1. Raíz del Proyecto y Configuración

- **`package.json`** Gestión de dependencias con *version pinning*. Incluye `next-auth@4` (Estabilidad RF-8.1), `firebase-admin`, `@google-cloud/tasks` (Colas RF-5.1), `@turf/turf` (Geo RF-6.6), `node-qrcode` (RF-2.1), `recharts` (RNF-1), `papaparse` (Importación RF-1.11) y `sharp` (Optimización RNF-5). Scripts de build, test y linting.
- **`tsconfig.json`** Configuración TypeScript estricta. Define alias `@/*` para imports limpios. Esencial para tipado de esquemas complejos (`GeoJSON`, `WalletTransaction`) y estandarización de fechas UTC (RF-4.2).
- **`.env.example`** Plantilla de variables de entorno. Documenta secretos (`TWILIO_AUTH_TOKEN`, `MAPS_SERVER_KEY`), flags (`NEXT_PUBLIC_ENABLE_DAY_D` para Scope Freeze RF-1.4) y modo Sandbox Financiero (`NEXT_PUBLIC_ENV` RNF-8).
- **`.nvmrc`** Fija Node.js v20 para garantizar paridad estricta entre el entorno local de desarrollo, los pipelines de CI/CD y el runtime de Cloud Run (RNF-2).
- **`.eslintrc.json`** Reglas de linting y accesibilidad (WCAG AA para RNF-4). Previene uso de `console.log` en producción y asegura dependencias correctas en `useEffect` para evitar memory leaks.
- **`.prettierrc`** Reglas de formato de código para mantener consistencia en un equipo distribuido.
- **`next.config.mjs`** Configuración del compilador Next.js. Headers de seguridad (CSP), dominios de imágenes permitidos (Google Storage, Twilio MMS) y configuración de PWA (`next-pwa` RF-3.1).
- **`tailwind.config.ts`** Sistema de diseño visual. Define la paleta semántica (`var(--primary-color)`) para *Dynamic Theming* (RF-3.7) y clases "Eco-Ink" para impresión segura (RF-7.8).
- **`firebase.json`** Configuración de emuladores locales (Firestore, Auth, Functions) para desarrollo Offline (RF-3.5) y reglas de hosting/deploy.
- **`firestore.rules`** **Crítico (Seguridad)**. Reglas de validación de esquema y RBAC. Bloquea escritura en campos reservados (`dayD_config`), valida inmutabilidad de logs y segrega datos entre campañas (RF-4.6).
- **`firestore.indexes.json`** Índices compuestos obligatorios (RF-4.4). Optimiza consultas complejas (ej: `location.geoL2` + `recruitedCount` para Leaderboards) y excluye campos costosos (`audit_logs.details`).
- **`storage.rules`** Reglas de seguridad GCS. Valida MIME types, tamaño (<3MB) y propiedad de archivos para Avatares, Logos de Campaña y Evidencia E-14 (RF-3.9).
- **`.firebaserc`** Alias de proyectos para segregación estricta de entornos `staging` y `production` (RNF-8).
- **`middleware.ts`** Edge Middleware. Intercepta rutas para validación de sesión (RF-8.9), redirección de Smart QRs (RF-2.2), bloqueo por "Legal Wall" (RF-3.8) y enrutamiento contextual.
- **`Dockerfile`** Construcción de imagen para Cloud Run. Optimiza "Cold Start" e inyección de secretos en runtime (RF-8.10) usando volúmenes montados en lugar de ENV vars.
- **`.gcloudignore`** Optimización de despliegue. Evita subir archivos innecesarios (tests, docs, node_modules local) al contexto de construcción de Google Cloud Build.
- **`components.json`** Configuración de `shadcn/ui`. Define la ubicación de los componentes base, estilos globales y variables CSS para garantizar la consistencia visual y el cumplimiento de contraste WCAG AA (RNF-4) en toda la PWA.
- **`playwright.config.ts`** Configuración para pruebas E2E (RF-8.12 / RNF-9). Define los perfiles de dispositivos móviles (Pixel 5, iPhone 12) requeridos para validar la "Experiencia Móvil" (RF-7.5) y configura el manejo de *fixtures* para interceptar red y simular el modo Offline (RF-3.5) durante los tests.
- **`jest.config.ts`** Configuración de Jest (Pág. 145). Necesaria para ejecutar las pruebas unitarias de la "Lógica de Negocio" crítica, específicamente la matriz de fusión de datos offline y la prevención de ciclos en la jerarquía (RF-1.9).
- **`jest.setup.ts`** Inicializador de entorno de pruebas. Se encarga de simular (mock) APIs del navegador que no existen en Node.js, como `IntersectionObserver` (para el RF-7.7 Virtual Scroll) y `window.matchMedia`.
- **`jest.polyfills.js`** Necesario para el entorno de pruebas (`/tests`). Polyfills críticos para `TextEncoder`, `TextDecoder` y `ReadableStream` que no existen en el entorno JSDOM de Jest pero son utilizados por `Auth.js v5` y las librerías de `Firebase`. Garantiza que los tests unitarios de lógica de negocio (RF-3.5) corran sin errores de entorno.
- **`postcss.config.mjs`** Configuración obligatoria para el procesador de CSS. Necesario para que `tailwind.config.ts` y las directivas de *Dynamic Theming* (RF-3.7) funcionen correctamente en el pipeline de construcción de Next.js.
- **`instrumentation.ts`** Hook de ciclo de vida de Next.js. Inicializa singletons críticos (como `Firebase Admin` y `RateLimitService`) al arrancar el servidor, evitando la latencia de "Cold Start" en conexiones a base de datos (RNF-1) y asegurando que los monitores de seguridad (RF-8.8) estén activos antes de procesar la primera solicitud.
- **`.env.local`** (Gitignored). Archivo real para desarrollo local. Debe contener las llaves de los emuladores de Firebase y los "Magic Numbers" de Twilio para simulación (RF-8 Testing).
- **`.dockerignore`** Exclusión estricta de archivos locales (`.env.local`, `node_modules`, `tests`, `docs`) del contexto de construcción de Docker. Reduce drásticamente el tamaño de la imagen y el tiempo de despliegue ("Cold Start").
- **`.env.test`** Archivo de configuración crítico para la ejecución de pruebas automatizadas (CI/CD). Define variables de entorno estáticas para los emuladores de Firebase (`FIRESTORE_EMULATOR_HOST`) y credenciales *dummy* para Twilio/Google Maps, asegurando que los tests unitarios y E2E (Playwright) nunca interactúen con la infraestructura de Producción ni Staging, protegiendo el presupuesto financiero.

1.1 Pipeline CI/CD (`/.github`)

- **`.github/workflows/deploy.yml`** Despliegue seguro (RF-8.12). Usa *Workload Identity Federation* (WIF) para autenticación keyless. Gestiona *Traffic Splitting* en producción y despliegue a staging.
- **`.github/workflows/quality-check.yml`** Barrera de calidad (RNF-9). Ejecuta linter, chequeo de tipos y pruebas (Unitarias/E2E) obligatorias antes de cualquier merge a rama principal.
- **`.github/CODEOWNERS`** Asigna responsables de módulos críticos (Financiero, Seguridad) para revisión obligatoria de Pull Requests.

---

2. Estructura de Aplicación (`/app`)

2.1 Configuración Global

- **`layout.tsx`** Layout raíz. Proveedores globales (`Session`, `Campaign`, `Offline`, `Theme`) y componentes UI críticos (`GlobalSnackBar`, `InstallPrompt` RF-3.1).
- **`not-found.tsx`** Página 404 personalizada para retener al usuario dentro de la navegación de la PWA, evitando fugas de contexto.
- **`global-error.tsx`** Error Boundary global. Captura "Pantalla Blanca" y permite reinicio suave de sesión (RNF-1) sin perder datos locales.
- **`loading.tsx`** Skeleton UI global para feedback inmediato en transiciones de ruta (RNF-1).
- **`manifest.ts`** Permite inyectar las variables de color (`primaryColor`) y logos definidos en `political_campaigns.branding` (RF-3.7) directamente en el manifiesto de instalación, asegurando que el icono y nombre de la PWA coincidan con la campaña activa del usuario y no con la marca genérica del software.
- **`robots.ts`** Control de indexación. Bloquea la indexación de `/dashboard` y permite `/join`. Crítico para evitar que motores de búsqueda expongan datos sensibles o URLs de administración, cumpliendo indirectamente con el principio de privacidad del RNF-7.

- **`globals.css`** Archivo de estilos globales imprescindible para **RF-3.7 (Adaptabilidad de Marca)**. Aquí se definen las variables CSS CSS (`:root { --primary-color: ... }`) que el `ThemeContext` manipulará dinámicamente según la configuración de la campaña (`political_campaigns.branding`). También contiene las directivas de Tailwind y los estilos base para el modo "Eco-Ink" (`@media print`).
- **`sitemap.ts`** Generador dinámico de mapa del sitio. Aunque el Dashboard es privado (RF-7.6), el catálogo público de campañas (`/join`) y las páginas legales deben ser indexables para maximizar la "Vinculación Orgánica" (Flujo 1B) y la transparencia (RF-3.2).
- **`apple-icon.tsx`** **RF-3.7:** Generación dinámica del icono de inicio para dispositivos iOS (Apple Touch Icon). Utiliza la librería `ImageResponse` de Next.js para renderizar el logo y color primario de la campaña activa (`political_campaigns.branding`) en tiempo de solicitud, asegurando que la PWA instalada refleje la identidad del candidato y no del software genérico.
- **`opengraph-image.tsx`** **RF-5.2:** Generación dinámica de imágenes de vista previa para redes sociales (WhatsApp/Facebook). Cuando se comparte el "Enlace de Invitación Único", esta ruta genera una imagen personalizada con la foto del candidato y el nombre del reclutador, aumentando la tasa de conversión de la vinculación orgánica.
- **`page.tsx`** Punto de entrada raíz de la aplicación. Implementa la lógica de redirección inteligente: si existe una sesión válida (JWT), envía al usuario a `/dashboard` ; si no, muestra la *Landing Page* pública o redirige a `/login` según la configuración de la campaña. Esencial para manejar el flujo de "Vinculación Orgánica" (Flujo 1B) y evitar pantallas blancas en la raíz.
- **`forbidden.tsx`** Pantalla de error 403 personalizada. Requerida para manejar visualmente los bloqueos de seguridad estrictos definidos en **RF-1.10 (Lista Negra / Fraude)** y **RF-8.7 (Bloqueo por IP)**. Informa al usuario la razón del bloqueo (ej: "Cuenta suspendida por actividad sospechosa") sin exponer detalles técnicos, cumpliendo con la política de seguridad ofensiva.
- **`offline/page.tsx`** Pantalla de fallback estática del Service Worker cuando no hay red ni caché disponible para la ruta solicitada (RF-3.5).

2.2 Autenticación y Legal (`/(auth)`)

- **`/(auth)/layout.tsx`** Layout limpio sin navegación principal ni sidebars para enfocar al usuario en el flujo de ingreso/registro.
- **`/(auth)/login/page.tsx`** Login Omnicanal. Detecta dispositivo móvil para forzar flujo OTP y evitar pérdida de contexto ("Code-Over-Link" RF-8.1).
- **`/(auth)/login/layout.tsx`** Layout específico para la pantalla de Login. Su función es eliminar cualquier elemento de navegación (Sidebar, Footer) y centrar el contenido, garantizando que el usuario se enfoque únicamente en el flujo de ingreso u OTP ("Distraction-Free").
- **`/(auth)/verify-otp/page.tsx`** Ingreso de OTP con *Rate Limiting* visual, cuenta regresiva y manejo de reintentos (RF-8.7).
- **`/(auth)/recover/page.tsx`** Recuperación de cuenta (RF-1.7). Flujo autoservicio vía Email o instrucciones para validación presencial con Coordinador (Identity Anchor).
- **`/(auth)/register/admin-request/page.tsx`** Formulario público para solicitar rol de Administrador de Campaña. Permite carga de hoja de vida y aval para revisión del Super Admin (RF-1.17).
- **`/(auth)/legal/update-required/page.tsx`** "Legal Wall" (RF-3.8). Bloqueo modal inescapable que obliga a aceptar nuevos términos si la versión legal cambia en la BD.

2.3 Acceso Público (`/(public)`)

- **`/(public)/go/[code]/page.tsx`** Middleware visual para Smart QRs (RF-2.2). Registra analítica y redirige (WhatsApp, Registro, Evento) según configuración dinámica.
- **`/(public)/legal/terms/page.tsx`** Vista estática pública de Términos y Política de Habeas Data para cumplimiento legal y referencia.
- **`/(public)/events/[id]/checkin/page.tsx`** Asistencia pública a eventos (RF-1.13). Valida geolocalización (Geo-Fence) vs ubicación del usuario antes de registrar el ingreso.

2.4 Dashboard (`/dashboard`)

- **`/dashboard/layout.tsx`** Layout autenticado. `Sidebar` progresiva según rol, `CampaignSwitcher` (RF-7.1) e indicador de estado de sincronización (RF-3.5).
- **`/dashboard/page.tsx`** Dashboard polimórfico. Enrutador interno que carga vistas dinámicas (`LinkView` , `CoordinatorView` , `AdminView`) según rol para optimizar carga (RF-7.1).
- **`/dashboard/error.tsx`** Manejador de errores granular para el Dashboard. Necesario para aislar fallos en widgets específicos (ej: fallo en carga de métricas de Twilio RF-5.6) sin colapsar toda la interfaz de la aplicación (RNF-3 Resiliencia).
- **`/dashboard/mobile-nav.tsx`** Implementación del **Bottom Bar** requerido por **RF-7.5** para dispositivos móviles. Reemplaza la `Sidebar` en viewports pequeños (<768px), ofreciendo acceso táctil rápido (48x48px) a las vistas principales (Inicio, Mapa, QR, Perfil) sin obstruir la visualización de datos.
- **`/dashboard/day-d/page.tsx`** **RF-3.9 / RF-1.4 (Scope Freeze):** Placeholder arquitectónico para la Fase 3. Debe renderizar un componente vacío o redirigir al Dashboard principal en la v1.0, pero su existencia valida que el enrutador soporta la carga futura de módulos logísticos sin refactorizar la estructura base.
- **`/dashboard/day-d/layout.tsx`** Layout aislado para el módulo de Día D. Implementa la lógica de **"Visual Sterilization"** (RF-1.4), asegurando que, incluso si se accede a la ruta, los menús y widgets de campaña estándar no se mezclen con la operación logística futura.

  Zona Neutra (Usuario sin Campaña):

  - **`/dashboard/join/page.tsx`** Catálogo público de campañas ("Zona Neutra"). Permite a usuarios buscar campañas activas por ubicación para adhesión voluntaria (Flujo 1B).
  - **`/dashboard/join/[id]/page.tsx`** Vista previa de campaña. Muestra perfil del candidato y botón de acción para ejecutar la auto-asignación territorial (Geo-Routing RF-1.3).

  Gestión de Plataforma (SUPER_ADMIN):

  - **`/dashboard/platform/requests/page.tsx`** Bandeja de entrada Super Admin. Revisión y aprobación/rechazo de solicitudes de nuevos administradores de campaña (RF-1.17).
  - **`/dashboard/platform/tenants/page.tsx`** Gestión global de campañas (Tenants). Pausado de instancias, auditoría técnica cross-tenant.
  - **`/dashboard/platform/divipol/page.tsx`** Gestión maestra de Divipol (RF-1.14). Carga masiva y geocodificación de puestos de votación base.
  - **`/dashboard/platform/audit/page.tsx`** Visor de Logs globales del sistema. Herramienta forense para soporte técnico.
  - **`/dashboard/platform/global-settings/page.tsx`** Configuración técnica (`system_config`). Precios base Twilio, IPs permitidas (RF-8.7) y Feature Flags.
  - **`/dashboard/platform/divipol/[departmentId]/page.tsx`** Vista detallada para el SUPER_ADMIN. Permite navegar la jerarquía oficial (Departamento -> Municipio -> Puesto -> Mesa) para auditar la cobertura geográfica y corregir geocodificaciones maestras antes de que las campañas las importen.

Configuración Inicial (ADMIN - Day Zero):

- **`/dashboard/setup/politician/page.tsx`** Paso 1 del Wizard. Creación del `politicalProfile` (Candidato, Bio, Partido) antes de definir la campaña (RF-1.17).
- **`/dashboard/setup/campaign/page.tsx`** Paso 2 del Wizard. Configuración de `political_campaign`, definición de Scope (Geo), Colores y Presupuesto inicial (RF-1.17).
- **`/dashboard/setup/layout.tsx`** Layout de aislamiento para el Wizard de configuración inicial. Debe impedir la renderización de la `Sidebar` estándar y bloquear la navegación hacia otras rutas hasta que se complete la configuración de la campaña ("Day Zero").

Gestión Operativa (Usuarios - Todos los Roles):

- **`/dashboard/users/page.tsx`** Lista de usuarios con *Virtual Scroll* (RF-7.7). Buscador servidor (RF-7.6) y filtros de alcance (RF-7.2).
- **`/dashboard/users/create/page.tsx`** Registro manual (RF-1.1). Validaciones edad, unicidad y manejo de "Puesto No Listado" y cola Offline.
- **`/dashboard/users/[id]/page.tsx`** Perfil 360. Auditoría de lectura (RF-8.8), botón "Prueba de Vida" (RF-3.11) y asignación de puesto (Time-Lock RF-1.1).
- **`/dashboard/users/disputes/page.tsx`** Bandeja de gestión de solicitudes de traslado y conflictos ("Divorcios") (RF-1.12).
- **`/dashboard/users/verification-queue/page.tsx`** Vista dedicada para el Rol Coordinador. Lista exclusivamente los usuarios con `fraudStatus: 'SUSPICIOUS_VELOCITY'` o `'NEEDS_VERIFICATION'`. Permite ejecución masiva de auditoría telefónica y desbloqueo, separando esta gestión crítica del listado general de usuarios para optimizar el flujo de trabajo.

Gestión Territorial (Coord/Admin):

- **`/dashboard/territory/map/page.tsx`** Mapa interactivo principal. Dibujo de polígonos (RF-6.6), visualización de conflictos, zonas vacantes y "Votos Foráneos".
- **`/dashboard/territory/fixer/page.tsx`** Herramienta "The Fixer" (RF-6.7). Cola de corrección manual para fallos de geocodificación automática.
- **`/dashboard/territory/zones/page.tsx`** Gestión tabular de zonas territoriales, métricas de saturación, asignación de responsables y estado de vacancia.
- **`/dashboard/territory/zones/conflicts/page.tsx`** Vista dedicada para el listado y resolución de zonas con estado `DISPUTED`. Aunque existe el modal, una vista de lista es necesaria para que el Coordinador gestione múltiples conflictos territoriales simultáneos.
- **`/dashboard/territory/map/loading.tsx`** Estado de carga específico para el módulo de mapas. Renderiza el `MapSkeleton` (Malla de carga) definido en el diseño de pantallas, evitando el *Layout Shift* acumulativo mientras se cargan las librerías de Google Maps y los polígonos GeoJSON (RNF-1).

Gestión de Eventos (RF-1.13):

- **`/dashboard/events/page.tsx`** Listado de eventos programados, activos e históricos con métricas de asistencia.
- **`/dashboard/events/create/page.tsx`** Formulario de creación. Define Geo-fence (Ubicación + Radio), fecha y asignación de Smart QR.
- **`/dashboard/events/[id]/page.tsx`** Detalle de evento. Lista de asistentes validada en tiempo real y descarga de reportes.

Comunicaciones (Admin/Coord):

- **`/dashboard/communications/campaigns/page.tsx`** Listado de envíos. Estados de envío, costos auditados y métricas de conversión (RF-7.10).
- **`/dashboard/communications/create/page.tsx`** Asistente de envío masivo (RF-5.1). Segmentación de audiencia, selección de plantilla y chequeo presupuestal "Pre-Flight".
- **`/dashboard/communications/templates/page.tsx`** Constructor visual de plantillas HSM (RF-5.3). Vista previa, validación de estado en Meta y monitor de costo en tiempo real.
- **`/dashboard/communications/audience/page.tsx`** Vista para la gestión de segmentos de audiencia y validación de "Health Score" (usuarios importados vs. orgánicos) antes de crear una campaña.
- **`/dashboard/communications/audience/preview/page.tsx`** Vista intermedia antes de aprobar una campaña. Renderiza una tabla con la muestra exacta de usuarios que recibirán el mensaje tras aplicar los filtros de "Health Score" y "Consentimiento", permitiendo al Admin verificar la segmentación antes de comprometer el presupuesto.
- **`/dashboard/communications/assets/page.tsx`** Gestor de archivos multimedia específicos para plantillas de WhatsApp. Permite subir, visualizar y validar imágenes/videos (tamaño, formato) antes de vincularlos a una plantilla HSM, desacoplando la gestión de *assets* de la edición de texto.

Financiero (Admin/Coord):

- **`/dashboard/wallet/page.tsx`** Billetera virtual (RF-5.6). Historial de transacciones (Ledger), recargas manuales (Admin), estado del "Circuit Breaker".

Reportes e Impresión:

- **`/dashboard/reports/print-list/page.tsx`** Generador de "Listados de Punteo" (RF-7.8). Vista con CSS `@media print` optimizada ("Eco-Ink") para impresión física segura y trazable.

Administración de Campaña (ADMIN - Nivel Tenant):

- **`/dashboard/admin/staffing/page.tsx`** Nombramientos jerárquicos (RF-1.15). Ascenso manual de Coordinadores/Enlaces dentro de la campaña.
- **`/dashboard/admin/bulk-move/page.tsx`** Herramienta de reasignación masiva (RF-1.8). Interfaz Drag & Drop para mover ramas completas de estructura.
- **`/dashboard/admin/export/page.tsx`** Módulo de exportación segura (RF-4.3). Modos "Master" y "Operativo" con enmascaramiento DLP y notificación.
- **`/dashboard/admin/import/page.tsx`** Interfaz de importación masiva CSV (RF-1.11). Validación de esquema, límite de 5k filas y declaración juramentada.
- **`/dashboard/admin/audit/page.tsx`** Logs de campaña. Auditoría de acciones dentro del alcance del tenant específico.
- **`/dashboard/admin/settings/page.tsx`** Configuración de campaña. Colores, Logo, Mensaje de Bienvenida y datos del Candidato.

Perfil:

- **`/dashboard/profile/page.tsx`** Autogestión de cuenta. Seguridad (Dispositivos activos RF-8.9) y datos personales.
- **`/dashboard/profile/my-campaigns/page.tsx`** Selector de contexto (Campaign Switcher Full Page) para usuarios que pertenecen a múltiples campañas.

2.5 Rutas de API (`/app/api`)

- **`/api/auth/[...nextauth]/route.ts`** Manejador de Auth.js. Configuración de providers, callbacks de sesión y lógica de rotación de Refresh Tokens (RF-

8.9).

- **`/api/auth/sessions/revoke/route.ts`** Endpoint de seguridad para la **Autogestión de Seguridad (RF-8.9.C)**. Permite al usuario revocar tokens de refresco de otros dispositivos activos desde la vista "Mi Perfil". Esencial para el botón "Cerrar sesión en otros dispositivos".
- **`/api/platform/approve-admin/route.ts`** Endpoint (Solo SUPER_ADMIN) para aprobar solicitud y elevar rol a `ADMIN`.
- **`/api/campaigns/create/route.ts`** Endpoint (Solo ADMIN) para registrar nueva campaña y vincularla al político.
- **`/api/campaigns/list/route.ts`** Endpoint público para el Catálogo de Campañas (Filtros geográficos).
- **`/api/users/join-campaign/route.ts`** Endpoint para adhesión orgánica. Lógica Geo-Routing (RF-1.3).
- **`/api/users/check-uniqueness/route.ts`** Validación ligera pre-envío (RF-1.1). Verifica duplicidad de Cédula/Teléfono devolviendo booleano.
- **`/api/users/[id]/geo-fix/route.ts`** Manejador específico para el **Protocolo de Resolución de Conflictos Geográficos (RF-6.7 / RF-97)**. Recibe la corrección manual del Coordinador ("Pin Manual"), actualiza coordenadas, cambia el estado a `COMPLETED` y dispara el recálculo de zona.
- **`/api/kyc/verify/route.ts`** Endpoint para iniciar validación de identidad asíncrona con proveedor externo (RF-1.16).
- **`/api/admin/bulk-move/route.ts`** Endpoint transaccional para ejecutar la **Herramienta de Reasignación Masiva (RF-1.8)**. Debe manejar la validación de ciclos (RF-1.9), procesar lotes de 500 escrituras (Batch Writes) y generar los logs de auditoría correspondientes. Conecta la UI de `bulk-move/page.tsx` con Firestore.
- **`/api/admin/import/route.ts`** Procesador de carga masiva (RF-1.11). Valida CSV línea por línea, sanitiza datos y encola tareas de geocodificación.
- **`/api/communications/templates/[id]/route.ts`** API REST para el **CRUD de Plantillas (RF-5.3)**. Permite guardar borradores, eliminar y recuperar el estado de aprobación de una plantilla específica. Complementa al `TemplateEditor.tsx`.
- **`/api/communications/templates/sync/route.ts`** Endpoint para forzar la sincronización de estado de plantillas contra la API de Twilio/Meta (RF-5.3.C) en caso de que los webhooks fallen o se requiera validación inmediata antes de un envío ("Pre-Flight Check").
- **`/api/territory/validate-polygon/route.ts`** Endpoint para validación topológica server-side (RF-6.6). Verifica "Donuts" y colisiones complejas antes de guardar.
- **`/api/webhooks/twilio/status/route.ts`** Webhook de entrega (RF-5.7). Procesa estados de mensajes y ejecuta lógica de reembolsos automáticos (Fair Billing).
- **`/api/webhooks/twilio/inbound/route.ts`** Webhook de entrada (RF-5.8). Clasifica tráfico, filtra respuestas interactivas, bloquea multimedia y dispara auto-respuestas.
- **`/api/cron/geocoding/route.ts`** Endpoint para Cloud Scheduler (RF-6.7). Dispara el procesamiento batch de la cola de geocodificación diferida (02:00 AM).
- **`/api/cron/finance-sweeper/route.ts`** Endpoint para Cloud Scheduler (RF-5.9). Ejecuta auditoría y reconciliación de saldos para corregir transacciones "zombies".
- **`/api/cron/data-retention/route.ts`** Endpoint para Cloud Scheduler (RF-8.11). Mueve logs antiguos a Cold Storage y ejecuta limpieza de datos anonimizados.
- **`/api/voting-places/search/route.ts`** Endpoint optimizado para la búsqueda de Puestos de Votación ("Local Searchable Select"). Filtra la colección `voting_places` por departamento/municipio y texto, devolviendo solo los campos necesarios para reducir latencia y consumo de lectura en Firestore.
- **`/api/users/disputes/create/route.ts`** Endpoint transaccional para crear una solicitud de traslado (`transfer_requests`). Valida que no exista una solicitud pendiente y notifica al Juez Jerárquico correspondiente (Coordinador o Admin) según la matriz de escalamiento.
- **`/api/users/disputes/resolve/route.ts`** Endpoint exclusivo para Coordinadores/Admin. Ejecuta la transacción atómica de cambio de líder (Aprobación) o el rechazo de la solicitud, actualizando historiales y métricas de ambos líderes (Viejo y Nuevo).
- **`/api/users/[id]/verify-life/route.ts`** Endpoint para la "Prueba de Vida". Recibe la geolocalización del dispositivo del Enlace, valida que esté dentro del polígono de la zona y marca al Multiplicador como `VERIFIED_IN_PERSON`.
- **`/api/user/push-subscription/route.ts`** Endpoint para guardar y actualizar las suscripciones de **Web Push** (RF-5.10). Asocia el `endpoint` y claves `p256dh` / `auth` del navegador con el UID del usuario para permitir el envío de notificaciones "Motivacionales" y alertas de seguridad.
- **`/api/integrations/meta/webhook/route.ts`** Webhook dedicado para recibir actualizaciones de estado de las Plantillas de WhatsApp (RF-5.3.C) directamente desde Meta, independiente del flujo de mensajes de Twilio. Actualiza el estado `APPROVED` / `REJECTED` en `message_templates`.
- **`/api/cron/security-monitor/route.ts`** Endpoint para Cloud Scheduler. Ejecuta la lógica de **Detección de Anomalías (RF-8.8.B)** y auditoría de **Rate Limiting (RF-8.7)** en segundo plano. Analiza patrones de logs para detectar scraping o fuerza bruta que no hayan sido bloqueados en tiempo real por el WAF, enviando alertas digestivas al Admin.
- **`/api/admin/export/stream/route.ts`** Complemento a `actions/audit-export.ts`. Las *Server Actions* tienen límites de tiempo de ejecución estrictos. Para la **Exportación Modo Master (RF-4.3)** de bases de datos grandes, se requiere un *Route Handler* que permita streaming de datos binarios (CSV/Excel) directamente desde Firestore/GCS al cliente, evitando timeouts en Cloud Run.
- **`/api/webhooks/twilio/template-status/route.ts`** **RF-5.3 / RF-5.7:** Webhook dedicado para actualizaciones de estado de plantillas HSM (Approved/Rejected/Paused) desde Meta/Twilio. A diferencia del estado de *mensajes*, este endpoint gestiona el ciclo de vida del contenido y bloquea campañas en borrador si la plantilla pierde aprobación.
- **`/api/admin/zone-override/route.ts`** Endpoint transaccional para ejecutar la **Asignación Manual (Parche)** descrita en **RF-93**. Permite al Coordinador forzar el `manualZoneId` de un usuario cuando la geocodificación automática falla o es un barrio nuevo, actualizando las métricas de zona instantáneamente.
- **`/api/webhooks/kyc/result/route.ts`** **RF-1.16:** Endpoint público (protegido por firma HMAC) para recibir los callbacks de proveedores de identidad (Truora/MetaMap). Actualiza asíncronamente el `identityValidationStatus` del usuario a `VERIFIED` o `FAILED` sin bloquear el hilo principal de registro.
- **`/api/admin/identity/swap-phone/route.ts`** **RF-1.7:** Endpoint administrativo crítico para ejecutar el "Protocolo de Recuperación de Identidad". Desvincula el UID del teléfono anterior y lo ancla al nuevo en Auth Provider y Firestore, registrando la evidencia de "Validación Documental" en `audit_logs`.
- **`/api/campaigns/[id]/branding/route.ts`** Endpoint público ligero. Permite obtener los colores y el logo de una campaña específica por su ID antes de que el usuario inicie sesión. Esencial para pintar la interfaz de Login con la identidad del candidato sin exponer datos sensibles.
- **`/api/assets/qr/[uid]/route.ts`** *Route Handler* para servir el código QR de un usuario como imagen dinámica (`image/png`). Necesario para incrustar el QR en las plantillas de correo transaccional (`InvitationEmail.tsx`) y en la vista de impresión (`EcoInkTable.tsx`) sin recurrir a URIs Base64 pesadas que bloquean clientes de correo o aumentan el tamaño del payload HTML. Consume `node-qrcode` y aplica caché HTTP.
- **`/api/health/route.ts`** Endpoint ligero para **Cloud Run Startup/Liveness Probes**. Verifica la conexión a Firestore y la inicialización de singletons. Esencial para que el balanceador de carga sepa cuándo la instancia "Always On" (RNF-1) está lista para recibir tráfico real tras un despliegue, evitando errores 500 en las primeras peticiones.
- **`/api/admin/users/[id]/restore/route.ts`** Endpoint administrativo para revertir manualmente una acción de "Archivado" o "Eliminación" (si no ha pasado el TTL de anonimización). Necesario para corregir errores operativos humanos (ej: borrar al líder equivocado).
- **`/api/reports/coverage/route.ts`** Endpoint optimizado para generar el reporte de "Penetración Electoral" (Mapa de Calor). Agrega datos geoespaciales en el servidor (usando consultas de límites geohash) para enviar al cliente solo los puntos de densidad, protegiendo la carga de datos masiva.
- **`/api/admin/system/flush-cache/route.ts`** Endpoint administrativo para invalidar forzosamente la caché de Redis/Memoria de las métricas del Dashboard. Necesario para situaciones de "Guerra Room" donde los Coordinadores requieren ver datos en tiempo real absoluto (Live Mode) ignorando el TTL de 5 minutos.
- **`/api/assets/avatar/[uid]/route.ts`** Route Handler para servir avatares de usuario optimizados y seguros. Valida que el usuario solicitante tenga permisos (visibilidad de rama) para ver la foto de otro usuario, evitando la exposición pública de imágenes de perfil en buckets abiertos.
- **`/api/webhooks/truora/status/route.ts`** Endpoint dedicado para recibir los callbacks asíncronos de la validación de identidad (Truora/MetaMap). Procesa el JSON de resultado, actualiza el `identityValidationStatus` del usuario y dispara notificaciones si la validación falla.
- **`/api/webhooks/resend/status/route.ts`** Endpoint para rastrear la entrega de correos transaccionales (Bounce/Delivery). Vital para monitorear la salud de las notificaciones críticas de seguridad (**RF-4.3 Security Broadcast**) y mantener limpia la lista de correos.
- **`/api/territory/zones/suggest-color/route.ts`** Endpoint ligero que devuelve un color hexadecimal no utilizado en la zona geográfica actual. Ayuda al

Coordinador a crear nuevas zonas (**RF-6.6**) sin generar colisiones visuales en el mapa.

2.6 Lógica de Servidor y Mutaciones (`/app/actions`)

- **`/actions/auth-actions.ts`** Maneja el inicio de sesión y la verificación de OTP server-side. Implementa la lógica de "Code-Over-Link" (RF-8.1) y la detección de User-Agent para alertas de dispositivos antiguos (RNF-6.C) antes de crear la sesión.
- **`/actions/campaign-setup.ts`** Gestiona las mutaciones del Wizard "Day Zero" (RF-1.17). Ejecuta transacciones atómicas para crear el perfil político, la campaña y el presupuesto inicial en un solo paso, garantizando integridad referencial.
- **`/actions/territory-mutations.ts`** Contiene la lógica de escritura para el dibujo de polígonos (RF-6.6). Implementa la validación de "No Donuts", el control de concurrencia optimista (Version Check) y la generación del snapshot para auditoría forense.
- **`/actions/financial-ops.ts`** Ejecuta la transacción crítica de **Recarga Manual de Saldo** (RF-5.6). Valida la unicidad del comprobante bancario (Idempotency Check) y verifica que el usuario esté en la `financialAdmins` allowlist antes de tocar la base de datos.
- **`/actions/audit-export.ts`** Dispara el proceso de Exportación Segura (RF-4.3). Registra el evento de seguridad, notifica a los stakeholders (Security Broadcast) y genera la URL firmada temporal para la descarga del archivo marcado digitalmente.
- **`/actions/identity-ops.ts`** Maneja las operaciones sensibles de identidad del **RF-1.7**. Contiene la función para el cambio administrativo de número telefónico (Swap) y la invalidación forzada de sesiones, separada de las operaciones financieras.
- **`/actions/user-promotion.ts`** Ejecuta el **Protocolo de Ascenso (RF-1.4)**. Valida que el usuario sea Seguidor, asigna el rol `MULTIPLIER`, genera su QR (M-XXX) y actualiza los permisos de sesión sin requerir un re-login completo.
- **`/actions/dispute-resolution.ts`** Centraliza la lógica para **RF-1.12 (Gestión de Disputas)** y **RF-6.6 (Conflictos Territoriales)**. Ejecuta las transacciones atómicas para mover usuarios entre líderes o redefinir límites de polígonos, asegurando integridad referencial.
- **`/actions/user-demotion.ts`** Maneja el **Protocolo de Democión (RF-1.6)**. Es la contraparte de `user-promotion.ts`. Ejecuta la lógica crítica de degradar un Coordinador a Seguidor, desencadenando la **Liberación de Responsabilidad Territorial** (marcar zonas como `VACANT`) y la transferencia de la red descendente al nodo padre.
- **`/actions/notifications-actions.ts`** Server Actions para marcar notificaciones como leídas o archivadas (RF-5.10). Permite interactuar con la colección `notifications` desde el componente cliente `NotificationCenter` sin exponer una API REST completa.
- **`/actions/generate-upload-signed-url.ts`** Genera URLs firmadas de corta duración (V4 Signing) para Google Cloud Storage. Permite a los clientes subir archivos (E-14, Avatares) directamente al bucket seguro sin pasar por el servidor de aplicaciones, cumpliendo la política **Keyless (RF-8.12)**.
- **`/actions/divipol-ops.ts`** Maneja la creación/edición unitaria de puestos de votación por parte del Super_Admin. Incluye la validación del **Mandato de Congelamiento (RF-4.6)** antes de escribir en la base de datos.
- **`/actions/user-archival.ts`** Maneja las transiciones de estado a `ARCHIVED` (por expatriación) o `ANONYMIZED` (por solicitud legal). Ejecuta la lógica de "Sobrescritura Destructiva" sobre los campos PII y gestiona el decremento de contadores en los líderes afectados dentro de una transacción.
- **`/actions/media-management.ts`** *Server Action* para gestionar el ciclo de vida de los medios de WhatsApp. Maneja la subida a GCS, la obtención de URLs públicas temporales (requeridas por la API de Meta) y la limpieza de archivos huérfanos rechazados por las políticas de contenido.
- **`/actions/survey-analysis.ts`** Acción para agregar resultados de encuestas bajo demanda. Calcula distribuciones estadísticas complejas (ej: correlación entre "Intención de Voto" y "Barrio") que son demasiado pesadas para calcularse en el cliente o mediante lecturas directas.
- **`/actions/export-operative-data.ts`** *Server Action* dedicada para la exportación "Safe-Field". A diferencia de la exportación maestra, esta acción aplica filtros de proyección en Firestore para excluir campos sensibles (`address`, `geopoint`) antes de generar el archivo, cumpliendo el protocolo DLP sin depender del enmascaramiento post-lectura.
- **`/actions/admin-override-logistics.ts`** Acción exclusiva para el rol ADMIN que permite editar el `votingStationId` de un usuario incluso durante el periodo de veda (Time-Lock), registrando obligatoriamente el evento `FORCE_UPDATE_VOTING_STATION` en los logs de auditoría.

---

3. Librerías y Lógica de Negocio (`/lib`)

- **`/lib/firebase.ts`** Inicialización Singleton del cliente Firebase y Admin SDK. Gestiona credenciales segregadas según entorno (Edge vs Node).
- **`/lib/auth.config.ts`** Configuración de Auth segregada para compatibilidad con Edge Runtime del Middleware (RF-8.1).
- **`/lib/auth.ts`** Lógica completa de NextAuth v4 para servidor. Maneja adaptadores Firestore y lógica de sesión compleja no apta para Edge.
- **`/lib/db/schema.ts`** Definiciones de tipos TypeScript (`User`, `Campaign`, `Transaction`) estrictamente alineados con el esquema Firestore (RF-4.2).
- **`/lib/db/converters.ts`** Convertidores de Firestore data <-> App Model. Garantiza tipado estricto y manejo de fechas UTC en lecturas/escrituras.
- **`/lib/services/platform-admin.ts`** Capa de servicio para lógica de SUPER_ADMIN (Aprobación de solicitudes, gestión de Tenants globales).
- **`/lib/services/campaign-manager.ts`** Lógica centralizada para creación de campañas, validación de scopes geográficos y configuración inicial.
- **`/lib/services/offline-sync.ts`** Motor de "Modo Vereda" (RF-3.5). Gestión de IndexedDB, cola de sincronización, estrategia Backoff y resolución de conflictos.
- **`/lib/services/maps.ts`** Utilidades geoespaciales. Wrapper de `Turf.js` para validación de polígonos, cálculo de áreas y "Point-in-Polygon" (RF-6.6).
- **`/lib/services/finance-ledger.ts`** Lógica transaccional financiera (RF-5.6). Manejo de consistencia ACID para gastos y recargas con "Fail-Safe Policy".
- **`/lib/services/audit-logger.ts`** Servicio centralizado de auditoría. Estandariza payloads para `audit_logs` y detecta anomalías de scraping (RF-8.8).
- **`/lib/services/storage.ts`** Servicio de abstracción para subida de archivos a GCS (Avatares, E-14) con compresión en cliente (RNF-5).
- **`/lib/services/event-manager.ts`** Lógica de negocio para creación de eventos y validación de asistencia por proximidad (Geo-Check-in) (RF-1.13).
- **`/lib/services/kyc-provider.ts`** Adaptador para servicios de validación de identidad (Truora/MetaMap) (RF-1.16).
- **`/lib/services/session-manager.ts`** Lógica de seguridad para control de concurrencia de dispositivos, rotación de tokens y Kill-Switch (RF-8.9).
- **`/lib/utils/formatters.ts`** Helpers para formato de moneda (USD), fechas UTC a local y normalización de `SearchKey` (RF-7.6).
- **`/lib/utils/csv-parser.ts`** Utilidad de parseo para importación (RF-1.11). Validación de cabeceras, limpieza de inyecciones y normalización.
- **`/lib/utils/constants.ts`** Constantes del sistema (Límites de jerarquía = 20, Timeouts, Roles, Configuración de reintentos, Precios).
- **`/lib/utils/phone-normalizer.ts`** Utilidad aislada para la **Estandarización Telefónica (RF-1.11)** y limpieza de números (eliminar +57, espacios, guiones) antes de la validación de unicidad o envío a Twilio. Crítico para evitar duplicados por formato.
- **`/lib/utils/math-precision.ts`** Librería para manejo estricto de aritmética de punto flotante. Garantiza que los cálculos de saldo y costos de SMS (4 decimales internos, 2 visuales) no sufran errores de redondeo JavaScript (`0.1 + 0.2 !== 0.3`).
- **`/lib/utils/file-validation.ts`** Validaciones *Client-Side* para archivos CSV. Verifica cabeceras obligatorias, detecta fórmulas maliciosas (CSV Injection) y cuenta filas para aplicar el "Hard Cap" de 5,000 registros antes de permitir la subida.
- **`/lib/hooks/use-offline.ts`** Hook para detectar estado de red, coordinar la sincronización de datos y gestionar el indicador visual (RF-3.5).
- **`/lib/hooks/use-role.ts`** Hook para validación de permisos (RBAC) en componentes de cliente (RF-8.2).
- **`/lib/hooks/use-geolocation.ts`** Hook para obtener GPS del dispositivo con manejo robusto de errores, timeouts y fallback a manual (RF-6.1).
- **`/lib/hooks/use-safe-mode.ts`** Hook que detecta el flag de "Modo Contingencia" (RF-3.10) y adapta la UI desactivando mapas y funciones pesadas.
- **`/lib/hooks/use-audit-view.ts`** Hook "The Silent Watcher". Se monta en las vistas de perfil (`[id]/page.tsx`) y dispara automáticamente el evento `VIEW_PROFILE` a la API de auditoría si el usuario logueado es distinto al perfil visitado, manejando *debouncing* para evitar spam de logs.
- **`/lib/hooks/use-geocoding-quota.ts`** Hook de "Circuit Breaker" en cliente. Consulta el consumo diario de geocodificación desde `system_config` y determina si el componente de Mapa debe cargarse o activar el "Modo Fallback" preventivo para ahorrar costos.

- **`/lib/contexts/CampaignContext.tsx`** Context Provider para manejar el estado de la campaña activa y su configuración global. Persiste selección en localStorage.
- **`/lib/contexts/ThemeContext.tsx`** Manejo de variables CSS dinámicas para identidad visual de campaña (RF-3.7). Asegura contraste WCAG AA.
- **`/lib/services/survey-engine.ts`** Motor de lógica para Encuestas (RF-5.8). Valida cardinalidad (Min/Max selecciones), consolida respuestas en ventanas de tiempo y formatea el payload para almacenamiento eficiente en Firestore.
- **`/lib/utils/topology-validator.ts`** Librería pura de geometría (basada en Turf.js) para ejecutar las validaciones topológicas complejas (RF-6.6) en el cliente y servidor: detección de auto-intersecciones, polígonos dentro de polígonos ("Donuts") y simplificación de vértices.
- **`/lib/utils/device-fingerprint.ts`** Utilidad para generar un hash de dispositivo consistente (User Agent + Canvas + Screen Resolution). Soporta la lógica de límite de sesiones (RF-8.9) y detección de fraude por velocidad (RF-1.10).
- **`/lib/db/indexeddb-schema.ts`** Definición formal del esquema de la base de datos local (Dexie.js o IDB raw). Separa la lógica de sincronización (`offline-sync.ts`) de la estructura de datos local, definiendo índices para `pending_uploads` y `sync_queue` necesarios para la persistencia offline (RF-3.5).
- **`/lib/hooks/use-confetti.ts`** Hook para gestionar la micro-interacción de celebración (Confeti Virtual) mencionada en RF-4.5 cuando un líder alcanza su meta o realiza su primer registro. Carga la librería dinámicamente para no afectar el bundle inicial.
- **`/lib/utils/hierarchy-manager.ts`** Implementa las validaciones matemáticas del **RF-1.9**. Funciones para calcular la profundidad del árbol (max 20 niveles) y detectar ciclos (A es jefe de B, B es jefe de A) antes de permitir cualquier operación de escritura en `hierarchyPath`.
- **`/lib/utils/search-key-generator.ts`** Implementa el algoritmo de normalización del **RF-7.6**. Convierte nombres y alias a formato canónico (Unicode NFD, sin diacríticos) para construir el campo `searchKey` utilizado en el motor de búsqueda indexado.
- **`/lib/constants/legal-texts.ts`** Archivo de configuración estática para los **Anexos A y B (RF-13)**. Centraliza los textos legales de Habeas Data y Autorización de WhatsApp para facilitar su actualización y versionado (Protocolo Legal Wall RF-3.8) sin tocar componentes UI.
- **`/lib/analytics/kpi-calculator.ts`** Librería pura para estandarizar las fórmulas de los **KPIs (RF-7.1)**. Asegura que el cálculo de "Saturación de Territorio" y "Cumplimiento de Meta" sea idéntico tanto en el Frontend (para visualización rápida) como en los Triggers de Backend.
- **`/lib/hooks/use-dashboard-metrics.ts`** Implementa la estrategia de **Caché SWR (RF-7.9)**. Wrapper sobre las peticiones de métricas que gestiona el TTL de 5 minutos y la invalidación automática ante mutaciones locales, protegiendo las cuotas de lectura de Firestore.
- **`/lib/hooks/use-sharded-count.ts`** Hook necesario para leer los contadores distribuidos (**RF-4.5 Sharded Counters**). Agrega (suma) en el cliente los valores de los fragmentos de la subcolección `shards` para mostrar totales precisos en tiempo real en nodos de alto tráfico.
- **`/lib/services/mailer.ts`** Servicio de abstracción para el envío de correos electrónicos transaccionales. Esencial para **RF-1.7 (Magic Code de Recuperación)**, **RF-5.6 (Alertas de Saldo Bajo)** y **RF-4.3 (Notificación de Exportación Segura)**. Centraliza la configuración del proveedor (ej: Resend/SendGrid) y el manejo de fallos.
- **`/lib/constants/regex-patterns.ts`** Centralización de expresiones regulares críticas para **RF-1.1 (Validación de Campos)** y **RF-1.11 (Limpieza de CSV)**. Contiene patrones probados para Cédulas, Celulares colombianos (+57), detección de fórmulas maliciosas (CSV Injection) y validación de contraseñas fuertes.
- **`/lib/utils/error-handler.ts`** Utilidad para normalizar errores. Captura excepciones de Firebase, Zod o Twilio y las transforma en objetos de respuesta estandarizados (`ApiError`) con códigos HTTP correctos y mensajes seguros para el cliente (evitando exponer stack traces), crucial para RNF-9.
- **`/lib/utils/anonymizer.ts`** Implementa la lógica de **Sobrescritura Destructiva (RF-8.3)**. Contiene las funciones puras para generar hashes de descarte (`deleted_[TIMESTAMP]`) y limpiar campos PII, asegurando que la lógica del "Derecho al Olvido" sea testearle unitariamente y reutilizable en scripts de mantenimiento.
- **`/lib/hooks/use-notifications.ts`** Hook para gestionar la suscripción a notificaciones In-App. Utiliza un *polling* inteligente (o suscripción Firestore si el costo lo permite) para actualizar el contador del icono de campana en tiempo real, respetando la **Regla de Silencio (Do Not Disturb)** configurada.
- **`/lib/utils/string-normalization.ts`** Extracción de la lógica de normalización Unicode NFD descrita en **RF-7.6**. Se separa de `search-key-generator.ts` para poder ser reutilizada en el cliente (para búsquedas optimistas en listas cargadas) y en el servidor.
- **`/lib/stores/map-editor-store.ts`** Gestor de estado para el **Módulo de Mapas (RF-6.6)**. Almacena las coordenadas temporales del polígono en edición, el historial de cambios (Undo/Redo para el dibujo), y el estado de la herramienta seleccionada. Desacopla la lógica de dibujo de los componentes de UI (`DrawingControls` y `TerritoryMap`), mejorando el rendimiento en móviles.
- **`/lib/stores/offline-queue-store.ts`** Store reactivo para la **Cola de Sincronización (RF-3.5)**. Se conecta con IndexedDB para mantener en memoria el conteo real de `pendingItems` y los estados de error (`PAUSED_NETWORK`, `PAUSED_AUTH`). Es la fuente de verdad para el componente `OfflineIndicator` y evita lecturas constantes a disco.
- **`/lib/hooks/use-form-persistence.ts`** Hook para el **Formulario de Registro (RF-1.1)**. Guarda automáticamente en `localStorage` o `IndexedDB` cada campo que el usuario escribe ("Draft Saving"). Si la PWA se cierra o recarga por error (RNF-3), este hook restaura el estado del formulario para no perder datos, cumpliendo con la "UX No Intrusiva" del manejo de sesión.
- **`/lib/hooks/use-media-query.ts`** Hook para **RNF-6 (UX de Incompatibilidad)** y **RF-6.6 (Restricción de Edición)**. Detecta cambios en el viewport y orientación del dispositivo en tiempo real para activar/desactivar el modo de edición de mapas en tabletas verticales o teléfonos, mostrando las alertas responsivas correspondientes.
- **`/lib/hooks/use-debounce.ts`** Hook necesario para el **Motor de Búsqueda Omnipresente (RF-7.6)**. Retrasa la ejecución de las consultas al índice de Firestore mientras el usuario escribe en el input global, protegiendo la cuota de lecturas y evitando "parpadeos" en la UI.
- **`/lib/utils/date-helpers.ts`** Separado de `formatters.ts`. Implementa estrictamente la **Global Timezone Policy (RF-4.2)**. Contiene funciones para convertir timestamps UTC de la base de datos a la hora local del dispositivo ("Client-Side Resolution") para logs de auditoría y fechas de registro, usando librerías ligeras como `date-fns` o `Intl`.
- **`/lib/utils/sms-segment-calculator.ts`** Utilidad crítica compartida entre Cliente y Servidor para calcular segmentos de mensajes SMS. Debe detectar caracteres *GSM-7* vs *UCS-2* (emojis, tildes) y calcular el multiplicador de costo exacto antes de enviar. Esencial para que el `CostEstimator.tsx` (Frontend) y la validación de API (Backend) coincidan matemáticamente, evitando fugas de presupuesto por errores de codificación.
- **`/lib/utils/export-fingerprinter.ts`** Implementa la lógica de "Marcado Forense". Recibe el stream de datos crudos y el objeto de auditoría, e inyecta las primeras 5 filas obligatorias de metadatos (Responsable, IP, Timestamp) desplazando los datos reales hacia abajo. Se aísla aquí para garantizar que esta manipulación de buffer no corrompa el formato CSV/Excel generado.
- **`/lib/utils/ip-cidr-matcher.ts`** Utilidad ligera para parsear y validar direcciones IP contra rangos CIDR (ej: `192.168.1.0/24`). Necesaria para el `middleware.ts` y el `system_config`, permitiendo que las IPs de las sedes de campaña (guardadas en configuración) eviten los bloqueos de tasa sin hardcodear IPs en el código fuente.
- **`/lib/services/messaging-compliance.ts`** **RF-5.1:** Servicio aislado para la "Estrategia de Calentamiento" (Warm-up) y "Health Score Guard". Contiene la lógica para fragmentar envíos a usuarios importados (máx 500/día) y validar reglas de *Cold Start*, separando esta complejidad del `messaging-worker`.
- **`/lib/services/rate-limit-service.ts`** **RF-8.7 / RF-8.10:** Servicio de limitación de tasa distribuido. A diferencia del middleware (stateless), este servicio utiliza contadores atómicos en Firestore/Redis para bloquear IPs abusivas o intentos de OTP a nivel global de infraestructura, persistiendo el estado entre instancias de Cloud Run.
- **`/lib/utils/color-contrast.ts`** **RF-3.7 / RNF-4:** Utilidad matemática para calcular la luminancia relativa del color primario configurado por la campaña. Determina automáticamente si el texto sobre botones debe ser Blanco o Negro para cumplir estrictamente con **WCAG AA** bajo luz solar directa.
- **`/lib/utils/time-windows.ts`** **RF-5.8.C / RF-8.11:** Helpers para cálculos de ventanas de tiempo deslizantes. Necesario para la regla "Anti-Loop" de auto-respuestas (1 vez cada 24h) y para determinar si un log de auditoría ha expirado y debe moverse a Cold Storage.
- **`/lib/hooks/use-network-quality.ts`** **RF-3.10 / RNF-3:** Hook avanzado que no solo detecta `offline`, sino la "calidad efectiva" de la conexión (RTT/Downlink) para activar automáticamente el **Modo Emergencia (Lite Version)** en redes 3G inestables, desmontando mapas preventivamente.
- **`/lib/services/web-push-client.ts`** Servicio de bajo nivel para interactuar con el protocolo VAPID (Web Push). Abstrae la librería `web-push` para firmar payloads y manejar errores de suscripciones caducadas (410 Gone) que deben limpiarse de la BD para mantener la higiene de datos del **RF-5.10**.
- **`/lib/utils/zone-math.ts`** Separación de lógica del **RF-6.6**. Contiene funciones puras para calcular intersecciones de polígonos y áreas de superposición

en metros cuadrados. Esencial para el componente `ZoneConflictResolver` y para la validación de "No Donuts" en el backend, desacoplándolo de la librería general de mapas.

- **`/lib/constants/default-feature-flags.ts`** Fallback estático de configuración. Si la lectura de `system_config` en Firestore falla o demora (Red inestable), este archivo provee los valores por defecto seguros (ej: `emergencyMode: false`, `allowNewRegistrations: true`) para garantizar la continuidad operativa (RNF-3).
- **`/lib/types/whatsapp-flow-schema.d.ts`** Definiciones de tipos para los flujos estructurados de WhatsApp (RF-5.8). Asegura que el constructor de encuestas (`SurveyBuilder`) genere JSONs válidos según la especificación estricta de la API de Meta/Twilio.
- **`/lib/types/external-integration.d.ts`** Definiciones de tipos estrictas para las respuestas de APIs de terceros no cubiertas por los SDKs estándar: Payloads de Webhooks de Meta (Estados de Plantilla), respuestas de verificación de Truora/MetaMap y estructuras de error de proveedores de SMS.
- **`/lib/constants/retention-policies.ts`** **RF-8.11:** Archivo de configuración que define los TTLs (Time-To-Live) para las diferentes colecciones (ej: `AUDIT_LOG_TTL = 180`, `SOFT_DELETE_GRACE_PERIOD = 30`). Centraliza las reglas de limpieza para los Cron Jobs de mantenimiento.
- **`/lib/services/circuit-breaker-manager.ts`** **RF-5.6 / RF-6.5:** Utilidad genérica para implementar lógica de "Parada de Emergencia". Gestiona estados de apertura/cierre de circuitos basados en umbrales de error o presupuesto, utilizada tanto en el cliente (Mapas) como en el servidor (Gasto Financiero).
- **`/lib/config/routes.ts`** Centraliza la definición de rutas públicas (`/login`, `/go/*`), protegidas y administrativas. Es consumido por el `middleware.ts` para aplicar la lógica de **Integridad de Sesión (RF-8.9)** sin hardcodear strings en el middleware.
- **`/lib/fonts.ts`** Configuración optimizada de `next/font/google`. Carga las fuentes `Inter` y `Roboto` (RNF-4 Diseño) con estrategias de carga eficientes para evitar *Layout Shifts* en conexiones 3G (RNF-1).
- **`/lib/hooks/use-toast.ts`** Hook personalizado para disparar las notificaciones visuales (`toaster.tsx`). Esencial para mostrar el feedback de "Guardando localmente..." (RF-3.5) o alertas de errores de API.
- **`/lib/utils/geohash.ts`** **RF-4.2.G / RF-6.7:** Utilidad matemática esencial para la indexación geoespacial en Firestore. Contiene funciones para codificar coordenadas (Lat/Lng) a cadenas Geohash y calcular los "vecinos" (neighbors) para consultas de proximidad eficientes. Necesario para que el backend busque "Puestos de Votación cercanos" sin descargar toda la colección `voting_places`.
- **`/lib/security/csrf-protection.ts`** **RF-8.5 / RF-8.7:** Middleware de seguridad adicional para rutas de API críticas (`/api/auth/*`, `/api/admin/*`). Implementa validación de tokens Anti-CSRF para prevenir ataques de falsificación de peticiones en navegadores antiguos donde las cookies `SameSite` podrían no ser suficientes, reforzando la seguridad del login omnicanal.
- **`/lib/config/secrets-loader.ts`** Utilidad para el **RF-8.10 (Gestión Centralizada de Secretos)**. Dado que Cloud Run monta los secretos como volúmenes (archivos) y no solo como variables de entorno, este script debe leer los archivos en `/secrets/` (ej: `TWILIO_AUTH_TOKEN`) y exponerlos a la aplicación Node.js, fallando de forma segura si no existen.
- **`/lib/hooks/use-storage-quota.ts`** Hook crítico para **RF-3.5.C (Protocolo de Saturación)**. Utiliza la API `navigator.storage.estimate()` para monitorear el espacio disponible en el dispositivo móvil. Debe exponer banderas reactivas (`isQuotaExceeded`, `isCritical`) para activar el bloqueo preventivo de nuevos registros antes de que ocurra corrupción de datos (Eviction).
- **`/lib/utils/whatsapp-markdown.ts`** Utilidad de formateo para **RF-5.3 (Editor Visual)**. Convierte el texto con formato de WhatsApp (*negrita*, *cursiva*, ~~tachado~~) a HTML seguro para la previsualización en el componente `WhatsAppPreview.tsx`. Garantiza que lo que ve el Admin sea idéntico a lo que renderiza la app móvil de Meta.
- **`/lib/constants/colombia-geo.ts`** Constantes estáticas con la lista básica de Departamentos y Capitales de Colombia. Sirve como *fallback* inmediato para los selectores de ubicación (RF-1.1) y para la configuración inicial de `master_geo_structure` (Colección O, Pág. 54) antes de que se sincronice la Divipol completa.
- **`/lib/db/query-scopes.ts`** Abstracción centralizada para **RF-7.2 (Data Scoping)**. Exporta funciones como `applyHierarchyFilter(query, user)` que inyectan automáticamente las cláusulas `.where()` requeridas según el rol del usuario (Coordinador vs Enlace), previniendo fugas de datos por errores manuales en las rutas de API.
- **`/lib/services/task-queue-client.ts`** Cliente tipado para encolar tareas en Google Cloud Tasks. Abstrae la complejidad de autenticación OIDC y la construcción de rutas de colas (`projects/.../queues/...`). Es el "Productor" genérico utilizado por `messaging-compliance.ts` y `import.ts`.
- **`/lib/security/twilio-validate.ts`** Middleware de verificación de firma (`X-Twilio-Signature`) para los Webhooks. Garantiza que las peticiones a `/api/webhooks/twilio/*` provengan realmente de Twilio y no de un atacante simulando eventos de entrega o respuestas falsas.
- **`/lib/utils/sharding-constants.ts`** Define las constantes compartidas para la lógica de contadores distribuidos (ej: `NUM_SHARDS = 10`). Debe ser importado tanto por el hook de lectura (`use-sharded-count.ts`) como por el trigger de escritura (`aggregations.ts`) para asegurar consistencia matemática.
- **`/lib/types/gamification.d.ts`** Definiciones de TypeScript para los niveles de incentivo visual ("Rojo", "Amarillo", "Verde") y las interfaces de metas personales. Centraliza los umbrales lógicos para que coincidan en el Dashboard y en las notificaciones.
- **`/lib/constants/role-permissions.ts`** Centralización de la Matriz RBAC en un objeto constante tipado. Permite que `use-role.ts` y `RoleGuard.tsx` validen permisos dinámicamente sin hardcodear reglas (ej: `CAN_EXPORT_EXCEL = ['ADMIN', 'SUPER_ADMIN']`).
- **`/lib/services/image-processor-server.ts`** Lógica de servidor (Node.js/Sharp) para validación forense de imágenes E-14 y Avatares. Aunque hay compresión en cliente (RNF-5), este servicio valida integridad de metadatos y re-comprime para almacenamiento final en *Cold Storage*.
- **`/lib/utils/masking.ts`** Utilidades para la sanitización de datos (DLP). Funciones para enmascarar cédulas (`******1234`) y teléfonos en el "Modo Operativo" de exportación.
- **`/lib/services/response-classifier.ts`** Lógica pura para distinguir entre "Ruido Conversacional" (Hola, Gracias, Memes) y "Payloads de Datos" (Respuestas a botones). Separa esta heurística del webhook principal de Twilio.
- **`/lib/hooks/use-idle-timer.ts`** Hook de cliente para detectar inactividad del usuario (sin movimiento de mouse/touch). Necesario para el cierre de sesión preventivo en dispositivos compartidos o públicos (Sedes de Campaña).
- **`/lib/hooks/use-infinite-scroll.ts`** Abstracción del `IntersectionObserver`. Hook genérico reutilizable para manejar la paginación por cursor en las tablas de Usuarios, Logs y Auditoría, desacoplando la lógica de UI de la lógica de carga de datos.
- **`/lib/hooks/use-clipboard.ts`** Hook para manejar la copia segura de enlaces al portapapeles en dispositivos móviles, gestionando permisos de navegador y feedback visual (Toast "Enlace Copiado") en flujos de compartir invitación.
- **`/lib/utils/currency.ts`** Utilidad estricta para el formateo de montos. Centraliza la lógica de visualización "USD" con 2 decimales y previene la conversión accidental a COP en la interfaz, manejando correctamente los redondeos de la base de datos (4 decimales).
- **`/lib/utils/name-formatter.ts`** Utilidad para la normalización visual de nombres (Capitalización de Títulos, manejo de apellidos compuestos) en la UI, mejorando la presentación de las credenciales digitales y reportes impresos.
- **`/lib/constants/http-codes.ts`** Centralización de códigos de estado HTTP semánticos (`429 Too Many Requests`, `402 Payment Required`). Asegura que el manejo de errores en el Frontend (especialmente los reintentos offline) reaccione consistentemente a los códigos del Backend.
- **`/lib/contexts/SystemConfigContext.tsx`** Proveedor de React que carga y cachea el documento singleton `system_config`. Distribuye globalmente las tarifas (`costPerMessage`), banderas de emergencia (`maintenanceMode`) y la versión legal actual a toda la app sin invocar la base de datos en cada navegación.
- **`/lib/hooks/use-system-config.ts`** Hook de consumo para el contexto anterior. Expone métodos directos como `isEmergencyMode()`, `getCurrentPricing()` y `checkLegalVersion(userVersion)` para simplificar la lógica en componentes visuales.
- **`/lib/middleware/geoblock-logic.ts`** Función ligera que valida la IP entrante contra una lista de países permitidos (CO, US, ES) y la lista blanca dinámica de sedes (`hqWhitelistedIps`). Se separa para ser testeable unitariamente.
- **`/lib/middleware/rate-limit-logic.ts`** Implementación de la lógica de *Token Bucket* o ventana deslizante utilizando Redis (si hay disponible) o memoria efímera del Edge para frenar ataques de fuerza bruta antes de que toquen la API.
- **`/lib/services/batch-writer.ts`** Utilidad para gestionar escrituras masivas en Firestore respetando el límite estricto de 500 operaciones por *commit*. Abstrae la lógica de dividir arrays grandes en "chunks" y reintentar fallos parciales.

- **`/lib/utils/browser-compression.ts`** Utilidad cliente explícita para usar `browser-image-compression`. Se separa de `image-optimization.ts` para garantizar que no se importen librerías de Node.js (como `sharp`) en el bundle del cliente, lo cual rompería la build.
- **`/lib/services/hierarchy-arbitrator.ts`** Servicio puro que encapsula la lógica del "Juez Jerárquico". Recibe el `hierarchyPath` del solicitante y determina dinámicamente quién es el `approverId` (Coordinador de Zona vs. Admin) aplicando la matriz de escalamiento y reglas de conflicto de interés. Desacopla esta complejidad de `disputes/create/route.ts`.
- **`/lib/services/integrity-monitor.ts`** Lógica de fondo para los Cron Jobs de seguridad. Implementa algoritmos de detección de ciclos infinitos ("Loop Detection") y escaneo de orfandad jerárquica que no se pueden ejecutar en tiempo real durante las escrituras.
- **`/lib/utils/qr-generator-server.ts`** Abstracción de `node-qrcode` para ejecución exclusiva en servidor. Genera el buffer SVG/PNG con configuraciones de corrección de error 'H' y colores de marca, separando la generación técnica de la lógica de negocio de las Server Actions.
- **`/lib/utils/color-generator.ts`** Utilidad para generar colores hexadecimales distintos y accesibles automáticamente para las nuevas zonas territoriales si el usuario no selecciona uno, garantizando contraste visual en el mapa.
- **`/lib/constants/system-messages.ts`** Centraliza los textos de los mensajes automáticos del sistema ("Plantilla de Desvío/Disclaimer", Mensajes de "Modo Vereda"). Evita hardcodear strings en los webhooks y permite corrección rápida de copy.
- **`/lib/utils/geo-scope-checker.ts`** Librería de utilidades puras que valida si una ubicación dada (Departamento/Municipio) se encuentra dentro del objeto `scope` definido en la configuración de la campaña (`political_campaigns`). Centraliza la lógica de "Geo-Fence" utilizada tanto en el registro (Validación de Admisión) como en los triggers de base de datos para detectar expatriación electoral.
- **`/lib/hooks/use-campaign-hierarchy.ts`** Hook que consume la configuración `campaign.settings.hierarchy`. Determina si roles intermedios (como Coordinador Regional) están activos o colapsados ("Compresión de Roles"). Esencial para que los formularios de creación de usuarios y los selectores de filtros oculten niveles jerárquicos irrelevantes en campañas locales pequeñas.
- **`/lib/hooks/use-hierarchy-check.ts`** Hook de seguridad cliente para validar relaciones de ancestro/descendiente. Permite a la UI habilitar/deshabilitar botones de "Auditoría" o "Ver Perfil" instantáneamente verificando si `targetUser.hierarchyPath` incluye el UID del usuario actual, sin hacer peticiones fallidas al servidor.
- **`/lib/hooks/use-page-visibility.ts`** Hook para detectar si la pestaña del navegador está activa o en segundo plano. Se usa para pausar automáticamente el *polling* de notificaciones y la actualización de mapas en tiempo real, ahorrando batería en móviles y lecturas de base de datos cuando el usuario no está mirando.
- **`/lib/utils/geo-containment.ts`** Lógica matemática separada de `topology-validator`. Verifica si una coordenada (Lat/Lng) está contenida dentro de los límites administrativos oficiales del Municipio/Departamento (GeoJSON importado de `master_geo_structure`). Previene registros válidos técnicamente pero fuera de la jurisdicción electoral legal.
- **`/lib/interfaces/sms-provider.interface.ts`** Contrato estricto (`Interface`) que define los métodos `sendSMS` y `getTemplateStatus`. Permite desacoplar la lógica de negocio de la implementación específica de Twilio, facilitando la inyección de mocks para pruebas o el cambio futuro de proveedor.
- **`/lib/adapters/twilio-live.adapter.ts`** Implementación real del proveedor para **Producción**. Consume `twilio-client` y maneja la lógica de reintentos, validación de `Health Score` y manejo de errores de API reales.
- **`/lib/adapters/twilio-mock.adapter.ts`** Implementación simulada para **Staging/QA** (RF-8). Intercepta los envíos a números mágicos, simula latencia de red y retorna respuestas predecibles (Éxito/Fallo) sin gastar saldo, permitiendo pruebas de carga (k6) seguras.
- **`/lib/utils/cn.ts`** Utilidad estándar para la fusión condicional de clases Tailwind (`clsx` + `tailwind-merge`). Indispensable para crear componentes reutilizables que acepten clases personalizadas sin conflictos de especificidad CSS.
- **`/lib/utils/vibration.ts`** Utilidad para invocar el motor de vibración del dispositivo (`navigator.vibrate`) en eventos clave (Error de Validación, Éxito de Escaneo QR). Refuerza el feedback táctil en entornos ruidosos o bajo luz solar (RNF-4).
- **`/lib/db/dexie-instance.ts`** Instancia Singleton inicializada de la base de datos local (usando Dexie.js o idb). Centraliza la gestión de versiones del esquema local y migraciones, asegurando que `offline-sync.ts` y los hooks de React consuman la misma conexión abierta.
- **`/lib/hooks/use-map-interaction.ts`** Hook para centralizar la lógica de interacción compleja del mapa (clic, drag, touch). Separa la gestión de eventos de Leaflet/Google Maps de la lógica de negocio de la campaña, facilitando el mantenimiento del "Modo Edición" vs "Modo Visualización".
- **`/lib/utils/template-parser.ts`** Utilidad pura para analizar el cuerpo de las plantillas de WhatsApp. Detecta variables, cuenta caracteres UCS-2 vs GSM-7 y valida la estructura contra las reglas de Meta antes de permitir el guardado.
- **`/lib/utils/signed-url-cache.ts`** Gestor de caché en memoria para las URLs firmadas de GCS. Evita solicitar una nueva `signedUrl` cada vez que se renderiza una imagen privada (E-14) si la anterior aún es válida, reduciendo latencia y costos de API.
- **`/lib/utils/boomerang-logic.ts`** Lógica de negocio aislada para determinar si un usuario que cambia de ubicación activa el "Protocolo Boomerang". Compara fechas de `createdAt` y estados previos en `audit_logs` para decidir si restaurar la antigüedad o tratarlo como nuevo.
- **`/lib/services/feature-flags-client.ts`** Cliente ligero para consultar Feature Flags en el frontend con estrategia de *caching* y *fallback*. Asegura que si Firebase Remote Config falla, la app use los valores seguros predeterminados (`default-feature-flags.ts`) instantáneamente.
- **`/lib/services/otp-manager.ts`** Servicio que desacopla la lógica de OTP de `auth-actions.ts`. Centraliza la generación de códigos criptográficamente seguros, el hashing antes de guardar en BD y la validación de ventanas de tiempo, permitiendo su reutilización en el "Login", "Recuperación de Cuenta" y "Validación de Transferencia".
- **`/lib/utils/password-strength.ts`** Utilidad de cliente para evaluar la entropía de la contraseña durante el registro de Admins (si usan credenciales). Asegura cumplimiento de políticas NIST (longitud, complejidad) antes de enviar el formulario.
- **`/lib/hooks/use-local-storage.ts`** Hook genérico tipado para manejar la lectura/escritura en `localStorage` con hidratación segura en Next.js (evitando errores de "Hydration Mismatch"). Esencial para recordar `lastSelectedCampaignId` y preferencias de UI (Colapso de Sidebar).
- **`/lib/utils/validators-async.ts`** Separación de validaciones asíncronas (consultar si Cédula existe en BD) de las validaciones síncronas (Zod). Permite implementar "Debounced Validation" en los formularios para dar feedback visual sin saturar la API `check-uniqueness`.
- **`/lib/firebase-admin.ts`** Inicialización segregada del SDK `firebase-admin`. A diferencia de `lib/firebase.ts` (Cliente), este archivo debe manejar las credenciales de *Service Account* inyectadas de forma segura. Separarlo evita errores de compilación en Next.js (Webpack) al prevenir que librerías de Node.js se filtren al bundle del cliente.
- **`/lib/stores/wizard-store.ts`** Store de estado global (Zustand) para el Wizard "Day Zero". Dado que la configuración inicial (Político -> Campaña -> Presupuesto) abarca múltiples rutas (`/setup/*`), se requiere persistencia de estado en memoria para evitar la pérdida de datos si el usuario navega entre pasos antes de guardar la transacción final.
- **`/lib/validations/auth.ts`** Esquemas Zod aislados para Login, OTP y Recuperación. Separado de `validators.ts` para reducir el tamaño del bundle en la página de Login crítica (RNF-1).
- **`/lib/validations/campaign.ts`** Esquemas de validación complejos para la creación de campañas y configuración financiera. Incluye reglas de negocio para los límites de presupuesto (`maxHourlySpendLimit`) y validación de colores hexadecimales para el Branding.
- **`/lib/validations/geo.ts`** Validadores específicos para objetos GeoJSON y topología. Asegura que los polígonos dibujados cumplan con la regla "No Donuts" y límite de vértices antes de enviarlos al servidor.
- **`/lib/constants/mime-types.ts`** Lista blanca estricta de tipos MIME permitidos (`image/jpeg`, `image/webp`, `image/png`) y sus "Magic Numbers" correspondientes para validación de archivos en `file-uploader.tsx` y `storage.rules`, previniendo la subida de ejecutables disfrazados.
- **`/lib/constants/twilio-error-codes.ts`** Diccionario de constantes para los códigos de error de la API de Twilio. Esencial para **RF-5.7 (Protocolo de Revocación)**, evitando el uso de "números mágicos" como `21610` (Unsubscribed) o `30008` (Unknown) en la lógica de negocio, facilitando el mantenimiento de las reglas de reintento y reembolso.
- **`/lib/hooks/use-before-unload.ts`** Hook de cliente que intercepta el evento `window.onbeforeunload`. Implementa la protección de **Integridad de Datos** para el formulario de registro (RF-1.1) y la edición de mapas, alertando al usuario si intenta cerrar la pestaña o recargar con cambios sin guardar ("Dirty State"), complementando la persistencia de `use-form-persistence.ts`.
- **`/lib/utils/excel-generator.ts`** Utilidad para generar reportes en formato `.xlsx` nativo. Necesario porque el requerimiento menciona "Excel/CSV" y

CSV no soporta múltiples hojas (útil para separar "Líderes" de "Seguidores" en un solo reporte de auditoría).

- **`/lib/hooks/use-long-press.ts`** Hook de accesibilidad para interfaces táctiles. Permite acciones secundarias en listas (ej: selección múltiple de usuarios para reasignación) mediante presión larga, optimizando la UX en móviles donde el espacio para checkboxes es limitado.
- **`/lib/constants/meta-policy-limits.ts`** Centralización de reglas de validación de Meta (Longitud de Header, Tipos de archivo soportados, Categorías permitidas). Evita rechazos de plantillas validando las restricciones del BSP antes de enviar la petición a la API.
- **`/lib/middleware/session-guard.ts`** Lógica aislada para la validación de sesión y **Integridad de Dispositivos (RF-8.9)**. Verifica la validez del token y detecta cambios de rol en tiempo real, permitiendo que el archivo `middleware.ts` principal importe esta función para mantener el código limpio y testeable.
- **`/lib/middleware/legal-guard.ts`** Implementación de la lógica del **Legal Wall (RF-3.8)**. Compara la versión de los términos aceptados por el usuario contra la configuración global y decide si redirigir forzosamente a `/legal/update-required`, separando esta responsabilidad crítica de la autenticación.
- **`/lib/db/repositories/offline-repository.ts`** Capa de abstracción sobre `Dexie.js`. Centraliza las operaciones CRUD directas contra IndexedDB para el **Modo Vereda (RF-3.5)**. Evita que los componentes de React conozcan la implementación de la base de datos local, facilitando pruebas unitarias de la lógica de persistencia sin navegador.
- **`/lib/services/sync-strategies.ts`** Define las estrategias de reintento específicas (**Backoff Exponencial** vs **Reintento Silencioso**) descritas en **RF-3.5**. Separa la *política* de sincronización de la *ejecución* (que reside en `offline-sync.ts`), permitiendo ajustar los tiempos de espera y límites de reintento según el tipo de error (5xx vs 401) de forma aislada.
- **`/lib/utils/high-contrast.ts`** Utilidad extendida para **RF-3.7 / RNF-4**. No solo calcula contraste blanco/negro, sino que implementa la lógica para forzar bordes sólidos (`1px solid #000`) en tarjetas y botones cuando se detecta el modo de "Alta Visibilidad" para operaciones bajo luz solar directa.
- **`/lib/utils/vibration-patterns.ts`** Centraliza los patrones de retroalimentación háptica (**Haptics**) requeridos para la interacción física en campo (RF-1.1 Éxito de Registro, RF-6.6 Error de Polígono). Permite estandarizar la duración de las vibraciones (ej: `[200] ms` para éxito, `[50, 50, 50]` para error) en toda la PWA.
- **`/lib/types/enums.ts`** Centralización de constantes de cadena críticas para evitar "Magic Strings" dispersos. Debe exportar `UserRole` ('COORDINATOR', 'LINK'...), `ZoneStatus` ('ACTIVE', 'DISPUTED', 'VACANT'), `CampaignStatus` y `FraudStatus`. Esencial para **RF-8.2 (RBAC)** y **RF-6.6 (Estados de Zona)**. Garantiza que el backend (validación Zod) y el frontend (Renderizado Condicional) hablen el mismo idioma estricto.
- **`/lib/types/master-geo.d.ts`** Definición de tipos para la colección `master_geo_structure` (Colección O, Pág. 54). Define la estructura recursiva de `hierarchy` (l1, l2, l3) y las `validationRegex` por país. Necesario para que el componente `VotingStationSelector` y el validador `phone-normalizer.ts` consuman la configuración regional de forma tipada, soportando la expansión futura a otros países sin refactorización.
- **`/lib/validations/survey.ts`** Esquema de validación para el constructor de encuestas (RF-5.3.D). Valida reglas de cardinalidad complejas (`minSelect <= maxSelect`), unicidad de valores en `options` y estructura del payload de respuesta. Separa la lógica de validación de encuestas de la configuración general de la campaña (`campaign.ts`), permitiendo reutilizar el validador en el webhook de entrada (`twilio/inbound`) para filtrar respuestas malformadas.
- **`/lib/validations/legal-versioning.ts`** Esquema para validar la integridad de la aceptación de términos. Verifica que la versión aceptada (`v2.1`) coincida con la activa en `system_config`. Crítico para el **RF-3.8 (Legal Wall)**. Asegura que el payload enviado por el `LegalConsentModal` cumpla con el formato estricto requerido por auditoría antes de escribir en `users.metadata`.
- **`/lib/utils/twilio-error-mapper.ts`** Función pura que traduce códigos de error técnicos de Twilio (ej: `21610`, `30008`) a mensajes amigables en español para el usuario final ("El número no tiene WhatsApp", "Celular apagado"). Complementa a `twilio-error-codes.ts` (que solo tiene constantes). Necesario para mostrar feedback útil en el `TemplateStatusBadge` y en los reportes de campaña (RF-5.7).
- **`/lib/utils/color-palette-generator.ts`** Algoritmo para generar variantes accesibles (Hover, Active, Disabled) basadas en el color primario de la campaña (`--primary-color`). Tailwind necesita estas variantes para mantener la usabilidad de los botones sin obligar al Admin a configurar manualmente cada tono de la paleta.
- **`/lib/env.ts`** Utilidad de validación de variables de entorno en tiempo de ejecución (usando Zod). Garantiza que la aplicación falle inmediatamente al arrancar ("Fail-Fast") si faltan secretos críticos como `TWILIO_AUTH_TOKEN` o `FIREBASE_ADMIN_KEY`, evitando errores silenciosos en producción y asegurando la integridad de la segregación de entornos (Staging vs Prod).
- **`/lib/utils/api-client.ts`** Wrapper tipado sobre `fetch`. Centraliza la inyección de headers de seguridad (CSRF), el manejo de tokens de sesión caducados (RF-3.5) y la normalización de errores (`ErrorHandler`). Esencial para que todos los componentes del cliente consuman la API de forma consistente y segura.
- **`/lib/utils/color-conversion.ts`** Utilidad matemática complementaria a `color-contrast.ts`. Convierte valores Hex (guardados en `political_campaigns.branding`) a valores RGB/HSL para permitir el uso de opacidades con variables CSS en Tailwind (ej: `bg-primary/20`), necesario para los estados *hover* y *focus* en la interfaz adaptativa (RF-3.7).
- **`/lib/validations/person.ts`** Esquema Zod centralizado para la entidad "Persona". Unifica las reglas de validación de Cédula, Teléfono (RF-1.11) y Nombre que se repiten en el Registro, Edición de Perfil, Importación Masiva y Creación de Candidato, asegurando consistencia en todo el ciclo de vida del dato.
- **`/lib/types/leaflet-custom.d.ts`** Definiciones de TypeScript para Leaflet. Dado que el **RF-6.4** exige un mapa de *Fallback* (OpenStreetMap) para desarrollo y contingencia, y Leaflet no incluye tipos por defecto, este archivo es necesario para evitar errores de compilación al usar plugins de dibujo o marcadores personalizados en el entorno estricto de CI/CD.
- **`/lib/validations/system-config.ts`** Esquema Zod estricto para validar el documento singleton `system_config` al leerlo. Vital para evitar caídas del sistema si un Admin edita mal manualmente los precios (`costPerMessage`) o las listas blancas de IP en la base de datos, garantizando "Fail-Safe" en runtime.
- **`/lib/security/cron-verify.ts`** Utilidad de verificación de tokens OIDC. Middleware específico para los *API Routes* de `/api/cron/*`. Asegura que las peticiones de mantenimiento (Backup, Sweeper) provengan legítimamente de *Google Cloud Scheduler* y no de un atacante externo intentando saturar recursos.
- **`/lib/utils/formula-injection-shield.ts`** Lógica de sanitización aislada para "CSV Injection". Detecta y neutraliza caracteres peligrosos (`=`, `+`, `-`, `@`) al inicio de campos de texto. Se separa de `csv-parser.ts` para ser reutilizada tanto en la **Importación** (Defensa) como en la **Exportación** (Protección de Admins).
- **`/lib/utils/vote-classifier.ts`** Función pura de lógica de negocio que determina el nivel de fidelización ("Simpatizante Digital" vs "Voto Fidelizado") basándose en `identityValidationStatus`, `kycStatus` y el historial de asistencia. Centraliza esta lógica para que los KPIs y los gráficos de `Embudo de Fidelización` sean consistentes.
- **`/lib/hooks/use-wake-lock.ts`** Hook que consume la API `navigator.wakeLock` para evitar que la pantalla se apague mientras el usuario está en la vista de "Mapa" o "Escaneo QR", vital para la operación continua bajo luz solar sin interrupciones de bloqueo de pantalla.
- **`/lib/hooks/use-form-draft.ts`** Hook especializado para guardar/restaurar el estado parcial del formulario de registro en `localStorage`. A diferencia de `use-form-persistence`, este hook maneja específicamente la lógica de limpieza tras un registro exitoso y la expiración de borradores viejos para no mostrar datos obsoletos.
- **`/lib/middleware/middleware-stack.ts`** Utilidad para componer múltiples funciones de middleware en una cadena secuencial. Permite ejecutar `RateLimit -> GeoBlock -> Auth -> LegalWall` de forma ordenada y aislada, facilitando el testing unitario de cada capa de seguridad (RF-8).
- **`/lib/middleware/request-context.ts`** Módulo para inyectar información contextual (GeoIP parseada, ID de correlación para logs) en los headers del request. Vital para que los logs de auditoría (RF-8.8) tengan trazabilidad completa desde el borde hasta la base de datos.
- **`/lib/config/navigation.ts`** Definición estática tipada de los menús laterales y bottom-bar. Mapea cada ruta (`/dashboard/users`, `/dashboard/territory`) con los roles permitidos (`['ADMIN', 'COORDINATOR']`). Esencial para que el componente `Sidebar.tsx` (RF-7.1) filtre opciones dinámicamente sin lógica condicional dispersa ("spaghetti code").
- **`/lib/utils/vitals.ts`** Utilidad para capturar métricas *Web Vitals* (LCP, FID, CLS) y enviarlas a un endpoint de análisis o Google Analytics. Crítico para auditar objetivamente el cumplimiento del **RNF-1 (Performance < 2s)** en dispositivos reales de campo, más allá de las pruebas sintéticas de Lighthouse.

- **`/lib/utils/print-helper.ts`** Funciones auxiliares para gestionar el evento `window.print()` y detectar cuándo el navegador ha terminado de generar el PDF (eventos `beforeprint` / `afterprint`). Necesario para el módulo de "Listados de Punteo" (RF-7.8), permitiendo limpiar estilos o mostrar avisos de seguridad al finalizar la impresión.
- **`/lib/utils/deep-link-manager.ts`** Gestor centralizado para parsear y construir *Deep Links* internos (`/dashboard/disputes/123`). Asegura que las notificaciones (RF-5.10) y los mensajes de WhatsApp siempre apunten a URLs válidas y manejen correctamente la redirección si el usuario no está logueado.
- **`/lib/hooks/use-orientation.ts`** Hook específico para **RF-6.6 (Restricción de Edición)**. Detecta cambios de orientación en tiempo real y fuerza el renderizado de la alerta "Gira tu dispositivo" en tabletas, asegurando que el editor de polígonos tenga suficiente espacio horizontal.
- **`/lib/utils/clipboard-fallback.ts`** Utilidad de respaldo para `use-clipboard.ts`. Implementa `document.execCommand('copy')` para navegadores móviles antiguos (Android WebViews dentro de apps de redes sociales) donde la API `navigator.clipboard` moderna puede fallar, vital para el **Flujo 2 (Compartir Enlace)**.
- **`/lib/constants/regex-colombia.ts`** Separación de `regex-patterns.ts`. Contiene validaciones específicas para la idiosincrasia de datos colombianos exigidos por la Registraduría: formato estricto de Cédulas nuevas vs antiguas y nomenclatura de direcciones (Cra, Cll, Av, Transv) para la normalización en **RF-1.11**.
- **`/lib/services/image-compression-worker.ts`** Web Worker dedicado para la compresión de imágenes (**RNF-5**). Mueve el procesamiento de `browser-image-compression` fuera del hilo principal para evitar que la UI se congele ("jank") mientras el usuario selecciona fotos de E-14 de alta resolución en gama baja.
- **`/lib/constants/colombia-holidays.ts`** Array de objetos con los días festivos de Colombia. Necesario para que la función de cálculo de SLAs financieros (`finance-ledger.ts`) determine correctamente las "Horas Hábiles" para la acreditación de saldo, excluyendo festivos y fines de semana.
- **`/lib/utils/export-sanitizer.ts`** Función pura separada que recibe un objeto `User` y aplica las reglas de "Modo Operativo" (Safe-Field). Elimina campos como `address` exacta y enmascara `phone` y `documentId` antes de que los datos pasen al generador de Excel. Garantiza que la lógica de limpieza sea centralizada y testearle.
- **`/lib/utils/screenshot-generator.ts`** Utilidad auxiliar para `opengraph-image.tsx` que compone visualmente la tarjeta de invitación con la foto del candidato y el nombre del líder usando `satori` o `ImageResponse` de Next.js, asegurando que la imagen generada cumpla con los límites de tamaño de WhatsApp.
- **`/lib/validations/event.ts`** Esquema Zod para la creación de eventos (**RF-1.13**). Valida que la fecha de inicio sea anterior a la de fin, que el radio del Geo-Fence sea positivo y que la ubicación tenga coordenadas válidas.
- **`/lib/utils/kml-parser.ts`** Utilidad para permitir la importación de zonas territoriales desde archivos KML/KMZ (Google Earth). Aunque no es explícito en el texto principal, es una necesidad tácita para la **Definición de Zonas (RF-6.6)** cuando las campañas ya tienen mapas previos, facilitando la migración.
- **`/lib/services/fraud-detection-engine.ts`** Separa la lógica heurística del **RF-1.10** (Velocity Checks, Volume Cap) del trigger de base de datos. Permite ejecutar simulaciones de fraude sobre datos históricos para ajustar los umbrales sin afectar la producción.
- **`/lib/hooks/use-camera-permission.ts`** Hook personalizado para gestionar el ciclo de vida de los permisos de cámara en navegadores móviles. Esencial para el **RF-1.13** (Eventos) y **RF-3.9** (Evidencia E-14), manejando los estados de "Permiso Denegado" con instrucciones de UX para el usuario.

---

4. Componentes UI (`/components`)

- **`/components/ui/*`** Componentes base (Shadcn/ui). Botones, Inputs, Modales, Toasts, Cards. Adaptados para alto contraste (RNF-4).
- **`/components/ui/feedback/ShakeWrapper.tsx`** Componente contenedor que implementa la animación de "agitación" (Shake Animation) requerida en RF-1.10. Se usa para envolver botones bloqueados cuando un usuario con *Soft Lock* intenta interactuar con ellos.
- **`/components/ui/visual/ConfettiCanvas.tsx`** Componente de canvas ligero para renderizar el efecto de confeti (RF-4.5). Debe estar optimizado para desmontarse del DOM inmediatamente después de la animación para no consumir GPU en móviles gama baja.
- **`/components/platform/AdminRequestTable.tsx`** Tabla para SUPER_ADMIN. Lista solicitudes de admin pendientes con acciones de aprobación/rechazo y visor de evidencia.
- **`/components/platform/TenantStats.tsx`** Widget de métricas globales (Total Campañas, SMS enviados, Usuarios activos) para el dashboard técnico.
- **`/components/campaigns/CampaignSetupWizard.tsx`** Stepper que guía el proceso "Day Zero": Creación de Político -> Configuración de Campaña.
- **`/components/campaigns/PoliticianProfileForm.tsx`** Formulario especializado para datos del candidato (Foto, Bio, Partido).
- **`/components/campaigns/CampaignCard.tsx`** Tarjeta visual para el catálogo de adhesión pública. Muestra logo, nombre y botón de unirse.
- **`/components/admin/WalletRechargeModal.tsx`** Modal crítico para la recarga de saldo (RF-5.6). Formulario estricto que solicita monto, referencia bancaria y confirmación de doble factor (si aplica) antes de invocar `financial-ops.ts`.
- **`/components/admin/ImportErrorsViewer.tsx`** Visualizador del reporte de errores post-importación masiva (RF-1.11). Muestra filas fallidas y razones (ej: "Cédula duplicada", "Formato inválido") permitiendo descargar un CSV de corrección.
- **`/components/shared/OfflineIndicator.tsx`** Indicador visual en header. Muestra estado de red y badge numérico de elementos en cola de sincronización (RF-3.5).
- **`/components/shared/RoleGuard.tsx`** HOC para renderizado condicional de elementos según permisos de usuario (RF-8.2).
- **`/components/shared/SoftLockBanner.tsx`** Banner informativo amarillo para usuarios en estado `NEEDS_VERIFICATION` (RF-1.10). Bloquea acciones avanzadas.
- **`/components/shared/PrintLayout.tsx`** Wrapper de estilos CSS `@media print` para reportes físicos limpios "Eco-Ink" (RF-7.8).
- **`/components/shared/InstallPrompt.tsx`** Componente "Bottom Sheet" específico para iOS que instruye cómo instalar la PWA (RF-3.1).
- **`/components/shared/LegalConsentModal.tsx`** Modal de aceptación de términos obligatorio ("Legal Wall") que bloquea la UI si la versión legal cambia (RF-3.8).
- **`/components/layout/Sidebar.tsx`** Barra de navegación principal con visibilidad progresiva de módulos según rol.
- **`/components/layout/CampaignSwitcher.tsx`** Selector de campañas activas con persistencia de selección (RF-7.1).
- **`/components/forms/RegisterForm.tsx`** Formulario de registro crítico. Integra validaciones, Age Gate y lógica Offline (RF-1.1).
- **`/components/forms/AgeGateInput.tsx`** Componente de fecha nacimiento con lógica de bloqueo UI inmediata para menores de 18 años (RF-1.1).
- **`/components/forms/AdminRequestForm.tsx`** Formulario para solicitud de rol Admin. Maneja validación y subida de archivos de soporte.
- **`/components/dashboard/views/CoordinatorView.tsx`** Vista estratégica para coordinadores. Widgets de integridad, financiero y conflictos (RF-7.1).
- **`/components/dashboard/views/LinkView.tsx`** Vista táctica para enlaces. Top Movers, Kit de Despliegue y alertas de inactividad (RF-7.1).
- **`/components/dashboard/views/FollowerView.tsx`** Vista básica para seguidores. Credencial digital y noticias (RF-7.1).
- **`/components/dashboard/views/AdminView.tsx`** Vista principal para el rol Admin. Orquestador de widgets administrativos y alertas de campaña.
- **`/components/dashboard/widgets/FinancialWidget.tsx`** Muestra saldo en USD y Burn Rate. Visible solo para Admin/Coordinador. Alertas de saldo bajo.
- **`/components/dashboard/widgets/ZoneIntegrityWidget.tsx`** Widget visual para Coordinadores/Admin que muestra las alertas de **Conflictos de Topología (RF-6.6)** (zonas superpuestas o disputadas) y accesos directos al "Panel de Resolución". Separado del mapa para mayor visibilidad en el Dashboard.
- **`/components/dashboard/widgets/GrowthVelocityWidget.tsx`** Widget visual para la "Velocidad de Crecimiento". Compara registros de hoy vs. ayer, calcula el porcentaje de variación y muestra la flecha de tendencia (verde/roja) requerida en la vista táctica.
- **`/components/dashboard/widgets/SurveyResultsCard.tsx`** Visualiza los resultados de las Encuestas (RF-5.8). Renderiza gráficos de barras simples para preguntas de selección única o múltiple procesadas desde la colección `campaign_responses`.
- **`/components/dashboard/widgets/MotivationGauge.tsx`** Componente visual tipo "Velocímetro" para la Meta Personal (RF-7.1.A/B). Cambia de color

(Rojo/Amarillo/Verde) dinámicamente según el porcentaje de cumplimiento (1x10) para incentivar al Multiplicador.

- **`/components/maps/TerritoryMap.tsx`** Componente de mapa interactivo. Soporta dibujo de zonas, visualización de conflictos y marcadores.
- **`/components/maps/OfflineWidget.tsx`** UI de fallback cuando no carga el mapa (RF-6.5). Permite entrada manual de dirección para geocodificación diferida.
- **`/components/maps/HeatMap.tsx`** Visualización de densidad de votos y distribución territorial (Fase 2).
- **`/components/maps/ZoneEditor.tsx`** Herramientas de edición de polígonos para Admin/Coord. Implementa "Snapshotting" en cambios.
- **`/components/maps/ConflictModal.tsx`** Modal para resolución de disputas territoriales (RF-6.6). Opciones: "Ceder", "Reclamar", "Editar".
- **`/components/maps/GeoFixerList.tsx`** Componente para la lista "The Fixer", permitiendo validación rápida de direcciones fallidas.
- **`/components/maps/LeafletMapWrapper.tsx`** Componente de mapa basado en OpenStreetMap/Leaflet. Requerido obligatoriamente por **RF-6.4 (Visualización Híbrida)** para entornos de desarrollo (ahorro de costos) y como **Fallback Automático (RF-3.10)** en producción si Google Maps falla o entra en "Modo Lite".
- **`/components/maps/ZoneStatusLegend.tsx`** Leyenda flotante sobre el mapa que explica la codificación visual de las zonas (Activa, Vacante, Disputada) y permite filtrar capas para reducir ruido visual en móviles (RF-6.6).
- **`/components/maps/MapFallbackAlert.tsx`** Componente de alerta inteligente (RF-6.5/RF-8.4) que aparece cuando el mapa no carga (Modo Lite o Error de Cuota). Instruye al usuario sobre cómo operar en modo "Solo Texto/Manual".
- **`/components/maps/DrawingManagerWrapper.tsx`** Wrapper específico para la librería `google.maps.drawing`. Encapsula la lógica de eventos `overlaycomplete`, ejecuta la "Auto-Simplificación" de vértices y previene la creación de "Donuts" antes de pasar los datos al `ZoneEditor` (RF-6.6).
- **`/components/charts/GrowthChart.tsx`** Gráfico de línea para métricas de crecimiento. Optimizado para móvil (RNF-1) usando Recharts.
- **`/components/communications/TemplateEditor.tsx`** Editor visual para plantillas HSM. Integra cotizador en tiempo real y validación de variables (RF-5.3).
- **`/components/communications/ConversionFunnel.tsx`** Visualización gráfica del embudo de mensajería (Enviado -> Entregado -> Leído -> Respondido). Consume datos pre-agregados de `responseDistribution` para renderizar el gráfico sin consultas costosas.
- **`/components/communications/CostEstimator.tsx`** Widget de cotización en tiempo real para el Editor de Plantillas. Calcula segmentos GSM-7/UCS-2, detecta caracteres especiales costosos y muestra alertas de presupuesto antes de guardar la plantilla.
- **`/components/reports/EcoInkTable.tsx`** Tabla optimizada para impresión en blanco y negro (RF-7.8). Elimina fondos y ajusta tipografía.
- **`/components/reports/PrintableVoterRow.tsx`** Componente atómico optimizado para "Eco-Ink". Renderiza una fila de la planilla de punteo con estilos CSS específicos (`@media print`) para ocultar fondos, forzar texto negro y garantizar legibilidad en papel A4.
- **`/components/security/DeviceSessionList.tsx`** Panel de "Dispositivos Activos" para la vista de Perfil. Lista sesiones abiertas, muestra IP/UserAgent y permite revocar tokens remotos (Cerrar sesión en otros dispositivos) llamando a `/api/auth/sessions/revoke`.
- **`/components/forms/VotingStationSelector.tsx`** Implementa el **Protocolo de Excepción (RF-1.1)**. Encapsula la lógica híbrida que inicia como un *Dropdown* de búsqueda local (indexada en cliente) y transmuta a un *Input de Texto* libre si se selecciona "Otro / No aparece", manejando la alerta amarilla de "Puesto Manual".
- **`/components/communications/SurveyBuilder.tsx`** Módulo para **RF-5.3.D (Configuración de Encuestas)**. Interfaz dentro del editor de plantillas que permite definir preguntas, opciones de respuesta y reglas de cardinalidad (Selección Única/Múltiple) para la recolección de datos estructurados vía WhatsApp.
- **`/components/onboarding/WelcomeKit.tsx`** Implementa la **Experiencia de Aterrizaje (RF-3.7)**. Gestiona el Modal de Bienvenida (Overlay) y la reproducción automática del video del candidato, asegurando que se muestre una única vez tras el registro exitoso (FTUE).
- **`/components/security/IdentityAnchorModal.tsx`** Interfaz para el **Protocolo de Recuperación de Identidad (RF-1.7)**. Permite al Coordinador registrar formalmente la "Validación Documental" (verificación física de la cédula) antes de escalar la solicitud de cambio de número al Admin.
- **`/components/layout/HierarchyBreadcrumb.tsx`** Componente de navegación contextual necesario para el **Data Scoping (RF-7.2)**. Muestra la ruta jerárquica actual (ej: *Admin > Zona Norte > Carlos Pérez*) y permite la navegación vertical rápida en vistas de auditoría.
- **`/components/maps/HeatMapLayer.tsx`** Componente aislado para **Fase 2 (RF-7.1.D)**. Encapsula la lógica de visualización de densidad. Se crea separado para facilitar su ocultamiento/feature-flagging en la versión MVP sin ensuciar el componente principal `TerritoryMap`.
- **`/components/emails/MagicCodeEmail.tsx`** Plantilla de correo (React Email) para el envío de códigos de acceso y recuperación (RF-1.7 / RF-8.1). Diseñada para ser simple, con texto grande y branding minimalista.
- **`/components/emails/FinancialAlertEmail.tsx`** Plantilla de correo de alta prioridad para **RF-5.6 (Monitor de Salud Financiera)**. Muestra el saldo actual, el *burn rate* y un botón de llamada a la acción (CTA) directo para recargar saldo.
- **`/components/emails/SecurityAlertEmail.tsx`** Plantilla para **RF-4.3 (Security Broadcast)** y **RF-8.8 (Scraping Alert)**. Informa al usuario sobre accesos sospechosos o descargas masivas de datos, incluyendo IP y ubicación aproximada.
- **`/components/ui/data-table/DataTable.tsx`** Componente base reutilizable para tablas complejas (Dashboard de Usuarios, Logs). Integra paginación de servidor, ordenamiento y filtros visuales, desacoplando la lógica de presentación de Shadcn/UI de la lógica de negocio.
- **`/components/ui/data-table/DataTablePagination.tsx`** Componente específico para manejar la paginación por cursores de Firestore (RF-7.7), permitiendo navegación "Siguiente/Anterior" eficiente sin cargar toda la colección.
- **`/components/maps/DrawingControls.tsx`** Controles flotantes separados del mapa principal para la gestión de polígonos (RF-6.6). Botones para "Iniciar Dibujo", "Borrar Selección", "Deshacer" y "Guardar Zona", mejorando la usabilidad en móviles (RNF-4) al evitar menús contextuales complejos.
- **`/components/layout/Header.tsx`** Contenedor superior omnipresente. Aloja el `CampaignSwitcher` (RF-7.1), el disparador del menú móvil (Hamburguesa) y sirve de anfitrión para los componentes críticos `GlobalSearch` y `NotificationCenter` que no caben en la Sidebar.
- **`/components/shared/GlobalSearch.tsx`** Implementación del **Motor de Búsqueda Omnipresente (RF-7.6)**. Input en la cabecera que ejecuta búsquedas por prefijo (Cédula, Nombre, QR) consumiendo los índices compuestos de Firestore. Maneja el *debouncing* y la navegación rápida al perfil encontrado.
- **`/components/shared/NotificationCenter.tsx`** Interfaz para la **Bandeja de Entrada In-App (RF-5.10)**. Desplegable (Popover) que lista alertas de sistema, logros y tareas pendientes. Gestiona el estado de "No Leído" y la redirección mediante `deepLinks`.
- **`/components/dashboard/widgets/StatCard.tsx`** Componente atómico para renderizar los **KPIs Transversales (RF-7.1.A)**. Estandariza la visualización de métricas numéricas con indicadores de tendencia (flechas verdes/rojas) y micro-gráficos (Sparklines) para "Velocidad de Crecimiento".
- **`/components/ui/feedback/NetworkStatusToast.tsx`** Componente especializado para el feedback de **Modo Vereda (RF-3.5)**. A diferencia de un toast normal, este es persistente y reactivo, mostrando el estado "Guardando localmente..." o "Conexión Requerida" según la lógica del hook `use-offline`.
- **`/components/forms/PhoneInput.tsx`** Input especializado para **RF-1.1 y RF-1.11**. Fuerza visualmente el prefijo `+57`, impide la entrada de caracteres no numéricos y formatea el número en tiempo real, previniendo errores de validación antes de llegar a `phone-normalizer.ts`.
- **`/components/maps/ZoneConflictResolver.tsx`** Implementación de la **UI de Resolución de Disputas (RF-6.6.C)**. Panel flotante que aparece cuando una zona tiene estado `DISPUTED`, ofreciendo los botones "Ceder", "Reclamar Todo" o "Editar Vértices" y visualizando el área de superposición en m².
- **`/components/communications/WhatsAppPreview.tsx`** Componente de visualización para **RF-5.3 (Editor Visual)**. Renderiza una simulación pixel-perfect de cómo se verá el mensaje en WhatsApp (burbujas, botones, header multimedia) basándose en el estado actual del `TemplateEditor`. Es vital para que el usuario apruebe el diseño antes de enviarlo a Meta.
- **`/components/dashboard/widgets/DisputeComparisonCard.tsx`** Componente para la **Resolución de Disputas (RF-1.12)**. Muestra una tarjeta comparativa "Cara a Cara" entre el Líder Actual y el Nuevo Líder (métricas, antigüedad), facilitando la decisión del Juez Jerárquico (Coordinador/Admin) de aprobar o rechazar el traslado.
- **`/components/maps/MapSkeleton.tsx`** Estado de carga visual para **RNF-1 (Performance)**. Se muestra mientras el script de Google Maps carga. A diferencia del `OfflineWidget` (que es para error), este es un placeholder visual transitorio que evita el "layout shift" acumulativo en el Dashboard.
- **`/components/shared/ErrorBoundary.tsx`** Envoltorio de React esencial para cumplir con la **UX de Degradación Elegante (RF-6.5)**. Captura errores de renderizado en el componente de Mapas (por fallos de API o Cuota) y muestra la *Fallback UI* ("Servicio de mapas no disponible") sin romper toda la aplicación.
- **`/components/shared/QrCodeCard.tsx`** Componente visual que renderiza el código QR (SVG) del Multiplicador. Incluye las opciones de "Descargar PNG" y

"Compartir Enlace" descritas en el **Flujo 2**. Es consumido tanto por el `LinkView` (Dashboard) como por el estado vacío de activación ("Tu Herramienta").

- **`/components/maps/HeatMapLegend.tsx`** Leyenda flotante explicativa para el Mapa. Vital para decodificar la densidad visual (Colores cálidos vs fríos) y diferenciar visualmente las zonas con estado `DISPUTED` (rayas) de las `VACANT` (gris). Mejora la accesibilidad cognitiva de la herramienta geográfica.
- **`/components/security/CriticalActionGate.tsx`** **RF-5.6 / RF-8.6:** Componente "Guard" que implementa el protocolo de **Segregación de Funciones (SoD)**. Envuelve botones de alto riesgo (Recargas de Saldo, Restauración de Backup) solicitando re-autenticación (2FA) o validación contra la lista `financialAdmins` antes de permitir el evento `onClick`.
- **`/components/maps/ClusterMarker.tsx`** **RF-6.4 / RNF-1:** Componente para agrupar marcadores en mapas densos. Necesario para el rendimiento en móviles cuando se visualizan "Votos por Puesto" (RF-7.1.D) o miles de "Votos Foráneos", evitando la saturación del DOM que bloquearía la interfaz táctil.
- **`/components/onboarding/ActivationHeroCards.tsx`** **RF-3.7-C (Estrategia de Estado Vacío):** Implementación exacta de las tres tarjetas de acción grandes ("Tu Herramienta", "Tu Misión", "Tu Meta") que reemplazan los dashboards analíticos cuando el Multiplicador tiene `recruitedCount == 0`.
- **`/components/onboarding/GuidedTourCarousel.tsx`** **RF-3.7-B (Micro-Instrucciones):** Componente de carrusel modal que se muestra una sola vez tras el registro. Explica en 3 pasos cómo usar el QR y compartir el enlace, fundamental para la activación inicial del usuario.
- **`/components/maps/ZoneOverrideToggle.tsx`** Control de interfaz para **RF-93 (Asignación Manual)**. Switch o botón en el perfil del usuario (vista Coordinador) que habilita la selección forzada de una zona, anulando el cálculo automático de `Point-in-Polygon`.
- **`/components/dashboard/widgets/LeaderConversionChart.tsx`** Widget para la métrica **"Tasa de Conversión de Líderes"** (RF-99). Gráfico especializado que visualiza la proporción de Multiplicadores vs. Seguidores Totales, distinto del gráfico de distribución general.
- **`/components/security/SessionKillSwitch.tsx`** Botón crítico para el **RF-8.9 (Security Kill-Switch)** administrativo. Permite al Admin forzar el cierre de sesión de un usuario específico desde su panel de detalle, invocando la Server Action `identity-ops.ts`.
- **`/components/charts/RoleDistributionChart.tsx`** Implementación del Gráfico de Dona requerido en **RF-7.1.C**. Visualiza la proporción de Multiplicadores vs. Seguidores vs. Enlaces usando `Recharts`, permitiendo a los coordinadores evaluar el equilibrio de su estructura jerárquica.
- **`/components/ui/toaster.tsx`** Componente orquestador de notificaciones (Toast Provider) estándar de `shadcn/ui`. Es la dependencia faltante para renderizar las alertas de `NetworkStatusToast` (RF-3.5) y `GeoFencingWarning` (RF-93) en la capa superior de la aplicación.
- **`/components/admin/IdentitySwapForm.tsx`** **RF-1.7 / Flujo 6B:** Formulario exclusivo para el Rol ADMIN. Permite ingresar el UID del usuario comprometido y el nuevo número telefónico. Exige la confirmación de haber recibido la evidencia física (Cédula) por parte del Coordinador antes de habilitar el botón de ejecución.
- **`/components/platform/VotingPlaceForm.tsx`** **RF-1.14:** Formulario para la creación manual de Puestos de Votación en el Dashboard (más allá de la carga masiva). Integra la API de Geocoding para validar la dirección física y bloquea la edición si el campo `dayD_config` está congelado (Scope Freeze).
- **`/components/communications/TemplateStatusBadge.tsx`** **RF-5.3 / RF-5.7:** Componente visual para el listado de plantillas. Muestra estados complejos (`APPROVED`, `REJECTED`, `PAUSED`) sincronizados con Meta, y renderiza un *tooltip* con la razón del rechazo si existe, vital para la gestión de campañas.
- **`/components/emails/InvitationEmail.tsx`** **RF-5.2:** Plantilla de correo React Email para el "Enlace de Invitación Único". Se envía automáticamente tras el registro o por acción manual. Debe contener el branding de la campaña y el link con token de atribución parametrizado.
- **`/components/visual/OrgChart.tsx`** **RF-1.15:** Componente de visualización de árbol para la herramienta de Staffing. Permite al Admin ver la estructura jerárquica (Coordinador -> Enlace -> Multiplicador) de forma gráfica para facilitar decisiones de ascenso o reasignación masiva.
- **`/components/feedback/MaintenanceScreen.tsx`** **RF-8.7 (Alertas de Anomalía):** Pantalla de bloqueo total que se activa cuando el sistema entra en "Modo Bajo Ataque". Oculta el login estándar y muestra solo mensajes de servicio, desactivando interacciones costosas (SMS/Maps).
- **`/components/providers/ClientProviders.tsx`** "Client Component" maestro que agrupa todos los proveedores (`SessionProvider`, `ThemeProvider`, `CampaignContext`, `TooltipProvider`). Envuelve a `children` en `layout.tsx` para habilitar el uso de hooks en toda la app.
- **`/components/providers/QueryProvider.tsx`** Configuración de `TanStack Query` (o SWR global). Necesario para manejar la **Estrategia de Caché y Revalidación (RF-7.9)** en el cliente, configurando los tiempos de deduplicación (5 min) y reintentos.
- **`/components/ui/file-uploader.tsx`** Componente reutilizable con *Drag & Drop*. Integra validación de tipo MIME y tamaño cliente-lado (RNF-5), y compresión de imagen antes de la subida. Usado en Registro (Fotos), Admin (CSV) y Builder (Multimedia).
- **`/components/shared/ConnectionStatus.tsx`** Componente visual que detecta y muestra la calidad de la red (3G/4G/Offline). Se integra con el `use-network-quality.ts` para sugerir activar el "Modo Lite" (RF-3.10) si la latencia es alta.
- **`/components/security/ReauthModal.tsx`** **RF-5.6 / RF-8.6:** Componente de interfaz para la "Segregación de Funciones" (SoD). Es invocado por el `CriticalActionGate` cuando un Admin intenta una acción destructiva (Recarga de Saldo, Restauración Backup). Solicita la contraseña actual o un segundo factor (si está habilitado) antes de permitir la ejecución de la Server Action.
- **`/components/dashboard/widgets/RecentActivityFeed.tsx`** **RF-7.1 / RF-8.8:** Widget para la vista de Admin/Coordinador que muestra un "Live Feed" de acciones operativas (Nuevos registros, Cambios de zona, Alertas). Consume datos de `audit_logs` en tiempo real (con debounce) para dar sensación de "Pulso de Campaña" sin sobrecargar lecturas.
- **`/components/maps/MapControls/LayerSwitcher.tsx`** **RF-6.4 / RF-6.6:** Control flotante en el mapa para alternar entre capas visuales: "Satélite" (para verificar techos/barrios en edición de zonas) y "Plano" (para visualización de datos). Vital para que los coordinadores validen la topología física del territorio dibujado.
- **`/components/feedback/StorageQuotaModal.tsx`** Componente visual para el **RF-3.5.C (Manejo de Excepción)**. Un modal crítico que aparece cuando `QuotaExceededError` es capturado. Debe instruir al usuario didácticamente sobre cómo liberar espacio en Android/iOS (ej: "Borrar videos de WhatsApp") para poder continuar trabajando offline.
- **`/components/maps/DrawingTools/UndoRedoControls.tsx`** Controles específicos para el **Módulo de Mapas (RF-6.6)**. Dado que la edición de polígonos en móviles es propensa a errores ("Fat-Finger"), se requiere una interfaz flotante dedicada para deshacer el último vértice sin borrar todo el polígono, interactuando con el `map-editor-store.ts`.
- **`/components/dashboard/widgets/QuickAuditCard.tsx`** Widget para la vista de Coordinador (**RF-7.1 / RF-1.10**). Muestra un resumen compacto de "Líderes en Revisión" (fraude potencial) con un botón de acción rápida para iniciar la auditoría telefónica, cumpliendo con la "UI Especializada por Rol" (Pág. 100).
- **`/components/ui/async-select.tsx`** Componente genérico de selección con búsqueda asíncrona y *debouncing*. Es necesario para reutilizar la lógica de búsqueda en el formulario de "Disputas" (Buscar Nuevo Líder) y en herramientas administrativas, más allá del selector de puestos de votación.
- **`/components/dashboard/widgets/HeatMapFilters.tsx`** Controles flotantes sobre el Mapa de Calor (Fase 2) para filtrar la densidad por "Intención de Voto" vs "Ubicación Real". Permite al usuario conmutar las capas de datos descritas en el diseño de pantalla 4.
- **`/components/maps/LocationPicker.tsx`** Componente unificado que encapsula la lógica de "Autocomplete + Drag & Drop Pin + Fallback Manual". Reutilizable tanto en el Registro (Flujo 1) como en la herramienta "The Fixer" (Corrección Admin).
- **`/components/dashboard/widgets/WhatsappHealthWidget.tsx`** Widget visual para el Admin que muestra el límite diario de envíos a usuarios importados (ej: "250/500 enviados hoy") y el estado de salud de la línea telefónica.
- **`/components/admin/UserAuditTimeline.tsx`** Componente de visualización cronológica de los `audit_logs` de un usuario específico. Permite al Admin ver la historia clínica del registro (Quién lo creó, cambios de zona, validaciones).
- **`/components/dashboard/disputes/DisputeResolutionPanel.tsx`** Panel lateral complejo para la resolución de conflictos topológicos. Muestra el área de superposición, mapas comparativos mini y los botones de acción booleana (Ceder/Reclamar).
- **`/components/shared/SensitiveDataShield.tsx`** Componente envoltorio que difumina (blur) visualmente datos sensibles (Cédula, Teléfono) en el Dashboard hasta que el usuario pasa el mouse o hace clic ("Reveal on Hover"), protegiendo contra "Shoulder Surfing" en campo.
- **`/components/communications/TemplateVariableInput.tsx`** Componente especializado para insertar variables dinámicas (`{{1}}`, `{{2}}`) en el cuerpo del mensaje, validando que correspondan a columnas existentes en la base de datos de usuarios.
- **`/components/providers/ThemeInjector.tsx`** Componente que se monta en el `RootLayout`. Lee la configuración de la campaña (Server Component) e inyecta las variables CSS de color (`--primary-color`) en el `:root` del documento antes de la hidratación, evitando el "Flash of Unstyled Content" (FOUC).
- **`/components/maps/MapControls/ZoomControls.tsx`** Controles de zoom explícitos (+ / -). Necesarios para cumplir con las áreas de toque mínimas

(48x48px) en móviles, ya que el gesto de "pellizcar" (pinch-to-zoom) puede ser difícil en condiciones de campo o con una sola mano.

- **`/components/feedback/NetworkQualityIndicator.tsx`** Componente visual pequeño (tipo semáforo) que muestra la calidad de la conexión (RTT). Informa al usuario por qué el mapa se ha desactivado automáticamente ("Conexión lenta detectada") antes de que intente recargar la página.
- **`/components/emails/DisputeAlertEmail.tsx`** Plantilla de notificación prioritaria para Coordinadores/Admin. Informa que se ha creado una solicitud de traslado ("Divorcio") que requiere arbitraje inmediato, incluyendo enlaces directos a la consola de resolución.
- **`/components/emails/WelcomeCampaignEmail.tsx`** Plantilla transaccional que se envía tras el registro exitoso. Contiene el "Welcome Kit" digital, enlaces a redes sociales y el código QR del usuario adjunto o renderizado, reforzando la identidad de pertenencia.
- **`/components/ui/secure-image.tsx`** Componente `Image` wrapper que maneja automáticamente la obtención y refresco de *Signed URLs* de Google Cloud Storage. Necesario porque las evidencias E-14 son privadas y sus URLs expiran; el componente estándar de Next.js no maneja esta rotación de seguridad.
- **`/components/admin/ImportDeclarationModal.tsx`** Modal con checkbox de declaración juramentada ("Declaro bajo gravedad de juramento..."). Debe bloquear el botón de procesamiento hasta que el Admin acepte explícitamente la responsabilidad legal del origen de los datos.
- **`/components/maps/MapControls/VertexEditor.tsx`** Componente especializado que aparece solo cuando el estado de una zona es `DISPUTED`. Permite el ajuste fino de nodos individuales del polígono para resolver superposiciones milimétricas, separado de la lógica general de dibujo.
- **`/components/feedback/MaintenanceGuard.tsx`** Componente de alto nivel que envuelve la aplicación. Si el flag `maintenanceMode` en `SystemConfig` es true y el usuario no es Super Admin, renderiza forzosamente la `MaintenanceScreen`, bloqueando cualquier otra interacción.
- **`/components/shared/FeatureFlagGuard.tsx`** Componente contenedor de alto orden (HOC). Consume `SystemConfigContext` para ocultar o renderizar componentes condicionalmente según las banderas (ej: ocultar módulo "Día D" en v1.0). Evita lógica condicional repetitiva en las vistas.
- **`/components/maps/MapControls/GeolocateButton.tsx`** Botón flotante dedicado para "Centrar en mi ubicación". Crítico para la usabilidad móvil en trabajo de campo, separado de los controles de zoom para cumplir con áreas de toque mínimas (48x48px).
- **`/components/dashboard/widgets/OrphanedZonesAlert.tsx`** Widget exclusivo para el Dashboard ADMIN. Visualiza alertas críticas sobre zonas marcadas como `VACANT` o en cuarentena tras la eliminación de un Coordinador, permitiendo acción rápida de reasignación.
- **`/components/forms/EvidenceCaptureStub.tsx`** Componente "Placeholder" estructural para la carga de E-14. En v1.0, renderiza un estado vacío o mensaje de "Módulo no activo", asegurando que el esquema de base de datos y la arquitectura de componentes soporten la futura implementación sin refactorización mayor.
- **`/components/shared/IosInstallHint.tsx`** Componente visual específico ("Tooltip Animado" o "Bottom Sheet") que solo se monta si la librería `UAParser.js` detecta iOS Safari. Renderiza las instrucciones gráficas exactas descritas en la **Pantalla 1B** del documento (Flecha señalando el botón "Compartir" y el icono "+"), vital para la adopción en iPhone donde no existe `beforeinstallprompt`.
- **`/components/dashboard/widgets/ScopeComplianceCard.tsx`** Widget de alerta para el usuario que visualiza su estado de "Congruencia Geográfica". Si el usuario se muda a una dirección fuera del alcance de la campaña (detectado por `scope-compliance.ts`), este componente explica por qué su membresía pasará a estado `ARCHIVED` y ofrece acciones correctivas (ej: "Solicitar traslado a campaña local").
- **`/components/dashboard/widgets/TopMoversWidget.tsx`** Widget específico para la vista táctica que renderiza el ranking en tiempo real de los 5 mejores Multiplicadores. Implementa lógica de ordenamiento local para evitar lecturas excesivas y conecta con el hook de navegación para "gestión motivacional" inmediata (Llamar/WhatsApp).
- **`/components/dashboard/widgets/SilentZonesMapWidget.tsx`** Componente de mapa simplificado (solo lectura) que filtra y resalta exclusivamente los polígonos territoriales con 0 registros en los últimos 7 días ("Zonas Silenciosas"). Vital para la toma de decisiones estratégicas sin el ruido visual del mapa operativo completo.
- **`/components/dashboard/widgets/PromotionActionCard.tsx`** Componente "Hero" condicional para el Dashboard de Seguidor. Contiene el botón destacado "¿QUIERES SER LÍDER?" que dispara la Server Action `user-promotion.ts`. Maneja el estado de carga y la transición visual inmediata al Dashboard de Multiplicador tras el éxito.
- **`/components/maps/GhostPolygonLayer.tsx`** Capa visual para el componente `TerritoryMap`. Su función es renderizar la geometría "anterior" (obtenida de `audit_logs`) en línea punteada gris sobre la geometría actual, permitiendo al Admin visualizar visualmente el alcance de una modificación territorial disputada o sospechosa.
- **`/components/security/SessionExpiryModal.tsx`** Modal de bloqueo crítico que se activa cuando el `refreshToken` caduca o es revocado remotamente. A diferencia del login estándar, este componente **no limpia el estado local** (IndexedDB), permitiendo al usuario re-autenticarse sin perder los formularios offline pendientes de envío.
- **`/components/admin/DataScrubbingConfirmation.tsx`** Modal de alta seguridad para confirmar la eliminación de usuarios. Exige escribir una frase de control ("CONFIRMAR BORRADO") y explica visualmente que la acción anonimizará los datos personales pero mantendrá la integridad estructural y financiera (Ledger Inmutable).
- **`/components/ui/button.tsx`** Componente base personalizado. **Debe incluir lógica de cálculo de contraste (RF-3.7)** para invertir automáticamente el color del texto (Blanco/Negro) basándose en la luminancia de la variable CSS `--primary-color` inyectada, garantizando cumplimiento WCAG AA dinámico.
- **`/components/ui/skeleton.tsx`** Primitiva visual para estados de carga. Necesaria para construir el `MapSkeleton` y las tarjetas de carga del Dashboard, reduciendo la percepción de latencia (RNF-1).
- **`/components/ui/alert.tsx`** Componente primitivo para mensajes de estado. Base para construir `SoftLockBanner`, `GeoFencingWarning` y `MaintenanceGuard` con variantes semánticas (Destructive, Warning, Info).
- **`/components/emails/components/EmailHeader.tsx`** Componente reutilizable para encabezados de correo. Renderiza el logo de la campaña dinámicamente (desde `political_campaigns.branding`), asegurando consistencia visual en todas las notificaciones transaccionales.
- **`/components/emails/components/EmailFooter.tsx`** Pie de página legal estandarizado. Incluye la dirección física de la sede, enlaces de desuscripción y el aviso de confidencialidad requerido por auditoría (RF-4.3).
- **`/components/maps/PolygonValidatorUI.tsx`** Componente visual flotante que se activa durante el dibujo. Muestra en tiempo real si el polígono es válido (Sólido) o inválido (Donut/Auto-intersección) antes de intentar guardar, evitando el "Toast Rojo" del servidor y mejorando la UX en edición.
- **`/components/maps/BoomerangAlert.tsx`** Notificación visual específica (Toast/Modal) que informa al usuario y al líder cuando un registro previamente archivado ("Voto Perdido") regresa a la zona y reactiva su antigüedad. Diferencia visualmente este evento de un "Nuevo Registro".
- **`/components/admin/audit/LogDiffViewer.tsx`** Componente especializado para renderizar el "Antes y Después" de un cambio territorial o de perfil. Muestra visualmente las diferencias en JSON o en mini-mapas (Capa Fantasma vs. Actual) dentro del detalle del log.
- **`/components/platform/SystemConfigForm.tsx`** Formulario maestro para editar el documento singleton `system_config`. Permite al SUPER_ADMIN ajustar tarifas (`costPerMessage`), alternar `maintenanceMode` y gestionar la lista blanca de IPs (`hqWhitelistedIps`) sin tocar la base de datos directamente.
- **`/components/admin/import/CsvPreviewTable.tsx`** Tabla de previsualización que renderiza las primeras 5 filas del archivo cargado *antes* de enviarlo al servidor. Permite al usuario verificar que las columnas (Cédula, Teléfono) se mapearon correctamente según el esquema estricto.
- **`/components/communications/AudienceSelector.tsx`** Componente lógico reutilizable que permite construir queries complejas (ej: "Mujeres en Zona Norte con Rol Líder"). Se usa tanto en el Wizard de Campañas como en la exportación de datos.
- **`/components/communications/VariableMappingInput.tsx`** Input inteligente para el Editor de Plantillas. Permite mapear las variables `{{1}}` a campos de la base de datos (ej: `profile.fullName`) y valida que el campo exista para evitar mensajes rotos ("Hola undefined").
- **`/components/security/RecoveryCodeInput.tsx`** Input especializado para el "Magic Code" de recuperación. Implementa auto-formato (XXX-XXX) y prevención de pegado de caracteres inválidos para mejorar la UX en situaciones de estrés (pérdida de acceso).
- **`/components/shared/NotificationPermissionRequest.tsx`** Componente visual (Toast/Modal no intrusivo) que solicita permiso al navegador para Web Push. Gestiona la lógica de "Pedir más tarde" si el usuario lo rechaza inicialmente, evitando el bloqueo permanente del navegador.
- **`/components/maps/UserMarker.tsx`** Componente atómico para renderizar el pin de un usuario individual en el mapa. Encapsula la lógica visual para diferenciar roles (Color del borde) y estado de verificación (Icono de Check), optimizando el renderizado masivo al ser más ligero que el marcador genérico.
- **`/components/dashboard/widgets/BurnRateChart.tsx`** Visualización gráfica específica para el Admin. Muestra la tendencia de gasto diario vs.

presupuesto restante, permitiendo predecir visualmente la fecha de "Wallet Empty" (agotamiento de fondos) antes de que salte la alerta crítica.

- **`/components/ui/visual/Sparkline.tsx`** Micro-gráfico de línea sin ejes ni leyendas, utilizado dentro de las `StatCard` para mostrar la tendencia visual ("Velocidad de Crecimiento") de los últimos 7 días de forma compacta en móviles.
- **`/components/maps/MapControls/SatelliteToggle.tsx`** Control independiente para conmutar a vista Satelital. Aunque existe un `LayerSwitcher`, este botón específico es vital para la fase de "Dibujo de Zonas" (RF-6.6), permitiendo al Coordinador verificar visualmente los techos de las casas y fronteras físicas (ríos, calles) con un solo toque.
- **`/components/admin/audit/GeoDiffMap.tsx`** Componente especializado para el `LogExplorer`. Renderiza dos capas de polígonos superpuestas (Estado Anterior en gris punteado vs. Estado Nuevo en color sólido) dentro de la tarjeta de detalle del log, permitiendo al Admin auditar visualmente cambios territoriales sospechosos ("Gerrymandering").
- **`/components/shared/OfflineBanner.tsx`** Componente visual distinto al `NetworkStatusToast`. Este es un banner persistente (no descartable) que aparece en la parte superior de formularios críticos (como el de E-14 o Registro) cuando se detecta desconexión, advirtiendo explícitamente: "Modo Vereda: Los datos se guardarán en este dispositivo".
- **`/components/ui/date-range-picker.tsx`** Componente de calendario interactivo para selección de rangos de fechas. Dependencia visual obligatoria para el **Visor de Auditoría (RF-8.8)** y los reportes de **Crecimiento (RF-7.1)**, permitiendo al Admin filtrar logs o métricas ("Últimos 7 días", "Mes pasado") de manera intuitiva. No incluido en los componentes base de `ui/*`.
- **`/components/communications/MediaAssetPicker.tsx`** Componente visual para seleccionar imágenes/videos desde la librería de campaña dentro del `TemplateEditor`. Incluye previsualización y validación de relación de aspecto requerida por WhatsApp.
- **`/components/platform/DivipolBrowser.tsx`** Componente de árbol o lista jerárquica optimizada para explorar la estructura nacional de votación (12k+ puestos). Implementa carga diferida (Lazy Loading) por departamento para no saturar el navegador.
- **`/components/dashboard/widgets/ActivityHeatmapCalendar.tsx`** Visualización tipo "GitHub Contributions" para el perfil del usuario. Muestra la intensidad de la actividad (registros, asistencias) en el tiempo, clave para evaluar la constancia de un líder más allá del total acumulado.
- **`/components/admin/audit/WalletTransactionTable.tsx`** Tabla especializada para el libro mayor financiero. Diferencia visualmente créditos (Recargas) de débitos (Campañas), muestra el saldo resultante por fila y permite filtrar por referencia bancaria o ID de campaña.
- **`/components/users/VerificationActionModal.tsx`** Modal específico para que el Coordinador resuelva una alerta de fraude (`SUSPICIOUS_VELOCITY`). Obliga a seleccionar un motivo de validación ("Llamada realizada", "Conocido") antes de desbloquear al usuario, generando el log de auditoría correspondiente.
- **`/components/layout/MobileNavBar.tsx`** Componente visual del menú inferior persistente. Gestiona el estado activo de la ruta y oculta automáticamente la barra cuando se abre el teclado virtual para no reducir el área visible del formulario de registro en pantallas pequeñas.
- **`/components/ui/cards/AdaptiveUserCard.tsx`** Implementación de "Tarjetas en vez de tablas" para listados móviles (**RF-7.5**). Transforma la fila de datos del usuario en un componente vertical optimizado para *scroll* táctil, mostrando acciones rápidas (Llamar, Auditar) con botones grandes.
- **`/components/maps/OfflineMapOverlay.tsx`** Implementación visual exacta de la **Pantalla 1B / 6B**. Renderiza el icono vectorial de "Satélite Tachado" y el mensaje de "Modo Manual" cuando el `Circuit Breaker` de mapas (RF-6.5) se activa, reemplazando el canvas del mapa de forma elegante.
- **`/components/ui/feedback/DataSyncBadge.tsx`** Badge numérico reactivo para el **Header (Pantalla 6)**. Se suscribe al store de la cola offline y cambia de color (Naranja/Rojo/Verde) según el estado de sincronización y errores pendientes, proporcionando visibilidad inmediata de la "Deuda de Datos" al usuario.
- **`/components/ui/icons/custom-icons.tsx`** Colección de iconos SVG personalizados que no existen en librerías estándar: "Satélite Tachado" (RF-6.5), "Simbolo E-14", y los iconos de estado de "Eco-Ink". Centraliza los vectores gráficos requeridos para los estados de **Modo Vereda** y **Reportes Físicos** (RF-7.8), permitiendo su optimización y tree-shaking.
- **`components/dashboard/widgets/HealthScoreGauge.tsx`** Widget visual para el Admin que muestra el "Health Score" de la línea de WhatsApp (Calidad de envíos vs. Bloqueos). Provee visibilidad inmediata sobre el riesgo de bloqueo de la línea telefónica, complementando al `WhatsappHealthWidget` con un indicador tipo velocímetro crítico para la toma de decisiones.
- **`/components/dashboard/disputes/DisputeHistoryTimeline.tsx`** Componente visual para mostrar el historial de conflictos de un usuario o zona específica. Lista cronológicamente: Creación, Asignación de Juez, y Resolución. Necesario para que el Juez Jerárquico entienda el contexto de una disputa (reincidencia) antes de emitir un veredicto.
- **`/components/ui/spinner.tsx`** Componente visual atómico para estados de carga locales (micro-interacciones). Necesario específicamente para el feedback visual dentro del campo de Cédula ("Validando...") descrito en la **Pantalla 1** (RF-1.1), diferenciándose de los Skeletons de carga de página completa.
- **`/components/ui/checkbox.tsx`** Primitiva de interfaz crítica. Requerida explícitamente para el **RF-3.2 (Consentimiento Habeas Data)** y la selección múltiple en el **RF-1.11 (Declaración Juramentada de Importación)** y **RF-7.8 (Planillas de Punteo)**. Debe soportar estados "indeterminados" y validación de formularios Zod.
- **`/components/ui/switch.tsx`** Componente de control binario. Necesario para el **RF-3.10 (Interruptor Modo Emergencia/Lite)** en el panel de Admin y para el control **RF-93 (Asignación Manual de Zona)** en el perfil de usuario. Su estado visual debe ser claro bajo luz solar directa (Alto Contraste RNF-4).
- **`/components/ui/dialog.tsx`** Base arquitectónica para todos los modales del sistema. Es la dependencia fundamental para **RF-3.7 (Modal de Bienvenida)**, **RF-3.8 (Legal Wall)** y **RF-6.6 (Resolución de Conflictos)**. Debe gestionar el bloqueo de scroll del `body` y el foco trap para accesibilidad.
- **`/components/ui/scroll-area.tsx`** Contenedor con scroll virtualizado. Indispensable para el **RF-3.2 (Términos y Condiciones)** y el **RF-5.10 (Centro de Notificaciones)**, permitiendo mostrar contenido extenso en móviles sin romper el layout de la PWA.
- **`/components/ui/badge.tsx`** Componente visual para etiquetas de estado. Necesario para renderizar los estados de las plantillas (**RF-5.3**: `APPROVED` / `REJECTED`) y los indicadores de sincronización (**RF-3.5**: `DataSyncBadge` utiliza este primitivo).
- **`/components/ui/slider.tsx`** Componente de entrada de rango. Requerido para la **RF-1.13 (Definición de Radio de Eventos)**, permitiendo al Coordinador ajustar el Geofence (ej: 50m - 500m) visualmente al crear un evento.
- **`/components/maps/OfflineTileLayer.tsx`** Componente para `LeafletMapWrapper` que gestiona el almacenamiento en caché de teselas (tiles) de mapa en IndexedDB. Permite que el mapa base (calles principales) siga siendo visible en "Modo Vereda" si el usuario ya había navegado la zona, mejorando la UX de contingencia.
- **`/components/dashboard/widgets/TargetAudiencePreview.tsx`** Componente para el Wizard de Comunicaciones. Antes de enviar, muestra una previsualización estadística de la audiencia: "Total" vs "Contactables" (con WhatsApp Consent) vs "Bloqueados" (Blacklist), dando claridad financiera al Admin sobre el alcance real.
- **`/components/ui/visually-hidden.tsx`** Componente primitivo de accesibilidad. Permite ocultar elementos visualmente pero mantenerlos legibles para lectores de pantalla (Screen Readers). Obligatorio para cumplir **RNF-4 (Accesibilidad WCAG AA)** en botones que solo tienen iconos (como los controles del mapa) o para etiquetas de formularios implícitas.
- **`/components/maps/MapControls/DrawSolidPolygonBtn.tsx`** Botón especializado que activa el `DrawingManager` con la configuración preestablecida de "Solo Polígonos". Abstrae la configuración de Google Maps API para garantizar que el usuario no pueda dibujar líneas o puntos cuando se requiere definir una Zona Territorial (RF-6.6).
- **`/components/ui/drawer.tsx`** Primitiva de "Bottom Sheet". Esencial para cumplir con **RF-7.1 (Campaign Switcher en Móvil)** y **RF-3.1 (Instrucciones de Instalación iOS)**. Reemplaza los modales centrados en dispositivos móviles para mejorar la ergonomía táctil (Thumb-zone friendly).
- **`/components/ui/tabs.tsx`** Primitiva de navegación por pestañas. Necesaria para organizar el **Dashboard de Coordinador** (RF-7.1), separando vistas densas como "Alertas de Fraude", "Conflictos Territoriales" y "Métricas Financieras" sin recargar la página.
- **`/components/ui/progress.tsx`** Barra de progreso lineal. Requisito visual para **RF-7.1.A (Progreso de Meta)** y **RF-3.9 (Carga de Evidencia E-14)**, donde se debe mostrar el porcentaje de subida de archivos grandes.
- **`/components/ui/input-otp.tsx`** Componente especializado para la entrada de códigos de 6 dígitos. Mejora la UX del **Login Omnicanal (RF-8.1)** y la **Recuperación de Cuenta**, manejando el foco automático entre casillas y el pegado directo desde el portapapeles.
- **`/components/ui/command.tsx`** Primitiva de búsqueda y filtrado tipo "Command Palette". Dependencia obligatoria para implementar **`VotingStationSelector` (RF-1.1)** y **`AsyncSelect`**, permitiendo filtrar listas de 12k+ puestos de votación con alto rendimiento en el cliente.
- **`/components/ui/sheet.tsx`** Panel lateral deslizante. Necesario para los filtros avanzados del **Visor de Auditoría (RF-8.8)** y el menú de **"The Fixer" (RF-6.7)**,

permitiendo trabajar en el mapa sin perder el contexto de la lista de tareas.

- **`/components/ui/calendar.tsx`** Dependencia del `date-range-picker.tsx`. Necesaria para seleccionar rangos de fechas en los reportes de **Crecimiento (RF-7.1)** y filtros de logs.
- **`/components/ui/accordion.tsx`** Lista colapsable. Necesaria para la sección de **Preguntas Frecuentes** en el módulo de Ayuda y para organizar la **Configuración Global (Colección E)** que contiene múltiples secciones anidadas (Legal, Financiero, Seguridad).
- **`/components/maps/OverlapHighlighter.tsx`** Componente visual para **RF-6.6**. Se encarga exclusivamente de renderizar en rojo parpadeante el área exacta de intersección entre dos polígonos `DISPUTED`, ayudando al coordinador a identificar visualmente qué pedazo de tierra está en conflicto.
- **`/components/feedback/SessionTimeoutModal.tsx`** Diferente al `SessionExpiryModal`. Este es un modal preventivo ("¿Sigues ahí?") que aparece 60 segundos antes de que expire el `Access Token` (RF-8.9) para permitir la renovación silenciosa en dispositivos compartidos, mejorando la seguridad en sedes de campaña.
- **`/components/admin/audit/DiffViewerJson.tsx`** Componente para visualizar los cambios técnicos en `audit_logs` que no son geográficos (ej: cambio de `branding` o `system_config`). Renderiza un árbol JSON comparativo con resaltado de sintaxis para el SUPER_ADMIN.
- **`/components/ui/date-picker.tsx`** Componente de selección de fecha única. La lista original incluía `date-range-picker`, pero el registro de nacimiento requiere un selector singular optimizado para móviles (selectores nativos o wheel picker) que facilite ir 18+ años atrás rápidamente.
- **`/components/communications/RichTextEditor.tsx`** Área de texto enriquecida que soporta y visualiza sintaxis de WhatsApp (*negrita*, *cursiva*). Debe integrar el `TemplateVariableInput` y proporcionar feedback visual inmediato sobre la longitud de los segmentos GSM-7/UCS-2.
- **`/components/dashboard/widgets/VotingStationStatsTable.tsx`** Tabla compacta para el Dashboard que lista los Puestos de Votación con mayor concentración de la "Tropa Propia". Complementa al Mapa de Calor ofreciendo datos exactos para la toma de decisiones logísticas (ej: dónde enviar transporte).
- **`/components/dashboard/widgets/GoalProgressRing.tsx`** Variación visual del `MotivationGauge`. Un anillo de progreso circular pequeño diseñado para insertarse en las tarjetas de lista de "Top Movers" o en el header móvil, mostrando el % de meta de forma menos intrusiva que el gráfico completo.
- **`/components/maps/DrawingTools/CutPolygonTool.tsx`** Herramienta específica para "cortar" o dividir un polígono existente en dos. Necesaria operativa para cuando un Coordinador debe ceder una parte de su zona a otro sin borrar y redibujar todo el territorio (mantiene la integridad del ID).
- **`/components/ui/qr-reader.tsx`** Este componente permite usar la cámara del dispositivo para **leer** QRs. Esencial para **RF-1.12** (Escanear nuevo líder para disputa) y **RF-1.13** (Check-in en Eventos). Debe manejar permisos de cámara y fallback en iOS.
- **`/components/ui/visual/ConfettiTrigger.tsx`** Componente lógico que envuelve al `ConfettiCanvas`. Escucha los eventos globales del **RF-4.5** (ej: `GOAL_REACHED`) y dispara la animación. Separa la lógica de "cuándo celebrar" de la lógica de "cómo renderizar".
- **`/components/events/CheckInScanner.tsx`** Implementación específica del escáner para el control de asistencia. Valida contra el endpoint de eventos y muestra feedback inmediato (Éxito/Error de Geofence) al operador logístico en la entrada del evento.
- **`/components/events/EventMapRadius.tsx`** Componente visual para el formulario de creación de eventos. Renderiza un círculo ajustable sobre el mapa para definir visualmente el **Geo-Fence** (Radio permitido) requerido en el **RF-1.13**.
- **`/components/security/ActionPermitModal.tsx`** Modal de "Doble Confirmación" para acciones destructivas (ej: "Borrar Zona"). A diferencia del `CriticalActionGate` (que es financiero), este componente implementa la **confirmación por frase** ("Escribe 'ELIMINAR' para confirmar") requerida en flujos operativos críticos (RF-6.6 Deadlock Breaker).

---

5. Cloud Functions (`/functions`)

- **`/functions/package.json`** Dependencias aisladas para el entorno de funciones Node.js (firebase-functions, firebase-admin).
- **`/functions/src/index.ts`** Punto de entrada. Exporta y agrupa los triggers disponibles.
- **`/functions/src/aggregations.ts`** Triggers `onCreate` / `onUpdate` en usuarios. Gestiona contadores métricos distribuidos (Sharded Counters) (RF-4.5).
- **`/functions/src/security.ts`** Protocolo Anti-Fraude. Detecta anomalías de velocidad de registro y asigna `fraudStatus` (RF-1.10).
- **`/functions/src/topology.ts`** Gestor de integridad territorial. Recalcula pertenencia a zonas de usuarios tras edición de polígonos (RF-6.6.E).
- **`/functions/src/maintenance.ts`** Tareas programadas. Limpieza de logs antiguos, movimiento a Cold Storage y anonimización diferida (RF-8.11).
- **`/functions/src/messaging-worker.ts`** Worker para Cloud Tasks. Procesa cola de envío de SMS, maneja Rate Limiting y evita timeouts (RF-5.1).
- **`/functions/src/geocoding-worker.ts`** Worker para Cloud Tasks. Procesa cola de normalización de direcciones diferidas (RF-6.7). Consumidor de la API de Maps.
- **`/functions/src/campaign-triggers.ts`** Trigger (`onCreate`) de Campaña. Inicializa contadores, crea zonas por defecto y configura permisos iniciales.
- **`/functions/src/hierarchy-triggers.ts`** Triggers dedicados para la integridad estructural. Maneja el **Protocolo de Herencia de Nodos (RF-1.5)** cuando se elimina un usuario y la **Liberación de Responsabilidad Territorial (RF-1.6)** cuando un coordinador es degradado. Separado de `aggregations.ts` por su complejidad lógica recursiva.
- **`/functions/src/email-worker.ts`** Worker para Cloud Tasks. El **RF-4.3 (Security Broadcast)** y el **RF-5.6 (Alertas Financieras)** requieren envío de correos críticos. Hacerlo sincrónicamente en una Server Action es riesgoso. Este worker procesa una cola de envío de emails (vía Resend/SendGrid) garantizando reintentos y que el Admin reciba la alerta incluso si la UI falla.
- **`/functions/src/divipol-worker.ts`** Worker de Cloud Tasks. Procesa la importación masiva del archivo de la Registraduría (RF-1.14). Ejecuta la geocodificación por lotes de los puestos de votación para no saturar la cuota de la API de Maps en tiempo real.
- **`/functions/src/backup-worker.ts`** **RF-8.6 / RF-8.11:** Función programada (Cloud Scheduler) para la gestión del ciclo de vida de los datos. Ejecuta la exportación diaria de Firestore a Google Cloud Storage (Cold Storage) y procesa la eliminación física de logs vencidos (>180 días) para mantener los costos de almacenamiento bajo control y cumplir normas de retención.
- **`/functions/src/limbo-resolver.ts`** Función en segundo plano para el **Protocolo de Excepción (RF-92)**. Escanea periódicamente usuarios con `calculatedZoneId: 'ZONE_UNCATEGORIZED'` y re-intenta la asignación espacial (Point-in-Polygon) contra nuevas zonas creadas, o notifica al Coordinador Jerárquico si la orfandad persiste más de 24h.
- **`/functions/src/notification-router.ts`** Implementación del **RF-5.10 (Matriz Centralizada de Notificaciones)**. Recibe eventos del sistema (ej: `FRAUD_ALERT`, `GOAL_REACHED`), consulta la matriz de canales por rol y despacha la tarea al `email-worker` o al servicio de Push, aplicando la regla de "Silencio Nocturno".
- **`/functions/src/soft-lock-manager.ts`** Lógica aislada para gestionar los estados de bloqueo preventivo (`NEEDS_VERIFICATION`). Separa la lógica de *detección* (que está en `security.ts`) de la lógica de *gestión de estado*, permitiendo re-evaluaciones periódicas o desbloqueos automáticos si se cambian las reglas.
- **`/functions/src/session-revocation.ts`** Función reactiva (`onUpdate` en `users`) que detecta cambios críticos de rol (ej: Coordinador -> Seguidor) e invoca la invalidación inmediata de tokens de sesión en Firebase Auth (Security Kill-Switch).
- **`/functions/src/storage-triggers.ts`** Triggers de Google Cloud Storage (`onFinalize`). Se encargan de generar miniaturas (thumbnails) de las evidencias E-14 y avatares para optimizar la carga en redes móviles (RNF-1).
- **`/functions/src/import-worker.ts`** Aunque existe un API route, procesar 5,000 registros con validaciones jerárquicas y de duplicados puede exceder el timeout de una Cloud Function HTTP (60s) o Server Action. Este worker procesa el archivo CSV en segundo plano (Background Trigger) línea por línea, garantizando atomicidad y robustez.
- **`/functions/src/integrity-check.ts`** Función programada (Scheduled Function) que implementa la lógica definida en `lib/services/integrity-monitor.ts`. Se ejecuta periódicamente para sanear la base de datos de inconsistencias estructurales que hayan podido evadir las validaciones transaccionales.
- **`/functions/src/scope-compliance.ts`** Cloud Function (`onUpdate` en `users`). Monitorea cambios en `location.geoL1` y `geoL2`. Si la nueva ubicación

del usuario viola el `scope` de su campaña activa, ejecuta automáticamente el "Soft-Exit": cambia el estado de la membresía a `ARCHIVED`, registra la causal `OUT_OF_JURISDICTION` y decrementa los contadores del líder, enviando la notificación de advertencia correspondiente.

- **`/functions/src/export-worker.ts`** Worker de Cloud Tasks dedicado para la generación asíncrona de reportes "Modo Master". Dado que la exportación de bases de datos grandes (>100k usuarios) con enmascaramiento DLP y marcado forense puede exceder el tiempo límite de una petición HTTP o Server Action, este worker genera el archivo en *stream* hacia Google Cloud Storage y notifica al Admin vía email cuando la descarga segura está lista.
- **`/functions/src/user-lifecycle.ts`** Trigger `onCreate` dedicado a la experiencia de bienvenida. Genera el token de invitación único, lo guarda en el perfil del usuario y despacha el correo de "Welcome Kit". Se separa de `aggregations.ts` para no mezclar lógica métrica con lógica de negocio/comunicación.
- **`/functions/src/event-triggers.ts`** Trigger programado o reactivo para cerrar eventos automáticamente cuando pasa su `endDateTime`. Calcula estadísticas finales de asistencia (`isVerified`) y actualiza el estado a `COMPLETED`, liberando recursos de monitoreo geoespacial.
- **`/functions/src/warmup-scheduler.ts`** Función programada dedicada a la "Estrategia de Calentamiento". Gestiona la liberación progresiva de mensajes a usuarios importados (límite 500/día), moviendo lotes de la cola de espera a la cola de envío activo diariamente.
- **`/functions/src/storage-cleanup.ts`** Función programada para eliminar archivos temporales de exportación (CSV/Excel) generados por el `export-worker` después de un TTL corto (ej: 1 hora). Reduce la superficie de ataque asegurando que los archivos con datos sensibles (incluso enmascarados) no permanezcan en el bucket de Storage indefinidamente si el Admin olvida descargarlos.
- **`/functions/src/auth-claims-sync.ts`** Trigger `onUpdate` de la colección `users` que sincroniza el campo `role` y `campaignId` hacia los **Custom Claims** de Firebase Auth.
  - **Justificación Crítica:** Permite que `firestore.rules` valide permisos (`request.auth.token.role`) sin tener que leer el documento del usuario en cada petición (costo de lectura extra + latencia). Esto es vital para cumplir con el RNF-7 (Costos) y RNF-1 (Rendimiento) en un sistema con alto volumen de lecturas.
- **`/functions/src/types/shared.d.ts`** Archivo de tipos monorepo. Permite compartir las interfaces de base de datos (`User`, `CampaignConfig`, `AuditLog`) entre el *Frontend* (Next.js) y las *Cloud Functions*. Garantiza que cuando el trigger `aggregations.ts` (Backend) escriba un contador, lo haga con el mismo nombre de propiedad que el hook `use-sharded-count.ts` (Frontend) espera leer, evitando desincronización de esquema.
- **`/functions/src/config/env.ts`** Esquema de validación para las variables de entorno dentro del entorno de Cloud Functions. A diferencia del frontend, el backend necesita validar tipos estrictos para secretos como `TWILIO_AUTH_TOKEN` al arrancar la instancia, evitando fallos silenciosos en los Workers (RF-5.1).
- **`/functions/test/aggregations.test.ts`** Pruebas unitarias para **RF-4.5 (Agregación Asíncrona)**. Simula eventos de Firestore (`onWrite`) para verificar que los contadores `recruitedCount` y `assignedVoters` se incrementan/decrementan correctamente y que la lógica de *Sharded Counters* distribuye la carga.
- **`/functions/test/security-triggers.test.ts`** Pruebas para **RF-1.10 (Anti-Fraude)**. Simula ráfagas de creación de usuarios para asegurar que la función detecta la anomalía de velocidad y marca el `fraudStatus` correctamente sin bloquear el hilo principal.
- **`/functions/test/topology.test.ts`** Pruebas para **RF-6.6 (Integridad Territorial)**. Verifica que, al actualizar un polígono en la base de datos simulada, la función recalcula correctamente el `calculatedZoneId` de los usuarios afectados y activa las banderas de `isCrossZone`.

---

6. Tests (`/tests`)

- **`/tests/unit/merging-logic.test.ts`** Pruebas unitarias críticas para la Matriz de Fusión de Datos Offline (RF-3.5).
- **`/tests/unit/cycle-prevention.test.ts`** Prueba lógica del algoritmo de prevención de ciclos infinitos en jerarquía (RF-1.9).
- **`/tests/e2e/super-admin-approval.spec.ts`** Prueba E2E del flujo de aprobación de rol Admin: Solicitud -> Revisión Super Admin -> Aprobación.
- **`/tests/e2e/campaign-creation.spec.ts`** Prueba E2E del Wizard "Day Zero". Verifica la creación correcta del Político y la Campaña.
- **`/tests/e2e/join-campaign.spec.ts`** Prueba E2E de adhesión orgánica. Verifica búsqueda de campaña y asignación territorial correcta (Geo-Routing).
- **`/tests/e2e/offline-flow.spec.ts`** Prueba Playwright. Simula desconexión de red, captura de datos local y sincronización posterior.
- **`/tests/e2e/registration.spec.ts`** Prueba Playwright. Valida flujo completo: QR -> Registro -> Atribución Correcta -> Redirección.
- **`/tests/security/firestore-rules.test.ts`** Suite de pruebas unitarias para validar `firestore.rules`. Esencial para garantizar el **Mandato de Congelamiento (RF-4.6)**, la segregación de datos por campaña (Tenant Isolation) y que los campos de auditoría sean inmutables.
- **`/tests/mocks/google-maps.mock.ts`** Mock global del objeto `google.maps` y `google.maps.drawing`. Necesario para ejecutar pruebas unitarias y de integración en componentes como `TerritoryMap` y `ZoneEditor` sin requerir una conexión real a la API ni un entorno de navegador con Canvas (RF-6.6).
- **`/tests/mocks/indexed-db.mock.ts`** Simulación de la API `IndexedDB` (`fake-indexeddb`). Esencial para probar la lógica de persistencia offline y la cola de sincronización (RF-3.5) en el entorno de CI/CD (GitHub Actions) donde no existe persistencia real del navegador.
- **`/tests/load/k6-scripts/registration-stress.js`** **RNF-2 / RNF-9:** Script de prueba de carga para Grafana k6. Simula "Picos 10x" de tráfico de registro concurrente, validando que la arquitectura de *Sharded Counters* (RF-4.5) y las *Cloud Functions* escalen sin degradar el tiempo de respuesta por debajo de 2s.
- **`/tests/load/k6-scripts/cold-start-benchmark.js`** **RNF-1:** Script específico para medir la latencia de arranque en frío de las Cloud Functions y el contenedor de Cloud Run, asegurando que la configuración `min-instances: 1` en producción sea efectiva.
- **`/tests/integration/map-topology.test.ts`** Pruebas de integración específicas para validar que la librería `Turf.js` en el backend calcula correctamente las intersecciones y la lógica de "No Donuts" con polígonos complejos geoJSON.
- **`/tests/unit/masking-logic.test.ts`** Pruebas unitarias para asegurar que las funciones de enmascaramiento (`lib/utils/masking.ts`) nunca expongan datos completos en el modo "Operativo" y manejen correctamente los casos borde.
- **`/tests/utils/firebase-emulator-connect.ts`** Helper para conectar los tests E2E (Playwright) y Unitarios (Jest) a los emuladores locales de Firebase Auth y Firestore. Garantiza que las pruebas no toquen la base de datos de producción ni staging, permitiendo un entorno de CI/CD aislado y seguro.
- **`/tests/mocks/cloud-tasks.mock.ts`** Mock específico para simular el comportamiento de Google Cloud Tasks. Necesario para probar la lógica de "Productor" (encolado de mensajes/geocodificación) sin requerir credenciales reales de GCP en el entorno de pruebas local.
- **`/tests/security/storage-rules.test.ts`** Pruebas unitarias para las reglas de seguridad de Storage (`storage.rules`). Valida que un usuario normal no pueda sobrescribir la foto de un E-14 y que los avatares tengan restricción de tamaño y tipo MIME.
- **`/tests/e2e/map-fallback.spec.ts`** Prueba E2E específica que intercepta las peticiones a Google Maps API para simular fallos (403/Quota) y verifica que la UI cambie correctamente al componente `OfflineWidget` (Formulario Manual) sin romper la experiencia.
- **`/tests/unit/boomerang.test.ts`** Prueba unitaria para la lógica de "Reingreso". Verifica matemáticamente que al mover un usuario fuera y dentro de la zona, la antigüedad se preserve y los contadores se ajusten correctamente (Idempotencia).
- **`/tests/privacy/data-scrubbing.test.ts`** Prueba de integración crítica para el **RF-8.3 (Derecho al Olvido)**. Simula la ejecución de `actions/user-archival.ts` y verifica aserciones estrictas: que los campos PII (`phone`, `fullName`) sean efectivamente sobrescritos, que el `documentId` sea liberado (hash) y que las métricas estructurales (`recruitedCount`) permanezcan intactas (Integridad Referencial).
- **`/tests/unit/cost-estimation.test.ts`** Pruebas unitarias críticas para la lógica financiera. Valida que el cálculo de segmentos SMS (GSM-7 vs UCS-2) y la multiplicación por tarifas (con 4 decimales) sea matemáticamente exacta para prevenir pérdidas por redondeo.
- **`/tests/integration/survey-flow.test.ts`** Prueba de integración que simula el ciclo completo de una encuesta: Recepción de Webhook -> Clasificación de Respuesta -> Validación de Cardinalidad -> Persistencia en DB. Asegura que las respuestas se guarden correctamente.
- **`/tests/e2e/mobile-viewport.spec.ts`** Suite de pruebas E2E específica para validar **RF-7.5**. Configura Playwright para emular viewport móvil y verifica que: 1) La `Sidebar` desaparezca, 2) La `MobileNavBar` aparezca, y 3) Las tablas se transformen en `AdaptiveUserCards`.
- **`/tests/e2e/offline-sync-strategies.spec.ts`** Prueba avanzada para **RF-3.5**. Simula respuestas 500 y 401 del backend para verificar que el cliente aplique correctamente la estrategia de espera (Backoff) o pausado de cola (Auth Fail), garantizando que no se pierdan datos por reintentos infinitos erróneos.

- `/tests/e2e/map-concurrency.spec.ts` Prueba automatizada que simula dos usuarios (Coordinador A y B) editando el mismo polígono simultáneamente para validar el **Protocolo de Integridad de Edición (RF-6.6)** y el bloqueo optimista. Escenario crítico de alta complejidad difícil de probar manualmente. Asegura que el mecanismo de `Version Check` funcione y evite la corrupción de datos territoriales.
- `/tests/e2e/circuit-breaker.spec.ts` Prueba que simula un gasto acelerado de presupuesto para verificar que el sistema active el "Queue Freeze" y detenga los envíos automáticamente. Valida la protección financiera del sistema ante errores de bucle infinito antes de pasar a producción.
- `/tests/e2e/performance/lighthouse.spec.ts` Script de auditoría automática. Ejecuta Google Lighthouse dentro de Playwright en cada despliegue. Verifica el cumplimiento estricto del **RNF-1 (Performance < 2s)** y **RNF-4 (Accesibilidad/WCAG)**, fallando el pipeline si el contraste de colores o la velocidad de carga de la PWA descienden por debajo de los umbrales definidos.
- `/tests/integration/financial-concurrency.test.ts` Prueba de integración avanzada que simula condiciones de carrera (Race Conditions). Ejecuta múltiples débitos y recargas simultáneas en la billetera virtual (usando emuladores) para validar que el saldo final sea matemáticamente exacto y que no ocurra "Doble Gasto".
- `/tests/unit/geohash-neighbors.test.ts` Pruebas unitarias para la utilidad de cálculo de vecinos Geohash. Verifica que la lógica de proximidad cubra correctamente los bordes de las celdas geoespaciales, asegurando que los "Eventos Cercanos" o "Puestos de Votación" siempre aparezcan correctamente en la búsqueda local.
- `/tests/load/scenarios/event-checkin-storm.js` Escenario de prueba de carga (k6) específico para el **RF-1.13**. Simula cientos de usuarios escaneando QRs simultáneamente en un evento masivo para asegurar que el endpoint de validación de Geo-Fence responda en < 1s.
- `/tests/security/rate-limit.test.ts` Prueba de integración para verificar la eficacia del **RF-8.7**. Ataca los endpoints de OTP y Login desde una misma IP simulada para confirmar que el bloqueo temporal se activa según lo configurado.

---

7. Assets y Públicos (`/public`)

- `/public/firebase-messaging-sw.js` Worker específico para recibir mensajes *push* en segundo plano (FCM) cuando la PWA está cerrada. Es distinto del Service Worker de la aplicación (`sw.js`) y es crucial para el **RF-5.10 (Notificaciones)** en Android/iOS.
- `/public/sw.js` Service Worker personalizado. Gestión de caché de assets, estrategia "NetworkFirst" para API y Background Sync (RF-3.5).
- `/public/.well-known/assetlinks.json` Archivo de verificación de Digital Asset Links para Android. Es obligatorio para habilitar *Android App Links* verificado, permitiendo que los enlaces mágicos o notificaciones abran forzosamente la PWA instalada en lugar del navegador, preservando el contexto de sesión y la base de datos local (IndexedDB).
- `/public/.well-known/apple-app-site-association` Archivo de configuración AASA para iOS. Habilita los *Universal Links*. Crítico para que el flujo de recuperación de cuenta o invitaciones en iPhones no expulse al usuario de la PWA a Safari, lo cual rompería la experiencia "Offline-First".
- `/public/images/` Logotipos por defecto, imágenes placeholder, marcadores de mapa personalizados y assets para emails.
- `/public/images/markers/cluster-icon.svg` Icono optimizado para los clusters de mapas (**RF-6.4**). Necesario para representar grupos de 100+ votantes sin saturar el renderizado visual.
- `/public/legal/habeas_data_full_v1.pdf` Documento PDF estático con la política de tratamiento de datos completa (RF-3.2, Anexo B). Se enlaza desde el Checkbox de registro y el "Legal Wall".
- `/public/sounds/error.mp3` Feedback auditivo para errores de escaneo o validación (ej: "Usuario fuera de zona"). Complementa al `success.mp3` existente para operaciones "Eyes-free" en campo.
- `/public/sounds/success.mp3` Recurso de audio breve para feedback positivo. Se reproduce junto con la vibración (Haptics) cuando se completa un registro exitoso o se escanea un QR válido, reforzando la confirmación en entornos de calle ruidosos.
- `/public/templates/import_users_template_v1.csv` Archivo plantilla descargable obligatorio para la Importación Masiva (RF-1.11). Contiene las cabeceras exactas (`cedula` , `nombre` , `telefono` , `direccion` , `codigo_lider`) para garantizar la validación del esquema.

8. Tipado y Definiciones (`/types`)

- `/types/api-responses.d.ts` Interfaces genéricas para estandarizar todas las respuestas JSON de la API (`ApiResponse<T>` , `PaginatedResponse<T>`). Garantiza que el Frontend siempre sepa cómo consumir errores o datos paginados, mejorando la robustez del código cliente.
- `/types/audit-actions.d.ts` Enum o Union Type exhaustivo para `actionType` (`LOGIN` , `BULK_MOVE` , `MANUAL_GEO_FIX` , etc.). Garantiza que en todo el código (frontend y backend) se usen las mismas cadenas constantes para registrar eventos en `audit_logs` , evitando errores de inconsistencia en reportes (RF-8.11).
- `/types/audit-log-filters.d.ts` Interfaces para los filtros complejos del "Visor de Auditoría" (Pantalla 11, Pág. 142). Define las estructuras para búsquedas por rangos de fecha, actores y tipos de acción (`actionType` enum). Necesario para tipar los props del componente `LogExplorer` y las queries compuestas de Firestore en **RF-8.8**.
- `/types/csv-import-schema.d.ts` Definición estricta de las interfaces para las filas del CSV de importación de usuarios. Tipa las columnas esperadas (`cedula` , `telefono` , `codigo_lider`) y los códigos de error de validación específicos para el reporte de feedback (`INVALID_FORMAT` , `DUPLICATE_ID` , `TI_DETECTED`).
- `/types/day-d.d.ts` Definiciones de tipos para el **Mandato de Congelamiento (RF-4.6)**. Define las interfaces para `witness_ids` , `transport_coordinator_id` y `e14_evidence` . Aunque no se usen en la UI v1.0, son necesarios para que TypeScript valide que el esquema de la base de datos respeta los placeholders reservados para la Fase 3.
- `/types/divipol.d.ts` Definiciones de tipos para la estructura de datos oficial de la Registraduría (Departamento, Municipio, Puesto, Mesa). Distinto de las zonas de campaña, necesario para la ingesta y validación de datos maestros.
- `/types/env.d.ts` Extensión de tipos para `NodeJS.ProcessEnv` . Garantiza que secretos críticos como `AUTH_SECRET` o `MAPS_SERVER_KEY` (RF-8.10) estén tipados y no sean `undefined` en tiempo de ejecución.
- `/types/geo-fixer.d.ts` Tipos para el payload de corrección manual de direcciones. Define la estructura del objeto que recibe el endpoint `/geo-fix` , asegurando que la corrección incluya tanto la coordenada manual (`LatLng`) como la referencia al `audit_log` original para cerrar el ciclo de trazabilidad.
- `/types/geo-json.d.ts` Definiciones estrictas para objetos GeoJSON (FeatureCollection, Polygon, Point) utilizadas por `Turf.js` y la API de Mapas. Asegura que la validación topológica (RF-6.6) tenga tipos correctos.
- `/types/next-auth.d.ts` Extensión de tipos para `NextAuth` . Crucial para añadir propiedades personalizadas (`uid` , `role` , `campaignId` , `hierarchyPath`) a los objetos `Session` y `JWT` . Sin esto, TypeScript fallará al intentar acceder a estos campos en el middleware o componentes (RF-8.1).
- `/types/polymorphic-user.d.ts` **NUEVO**. Definición de tipo avanzada para manejar la dualidad del usuario en modo Offline. Un usuario puede existir como registro completo en Firestore (`User`) o como un borrador parcial en IndexedDB (`DraftUser`). Este tipo unifica ambas interfaces para que los componentes de UI puedan renderizar datos sin importar la fuente (RF-3.5).
- `/types/sync-conflicts.d.ts` Definiciones de tipos para la **Matriz de Fusión de Datos (RF-3.5)**. Define las estrategias de resolución (`server_wins` , `merge` , `client_wins`) y la estructura de los documentos en `sync_conflicts` , vital para el tipado estricto en los tests unitarios.
- `/types/system-config.d.ts` Definición estricta de la interfaz TypeScript para el documento singleton `system_config` (Colección E, Pág. 49). Es vital para que el código consuma de forma segura las tarifas (`costPerMessage`), banderas (`featureFlags`), configuración legal (`currentTermsVersion`) y la lista blanca de IPs (`globalWhitelistedIps`) mencionadas en **RF-4.2.E** y **RF-5.6**, evitando errores de tipo en tiempo de ejecución.
- `/types/topology.d.ts` Definiciones estrictas para las operaciones geométricas complejas de **RF-6.6**. Tipa las estructuras de intersección, validación de "Donuts" y los resultados del cálculo de área superpuesta necesarios para el componente `ZoneConflictResolver` .

- **`/types/twilio-custom.d.ts`** Definiciones de tipos para los payloads específicos de la **API de Contenidos de Twilio** (que difieren del SMS estándar). Necesario para tipar correctamente las plantillas HSM, botones `QUICK_REPLY` y configuraciones de encuestas en `message_templates`.

9. Scripts de Mantenimiento (`/scripts`)

- **`/scripts/audit-hierarchy-local.ts`** Script de ejecución local para desarrolladores. Recorre toda la base de datos (emulada o staging) buscando ciclos infinitos o nodos huérfanos en la jerarquía `hierarchyPath`, útil para debugging antes de desplegar cambios en la lógica de movimiento.
- **`/scripts/create-super-admin.ts`** Script de seguridad para crear el primer usuario `SUPER_ADMIN` en Firebase Auth y Firestore (Bootstrap), ya que no existe flujo de registro público para este rol (RF-2. Estructura Jerárquica).
- **`/scripts/check-dependencies.ts`** Script de validación pre-commit o CI. Verifica que las versiones de librerías críticas (`next-auth`, `firebase`) coincidan exactamente con las definidas en la política de "Freeze" (Sección 11), evitando actualizaciones accidentales que rompan la estabilidad en la semana electoral.
- **`/scripts/rotate-secrets.ts`** Script de utilidad para facilitar la rotación de credenciales. Valida que los nuevos secretos en Google Secret Manager sean accesibles y funcionales (ej: prueba de conexión a Twilio) antes de reiniciar los servicios de Cloud Run.
- **`/scripts/seed-divipol.ts`** Script de ejecución local o CI. Permite la carga inicial o actualización masiva de la colección `voting_places` desde archivos oficiales de la Registraduría (RF-1.14), manejando la geocodificación por lotes inicial.
- **`/scripts/seed-system-config.ts` RF-4.2.E / RF-5.6:** Script de inicialización para la colección `system_config`. Crítico para establecer las tarifas base (`costPerMessage`), los textos legales iniciales y las banderas de seguridad antes del primer despliegue, ya que la aplicación fallará si estos documentos singleton no existen.
- **`/scripts/verify-env-integrity.ts`** Script de "Pre-Build Check" para **RNF-8 (Segregación de Entornos)**. Valida que las llaves de producción (Live Credentials) no estén presentes cuando `NEXT_PUBLIC_ENV="staging"` y viceversa, previniendo accidentes financieros catastróficos antes del despliegue.
- **`/scripts/generate-vapid-keys.ts`** Script de utilidad para generar el par de llaves (Pública/Privada) VAPID necesarias para el servidor de notificaciones Push. Facilita la configuración del entorno local y de producción (`.env`) sin dependencias externas.
- **`/scripts/analyze-bundle-size.ts`** Script de configuración para `@next/bundle-analyzer`. Crítico para monitorear que las librerías pesadas (como `Mapbox` o `XLSX`) no se incluyan en el bundle principal de la PWA, garantizando la carga en < 2s.

10. Service Worker & Offline (`/worker`)

- **`/worker/index.ts`** Código fuente del Service Worker. Implementa la lógica de intercepción de red, gestión de `IndexedDB` y la estrategia de reintentos "Exponential Backoff" (RF-3.5) con tipado estricto. Se compila a `/public/sw.js`.
- **`/worker/sync-manager.ts`** Módulo aislado dentro del worker encargado de procesar la cola `sync_queue`. Contiene la lógica de "Fusión de Datos" (RF-3.5) para decidir si un registro offline debe sobrescribir o fusionarse con el servidor.