

Manual Tecnico

Página Web

Inventario Forestal Nacional

Autores del manual

Carlos Martin Pinto Grisales
Edinson Ferney Beltran Gomez

Grupo

5A

Fecha de finalización

17/11/2025

Índice

Introducción	2
Requisitos y Requerimientos	6
Descripción general del sistema	6
Arquitectura y Tecnologías	6
Estructura del Proyecto	9
Base de datos	10
APIs y Endpoints	11
Mantenimiento y Actualización de la Arquitectura de Microservicios	14
Consejos y buenas prácticas de uso	15
Soporte y contacto	15

Introducción

Requisitos y Requerimientos

1. Gestión de Empleados
 - Permite registrar nuevos empleados con información como nombre, cargo, región, teléfono, cédula, correo, fecha de ingreso, foto y hoja de vida (PDF). También debe permitir filtrar empleados por nombre, cédula y correo, y visualizar detalles individuales de cada empleado.
2. Gestión de Brigadas
 - Permite crear, editar y visualizar brigadas, asignar brigadistas, un jefe de brigada y asociar brigadas a conglomerados. Los brigadistas deben ser seleccionados de la lista de empleados registrados en la base de datos de Supabase.
3. Aceptación o Rechazo de asignaciones
 - Permite que un brigadista acepte o rechace una asignación a un conglomerado por motivos imprevistos o imposibilidad de ejecución.
4. Generación de Informes
 - Permite generar informes basados en los datos de brigadas y conglomerados almacenados en Supabase. Los informes deben ser exportables en formatos como PDF y contener información relevante sobre las actividades de campo.

5. Generación de informes para Jefe de brigada
 - Permite generar informes al jefe de brigada, donde tendrá distintas opciones de informes, tales como de incidencias (se le pedirá una descripción de la incidencia), avances en la expedición (número de parcelas muestreadas). cantidad de especies registradas, altura promedio de los árboles, coordenadas GPS tomadas, fotográficas.
6. Actualización de datos personales de trabajadores
 - Permite actualizar datos personales de los trabajadores, como números de contacto y correos electrónicos.
7. Control de Asignación de Roles
 - Permite gestionar la asignación de roles por ejemplo (brigadista o administrador) mediante un listado interactivo de empleados registrados en la base de datos de Supabase. El listado debe mostrar información relevante como nombre, cédula, cargo, región y estudios (descripción) para facilitar la selección. La asignación de roles debe integrarse con el proceso de creación de brigadas, permitiendo al administrador seleccionar empleados directamente desde la interfaz por ejemplo, en ConformarBrigada y actualizar el campo rol en la tabla usuarios de Supabase.
8. Control de Asignación de Brigadas a Conglomerados
 - Permite asignar una brigada a un conglomerado mediante una interfaz interactiva. La funcionalidad debe incluir:
 1. Un listado de conglomerados disponibles (de la tabla conglomerados en Supabase) con información relevante como ubicación y descripción.
 2. La capacidad de seleccionar una brigada y un conglomerado para crear una asignación, almacenada en la tabla asignaciones_conglomerados de Supabase via el endpoint /api/asignar-conglomerado.
 3. Validación de permisos para garantizar que solo usuarios con rol admin (verificado por el middleware esAdmin) puedan realizar esta acción.

9. Control de Registro de Usuarios

- Permite registrar nuevos usuarios/empleados mediante endpoints protegidos POST /api/usuarios y /api/empleados, accesibles solo para usuarios con rol admin (validado por middleware esAdmin). El registro captura datos como nombre_completo, correo, cedula, telefono, region, descripcion, fecha_ingreso, cargo, y opcionalmente hoja_vida_url (archivos PDF almacenados en Supabase Storage bucket "hojas_de_vida"). Los datos se insertan en la tabla usuarios de Supabase con rol predeterminado 'brigadista'. La interfaz incluye validación de campos obligatorios y filtrado de empleados existentes por nombre, cédula y correo (implementado en empleados.html). Los nuevos usuarios pueden autenticarse posteriormente vía el servicio externo para acceder al sistema.

10. Detección Automática de Árboles mediante Análisis Satelital

- El conglomerado recibe las coordenadas (latitud, longitud) y automáticamente simula la detección de árboles. Para cada árbol detectado, el sistema calcula: el índice NDVI (Normalized Difference Vegetation Index) con valores entre 0.6 y 0.9 que indican la salud de la vegetación, el azimut usando trigonometría (ángulo de 0 a 360 grados medido desde el norte), y la distancia en metros desde el punto central del conglomerado usando la fórmula de Haversine. Cada árbol se clasifica automáticamente por categoría IFN según su valor NDVI: Brinzal (B) para árboles pequeños, Latizal (L) para medianos, Fustal (F) para grandes, y Fustal Grande (FG) para muy grandes. Los datos se guardan en la tabla de la BD y se retorna un JSON con las coordenadas, azimut, distancia y categoría para visualizar en el mapa.

11. Cálculo de Resumen Estadístico por Conglomerado

- El ID asignado procesa todos los árboles detectados en un conglomerado y calcula automáticamente estadísticas: total de árboles contados, cantidad de árboles por categoría (B, L, F, FG), diámetro promedio (DAP) en centímetros, altura promedio en metros, número de especies diferentes encontradas, cantidad de árboles vivos/muertos/enfermos, y distribución de azimut. El sistema consulta la tabla BD, aplica funciones SQL de agregación, y retorna los resultados en JSON. Las estadísticas se guardan en la tablas de BD para consultas futuras. Esta información es crucial para generar informes técnicos sobre el estado de los bosques.

12. Control de Verificación con Tokens

- La aplicación web implementa un sistema de autenticación segura mediante tokens JWT (JSON Web Tokens) validados externamente. Solo usuarios con token validos pueden acceder a usuarios y brigadas ya el usuario verifica la información del usuario autenticado.

13. Autenticación en el Inicio de Sesión

- La aplicación web utiliza verificación externa de tokens JWT mediante peticiones HTTP al servicio de autenticación iam-autenVerifi-service-ifn en lugar de 2FA tradicional. El middleware realiza una llamada axios, pasando el token en el header, recibiendo la información del usuario validado. Esta arquitectura descentralizada mejora la seguridad al mantener la lógica de autenticación en un servicio dedicado, permite escalabilidad, y asegura que solo tokens emitidos por el servicio oficial sean aceptados. En caso de fallo en la verificación (token expirado, inválido, o servicio caído), se devuelve HTTP 403 con mensaje de error.

14. Interfaz de Usuario Moderna

- La interfaz implementa un diseño moderno y de fácil uso, como se especifica en el prototipo del proyecto. La aplicación es responsiva y compatible con dispositivos móviles y de escritorio.

15. Rendimiento de la Aplicación

- La aplicación responde en menos de 2 segundos para solicitudes de visualización de datos (como listas de empleados o brigadas) bajo condiciones normales de red. El uso de Vercel para el despliegue debe garantizar escalabilidad para múltiples usuarios concurrentes.

16. Precisión de Detección de Árboles

- El Monitoring-Backend logra una precisión mínima del 80% en la detección de árboles cuando se compara con observaciones de campo. El índice NDVI debe capturar correctamente árboles con densidad de vegetación entre 0.6 y 0.9. Los cálculos de azimuth deben ser exactos dentro de 5 grados de margen de error. Las distancias calculadas con Haversine deben ser precisas dentro de 10 metros. El sistema debe manejar correctamente diferentes tipos de bosque (tropical, templado, boreal). Se recomienda validar los resultados de detección comparándolos con datos de campo y ajustar los algoritmos si es necesario.

17. Rendimiento en Procesamiento de Detecciones

- El endpoint POST completa el procesamiento y retorna resultados en máximo 7 segundos, incluso con conexión de internet lenta (3G/4G). El endpoint GET resumen debe responder en máximo 2 segundos para un conglomerado con hasta 500 árboles registrados(en este caso 20). El Monitoring-Backend debe escalar horizontalmente en Vercel para manejar múltiples solicitudes simultáneas de diferentes brigadas.

18. Dependencia de Servicios Externos

- El sistema depende de servicios externos como Supabase para la gestión de datos y el servicio de autenticación (iam-autenVerifi-service-ifn) para la verificación de tokens. Cualquier interrupción en estos servicios puede afectar la funcionalidad del sistema.

19. Limitaciones de Integración de Mapas

- La funcionalidad de visualización de mapas (pendiente, con posible integración de Leaflet según el README) no está implementada, lo que limita la capacidad de los usuarios para visualizar conglomerados geográficamente.

20. Limitaciones de Precisión en Imágenes Satelitales

- El servicio de levantamiento de datos depende de la calidad y disponibilidad de imágenes satelitales para calcular el NDVI y detectar árboles. Sin embargo, la calidad de estas imágenes varía según la época del año, la presencia de nubes, la resolución del satélite disponible, y el tipo de terreno. En épocas de lluvia o con mucha nubosidad, las imágenes satelitales pueden no ser claras, lo que reduce significativamente la precisión de la detección. Además, árboles muy pequeños, árboles bajo el dosel (cubiertos por árboles más grandes), o árboles en zonas montañosas con sombras pueden no ser detectados correctamente por el análisis NDVI.

Descripción general del sistema

Arquitectura y Tecnologías

Para el Frontend utilizamos tecnologías tales como React v19.1.1, Javascript, en despliegue preferimos desplegarlo en vercel.com.

Para el backend utilizamos varias tecnologías tales como FastAPI, Wordpress, Python, para generar la comunicación entre el Frontend y la base de datos, separando responsabilidades, en total se crearon 3 backend que corresponden a Login, Brigadas y Levantamiento de datos, cada backend ha sido desplegado en un vercel independiente.

Hemos utilizado tecnologías tales como supabase.com para desplegar 3 bases de datos, para Login, Brigadas y Monitoreo (Levantamiento de datos).

Hemos optado por una arquitectura de microservicios, en total 3 microservicios independientes de cada uno, cada uno con un backend independiente desplegado y comunicado correctamente.

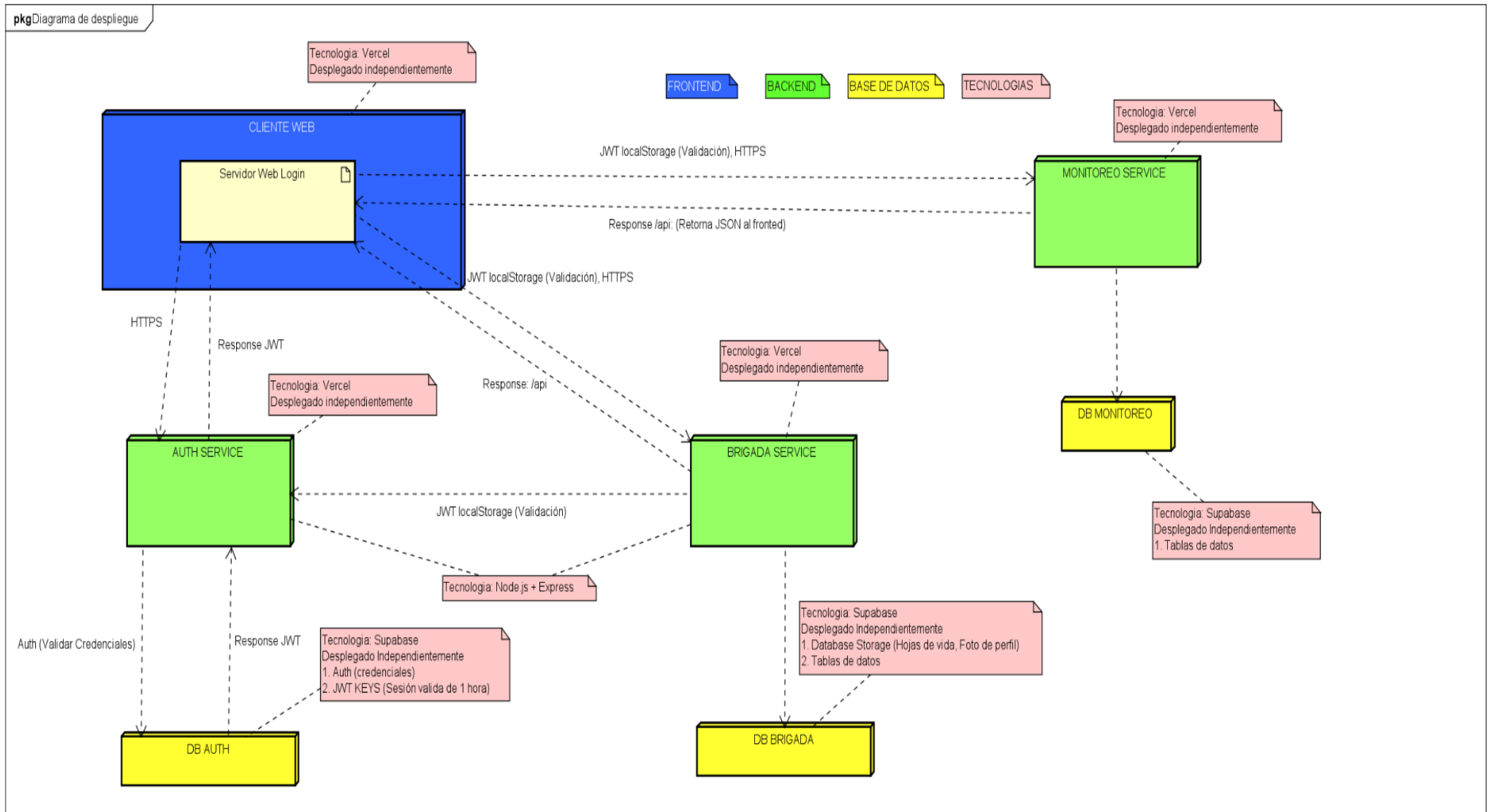
Otras tecnologías que utilizamos fueron las siguientes:

Bootstrap v5.3.8, AOS v2.3.4, Axios v1.13.2, leaflet v1.9.4, lucide-react v0.545.0, react-bootstrap v2.10.10, react-dom v19.1.1, react-leaflet v5, react-router-dom v7.9.1

Nota:

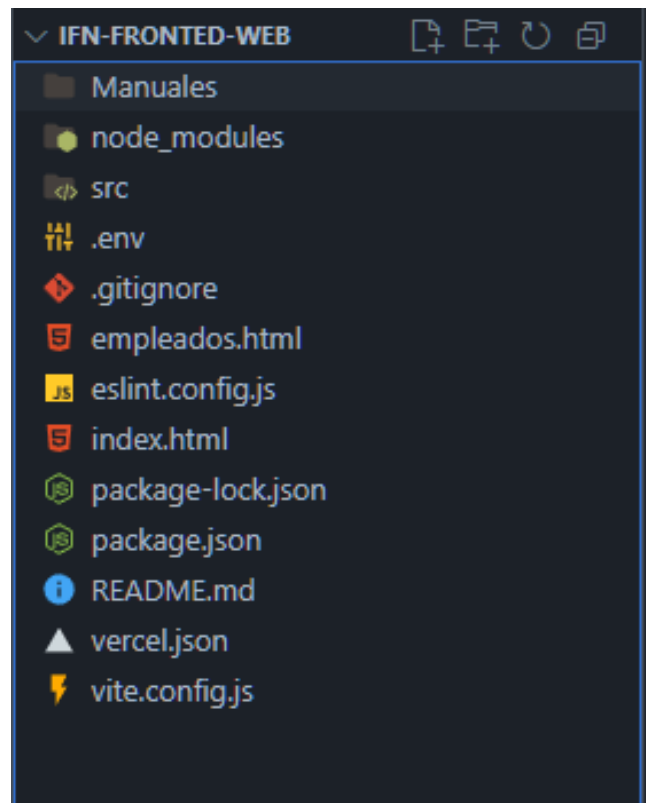
Hemos decidido utilizar el Framework de FastAPI por su documentación automática y respuesta a peticiones ante otros Frameworks.

Diagrama de despliegue



Estructura del Proyecto

Actualmente el Frontend está estructurado de la siguiente manera



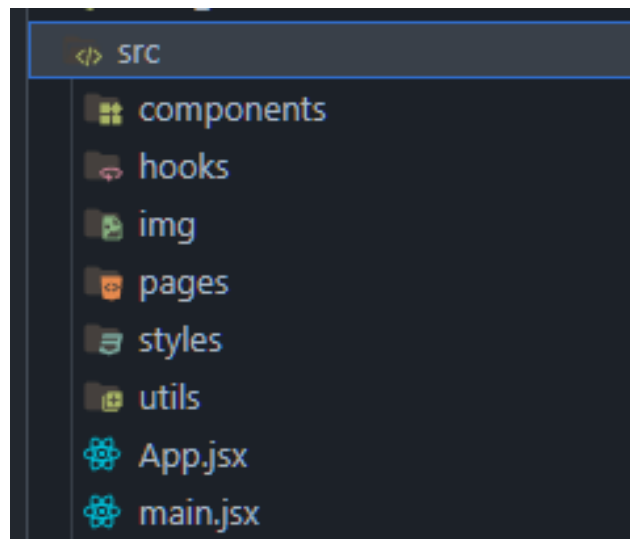
En esta imagen podemos observar una carpeta de “Manuales” donde se guardan todos los manuales (Usuario y Tecnico).

Tenemos un archivo .env donde guardamos las variables secretas que requieren seguridad.

En el .gitignore es el archivo donde guardamos los archivos del proyecto que no queremos que se carguen en el GitHub.

En el archivo vercel.json guardamos las configuraciones básicas requeridas por Vercel para ser desplegado.

En el archivo [vite.config.js](#) guardamos la configuración básica requerida por Vite para funcionar en acoplamiento con React.



En esta imagen podemos observar el contenido de la carpeta `src` donde tenemos nuestro proyecto, en el cual encontramos otra carpeta de `assets`, donde principalmente es una carpeta creada por React automáticamente donde se comparten recursos tales como imágenes.

En la carpeta de **componentes** se guardan las páginas que se utilizan en la parte del Frontend.

En la carpeta **hooks** guardamos el controlador de la sesión el cual verifica la autenticidad del token del usuario logueado.

En la carpeta **img** guardamos las imágenes que vamos a llamar durante el transcurso de la ejecución del sistema.

En la carpeta de **pages**, es donde controlamos las rutas del Frontend y administramos de mejor forma las rutas del administrador y el usuario, separando responsabilidades.

En la carpeta de **styles** guardamos los estilos de cada apartado.

En la carpeta de **utils** guardamos un archivo JSON el cual guarda atributos de Tailwind que nos ayudan a tener en cuenta ciertos diseños.

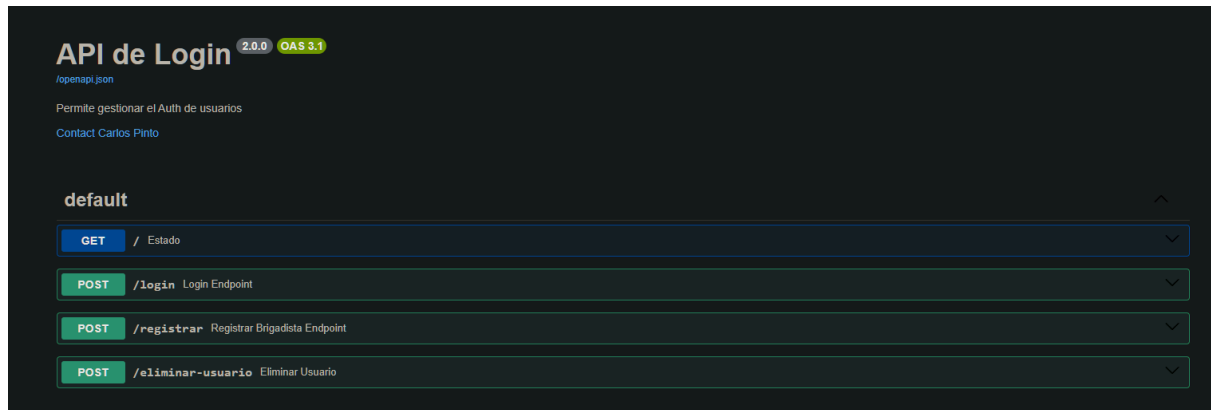
Base de datos

APIs y Endpoints

Hemos utilizado APIs como FastAPI y REST API WordPress.

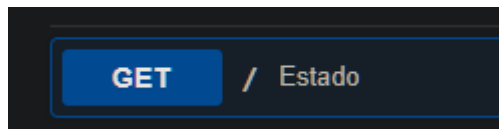
LOGIN FastAPI

Esté es el /docs de la API FastAPI, con sus rutas a la cual puedes acceder con este enlace <https://fast-api-login-six.vercel.app/docs>, que controla las rutas del backend de Login y solo utilizamos el Auth de Supabase.



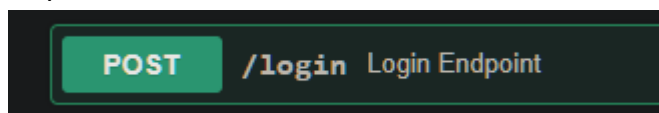
Ruta <https://fast-api-login-six.vercel.app/>

Es para ver el estado actual de la API.



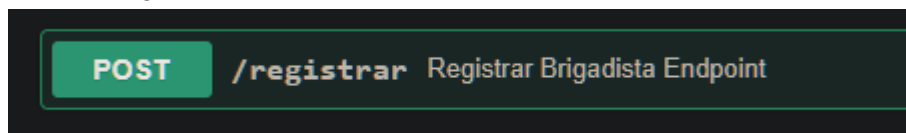
Ruta <https://fast-api-login-six.vercel.app/login>

Es para confirmar las credenciales de un usuario en el Auth de Supabase.



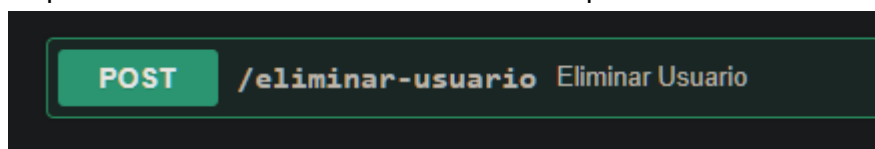
Ruta <https://fast-api-login-six.vercel.app/registrar>

Es para registrar a un nuevo usuario en el Auth de Supabase del microservicio de Login.



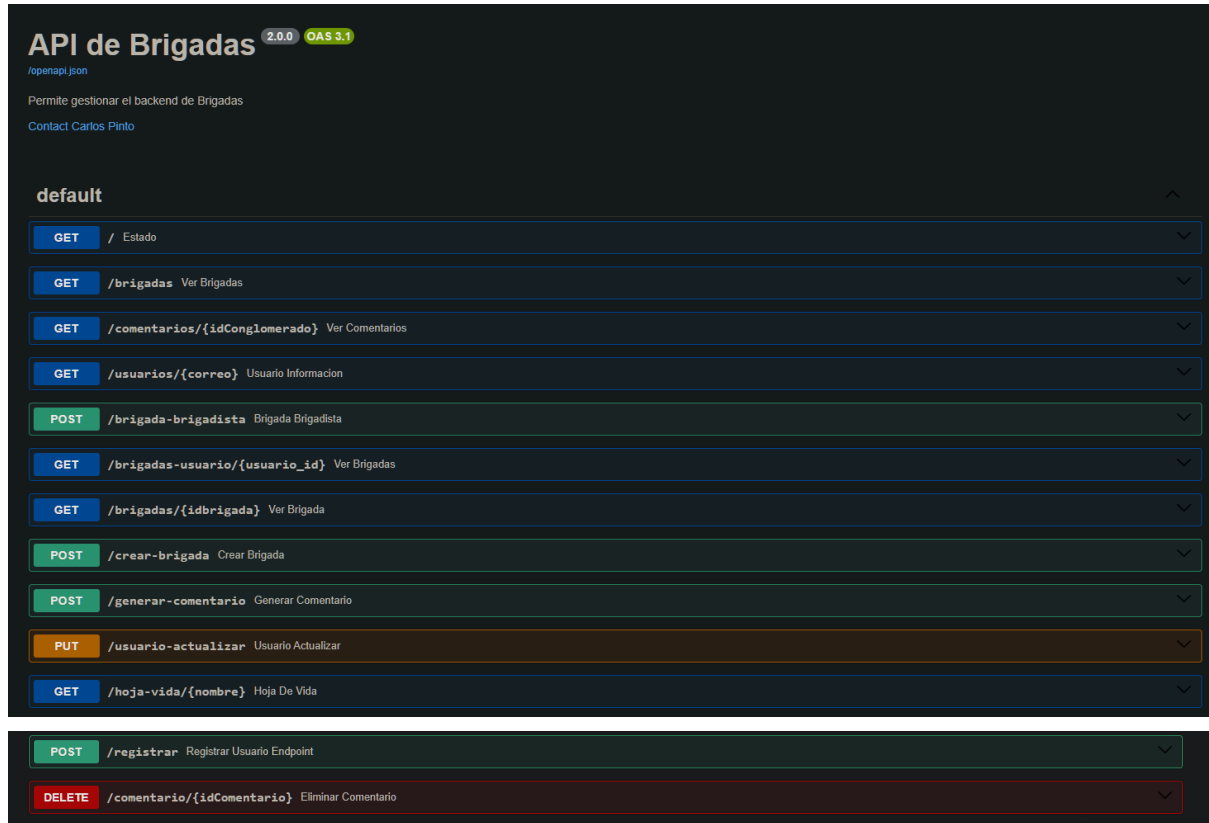
Ruta <https://fast-api-login-six.vercel.app/eliminar-usuario>

Es para eliminar a un usuario del Auth de Supabase.

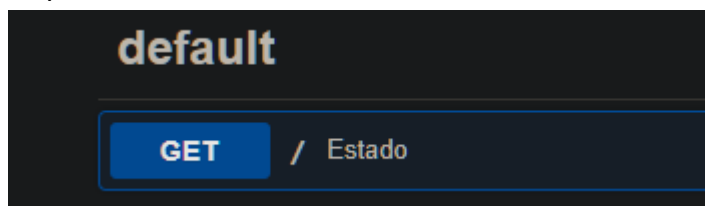


BRIGADAS FastAPI

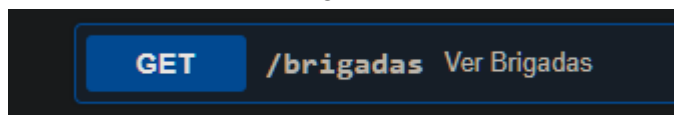
Esté es el /docs de Login FastAPI, con sus rutas a la cual puedes acceder con este enlace <https://fast-api-brigada.vercel.app/docs>, que controla las rutas del backend como brigadas, conglomerados, usuarios y asignaciones.



Ruta <https://fast-api-brigada.vercel.app/>
Es para ver el estado actual de la API.



Ruta <https://fast-api-brigada.vercel.app/brigadas>
Es para ver todas las brigadas existentes en la base de datos de Supabase.



Ruta <https://fast-api-brigada.vercel.app/comentarios/idConglomerado>

Es para ver todos los comentarios referentes a un conglomerado en específico a través de su ID.

GET /comentarios/{idConglomerado} Ver Comentarios

Ruta <https://fast-api-brigada.vercel.app/usuarios/martin5435412@gmail.com>

Es para ver la información completa de un usuario a través del correo electrónico registrado en el sistema.

GET /usuarios/{correo} Usuario Informacion

Ruta <https://fast-api-brigada.vercel.app/brigada-brigadista>

Es para solicitar todos los empleados que están asignados a una brigada.

POST /brigada-brigadista Brigada Brigadista

Ruta https://fast-api-brigada.vercel.app/brigadas-usuario/usuario_id

Es para ver las brigadas a las que está asignado un usuario a partir de su ID

GET /brigadas-usuario/{usuario_id} Ver Brigadas

Ruta https://fast-api-brigada.vercel.app/brigadas/brigada_id

Es para solicitar la información completa de una brigada en específico a través de la ID.

GET /brigadas/{idbrigada} Ver Brigada

Ruta <https://fast-api-brigada.vercel.app/crear-brigada>

Es para crear una brigada en el Supabase.

POST /crear-brigada Crear Brigada

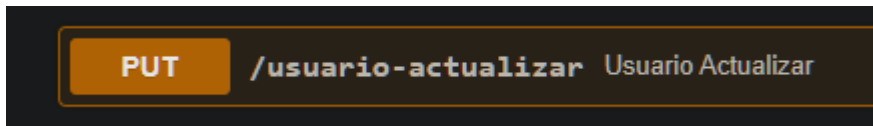
Ruta <https://fast-api-brigada.vercel.app/generar-comentario>

Es para registrar un nuevo comentario en una brigada.

POST /generar-comentario Generar Comentario

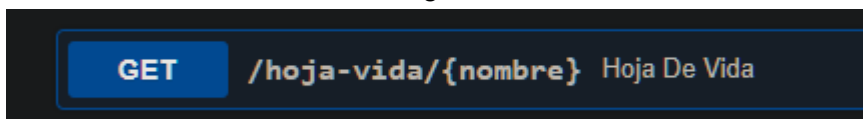
Ruta <https://fast-api-brigada.vercel.app/usuario-actualizar>

Es para que el usuario presente en la página pueda cambiar su nueva información, tales como región, descripción y teléfono..



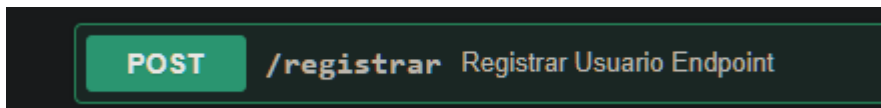
Ruta <https://fast-api-brigada.vercel.app/hoja-vida/nombre>

Es para solicitar la hoja de vida de un usuario en específico, utilizando el nombre del archivo obtenido a través de un atributo guardado en la tabla Usuario.



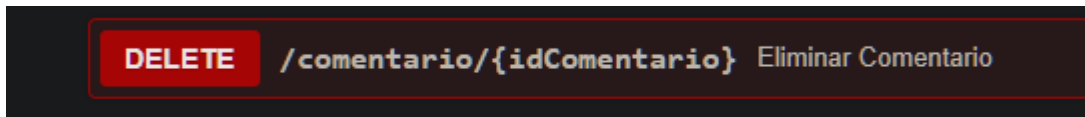
Ruta <https://fast-api-brigada.vercel.app/regarstrar>

Para registrar a un usuario en la tabla Usuarios de el microservicio de Brigadas.



Ruta https://fast-api-brigada.vercel.app/comentario/conglomerado_id

Para eliminar un comentario en específico utilizando el ID.



Mantenimiento y Actualización de la Arquitectura de Microservicios

1. Actualizar API KEYS

Los enlaces a cada API están protegidos en un archivo llamado .env que se encuentra en la carpeta principal de cada servicio, en él podrán ir cambiando los enlaces a las APIs dependiendo si lo requiera.

2. Dependencias actualizadas

Debe estar al tanto del versionamiento de las dependencias y sus actualizaciones, en caso de fallar una dependencia, debe mantenerlas en su última versión para un correcto uso y soporte.

3. Despliegue actualizado en Vercel

Para realizar realizar en el software que está desplegado en Vercel, deberá subir los cambios al mismo repositorio donde se encuentra, ya sea por medio de GitHub Desktop o directamente desde el GitHub, estos cambios automáticamente se verán reflejados en la página desplegada.

Consejos y buenas prácticas de uso

1. Documentar cada cambio en el mismo archivo que está actualizando.
2. Nunca actualice directamente en el servidor de despliegue, primero haga pruebas unitarias en su computador, desplegando de forma local del software y verificando su funcionamiento correcto.
3. Nunca utilices las llaves de APIs o información sensible por fuera del archivo .env por temas de seguridad.

Soporte y contacto

Correo: cpinto5@udi.edu.co, ebeltran7@udi.edu.co

Teléfono: 3114589229

Horario: Lunes a sábado, 9:00 a.m. a 8:00 p.m.