SHELFMIND

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

Versión 2.0 — Febrero 2026

Estado: Fase 4 completada — Fase 6 (Hardening) en progreso

Anteriormente: CenterMind · Renombrado a ShelfMind

Identidad de marca — Contexto

ShelfMind nace del concepto **Shelf Mindset**: la mentalidad orientada a ganar la góndola. En retail, la mayoría de las decisiones de compra se toman frente al anaquel — el consumidor decide en segundos mirando la góndola. ShelfMind es el sistema que documenta, evalúa y optimiza la ejecución en punto de venta para que los distribuidores compitan con inteligencia en el estante.

1. ARQUITECTURA GENERAL

Componentes del sistema

Componente	Ubicación	Descripción
`centermind_core.py`	Servidor	Orquestador: levanta un BotWorker por distribuidora activa en hilo separado. Auto-reinicio hasta 10 veces, delay 15s. Integra hardening (logs + backup + monitor).
`bot_worker.py`	Servidor	Bot de Telegram por distribuidora. Recibe fotos, sube a Drive, registra en SQLite, sincroniza evaluaciones cada 30s. Envía heartbeat al monitor.
`hardening/`	Servidor	Módulo de producción: logger.py (logs diarios), backup_manager.py (backup automático + Drive), monitor.py (uptime + alertas Telegram).
`StreamLitApp/`	Servidor	App web Streamlit multipage: login unificado, visor, dashboard TV, admin y reportes. Accesible desde el navegador.
`centermind.db`	Servidor	Base de datos SQLite. Única fuente de verdad. NO tocar con DB Browser mientras el bot corre (database is locked).
`Google Drive`	Nube	Almacén de fotos. Estructura: Distribuidora / Grupo / DD-MM-YYYY / foto.jpg. La DB solo guarda el link, no la foto.
`Telegram API`	Nube	Canal de comunicación con los vendedores. Un bot por distribuidora para evitar saturar cuota de la API.

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 (8 segundos).
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: estado='Pendiente', synced_telegram=0.
6	BotWorker	Envía confirmación al grupo con stats del mes del vendedor (aprobadas, rechazadas, puntos, racha).
7	Supervisor	Abre el Visor en Streamlit, se loguea y evalúa: Aprobado / Rechazado / Destacado.
8	Visor	Actualiza estado en SQLite. Pone synced_telegram=0 para marcar 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. Integrado con BotMonitor.
CenterMind/centermind_core.py	[OK]	Orquestador multi-bot. Integra hardening completo: logs, backup y monitor.
CenterMind/setup_drive_oauth.py	[OK]	Setup OAuth2 Google Drive. Correr UNA SOLA VEZ por servidor.
CenterMind/credencial_oauth.json	[CRIT]	Client secrets de Google Cloud. NUNCA subir a git.
CenterMind/token_drive.json	[ALTO]	Token OAuth2. Auto-refresh si expira. Regenerar con setup_drive_oauth.py si se pierde.
CenterMind/base_datos/centermind.db	[CRIT]	Base de datos SQLite. Backup diario automático via hardening/backup_manager.py.
CenterMind/hardening/__init__.py	[OK]	Paquete hardening. Exporta setup_logging, BackupManager, BotMonitor.
CenterMind/hardening/logger.py	[OK]	Logs diarios con rotación. Archivos en CenterMind/logs/centermind_YYYY-MM-DD.log. 30 días de historial.
CenterMind/hardening/backup_manager.py	[OK]	Backup automático a medianoche. Retención 7 días local. Empaqueta en ZIP y sube a Drive al llegar a 30 backups.
CenterMind/hardening/monitor.py	[OK]	Monitor de uptime cada 60s. Alerta por Telegram si un bot no da señal en 180s.
CenterMind/logs/	[AUTO]	Carpeta creada automáticamente al arrancar. Contiene los archivos .log diarios.
CenterMind/backups/	[AUTO]	Carpeta creada automáticamente. Contiene copias .db fechadas y ZIPs para subir a Drive.
StreamLitApp/app.py	[OK]	Entry point Streamlit: login unificado + menú con cards clicables según rol.
StreamLitApp/pages/1_Visor.py	[OK]	Evaluación de exhibiciones pendientes con agrupación de ráfagas y stats en tiempo real.
StreamLitApp/pages/2_Dashboard.py	[OK]	Dashboard TV: ranking, KPIs y carrusel de fotos aprobadas. Auto-refresh 60s.
StreamLitApp/pages/3_Admin.py	[OK]	Panel admin (solo superadmin): CRUD usuarios del portal + roles de Telegram.
StreamLitApp/pages/4_Reportes.py	[OK]	Reportes: filtros (fecha, vendedor, estado, tipo PDV, cliente), tabla, gráficos y exportar a Excel.

▲ **Comando: streamlit run CenterMind\StreamLitApp\app.py (ejecutar 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.**

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 y del monitor de uptime.

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 — pendiente Fase 6).
rol	TEXT	'superadmin' | 'admin' | 'evaluador'. Ver tabla de permisos.

Roles de usuarios_portal

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

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 en Telegram. Se registra automáticamente cuando el usuario envía su primera foto.

nombre_integrante	nombre	TEXT	Primer nombre del usuario en Telegram.
rol_telegram	rol	TEXT	'vendedor' (procesa 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 — tabla principal del sistema

■ comentarios_telegram guarda el TIPO DE PDV. Nombre confuso y heredado. Corregir en futura migración.

■ id_integrante es el PK de integrantes_grupo, NO el telegram_user_id. bot_worker.py resuelve el PK antes del INSERT.

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.
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. Nombre heredado — no es un comentario.
url_foto_drive	drive_link	TEXT	webViewLink de Google Drive. La foto NO se guarda en la DB, solo el link.
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 a 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 = notificación pendiente | 1 = ya sincronizado. El job del bot lo chequea cada 30s.
timestamp_subida	created_at	DATETIME	Momento en que el vendedor cargó la foto.
evaluated_at	evaluated_at	DATETIME	Momento en que el supervisor evaluó.

4. ALMACENAMIENTO — SQL vs NoSQL Y PROYECCIONES

Qué guarda la DB de cada exhibición

La foto NO se guarda en la base de datos. SQLite solo almacena los metadatos de texto: IDs, nombres, estados, fechas y el link a Google Drive. Esto significa que el crecimiento de la DB es predecible y controlado, independientemente del tamaño o cantidad de fotos.

Campo	Ejemplo	Tamaño aprox.
IDs numéricos (5 campos)	1547, -1001234567890...	~40 bytes
numero_cliente_local	"333"	3–10 bytes
comentarios_telegram (tipo PDV)	"Comercio con Ingreso"	~20 bytes
url_foto_drive	"https://drive.google.com/..."	~80 bytes
estado	"Aprobado"	~8 bytes
supervisor_nombre	"Juan"	4–20 bytes
comentarios del supervisor	"Bien ubicado"	0–100 bytes
timestamps (2 campos)	"2026-02-24 14:30:00"	~40 bytes
synced_telegram + otros flags	0 o 1	~16 bytes
TOTAL datos puros	—	**~330 bytes**
TOTAL real en disco (con overhead SQLite)	—	**500–800 bytes**

Proyecciones de crecimiento

Exhibiciones acumuladas	Tamaño de la DB	Equivale a...
10.000	**~8 MB**	1 año — 1 distribuidora, 5 vendedores, 5 fotos/día
100.000	**~80 MB**	5 años — escenario pequeño
500.000	**~400 MB**	5 años — 20 distribuidoras, 20 vendedores, 5 fotos/día
1.000.000	**~800 MB**	10 años — escenario grande
10.000.000	~8 GB	Límite práctico de SQLite — nunca se llegará en este uso

Escenario real proyectado a 5 años

Elemento	Tamaño en disco
centermind.db (5 años, escenario agresivo)	**< 400 MB**
Logs (rotación 30 días, siempre)	**< 5 MB**
Backups locales (retención 7 días)	**< 3 GB**
Fotos en Google Drive (no en servidor)	250–1250 MB/día
TOTAL en servidor	**< 5 GB para siempre**

Por qué SQLite (SQL) y no NoSQL (MongoDB, Firebase, etc.)

Criterio	SQL — SQLite	NoSQL — MongoDB / Firebase
Estructura de datos	Tablas con relaciones estrictas (FK). Cada exhibición tiene UN vendedor, UNA distribuidora, UN estado. Estructura fija y conocida.	Documentos flexibles. Bueno cuando la estructura cambia constantemente o es muy variable — no es el caso aquí.
Consultas complejas	JOIN nativo entre exhibiciones, integrantes y distribuidoras en una sola query. Rankings, filtros cruzados y stats en código mínimo.	Joins son costosos o inexistentes. Requieren múltiples queries o desnormalización.
Integridad de datos	FOREIGN KEY constraints garantizan que no puede existir una exhibición sin vendedor válido. Imposible crear datos huérfanos.	Sin constraints por defecto. Es responsabilidad del código mantener la integridad.
Escala del proyecto	SQLite soporta hasta 281 TB y millones de filas. Para este volumen es más que suficiente y sin servidor externo.	Necesita servidor externo (MongoDB Atlas, Firebase). Costo mensual, latencia de red, dependencia de internet.
Cero infraestructura	Un solo archivo .db en el servidor. Sin puertos, sin servicios adicionales, sin configuración. Backup = copiar el archivo.	Requiere instalar y mantener un servidor de base de datos separado.
Concurrencia	Modo WAL permite lectura y escritura simultánea. El bot escribe mientras Streamlit lee — sin bloqueos.	Mejor concurrencia para miles de escrituras/segundo — innecesario para este volumen.
Veredicto	**IDEAL para este caso: estructura conocida, consultas relacionales, un solo servidor, volumen moderado.**	Útil si los datos fueran documentos muy variables o si hubiera miles de usuarios simultáneos.

5. STREAMLIT APP — DESCRIPCIÓN DE INTERFACES

`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 en st.session_state.
Menú	Cards clicables (260px) según rol: superadmin ve 4 cards (Visor + Dashboard + Reportes + Admin). admin y evaluador ven 3 cards.
Datos que carga	Solo consulta usuarios_portal y distribuidores. 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 fotos pendientes y permite evaluarlas.
Acceso	evaluador, admin y superadmin.
Datos que carga	exhibiciones WHERE estado='Pendiente' AND id_distribuidor=? — Agrupa por telegram_msg_id para mostrar ráfagas juntas. También carga integrantes_grupo para 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. El job del bot notifica en Telegram en ≤ 30s.
Ráfagas	Fotos del mismo telegram_msg_id agrupadas en un card. Navegación interna: flechas ← → + thumbnails.
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 y carrusel de últimas fotos aprobadas.
Auto-refresh	Cada 60 segundos. Barra de progreso en topbar. Botón FORZAR para actualizar manualmente.
Datos que carga	exhibiciones para KPIs y ranking. integrantes_grupo para nombres. Últimas 8 fotos aprobadas/destacadas para el carrusel.
Selector período	HOY / MES / HISTÓRICO. Recalcula KPIs y ranking al cambiar.
Ranking	Top 15 vendedores. #1 dorado, #2 plateado, #3 bronce. Banner '¡NUEVO LIDER!' si cambia el 1er puesto entre refreshes.

Carrusel	Últimas 8 fotos aprobadas/destacadas. Iframe embed de Drive. Overlay con vendedor, cliente y tipo PDV.

3_Admin.py — Panel de Administración

Aspecto	Detalle
Acceso	SOLO superadmin. Guard doble: sesión activa + rol='superadmin'.
Tab: Usuarios	CRUD completo de usuarios del portal. No se puede eliminar al propio usuario logueado.
Tab: Telegram	Lista todos los integrantes registrados por el bot. Botón por fila que alterna vendedor ↔ observador directamente.
Datos que carga	usuarios_portal JOIN distribuidores. integrantes_grupo JOIN distribuidores.
Datos que NO carga	No carga exhibiciones, rankings ni fotos. Solo gestión de accesos y roles.

4_Reportes.py — Análisis y Exportación

Aspecto	Detalle
Función	Análisis histórico con filtros cruzados. Vista de tabla + gráficos interactivos + exportar a Excel.
Acceso	evaluador, admin y superadmin.
Filtros	Fecha desde/hasta (atajos HOY / MES / TODO), vendedor (multi), estado (multi), tipo PDV (multi), número de cliente (búsqueda parcial).
KPIs del resultado	Total, aprobadas, destacadas, rechazadas, pendientes y puntos del período filtrado.
Gráficos	3 pestañas: torta de distribución por estado, barras apiladas por vendedor (top 15), timeline de cargas por día.
Excel	Archivo .xlsx con fondo oscuro, colores por estado, encabezados dorados y fila de total. Nombre incluye distribuidora y período.
Datos que carga	exhibiciones con JOIN a integrantes_grupo según los filtros seleccionados.
Datos que NO carga	No carga fotos. No permite evaluar.

6. HARDENING — MÓDULOS DE PRODUCCIÓN

hardening/logger.py — Logs centralizados

Aspecto	Detalle
Función	Reemplaza el get_logger() local de bot_worker.py y centermind_core.py con un sistema centralizado.
Archivos	CenterMind/logs/centermind_YYYY-MM-DD.log — uno por día, rotación automática a medianoche.
Retención	30 días de historial. Los logs más viejos se eliminan automáticamente.
Formato	YYYY-MM-DD HH:MM:SS | LEVEL | nombre_módulo | función | mensaje (zona horaria Argentina)
Activar	setup_logging() en centermind_core.py al arrancar. Fallback automático si el módulo no está disponible.

hardening/backup_manager.py — Backup automático

Aspecto	Detalle
Cuándo	Cada medianoche (Argentina). Al arrancar el orquestador si no hay backup del día. Al apagar el orquestador (backup de cierre).
Método	VACUUM INTO (SQLite 3.27+): copia limpia y compacta. Fallback a shutil.copy2 si la versión es antigua.
Retención local	7 días. Los backups más viejos se eliminan automáticamente.
Empaquetado a Drive	Cuando acumula 30 backups: crea un ZIP fechado y lo sube a la carpeta 'CenterMind_Backups' en Google Drive. Elimina los individuales una vez subido.
Backup manual	bm.backup_now() disponible desde el Panel Maestro (GUI futura) o desde código.
Estado	bm.get_status() devuelve: running, total_backups, latest, latest_size_mb, backup_dir.

hardening/monitor.py — Monitor de uptime

Aspecto	Detalle
Chequeo	Cada 60 segundos verifica que cada bot haya enviado heartbeat en los últimos 180 segundos.
Heartbeat	Enviado automáticamente en sync_evaluaciones_job() de bot_worker.py — cada 30s.
Alerta caída	Mensaje Telegram al admin_telegram_id de la distribuidora: bot caído, motivo, reinicio #N.
Alerta recuperación	Mensaje Telegram cuando el bot se recupera: uptime actual, todos los sistemas activos.
Cooldown	10 minutos entre alertas repetidas del mismo bot caído (evita spam).
Estado	monitor.get_summary() devuelve: total, alive, down. monitor.get_all_states() devuelve snapshot completo para el Panel Maestro.

7. 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, carpeta Drive autorizada.

Fase 2 — Bot de Telegram

COMPLETADO — testing OK

bot_worker.py completo: foto → Drive → SQLite → sync Telegram. centermind_core.py con reinicio automático. Ráfaga (5 fotos/8s). Hibernación configurable. Sync job 30s. Stats del mes.

- ■ 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 iframe de Drive. Botones Aprobar / Rechazar / Destacado. Agrupación de ráfagas. Filtro por vendedor. Stats del día.

- ■ PENDIENTE: confirmación antes de evaluar (evitar errores de click).
- ■ PENDIENTE: filtros adicionales por fecha y tipo PDV en el visor.

Fase 4 — Streamlit: Dashboard, Admin, Reportes y Login unificado

COMPLETADO — 100%

Login unificado con menú de servicios. Dashboard TV. Panel admin CRUD. Reportes con filtros, gráficos y exportación Excel.

- ✓ app.py — login + menú con cards clicables según rol (4 cards para superadmin).
- ✓ 2_Dashboard.py — ranking, KPIs, carrusel, auto-refresh 60s.
- ✓ 3_Admin.py — CRUD usuarios portal + roles Telegram.
- ✓ 4_Reportes.py — filtros cruzados, tabla, gráficos (torta + barras + timeline), exportar Excel.

Fase 5 — Portal Exhibiciones (.exe cliente)

REEMPLAZADA — App Streamlit en servidor

Decisión: todo en Streamlit en el servidor. Los clientes acceden desde el navegador. Sin instalación, actualizaciones instantáneas, multiplataforma.

Fase 6 — Hardening y producción

EN PROGRESO — 60%

Módulos de producción construidos e integrados en centermind_core.py.

- ✓ hardening/logger.py — logs diarios con rotación (30 días).
- ✓ hardening/backup_manager.py — backup automático medianoche + empaquetado a Drive.
- ✓ hardening/monitor.py — detector de caídas con alertas por Telegram.
- ✓ bot_worker.py — integrado con monitor (heartbeat cada 30s).
- ■ PENDIENTE: Panel Maestro (.exe tkinter) — consola de bots, logs, backups, distribuidoras.
- ■ PENDIENTE: Script de onboarding — agregar nueva distribuidora desde consola.

- PENDIENTE: Hasheo de contraseñas en usuarios_portal.

Fase 7 — Rebrand y estética: ShelfMind

PENDIENTE — próxima fase

Rediseño visual completo de toda la app Streamlit bajo la identidad ShelfMind.

- Nuevo nombre en todos los archivos, títulos y mensajes (CenterMind → ShelfMind).
- Identidad visual: Shelf Mindset — mentalidad orientada a ganar la góndola.
- Sistema de diseño: paleta, tipografía, iconografía e ilustraciones de anaquel/retail.
- Aplicar a: app.py, Visor, Dashboard, Admin, Reportes, bot_worker.py (mensajes Telegram).

8. CÓMO CORRER EL SISTEMA

Comando	Para qué
`pip install python-telegram-bot google-auth google-auth-oauthlib google-api-python-client pillow certifi streamlit pandas plotly openpyxl`	Instalar todas las 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). Sin hardening completo.
`python centermind_core.py`	Levantar TODOS los bots (producción). Incluye logs, backup automático y monitor de uptime.
`streamlit run CenterMind\StreamLitApp\app.py`	Levantar la app web. Ejecutar desde el directorio BOT-SQL\CenterMind\.

9. 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 (id_distribuidor AS id).
no such column: rol	Columna real es rol_telegram.	Aliases SQL (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 carpeta equivocada. Calcular .parent desde __file__.	app.py: BASE_DIR.parent.parent. pages/*.py: BASE_DIR.parent.parent.parent
StreamlitAPIException: Could not find page	app.py estaba dentro de pages/.	app.py debe vivir en StreamLitApp/, NO en pages/.
JobQueue.run_daily() unexpected keyword 'timezone'	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.	ALTER TABLE sin IF NOT EXISTS, ignorar 'duplicate column'.
Error 400: redirect_uri_mismatch	Credencial tipo 'Aplicación web'.	Crear credencial de tipo 'Aplicación de escritorio' (Desktop app).
Error 403: access_denied	App en 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.	Fix con certifi en setup_drive_oauth.py.
datatype mismatch (INSERT exhibiciones)	Insertaba UUID (texto) en id_exhibicion INTEGER.	Omitir id_exhibicion del INSERT, usar lastrowid.
FOREIGN KEY constraint failed	Se pasaba telegram_user_id en lugar del PK de integrantes_grupo.	Resolver PK via SELECT id_integrante FROM integrantes_grupo WHERE telegram_user_id=?
Ranking muestra HTML crudo	F-strings con HTML y emojis anidados se escapan.	Renderizar cada fila con st.markdown() individual. Usar entidades HTML.
Archivos en hardening/ con prefijo extra	El sistema de outputs los nombró hardening_logger.py etc.	Renombrar manualmente: quitar el prefijo 'hardening_' de cada archivo.

10. ARCHIVOS CRÍTICOS

Archivo	Criticidad	Qué hacer si se pierde
`base_datos/centermind.db`	CRÍTICO	Se pierden TODOS los datos. Backup diario automático via hardening/backup_manager.py.
`credencial_oauth.json`	CRÍTICO	Sin esto no se suben fotos. Regenerar en Google Cloud Console → APIs → Credenciales.
`token_drive.json`	ALTO	Correr setup_drive_oauth.py para regenerarlo. Auto-refresh si el token solo expira.

`token_bot en distribuidores`	**ALTO**	Crear bots nuevos en @BotFather y actualizar token_bot en la tabla distribuidores.
`hardening/*.py`	MEDIO	En el repositorio git. Recuperar con git pull.
`bot_worker.py / centermind_core.py`	MEDIO	En el repositorio git.
`StreamLitApp/*.py`	MEDIO	En el repositorio git.

11. GLOSARIO

Término	Significado
ShelfMind	Nombre del sistema. Evolución de CenterMind. Inspirado en Shelf Mindset: la mentalidad orientada a ganar la góndola en retail.
Shelf Mindset	Filosofía de trabajo que prioriza la ejecución en punto de venta: posicionamiento, visibilidad y espacio en el estante.
Distribuidora	Cliente del sistema. Tiene su propio bot de Telegram, carpeta en Drive y usuarios del portal.
PDV	Punto de Venta (kiosco, supermercado, almacén). 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	Exhibiciones consecutivas aprobadas/destacadas. Se muestra en el bot si es >= 2.
Puntos	Aprobadas + Destacadas×2. Métrica de rendimiento mensual del vendedor.
synced_telegram	Flag en exhibiciones. 0 = bot debe notificar. 1 = ya notificó.
Guard de sesión	Bloque 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 para lectura/escritura concurrente. Activo en todo el sistema.
Heartbeat	Señal de vida que bot_worker.py envía al monitor cada 30s via sync_evaluaciones_job.
VACUUM INTO	Método SQLite para crear una copia limpia y compacta de la DB. Usado por backup_manager.py.