, consultar reportes y administrar la cuenta. Estas interacciones se modelan mediante un diagrama de casos de uso UML, que permite identificar claramente qué funcionalidades están disponibles para cada tipo de usuario.

Esta vista facilita la comprensión del sistema por parte de los stakeholders no técnicos, como docentes, patrocinadores o instituciones educativas, ya que muestra de forma simple y visual cómo MiniFit contribuye al acompañamiento familiar en la formación de hábitos saludables.

**Actores definidos**

•**Apoderado**: usuario principal de la aplicación móvil. Registra la alimentación y actividad física del niño, consulta reportes y recibe recomendaciones.

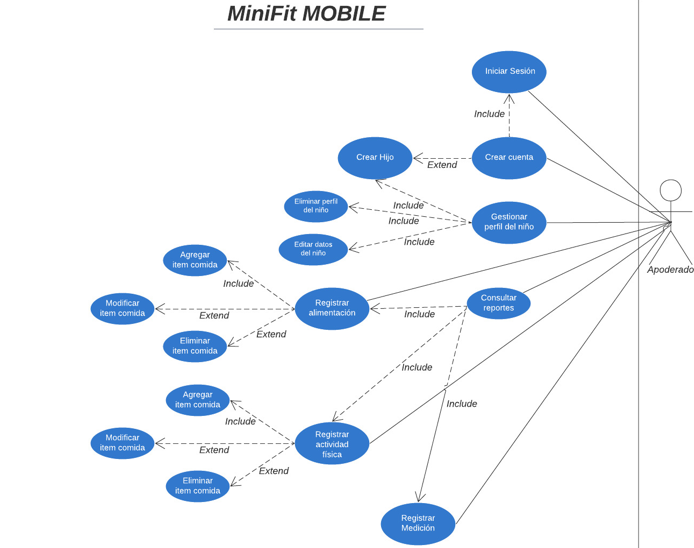
• **Administrado**r: usuario de la aplicación web. Gestiona el catálogo de alimentos y actividades, configura parámetros del sistema y administra perfiles de usuarios.

• **Niño**: usuario pasivo. No interactúa directamente con el sistema, pero es el beneficiario de los registros realizados por el apoderado.

### **MiniFit MOBILE – Diagrama de Casos de Uso**

Este diagrama representa las funcionalidades disponibles para el Apoderado en la aplicación móvil. Incluye acciones como:

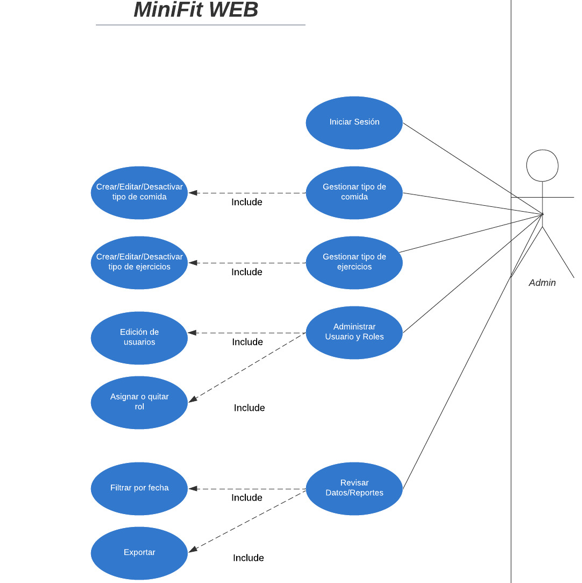
* Crear cuenta e iniciar sesión
* Registrar actividad del niño
* Registrar alimentación del niño
* Visualizar gráficos de progreso
* Navegar por menús principales y secundarios



### **MiniFit WEB – Diagrama de Casos de Uso**

Este diagrama representa las funcionalidades disponibles para el Administrador en la aplicación web MiniFit. Este actor tiene un rol clave en la mantención y supervisión del sistema, asegurando que los datos estén actualizados, consistentes y correctamente organizados para el uso de los apoderados.

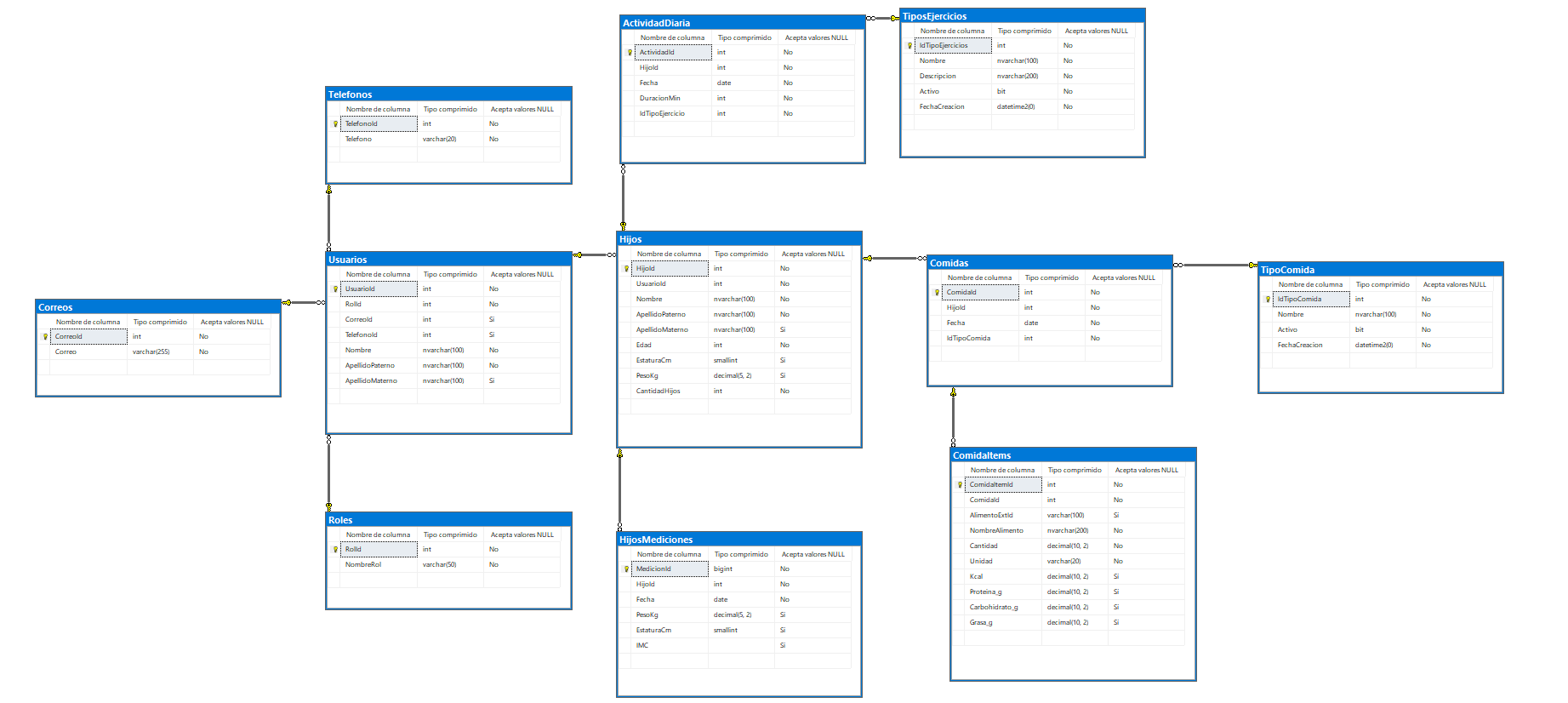
* Iniciar Sesión
* Gestionar Catálogo de Actividades
* Gestionar Catálogo de Alimentos
* Revisar Datos/Reportes
* Administras Usuario y Roles



## Vistas Lógicas

La vista lógica representa la estructura funcional del sistema MiniFit, mostrando los principales módulos que componen la aplicación y cómo se relacionan entre sí. Esta vista permite comprender la organización interna del software, facilitando el desarrollo, mantenimiento y escalabilidad del proyecto.

Sistema para registrar y visualizar información de familias: padres (usuarios) gestionan hijos, sus actividades físicas, comidas (con ítems y macros) y mediciones corporales. La app web (admin) supervisa y parametriza catálogos (tipos de comida, tipos de ejercicio), mientras la app móvil registra el día a día.



### Entidades y responsabilidades

* **Usuarios o Campos clave**, UsuarioId, Nombre, ApellidoPaterno, ApellidoMaterno, CorreoId, TelefonoId, RolId. o Rol de negocio: actor principal (padre/admin). Posee contacto (correo/teléfono) y rol.
* **Telefonos,** Datos de contacto normalizados TelefonoId para referenciar desde Usuarios.
* **Roles** , NombreRol. Control de permisos (Admin / Padre).
* **Correos,** Datos de contacto normalizados CorreoId para referenciar desde Usuarios.
* **Hijos o HijoId**, UsuarioId, datos demográficos: nombres, Edad, EstaturaCm, PesoKg, CantidadHijos. o Relación: pertenece a un Usuario (Padre).
* **HijosMediciones**, HijoId, Fecha, PesoKg, EstaturaCm, IMC. o Historial antropométrico del hijo.
* **ActividadDiaria**, o ActividadId, HijoId, Fecha, DuracionMin, IdTipoEjercicio. o Registro de ejercicio diario por hijo.
* **TiposEjercicios,** IdTipoEjercicio, Nombre, Descripcion, Activo.
* **Comidas,** HijoId, Fecha, IdTipoComida. o Encabezado de una ingesta (desayun/almuerzo/cena, etc.).
* **TipoComida**, IdTipoComida, Nombre, Activo.
* **ComidaItems**, ComidaItemId, ComidaId, NombreAlimento, Cantidad, Unidad, macros (Kcal, Proteinas\_g, Carbohidratos\_g, Grasas\_g)

### Relaciones (cardinalidades)

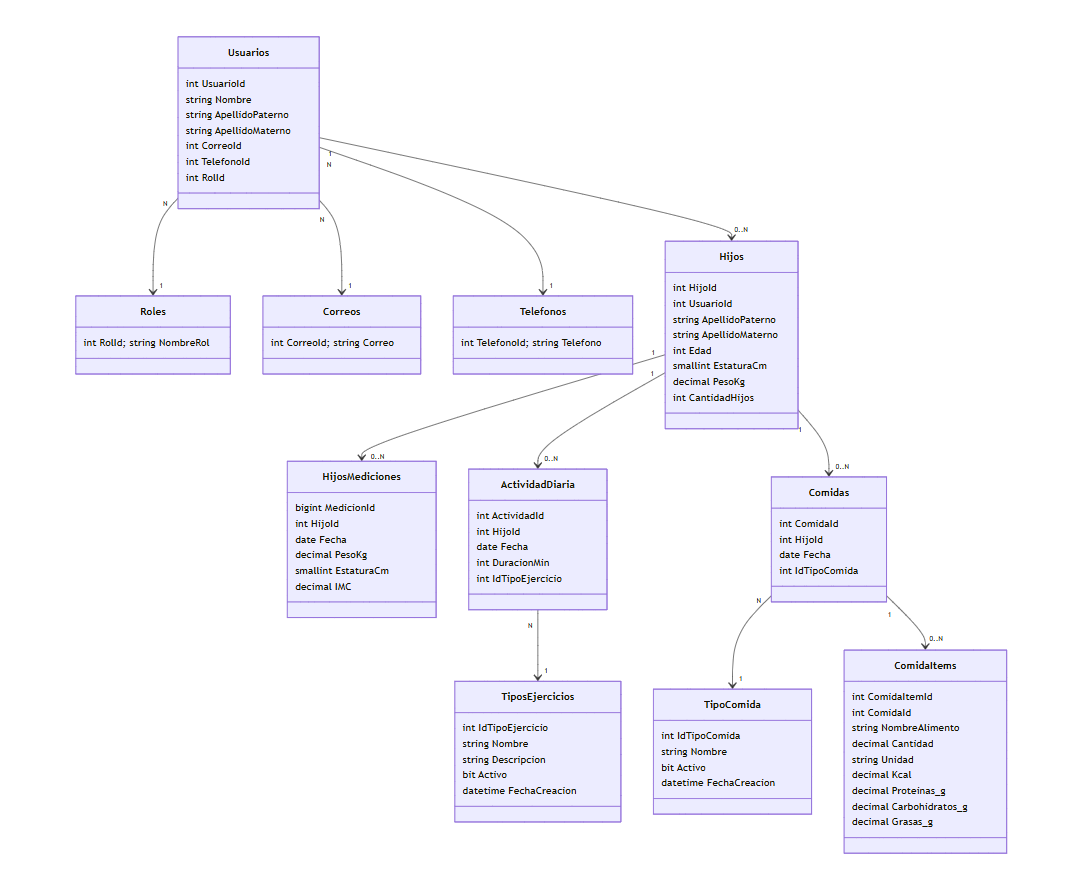
* **Usuario (1) — (N) Hijos**: un padre registra varios hijos.
* **Usuario (N) — (1) Rol**: cada usuario tiene un rol (admin/padre).
* **Usuario (N) — (1) Correo y (N) — (1) Teléfono**: referencia a contacto (modelo actual usa FK simple).
* **Hijo (1) — (N) HijosMediciones**: histórico físico del hijo.
* **Hijo (1) — (N) ActividadDiaria**; ActividadDiaria (N) — (1) TiposEjercicios.
* **Hijo (1) — (N) Comidas**; Comidas (N) — (1) TipoComida.
* **Comidas (1) — (N) ComidaItems**: detalle de alimentos y macros por comida

### Reglas de negocio clave

* **Integridad por FK**: no hay Actividad/Comida/Medición sin Hijo válido.
* **Catálogos activos**: TiposEjercicios y TipoComida controlan opciones disponibles.
* **Agregación nutricional**: totales por comida = suma de ComidaItems (Kcal, macros).
* **Trazabilidad temporal**: Fecha en Actividad/Comida/Medición permite reportes diarios/semanales

### Casos de uso núcleo (valida esta vista)

* **Apoderado**: crear hijo → registrar actividad/comida/medición → ver progreso.
* **Admin** (web): gestionar catálogos (tipos de ejercicio/comida) y revisar datos.



## Vistas de Procesos

### Apoderado/Niño

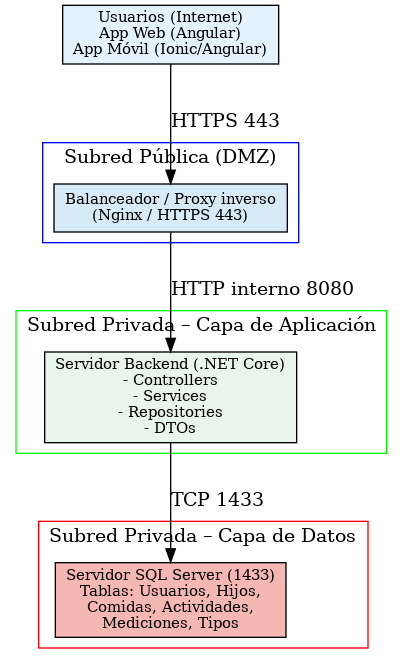
Este diagrama muestra los escenarios representativos para el apoderado/niño en la App Móvil (Ionic/Angular). Se incluyen: Registrar Hijo, Registrar Comida (con encabezado e ítems), Registrar Actividad, Registrar Medición y Consultar Historial. Cada acción inicia en la aplicación, pasa por la API Backend (.NET Core) y se almacena o consulta en SQL Server (1433).

Administrador

Este diagrama muestra los escenarios representativos para el administrador en la App Web (Angular). Se incluyen: Gestionar Tipos de Comida, Gestionar Tipos de Ejercicio, Gestionar Usuarios/Roles y Revisar Datos/Reportes. Cada acción inicia en la aplicación web, pasa por la API Backend (.NET Core) y actualiza o consulta la base de datos SQL Server (1433).

## Vistas de Física(Despliegue)

La vista física describe la infraestructura donde se despliega el sistema. Está organizada en tres capas principales: acceso, aplicación y datos. Los usuarios acceden desde Internet mediante la App Web (Angular) o la App Móvil (Ionic/Angular), conectándose por HTTPS (443) a un balanceador o proxy inverso en la subred pública (DMZ). Este balanceador enruta las solicitudes hacia el backend desarrollado en .NET Core, desplegado en la subred privada de aplicación. Finalmente, la base de datos SQL Server (1433) se encuentra en una subred privada de datos y solo es accesible desde el backend, garantizando la seguridad e integridad de la información.



## Vista de Desarrollo (Implementación)

La vista de desarrollo muestra cómo se organiza el sistema a nivel de componentes y capas de software. El proyecto está compuesto por dos aplicaciones frontend (App Web en Angular y App Móvil en Ionic/Angular), un backend desarrollado en .NET Core y una base de datos en SQL Server. Cada frontend se comunica con la API a través de servicios REST, y la API gestiona la lógica de negocio, validaciones y acceso a datos mediante Entity Framework Core. La base de datos centraliza todas las entidades principales: usuarios, roles, hijos, comidas con sus ítems, actividades, mediciones, tipos de comida y tipos de ejercicio.

