INSTITUTO TECNOLÓGICO DE CULIACÁN

Ingeniería en Sistemas Computacionales

Tópicos de Inteligencia Artificial

Hora: 12:00 – 01:00 PM

Manual de Instalación

Sistema Detector de Placas Vehiculares

Equipo:

Peña López Miguel Ángel

Robles Rios Jacquelin

Docente:

Mora Félix Zuriel Dathan

Culiacán, Sinaloa

15/11/2025

Documentación Técnica y Manual de Instalación

Especificaciones Técnicas del Sistema

1 Índice

2 Arquitectura General del Sistema 4

3 Tecnologías y Dependencias Clave 4

4 Esquema de la Base de Datos (PostgreSQL) 4

5 Especificaciones de las APIs 5

5.1 Web Service Principal (Backend/Datos) 5

5.2 Model Server (Inferencia/IA) 5

6 Manual de Instalación y Configuración (Backend) 5

6.1 Paso 1: Configuración del Entorno Python 6

6.2 Paso 2: Configuración de Variables de Entorno 6

6.3 Paso 3: Inicialización y Despliegue (Producción) 6

6.4 Paso 4: Inicialización de la Base de Datos 7

2 Arquitectura General del Sistema

El sistema opera bajo una **Arquitectura de Microservicios Desacoplada** que consta de cuatros componentes principales:

- **Capa de Cliente (Móvil):** Aplicación nativa Android desarrollada en **Kotlin**.
- **Capa de Servicios REST (Web Service):** Aplicación Python basada en **Flask**.
- **Capa de Inferencia (Model Server):** Aplicación Python basada en **Flask** con modelos de Visión Artificial.
- **Capa de la Base de Datos:** BD hosteada en la web desde Render

3 Tecnologías y Dependencias Clave

Componente	Rol	Tecnologías Principales
Model Server	Servidor de Modelos	Python 3.x, **YOLOv8n** (Ultralytics), **EasyOCR**, OpenCV, Flask.
Backend / API	Gestión de Datos (ORM)	Flask, Flask-SQLAlchemy, **Gunicorn** (Servidor WSGI), CORS.
Base de Datos	Persistencia de Datos	**PostgreSQL** (Alojado en Render).
Frontend	Interfaz de Usuario	Android Studio, **Kotlin** (Nativo), Retrofit (Networking).

4 Esquema de la Base de Datos (PostgreSQL)

El sistema utiliza un modelo relacional con las siguientes entidades principales. La relación es de uno a muchos (1:N) y las claves foráneas (FK) aseguran la integridad referencial.

Tabla	Clave Primaria (PK)	Relaciones (FK)	Campos Clave
Propietario	PropietarioID (INT)	N/A	Nombre, Correo (UNIQUE), CURP (UNIQUE).
Vehiculos	NumPlaca (STRING)	PropietarioID (1:N)	Modelo, Marca, Año.
Usuario	UsuarioID (INT)	N/A	Nombre, Contraseña, Correo (UNIQUE).

Reporte	ReporteID (INT)	NumPlaca (N:1), UsuarioID (N:1)	FechaEmision, Coordenadas, Descripcion, ImgEvidencia.

5 Especificaciones de las APIs

El sistema utiliza dos servicios web, ambos en Python, cada uno con un rol específico.

5.1 Web Service Principal (Backend/Datos)

Endpoint	Método	Descripción	Uso (Cliente)
/api/usuarios/register	POST	Crea un nuevo registro de usuario.	Pantalla de Registro.
/api/reportes/	POST	Registra una nueva incidencia en la BD.	Pantalla de Completar Reporte.
/api/reportes/	GET	Obtiene el listado de reportes del usuario.	Pantalla de Mis Reportes (Inicio).

5.2 Model Server (Inferencia/IA)

Endpoint	Método	Descripción	Uso (Cliente)
/analizar	POST	Ejecuta el modelo YOLOv8 + EasyOCR sobre una imagen binaria.	Pantalla de Generar Reporte (Despúes de tomar foto).
Entrada	multipart/form-data (archivo: imagen)	**Salida**	JSON: {"placa": "VLD817D", "confianza": 0.95}

6 Manual de Instalación y Configuración (Backend)

Esta sección describe cómo configurar y desplegar los dos servicios de Python. Se asume un entorno Linux/Git.

6.1 Paso 1: Configuración del Entorno Python

Clonar el Repositorio:

git clone https://github.com/jacq1813/Topicos_IA

cd \Unidad4\ProyectoDeteccionPlacas\WebServices>

Crear y Activar Entorno Virtual: (Recomendado para manejar dependencias)

python3 -m venv venv

source venv/bin/activate # En Linux/macOS

.\venv\Scripts\activate # En Windows (CMD/PowerShell)

Instalar Dependencias: (Requiere los archivos requirements.txt del Web Service y del Model Server)

pip install -r requirements_backend.txt

pip install -r requirements_model.txt

6.2 Paso 2: Configuración de Variables de Entorno

El sistema depende de la variable de entorno DATABASE_URL para conectarse a PostgreSQL.

Crear Archivo .env (para desarrollo local):

Archivo: .env

DATABASE_URL="postgresql://bddetectorplates_user:Je6U9C08KLCWhINAyfPVkVZaQi41t68L@dpg-d4d9h6qli9vc73cdf4s0-a.oregon-postgres.render.com/bddetectorplates"

Ejemplo de Render (producción)

#DATABASE_URL="postgresql://bddetectorplates_user:Je6U9C08KLCWhINAyfPVkVZaQi41t68L@dpg-d4d9h6qli9vc73cdf4s0-a.oregon-postgres.render.com/bddetectorplates "

Model Server (Archivos de Pesos): Asegúrese de que el archivo de pesos del modelo entrenado (**best.pt**) y el archivo de datos base de YOLOv8 (**yolov8n.pt**) se encuentren en el directorio raíz del servidor de modelos.

Si quieres probar otro modelo puedes cambiar el best.pt por el tuyo

6.3 Paso 3: Inicialización y Despliegue (Producción)

Para el despliegue en Render, se utilizan los archivos Procfile y requirements.txt.

Web Service Principal (Procfile): Utiliza Gunicorn para un inicio estable.

Comando en el Procfile del Web Service (app.py)

web: gunicorn app:app -w 4 -b 0.0.0.0:$PORT

Model Server (Procfile): Debe iniciarse en un puerto diferente (ej: 7860) y ser accesible desde el Web Service principal.

Comando en el Procfile del Model Server (inference.py)

web: gunicorn inference:app -w 2 -b 0.0.0.0:$PORT

Nota: En Render, la variable de entorno $PORT es asignada automáticamente por la plataforma.

6.4 Paso 4: Inicialización de la Base de Datos

Una vez que el Web Service principal está configurado con la DATABASE_URL, las tablas se crearán automáticamente al iniciar la aplicación, gracias al bloque de inicialización en app.py:

with app.app_context():

db.create_all() # Crea las tablas si no existen

Verificación: Confirme la conexión a la BD accediendo al panel de PostgreSQL y asegurándose de que las tablas propietario, vehiculos, reporte y usuario hayan sido creadas.