TECNOLOGICO NACIONAL DE MEXICO.

INSTITUTO TECNOLOGICO DE CULIACAN.

SISTEMA DE DETECCIÓN Y GESTIÓN DE PLACAS VEHICULARES (FULL-STACK)

TOPICOS DE INTELIGENCIA ARTIFICIAL 12-13.

DOCENTE: MORA FELIX ZURIEL DATHAN.

CLAVE DE LA MATERIA: GTD2102

NOMBRE DE LOS ALUMNOS:

- BORBÓN SÁNCHEZ EDGAR - 16171301
- MILLÁN LÓPEZ ANA KAREN - 20170985

UNIDAD: 4.

ACTIVIDAD: PROYECTO DE SISTEMA DE DETECCIÓN DE PLACAS VEHICULARES.

CULIACÁN, SINALOA.

Contenido

1. Objetivo General

Diseñar, implementar y documentar un sistema *full-stack* capaz de detectar con precisión las matrículas de vehículos a partir de imágenes (visión artificial) y asociarlas eficientemente con los datos de sus respectivos propietarios almacenados en una base de datos, facilitando la gestión y el reporte de incidentes en tiempo real.

2. Descripción del Problema

La falta de respeto a los espacios designados para personas con discapacidades en estacionamientos es un problema recurrente que dificulta el acceso y la movilidad de quienes más lo necesitan. El proceso de identificación de infractores suele ser manual y tardado.

Este proyecto aborda el problema ofreciendo una **solución móvil y automatizada** que permite a los usuarios generar un reporte instantáneo: fotografiando la placa, detectándola con IA, y registrando la ubicación GPS exacta del vehículo mal estacionado. Esto permite una acción correctiva inmediata y la notificación al infractor.

3. Justificación

- **Aplicación de la Ley y Gestión de Vehículos:** El sistema es una herramienta de respuesta rápida que asocia una placa vehicular detectada por IA con la información del propietario y su historial de incidentes, lo cual es invaluable para administradores de estacionamientos o cuerpos de seguridad.
- **Impacto Social y Ético:** El proyecto contribuye directamente a garantizar el respeto a los derechos y espacios de las personas con necesidades especiales, promoviendo la seguridad y la responsabilidad cívica.
- **Integración de Tópicos de IA:** El proyecto es un ejercicio práctico de la integración de modelos de visión artificial (YOLO y OCRs) en un entorno de producción *full-stack*.

4. Especificaciones Técnicas

4.1 Arquitectura del Sistema

El proyecto opera bajo una arquitectura **Cliente-Servidor de tres capas**, con un fuerte enfoque en la IA y la movilidad.

- **Capa Cliente (Frontend): Aplicación Móvil Nativa de Android** (*Kotlin, Jetpack Compose*). Responsable de la **UI**, la captura de imágenes con la cámara, la obtención de coordenadas **GPS** y la comunicación con el servidor mediante **Retrofit2**.
- **Capa Servidor (Backend/API):** Servidor **API REST(**Representational State Transfer) desarrollado en **Python** utilizando el framework **Flask**, y servido por **Gunicorn**. Es el *API Gateway* que gestiona el flujo de datos.
- **Capa de IA (Integrada en Backend):** Utiliza para la detección precisa de la placa ***Ultralytics YOLO*** y en el reconocimiento híbrido de texto (***EasyOCR + PaddleOCR****)* para mayor robustez.
- **Capa de Datos: MySQL** para la persistencia, con tablas que gestionan personas, vehículos y su historial de incidencias.

4.2 Esquema de la Base de Datos (MySQL)

La base de datos se llama **appPlacas** y contiene las siguientes tablas, con sus relaciones y propósitos:

Tabla	Propósito	Campos Clave	Relaciones (Claves Foráneas)
Personas	Almacena datos del propietario.	personaID (PK), nombre, apPaterno, apMaterno, email (UNIQUE)	Ninguna
Carros	Almacena información del vehículo.	carroID (PK), placa (UNIQUE)	Ninguna
Propietarios	Relaciona Personas con Carros (N:M).	personaID (PK/FK), carroID (PK/FK)	FK a Personas.personaID, FK a Carros.carroID
Incidencias	Almacena el historial y contador de reportes.	personaID (PK/FK), carroID (PK/FK), numIncidencia	FK a Personas.personaID, FK a Carros.carroID

4.3 API Utilizadas (Endpoints y Servicios)

La comunicación se realiza mediante **HTTP POST** (JSON y datos binarios de imágenes).

Endpoint Principal	Descripción	Flujo de Datos

/enviarImagen	Modo *solo lectura*: Recibe imagen, detecta placa y devuelve el texto reconocido sin registrar nada.	Cliente → Servidor (*IA*) → Cliente
/registrarPlaca	Registra una nueva persona y asocia su vehículo.	Cliente → Servidor (*Datos JSON*) → *MySQL*
/crearReporte	Procesa la imagen con IA, registra la incidencia e inicia el flujo de notificación.	Cliente (Imagen + GPS) → Servidor (*IA* + *MySQL* + *Email*)
Servicio de Notificación	Envía un correo electrónico al propietario cuando alcanza el umbral de incidentes (≥ 3 en producción).	Backend (*smtplib*) → Servidor Gmail (*SMTP*)

5. Manual de Instalación (Proceso Detallado)

La configuración requiere la inicialización y coordinación de tres procesos: Base de Datos, Backend y Red (ngrok).

5.1 Requisitos de Software

- Servidor ***MySQL*** (ej. *XAMPP, MySQL Workbench*).
- Python 3.x con ***pip***.
- Android Studio.
- Herramienta ***ngrok***.

5.2 Configuración del Backend (Python)

1. **Clonación y Entorno:** Clonar el repositorio y crear/activar un entorno virtual de Python.

```
python -m venv .venv
source .venv/bin/activate  # (En Linux/macOS)
.\.venv\Scripts\activate   # (En Windows)
```

2. **Dependencias:** Instalar las librerías necesarias para el servidor, la base de datos y la IA.

   pip install flask gunicorn ultralytics easyocr paddlepaddle paddleocr Pillow opencv-python-headless mysql-connector-python python-dotenv numpy

3. **Modelo IA**: Colocar el archivo del modelo ***YOLO*** entrenado (*bestV2.pt*) en la raíz del backend.

4. **Credenciales Seguras**: Crear un archivo ***.env*** en la raíz del backend para almacenar las contraseñas de la DB y del servicio de correo (**EMAIL_PASSWORD**), asegurándose de que este archivo esté en el *.gitignore*.

```
# Contraseña de tu usuario 'root' de MySQL
DB_PASSWORD="tu_password_de_mysql"

# Contraseña de App de Google (NO tu contraseña normal de Gmail)
EMAIL_PASSWORD="xxxx xxxx xxxx xxxx"
```

5.3 Configuración de Red y Ejecución

Para que el sistema funcione, se deben ejecutar tres terminales simultáneamente:

Terminal 1: Base de Datos

- Asegúrese de que el servicio MySQL esté corriendo (ej. a través de *XAMPP*).

Terminal 2: Servidor Backend (*Gunicorn*)

- Activar el entorno virtual e iniciar la API:

```
source .venv/bin/activate
gunicorn -w 4 -b 0.0.0.0:5000 main:app
```

Terminal 3: Túnel Público (*ngrok*)

- Exponer el puerto 5000 de forma segura a la web:

```
ngrok http 5000
```

  - **Paso Crítico:** Copiar la URL HTTPS generada por *ngrok* (ej. https://xxxx.ngrok-free.dev).

5.4 Configuración del Frontend (Android)

1. Abrir el proyecto en Android Studio.
2. Navegar al objeto ***RetrofitClient*** (en ***MainActivity.kt***).
3. **Actualizar la BASE_URL** con la URL HTTPS de ***ngrok*** copiada. Este paso es obligatorio y se debe repetir si el túnel ***ngrok*** se reinicia.
4. Compilar y ejecutar la App en un dispositivo físico.

6. Manual de Usuario (Operación)

El sistema opera a través de una aplicación móvil nativa de Android.

6.1 Inicio y Registro

1. **Instalación:** Instale la aplicación en su dispositivo Android.

2. **Registro de Placas:** Utilice la opción de registro para ingresar los datos del propietario (nombre, email) y la placa de su vehículo. Esta información es almacenada en el sistema para futuras consultas.

6.2 Generación de Reportes de Incidencia

1. **Captura de Imagen:** Acceda a la función de reporte de incidentes en la aplicación.
2. **Tomar Foto:** Utilice la cámara para capturar una imagen clara de la placa vehicular.
3. **Envío y Procesamiento:**
    - La aplicación automáticamente obtiene y adjunta la ubicación **GPS** del incidente.
    - El servidor utiliza el modelo híbrido de Visión Artificial (***YOLO, EasyOCR y PaddleOCR***) para identificar el texto de la placa.
    - Se registra la incidencia y se incrementa el contador del vehículo en la base de datos.
4. **Notificación:** Si el vehículo tiene **3 o más reportes**, se enviará un correo electrónico de alerta al propietario, incluyendo los detalles del incidente y un enlace a la ubicación GPS.

7. Consejos para la Solución de Problemas (Troubleshooting)

Problema	Causa Potencial	Solución
Error de conexión al servidor	El servidor Python está inactivo o la URL de ***ngrok*** ha caducado.	Es necesario actualizar la variable **BASE_URL** en la aplicación Android y verificar que *Gunicorn* esté en ejecución en el servidor.
Detección de placa fallida/incorrecta	La imagen está borrosa, mal iluminada, la placa está en ángulo o parcialmente oculta.	Asegúrese de tomar la foto con buena luz, de un ángulo favorable y que la placa se vea visible en su mayoría. En casos extremos, tomar foto en modo horizontal.
La aplicación no se ejecuta	Problemas de compatibilidad con dispositivos antiguos.	La aplicación debería funcionar en la mayoría de los dispositivos Android, pero verifique los requisitos mínimos de Android Studio para el ***target SDK***.
No se reciben notificaciones por correo	La contraseña de la aplicación de Google no está configurada correctamente en el archivo .***env*** del backend.	Verifique las credenciales de correo (*EMAIL_PASSWORD*) en el archivo ***.env*** del backend.

8. Repositorio de GitHub

https://github.com/Edgarborbon/TopicosIA/tree/main/MODULO%204/Sistema%20de%20detección%20de%20placas%20vehiculares