# **Plan Integrado de Evolución MUNDERO v2.1**

## **1. Propósito General**

Convertir a **Mundero** en el **hub universal de identidad, acceso y sincronización** del Grupo Servat.  
 Todas las aplicaciones —Legalty, We Consulting, Portales, Pitahaya— **consumen y devuelven datos a Mundero**, quien mantiene la trazabilidad de usuarios, empresas, referidos y comisiones.  
 La prioridad no es agregar complejidad, sino **hacer que todo se conecte y se mantenga vivo sin romper nada.**

## **2. Arquitectura Base (No se toca, se refuerza)**

Ya existe un núcleo sólido con:

* **Autenticación híbrida Firebase + Supabase** Firebase Auth = identidad principal (Google SSO)  
   Supabase = perfiles, roles, relaciones empresa-usuario.
* **Hooks estables**: useAuth y useHybridAuth — seguirán siendo la capa base.
* **Zustand + React 18 + Tailwind + Vite + TypeScript + Framer Motion**.  
   Mantener este stack como *core estable de front*.

### **Refuerzo clave**

* Firebase seguirá autenticando.
* Supabase centralizará datos y relaciones.
* Las demás apps **no autenticaran directamente**: usarán los **tokens o endpoints públicos de Mundero** para acceder a datos de perfil, empresa o referidos.
* Mundero se convierte en el **orquestador de sesión y permisos**.

## **3. Estructura Modular Consolidada**

Basada en el wiki.md

wiki

, los módulos actuales se reorganizan y amplían así:

### **Core (sin cambios)**

* /core/hooks: mantiene useAuth y useHybridAuth.
* /lib: Firebase y Supabase config, sin alteraciones.
* /scripts: protección y sanación de entorno (persistEnv, autoEnvHeal, lockEnv).

### **Módulos funcionales existentes**

| **Módulo** | **Estado actual** | **Mejora integrada** |
| --- | --- | --- |
| **Auth** | Estable | Agregar endpoint /auth/external-login para otras apps del grupo (login federado vía token JWT). |
| **ProfileManager** | Funcional con Supabase | Agregar campos empresa\_id, rol\_global, y endpoint /api/profile/sync para apps externas. |
| **AppGrid** | Muestra apps | Añadir API dinámica /api/apps/list para que cada app registrada publique su metadata en Mundero. |
| **AdminPanel** | Base sólida | Incluir vista de integraciones y control de roles por app. |
| **Messaging** | Interna | Mantener como opcional. Posterior migración a socket interno. |

### **Nuevos módulos**

| **Módulo** | **Descripción** |
| --- | --- |
| **/modules/Referrals/** | Controla referidos (empresa, contacto, estado). Expone endpoints POST /referral, GET /referrals/:userId. |
| **/modules/Commissions/** | Muestra y actualiza comisiones por app. Cada app puede enviar actualizaciones vía webhook POST /api/events/lead\_converted. |
| **/modules/Agreements/** | Registra la firma digital de cada usuario (hash/IP/version) antes de activar comisiones. |
| **/modules/Integrations/** | Administra conexión bidireccional entre Mundero y otras apps (We, Legalty, etc.). |

## **4. Flujo de Interoperabilidad (el punto crítico)**

### **A. Autenticación**

1. Usuario inicia sesión con **Google en Mundero**.
2. Mundero genera un **JWT temporal** y lo guarda en Supabase.
3. Otras apps (We, Legalty, etc.) consumen https://mundero.app/api/auth/external-login?token=XYZ.
4. Validan token → crean o sincronizan perfil local (si aplica).
5. Si el usuario actualiza su perfil en Mundero, se envía un **webhook de sincronización** a las apps asociadas.

### **B. Flujo de datos bidireccional**

| **Dirección** | **Qué datos fluyen** | **Método** |
| --- | --- | --- |
| **Mundero → Apps** | Perfil, empresa, roles, firma, idioma, país | Webhook /sync/profile |
| **Apps → Mundero** | Leads, referidos, comisiones, actividad | API POST /api/events/\* |

Cada app **publica y consume solo lo que le toca**. No hay DB cruzadas, solo sincronización por evento.

## **5. Fases Simplificadas de Implementación**

| **Fase** | **Objetivo** | **Módulos activos** |
| --- | --- | --- |
| **1. Estabilización** (2 semanas) | Cerrar autenticación híbrida + perfilar usuarios con empresa\_id y rol\_global. | Auth, ProfileManager |
| **2. Interoperabilidad mínima viable** (2 semanas) | JWT federado + endpoint /external-login + registro de apps en AppGrid. | Auth, AppGrid, Integrations |
| **3. Referidos y Comisiones** (3 semanas) | Formularios, firmas, panel de comisiones y webhooks activos. | Referrals, Commissions, Agreements |
| **4. Escalabilidad y UI** (2 semanas) | Feed corporativo, selector de idioma/moneda, optimización visual. | Integrations, Settings |

## **6. API Central (documentación resumida)**

### **Endpoints Clave**

| **Endpoint** | **Método** | **Descripción** |
| --- | --- | --- |
| /api/auth/external-login | POST | Autenticación federada desde apps del ecosistema. |
| /api/profile/sync | POST | Sincronización de perfil hacia otras apps. |
| /api/events/lead\_converted | POST | Reporte de conversión de lead (activación de comisión). |
| /api/referrals | GET/POST | Gestión de referidos. |
| /api/commissions/:userId | GET | Histórico de comisiones. |
| /api/integrations/register | POST | Registro de nuevas apps del ecosistema Servat. |

## **7. Principios de Gobernanza Técnica**

* **Mundero es el único punto de verdad para usuarios, empresas y referidos.**
* **Las demás apps solo consumen o devuelven datos**, nunca almacenan identidad.
* **No se repite autenticación**. Si ya estás en Mundero, estás dentro.
* **Las apps se sincronizan por eventos**, no por consultas directas.
* **Toda integración nueva requiere registro formal en /integrations**.

## **8. Entregable final**

Una **infraestructura viva**:

* Usuarios entran una vez (Google → Mundero).
* Mundero orquesta accesos, datos y firmas.
* Apps sincronizan automáticamente.
* Roles, comisiones y referidos trazables en una sola fuente.

¿Deseas que lo deje preparado como documento README\_OPERATIVO\_MUNDERO.md para integrarlo directamente al repo (con tabla de endpoints, fases y arquitectura)? Puedo formatearlo para MGX o Supabase Docs según prefieras.

Fuentes

##### **Dijiste:**

explica que esta fase la haremos mediante mockups para ver como funciona y se ve todo el ux

###### **ChatGPT dijo:**

Perfecto. Así se incorporaría formalmente esa etapa en el plan, **dejando claro que el siguiente paso no es programar, sino visualizar**. Aquí tienes la versión actualizada del cierre del documento (puede ir al final o justo antes de las fases de implementación):

## **9. Fase de Validación UX/UI con Mockups (Antes del desarrollo final)**

Antes de escribir una sola línea de código, realizaremos una **etapa de validación integral mediante mockups interactivos**.  
 Esta fase permitirá **simular el comportamiento completo del sistema** —autenticación, navegación, referidos, comisiones e integración entre apps— para validar que la experiencia sea fluida, lógica y visualmente coherente.

### **Objetivo**

Visualizar el **flujo completo del usuario dentro de Mundero y su interacción con las apps del ecosistema Servat**, garantizando que todo tenga sentido antes de implementarlo técnicamente.

### **Qué se validará**

1. **Flujo de ingreso y autenticación híbrida** Cómo el usuario entra con Google, cómo se registra su perfil en Supabase, y cómo las otras apps lo reconocen desde Mundero.
2. **Experiencia de navegación centralizada** Qué ve un usuario al ingresar: panel, feed, apps disponibles, empresa asociada y su rol global.
3. **Sistema de referidos y comisiones** Interacción visual del usuario al registrar un referido, firmar el acuerdo y visualizar su comisión en el panel.
4. **Panel de integraciones y flujo de datos entre apps** Cómo se representan las conexiones con Legalty, We Consulting, Portales, Pitahaya, y cómo se muestra el intercambio de información.
5. **UX visual de ajustes personales y empresariales** Preferencias de idioma, país, moneda, y vista general de empresa y roles.

### **Herramientas y entregables**

* **Figma o Framer**: desarrollo de mockups interactivos por flujo (Auth, Profile, Referrals, Integrations, etc.).
* **Mapa de navegación (User Flow)** con pantallas enlazadas.
* **Guía visual de componentes UI**: tarjetas, botones, modales, loaders, etc.
* **Prototipo navegable** para revisión interna y aprobación antes del desarrollo (versión MVP).

### **Criterio de cierre**

La fase de mockups se considera completada cuando:

* Cada módulo (Auth, Profile, Company, Referrals, Commissions, Integrations) tiene su flujo validado.
* Todos los estados visuales (loading, success, error, empty) están definidos.
* El equipo Servat y el equipo técnico aprueban el flujo y estilo visual.

Solo entonces se pasa a **implementación técnica directa**, minimizando retrabajos y asegurando que el código refleje exactamente la intención UX validada.

¿Quieres que te deje este bloque ya integrado dentro del documento operativo (README\_OPERATIVO\_MUNDERO.md) como **sección 9 “Fase de Validación UX/UI”** y te prepare un esquema de mockups por módulo (Auth, Profile, Company, Referrals, Integrations, etc.) para pasarlo a Figma? Puedo estructurarlo con nombres de pantallas y descripciones para que el diseñador arranque sin interpretación.

##### **Dijiste:**

no veo que el panel del admin hallas habaldo que es quien aprueba o no que usuario va a que empresa o aplicatovo ni tampoco veo los requisitos estandarizados para la api de integracion con las appas

###### **ChatGPT dijo:**

Perfecto, Gabriel. Te explico cómo se integran **esas dos piezas que faltaban** dentro del plan y por qué son claves para cerrar el círculo operativo de Mundero:

## **10. Panel Administrativo — Gobernanza de Accesos y Aprobaciones**

### **Rol y Propósito**

El **Panel del Administrador** es el **órgano de control central** de Mundero.  
 No solo gestiona usuarios, sino que **aprueba las relaciones** entre:

* Usuario ↔ Empresa
* Usuario ↔ Aplicativo

Las empresas las crean los usuarios en mundero, lo que si le da acceso a los aplicativos sean como colaboradores o clientes y a las empresas de alta es el admin de mundero, pero el registro inicial lo puede hacer cualquier persona y el ruc es su identificador universal

Es decir: ningún usuario entra a una app del ecosistema sin ser **asignado o aprobado desde este panel.**

### **Funciones Clave del Panel Admin**

1. **Gestión de Empresas**
   * Crear nuevas empresas o importar desde Supabase.
   * Ver responsables, analistas y referidores asociados.
   * Asignar o revocar roles globales (owner, admin, analyst, referrer).
2. **Aprobación de Accesos**
   * Visualizar solicitudes de usuarios para unirse a una empresa o aplicativo.
   * Aprobar o rechazar con un clic (flujo con estado: *pendiente → aprobado → activo*).
   * Generar notificación automática a la app correspondiente.
3. **Gestión de Aplicativos Vinculados**
   * Panel de integraciones activas (Legalty, We, Portales, Pitahaya).
   * Control de roles específicos por aplicativo.
   * Activar o pausar acceso global a una app sin alterar su autenticación local.
4. **Auditoría y Logs**
   * Registro de cada acceso, alta o baja de usuario.
   * Trazabilidad completa de quién aprobó qué y cuándo.
   * Exportable en CSV o JSON (para revisión ISO o SGC).

## **11. Estándares API de Integración entre Mundero y las Apps del Ecosistema**

### **Propósito**

Garantizar que todas las aplicaciones del Grupo Servat se integren de forma **segura, coherente y trazable**, sin reinventar la rueda en cada caso.

### **A. Requisitos de Autenticación**

Toda app integrada debe:

Consumir el endpoint de autenticación:  
  
 POST https://mundero.app/api/auth/external-login

* con token JWT emitido por Firebase y validado por Supabase.
* Almacenar solo el user\_uuid y empresa\_id como claves de referencia.
* Validar tokens mediante public\_key publicada por Mundero.

### **B. Estructura Base de Datos de Integración**

Cada app debe tener en su BD local una tabla espejo:

CREATE TABLE mundero\_sync (

user\_uuid UUID PRIMARY KEY,

empresa\_id UUID,

rol\_global TEXT,

last\_sync TIMESTAMP,

sync\_status TEXT DEFAULT 'active'

);

Esto permite sincronización unidireccional sin duplicar datos sensibles.

### **C. Webhooks Estándar**

| **Evento** | **Dirección** | **Endpoint** | **Descripción** |
| --- | --- | --- | --- |
| profile.updated | Mundero → App | /api/mundero/webhook/profile | Actualiza datos de perfil. |
| referral.created | App → Mundero | /api/events/referral\_created | Registra nuevo referido. |
| lead.converted | App → Mundero | /api/events/lead\_converted | Confirma conversión de lead. |
| commission.paid | App → Mundero | /api/events/commission\_paid | Registra pago de comisión. |
| integration.status | App ↔ Mundero | /api/integrations/status | Confirma que la app está sincronizando. |

Cada webhook debe incluir encabezado x-mundero-signature con firma HMAC (256) y payload JSON estructurado.

### **D. Payload JSON Estándar**

Ejemplo:

{

"event": "lead.converted",

"user\_uuid": "a31b9c92-9135-4f39-bc2f-900b99a42f9d",

"empresa\_id": "45ffcd70-62e7-4313-8f2f-bc244d109d5b",

"app\_source": "we\_consulting",

"data": {

"lead\_id": "2025-00045",

"status": "validated",

"commission\_amount": 2000

},

"timestamp": "2025-10-27T18:00:00Z"

}

### **E. Versionado y Compatibilidad**

* Todas las integraciones se rigen por un **manifest JSON** que se registra en Mundero:

{

"app\_name": "we\_consulting",

"version": "1.1.0",

"base\_url": "https://weconsulting.app",

"webhooks": [

"profile.updated",

"lead.converted",

"commission.paid"

]

}

* Este manifest se publica en /api/integrations/register y permite que Mundero **valide compatibilidad antes de conectar**.

### **F. Validación de Integraciones**

Antes de que una app pase a producción:

1. Prueba de handshake (/ping de ambas direcciones).
2. Validación de JWT y HMAC.
3. Confirmación de logs y estado activo en Admin Panel.
4. Aprobación manual del **Administrador de Mundero**.

Con esto queda claro que:

* **El panel administrativo controla accesos humanos.**
* **La API de integración controla accesos entre sistemas.** Ambos flujos se cruzan bajo la **gobernanza de Mundero**, que autentica, autoriza y registra cada relación.