INSTITUTO PROFESIONAL DUOC UC

SEDE MAIPU

ESCUELA DE INFORMÁTICA Y TELECOMUNICACIONES

| **Antecedentes Personales** |
| --- |

| Nombre estudiante | **José Caamaño**  **Esteban Caamaño**  **Elias Gonzalez** |
| --- | --- |
| Rut | **21287150-8**  **21287169-9**  **21503169-1** |
| Carrera | **Ing. informática** |
| Sede | **Maipú** |

Índice

[**1. Vista Lógica**](#_lgi2rg2mu7qs) **2**

[**2. Vista de Desarrollo**](#_624r4vvhppgf) **4**

[Estructura del proyecto](#_brwjxoafbqfw) 4

[Descripción técnica](#_jzdummktp4i) 5

[Diseño técnico](#_udechmagrzqf) 5

[**3. Vista de Procesos**](#_1wiuxfic1rjg) **6**

[**4. Vista Física**](#_s0obewxsr1db) **7**

[**5. Vista +1 (Escenarios de Uso)**](#_351akvh5wock) **8**

[Escenario 1 – Registro e inicio de sesión](#_u685hgrsxka6) 8

[Escenario 2 – Creación de nota](#_lr4ut9nfy0s8) 8

[Escenario 3 – Programación de rutina con recordatorio](#_d682e2kzzrac) 9

[Escenario 4 – Resumen diario](#_mvwb31dnetp0) 9

[Escenario 5 – Cierre de sesión](#_wb72ousax4n7) 9

[**6. Conclusión Final**](#_5kol9b71ij9i) **9**

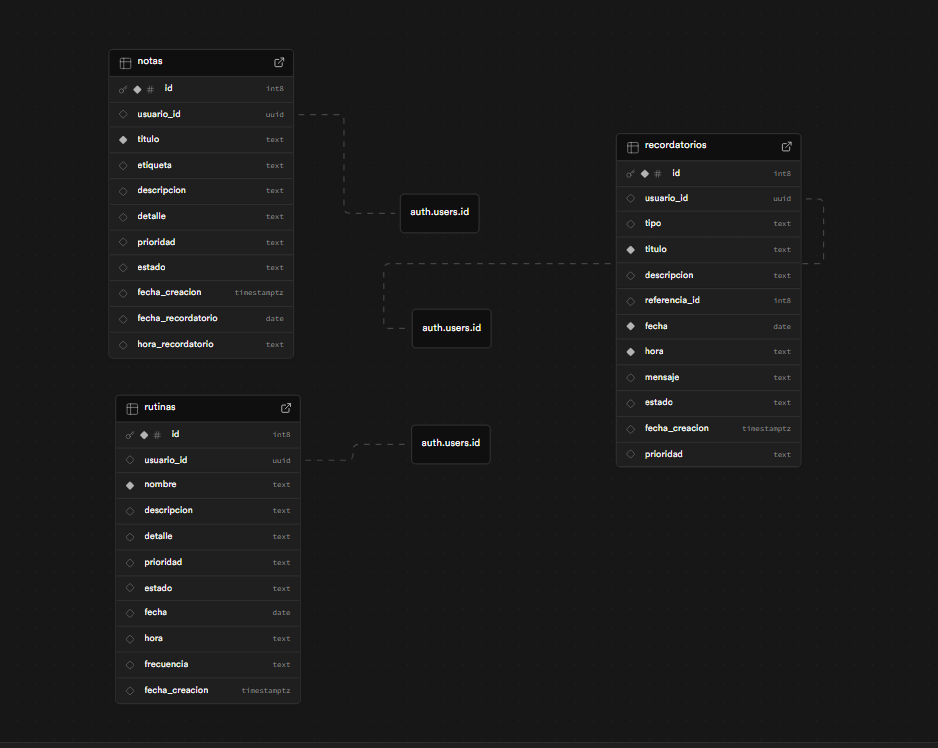
## 1. Vista Lógica

La vista lógica define la estructura funcional del sistema y las relaciones entre sus principales componentes.  
**AgendIO** es una aplicación móvil y web orientada a la organización personal. Permite gestionar **notas, rutinas y recordatorios**, integrando autenticación segura y notificaciones locales.

**Componentes funcionales principales:**

* **Gestión de Usuarios:** registro, inicio y cierre de sesión mediante Supabase Auth. Cada usuario mantiene su propio entorno de información.
* **Gestión de Notas:** permite crear, editar, eliminar y visualizar notas con campos de título, descripción, etiqueta y prioridad.
* **Gestión de Rutinas:** registra actividades recurrentes, asignando fecha, hora, estado y nivel de prioridad.
* **Gestión de Recordatorios:** genera alertas y notificaciones locales asociadas a notas o rutinas.
* **Configuración:** administra opciones del usuario como tema (modo claro/oscuro) y ajustes personales.
* **Página de Inicio:** centraliza el acceso a las funcionalidades, mostrando el resumen de notas, rutinas y recordatorios del día.

Cada módulo se conecta con la base de datos de Supabase y mantiene independencia lógica, promoviendo la **modularidad**, **reutilización** y **coherencia visual** de la interfaz.



## 2. Vista de Desarrollo

La vista de desarrollo describe la organización del código fuente y los componentes que integran la aplicación.  
 El sistema utiliza una **arquitectura modular por capas**, separando datos, lógica de negocio, servicios y presentación visual.

#### **Estructura del proyecto**

C:.

| main.dart ← Punto de entrada de la aplicación

|

+---core

| supabase\_client.dart ← Configuración e inicialización del cliente Supabase

|

+---data

| data\_local.dart ← Fuente de datos local para pruebas o modo offline

|

+---models

| nota.dart ← Modelo de datos de Notas

| recordatorio.dart ← Modelo de datos de Recordatorios

| rutina.dart ← Modelo de datos de Rutinas

|

+---pages

| | settings\_page.dart ← Página de configuración (tema, preferencias, perfil)

| |

| +---auth

| | login\_page.dart ← Pantalla de inicio de sesión (Supabase Auth)

| | register\_page.dart ← Pantalla de registro de usuario

| |

| +---home

| | inicio\_page.dart ← Pantalla principal con resumen de notas, rutinas y recordatorios

| |

| +---nota

| | notas\_page.dart ← Listado principal de notas

| | form\_nota\_page.dart ← Formulario para crear o editar notas

| | detalle\_nota\_page.dart ← Detalle individual de nota seleccionada

| |

| +---rutina

| | rutinas\_page.dart ← Listado de rutinas activas

| | form\_rutina\_page.dart ← Creación o edición de rutina

| | detalle\_rutina\_page.dart ← Visualización detallada de una rutina

| |

| \---recordatorio

| recordatorios\_page.dart ← Listado de recordatorios

| form\_recordatorio\_page.dart← Formulario para crear o editar recordatorio

| detalle\_recordatorio\_page.dart ← Detalle del recordatorio programado

|

+---services

| auth\_service.dart ← Gestión de autenticación, sesiones y verificación de usuario

| notification\_service.dart ← Servicio de notificaciones locales y programación de alertas

|

+---test

| test\_notificacion\_page.dart ← Página de pruebas para validar notificaciones y eventos

|

\---widgets

card\_item.dart ← Widget reutilizable para mostrar notas, rutinas o recordatorios

#### **Descripción técnica**

* **main.dart:** define el punto de inicio, las rutas principales y el tema general de la aplicación.
* **core/:** configura la conexión a **Supabase**, inicializando servicios de autenticación y base de datos.
* **data/:** actúa como capa de datos local, útil para pruebas o funcionamiento sin conexión.
* **models/:** contiene las clases de datos que representan las tablas de Supabase (Notas, Rutinas, Recordatorios).
* **pages/:** agrupa las vistas de usuario organizadas por módulo funcional.
* **services/:** concentra la lógica de negocio y la integración con servicios externos (Auth, Notificaciones).
* **widgets/:** guarda componentes reutilizables para tarjetas y elementos visuales.
* **test/:** espacio destinado a pruebas controladas y depuración.

#### **Diseño técnico**

* Arquitectura **modular y desacoplada**, orientada a la mantenibilidad.
* Integración directa con **Supabase** mediante supabase\_client.dart y auth\_service.dart.
* Soporte para **tema dinámico** (modo oscuro/claro) desde settings\_page.dart.
* Uso de **programación asíncrona** (FutureBuilder, async/await) para rendimiento fluido.
* Preparación para futuras expansiones, como la integración de una **IA de organización inteligente**.

Esta estructura facilita la escalabilidad y la evolución continua del proyecto, manteniendo claridad y orden en el desarrollo.

## 

## 

## 3. Vista de Procesos

La vista de procesos representa el comportamiento dinámico del sistema, mostrando cómo interactúan los módulos y servicios.

**Flujo principal del sistema:**

1. El usuario se autentica mediante **Supabase Auth** (login o registro).
2. Al iniciar sesión, se carga la **InicioPage**, que consulta la base de datos y muestra la información diaria.
3. El usuario puede:  
   * Crear o editar notas → guardadas en notas.
   * Programar rutinas → almacenadas en rutinas.
   * Configurar recordatorios → agendados mediante notification\_service.dart.
4. Cada acción genera una solicitud a Supabase (insert/update/delete) que actualiza la interfaz en tiempo real.
5. El usuario puede cerrar sesión, limpiando los datos locales y redirigiendo a la pantalla de autenticación.

**Procesos paralelos:**

* Autenticación y validación de sesión.
* Sincronización de datos locales con la nube.
* Gestión de notificaciones asíncronas.
* Renderizado independiente de cada módulo.

La arquitectura maneja múltiples tareas simultáneamente sin comprometer la estabilidad ni la experiencia del usuario.

## 4. Vista Física

La vista física muestra la infraestructura tecnológica donde se despliega **AgendIO**.

**Cliente (Front-End):**

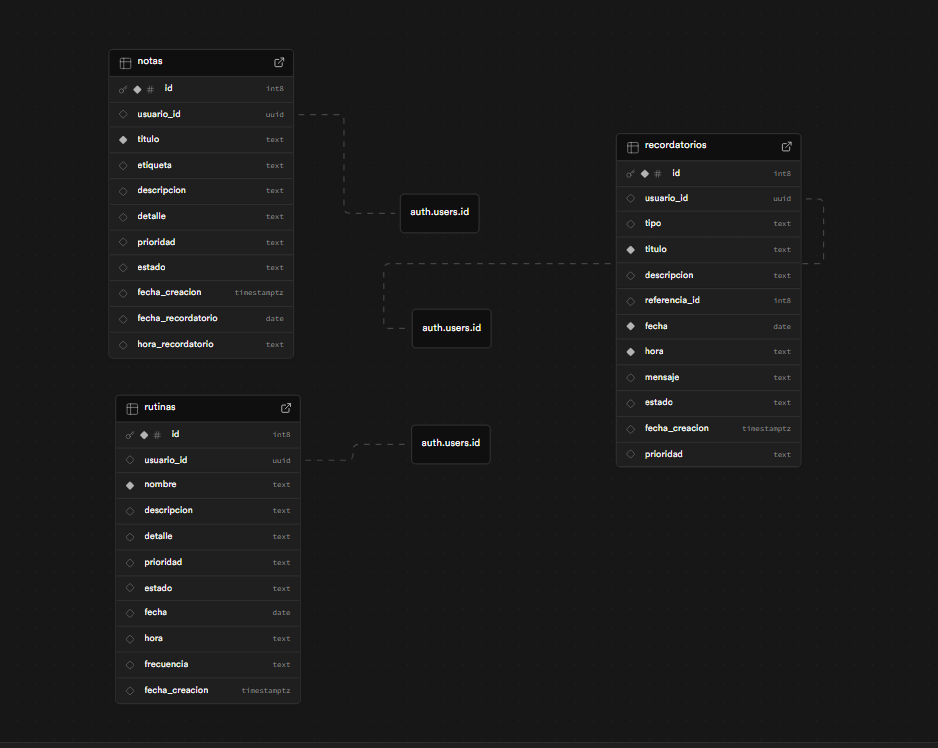
* Aplicación Flutter multiplataforma (Android, iOS y Web).
* Manejo local de sesión y preferencias mediante SharedPreferences o localStorage.

**Servidor (Backend / Nube):**

* **Supabase Auth:** autenticación segura con JWT.
* **Supabase Database (PostgreSQL):** almacenamiento estructurado de notas, rutinas y recordatorios.
* **Supabase Storage:** para archivos o adjuntos (en versiones futuras).
* **Supabase Edge Functions:** potencial para automatización y lógica avanzada.

**Despliegue:**

* Código fuente en **GitHub** o entorno local.
* Backend en **Supabase Cloud**.
* Comunicación cifrada mediante HTTPS.



## 5. Vista +1 (Escenarios de Uso)

Los escenarios de uso validan la funcionalidad integral del sistema y la coherencia de su arquitectura.

#### **Escenario 1 – Registro e inicio de sesión**

El usuario crea una cuenta o accede con sus credenciales. Supabase valida la identidad y devuelve el token de sesión.  
-Confirma autenticación y vinculación con la base de datos.

#### **Escenario 2 – Creación de nota**

El usuario genera una nota nueva desde form\_nota\_page.dart. Se almacena en la tabla notas.  
-Valida el flujo CRUD (crear, leer, actualizar, eliminar).

#### **Escenario 3 – Programación de rutina con recordatorio**

El usuario crea una rutina y asocia un recordatorio.  
 notification\_service.dart programa una alerta local.  
-Verifica integración entre módulos y notificaciones.

#### **Escenario 4 – Resumen diario**

Desde inicio\_page.dart, el usuario visualiza las tareas, notas y recordatorios activos.  
-Comprueba la comunicación entre capas y el refresco de datos.

#### **Escenario 5 – Cierre de sesión**

El usuario sale de su cuenta y la app limpia los datos locales.  
 Garantiza control de sesión y privacidad.

## 6. Conclusión Final

El modelo arquitectónico **4+1** aplicado a **AgendIO** demuestra una estructura sólida, coherente y alineada con buenas prácticas de desarrollo moderno.  
 La integración entre **Flutter** y **Supabase** permite ofrecer una aplicación ágil, segura y escalable, capaz de cubrir necesidades reales de productividad personal.

El diseño modular facilita futuras ampliaciones, como:

* Incorporación de **inteligencia artificial** para priorizar tareas o sugerir rutinas.
* Envío de **notificaciones push** automatizadas.
* Sincronización multiplataforma completa (web, móvil, escritorio).

En conclusión, **AgendIO** representa una solución integral de gestión personal, sustentada en una arquitectura moderna, flexible y orientada al crecimiento.