

Arquitectura del Sistema

Modelo de Aplicación

Sistema de Información para el Cambio Climático del Ministerio
de Minas y Energía

GearsMap

25 de junio de 2026

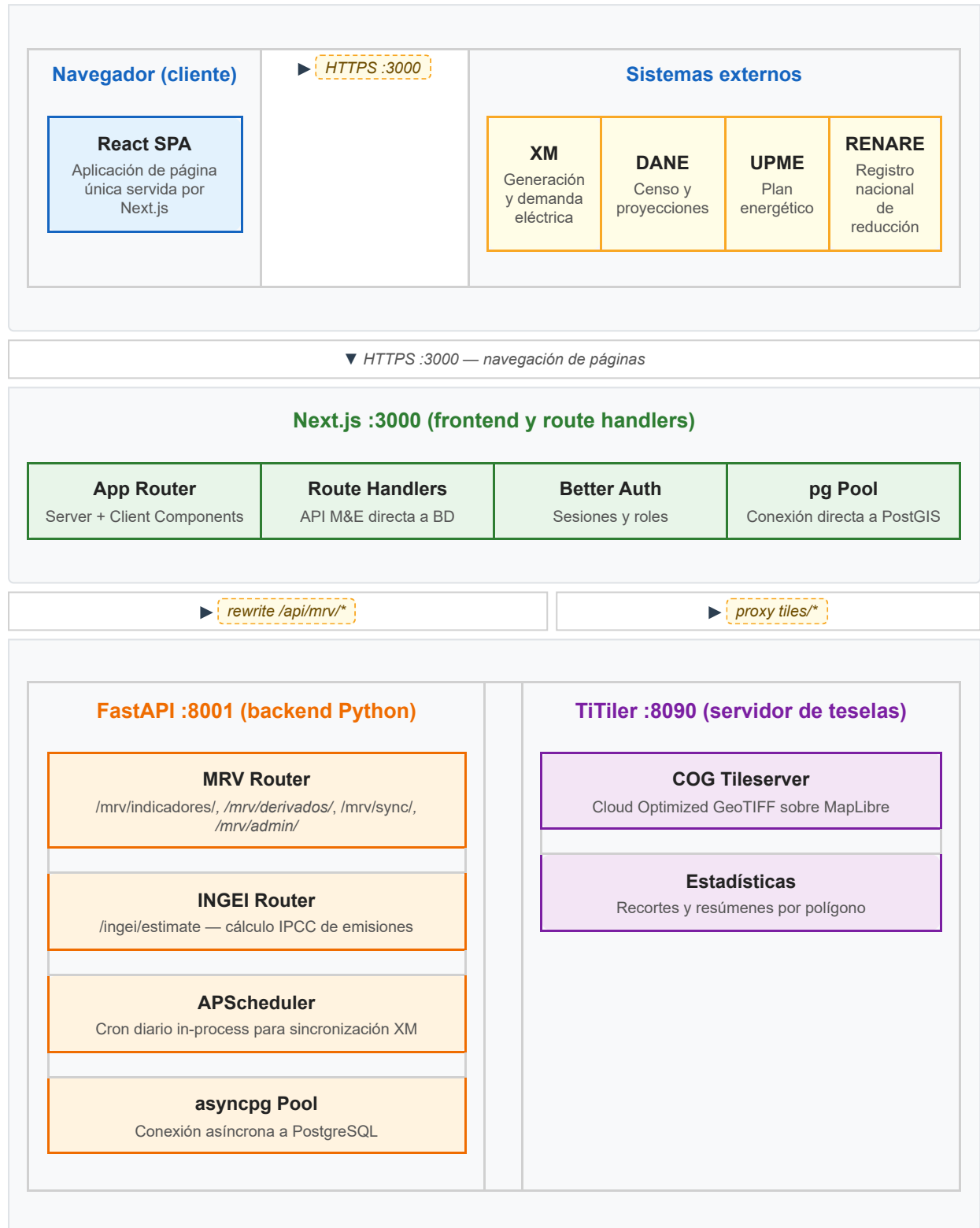
Versión 1.0

Tabla de contenido

1. Diagrama de despliegue	1
.....	
2. Diagrama de componentes	3
.....	
3. Flujo de solicitudes	5
.....	
4. Estrategia de comunicación entre componentes	6
.....	
5. Estilos arquitectónicos	7
.....	
6. Puertos y servicios	8
.....	
7. Consideraciones de seguridad	9
.....	

1. Diagrama de despliegue

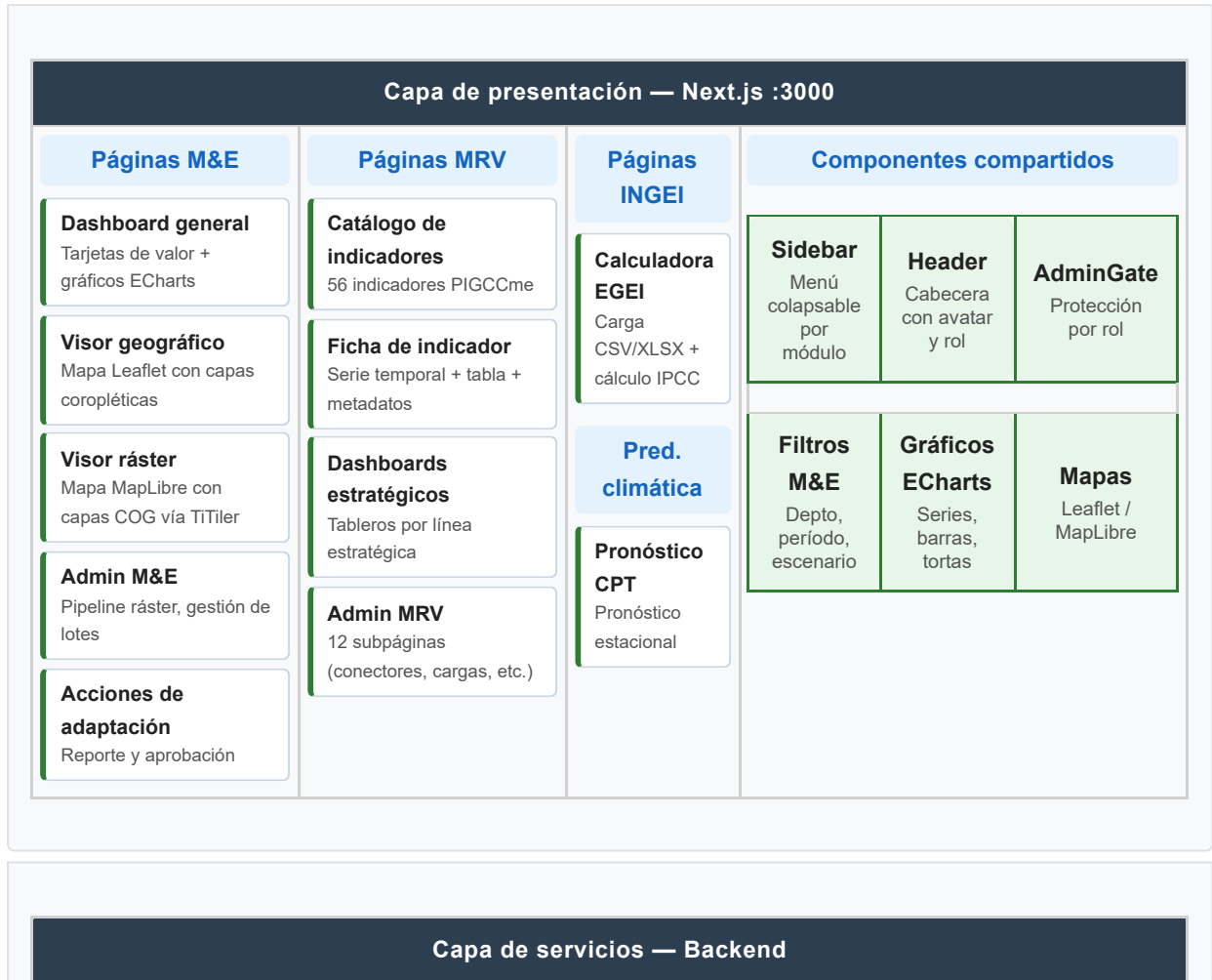
Figura 1: Diagrama de despliegue de la arquitectura SICMME

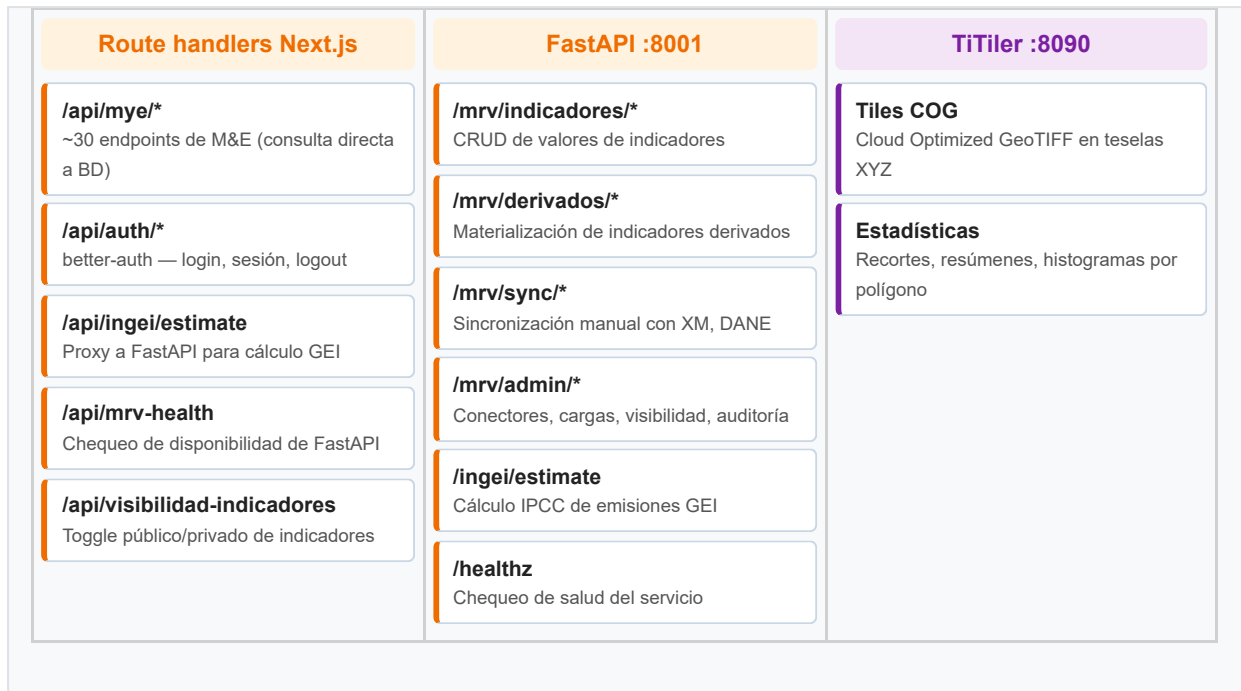




2. Diagrama de componentes

Figura 2: Diagrama de componentes por capas (presentación, servicios, datos, integración)





3. Flujo de solicitudes

Solicitud M&E (sin backend adicional)

```
Navegador → GET /cambio-climatico/mye/visor-geografico
→ Next.js App Router
→ Server Component (carga datos)
→ POST /api/mye/geometrias
→ Route Handler
→ pg Pool (directo)
→ PostGIS: SELECT * FROM public.municipio WHERE ...
← JSON geojson
← Response 200
← Página renderizada con mapa Leaflet
← HTML al navegador
```

Solicitud MRV (con backend FastAPI)

```
Navegador → GET /cambio-climatico/mitigacion/mrv/indicador/W
→ Next.js App Router
→ Server Component
→ fetch /api/mrv/indicadores/W/valores
→ next.config.ts rewrite
→ http://localhost:8001/mrv/indicadores/W/valores
→ FastAPI Router
→ asyncpg pool
→ PostgreSQL: SELECT * FROM mrv.hecho_indicador_entrada WHERE ...
← JSON valores
← Response 200
← Response 200
← JSON
← Página renderizada con gráficos
← HTML al navegador
```

4. Estrategia de comunicación entre componentes

Origen	Destino	Protocolo	Formato	Propósito
Navegador	Next.js	HTTPS	HTML + JSON	Carga de páginas y datos
Next.js (rewrite)	FastAPI	HTTP	JSON	Datos MRV en vivo
Next.js (proxy)	TiTiler	HTTP	PNG / JSON	Tiles ráster y estadísticas
Next.js	PostgreSQL	pg (TCP)	SQL / JSON	Consultas directas M&E
FastAPI	PostgreSQL	asyncpg (TCP)	SQL / JSON	Consultas MRV
FastAPI	XM/DANE/UPME	HTTPS	JSON / XML	Sincronización de datos

5. Estilos arquitectónicos

Aspecto	Decisión	Justificación
Patrón general	Backend-for-Frontend (BFF) con ruta directa a BD para M&E	MRV necesita Python por los conectores; M&E no
Frontend	Server Components + Client Components (App Router)	SEO, rendimiento, interactividad donde se necesita
API M&E	Route handlers Next.js con Pool pg directo	Simplicidad, menor latencia, sin servicio intermedio
API MRV	Rewrite a FastAPI dedicado	Aislamiento del cómputo pesado (pandas, fórmulas)
Autenticación	better-auth con sesiones en BD	Madurez, soporte para roles, sesiones persistentes
Mapas	Leaflet (vectorial) + MapLibre (ráster)	Leaflet para coroplético simple, MapLibre para COG
Procesamiento ráster	Subprocess Python (offline)	Operación bajo demanda del administrador
Sincronización MRV	APScheduler in-process	Simplicidad, sin dependencia externa de cola

6. Puertos y servicios

Servicio	Puerto	Tecnología	Dependencia
Next.js (frontend)	3000	node 20+	-
FastAPI (MRV + INGEI)	8001	uvicorn + python 3.11+	Next.js lo llama vía rewrite
TiTiler (ráster)	8090	python + Docker	Next.js lo llama vía proxy
PostgreSQL	5432	postgres 14+ / PostGIS	Externo (VPN MinEnergía)

7. Consideraciones de seguridad

Aspecto	Implementación
Autenticación	better-auth con sesiones JWT almacenadas en BD
Roles	Admin (todo), Analista (lectura + reportes), Público (solo lectura)
Protección admin	<code>AdminGate</code> componente + <code>assertAdmin()</code> para rutas destructivas
Sesiones	Expiración a 30 días, renovación en cada solicitud
Invitado	Cookie <code>sicmme_guest</code> sin persistencia en BD
Contraseñas	Hash bcrypt, nunca almacenadas en texto plano
API MRV	Solo accesible desde localhost (rewrite), no expuesta públicamente
Conectores	Configuración almacenada en BD, endpoints validados