Proyecto: Panel Enriquecedor para Superadmin

Objetivo: agregar al panel ADMIN (solo superadmin) un monitor en vivo con mapa por Provincia/Ciudad, lista de usuarios online y métricas de uso (sesiones, páginas, acciones y alertas). El diseño evita ubicación precisa (GPS) y se apoya en datos de distribuidora/localidad y, opcionalmente, geolocalización aproximada por IP.

Alcance pedido: Ciudad/Provincia alcanza. No se captura GPS.

1) Qué incluye el panel

- Mapa vivo con puntitos verdes (usuarios con sesión activa). Cada punto representa una Ciudad/Provincia.
- Tabla “Online ahora”: usuario, rol, distribuidora, ciudad, último movimiento (last_seen) y duración de sesión.
- Métricas del día: logins totales, usuarios únicos, tiempo medio de uso, pantallas más visitadas, exportaciones.
- Alertas: sesiones “idle” largas, exceso de errores, múltiples logins en poco tiempo, uso fuera de horario.
- Actividad Telegram (opcional): contadores por vendedor/grupo (sin ubicación) para correlacionar con uso del portal.

2) Diseño de datos (mínimo necesario)

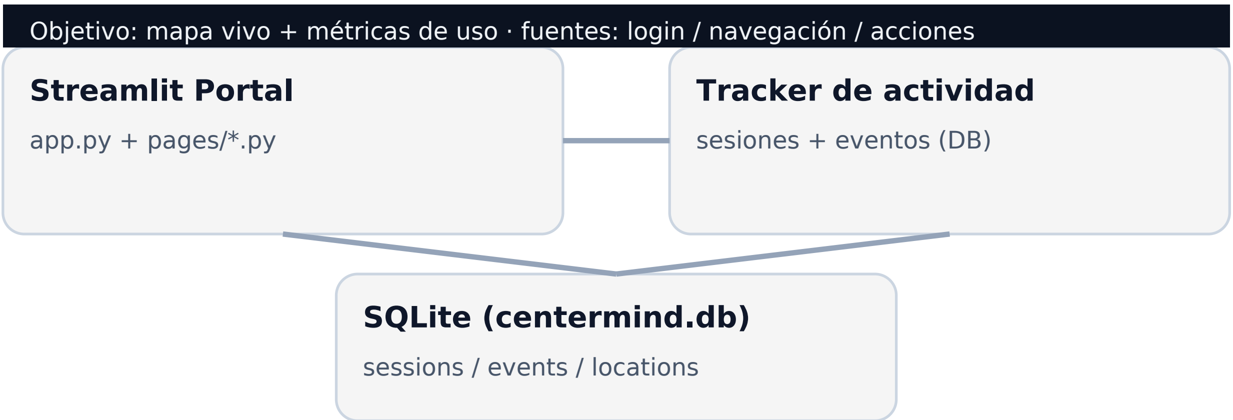
Para que el panel sea confiable y “a prueba de bypass” (no solo ocultar UI), la fuente de verdad es la base SQLite. Se agregan tablas para sesiones/eventos y un catálogo de ubicaciones por distribuidora (Ciudad/Provincia + coordenadas de referencia).

Tabla	Para qué sirve	Campos clave (resumen)
sessions	Quién está online y desde cuándo	session_id, user_id, role, distribuidora_id, login_at, last_seen_at
events	Qué hace cada usuario (páginas, exportes)	event_id, session_id, user_id, ts, event_type, metadata_json
locations	Mapa por Ciudad/Provincia (sin GPS)	location_id, distribuidora_id, city, province, lat, lon, label
telegram_activity (opcional)	Resumen de actividad del bot	ts, distribuidora_id, telegram_user_id, event_type, count

Nota de privacidad: “city/province” se obtiene por (a) catálogo por distribuidora (recomendado) o (b) geolocalización aproximada por IP. No se guarda lat/lon exacto del usuario.

3) Arquitectura y flujo de información

El flujo se basa en instrumentación liviana: cuando un usuario inicia sesión o navega, se registran eventos y se actualiza su last_seen. El panel consulta sesiones activas cada N segundos.



4) Qué archivos tocar y qué poner en cada uno

Tu repo ya tiene Streamlit con login en CenterMind/StreamLitApp/app.py y el panel en CenterMind/StreamLitApp/pages/3_Admin.py. La base está en CenterMind/base_datos/centermind.db y su inicialización en CenterMind/base_datos/setup_db.py.

Archivo / Carpeta	Cambios a realizar	Resultado esperado
CenterMind/StreamLitApp/app.py	Agregar tracking al login: crear session_id, registrar login_at y guardar metadatos b…	Cada login queda registrado. El panel p…
CenterMind/StreamLitApp/pages/1_Visor.py	Registrar eventos de uso relevantes (ej: abrir visor, aprobar/rechazar)…	Métricas por módulo y sesiones…
CenterMind/StreamLitApp/pages/2_Dashboard.py	Registrar dashboard_view y refresh. Actualizar last_seen.	Uso de modo TV contabilizado y visible en …
CenterMind/StreamLitApp/pages/4_Reportes.py	Registrar report_view, filtros aplicados (solo metadata), export_excel/pdf…	…
CenterMind/StreamLitApp/pages/3_Admin.py	Agregar nueva pestaña o sección (solo superadmin): Monitor en vivo (mapa + tabla…	Superadmin ve el mapa y el panel…
CenterMind/base_datos/setup_db.py	Extender el esquema: crear tablas sessions/events/locations (+ índices)…	DB preparada…
CenterMind/centermind_core.py (o nuevo módulo tracking.py)	Centralizar helpers: create_session, touch_session, log_event, resolve_city_province…	Evitar duplicación y asegurar consistencia…
CenterMind/bot_worker.py (opcional)	Registrar actividad Telegram (subida de foto, comandos)…	Panel puede cruzar uso portal vs actividad…

5) Hoja de ruta (pasos ordenados)

1. Paso 0 - Definir “online”: ventana de actividad (ej: online si last_seen < 3 min; idle si 3-10 min; offline si > 10 min).
2. Paso 1 - Migración de DB: agregar tablas sessions/events/locations e índices. Incluir migración para bases ya existentes.
3. Paso 2 - Helper único de tracking: funciones centrales (crear sesión, actualizar last_seen, registrar evento, resolver ciudad/provincia).
4. Paso 3 - Instrumentar login: al login correcto se crea sesión y se guarda en st.session_state (session_id).
5. Paso 4 - Instrumentar navegación: en cada page se ejecuta touch_session + log_event(page_view).
6. Paso 5 - Instrumentar acciones clave: visor (aprobar/rechazar), reportes (exportar), admin (cambios).
7. Paso 6 - Construir panel Superadmin: mapa + tabla + KPIs del día + alertas. Refresco cada 5-15 segundos.
8. Paso 7 - Validación y hardening: limpieza de sesiones zombie, y límites de crecimiento del log (retención).
9. Paso 8 (opcional) - Telegram activity: registrar eventos del bot para ver actividad paralela.

6) Detalles clave (decisiones recomendadas)

- Fuente de Ciudad/Provincia: preferir catálogo por distribuidora (tabla locations). Si falta, fallback por IP (coarse).
- Retención de eventos: guardar detalle 30-90 días y luego compactar a agregados diarios.
- Eventos mínimos: login, page_view, export, approve/reject, admin_change.
- Auditoría: para acciones sensibles, guardar metadata (qué se cambió) sin exponer datos personales en UI.
- Performance: índices por last_seen_at, ts y distribuidora_id; consultas agregadas por día.

7) Definición de terminado

- Superadmin ve una pestaña “Monitor” en ADMIN con mapa y lista de usuarios online.
- Los puntos se agrupan por Ciudad/Provincia y muestran conteos por distribuidora/rol.
- Las métricas del día (logins, usuarios únicos, tiempo medio, top acciones) se actualizan.
- Admins/evaluadores no pueden acceder a esta vista (guard).
- El sistema mantiene performance (consultas rápidas) y no crece sin límite (retención/compactación).

Apéndice: widgets sugeridos

- Mapa: círculos por ciudad (tamaño = usuarios online). Verde = online, ámbar = idle.
- KPIs: logins hoy, usuarios únicos, tiempo total, exportaciones.
- Top listas: usuarios por actividad, pantallas más vistas, exportadores.
- Alertas: idle, errores, múltiples logins, uso fuera de horario.