Nombres: Sebastian Garcia Rios, Adrian Jaret Rendon Rios

Numero de control: 20170669, 20170793

Carrera: Ingeniería en sistemas computacionales

Unidad:4

Materia: Tópicos de Inteligencia Artificial

Documento versión full (Manual de Usuario, Manual de Desarrolladores, Resumen del proyecto y sus objetivos).

Resumen del proyecto y sus objetivos

Sistema de detección de matriculas

El **objetivo** principal de este proyecto es desarrollar un sistema automatizado y eficiente capaz de detectar y reconocer matrículas de vehículos a partir de imágenes.

Este sistema no solo se limita a la lectura de la matrícula, sino que integra un componente crucial de vinculación con una base de datos centralizada para identificar y mostrar información relevante del propietario del vehículo.

La importancia de este sistema radica en su capacidad para agilizar procesos de identificación, mejorar la seguridad y facilitar la gestión vehicular en diversos contextos, como estacionamientos y controles de acceso.

Descripción del problema

La identificación manual de vehículos a través de sus matrículas es un proceso lento, propenso a errores humanos y difícil de escalar en escenarios de alto tráfico. Actualmente, muchas organizaciones y autoridades dependen de la verificación visual y la entrada manual de datos para registrar o validar vehículos, lo que resulta ineficiente y costoso.

Información de fondo: El aumento del parque vehicular y la necesidad de controles de seguridad más estrictos han creado una demanda urgente de soluciones tecnológicas que puedan procesar información vehicular de manera rápida y precisa. Los sistemas tradicionales de vigilancia a menudo carecen de la capacidad de interpretar texto (OCR) de manera efectiva en condiciones variables de iluminación y ángulo, y rara vez están integrados directamente con bases de datos de propietarios, lo que deja una brecha de información crítica.

Este proyecto aborda directamente esta necesidad mediante la implementación de un modelo de visión artificial robusto basado en redes neuronales convolucionales (CNN) y una arquitectura de base de datos relacional optimizada.

Justificación

La implementación de este sistema de detección de matrículas es de vital importancia por varias razones:

1. **Eficiencia en la Gestión de Tráfico y Accesos:** En el sector privado, facilita la automatización de entradas y salidas en estacionamientos y zonas residenciales, reduciendo tiempos de espera y mejorando la experiencia del usuario.
2. **Reducción de Errores y Costos:** Al automatizar la lectura y vinculación de datos, se eliminan los errores de transcripción manual y se reduce la necesidad de personal dedicado exclusivamente a estas tareas repetitivas.
3. **Impacto Social:** Contribuye a un entorno más ordenado y seguro, donde la tecnología actúa como un facilitador para el cumplimiento de normativas viales y la protección de la propiedad.
4. **Innovación Académica Aplicada:** Este proyecto es una demostración practica del talento técnico desarrollado en el tec de culiacan. Evidencia como los estudiantes pueden aplicar conocimientos avanzados de Inteligencia Artificial y desarrollo de software, para resolver problemas de su propio entorno inmediato.

Manual de Usuario detector de placas

Aplicación: App-Detector-de-placas

Propósito: Esta guía le enseñará cómo instalar y operar el sistema de detección de matrículas "App-Detector-de-placas" en su dispositivo móvil Android.

Cómo Instalar la Aplicación

Esta aplicación se distribuye a través de un archivo de instalación .apk que se encuentra en el repositorio del proyecto.

Pasos para instalar:

1. **Obtener el Archivo:**

- Vaya al repositorio oficial del proyecto en GitHub: https://github.com/JaretRendon/App-Detector-de-placas
- En el lado derecho de la página, busque la sección "Releases".
- Haga clic en la última versión y descargue el archivo app-debug.apk.

2. **Permitir Fuentes Desconocidas:**

- Para instalar un .apk fuera de la Google Play Store, su teléfono necesita un permiso especial.
- Vaya a Configuración > Seguridad en su teléfono.
- Busque la opción "Instalar aplicaciones desconocidas" o "Fuentes desconocidas" y actívela para su navegador o su explorador de archivos.

3. **Instalar la Aplicación:**

- Abra el archivo app-debug.apk que descargó.
- Su teléfono le preguntará si desea instalar la aplicaciónSeleccione "Instalar".
- Una vez completado, encontrará la app "DetectorPlacas" en su lista de aplicaciones.

2. Guía de Uso del Sistema

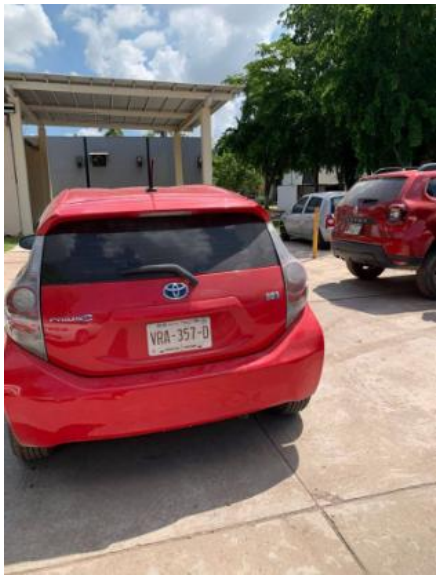
Siga estos pasos para identificar un vehículo:

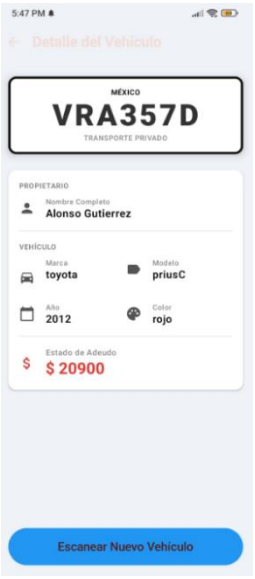
1. Abrir la Aplicación:
- Toque el ícono de "DetectorPlacas" en su menú de aplicaciones.
2. Otorgar permisos:
- La primera vez que abra la app, le pedirá permiso para usar la cámara.
- Debe seleccionar "Permitir" o "Mientras la app está en uso". Si deniega este permiso, la cámara no funcionará.
3. Enfocar la Matrícula:
- Se activará la cámara. Vera un recuadro blanco en el centro de la pantalla.
- Apunte la cámara de su teléfono hacia la matrícula del vehículo.
- Asegúrese de que la matricula completa queda dentro del recuadro blanco.

Escanear:

- Una vez que la placa esté bien alineada dentro del recuadro, presione el botón flotante de la cámara que se encuentra en la parte inferior de la pantalla.
4. Ver resultados:
- La aplicación analizara la imagen y buscara la placa en la base de datos.
- Si la placa se encuentra: Aparecerá una nueva pantalla de datos del propietario.

- Placa escaneada (VRA357D)
- Nombre del propietario (Alonso Gutierrez)
- Marca de vehículo (toyota)
- Modelo (PriusC)
- Año (2012)
- Color (Rojo)
- Estado de adeudo ($ 20900)





Si la placa NO se encuentra: La pantalla de resultados mostrará un mensaje "Error de datos, no se pudo vincular el propietario".

5. Escanear de Nuevo:

Para consultar otra placa, presione el botón "Escanear Nuevo Vehiculo" en la pantalla de resultados, o presione el botón "Atrás" (←) en la barra superior.

Aquí están algunas placas para que prueben el funcionamiento:

3.Solución de problemas comunes

¿Presiono el botón de la cámara, pero me dice "No se encontró placa en el recuadro"?

Solución: Esto significa que la cámara tomó la foto, pero la inteligencia artificial no pudo leer un número de placa claro.

Asegúrese de tener buena iluminación. Si es de noche o hay mucha sombra, la cámara no podrá leer.

Evite el movimiento. Mantenga el teléfono estable al presionar el botón para que la foto no salga borrosa.

Limpie el lente de la cámara de su teléfono.

Aléjese o acérquese para que la placa llene la mayor parte del recuadro blanco, sin salirse.

La aplicación indica "Placa no encontrada en la base de datos."

Solución: Esto significa que la app SÍ leyó la placa ("ABC1234"), pero ese número no existe en la base de datos central. Verifique que la matrícula que está escaneando sea una de las registradas.

La aplicación se cierra sola o no abre.

Solución 1: Intente cerrar la aplicación forzosamente (desde Configuración > Aplicaciones > DetectorPlacas > Forzar Cierre) y vuelva a abrirla.

Solución 2: Reinicie su teléfono.

Solución 3 (Permisos): Vaya a Configuración > Aplicaciones > DetectorPlacas > Permisos y asegúrese de que el permiso de "Cámara" esté concedido.

MANUAL PARA DESAROLLADORES

Propósito: Esta documentación proporciona detalles técnicos exhaustivos para desarrolladores, administradores de sistemas y evaluadores que deseen configurar, compilar, entender y extender el proyecto "Detector de Placas".

Requisitos previos

Dependencias de Software;

- Git: El sistema de control de versiones para clonar el repositorio.
- Android Studio (Hedgehog 2023.1.1 o superior): El IDE oficial para el desarrollo de Android.
- JDK 17: El Kit de Desarrollo de Java requerido por las versiones modernas de Android Gradle Plugin. (Generalmente incluido con Android Studio).
- Cuenta de Google Firebase: Se requiere una cuenta para configurar los servicios de backend (Base de Datos e IA).

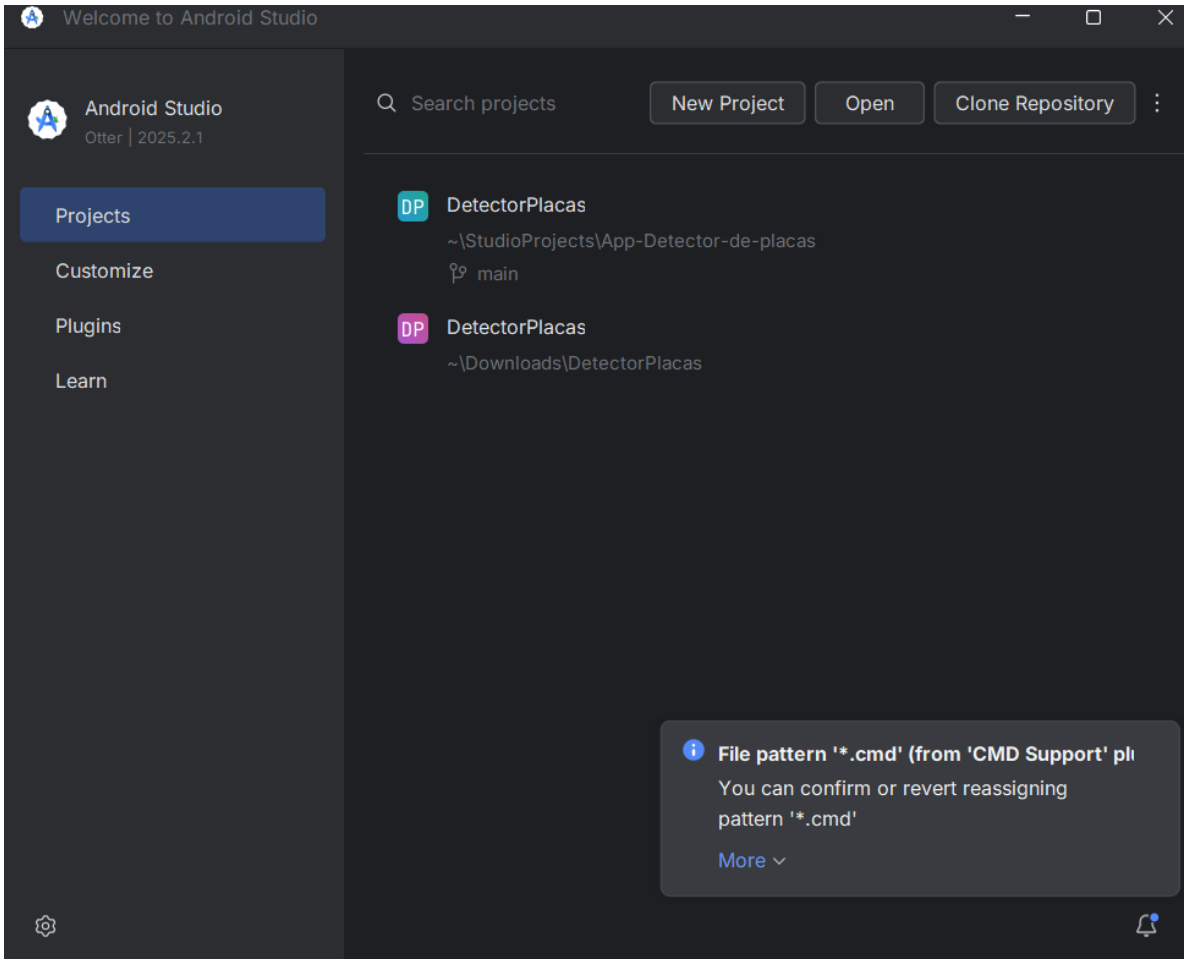
Comandos de Instalación y Ajustes

Realizar este seguimiento de pasos en orden, para la compilación de el proyecto:
Paso 1: Clonar el Repositorio Utilice Git para clonar el repositorio TopicosIA en una carpeta local.

git clone [https://github.com/JaretRendon/App-Detector-deplacas.git](https://github.com/JaretRendon/App-Detector-de-placas.git

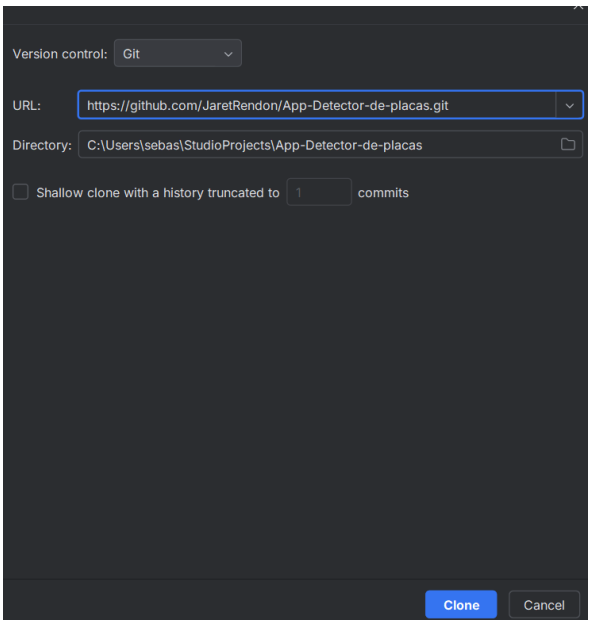
Paso 2: Abrir el Proyecto en Android Studio

1. Inicie Android Studio.



2. Seleccione clone repository

3. en URL inserte el: https://github.com/JaretRendon/App-Detector-de-placas.git



4. Android Studio detectará que es un proyecto de Gradle y comenzar a sincronizar lasa dependencis

Archivo local.properties: Este archivo contiene la ruta específica del SDK de Android en la computadora del desarrollador (sdk.dir=C:\Users\Usuario\AppData...). Por esta razón, **no se incluye en el repositorio de GitHub**.

- **Nota para los adminis o desarrolladores:** No es necesario crear este archivo manualmente. Al abrir el proyecto por primera vez, Android Studio detectará automáticamente la ubicación de su SDK y generará este archivo por usted.

Paso 3: Sincronizar y Ejecutar

En Android Studio, haga clic en Sync Project with Gradle Files si no lo hace automáticamente.

2. El proyecto se sincronizará y compilará sin errores, ya que el archivo google-services.json (ubicado en app/) está incluido.

3. Conecte un dispositivo Android físico (con Depuración USB activada) o inicie un Emulador de Android (Pixel 6 o otro).

4. Haga clic en el botón "Run 'app'" para compilar, instalar y ejecutar la aplicación.

5. La aplicación se conectará automáticamente a la base de datos de demostración de Firebase.

Paso 4: (Opcional) Probar la Base de Datos La base de datos de demostración contiene varias placas de ejemplo. Para probar la funcionalidad completa:

- Abra la app y escanee una de estas matrículas:
- VRA357D
- VHR434B
- VNG232D
- La aplicación debe reconocer la placa y mostrar los datos del propietario asociados desde Cloud Firestor

Especificaciones Técnicas

Esta sección detalla la arquitectura del sistema, el esquema de la base de datos y las API utilizadas que hacen funcionar el proyecto.

Arquitectura del Sistema

El proyecto está construido como una aplicación Android Nativa que implementa una arquitectura moderna Serverless (BaaS), desacoplando la lógica de la aplicación de la gestión de servidores.

Frontend (Cliente Android):

- Lenguaje: 100% Kotlin.
- Interfaz de Usuario (UI): Jetpack Compose. Se utiliza un paradigma de UI declarativo en lugar del sistema antiguo de XML.
- Navegación: Se emplea Navigation Compose (NavHost) para gestionar el flujo entre las dos pantallas principales (scanner y resultados) en una arquitectura de Actividad Única (MainActivity.kt).

Backend (Serverless - BaaS):

- Plataforma: Google Firebase. Se eligió esta plataforma para eliminar la necesidad de un servidor backend dedicado y para una rápida integración con el cliente Android.
- Base de Datos: Cloud Firestore.
- Inteligencia Artificial: Firebase ML kit.

Lógica de IA y Procesamiento (Flujo de MainActivity.kt):

- La API CameraX muestra el visor (AndroidView con PreviewView).

- Al presionar el botón (FloatingActionButton), se captura una imagen (cameraController.takePicture).

- La imagen completa se guarda en un archivo temporal.

- El Bitmap se carga desde ese archivo y se realiza una optimización clave: la imagen se recorta en el dispositivo (Bitmap.createBitmap) basándose en las dimensiones de la retícula visual (el recuadro blanco).

- Este bitmap recortado (que solo contiene la placa) se envía a la API ML Kit (Text Recognition), mejorando drásticamente la precisión y velocidad del OCR.

- El texto devuelto (visionText) se limpia y valida en la función buscarPlaca usando una Expresión Regular

- Con la placa validada, la app navega a PantallaResultados.

- PantallaResultados usa LaunchedEffect para ejecutar una consulta a Cloud Firestore, buscando un documento cuyo ID sea la placa.

- La UI reacciona al estado de la búsqueda (cargando, exito, no_encontrado) y muestra los resultados.

Esquema de la Base de Datos

Se utiliza Cloud Firestore, una base de datos NoSQL basada en documentos. Para maximizar la eficiencia, se implementó un diseño de consulta de un solo documento.

- Servicio: Cloud Firestore
- Colección Principal: vehiculos
- Diseño de la Colección vehiculos:
- Para optimizar la velocidad de lectura, el ID del Documento es la propia matrícula del vehículo (en mayúsculas y sin guiones/espacios).
- ID del Documento (Ejemplo): VLA551A
- Campos del Documento (Ejemplo):
- nombre (String): "Sebastian Garcia "
- modelo (String): "Chevrolet Spark "
- adeudos (Any): 15000 (Number) "Ninguno" (String) (El data class Propietario en Kotlin maneja este tipo flexible).
- Justificación de este diseño: Este esquema es extremadamente rápido. Permite verificar la existencia de una placa y obtener toda la información del propietario con una única operación de lectura (db.collection("vehiculos").document(placaId).get()).

APIs y Librerías Utilizadas

El proyecto depende de un conjunto de APIs modernas de Android y Google, definidas en el archivo build.gradle.kts (Module :app):

Google Firebase (BaaS):

- com.google.firebase:firebase-firestore-ktx: API de Cloud Firestore para la persistencia y consulta de datos. O
- com.google.android.gms:play-services-mlkit-text-recognition: API de ML Kit para el Reconocimiento Óptico de Caracteres (OCR) en el dispositivo.
- Android Jetpack (CameraX): androidx.camera:camera-camera2, camera-lifecycle, camera-view: Conjunto de APIs de CameraX para un control moderno y robusto de la cámara, gestionando el ciclo de vida automáticamente.

Android Jetpack (UI):

- androidx.compose.ui, material3, foundation: El kit de herramientas de Jetpack Compose para construir la UI declarativa.
- androidx.activity:activity-compose: Proporciona el puente (setContent) entre la Actividad y Compose.
- androidx.navigation:navigation-compose: Proporciona el NavHost para la navegación entre composables.

Threading (Asincronía):

- java.util.concurrent.Executors: Se utiliza newSingleThreadExecutor para manejar las operaciones de captura de imagen (I/O) fuera del hilo principal, previniendo que la UI se congele.
- androidx.compose.runtime.LaunchedEffect: Se utiliza para ejecutar la consulta de red a Firestore de forma asíncrona cuando la pantalla PantallaResultados se carga.