UNIVERSIDAD PILOTO DE COLOMBIA
SECCIONAL DEL ALTO MAGDALENA
FACULTAD DE INGENIERÍA
PROGRAMA INGENIERÍA DE SISTEMAS

MANUAL TÉCNICO DEL SISTEMA
OBSERVATORIO DE TRANSPORTE PÚBLICO DE GIRARDOT

Diseño, Desarrollo y Despliegue del Front-End de una Aplicación Web Administrativa y Móvil para la Gestión de Rutas y Paraderos de Transporte Público Colectivo para la Secretaría de Tránsito, en Girardot, Cundinamarca, para el Periodo 2026

LINDY YARITZA ARIAS GARCÍA
JONATHAN AFANADOR TORRES

Docentes:
LUDWIG IVÁN TRUJILLO HERNÁNDEZ
FRANKLIN GUILLERMO CARDOZO TORRES

Girardot, Cundinamarca
2026

TABLA DE CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

PRESENTACIÓN

El presente documento constituye el manual técnico del sistema Observatorio de Transporte Publico de Girardot, una plataforma web integral diseñada para la gestión, monitoreo y visualización geoespacial del transporte público colectivo en el municipio de Girardot, Cundinamarca. Este manual ha sido elaborado con el propósito de servir como referencia técnica completa para el personal encargado de la instalación, configuración, mantenimiento, soporte y evolución del sistema.

El documento está dirigido a los siguientes perfiles técnicos:

- **Ingenieros de sistemas:** responsables de la comprensión global de la arquitectura, integración de módulos y toma de decisiones técnicas sobre la evolución del sistema.
- **Desarrolladores de software:** encargados de la implementación de nuevas funcionalidades, corrección de errores y mantenimiento evolutivo del código fuente tanto en el backend (Laravel/PHP), el frontend web (Blade/JavaScript) como en la aplicación móvil (React Native/Expo).
- **Administradores de bases de datos:** responsables de la gestión, optimización y respaldo de la base de datos PostgreSQL, incluyendo la administración de migraciones y la integridad referencial.
- **Administradores de sistemas:** encargados del despliegue en servidores de producción, configuración de variables de entorno, gestión de certificados SSL y monitoreo del rendimiento.
- **Personal de soporte técnico:** encargado de atender incidencias de primer y segundo nivel, diagnosticar fallos y ejecutar procedimientos de recuperación.

Se requiere que el lector posea conocimientos intermedios a avanzados en las siguientes áreas: desarrollo web con frameworks MVC (preferiblemente Laravel), administración de bases de datos relacionales (PostgreSQL), gestión de servidores Linux o Windows Server, control de versiones con Git, y conceptos fundamentales de arquitectura de software cliente-servidor en capas.

El sistema adopta una arquitectura cliente-servidor en capas, organizando el software en tres niveles claramente diferenciados: una capa de presentación construida con Blade, Tailwind CSS 4 y JavaScript modular; una capa de lógica de negocio implementada en Laravel 12 con PHP 8.2, expuesta a través de una API RESTful documentada con Swagger (L5-Swagger); y una capa de persistencia soportada por PostgreSQL 16 con extensión PostGIS para el manejo de datos geoespaciales. Adicionalmente, el sistema incluye una aplicación móvil multiplataforma desarrollada con React Native y Expo SDK 54.

RESUMEN EJECUTIVO

El Observatorio de Transporte Publico de Girardot es un sistema web integral desarrollado como proyecto de grado para la Universidad Piloto de Colombia, en convenio con la Alcaldía de Girardot y su Secretaria de Tránsito y Transporte. Su objetivo principal es centralizar, digitalizar y automatizar la gestión operativa del transporte público colectivo del municipio, reemplazando los procesos manuales basados en hojas de cálculo y documentos físicos que históricamente han limitado la capacidad de toma de decisiones de la entidad.

El sistema permite la gestión completa del ciclo de vida de las empresas de transporte, sus flotas vehiculares, conductores, licencias de conducción, rutas con trazados geoespaciales, paraderos georreferenciados y documentos administrativos. Paralelamente, ofrece un módulo de geovisor público que permite a los ciudadanos visualizar las rutas de transporte sobre un mapa interactivo y planificar sus viajes.

Tabla 1
Tecnologías utilizadas

Componente	Tecnología	Versión
Lenguaje backend	PHP	8.2
Framework backend	Laravel	12.x
Motor de base de datos	PostgreSQL + PostGIS	16 / 17
Autenticación API	Laravel Sanctum	4.1
Documentación API	L5-Swagger (OpenAPI)	9.0
Auditoria	Laravel Auditing	14.0
Respaldos	Spatie Laravel Backup	9.3
Almacenamiento nube	Google Drive API	-
Bundler frontend	Vite	6.2
Framework CSS	Tailwind CSS	4.0
Mapas interactivos	Leaflet.js	1.9.4
Parser geoespacial	@mapbox/togeojson + JSZip	0.16 / 3.10
Componentes UI	TW Elements	2.0
Carruseles	Swiper.js	12.0
Framework móvil	React Native (Expo)	SDK 54 / RN 0.76
Navegación móvil	React Navigation	7.x
Mapas móvil	react-native-maps + react-native-webview	1.x / 13.x
Distribución móvil	APK + Expo Web (PWA)	-

Nota. Elaboración propia.

Estructura del manual:

El presente manual se organiza en seis capítulos: (1) Requisitos del sistema, (2) Diagramas de modelamiento, (3) Aspectos técnicos del desarrollo, (4) Instalación del sistema y servicios, (5) Requerimientos de hardware para operación, y (6) Bibliografía.

FINALIDAD DEL MANUAL

El manual técnico del Observatorio de Transporte Publico de Girardot ha sido elaborado con las siguientes finalidades:

- **Capacitar al personal técnico en la instalación del sistema:** proporcionar instrucciones detalladas y reproducibles para desplegar el sistema en un entorno de desarrollo local o en un servidor de producción, reduciendo la curva de aprendizaje y minimizando errores durante el proceso de puesta en marcha.
- **Facilitar tareas de mantenimiento correctivo y evolutivo:** ofrecer una documentación clara de la arquitectura, la estructura de archivos, los modelos de datos y las dependencias del sistema.
- **Servir como referencia para futuras modificaciones:** documentar las decisiones arquitectónicas, los patrones de diseño utilizados y las convenciones de codificación adoptadas.
- **Documentar técnicamente el sistema desarrollado:** cumplir con los requisitos académicos de la Universidad Piloto de Colombia para la entrega de proyectos de grado.
- **Garantizar la continuidad operativa:** asegurar que, ante la eventual rotación del personal técnico, el conocimiento del sistema no se pierda.

INTRODUCCIÓN

El presente manual técnico se estructura de manera progresiva, comenzando por los aspectos más generales del sistema y avanzando hacia los detalles específicos de implementación. Esta organización permite que el lector pueda navegar directamente al capítulo de su interés según la tarea técnica que necesite realizar.

En el Capítulo 1 se presentan los requisitos mínimos y recomendados de hardware y software necesarios para ejecutar el sistema, tanto en un entorno de desarrollo como en producción. El Capítulo 2 contiene los diagramas de modelamiento que representan la arquitectura, la estructura y el comportamiento del sistema. El Capítulo 3 constituye el núcleo técnico del manual, documentando cada una de las capas de la arquitectura. El Capítulo 4 proporciona una guía de instalación paso a paso. El Capítulo 5 aborda los requerimientos de hardware para la operación continua en producción. Finalmente, el Capítulo 6 presenta la bibliografía técnica consultada.

Se recomienda al lector iniciar por el Capítulo 1 para verificar los requisitos, continuar con el Capítulo 4 si su objetivo es la instalación, y consultar los Capítulos 2 y 3 para comprender la arquitectura y la estructura interna del sistema.

1. REQUISITOS DEL SISTEMA

1.1 Requisitos de Hardware

A continuación, se detallan los requisitos mínimos y recomendados de hardware para ejecutar el sistema, diferenciando entre el entorno de desarrollo y el entorno de producción.

Tabla 2
Requisitos de hardware — Entorno de desarrollo

Componente	Mínimo	Recomendado
Procesador	Intel Core i3 / AMD Ryzen 3 (2 núcleos)	Intel Core i5 / AMD Ryzen 5 (4 núcleos)
Memoria RAM	4 GB	8 GB o superior
Almacenamiento	10 GB disponibles (SSD recomendado)	20 GB en SSD NVMe
Arquitectura	64 bits (x86_64)	64 bits (x86_64)
Conexión a Internet	Requerida (instalación de dependencias)	Banda ancha >= 10 Mbps

Nota. Elaboración propia.

Tabla 3
Requisitos de hardware — Entorno de producción

Componente	Mínimo	Recomendado
Procesador	2 vCPUs	4 vCPUs
Memoria RAM	2 GB	4 GB o superior
Almacenamiento	20 GB SSD	40 GB SSD NVMe
Arquitectura	64 bits (x86_64)	64 bits (x86_64 / ARM64)
Ancho de banda	100 Mbps simétrico	1 Gbps simétrico

Nota. Elaboración propia.

1.2 Requisitos de Software

Tabla 4
Requisitos de software

Componente	Tecnología	Versión mínima	Notas
S.O. (desarrollo)	Windows 10/11, macOS 12+, Ubuntu 22.04+	-	Multiplataforma
S.O. (producción)	Ubuntu Server / Debian	22.04 LTS	Recomendado para estabilidad
Lenguaje backend	PHP	8.2	Con extensiones: pgsql, mbstring, openssl, curl, zip, gd
Framework backend	Laravel	12.x	-
Gestor dep. PHP	Composer	2.6+	-

Componente	Tecnología	Versión mínima	Notas
Motor BD	PostgreSQL	16	Con extensión PostGIS habilitada
Runtime JS	Node.js	18 LTS	Recomendado: 20 LTS
Gestor paquetes JS	npm	9+	Incluido con Node.js
Bundler	Vite	6.2	Plugin: @tailwindcss/vite
Framework CSS	Tailwind CSS	4.0	-
Framework móvil	React Native (Expo SDK)	54 (RN 0.76)	Android /Web
Navegación móvil	React Navigation	7.x	Stack + Drawer navigators
Control versiones	Git	2.40+	-
Navegadores	Chrome, Firefox, Edge, Safari	Ultimas 2 versiones	Expo Web compatible

Nota. Elaboración propia.

Dependencias Backend (Composer)

Las siguientes librerías PHP son requeridas y se instalan mediante composer install:

- laravel/framework ^12.0 — Framework principal.
- laravel/sanctum ^4.1 — Autenticación basada en tokens para la API RESTful.
- clickbar/laravel-magellan ^2.0 — Integración de tipos de datos geoespaciales de PostGIS con Eloquent ORM.
- darkaonline/l5-swagger ^9.0 — Generación automática de documentación API mediante anotaciones OpenAPI.
- owen-it/laravel-auditing ^14.0 — Registro automático de auditoría sobre operaciones CRUD.
- spatie/laravel-backup ^9.3 — Sistema de respaldos automáticos de la base de datos y archivos.
- masbug/flysystem-google-drive-ext — Driver Flysystem para almacenar respaldos en Google Drive.

Dependencias Frontend (npm)

Las siguientes librerías JavaScript se instalan mediante npm install:

- leaflet ^1.9.4 — Librería de mapas interactivos para el geovisor.
- leaflet-polylinedecorator ^1.6.0 — Decoración de polilíneas (flechas de dirección en rutas).
- @mapbox/togeojson ^0.16.2 — Conversión de archivos KML/KMZ a formato GeoJSON.
- jszip ^3.10.1 — Descompresión de archivos KMZ para extracción de datos geoespaciales.
- swiper ^12.0.3 — Carruseles táctiles para la interfaz de bienvenida.

- tw-elements ^2.0.0 — Componentes UI avanzados basados en Tailwind CSS.
- axios ^1.8.2 — Cliente HTTP para comunicación asíncrona con la API.

1.3 Configuraciones Previas

Variables de entorno

El sistema utiliza un archivo.env ubicado en la raíz del proyecto para almacenar las variables de configuración sensibles. Este archivo no se versiona en Git y debe ser creado manualmente en cada nuevo entorno a partir de la plantilla.env.example.

Tabla 5
Variables de entorno principales

Variable	Descripción	Ejemplo
APP_ENV	Entorno de ejecución	local / producción
APP_KEY	Clave de cifrado (php artisan key:generate)	base64:XXXX...
APP_DEBUG	Modo depuración (false en producción)	true / false
APP_URL	URL base de la aplicación	http://localhost
DB_CONNECTION	Driver de base de datos	pgsql
DB_HOST	Host del servidor de BD	127.0.0.1
DB_PORT	Puerto del servidor de BD	5432
DB_DATABASE	Nombre de la base de datos	backend_sm
DB_USERNAME	Usuario de base de datos	postgres
DB_PASSWORD	Contraseña de base de datos	********
PG_DUMP_PATH	Ruta al binario pg_dump para respaldos	C:/Program Files/PostgreSQL/17/bin
SANCTUM_STATEFUL_DOMAINS	Dominios autorizados para autenticación	localhost:8000,localhost:5173
GOOGLE_DRIVE_CLIENT_ID	ID de cliente OAuth para Google Drive	1072768...
GOOGLE_DRIVE_CLIENT_SECRET	Secreto del cliente OAuth	GOCSPX-...
GOOGLE_DRIVE_REFRESH_TOKEN	Token de refresco para acceso continuo	1//04ouC7...
GOOGLE_DRIVE_FOLDER_ID	ID de la carpeta destino en Google Drive	1iuogq...

Nota. Elaboración propia.

Tabla 6
Puertos utilizados

Servicio	Puerto	Protocolo
Laravel Development Server	8000	HTTP
Vite Dev Server (HMR)	5173	HTTP/WS
PostgreSQL	5432 (estándar) / 5433 (configurable)	TCP

Nota. Elaboración propia.

Permisos y servicios necesarios

- El directorio storage/ y bootstrap/cache/ deben tener permisos de escritura (775 en Linux).
- El servicio de PostgreSQL debe estar activo y aceptando conexiones en el puerto configurado.
- En producción, se requiere un proceso de cola activo: php artisan queue:listen.
- Para Google Drive, se requiere una cuenta de servicio OAuth configurada en Google Cloud Console.

1.4 Portabilidad y Adaptabilidad

El sistema ha sido diseñado con un alto grado de portabilidad gracias a las siguientes decisiones arquitectónicas:

- **Multiplataforma:** el stack tecnológico (PHP, Node.js, PostgreSQL) es compatible con Windows, macOS y Linux.
- **Containerización:** el proyecto incluye soporte para Laravel Sail, permitiendo ejecutar el entorno completo mediante Docker Compose.
- **Configuración externalizada:** toda la configuración sensible reside en el archivo.env, permitiendo adaptar el sistema a diferentes entornos sin modificar el código fuente.
- **Migraciones versionadas:** la estructura de la base de datos se gestiona mediante 51 archivos de migración Laravel, garantizando la reproducibilidad del esquema.
- **Assets compilados:** el frontend utiliza Vite como bundler, generando assets estáticos optimizados para producción mediante npm run build.

1.5 Contingencia

El sistema implementa los siguientes mecanismos de contingencia para garantizar la continuidad operativa ante fallos:

Sistema de respaldos automatizados

Se utiliza el paquete spatie/laravel-backup configurado para realizar respaldos completos de la base de datos PostgreSQL y los archivos subidos por los usuarios. Los respaldos se almacenan con compresión máxima y se suben automáticamente a Google Drive. La política de retención configurada es:

- Se conservan todos los respaldos durante 7 días.
- Después del día 7, se conserva un respaldo diario durante 16 días.

- Después del día 23, se conserva un respaldo semanal durante 8 semanas.
- Después de la semana 8, se conserva un respaldo mensual durante 4 meses.
- Después del mes 4, se conserva un respaldo anual durante 2 años.
- El espacio máximo permitido para respaldos es de 5.000 MB.

Recuperación ante fallos

- **Fallo de la base de datos:** restaurar el ultimo respaldo desde Google Drive utilizando pg_restore y ejecutar php artisan migrate.
- **Fallo del servidor web:** redesplegar la aplicación desde el repositorio Git y reconstruir los assets con npm run build.
- **Corrupción de archivos subidos:** restaurar el directorio storage/app/public desde el respaldo comprimido.
- **Perdida de credenciales:** regenerar la clave con php artisan key:generate y actualizar los tokens OAuth de Google Drive.

Auditoria y trazabilidad

El paquete owen-it/laravel-auditing registra automáticamente cada operación de creación, actualización y eliminación sobre los modelos auditables del sistema. Estos registros se almacenan en la tabla audits de PostgreSQL e incluyen el usuario que ejecuto la acción, la IP de origen, los valores anteriores y nuevos de los campos modificados, y la marca de tiempo exacta.

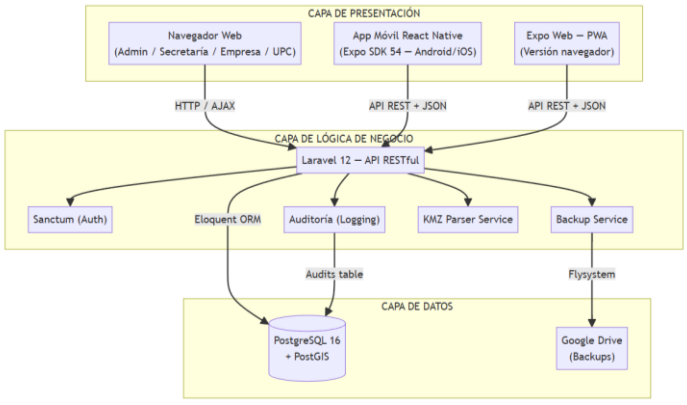
2. DIAGRAMAS DE MODELAMIENTO

En este capítulo se presentan los diagramas que describen la estructura y el comportamiento del sistema. Cada diagrama incluye una descripción textual de su propósito y su relación con el sistema.

2.1 Diagrama de Arquitectura del Sistema

El siguiente diagrama representa la arquitectura general del sistema, ilustrando la separación en capas y la comunicación entre los componentes.

Figura 1
Diagrama de arquitectura general del sistema



Nota. Elaboración propia.

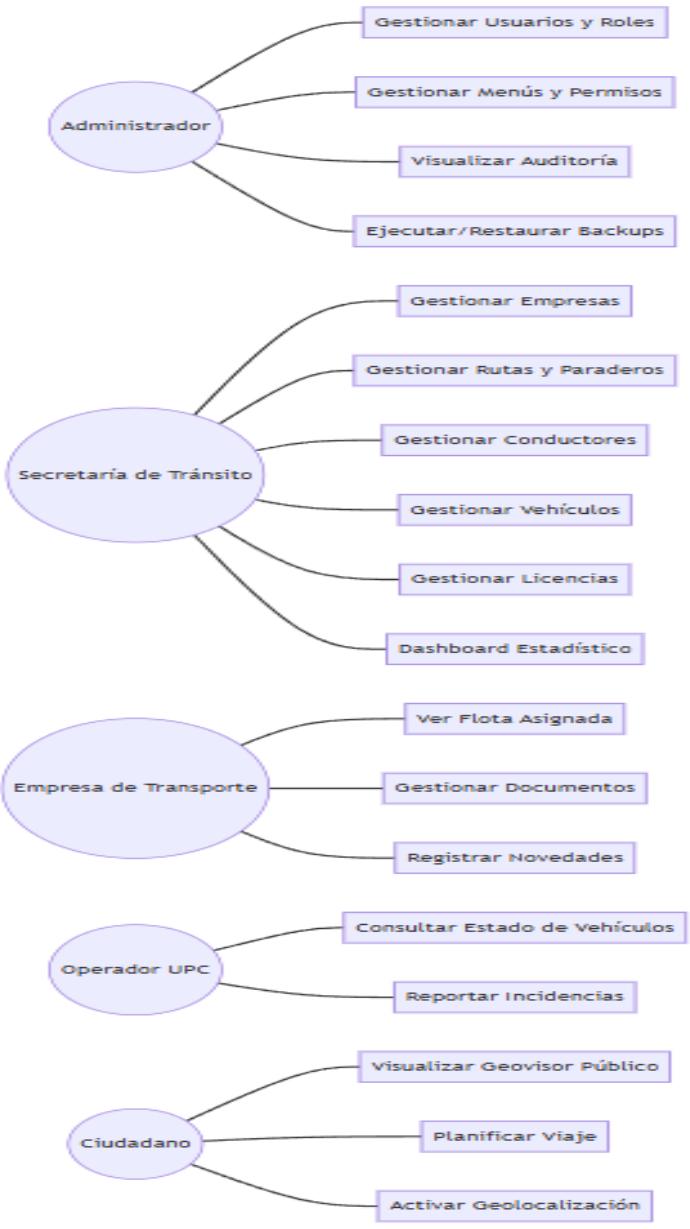
El diagrama evidencia tres capas claramente diferenciadas: la capa de presentación comprende tres clientes distintos — el panel web administrativo (Blade/JS), la aplicación móvil nativa (React Native con Expo SDK 54) y su versión web progresiva (Expo Web/PWA); la capa de lógica de negocio centraliza toda la operación en una API RESTful construida en Laravel 12; y la capa de datos persiste la información en PostgreSQL 16 con PostGIS, complementada con Google Drive como almacenamiento externo de respaldos.

2.2 Diagrama de Casos de Uso

El siguiente diagrama describe los casos de uso del sistema organizados por actor. Su propósito es identificar las funcionalidades principales y los roles que interactúan con cada una de ellas.

Figura 2

Diagrama de casos de uso del sistema por actor



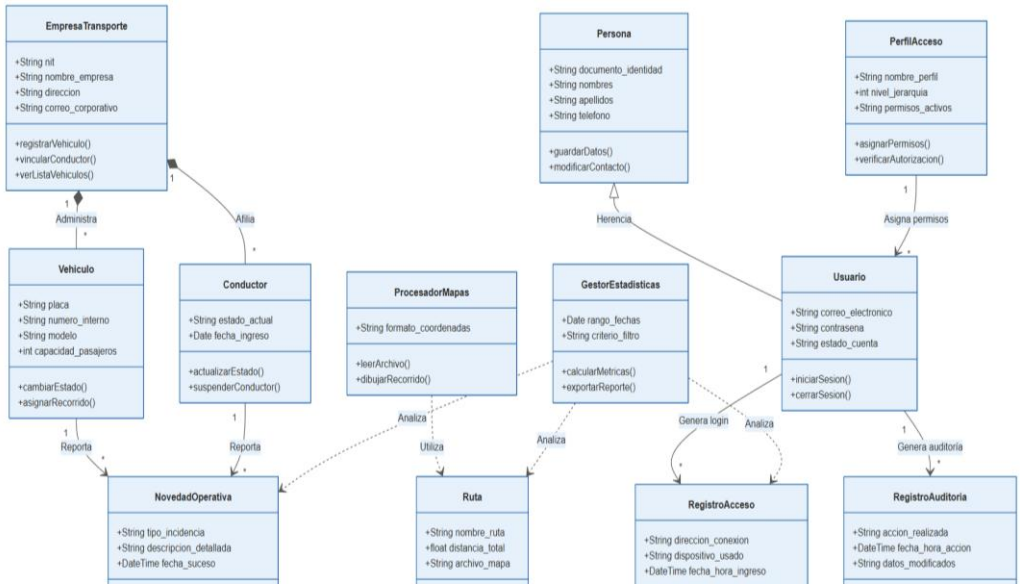
Nota. Elaboración propia.

2.3 Diagrama de Clases

El diagrama de clases representa los modelos Eloquent del sistema y sus relaciones. El sistema cuenta con 32 modelos Eloquent organizados en los siguientes grupos funcionales:

Figura 3

Diagrama de clases del sistema (32 modelos Eloquent)



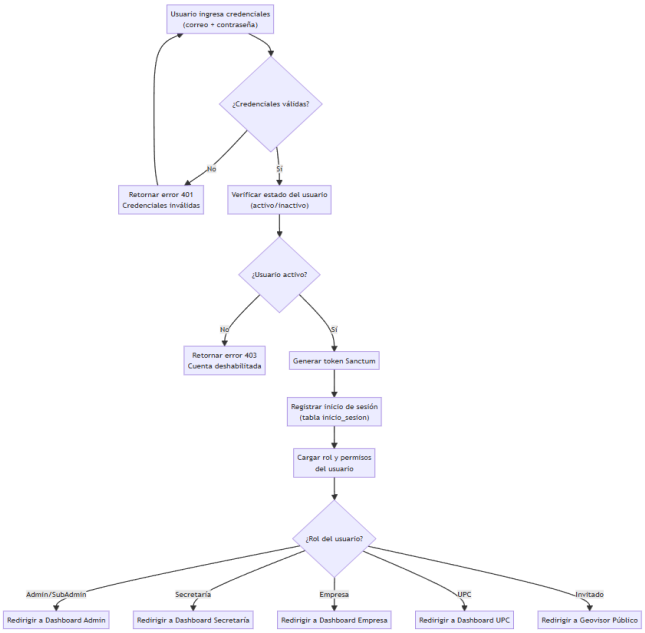
Nota. Elaboración propia.

2.4 Diagrama de Actividades — Flujo de Autenticación

El siguiente diagrama describe el flujo de actividades que sigue un usuario al autenticarse en el sistema, documentando el mecanismo de autenticación basado en tokens Sanctum implementado en la API.

Figura 4

Diagrama de actividades — Flujo de autenticación y redirección por rol



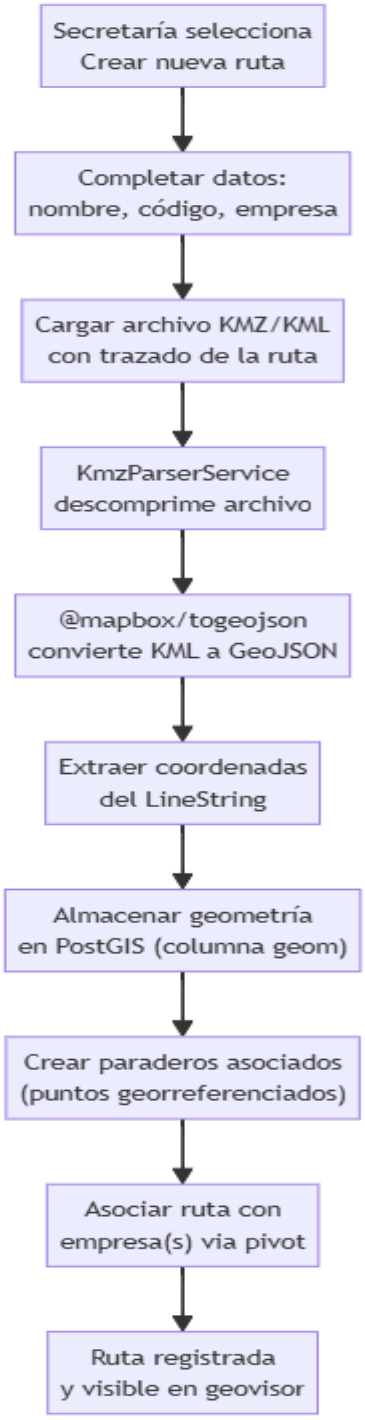
Nota. Elaboración propia.

2.5 Diagrama de Actividades — Gestión de Rutas

El siguiente diagrama describe el flujo de actividades para el registro de una nueva ruta de transporte con su trazado geoespacial, documentando el proceso de carga y procesamiento de archivos KMZ/KML.

Figura 5

Diagrama de actividades — Flujo de registro de ruta con datos geoespaciales



Nota. Elaboración propia.

3. ASPECTOS TÉCNICOS DEL DESARROLLO DEL SISTEMA

3.1 Arquitectura del Sistema

El sistema adopta una arquitectura cliente-servidor en capas (three-tier architecture), donde cada capa tiene responsabilidades claramente delimitadas y se comunica con las adyacentes a través de interfaces definidas. Esta decisión arquitectónica responde a la necesidad de separar las responsabilidades de presentación, procesamiento y almacenamiento, facilitando el mantenimiento independiente de cada componente.

A. Capa de Presentación

La capa de presentación corresponde al componente del sistema encargado de la interacción con el usuario final. Su función principal es permitir la visualización de la información y la captura de datos de manera clara, intuitiva y controlada.

Tecnologías utilizadas:

- Blade (motor de plantillas de Laravel) para la generación de vistas HTML dinámicas del lado del servidor.
- Tailwind CSS 4 con plugin Vite (@tailwindcss/vite) para el diseño visual responsivo.
- JavaScript modular (ES Modules) para la interactividad del lado del cliente.
- Leaflet.js 1.9.4 para los mapas interactivos del geovisor, con extensiones para polilíneas decoradas.
- Axios para las peticiones HTTP asíncronas hacia la API RESTful.
- Swiper.js para carruseles interactivos en la landing page.

Tipo de interfaz

El sistema cuenta con dos tipos de interfaz:

- **Interfaz web administrativa:** accesible desde navegadores de escritorio y tabletas. Presenta un dashboard personalizado según el rol del usuario (Admin, Secretaria, Empresa, UPC), con tablas interactivas, formularios modales, gráficos estadísticos y gestión CRUD completa.
- **Interfaz móvil (React Native):** aplicación nativa desarrollada con React Native y Expo SDK 54, disponible para Android. Presenta el geovisor público con mapa interactivo, planificador de viajes, geolocalización nativa del dispositivo y panel de control de rutas/paraderos.

Principios de usabilidad aplicados:

- Diseño responsivo (mobile-first mediante Tailwind CSS).
- Feedback inmediato al usuario mediante alertas contextuales (éxito, error, confirmación).
- Navegación consistente mediante sidebar persistente y breadcrumbs.
- Formularios con validación en tiempo real (frontend) y validación de seguridad (backend).
- Accesibilidad básica: etiquetas ARIA, contraste de colores adecuado, navegación por teclado.

I. Relación de Directorios

Tabla 7
Estructura de directorios — Capa de presentación

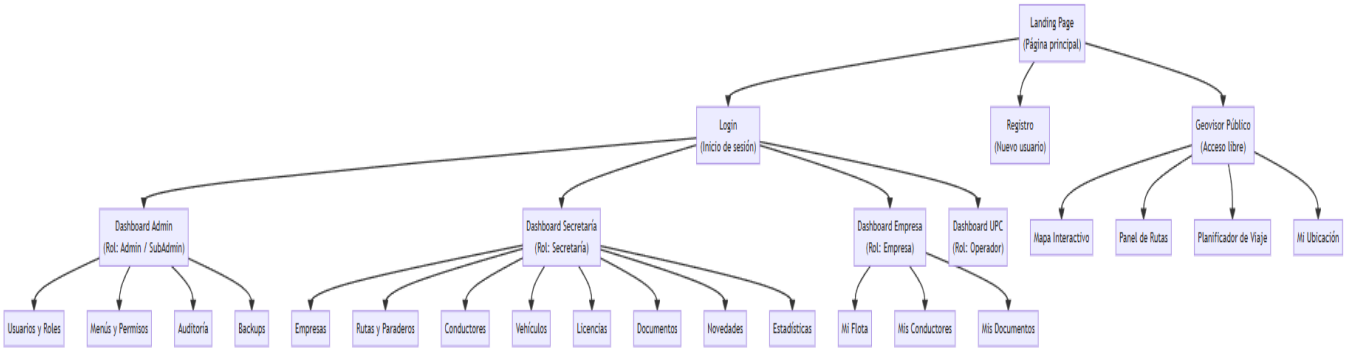
Directorio	Descripción
resources/views/	Contiene todas las plantillas Blade del sistema, organizadas por modulo funcional.
resources/views/auth/	Vistas de autenticación: login.blade.php y register.blade.php.
resources/views/dashboard/	Dashboards principales: admin.blade.php, secretaria.blade.php, empresa.blade.php, upc.blade.php.
resources/views/geovisor/	Vista del geovisor interactivo: geovisor_vite.blade.php.
resources/views/layouts/	Layout base de la aplicación: landing.blade.php.
resources/css/	Hojas de estilo CSS: app.css (global), dashboard.css (41 KB), geovisor.css (87 KB).
resources/css/modules/	Estilos modularizados por componente del dashboard.
resources/js/	Archivos JavaScript principales y módulos de la aplicación.
resources/js/modules/	Módulos JS organizados por área funcional (admin, empresa, secretaria, UPC, app).
resources/js/services/	Servicio MapCore.js (40 KB): motor de mapas del Geovisor.
resources/images/	Recursos de imagen utilizados en las vistas (logos, iconos, fondos).
public/images/	Imágenes publicas accesibles directamente desde el navegador.
public/maps/	Archivos KMZ/KML de rutas cargados para procesamiento.
public/build/	Assets compilados por Vite para producción (CSS y JS minificados).

Nota. Elaboración propia.

II. Mapa de Navegación

El mapa de navegación describe la organización lógica de las pantallas y la forma en que el usuario se desplaza entre ellas. El sistema presenta un flujo de navegación condicionado por el rol del usuario autenticado.

Figura 6
Mapa de navegación del sistema por rol de usuario



Nota. Elaboración propia.

Desde la pantalla principal (Landing Page), el usuario puede acceder al formulario de inicio de sesión o directamente al geovisor público (sin autenticación). Una vez autenticado, el sistema redirige automáticamente al dashboard correspondiente según el rol asignado. Todas las operaciones se realizan mediante modales y formularios dentro del mismo dashboard, sin navegación entre paginas independientes (Single Page Application parcial).

III. Archivos de Configuración (Frontend)

Tabla 8
Archivos de configuración relevantes — Frontend

Archivo	Propósito	Configuración principal
vite.config.js	Configuración del bundler Vite	Define los puntos de entrada (resources/css/app.css y resources/js/app.js), habilita el hot reload de Laravel y registra los plugins de Tailwind CSS y ESLint.
tailwind.config.js	Configuración de Tailwind CSS	Define los archivos a escanear para la purga de clases no utilizadas, colores personalizados y plugins extendidos.
eslint.config.js	Reglas de calidad de código JavaScript	Configura reglas de linting para JS, CSS, JSON y Markdown.
package.json	Manifiesto del proyecto frontend	Define scripts de ejecución (dev y build) y lista las dependencias.

Nota. Elaboración propia.

B. Capa de Lógica de Negocio

La capa de lógica de negocio está construida sobre el framework Laravel 12 con PHP 8.2. Su función es centralizar todas las reglas de negocio, validaciones, transformaciones de datos y la orquestación de servicios del sistema. Esta capa expone su funcionalidad a la capa de presentación a través de una API RESTful documentada con Swagger (OpenAPI 3.0).

Tabla 9
Estructura de directorios — Capa de lógica de negocio

Directorio	Descripción
app/Http/Controllers/	33 controladores que implementan la lógica CRUD y las operaciones especializadas de cada módulo.
app/Http/Middleware/	Middleware personalizado: CheckRole.php, CheckDashboardAccess.php, ForceJsonResponse.php.
app/Http/Requests/	Form Requests para validación centralizada de datos entrantes.
app/Http/Resources/	API Resources para la transformación estandarizada de respuestas JSON.
app/Models/	32 modelos Eloquent que representan las entidades del dominio del negocio.
app/Enums/	Enumeraciones PHP 8.1+: RolesEnum (6 roles), Tablas (28 tablas), Genders, Acciones.

Directorio	Descripción
app/Services/	Servicios de negocio: KmzParserService.php para procesamiento de archivos geoespaciales KMZ/KML.
config/	15 archivos de configuración del framework y paquetes de terceros.
routes/api.php	Definición de todas las rutas de la API RESTful (20 KB, ~400 rutas agrupadas por recurso).
routes/web.php	Rutas web para vistas Blade (landing, auth, dashboards, geovisor).

Nota. Elaboración propia.

I. Roles del sistema

El sistema implementa un esquema de autorización basado en seis roles, definidos mediante la enumeracion RolesEnum:

Tabla 10
Roles del sistema

ID	Nombre interno	Descripción funcional	Dashboard asignado
1	ADMIN	Super Administrador con acceso total al sistema	admin.blade.php
2	SECRETARIA	Funcionario de la Secretaria de Transito de Girardot	secretaria.blade.php
3	EMPRESA	Representante de una empresa de transporte	empresa.blade.php
4	OPERADOR	Operador de UPC en terreno	upc.blade.php
5	INVITADO	Ciudadano / Transeunte (acceso solo al geovisor)	geovisor_vite.blade.php
6	SUBADMIN	Sub-Administrador con permisos delegados	admin.blade.php

Nota. Elaboración propia.

II. Archivos de configuración (Backend)

Tabla 11
Archivos de configuración relevantes — Backend

Archivo	Propósito
config/app.php	Configuración general: nombre, zona horaria, locales, proveedores y aliases.
config/database.php	Configuración de las conexiones de base de datos (PostgreSQL como driver principal).
config/sanctum.php	Configuración de autenticación API: dominios stateful, TTL de tokens, middleware.
config/cors.php	Configuración CORS: orígenes permitidos, métodos y cabeceras autorizadas.

Archivo	Propósito
config/audit.php	Configuración del sistema de auditoria: modelos auditables, eventos capturados.
config/backup.php	Configuración de backups: fuentes, destinos (Google Drive), compresión y retención.
config/filesystems.php	Configuración de discos de almacenamiento: local, publico y Google Drive.
config/l5-swagger.php	Configuración de la documentación automática de la API con Swagger/OpenAPI.
.env	Variables de entorno sensibles (no versionado). Contiene credenciales de BD y tokens OAuth.

Nota. Elaboración propia.

C. Capa de Base de Datos

La capa de base de datos es responsable del almacenamiento, organización y recuperación de la información del sistema de manera estructurada y segura. El sistema utiliza PostgreSQL 16 como motor de base de datos relacional, complementado con la extensión PostGIS para el almacenamiento y procesamiento de datos geoespaciales.

La elección de PostgreSQL se fundamenta en:

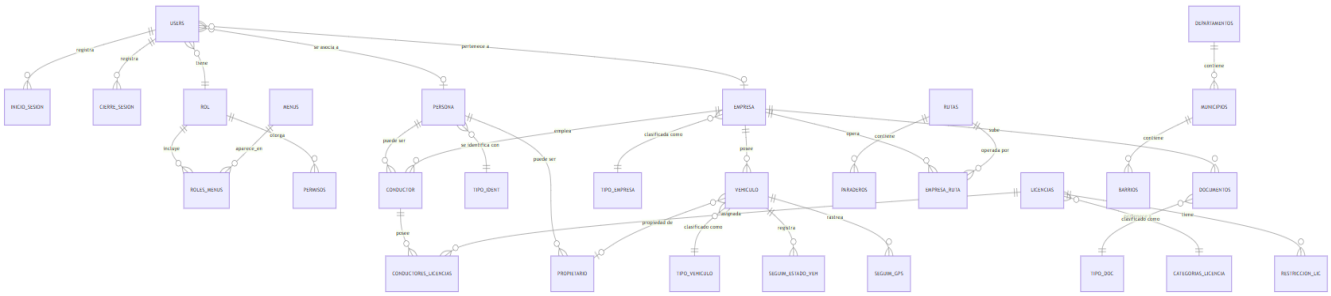
- Soporte nativo de tipos de datos geoespaciales mediante PostGIS, lo que permite almacenar y consultar geometrías (LineString para rutas, Point para paraderos) sin necesidad de extensiones externas.
- Cumplimiento estricto de ACID para garantizar la integridad transaccional de las operaciones de flota.
- Tipado fuerte y validación de datos a nivel de motor, complementando la validación de Laravel.
- Capacidades avanzadas de indexación (B-tree, GiST, GIN) para optimización de consultas geoespaciales.

La estructura de la base de datos se gestiona mediante 51 archivos de migración de Laravel, organizados cronológicamente y ejecutables mediante php artisan migrate. Las políticas de integridad incluyen llaves foráneas con restricciones ON DELETE CASCADE, campos timestamps automáticos en todas las tablas, soft deletes en entidades criticas e índices únicos en campos de identificación.

I. Diagrama Entidad-Relación

A continuación, se presenta el diagrama entidad-relación (E-R) del sistema, organizado por dominios funcionales:

Figura 7
Diagrama entidad-relación del sistema



Nota. Elaboración propia.

Las entidades principales del sistema y sus relaciones son las siguientes: la entidad Users se relaciona con Persona (datos personales), Rol (autorización) y opcionalmente con Empresa. La entidad Empresa centraliza la relación con Vehículos, Conductores, Documentos y Rutas (mediante la tabla pivot empresa_ruta). La entidad Conductor se vincula con Persona y posee Licencias a través de la tabla pivot conductores_licencias. La entidad Ruta contiene Paraderos georreferenciados y almacena su trazado como geometría PostGIS.

II. Diccionario de Datos

A continuación, se documentan las tablas y campos principales del sistema, organizadas por dominio funcional.

Tabla 12
Diccionario de datos — Tabla users

Campo	Tipo	Descripción	Restricciones
id	BIGINT	Identificador único del usuario	PK, auto-increment
name	VARCHAR (255)	Nombre de usuario para login	NOT NULL, UNIQUE
email	VARCHAR (255)	Correo electrónico	NOT NULL, UNIQUE
password	VARCHAR (255)	Contraseña hasheada (bcrypt, 12 rondas)	NOT NULL
rol_id	BIGINT	Rol asignado al usuario	FK -> rol.id
persona_id	BIGINT	Persona asociada	FK -> personas.id, NULLABLE
estado	VARCHAR (20)	Estado del usuario (activo/inactivo)	DEFAULT 'activo'
created_at	TIMESTAMP	Fecha de creación	Automático
updated_at	TIMESTAMP	Fecha de última modificación	Automático

Nota. Elaboración propia.

Tabla 13
Diccionario de datos — Tabla empresas

Campo	Tipo	Descripción	Restricciones
id	BIGINT	Identificador único	PK, auto-increment
nombre	VARCHAR (255)	Razón social de la empresa	NOT NULL
nit	VARCHAR (50)	Numero de Identificación Tributaria	UNIQUE
dirección	VARCHAR (255)	Dirección de la sede principal	NULLABLE
teléfono	VARCHAR (20)	Teléfono de contacto	NULLABLE
tipo_empresa_id	BIGINT	Clasificación de la empresa	FK -> tipo_empresa.id
estado	VARCHAR (20)	Estado (activa/inactiva/suspendida)	DEFAULT 'activa'

Nota. Elaboración propia.

Tabla 14
Diccionario de datos — Tabla rutas

Campo	Tipo	Descripción	Restricciones
id	BIGINT	Identificador único	PK, auto-increment
nombre	VARCHAR (255)	Nombre descriptivo de la ruta	NOT NULL
código	VARCHAR (50)	Código alfanumérico de la ruta	UNIQUE
color	VARCHAR (7)	Color hexadecimal para visualización	DEFAULT '#3388ff'
geom	GEOMETRY(LineString)	Trazado geoespacial de la ruta (PostGIS)	NULLABLE
estado	VARCHAR (20)	Estado (activa/inactiva)	DEFAULT 'activa'

Nota. Elaboración propia.

Tabla 15
Diccionario de datos — Tabla paraderos

Campo	Tipo	Descripción	Restricciones
id	BIGINT	Identificador único	PK, auto-increment
nombre	VARCHAR (255)	Nombre del paradero	NOT NULL
latitud	DECIMAL (10,7)	Coordenada de latitud	NOT NULL
longitud	DECIMAL (10,7)	Coordenada de longitud	NOT NULL
ruta_id	BIGINT	Ruta a la que pertenece	FK -> rutas.id, ON DELETE CASCADE
orden	INTEGER	Posición secuencial en la ruta	NOT NULL
tipo	VARCHAR (20)	Tipo: oficial / informal	DEFAULT 'oficial'

Nota. Elaboración propia.

Tabla 16
Diccionario de datos — Tabla vehículo

Campo	Tipo	Descripción	Restricciones
id	BIGINT	Identificador único	PK, auto-increment
placa	VARCHAR (10)	Placa del vehículo	NOT NULL, UNIQUE
marca	VARCHAR (100)	Marca del vehículo	NOT NULL
modelo	VARCHAR (100)	Modelo del vehículo	NOT NULL
anio	INTEGER	Año de fabricación	NOT NULL
tipo_vehiculo_id	BIGINT	Tipo de vehículo	FK -> tipo_vehiculo.id
empresa_id	BIGINT	Empresa propietaria/operadora	FK -> empresas.id
propietario_id	BIGINT	Propietario del vehículo	FK -> propietarios.id, NULLABLE
estado	VARCHAR (20)	Estado (activo/inactivo/fuera de servicio)	DEFAULT 'activo'
matricula	VARCHAR (255)	Ruta al documento de matricula	NULLABLE

Nota. Elaboración propia.

Tabla 17
Diccionario de datos — Tabla audits

Campo	Tipo	Descripción	Restricciones
id	BIGINT	Identificador del registro de auditoria	PK, auto-increment
user_type	VARCHAR (255)	Tipo de modelo del usuario	NULLABLE
user_id	BIGINT	ID del usuario que realizo la acción	NULLABLE
event	VARCHAR (255)	Tipo de evento (created/updated/deleted)	NOT NULL
auditable_type	VARCHAR (255)	Modelo afectado	NOT NULL
auditable_id	BIGINT	ID de la entidad afectada	NOT NULL
old_values	JSON	Valores anteriores a la modificación	NULLABLE
new_values	JSON	Valores nuevos tras la modificación	NULLABLE
url	TEXT	URL desde donde se ejecutó la acción	NULLABLE
ip_address	VARCHAR (45)	Dirección IP del cliente	NULLABLE
user_agent	TEXT	User-Agent del navegador	NULLABLE
created_at	TIMESTAMP	Fecha y hora de la acción	Automático

Nota. Elaboración propia.

3.2 Aplicación Movil (React Native + Expo)

Adicionalmente a la plataforma web administrativa, el sistema incorpora una aplicación móvil multiplataforma desarrollada con React Native y el framework de desarrollo Expo SDK 54. Esta aplicación está orientada al ciudadano y permite acceder al geovisor de transporte público desde dispositivos Android con una experiencia completamente nativa: acceso directo a la geolocalización del dispositivo, notificaciones push, navegación mediante gestos táctiles y rendimiento optimizado mediante componentes nativos.

La elección de React Native con Expo como tecnología de desarrollo móvil responde a las siguientes razones técnicas:

- **Código único multiplataforma:** un único código base en JavaScript/React genera la aplicación para Android y Web, reduciendo el tiempo de desarrollo a un tercio respecto al desarrollo nativo por separado.
- **Ecosistema Expo:** Expo SDK 54 proporciona acceso simplificado a las APIs nativas del dispositivo (cámara, GPS, sensores, almacenamiento) sin necesidad de escribir código nativo.
- **Hot Reload:** permite ver los cambios en tiempo real durante el desarrollo sin recompilar la aplicación completa.
- **Over-The-Air Updates (OTA):** mediante EAS Update, se envían actualizaciones de JavaScript directamente a los dispositivos sin pasar por Google Play o App Store.
- **Consumo de la misma API:** la app móvil consume la misma API RESTful de Laravel que utiliza el panel web, garantizando consistencia de datos entre todas las plataformas.

Tabla 18
Stack tecnológico de la aplicación móvil

Componente	Tecnología	Versión	Propósito
Framework base	React Native	0.76.x	Renderizado de componentes nativos en Android
Plataforma de desarrollo	Expo SDK	54	Toolchain de desarrollo, build y distribución
Lenguaje	JavaScript (ES2022+)	-	Logica de la aplicación y componentes React
Navegación	React Navigation	7.x	Stack Navigator y Drawer Navigator (menu lateral)
Mapas	react-native-maps	1.x	Renderizado de mapas nativos (leaflet)
WebView	react-native-webview	13.x	Integración de Leaflet.js para el geovisor avanzado
Geolocalización	expo-location	~18.x	Acceso nativo al GPS del dispositivo
Cliente HTTP	fetch / axios	-	Comunicación con la API RESTful de Laravel
Almacenamiento local	AsyncStorage	-	Persistencia del token de sesion y preferencias
Autenticación	Laravel Sanctum (tokens)	4.1	Autenticacion Bearer Token para peticiones API
Iconos	@expo/vector-icons	-	Iconografia nativa (MaterialIcons, Ionicons)

Nota. Elaboración propia.

Tabla 19
Estructura de directorios — App móvil React Native

Directorio / Archivo	Descripción
App.js	Punto de entrada principal. Configura los proveedores de navegación (NavigationContainer), SafeAreaProvider y el esquema de rutas.
app.json	Manifiesto de Expo: nombre de la app, slug, versión, orientación, splash screen, iconos (1024x1024), permisos de ubicación.
src/screens/	Pantallas de la aplicación: LoginScreen.js, RegisterScreen.js, GeovisorScreen.js, TripPlannerScreen.js, ProfileScreen.js.
src/components/	Componentes reutilizables: MapView.js, RouteCard.js, StopMarker.js, SearchBar.js, ToggleSwitch.js.
src/navigation/	Configuración de navegación: AppNavigator.js (Stack principal), DrawerNavigator.js (menú lateral del geovisor).
src/services/	Servicios de comunicación con la API: api.js (instancia Axios), authService.js, routeService.js, locationService.js.
src/hooks/	Custom hooks: useAuth.js, useLocation.js (geolocalización), useRoutes.js (rutas y paraderos).
src/context/	Contextos de React: AuthContext.js (proveedor global de autenticación con token Sanctum).
src/utils/	Utilidades: cálculo de distancias geodésicas (formula Haversine), formateo de coordenadas, constantes.
assets/	Recursos estáticos: iconos de la app, splash screen, logos institucionales.

Nota. Elaboración propia.

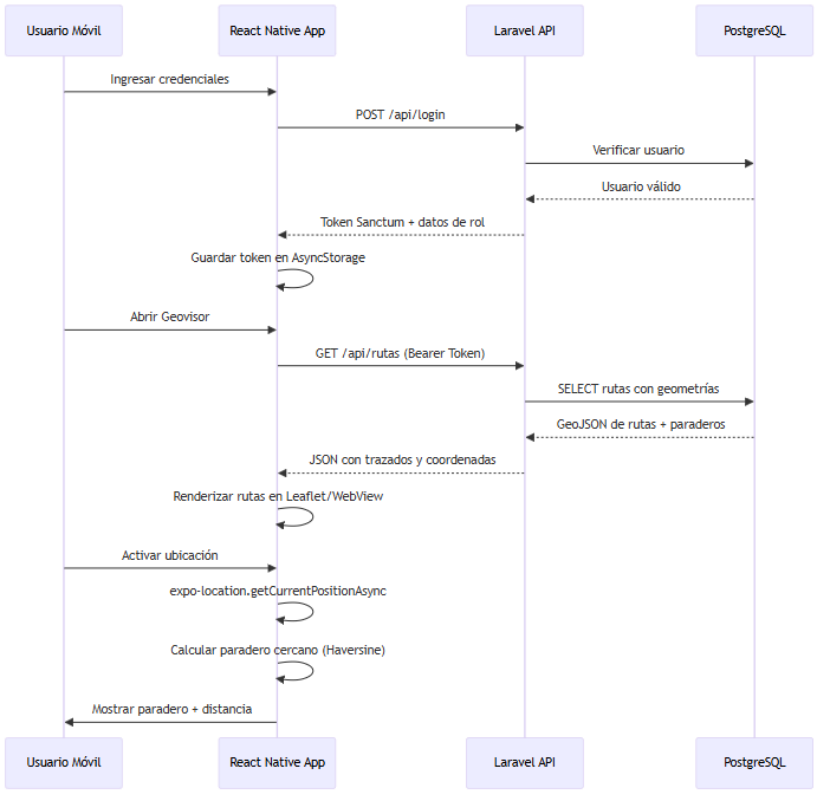
Módulos funcionales de la aplicación móvil:

- **Módulo de autenticación (US-AUTH-001 a US-AUTH-005):** inicio de sesión con credenciales contra la API Sanctum, registro de nuevo usuario, registro con Google (expo-auth-session), persistencia del token en AsyncStorage y cierre de sesión con revocación del token.
- **Módulo de geovisor (US-GEO-001 a US-GEO-012):** visualización del mapa interactivo con Leaflet.js mediante react-native-webview, activacion/desactivacion de rutas de transporte mediante toggles nativos, filtro de búsqueda de rutas y visualización de paraderos con marcadores georreferenciados.
- **Módulo de geolocalización (US-LOC-001 a US-LOC-003):** activación de la ubicación actual mediante expo-location, cálculo del paradero más cercano mediante la formula Haversine y visualización del marcador 'Tu estas aquí' en el mapa.
- **Módulo de planificación de viaje (US-TRIP-001 a US-TRIP-004):** selección de origen y destino sobre el mapa, cálculo de la ruta optima, estimación de tiempo de viaje y distancia de caminata.

Flujo de comunicación App Movil con la API:

Figura 8

Diagrama de secuencia — Comunicación entre la app móvil React Native y la API Laravel



Nota. Elaboración propia.

Tabla 20

Canales de distribución de la app móvil

Canal	Formato	Proceso	Ventaja
Expo Web (PWA)	HTML/JS/CSS estático	Se compila mediante npx expo export:web y los archivos generados se despliegan en el servidor.	Acceso inmediato sin descarga, funciona en cualquier navegador moderno, actualizaciones instantáneas.

Nota. Elaboración propia.

Instalación del entorno de desarrollo móvil:

```
# 1. Instalar Expo CLI globalmente
npm install -g expo-cli

# 2. Clonar el repositorio de la app movil
git clone https://github.com/[usuario]/ObservatorioMovil.git
cd ObservatorioMovil

# 3. Instalar dependencias
npm install
```

```
# 4. Configurar la URL de la API en src/services/api.js
# API_BASE_URL = 'http://[IP_SERVIDOR]:8000/api'

# 5. Iniciar el servidor de desarrollo Expo
npx expo start

# 6. Escanear el código QR con Expo Go (Android/iOS)
# o presionar 'w' para abrir en el navegador (Expo Web)
```

Tabla 21
React Native (Expo) vs Desarrollo nativo vs PWA pura

Criterio	React Native (Expo)	Nativo (Kotlin/Swift)	PWA pura
Código base	1 (JS/React)	2 (Kotlin + Swift)	1 (HTML/JS)
Acceso a GPS	Nativo completo (expo-location)	Nativo completo	Limitado (Geolocation API)
Rendimiento	Cercano a nativo (JSI Bridge)	Máximo rendimiento	Limitado por el navegador
Actualizaciones OTA	Si (EAS Update)	No	Si (automaticas)
Versión web simultanea	Si (Expo Web)	No (requiere desarrollo aparte)	Si
Notificaciones push	Nativas (expo-notifications)	Nativas	Limitado (Web Push API)
Tiempo de desarrollo	Medio	Alto (x2)	Bajo
Costo de publicación	$25 USD (Google) + $0 (Web)	$25 USD (Google)	$0

Nota. Elaboración propia.

4. INSTALACIÓN DEL SISTEMA Y SERVICIOS

4.1 Instalación del Sistema

Requisitos previos

Antes de iniciar la instalación, asegúrese de tener instalados los siguientes componentes:

- PHP 8.2 con extensiones: pgsql, mbstring, openssl, curl, zip, gd, xml, bcmath.
- Composer 2.6 o superior.
- Node.js 18 LTS o superior con npm 9+.
- PostgreSQL 16 o superior con extension PostGIS habilitada.
- Git 2.40 o superior.

Paso a paso — Instalación desde cero

Paso 1. Clonar el repositorio

```
git clone https://github.com/[usuario]/ObservatorioFront.git
cd ObservatorioFront/theftp
```

Paso 2. Instalar dependencias de PHP

composer install

Este comando descarga todas las dependencias listadas en composer.json y genera el autoloader optimizado. En caso de error, verificar que la extensión pgsql este habilitada en php.ini.

Paso 3. Instalar dependencias de JavaScript

npm install

Descarga las dependencias del frontend, incluyendo Vite, Tailwind CSS, Leaflet.js y demás librerías.

Paso 4. Configurar las variables de entorno

```
cp .env.example .env
php artisan key:generate
```

Editar el archivo .env con los datos de conexión a la base de datos PostgreSQL:

```
DB_CONNECTION=pgsql
DB_HOST=127.0.0.1
DB_PORT=5432
DB_DATABASE=backend_sm
DB_USERNAME=postgres
DB_PASSWORD=su_contrasena_segura
```

Paso 5. Crear la base de datos

```
-- Ejecutar en psql o pgAdmin:
CREATE DATABASE backend_sm;
\c backend_sm
CREATE EXTENSION IF NOT EXISTS postgis;
```

Paso 6. Ejecutar las migraciones

php artisan migrate

Este comando ejecuta las 51 migraciones en orden cronológico, creando todas las tablas, relaciones, índices y restricciones del sistema.

Paso 7. Sembrar datos iniciales (opcional)

php artisan db:seed

Ejecuta los seeders que crean datos de prueba: roles, permisos, menús, departamentos, municipios y un usuario administrador por defecto.

Paso 8. Crear el enlace simbólico de almacenamiento

php artisan storage:link

Crea un enlace simbólico desde public/storage a storage/app/public, necesario para servir los archivos subidos (documentos, KMZ).

Paso 9. Compilar assets del frontend

```
# Desarrollo (con hot reload):
npm run dev

# Produccion (assets optimizados):
npm run build
```

Paso 10. Iniciar el servidor de desarrollo

```
php artisan serve
```

El sistema estará disponible en http://localhost:8000. Para desarrollo completo con hot reload, ejecutar simultáneamente:

```
# Terminal 1:
php artisan serve

# Terminal 2:
npm run dev

# Terminal 3 (opcional, para colas):
php artisan queue:listen
```

Nota: El comando composer dev ejecuta los cuatro servicios simultáneamente usando concurrently: servidor PHP, cola de trabajos, monitor de logs y Vite dev server.

Errores comunes durante la instalación

Tabla 22
Errores comunes y soluciones

Error	Causa	Solución
could not find driver	Extension pgsql no habilitada en PHP	Descomentar extension=pgsql en php.ini y reiniciar PHP
SQLSTATE[08006] Connection refused	PostgreSQL no esta ejecutandose o el puerto es incorrecto	Verificar que el servicio PostgreSQL este activo y el puerto en .env coincida
Vite manifest not found	Los assets no han sido compilados	Ejecutar npm run build o npm run dev
storage/logs/laravel.log could not be opened	Permisos insuficientes en storage/	Ejecutar chmod -R 775 storage bootstrap/cache (Linux)
pg_dump not found	Ruta a pg_dump no configurada	Configurar PG_DUMP_PATH en .env
PostGIS extension not available	PostGIS no instalado en PostgreSQL	Instalar el paquete postgis y ejecutar CREATE EXTENSION postgis;

Nota. Elaboración propia.

5. REQUERIMIENTOS DE HARDWARE (OPERACIÓN)

Define los recursos de hardware necesarios para la operación del sistema en producción, considerando la carga de usuarios esperada en el contexto municipal de Girardot.

Tabla 23
Requerimientos de hardware para operación en producción

Componente	Especificación	Justificación
Procesador	4 vCPUs (x86_64)	Soporta concurrencia de PHP-FPM (4 workers), consultas PostgreSQL con PostGIS y procesamiento de archivos KMZ simultáneamente.
Memoria RAM	4 GB (mínimo), 8 GB (recomendado)	PHP-FPM requiere ~50 MB por proceso. PostgreSQL utiliza ~1 GB para cache de consultas. El geovisor genera JSON de rutas de hasta 5 MB.
Almacenamiento	40 GB SSD NVMe	Base de datos (~2 GB estimado), archivos subidos (~5 GB), respaldos temporales (~3 GB), sistema operativo y dependencias (~10 GB), margen de crecimiento (~20 GB).
Ancho de banda	100 Mbps simétrico	Servir tiles de mapas en cache, transferir archivos KMZ y atender múltiples sesiones concurrentes del geovisor.
Disponibilidad	99.5% uptime (SLA básico)	Equivale a ~1.83 días de downtime permitido al año, aceptable para un sistema municipal.

Nota. Elaboración propia.

Escalabilidad básica

El sistema ha sido diseñado para escalar de manera vertical en el corto plazo.

Las estrategias de escalabilidad incluyen:

- **Escalabilidad vertical:** incrementar vCPUs y RAM; soporta hasta 500 usuarios concurrentes con 8 vCPUs y 16 GB de RAM.
- **Caché de consultas:** implementar Redis o Memcached (CACHE_STORE=redis) para reducir la carga en PostgreSQL.
- **CDN para assets estáticos:** migrar tiles de mapas y assets compilados a CloudFlare o AWS CloudFront para aliviar el servidor.
- **Réplica de base de datos:** configurar una réplica de lectura en PostgreSQL para distribuir las consultas del geovisor.

Recomendaciones técnicas

- Usar hosting con servidores en Colombia o cercanos (DigitalOcean NYC, AWS São Paulo o Google Cloud southamerica-east1) para reducir la latencia.
- Configurar monitoreo con Uptime Robot o Grafana para detectar caídas del servicio.
- Implementar rotación de logs con logrotate para evitar consumo excesivo de almacenamiento.
- Mantener actualizadas las dependencias PHP y el sistema operativo con parches de seguridad mensuales.
- Verificar periódicamente los respaldos de Google Drive restaurándolos en un entorno de pruebas.

6. BIBLIOGRAFÍA

Normas técnicas

INSTITUTO COLOMBIANO DE NORMAS TÉCNICAS Y CERTIFICACIÓN (ICONTEC). *NTC 1486: Documentación — Presentación de tesis, trabajos de grado y otros trabajos de investigación.* Sexta actualización. Bogotá: ICONTEC, 2008.
INSTITUTO COLOMBIANO DE NORMAS TÉCNICAS Y CERTIFICACIÓN (ICONTEC). *NTC 5613: Referencias bibliográficas — Contenido, forma y estructura.* Bogotá: ICONTEC, 2008.

Documentación oficial de frameworks y lenguajes

LARAVEL LLC. *Laravel 12.x Documentation* [en línea]. 2026. Disponible en: https://laravel.com/docs/12.x
THE PHP GROUP. *PHP 8.2 Manual* [en línea]. 2024. Disponible en: https://www.php.net/manual/es/
THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. *PostgreSQL 16 Documentation* [en línea]. 2024. Disponible en: https://www.postgresql.org/docs/16/
POSTGIS PROJECT. *PostGIS 3.4 Documentation* [en línea]. 2024. Disponible en: https://postgis.net/documentation/

Libros de referencia

PRESSMAN, Roger S. *Ingeniería del Software: Un Enfoque Práctico.* 7.ª ed. México: McGraw-Hill, 2010. ISBN 978-607-15-0314-5.
SOMMERVILLE, Ian. *Ingeniería de Software.* 10.ª ed. México: Pearson, 2016. ISBN 978-607-32-3768-2.
FOWLER, Martin. *Patterns of Enterprise Application Architecture.* Boston: Addison-Wesley, 2002. ISBN 978-0-321-12742-6.

Frameworks y librerías frontend

EVAN YOU. *Vite 6 Documentation* [en línea]. 2025. Disponible en: https://vitejs.dev/guide/
TAILWIND LABS. *Tailwind CSS 4 Documentation* [en línea]. 2025. Disponible en: https://tailwindcss.com/docs
AGAFONKIN, Vladimir. *Leaflet.js — An Open-Source JavaScript Library for Mobile-Friendly Interactive Maps* [en línea]. 2024. Disponible en: https://leafletjs.com/reference.html

Paquetes de terceros

SPATIE. *Laravel Backup — A Modern PHP Backup Manager* [en línea]. 2024. Disponible en: https://spatie.be/docs/laravel-backup/v9
OWEN-IT. *Laravel Auditing — Record Changes to Eloquent Models* [en línea]. 2024. Disponible en: https://laravel-auditing.com
TAYLOR OTWELL. *Laravel Sanctum — Lightweight API Authentication* [en línea]. 2025. Disponible en: https://laravel.com/docs/12.x/sanctum

Estándares web

WORLD WIDE WEB CONSORTIUM (W3C). *Progressive Web Apps* [en línea]. 2024. Disponible en: https://www.w3.org/TR/appmanifest/
WORLD WIDE WEB CONSORTIUM (W3C). *Service Workers* [en línea]. 2024. Disponible en: https://www.w3.org/TR/service-workers/
OPEN GEOSPATIAL CONSORTIUM (OGC). *KML — OGC Standard* [en línea]. 2023. Disponible en: https://www.ogc.org/standard/kml/