Historia: Medianos para ti, para mí y para todos

En lo profundo de la **Calle Colombia en Medellín**, un hombre llamado **Juan Pablo Agudelo** decidió cambiar su vida. Luego de años de luchar contra **la adicción al LoL** y un pasado marcado por **las drogas y la desidia**, tuvo un momento de iluminación: debía emprender. Inspirado en su amor por la noche y las oportunidades que esta ofrece, fundó el **burdel "Medianos para ti, para mí y para todos"**.

El negocio iba bien, pero tenía un **problema logístico**: **el manejo de las placas de los carros de los clientes**. La libreta donde registraban las placas se perdía constantemente, lo que generaba caos y discusiones innecesarias.

Para solucionar esto, **Juan Pablo** decidió contratar a un desarrollador para crear un **sistema digital** que registrara y gestionara las placas de los vehículos que ingresaban al establecimiento.

El encargado de la portería, **Juanjo**, tenía una **misión doble**: recibir a los clientes y asegurarse de que el nuevo sistema funcionara. Aunque su mayor desafío personal era **superar su miedo a ISIS y a las estructuras de datos**, decidió que este era el momento perfecto para **enfrentar sus temores** y evolucionar como profesional.

Tu tarea es desarrollar este sistema en **Nest.js con PostgreSQL** y un **frontend en HTML y CSS**.

Parte 1: Desarrollo del Backend (Nest.js + PostgreSQL)

Requisitos del Backend

Debes construir una API RESTful en **Nest.js** que gestione una **entidad "Placas"** en una base de datos **PostgreSQL** con la siguiente estructura:

Tabla: placas

Campo	Tipo de Dato	Restricciones
id	String (PK)	UUID
placa	VARCHAR(10)	Único, No Nulo
marca	VARCHAR(50)	No Nulo
modelo	INT	No Nulo
color	VARCHAR(30)	No Nulo
fecha_ingreso	TIMESTAMP	Default NOW()

Endpoints Obligatorios

Tu API debe incluir los siguientes **endpoints**:

1. **POST /placas** – Registrar una nueva placa.
2. **GET /placas** – Listar todas las placas registradas.
3. **GET /placas/:placa** – Consultar una placa específica por su número.
4. **DELETE /placas/:placa** – Eliminar una placa.

Requisitos Técnicos:

✅ **Uso de TypeORM** para manejar la base de datos PostgreSQL.
✅ **Validaciones** con class-validator para verificar formato de placa y datos obligatorios.
✅ **Uso de DTOs** para estructurar las solicitudes y respuestas.

Parte 2: Diseño del Frontend (HTML + CSS)

El frontend debe ser **una página HTML estática** con CSS que incluya:

1. 📜 **Una tabla** con todas las placas registradas.
2. 📋 **Un formulario** para registrar una nueva placa con los siguientes campos:

- Número de placa
- Marca
- Modelo
- Color

3. 🗑️ **Un botón para eliminar placas** del registro.

🔹 **Puntos extra** por:

✅ **Un diseño llamativo y bien estructurado.**

✅ **Uso de CSS para mejorar la presentación de la tabla y formulario.**

Evaluación y Entrega

Entrega:

- Sube el código a un repositorio en **GitHub** y envía el enlace.
- El backend debe estar documentado con Postman.

Criterios de Evaluación:

- ✅ **Funcionalidad correcta** del backend con Nest.js y PostgreSQL.
- ✅ **Estructura limpia del código** y uso adecuado de TypeORM y DTOs.
- ✅ **Interfaz funcional y estéticamente agradable** en el frontend.
- ✅ **Uso de validaciones y buenas prácticas.**