

TankGo

Plataforma Inteligente de Precios de Carburantes

IKER ALVIS VELOSO

DESARROLLO DE APLICACIONES PARA LA WEB DE DATOS

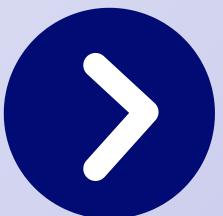


Objetivo del proyecto TankGo

Crear una plataforma modular, escalable y desplegada en la nube para consultar, gestionar y visualizar información de gasolineras en España.

Ofrecer al usuario:

- autenticación segura
- consulta en tiempo real de precios
- búsqueda por filtros
- mapa interactivo y sistema de favoritos





TankGo

TankGo

Plataforma Inteligente de Precios de Carburantes

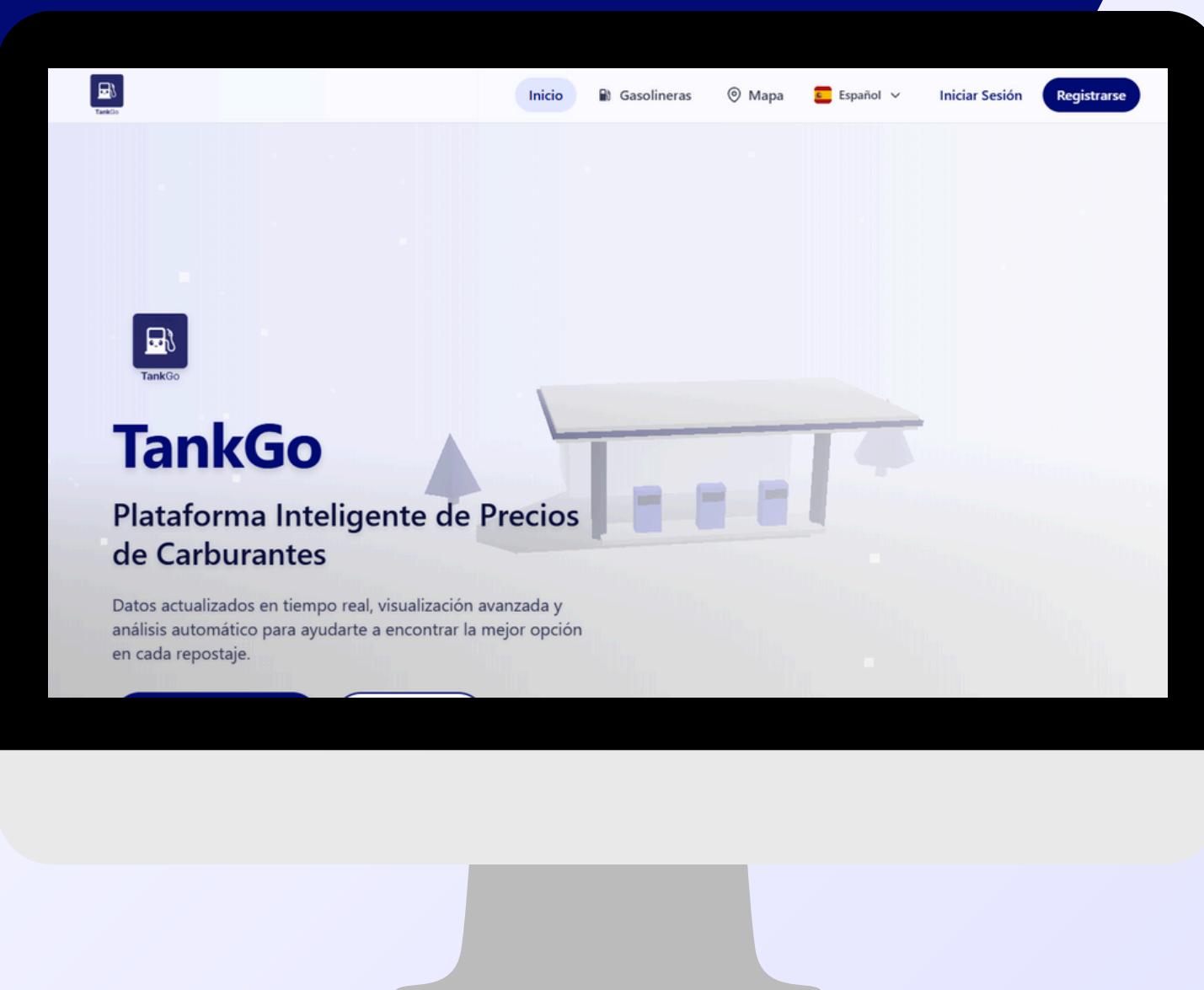
Datos actualizados en tiempo real, visualización avanzada y análisis automático para ayudarte a encontrar la mejor opción en cada repostaje.

Explorar Gasolineras →

Ver Mapa

Funcionalidades Principales

[LINK AL VIDEO](#)



Funcionalidades implementadas

Funcionalidades clave

Registro / Login con Google:

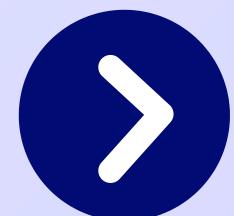
- Autenticación con OAuth 2.0 directamente desde el frontend.
- O de manera tradicional con usuario y contraseña.

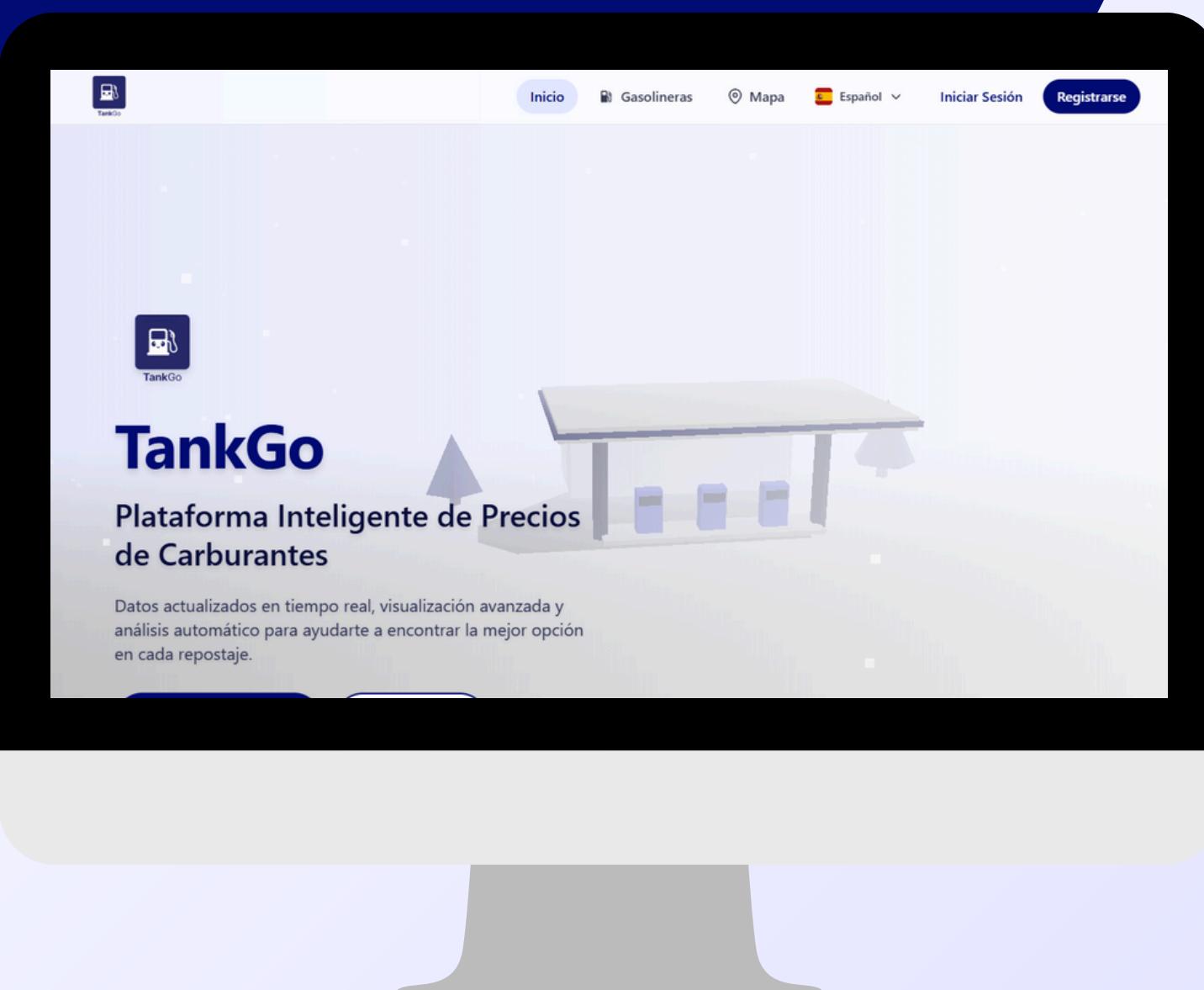
Gestión de usuarios y favoritos

- Microservicio dedicado (Node.js + PostgreSQL).
- El usuario puede guardar sus gasolineras favoritas.

Internacionalización de la página web

- Uso de i18n para traducir la página a diferentes idiomas.





Funcionalidades implementadas

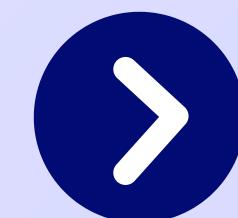
Funcionalidades clave

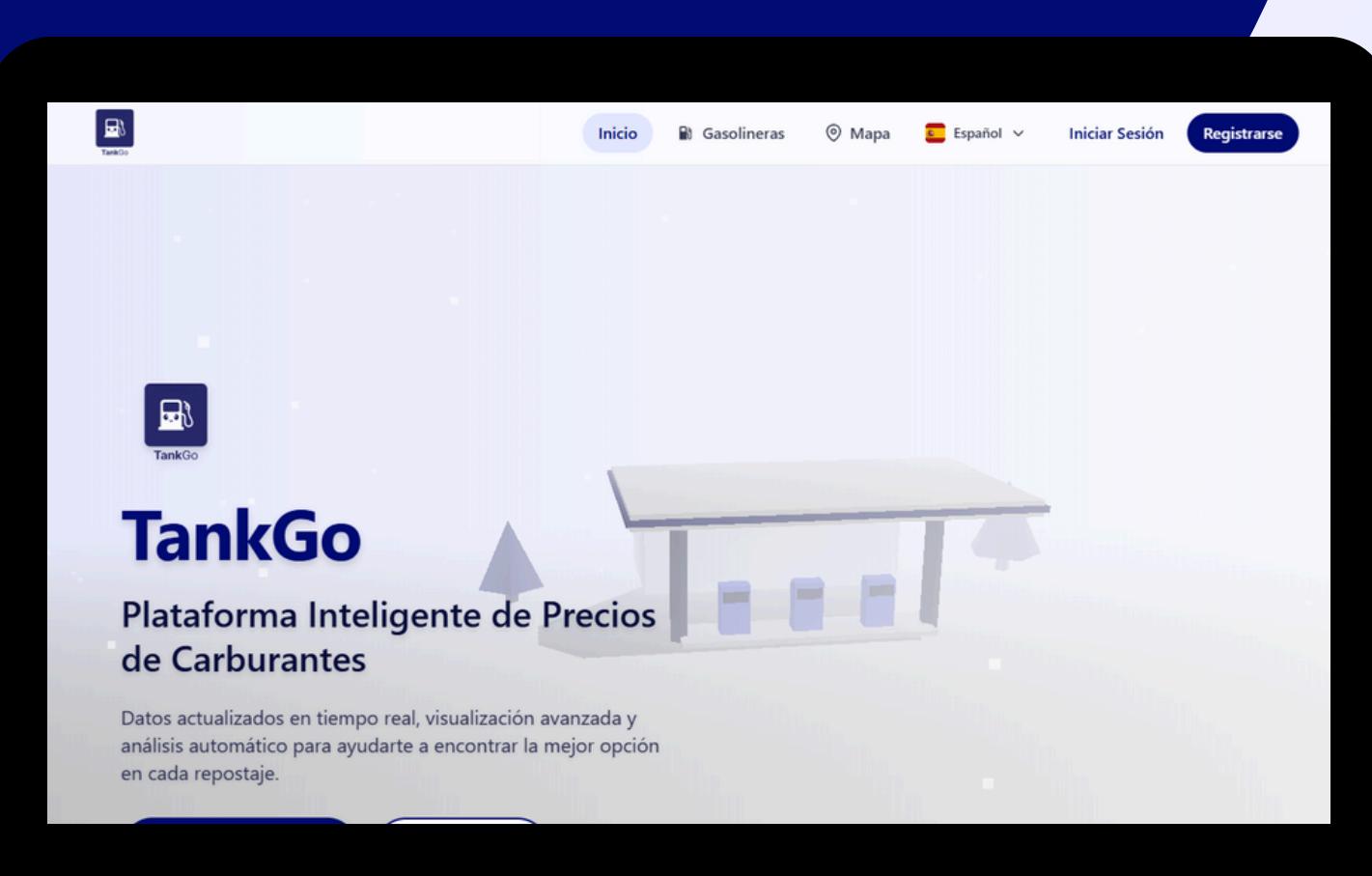
Listado y filtrado avanzado de gasolineras

- Filtros por provincia, municipio, marca, precios, tipo de combustible, etc.
- Autocompletado inteligente de localizaciones.
- Resultados ordenados según precio o cercanía.

Mapa interactivo

- Renderizado con Leaflet.js.
- Marcadores personalizados con el logo de la marca.
- Posibilidad de localizar gasolineras cercanas por geolocalización real del usuario.





Funcionalidades implementadas

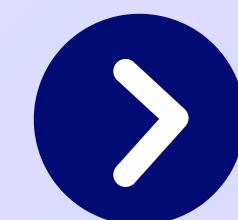
Funcionalidades clave

Sincronización automática con la API del Ministerio

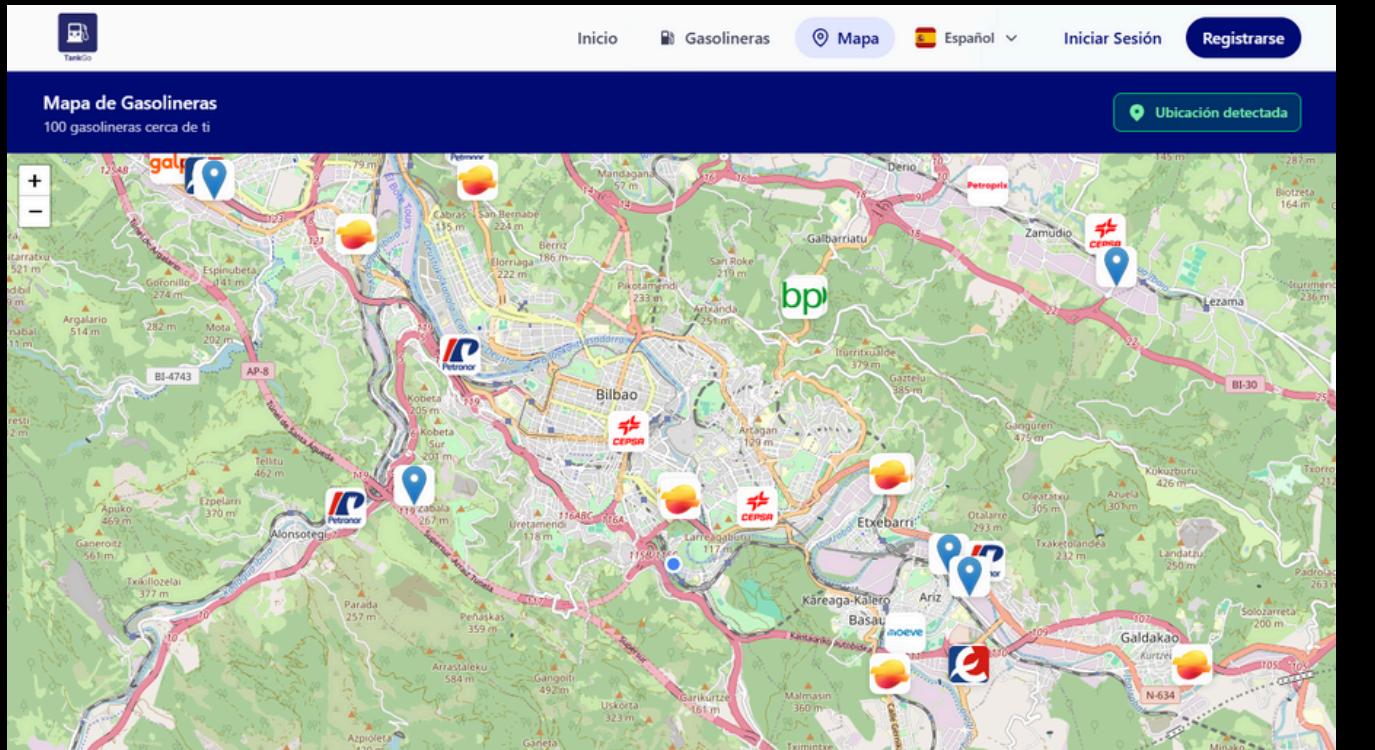
- El microservicio de gasolineras ejecuta sincronizaciones periódicas.
- Asegura que la app siempre trabaje con datos actuales.
- Los datos son todas las gasolineras de España, con los precios de los carburantes y su localización.

SPA moderna y responsive (React + TypeScript + TailwindCSS)

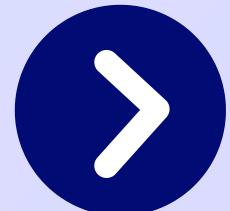
- Interfaz fluida sin recargas.
- Componentes reutilizables + hooks personalizados.
- Adaptado para móvil (uso en carretera) y escritorio.



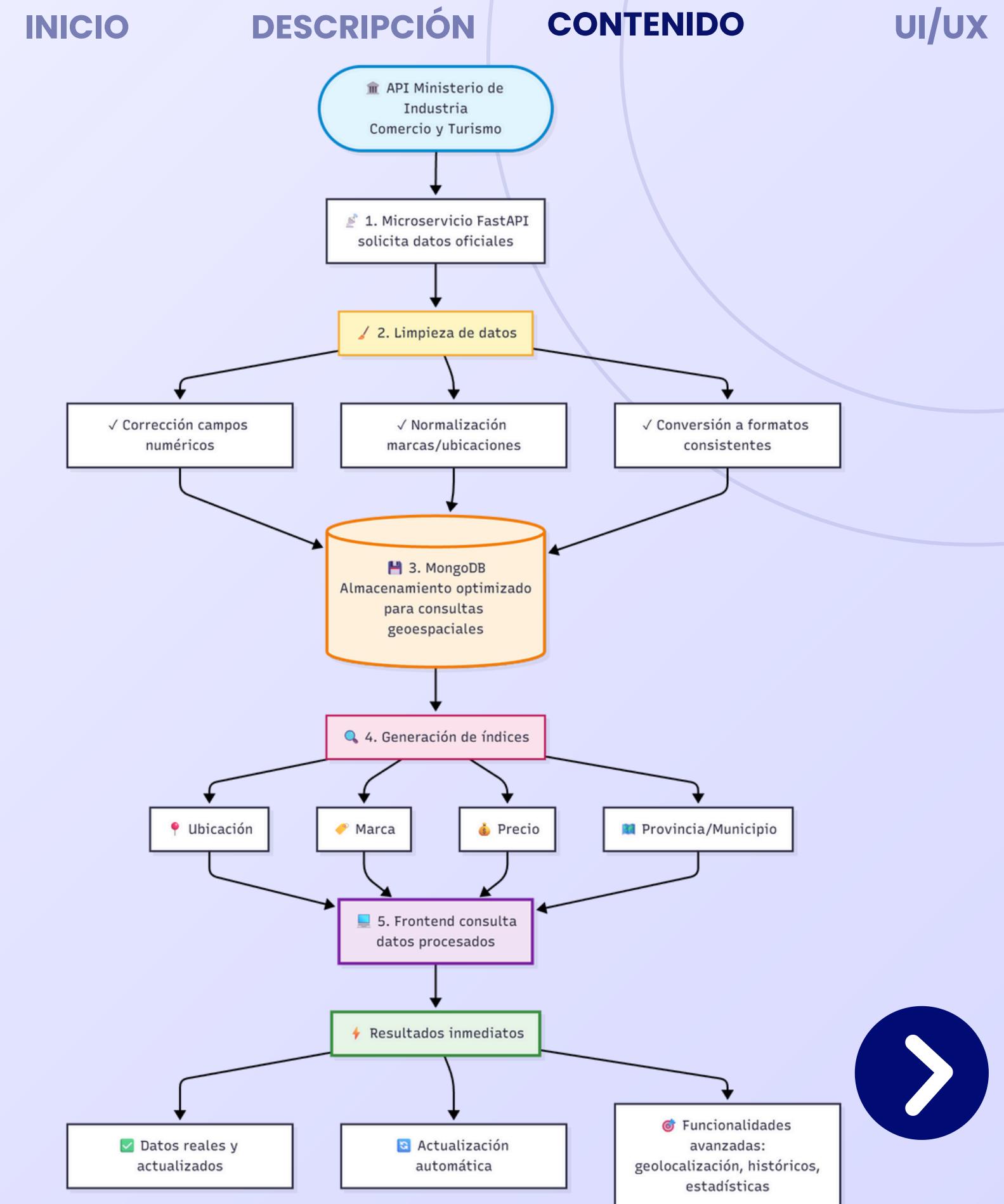
APIs Implementadas



- Usuarios Service (Node.js + PostgreSQL)
 - POST /register
 - POST /login
 - GET /favoritos
- Gasolineras Service (FastAPI + MongoDB)
 - GET /gasolineras
 - GET /gasolineras/:id
 - POST /gasolineras/sync
- API Gateway (Node.js)
 - Expone un único punto de entrada
 - Swagger UI
 - Balanceo básico / redirección a microservicios

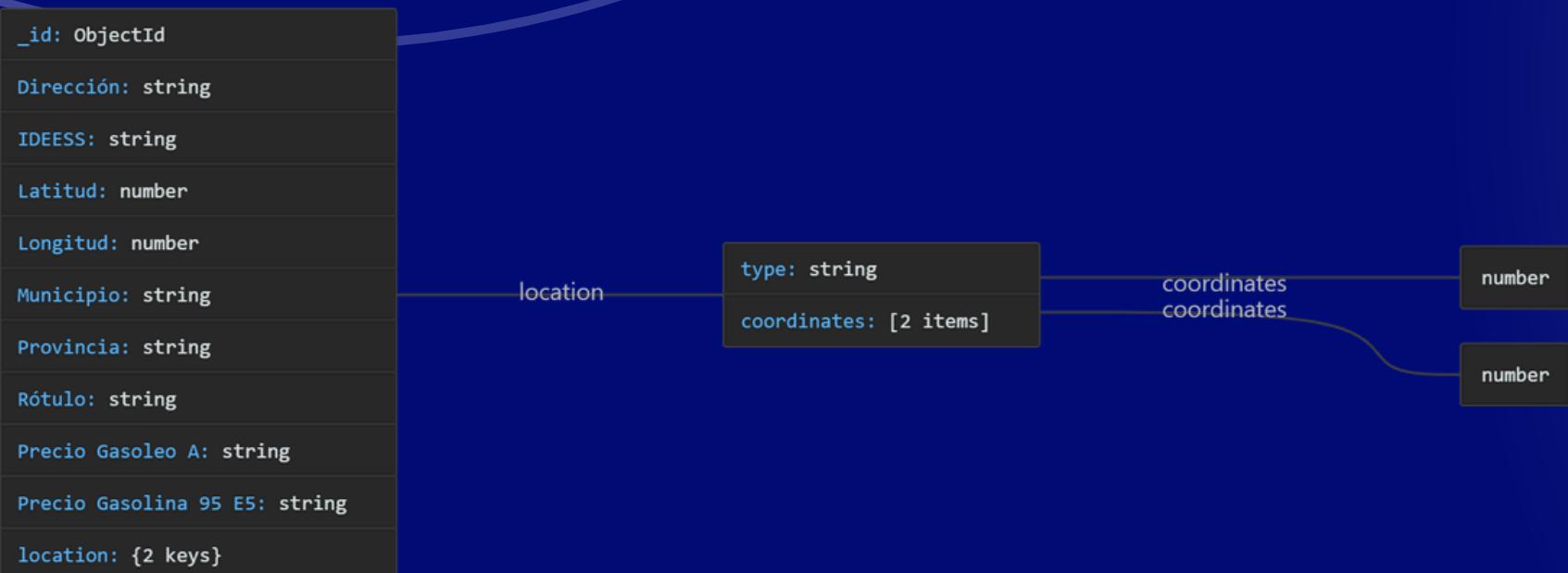


¿De dónde salen los datos?

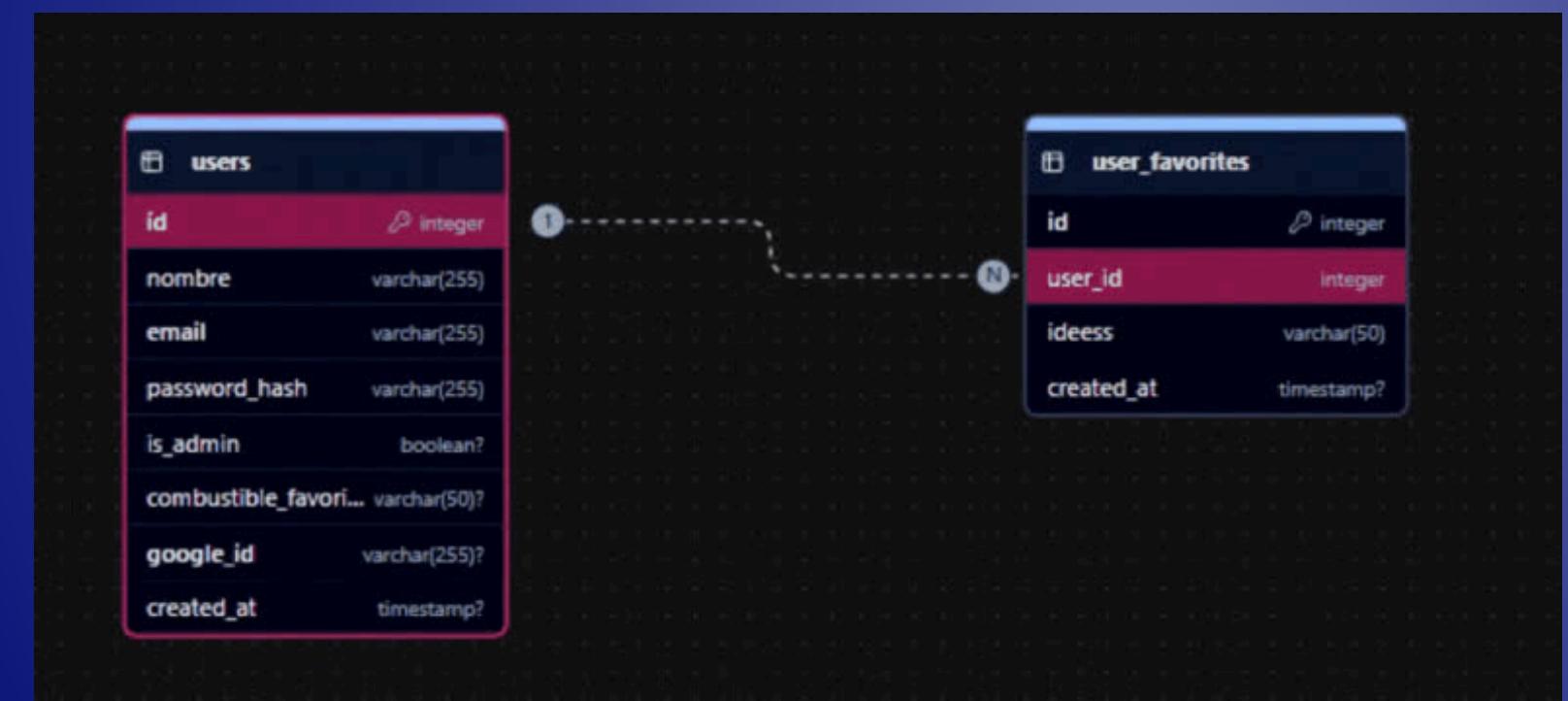


Modelo de datos

Gasolineras (MongoDB):



Usuarios
(PostgreSQL)



Arquitectura de la Solución

Microservicios

Usuarios
Gasolineras
Gateway

Bases de datos:

- PostgreSQL / MongoDB

API Gateway (Node.js)

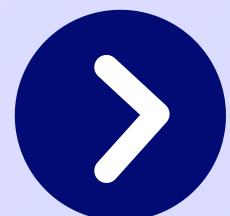
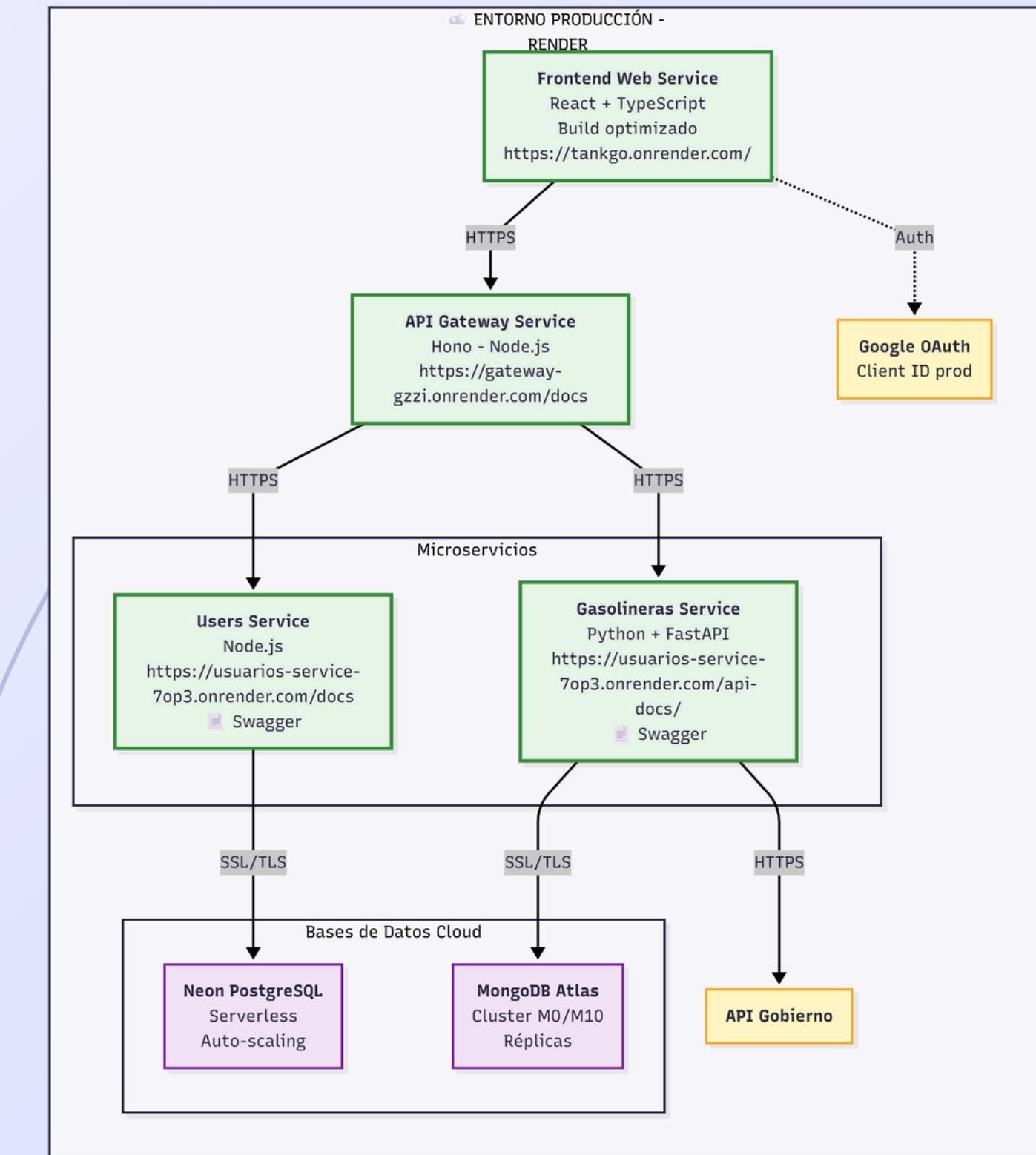
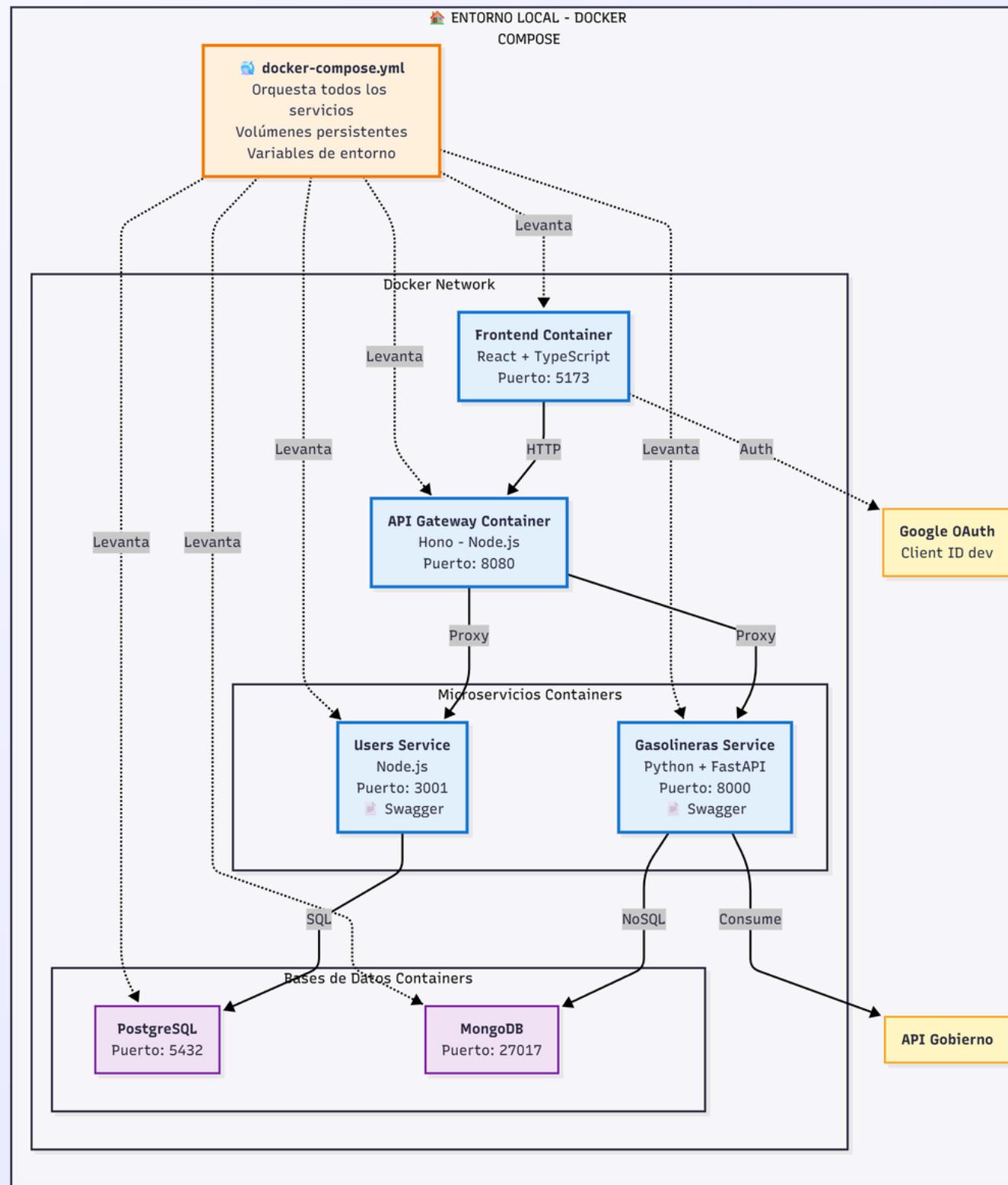
Es un API Gateway en Node.js, punto de entrada único. Permite consolidar la documentación (Swagger), simplificar llamadas frontend y backend

Frontend

Single Page Application (SPA) con React + TypeScript. Funciona como cliente HTML5 Diseño responsive. Manejo de estado y rutas.



Arquitectura



Stack Tecnológico

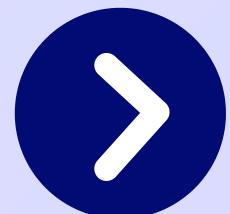
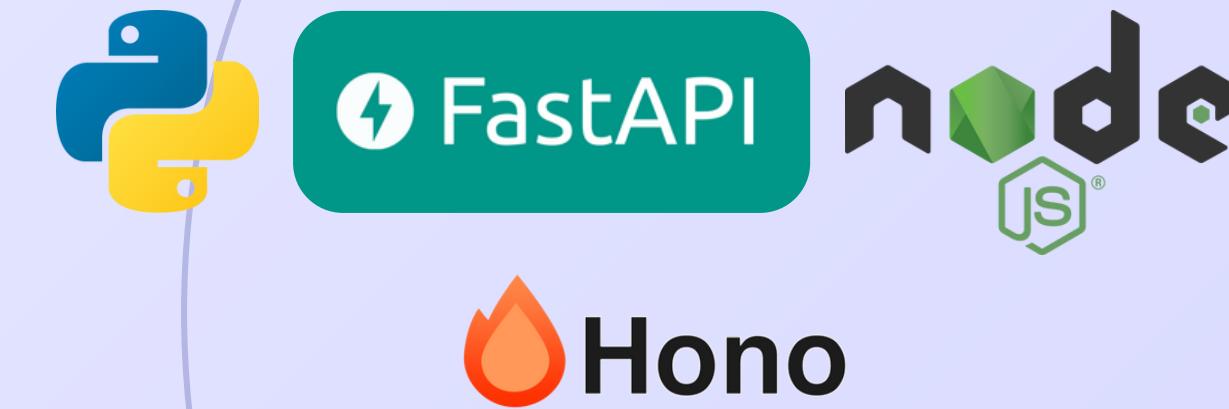
Frontend:

- **Librería:** React.js
- **Lenguajes:** HTML5, CSS3, Typescript
- **Estilos:** TailwindCSS
- **Mapas:** Leaflet.js



Backend:

- **Microservicio 1 Ingestor-Datos:** Python 3 + FastAPI
- **Microservicio 2 Servicio de Usuarios:** Node.js
- **API Gateway (Receptor de peticiones):** Node.js - Hono (por su rapidez)



Despliegue en la nube

Despliegue en producción

- Deploy completo en Render: Frontend, API Gateway y microservicios.
- Arquitectura basada en contenedores Docker → mismo entorno en local y en la nube.

Bases de datos externas:

- Neon (PostgreSQL) – servicio de usuarios.
- MongoDB Atlas (MongoDB) – servicio de gasolineras.

Objetivo del despliegue

- Garantizar escalabilidad, seguridad y consistencia entre desarrollo y producción.

Para tener un entorno de desarrollo adaptado y en local también funcione con sus contenedores de bases de datos, todo está estructurado con variables de entorno, en las que cada microservicio usa unas específicas, y en el caso de producción usan las conexiones de Mongo Atlas y Neon, así como las URLs de los servicios de Render para poder interactuar entre ellos.

Seguridad y buenas prácticas

- Credenciales gestionadas solo desde Render.
- SSL obligatorio en Neon y Atlas.
- API Gateway centraliza acceso, rutas y CORS.
- Principio de "secrets out of code".
- Arquitectura diseñada para escalar fácilmente.



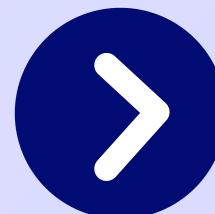
Lecciones aprendidas & logros técnicos

Arquitectura de microservicios “de verdad”

- Implementación real de servicios independientes con responsabilidades bien definidas.
- Resolución de problemas típicos del mundo real: CORS, comunicación entre servicios, autenticación compartida y gestión de logs.
- Aprendizaje sobre cómo diseñar sistemas que escalan y no se rompen entre sí.

Frontend profesional con React + TypeScript

- Uso avanzado del React Router, componentes bien encapsulados and reutilizables.
- Creación de custom hooks para separar lógica y vista.
- Tipado estricto con TypeScript, aprendiendo patrones propios de código profesional.
- Diseño responsive real pensado para móvil y escritorio.



Lecciones aprendidas & logros técnicos

Autenticación con Google (aprendida por mi cuenta)

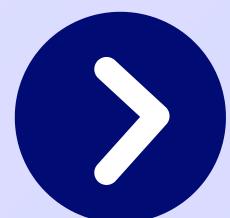
- Uso de proveedores externos de identidad OAuth 2.0.
- Comprensión del funcionamiento de tokens, redirectiones, callbacks y validación.

Coordinación de dos bases de datos muy diferentes (SQL + NoSQL)

- PostgreSQL para datos estructurados y relacionales.
- MongoDB para información flexible, geoespacial y de alto volumen.
- Aprendizaje de modelos híbridos y cuándo usar uno u otro.
- Reconocimiento práctico de las ventajas de cada enfoque

Despliegue completo en la nube con Render

- Configuración de 4 servicios independientes desplegados de forma coordinada.
- Manejo de variables de entorno, secretos, SSL, dominios y comunicación interna.
- Aprendizaje real de cómo un sistema pasa de “funcionar en mi PC” a funcionar para cualquiera.



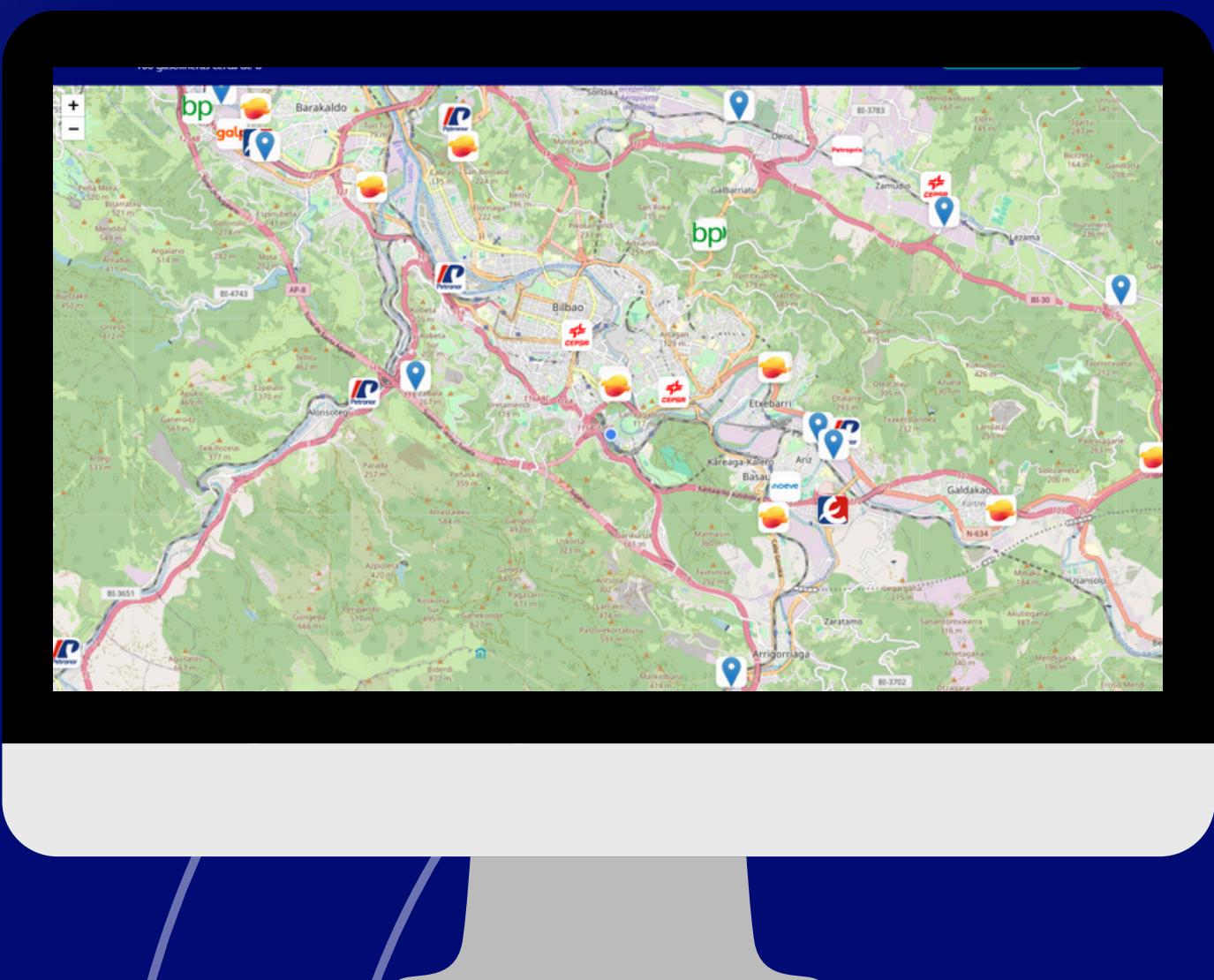
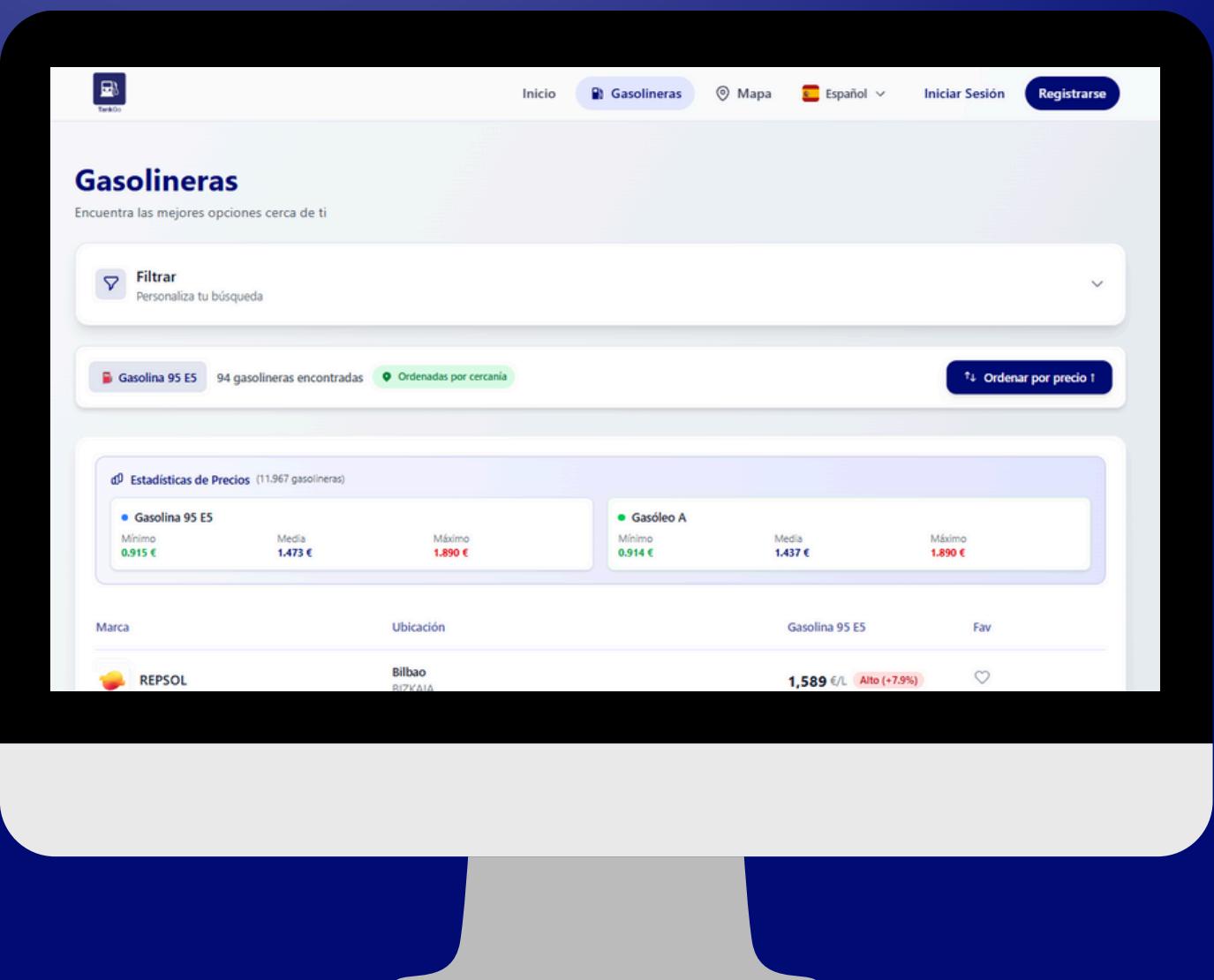
Interfaz de Usuario

Interfaz y Experiencia de Usuario (UX)

- Diseño **responsive**, optimizado para **móvil** y escritorio.
- **SPA** con **React + TypeScript** → navegación fluida sin recargas.
- **Interfaz intuitiva**: búsqueda clara, tarjetas de gasolineras y filtros simples.
- **Accesibilidad**: buen contraste, **botones grandes** y **navegación sencilla**.
- Componentes **reutilizables** y **código modular** (hooks personalizados).
- **Coherencia visual**: colores, tipografías e iconos alineados.



Prototipo



Prototipo

