

## **DOCUMENTACIÓN TÉCNICA A ENTREGAR**

Alejandro Garcia-Mauriño Salas

Fco.Javier Saravia Ogazon

Sergio Mitchell Bocero

Antonio Moro del Toro

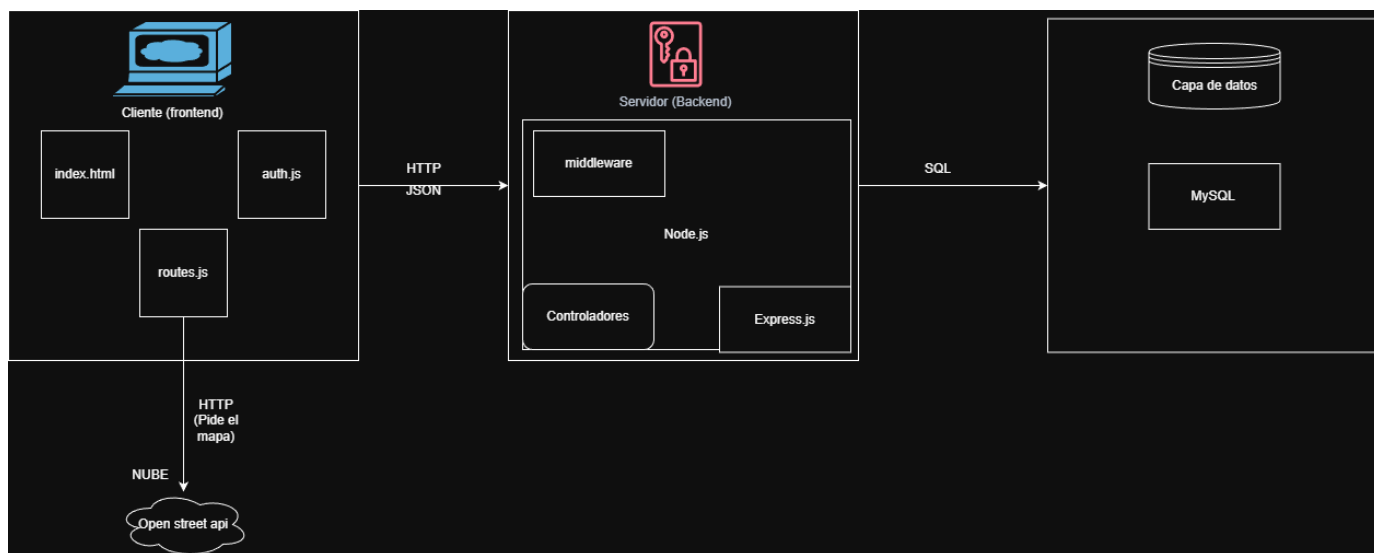


# Índice

1. **Diagrama de Arquitectura Real**
2. **Modelo de Datos Definitivo**
3. **Documentación la API**
4. **Informe de Problemas**

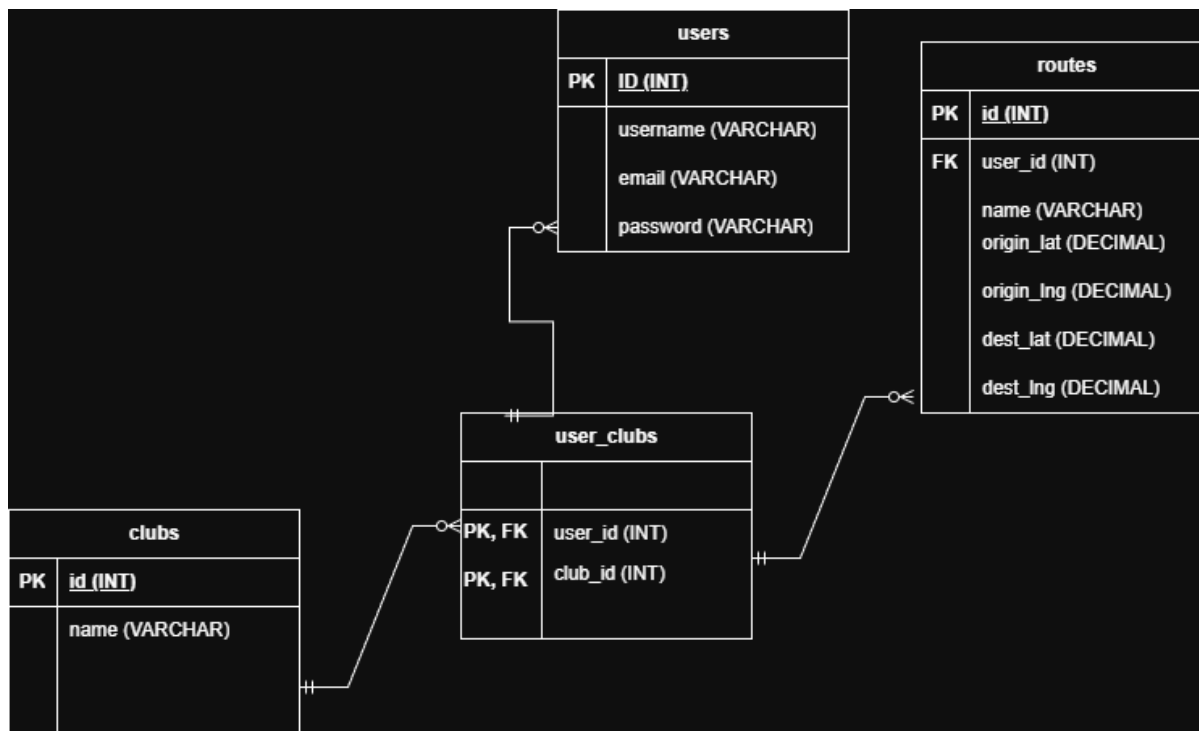


# 1. Diagrama de Arquitectura Real





## 2. Modelo de Datos Definitivo





## 3. Documentación la API

- Hemos utilizado una API para la autenticación y el registro de usuarios.

### Documentación de dicha API:

- POST /api/auth/register→ Esto la función que tiene es de Registrar un nuevo usuario.
- POST /api/auth/login→ Esto la función que tiene es iniciar sesión el usuario.
- Este endpoint no es de dicha API sino de nuestra propia api del backend→ POST /guardar: Esto sirve para guardar las rutas.

## 4. Informe de Problemas

### **Problema:**

- Al registrarse el usuario no se guardaba correctamente en la base de datos, por lo cuál al iniciar sesión no lo reconocía y salía usuario no encontrado.

**Solución:** Hemos configurado un JWT (JSON Web Tokens) seguro, para que así el usuario pueda autenticarse correctamente y acceder a los endpoints.

- Al guardar una ruta, la base de datos devolvió un error o no guardaba nada

**Solución:** Se modificó la estructura de la tabla routes, eliminando las columnas individuales y creando una columna de tipo JSON llamada coordinates. Esto permite guardar tanto puntos únicos como trazados complejos.



- Error 404: Al intentar guardar una ruta o unirse a un club, la consola del navegador mostraba POST http://localhost:3000/api/... 404 (Not Found)

**Solución:** Se añadieron las líneas de conexión en app.js y se estableció un protocolo de reinicio del servidor tras cada cambio en el Backend

- Un usuario podía unirse infinitas veces al mismo club pulsando el botón repetidamente, llenando la base de datos de registros duplicados

**Solución:** Se implementó una lógica de verificación en el endpoint /join: antes de insertar, se hace un SELECT para ver si la combinación user\_id + club\_id ya existe.

- No se podía distinguir visualmente a qué clubes pertenecía el usuario; todos aparecían con el botón "Unirse".

**Solución:** Se creó una nueva ruta con un JOIN SQL entre la tabla clubs y la tabla intermedia user\_clubs para filtrar las membresías

Se implementó una lógica en JavaScript para comparar los IDs y separar las tarjetas en dos secciones visuales distintas (Verde para "Mis Clubes", Blanca para "Explorar").