CENTERMIND

Documento de Proyecto · Referencia Técnica · Hoja de Ruta · Lecciones Aprendidas

Versión actualizada — Febrero 2026

Estado: Fase 4 en progreso — Streamlit App operativa (Login + Visor + Dashboard + Admin)

1. ARQUITECTURA GENERAL

Componentes del sistema

Componente	Ubicación	Descripción
centermind_core.py	Servidor	Orquestador: levanta un BotWorker por cada distribuidora activa en un hilo separado. Auto-reinicio hasta 10 veces con delay de 15s.
bot_worker.py	Servidor	Bot de Telegram por distribuidora. Recibe fotos, sube a Drive, registra en SQLite, y sincroniza evaluaciones cada 30s.
StreamLitApp/	Servidor	Aplicación web Streamlit multipage: login unificado, visor, dashboard TV y panel admin. Accesible desde el navegador.
centermind.db	Servidor	Base de datos SQLite. Única fuente de verdad. NO tocar con DB Browser mientras el bot corre.
Google Drive	Nube	Almacén de fotos. Estructura: Distribuidora / Grupo / DD-MM-YYYY / foto.jpg
Telegram API	Nube	Canal de comunicación con los vendedores. Un bot por distribuidora para evitar saturar cuota.

Flujo completo de una exhibición

Paso	Actor	Qué ocurre
1	Vendedor	Envía foto al grupo de Telegram de su distribuidora.
2	BotWorker	Solicita NRO CLIENTE (solo números). Acepta hasta 5 fotos en ráfaga (8s).
3	BotWorker	Muestra botones inline con los tipos de PDV configurados (PDV_TYPES en bot_worker.py).
4	BotWorker	Descarga la foto de Telegram y la sube a Google Drive via OAuth2.
5	BotWorker	Registra la exhibición en SQLite con estado='Pendiente' y synced_telegram=0.
6	BotWorker	Envía mensaje de confirmación al grupo con stats del mes del vendedor (aprobadas, rechazadas, puntos, racha).
7	Supervisor	Abre el Visor en Streamlit, se loguea, y evalúa la foto: Aprobado / Rechazado / Destacado.

8	Visor	Actualiza estado en SQLite y pone synced_telegram=0 para marcar la notificación pendiente.
9	BotWorker	Job cada 30s detecta synced_telegram=0, edita el mensaje original en Telegram con el resultado.
10	BotWorker	Marca synced_telegram=1. El vendedor ve el resultado en el grupo.

2. ESTRUCTURA DE ARCHIVOS

Ruta	Estado	Descripción
CenterMind/bot_worker.py	[OK]	Bot por distribuidora. Flujo completo foto → Drive → SQLite → sync Telegram.
CenterMind/centermind_core.py	[OK]	Orquestador multi-bot. Un hilo por distribuidora activa.
CenterMind/setup_drive_oauth.py	[OK]	Setup OAuth2 con Google Drive. Correr UNA SOLA VEZ por servidor.
CenterMind/credencial_oauth.json	[CRITICO]	Client secrets de Google Cloud. NUNCA subir a git.
CenterMind/token_drive.json	[ALTO]	Token OAuth2 generado por setup_drive_oauth.py. Auto-refresh si expira.
CenterMind/base_datos/centermind.db	[CRITICO]	Base de datos SQLite. Hacer backup diario. NO abrir con DB Browser mientras el bot corre.
StreamLitApp/app.py	[OK]	Entry point Streamlit: login unificado + menú de servicios con cards clicables.
StreamLitApp/pages/1_Visor.py	[OK]	Evaluación de exhibiciones pendientes. Antes llamado visor_streamlit.py.
StreamLitApp/pages/2_Dashboard.py	[OK]	Dashboard TV modo pantalla grande. Antes llamado 02_dashboard.py.
StreamLitApp/pages/3_Admin.py	[OK]	Panel admin: CRUD usuarios del portal + gestión de roles de Telegram.

▲ **Para correr el sistema: streamlit run CenterMind\StreamLitApp\app.py (desde BOT-SQL\CenterMind\)**

3. BASE DE DATOS — ESQUEMA COMPLETO

■ **IMPORTANTE: El esquema fue creado antes del código. Los nombres de columnas en la DB difieren de los usados en Python. La clase Database en bot_worker.py hace el mapeo con aliases SQL (AS nombre). No renombrar columnas en la DB sin actualizar los queries en el código.**

distribuidores

Columna en DB	Alias en código	Tipo	Notas
id_distribuidor	id	INTEGER PK	Autoincremental.
nombre_empresa	nombre	TEXT	Nombre del cliente / distribuidora.
token_bot	token_bot	TEXT	Token de BotFather. ÚNICO por distribuidora.
ruta_credencial_drive	—	TEXT	No se usa. Credencial es global (OAuth2 único).
id_carpeta_drive	drive_folder_id	TEXT	ID de la carpeta raíz en Drive para subir fotos.
estado	estado	TEXT	'activo' o 'inactivo'. Solo los activos generan bots.
admin_telegram_id	admin_telegram_id	TEXT	Telegram user ID que recibe notificaciones del bot.

usuarios_portal

Columna en DB	Tipo	Notas
id_usuario	INTEGER PK	Autoincremental.
id_distribuidor	INTEGER FK	FK a distribuidores. El usuario pertenece a una sola distribuidora.
usuario_login	TEXT UNIQUE	Nombre de usuario para el login en Streamlit. Único en toda la tabla.
password	TEXT	Contraseña en texto plano (sin hash por ahora). Mejorar en Fase 6.
rol	TEXT	'superadmin' | 'admin' | 'evaluador'. Ver tabla de permisos abajo.

Tabla de roles — usuarios_portal

Rol	Visor	Dashboard	Admin	Descripción
superadmin	✓	✓	✓	Dueño del servidor. Acceso total incluido el panel admin.
admin	✓	✓	✗	Supervisor de distribuidora. Puede evaluar y ver dashboard.
evaluador	✓	✓	✗	Solo puede evaluar exhibiciones y ver el dashboard.

integrantes_grupo

Columna en DB	Alias en código	Tipo	Notas
id_integrante	pk_integrante	INTEGER PK	PK autoincremental. Es la FK que usa exhibiciones.id_integrante. NO confundir con telegram_user_id.
id_distribuidor	distribuidor_id	INTEGER FK	FK a distribuidores.
telegram_user_id	user_id	INTEGER	ID numérico del usuario en Telegram. Se registra automáticamente cuando el usuario envía una foto.
nombre_integrante	nombre	TEXT	Primer nombre del usuario en Telegram.
rol_telegram	rol	TEXT	'vendedor' (puede cargar fotos) | 'observador' (ignorado por el bot). Cambiar desde panel Admin.
telegram_group_id	chat_id	INTEGER	ID del grupo de Telegram donde opera el integrante.
nombre_grupo	—	TEXT	Título del grupo en Telegram. Solo referencial.

exhibiciones

■ **comentarios_telegram guarda el tipo de PDV. Nombre confuso y heredado del diseño original. Corregir en futura migración.**

■ **id_integrante en exhibiciones es el PK autoincremental de integrantes_grupo, NO el telegram_user_id. bot_worker.py resuelve el PK antes de insertar.**

Columna en DB	Alias en código	Tipo	Notas
`id_exhibicion`	id	INTEGER PK	Autoincremental. NO insertar UUID aquí.
`id_distribuidor`	distribuidor_id	INTEGER FK	FK a distribuidores.
`id_grupo`	chat_id	INTEGER	Grupo de Telegram donde se cargó la foto.
`id_integrante`	vendedor_id	INTEGER FK	FK al PK de integrantes_grupo. bot_worker.py resuelve el PK antes del INSERT.
`numero_cliente_local`	nro_cliente	TEXT	Número de cliente ingresado por el vendedor (solo números).
`comentarios_telegram`	tipo_pdv	TEXT	■ Guarda el TIPO DE PDV seleccionado. Nombre heredado — no es un comentario.
`url_foto_drive`	drive_link	TEXT	webViewLink de Google Drive a la foto subida.
`estado`	estado	TEXT	'Pendiente' → 'Aprobado' | 'Rechazado' | 'Destacado'
`supervisor_nombre`	supervisor_nombre	TEXT	Nombre del usuario del portal que evaluó.
`comentarios`	comentarios	TEXT	Comentario opcional del supervisor al evaluar.
`telegram_msg_id`	telegram_msg_id	INTEGER	ID del mensaje en Telegram para editar con el resultado.
`telegram_chat_id`	telegram_chat_id	INTEGER	Chat donde vive ese mensaje (puede diferir de id_grupo).
`synced_telegram`	synced_telegram	INTEGER	0 = hay notificación pendiente | 1 = ya sincronizado. El job del bot lo monitorea cada 30s.
`timestamp_subida`	created_at	DATETIME	Momento en que el vendedor cargó la foto (UTC).
`evaluated_at`	evaluated_at	DATETIME	Momento en que el supervisor evaluó (UTC).

4. STREAMLIT APP — DESCRIPCIÓN DE INTERFACES

La aplicación se levanta con un solo comando y gestiona todas las interfaces desde un login centralizado.

`Comando: streamlit run CenterMind\StreamLitApp\app.py`

app.py — Login y Menú Principal

Aspecto	Detalle
Función	Punto de entrada único. Maneja el login y redirige al servicio elegido.
Autenticación	Consulta tabla usuarios_portal. Verifica usuario + password. Guarda datos del usuario en st.session_state.
Menú	Muestra cards clicables (grandes, 260px) según el rol del usuario. superadmin ve 3 cards (Visor + Dashboard + Admin). admin y evaluador ven 2 cards.
Sesión	session_state['logged_in'] y session_state['user'] persisten entre páginas. Cada página arranca con un guard que redirige al login si no hay sesión activa.
Datos que carga	Solo consulta usuarios_portal y distribuidores para validar el login. No toca exhibiciones.
Datos que NO carga	No carga fotos, rankings ni exhibiciones. Es solo autenticación y navegación.

1_Visor.py — Evaluación de Exhibiciones

Aspecto	Detalle
Función	Herramienta de trabajo diario del supervisor. Muestra las fotos pendientes y permite evaluarlas.
Acceso	evaluador, admin y superadmin. Guard redirige a app.py si no hay sesión.
Datos que carga	exhibiciones WHERE estado='Pendiente' AND id_distribuidor=? — Agrupa por telegram_msg_id para mostrar ráfagas juntas. También carga integrantes_grupo para mostrar nombre del vendedor.
Datos que NO carga	No carga exhibiciones de otras distribuidoras. No accede a Drive directamente: usa iframe embed con la URL guardada en url_foto_drive.
Evaluar	UPDATE exhibiciones SET estado=?, supervisor_nombre=?, comentarios=?, evaluated_at=NOW(), synced_telegram=0 WHERE id_exhibicion IN (?). El job del bot detecta synced_telegram=0 en los próximos 30s y notifica en Telegram.
Ráfagas	Fotos del mismo telegram_msg_id se agrupan en un solo card. Se evalúan todas juntas con el mismo estado y comentario. Navegación interna: flechas ← → + thumbnails.
Filtro	Panel expandible para filtrar por vendedor. Solo muestra vendedores con pendientes activos.
Stats en tiempo real	Contadores del día: Pendientes / Aprobadas / Rechazadas / Destacadas. Se actualizan en cada evaluación.

2_Dashboard.py — Modo TV / Ranking

Aspecto	Detalle
Función	Pantalla de motivación para la oficina. Ranking en tiempo real, KPIs del período y carrusel de últimas fotos aprobadas.

Acceso	evaluador, admin y superadmin. Guard redirige a app.py si no hay sesión.
Auto-refresh	Cada 60 segundos recarga todos los datos automáticamente. Barra de progreso en el topbar indica cuánto falta para el próximo refresh. Botón 'FORZAR' para actualizar manualmente.
Datos que carga	exhibiciones para KPIs (totales por estado) y ranking (agrupado por vendedor). integrantes_grupo para nombres. exhibiciones WHERE estado IN ('Aprobado','Destacado') para el carrusel (últimas 8).
Datos que NO carga	No carga exhibiciones pendientes ni rechazadas (solo aprobadas/destacadas en el carrusel). No permite evaluar.
Selector de período	HOY / MES / HISTÓRICO. Recalcula KPIs y ranking al cambiar. Botones en la topbar.
Carrusel	Últimas 8 fotos aprobadas/destacadas. Iframe embed de Drive. Overlay con nombre del vendedor, cliente y tipo PDV. Dots de posición. Flechas ← → manuales.
Ranking	Top 15 vendedores. Posiciones con colores: #1 dorado, #2 plateado, #3 bronce. Muestra aprobadas y puntos (aprobadas + destacadas). Banner '¡NUEVO LIDER!' si cambia el 1er puesto entre refreshes.
KPIs	3 cards: Aprobadas (verde), Destacadas (ámbar), Rechazadas (rojo). Total y pendientes al pie.

3_Admin.py — Panel de Administración

Aspecto	Detalle
Función	CRUD completo de usuarios del portal y gestión de roles de vendedores de Telegram.
Acceso	SOLO superadmin. Guard doble: primero verifica sesión activa, luego verifica rol='superadmin'. Cualquier otro rol es redirigido al login.
Tab: Usuarios del portal	Lista todos los usuarios con filtro por distribuidora. Botones Editar y Borrar por fila. Formulario de nuevo usuario (distribuidora, login, password, rol). Confirmación antes de borrar. No se puede eliminar al propio usuario logueado.
Tab: Integrantes Telegram	Lista todos los vendedores registrados automáticamente por el bot. Filtro por distribuidora. Botón por fila que alterna directamente vendedor ↔ observador. Los observadores son ignorados por el bot (sus fotos no se procesan).
Datos que carga	usuarios_portal con JOIN a distribuidores. integrantes_grupo con JOIN a distribuidores.
Datos que NO carga	No carga exhibiciones, rankings ni fotos. Solo gestión de accesos y roles.
Cuándo se actualiza la DB	Inmediatamente al confirmar el formulario. Cada operación hace conn.commit() en el momento. SQLite en modo WAL permite escritura concurrente con el bot.

5. HOJA DE RUTA

Fase 1 — Infraestructura base

COMPLETADO

Estructura de carpetas, base de datos SQLite con tablas relacionales, datos de prueba, Google Cloud OAuth2 Desktop configurado, carpeta Drive autorizada.

Fase 2 — Bot de Telegram

COMPLETADO — testing OK

bot_worker.py con flujo completo foto → Drive → SQLite. centermind_core.py con reinicio automático. Ráfaga de fotos (hasta 5 en 8s). Hibernación configurable. Sync job cada 30s. Stats del mes en el mensaje de confirmación.

- ■ PENDIENTE MENOR: mes en el mensaje aparece en inglés — corregir con dict manual (parche disponible).

Fase 3 — Streamlit: Visor de evaluación

COMPLETADO — operativo

Login con usuarios_portal. Vista de pendientes con foto via iframe de Drive. Botones Aprobar / Rechazar / Destacado con comentario opcional. Agrupación de ráfagas. Filtro por vendedor. Stats del día en tiempo real.

- ■ PENDIENTE: pulir estética general (detalles visuales).
- ■ PENDIENTE: confirmación antes de evaluar (evitar errores de click).
- ■ PENDIENTE: filtros adicionales por fecha y tipo PDV.

Fase 4 — Streamlit: Dashboard, Admin y Login unificado

EN PROGRESO — 80% completado

Login unificado con menú de servicios (app.py). Dashboard modo TV con ranking, KPIs y carrusel (2_Dashboard.py). Panel admin CRUD usuarios y roles Telegram (3_Admin.py).

- ✓ COMPLETADO: app.py — login + menú con cards clicables según rol.
- ✓ COMPLETADO: 2_Dashboard.py — ranking, KPIs, carrusel, auto-refresh 60s.
- ✓ COMPLETADO: 3_Admin.py — CRUD usuarios portal + roles Telegram.
- ■ PENDIENTE: Reportes exportables CSV/Excel por período y filtros.
- ■ PENDIENTE: Mobile-friendly del visor.

Fase 5 — Portal Exhibiciones (.exe cliente)

REEMPLAZADA — App Streamlit en servidor

La decisión fue mantener todo en Streamlit corriendo en el servidor en lugar de distribuir ejecutables. Los clientes acceden desde el navegador. Ventajas: sin instalación, actualizaciones instantáneas, multiplataforma.

Fase 6 — Hardening y producción

PENDIENTE

Logs centralizados con rotación. Backup automático de centermind.db. Monitoreo de uptime de los bots. Script de onboarding para nuevas distribuidoras. Hasheo de contraseñas en usuarios_portal.

6. CÓMO CORRER EL SISTEMA

Comando	Para qué
`pip install python-telegram-bot google-auth google-auth-oauthlib google-api-python-client pillow certifi streamlit`	Instalar dependencias (UNA SOLA VEZ).
`python setup_drive_oauth.py`	Autorizar Google Drive con OAuth2 (UNA SOLA VEZ por servidor). Genera token_drive.json.
`python bot_worker.py --distribuidor-id 1`	Levantar UN solo bot (debug / testing).
`python centermind_core.py`	Levantar TODOS los bots (producción). Un hilo por distribuidora activa.
`streamlit run CenterMind\StreamLitApp\app.py`	Levantar la app web. Ejecutar desde el directorio BOT-SQL\CenterMind\.

7. LECCIONES APRENDIDAS — TRAMPAS YA ENCONTRADAS

Error / Síntoma	Causa	Solución
database is locked	DB Browser abierto al correr el bot.	Cerrar DB Browser antes de levantar el bot.
no such column: id	Columna real es id_distribuidor.	Aliases SQL en queries (id_distribuidor AS id).
no such column: rol	Columna real es rol_telegram.	Aliases SQL en queries (rol_telegram AS rol).
no such column: distribuidor_id	En exhibiciones es id_distribuidor.	Corregido en queries con alias.
unable to open database file	DB_PATH apunta a la carpeta equivocada. Calcular los .parent desde __file__.	app.py: BASE_DIR.parent.parent. pages/*.py: BASE_DIR.parent.parent.parent — verificar siempre con print(DB_PATH).
StreamlitAPIException: Could not find page	app.py estaba dentro de pages/ en lugar de StreamLitApp/.	app.py debe vivir en StreamLitApp/, NO en pages/.
JobQueue.run_daily() unexpected keyword 'timezone'	Esta versión de PTB no acepta timezone como kwarg separado.	Pasar timezone en el objeto time: dt_time(22, 0, tzinfo=AR_TZ).
ADD COLUMN IF NOT EXISTS — syntax error	DB Browser usa SQLite < 3.37.	Ejecutar ALTER TABLE sin IF NOT EXISTS, ignorar 'duplicate column'.
Error 400: redirect_uri_mismatch	Se creó credencial tipo 'Aplicación web'.	Crear nueva credencial de tipo 'Aplicación de escritorio' (Desktop app).
Error 403: access_denied	App en modo Testing, email no está en usuarios de prueba.	Pantalla de consentimiento OAuth → Usuarios de prueba → agregar email.
Could not find TLS CA certificate bundle	PostgreSQL sobreescribe SSL_CERT_FILE del sistema.	Fix con certifi en setup_drive_oauth.py. Equipo nuevo: PowerShell Admin → reinstalar PostgreSQL o setear SSL_CERT_FILE.
datatype mismatch (INSERT exhibiciones)	Código insertaba UUID (texto) en id_exhibicion que es INTEGER PK autoincremental.	Omitir id_exhibicion del INSERT, usar lastrowid como new_id.
FOREIGN KEY constraint failed (INSERT exhibiciones)	Se pasaba telegram_user_id como id_integrante, pero la FK espera el PK autoincremental de integrantes_grupo.	En registrar_exhibicion: resolver PK via SELECT id_integrante FROM integrantes_grupo WHERE telegram_user_id=?.
Ranking muestra HTML crudo en Streamlit	Variables Python con HTML y emojis dentro de f-strings se escapan si están anidadas en strings grandes.	Renderizar cada fila con su propio st.markdown() individual. Usar entidades HTML (■) en lugar de emojis directos.
Bot ignora fotos de noche	Hibernación activa (HIBERNACION_ACTIVA = True).	Setear HIBERNACION_ACTIVA = False en bot_worker.py para testing.

8. ARCHIVOS CRÍTICOS

Archivo	Criticidad	Qué hacer si se pierde
`base_datos/centermind.db`	CRÍTICO	Se pierden TODOS los datos. Hacer backup diario obligatorio.

`credencial_oauth.json`	CRÍTICO	Sin esto no se suben fotos a Drive. Regenerar en Google Cloud Console → APIs → Credenciales.
`token_drive.json`	ALTO	Correr setup_drive_oauth.py para regenerarlo. Se auto-regenera si expira.
`token_bot` en `distribuidores`	ALTO	Crear bots nuevos en @BotFather y actualizar el campo token_bot en la tabla distribuidores.
`bot_worker.py` / `centermind_core.py`	MEDIO	Están en el repositorio git, se recuperan con git pull.
`StreamLitApp/*.py`	MEDIO	Están en el repositorio git, se recuperan con git pull.

9. GLOSARIO

Término	Significado
Distribuidora	Cliente del sistema. Tiene su propio bot de Telegram, carpeta en Drive y usuarios del portal.
PDV	Punto de Venta (kiosco, supermercado, almacén, etc.). Lugar donde el vendedor toma la foto.
Exhibición	Una carga de foto por un vendedor, con sus metadatos (cliente, tipo PDV, estado, evaluador).
Ráfaga	Hasta 5 fotos enviadas en 8 segundos que se agrupan en una sola exhibición lógica.
Integrante	Usuario de Telegram registrado automáticamente en un grupo cuando envía su primera foto.
Racha	Cantidad de exhibiciones consecutivas aprobadas/destacadas del vendedor. Se muestra en el bot si es >= 2.
Puntos	Aprobadas + Destacadas×2. Métrica de rendimiento mensual del vendedor usada en el ranking.
synced_telegram	Flag en exhibiciones. 0 = el bot debe notificar el resultado en Telegram. 1 = ya notificó.
Guard de sesión	Bloque de código al inicio de cada página Streamlit que redirige al login si no hay sesión activa.
superadmin	Rol exclusivo del dueño del servidor. Acceso completo incluyendo panel Admin.
WAL	Write-Ahead Log. Modo de SQLite que permite lectura/escritura concurrente. Activo en todo el sistema.