SISTEMA INTELIGENTE DE CONTROL DE ACCESO VEHICULAR EN EL CONJUNTO CERRADO AMPARO COUNTRY EN LA CIUDAD DE VALLEDUPAR-CESAR POR MEDIO DE IoT.

PROYECTO DE DESARROLLO TECNOLÓGICO

HERMES DAVID RAMOS RAMIREZ

YENIS PAOLA PLATA LÓPEZ

PROGRAMA DE INGENIERÍA ELECTRÓNICA

FACULTAD DE INGENIERÍA Y TECNOLÓGICAS

UNIVERSIDAD POPULAR DEL CESAR

VALLEDUPAR- CESAR

2025-2

SISTEMA INTELIGENTE DE CONTROL DE ACCESO VEHICULAR EN EL CONJUNTO CERRADO AMPARO COUNTRY EN LA CIUDAD DE VALLEDUPAR-CESAR POR MEDIO DE IoT

HERMES DAVID RAMOS RAMIREZ

YENIS PAOLA PLATA LÓPEZ

PROYECTO DE DESARROLLO TECNOLÓGICO DE GRADO PRESENTADO AL COMITÉ EVALUADOR DE PROYECTOS DE GRADO DEL PROGAMA DE INGENIERÍA ELECTRÓNICA DE LA UNIVERSIDAD POPULAR DEL CESAR

DIRECTOR

ING. JUSTO ELIAS MONTENEGRO

Ingeniero Electrónico

COODIRECTORA:

ING. YAILETH MORALES DAZA

PROGRAMA DE INGENIERÍA ELECTRÓNICA

FACULTAD DE INGENIERÍA Y TECNOLÓGICAS

UNIVERSIDAD POPULAR DEL CESAR

VALLEDUPAR- CESAR

2025-2

DEDICATORIA

En primer lugar, agradecemos a Dios, quien ha sido nuestra luz y nuestra fuerza en cada paso de este camino. Su guía nos sostuvo en los momentos difíciles y nos permitió seguir adelante cuando las dudas o el cansancio intentaron detenernos. Cada avance, cada aprendizaje y cada logro de este proyecto lleva también Su huella.

A nuestras familias, quienes han sido el motor silencioso que impulsa nuestros sueños. Gracias por su amor incondicional, por la paciencia con la que acompañaron nuestras largas jornadas, y por el sacrificio que muchas veces hicieron sin pedir nada a cambio. Ustedes son el ejemplo más claro de perseverancia y entrega, y este logro es tan suyo como nuestro. Gracias por creer en nosotros incluso cuando nosotros mismos titubeamos.

También extendemos nuestra gratitud a todas las personas que, de una forma u otra, hicieron parte de este proceso: docentes, compañeros, amigos y quienes, con un consejo, una orientación o una palabra de ánimo nos ayudaron a avanzar. Cada gesto, por pequeño que pareciera, sumó fuerzas y marcó nuestro camino.

A todos ustedes, gracias. Este proyecto no solo refleja conocimiento y esfuerzo, sino también el apoyo, el cariño y la fe de quienes nos rodean. Nos sentimos verdaderamente bendecidos de haber contado con su compañía en esta etapa tan importante de nuestra formación académica y personal.

AGRADECIMIENTOS

Queremos expresar nuestros más profundos agradecimientos primeramente a Dios, quien ha sido nuestra guía y fortaleza en cada paso de este camino. Su presencia nos recordó siempre que todo esfuerzo tiene su recompensa.

A nuestros padres, queremos dedicarles un agradecimiento especial. Ustedes han sido nuestro ejemplo, nuestro refugio y el motor que nos impulsó a seguir adelante. Con su amor incondicional, paciencia y sacrificio, nos enseñaron que los sueños se alcanzan con dedicación y esfuerzo. Este logro es tanto nuestro como suyo, porque sin su apoyo nada de esto hubiera sido posible.

A nuestra familia en general, gracias por acompañarnos con su cariño y comprensión. Cada palabra de ánimo y cada gesto de apoyo se convirtió en fuerzas que nos levantaron en los momentos de cansancio. Ustedes nos recordaron que no estábamos solos en este proceso y que siempre había un motivo para continuar.

También queremos agradecer a los docentes de la |Universidad Popular del Cesar, quienes compartieron generosamente sus conocimientos y nos motivaron a crecer tanto académica como personalmente. En especial, expresamos nuestra gratitud al ingeniero Justo Elías Montenegro y a la ingeniera Yaileth Morales Daza, quienes, como director y codirectora de este proyecto de grado, nos brindaron su tiempo, orientación y apoyo en cada etapa. Sus enseñanzas han dejado una huella imborrable en nuestra formación profesional.

Finalmente, a todas las personas que, de una u otra forma, nos tendieron la mano en este proceso, queremos decirles gracias. Cada uno contribuyó a que este sueño se hiciera realidad y por ello guardamos un sincero y eterno agradecimiento.

ÍNDICE GENERAL

LISTA DE FIGURA

LISTA DE TABLAS

1. LÍNEA DE INVESTIGACIÓN

Proyecto de desarrollo Tecnológico enfocado en la línea de Energía, automática e Instrumentación electrónica y la sub-línea de Automatización y Control. definidas en el acuerdo No 003 del 08 de julio de 2021 “POR MEDIO DEL CUAL SE ADOPTAN LAS LÍNEAS DE INVESTIGACIÓN DE LOS PROGRAMAS DE PREGRADO DE LA FACULTAD DE INGENIERÍA Y TECNOLÓGICAS SEDE VALLEDUPAR, Y SE DICTAN OTRASDISPOSICIONES”, emanado por el honorable Consejo de Facultad de la facultad de Ingeniería y Tecnológicas de la Universidad Popular del Cesar.

2.RESUMEN

El presente proyecto desarrolla un sistema inteligente de control de acceso vehicular para el conjunto cerrado Amparo Country, ubicado en Valledupar-Cesar, empleamos tecnologías basadas en Internet de las Cosas (IoT), visión artificial y arquitectura web moderna. Su propósito es solucionar las dificultades actuales relacionadas con el control manual de vehículos, la falta de seguridad, la ausencia de registro automatizado y la limitada gestión administrativa del conjunto.

La investigación tuvo como objetivo principal diseñar e implementar un sistema capaz de identificar vehículos, registrar accesos, monitorear parqueaderos y optimizar la administración, todo desde una plataforma integrada. Metodológicamente, se desarrolló un modelo de detección de matrículas basado en YOLOv8, complementado con OCR, procesamiento de imágenes y comunicación con un backend construido en FastAPI, sincronizado con Firebase para la gestión de datos en tiempo real. El proceso incluyó entrenamiento de modelos, pruebas de validación, simulaciones en condiciones reales y el diseño de interfaces web y móviles.

Entre los resultados, se obtuvo un sistema robusto que identifica matrículas con alta precisión, reduce los tiempos de ingreso, mejora la seguridad mediante validaciones automáticas y permite un monitoreo en tiempo real del estado de los estacionamientos. Además, facilita la trazabilidad de residentes, pagos y permisos. Como conclusión, se demuestra que la integración de IoT, visión artificial e infraestructura en la nube sí mejora significativamente la seguridad**,** la eficiencia administrativa y la experiencia de los residentes, aportando una solución moderna, escalable y adaptable a otros conjuntos residenciales.

Palabras clave: IoT, control de acceso vehicular, visión artificial, reconocimiento de matrículas, seguridad residencial.

3. ABSTRACT

This project presents the development of an intelligent vehicular access control system for the gated community Amparo Country in Valledupar, Cesar, using Internet of Things (IoT) technologies, computer vision, and modern web architecture. Its purpose is to address the limitations of traditional manual control methods, which affect security, efficiency, and administrative management within the residential complex.

The main objective of the research was to design and implement an automated system capable of identifying vehicles, recording access events, monitoring parking spaces, and improving administrative processes through an integrated digital platform. The methodology included the development and training of a license plate detection model based on YOLOv8, complemented by Optical Character Recognition (OCR), image processing techniques, and real-time communication with a FastAPI backend connected to Firebase for cloud data management. The process involved model training, validation tests, scenario simulations, and the creation of both web and mobile user interfaces.

The results demonstrate a robust system capable of achieving high-precision license plate detection, reducing vehicle entry times, enhancing security through automated validations, and enabling real-time monitoring of parking availability. Additionally, the system improves recordkeeping for residents, payments, and access permissions. In conclusion, the integration of IoT, computer vision, and cloud-based architecture significantly enhances safety, operational efficiency, and resident experience, providing a scalable and modern solution suitable for other residential environments.

Keywords: IoT, vehicular access control, computer vision, license plate recognition, residential security.

4. INTRODUCCION

En los conjuntos residenciales modernos, la seguridad y la eficiencia en el control vehicular se han convertido en necesidades fundamentales. El crecimiento urbano, el aumento del flujo de residentes y visitantes, y la complejidad administrativa de los espacios compartidos demandan soluciones tecnológicas capaces de reemplazar los métodos manuales que hoy resultan insuficientes. En el conjunto cerrado Amparo Country, estas limitaciones se evidencian en procesos lentos, ausencia de verificación automatizada, riesgos de ingresos no autorizados y dificultades para gestionar información clave como pagos, permisos y disponibilidad de parqueaderos.

A lo largo de los últimos años, diversas investigaciones han demostrado el potencial de la visión computacional, el reconocimiento óptico de matrículas (OCR) y los sistemas basados en Internet de las Cosas (IoT) para transformar la forma en que se controla el acceso a espacios privados. Tecnologías como YOLO, redes neuronales, procesamiento de imágenes en tiempo real y plataformas en la nube han evolucionado hasta permitir sistemas precisos, ágiles y totalmente integrados. Estos avances representan una oportunidad para implementar soluciones inteligentes que superen las limitaciones de los métodos tradicionales y fortalezcan la seguridad residencial.

Este proyecto nace justamente de esa necesidad: crear un sistema inteligente de control de acceso vehicular que automatice la identificación de vehículos, registre entradas y salidas, y facilite la gestión administrativa del conjunto. El desarrollo integra visión artificial de última generación, arquitectura web moderna, tecnologías IoT, bases de datos en la nube y una interfaz pensada para administradores, vigilantes y residentes. Además, se incluyeron etapas de entrenamiento, prueba y validación del modelo de detección, garantizando que la solución funcione en condiciones reales: diferentes iluminaciones, ángulos, distancias y tipos de placas.

El aporte de este trabajo va más allá de la simple automatización. Representa una alternativa confiable, escalable y adaptable para mejorar la seguridad comunitaria, optimizar tiempos, reducir errores humanos y brindar a los residentes un entorno más organizado, seguro y tecnológicamente actualizado. Desde esta perspectiva, el proyecto se enmarca en la línea de automatización y control, y busca contribuir al avance de sistemas inteligentes aplicados a la vida cotidiana.

5. PLANTEAMIENTO DEL PROBLEMA

El crecimiento urbano y la necesidad de mejorar la seguridad en los conjuntos cerrados han generado problemas comunes, como la falta de integración tecnológica en la gestión de accesos vehiculares y la administración de recursos. El conjunto cerrado Amparo Country en Valledupar, Cesar, enfrenta desafíos significativos ya que los métodos tradicionales de control, como la vigilancia manual, presentan limitaciones en términos de seguridad, rapidez y comodidad, estos no siempre garantizan que solo los vehículos autorizados ingresen, lo que representa un riesgo para los residentes.

Además, no existe un control automatizado de la información financiera relacionada con los propietarios, como el estado de pagos de la administración, lo que dificulta la gestión administrativa. También hay problemas en la distribución y disponibilidad de los parqueaderos, lo que puede generar inconvenientes tanto para residentes como para visitantes [1].

El control manual del acceso genera tiempos de espera prolongados, especialmente en horas pico, causando molestias y afectando la fluidez del ingreso vehicular. Por lo tanto, el problema radica en la falta de un sistema inteligente de control vehicular que no solo permita la identificación automática de los vehículos autorizados, sino que también facilita la caracterización de las viviendas y el control eficiente de los parqueaderos.

¿Puede un sistema inteligente de control de acceso vehicular basado en IoT mejorar la seguridad, la gestión administrativa y facilitar el acceso en el conjunto cerrado Amparo Country, en comparación con los métodos tradicionales?

6. JUSTIFICACIÓN

El sistema inteligente de control de acceso vehicular basado en IoT es una respuesta integral a las necesidades de seguridad y eficiencia en el conjunto cerrado Amparo Country. Una de las principales mejoras que ofrece este sistema es un control más riguroso de los accesos, reduciendo significativamente la posibilidad de ingresos no autorizados mediante el uso de sensores, cámaras y monitoreo automatizado. Estas herramientas garantizan la autenticidad de los vehículos y personas que acceden, aumentando la percepción de seguridad entre los residentes y generando un entorno más confiable [2].

Además, la plataforma de monitoreo en tiempo real permitirá a los administradores del conjunto y a los propios residentes tener acceso a información actualizada sobre el ingreso y salida de vehículos. Esto facilita la toma de decisiones rápidas ante posibles incidentes, mejorando la reacción ante situaciones críticas y permitiendo un control más efectivo de los accesos [3].

Por lo tanto, este sistema no solo mejora la seguridad y eficiencia operativa, sino que también contribuye a optimizar la gestión de los recursos del conjunto cerrado, elevando la calidad de vida de los residentes y brindando un entorno más seguro y ágil para la comunidad.

7. OBJETIVOS

7.1. OBJETIVO GENERAL

- Desarrollar un sistema inteligente de control de acceso vehicular en el conjunto cerrado Amparo Country en la ciudad de Valledupar-Cesar por medio de IoT.

7.2. OBJETIVOS ESPECÍFICOS

- Implementar un sistema IoT para detectar la ocupación de áreas de estacionamiento y registrar el acceso de vehículos de forma eficiente.
- Desarrollar una interfaz web de administración que permita gestionar el control de accesos, configurar el sistema, monitorear reportes históricos de entradas/salidas y llevar un registro preciso del estado de pagos de los residentes.
- Desarrollar una aplicación móvil que permita a los residentes y administradores del conjunto cerrado monitorear en tiempo real el flujo vehicular, el estado de las plazas de estacionamiento y la seguridad de los accesos.
- Capacitar a los usuarios (residentes, administración y personal de seguridad) en el uso de la aplicación y del sistema de control, facilitando una adopción eficiente de la solución tecnológica.

8. ESTADO DEL ARTE

El avance tecnológico en las últimas décadas ha transformado significativamente diversos campos, incluida la gestión de accesos y la seguridad en conjuntos residenciales. Los sistemas de control de acceso vehicular han evolucionado con la incorporación de tecnologías como el Internet de las Cosas (IoT), la visión artificial y el procesamiento de datos en la nube. Este cambio ha generado nuevas oportunidades para la automatización y optimización de la seguridad en comunidades cerradas.

En los últimos años, diversas investigaciones han abordado la optimización del control de acceso vehicular mediante tecnologías avanzadas. En abril de 2023, fue realizada la investigación por los especialistas en materia: Luis Martínez, Andrea Gómez y Roberto Sánchez, en su tesis titulada "**Control de acceso vehicular mediante machine learning**", la cual propone el desarrollo de un sistema basado en reconocimiento de patrones para la identificación automática de placas vehiculares mediante técnicas de aprendizaje automático. A través del uso de modelos de machine learning, el sistema analiza y reconoce matrículas con una precisión del 95%, optimizando el control de ingreso de vehículos en instalaciones universitarias. La investigación también aborda la integración de un módulo de detección de anomalías que mejora la seguridad al identificar matrículas no registradas o alteradas, garantizando así un acceso vehicular más eficiente y seguro [4]

En marzo de 2023, fue realizada la investigación por los especialistas en materia: Andrés Bernuy Alva, Guillermo Zambrano Loli, Beatriz Morón Casana y Patricia Rodríguez Zeta, en su tesis titulada "**Sistema de reconocimiento de placas vehiculares para identificar vehículos con requisitoria utilizando inteligencia artificial**", la cual presenta un algoritmo avanzado de procesamiento de imágenes para la detección de matrículas vehiculares. Mediante técnicas de segmentación de caracteres y clasificación con inteligencia artificial, el sistema convierte automáticamente las imágenes en texto digital, logrando una precisión del 97.5% en la identificación de vehículos con requisitoria en vías públicas. La implementación de este sistema contribuye a mejorar la seguridad vial y la eficiencia en el control de vehículos sospechosos en distintos puntos estratégicos [5].

En junio de 2022, fue realizada la investigación por los especialistas en materia: Carolina Ramírez, David Méndez y Juan Carlos Torres, en su tesis titulada "**Sistema automático de reconocimiento de placas vehiculares**", la cual desarrolla un sistema basado en reconocimiento óptico de caracteres (OCR) para mejorar el control y la seguridad en accesos vehiculares. Utilizando técnicas avanzadas de procesamiento de imágenes y aprendizaje profundo, el sistema permite una lectura rápida y precisa de las matrículas vehiculares, reduciendo los tiempos de validación en un 40%. La investigación también explora la integración con bases de datos gubernamentales para verificar en tiempo real el estatus legal de los vehículos detectados [6].

En agosto de 2023, fue realizada la investigación por los especialistas en materia: Sebastián Viteri y María Fernanda Castro, en su tesis titulada "**Sistema de identificación de placas automotrices para la Universidad Metropolitana de Ecuador**", la cual implementa un sistema de reconocimiento de matrículas vehiculares basado en inteligencia artificial. La investigación propone el uso de redes neuronales convolucionales para optimizar la lectura de placas de automotores que ingresan a los estacionamientos de la universidad, logrando una precisión del 96%. Además, se desarrolla una interfaz de usuario que permite la gestión en tiempo real de los vehículos autorizados, mejorando la eficiencia en la administración de los espacios de estacionamiento y reforzando la seguridad del campus [7].

En diciembre de 2022, fue realizada la investigación por los especialistas en materia: Edwin Wildor Pérez Silva, en su tesis titulada "**Reconocimiento de placas vehiculares mediante visión computacional**", la cual presenta un sistema basado en visión artificial para la identificación de matrículas vehiculares en entornos urbanos. La investigación desarrolla un algoritmo que, mediante técnicas de reconocimiento óptico de caracteres (OCR) y segmentación de imágenes, permite extraer y convertir los caracteres de una matrícula en texto digital con una precisión del 96%. Se emplearon modelos de aprendizaje profundo y procesamiento de imágenes en tiempo real, optimizando el sistema para operar en diversas condiciones de iluminación y ángulos de captura. Los resultados obtenidos evidenciaron una mejora en los tiempos de procesamiento y en la precisión de reconocimiento en comparación con sistemas convencionales [8]

9. MARCO TEORICO

En este apartado se presentan los fundamentos teóricos y tecnológicos que sustentan el desarrollo del sistema inteligente de control de acceso vehicular. Se aborda el papel del Internet de las Cosas (IoT) en la interconexión de dispositivos para la recolección y análisis de datos en tiempo real, garantizando la automatización y seguridad del proceso.

Asimismo, se introducen las herramientas de visión artificial y reconocimiento óptico de caracteres (OCR), que permiten la detección y lectura automática de placas vehiculares mediante el uso de modelos como YOLOv8 y librerías como OpenCV y Pytesseract. Además, se destacan las tecnologías empleadas en el desarrollo del sistema, como FastAPI y Uvicorn para la gestión del backend, Astro y React en el frontend, y PostgreSQL para el almacenamiento de datos en la nube. Finalmente, se subraya la importancia de Docker para el despliegue de contenedores y la portabilidad del sistema, conformando un entorno integral que combina inteligencia artificial, IoT y desarrollo web moderno para lograr un control vehicular eficiente y automatizado.

9.1. Internet de las Cosas (IoT)

El Internet de las Cosas (IoT, por sus siglas en inglés) es un paradigma tecnológico que permite la conexión e interacción de dispositivos físicos a través de redes de comunicación, principalmente internet, para recopilar, enviar y analizar datos en tiempo real. Como se muestra en la Figura 1, estos dispositivos pueden incluir sensores, actuadores, cámaras, tarjetas electrónicas, entre otros, que se comunican sin intervención humana directa, generando sistemas inteligentes distribuidos.
La arquitectura del IoT se basa en tres capas fundamentales: la capa de percepción (donde se capturan los datos mediante sensores), la capa de red (que transporta la información), y la capa de aplicación (donde los datos se procesan y se convierten en acciones o decisiones). Gracias a este enfoque, el IoT se ha consolidado como una herramienta clave en sectores como la salud, la industria, la agricultura y la gestión urbana [9].

FIGURA 1: INTERNET DE LAS COSAS (IOT) [10]

9.2. Reconocimiento Óptico de Caracteres (OCR)

El Reconocimiento Óptico de Caracteres (OCR) es una tecnología que permite identificar y extraer texto desde imágenes o documentos escaneados, convirtiéndolos en datos digitales editables y procesables por sistemas computacionales. Su aplicación es fundamental en sistemas inteligentes donde se requiere identificar información visual de manera automatizada, como el reconocimiento de matrículas vehiculares en tiempo real [11].

9.2.1 Google Cloud OCR

Google Cloud OCR ha sido seleccionado para este proyecto debido a su capacidad de adaptación a distintos entornos, su precisión en condiciones de iluminación variable y su integración con servicios en la nube, facilitando el procesamiento remoto de datos. [12].

9.2.2 Tesseract OCR

Tesseract OCR es un motor de reconocimiento de texto de código abierto que ofrece soporte para múltiples idiomas y se puede entrenar con datos personalizados. Su arquitectura basada en redes neuronales profundas le permite obtener resultados precisos en ambientes controlados. Es ideal para sistemas que requieren reconocimiento local, sin depender de conexión a internet o servicios en la nube. Tesseract puede integrarse fácilmente con bibliotecas de visión como OpenCV para mejorar la calidad de las imágenes antes del análisis [13].

9.2.3 OpenCV (Open-Source Computer Vision Library)

OpenCV es una biblioteca de visión artificial ampliamente utilizada para el procesamiento y análisis de imágenes y video. Si bien no realiza Reconocimiento Óptico de Caracteres (OCR) por sí misma, se emplea como complemento para mejorar la legibilidad de texto antes del reconocimiento, mediante técnicas como el escalado, la binarización, la detección de bordes y la reducción de ruido. Como se muestra en la Figura 2, OpenCV puede utilizarse para detectar y aislar regiones de interés dentro de una imagen, como las placas de vehículos, facilitando posteriormente la extracción y reconocimiento de caracteres mediante herramientas. OpenCV es de código abierto, multiplataforma y ofrece una integración eficaz con lenguajes como Python, Java y C++ [14].

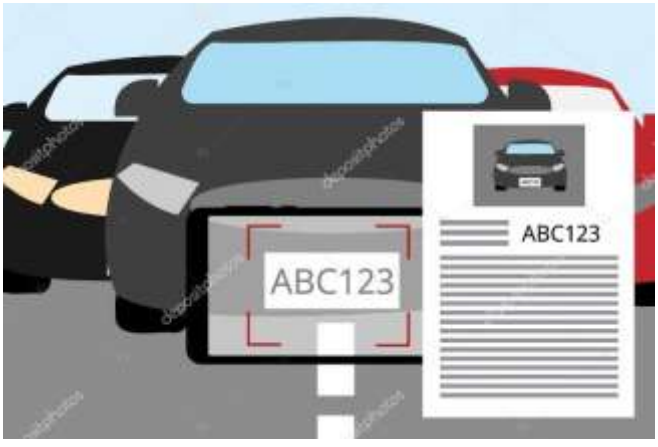


FIGURA 2: RECONOCIMIENTO ÓPTICO DE CARACTERES (OCR) [15]

9.3. Bases de Datos

Las bases de datos son sistemas diseñados para almacenar, organizar, recuperar y gestionar información estructurada. En el contexto de sistemas inteligentes, permiten el almacenamiento seguro y confiable de datos operativos, como historiales de acceso, usuarios autorizados, pagos administrativos, disponibilidad de parqueaderos, entre otros. La base de datos se convierte en el eje de integración entre los diferentes módulos del sistema: sensores, aplicación móvil, interfaz web y servidor, garantizando una administración eficiente y centralizada de la información [16].

9.3.1. Firebase

Es una plataforma de desarrollo en la nube proporcionada por Google LLC que ofrece un conjunto integrado de servicios backend para aplicaciones móviles y web, tales como bases de datos en tiempo real, autenticación de usuarios, almacenamiento, mensajería en la nube y análisis. Como se muestra en la Figura 3, Firebase centraliza la gestión y sincronización de datos entre múltiples dispositivos y aplicaciones en tiempo real. Esta plataforma permite a los desarrolladores construir y desplegar aplicaciones sin necesidad de gestionar servidores tradicionales, aprovechando su escalabilidad automática, sincronización de datos en tiempo real entre dispositivos y facilidad de integración multiplataforma. Es decir, Firebase simplifica enormemente el desarrollo, la gestión y el escalado de aplicaciones modernas en la nube [17].

FIGURA 3: BASE DE DATOS [18]

9.4. Bibliotecas

Una **biblioteca de software** es un conjunto de funciones, rutinas o clases ya desarrolladas que pueden ser utilizadas por otros programas para facilitar tareas específicas. Su objetivo es reutilizar código probado, reducir tiempos de desarrollo y garantizar mayor confiabilidad en los sistemas. Las bibliotecas pueden abarcar desde operaciones matemáticas básicas hasta procesos más complejos como seguridad, comunicación en red o autenticación [19].

9.4.1 Tailwind CSS

Es un **framework de diseño web** basado en utilidades que permite crear interfaces de usuario de manera rápida y flexible. A diferencia de otros frameworks que traen estilos predefinidos, Tailwind ofrece una colección de clases listas para aplicar directamente en el HTML, lo que

facilita personalizar colores, tamaños, tipografías y diseños sin necesidad de escribir hojas de estilo largas. [20] En otras palabras, es como una **caja de herramientas de diseño** que acelera la creación de páginas web modernas, responsivas y visualmente atractivas, sin limitar la creatividad del desarrollador.

9.4.2 Astro Icon

Es una biblioteca que facilita la **inclusión de íconos en proyectos web desarrollados con Astro**. En lugar de descargar y configurar manualmente cada ícono, esta herramienta permite integrar fácilmente colecciones de íconos conocidos (como Material Icons, Heroicons, FontAwesome, entre otros) dentro de las plantillas y componentes [21]. En términos simples, Astro Icon funciona como un **catálogo de íconos listo para usar**, lo que ahorra tiempo a los desarrolladores y asegura que las páginas tengan una apariencia más clara, atractiva y profesional.

9.5. Framework

Es una plataforma de desarrollo que proporciona un conjunto de componentes reutilizables, librerías y directrices que facilitan la creación de aplicaciones de software de manera más eficiente y estructurada. Su propósito principal es ofrecer una **base sólida y estandarizada** sobre la cual los desarrolladores pueden construir, asegurando buenas prácticas y reduciendo el tiempo de programación al reutilizar funcionalidades ya implementadas. A diferencia de una biblioteca, que el programador invoca cuando lo necesita, en un framework el control está invertido: es el propio framework el que dirige el flujo de ejecución y establece la estructura del proyecto. En síntesis, un framework funciona como un **andamiaje de apoyo** que guía y acelera el desarrollo, garantizando orden, escalabilidad y mantenibilidad en el software [22]

.

9.6. Astro

Es un framework de desarrollo web orientado a la construcción de sitios rápidos y optimizados, con un enfoque en la generación de contenido estático y la reducción del uso innecesario de JavaScript en el cliente. Su arquitectura permite integrar múltiples tecnologías y librerías de frontend (como React, Vue o Svelte), lo que lo convierte en una herramienta flexible para proyectos modernos. [23] En términos sencillos, Astro actúa como un **ensamblador de páginas web eficientes**, combinando lo mejor de diferentes entornos para ofrecer un alto rendimiento y una mejor experiencia de usuario.

9.7. Lenguaje de programación

Un lenguaje de programación es un sistema formal diseñado para expresar instrucciones que una computadora puede interpretar y ejecutar. Estos lenguajes proporcionan una sintaxis y una semántica específicas que permiten a los programadores describir algoritmos, manipular datos y controlar el comportamiento de los sistemas de software. Existen diferentes niveles de lenguajes: los de bajo nivel, cercanos al lenguaje máquina, y los de alto nivel, que ofrecen mayor abstracción y facilitan la comprensión por parte del ser humano. [24] En esencia, los lenguajes de programación constituyen el medio principal de comunicación entre el programador y la computadora, posibilitando el desarrollo de aplicaciones, sistemas y soluciones tecnológicas en diversos ámbitos.

9.7.1 Python

Es un lenguaje de programación interpretado, de alto nivel y multiparadigma, conocido por su sintaxis clara y legible que facilita el aprendizaje y el desarrollo rápido de aplicaciones. Ofrece soporte para programación orientada a objetos, funcional y procedural, además de contar con una amplia colección de bibliotecas para inteligencia artificial, análisis de datos, desarrollo web y automatización. Gracias a su versatilidad, Python se ha consolidado como uno de los lenguajes más utilizados tanto en la industria como en la investigación académica [25].

9.8. Librerías de Python

Python se ha consolidado como uno de los lenguajes más utilizados en el ámbito de la inteligencia artificial y el Internet de las Cosas (IoT), debido a la amplia variedad de librerías que ofrece para facilitar el desarrollo de soluciones complejas. En el caso de nuestro proyecto, que busca implementar un sistema inteligente de control de acceso vehicular, estas librerías se convierten en la base tecnológica que permite integrar procesos de visión artificial, comunicación entre servicios y gestión de datos de manera eficiente. A continuación, se presentan las principales librerías utilizadas:

- **FastAPI**
  Es un framework que permite desarrollar interfaces de programación de aplicaciones (APIs) de manera rápida y segura. Gracias a su diseño asíncrono, favorece el manejo de múltiples solicitudes en tiempo real, lo que resulta útil en sistemas que requieren comunicación constante entre dispositivos y servidores. [26]
- **Uvicorn**
  Funciona como un servidor liviano que ejecuta aplicaciones construidas con FastAPI. Su papel principal es garantizar la estabilidad y rapidez en la comunicación entre los diferentes módulos del sistema, lo que contribuye a que el proyecto pueda responder con eficiencia a los usuarios. [27]
- **Requests**
  Es una librería que simplifica la comunicación con servicios externos a través de solicitudes HTTP. En este proyecto se utiliza, por ejemplo, para enviar imágenes a la API de reconocimiento de caracteres (OCR) y obtener resultados de manera sencilla. [28]
- **Python-dotenv**
  Esta librería permite manejar de manera segura las variables de entorno, evitando que credenciales o claves de acceso queden expuestas en el código fuente. Esto resulta especialmente importante en proyectos de IoT, donde la protección de información sensible es prioritaria. [29]
- **Python-multipart**
  Es una herramienta diseñada para procesar datos en formato multipart/form-data, lo que facilita la recepción de archivos como imágenes desde formularios o aplicaciones web. En el proyecto resulta clave para manejar las capturas enviadas por las cámaras de vigilancia. [30]
- **OpenCV (cv2)**
  Es una librería de visión por computadora ampliamente utilizada para procesar imágenes y videos. En este proyecto permite detectar las coordenadas de las placas, recortarlas y mejorarlas antes de enviarlas al modelo de detección o al servicio de OCR. [31]
- **Pytesseract**

Pytesseract conecta Python con el motor de reconocimiento óptico de caracteres Tesseract. Su función es extraer texto a partir de imágenes, lo que posibilita identificar la información contenida en las placas vehiculares. [32]

- **PyTorch**
  Es un framework de aprendizaje profundo que facilita la creación y entrenamiento de redes neuronales. En este proyecto es la base sobre la que se entrena el modelo de detección, permitiendo optimizar su rendimiento y precisión. [33]
- **Ultralytics / YOLOv8**
  Ultralytics ofrece la implementación del modelo YOLOv8, especializado en detección de objetos en tiempo real. En este caso, se entrenó un modelo específico para identificar vehículos y placas, obteniendo un archivo optimizado de pesos (best.pt) para la fase de producción. [34]
- **NumPy**
  Es la librería base para realizar operaciones matemáticas con matrices y arreglos multidimensionales. Su uso es esencial en el preprocesamiento de datos y en las fases de entrenamiento del modelo. [35]
- **Docker**
  Docker permite encapsular la aplicación completa (modelo, librerías y dependencias) en contenedores, lo que garantiza que el sistema pueda ejecutarse en distintos entornos sin errores de compatibilidad. Esta herramienta asegura que el sistema no quede limitado al laboratorio y pueda desplegarse en producción. [36]
- **API OCR.Space**
  Debido a las dificultades para contar con un dataset completo de placas, se decidió integrar la API de OCR.Space. Este servicio externo facilita la lectura confiable de caracteres en imágenes, complementando el trabajo del modelo entrenado y reduciendo tiempos de desarrollo. La Figura 4 muestra algunos de los datases empleados para el entrenamiento del modelo [37]

FIGURA 4: OCR.SPACE [37]

9.9 Dependencias del Backend

El desarrollo del backend en el sistema de automatización y control, requiere de una arquitectura basada en servicios eficientes, seguros y escalables. Para ello, se hace uso de una serie de dependencias que proporcionan las herramientas necesarias para la comunicación, el procesamiento de datos, la autenticación de usuarios y la integración con servicios en la nube.

FastAPI se ha consolidado como uno de los frameworks más modernos y eficientes para la creación de interfaces de programación de aplicaciones (API). Su diseño asíncrono y su compatibilidad con Python permiten desarrollar servicios ligeros y de alta velocidad, lo que resulta ideal en sistemas donde se manejan datos en tiempo real [26]. A su vez, la ejecución de estas aplicaciones se apoya en **Uvicorn**, Se utilizó un servidor basado en ASGI (Asynchronous

Server Gateway Interfaz), el cual permite gestionar múltiples solicitudes de forma simultánea, garantizando una comunicación estable entre el hardware y la nube [27]. Asimismo, la validación y estructuración de los datos se realizó mediante **Pydantic**, herramienta que facilita la definición de modelos con tipado estricto y reduce errores en la comunicación entre los distintos módulos del sistema [38].

Por otra parte, la seguridad del sistema constituye un pilar fundamental dentro del diseño del backend. Dependencias como **bcrypt** y **passlib** proporcionan mecanismos de cifrado robustos mediante funciones hash, mientras que **Python-JOSE** y **PyJWT** se encargan de la generación y validación de tokens de autenticación (JWT), los cuales garantizan la identidad y los permisos de cada usuario o dispositivo conectado [39]. Este enfoque es coherente con los principios de seguridad de la información, donde la confidencialidad y la autenticación resultan esenciales en la transmisión de datos sensibles. Asimismo, la librería **Cryptography** ofrece soporte para algoritmos de cifrado simétrico y asimétrico, permitiendo asegurar la integridad y la privacidad de la información transmitida entre los diferentes componentes del sistema [40]. Su implementación garantiza que la información generada por los sensores o cámaras sea protegida durante la comunicación con los servidores remotos, aspecto crítico en sistemas IoT.

El almacenamiento y gestión de datos se fundamentan en servicios como **Firebase Admin SDK** y **Google Cloud Firestore**, los cuales ofrecen una infraestructura en la nube confiable y escalable, capaz de registrar eventos, imágenes o mediciones en tiempo real. Estas herramientas permiten mantener sincronización constante entre los dispositivos físicos y la plataforma web, característica clave en proyectos donde la información debe actualizarse de manera inmediata [41]. Complementariamente, **SQLAlchemy** aporta una capa de abstracción sobre las bases de datos relacionales, permitiendo manipular los registros mediante objetos y clases sin necesidad de sentencias SQL directas. Esto facilita la organización estructurada de la información, optimizando la interacción con los distintos componentes del sistema [42].

En cuanto a la comunicación con servicios externos, librerías como **Requests** y **HTTPX** posibilitan el intercambio de información con otras plataformas, mediante solicitudes **HTTP** seguras y confiables. Estas dependencias resultan esenciales cuando el sistema requiere obtener información adicional o enviar datos procesados hacia otros servidores [43].

Dentro del ámbito del procesamiento visual y la inteligencia artificial, se destacan **PyTorch, Torchvision, Ultralytics y OpenCV,** que conforman un conjunto de herramientas potentes para la detección, reconocimiento y análisis de imágenes. En el contexto de la ingeniería electrónica, estas librerías permiten aplicar algoritmos de visión por computador y aprendizaje profundo para la identificación automática de objetos o patrones, aportando un enfoque moderno y eficiente en sistemas de automatización [44]. Para el manejo matemático, la simulación de datos y la representación gráfica de resultados se utilizan **NumPy, SciPy y Matplotlib**, que permiten realizar operaciones numéricas avanzadas y graficar comportamientos experimentales de las variables físicas medidas en el sistema [45].

La estabilidad y el monitoreo del entorno de ejecución se fortalecen mediante **Sentry SDK**, herramienta que facilita la detección y registro de errores en tiempo real, asegurando la confiabilidad operativa del backend [46]

Todas estas dependencias conforman la base teórica y tecnológica que sustenta el backend del sistema. Su integración responde a la necesidad de combinar eficiencia, seguridad y escalabilidad, aspectos esenciales en el desarrollo de soluciones electrónicas basadas en la comunicación entre hardware inteligente, servicios en la nube y aplicaciones de análisis de datos.

9.10 Métricas de Evaluación para Modelos de Detección de Objetos

Las métricas de evaluación constituyen herramientas fundamentales para medir el desempeño de los modelos de visión por computador y aprendizaje profundo. Estas permiten cuantificar la capacidad del sistema para detectar correctamente los objetos de interés, evaluar la precisión de las predicciones realizadas y determinar la confiabilidad de los resultados obtenidos durante el entrenamiento y la validación del modelo. En sistemas de detección de matrículas vehiculares, métricas como IoU (Intersection over Union), Precisión (Precisión), Sensibilidad (Recall), F1-Score, AP (Average Precision) y mAP (Mean Average Precision) son ampliamente utilizadas para analizar la calidad de las detecciones y comparar el rendimiento de diferentes modelos.

- **Intersection over Union (IoU)**

  IoU (Intersection over Union) es una métrica utilizada para medir el grado de coincidencia entre la región detectada por el modelo y la ubicación real del objeto. Se calcula como la relación entre el área de intersección y el área de unión de ambas regiones. Cuanto más cercano sea el valor a 1, mayor será la precisión de la detección.

- **Precisión (Precision)**

  La precisión mide la proporción de detecciones correctas realizadas por el modelo respecto al total de detecciones efectuadas. Esta métrica permite determinar qué tan confiables son las predicciones generadas por el sistema.

- **Sensibilidad (Recall)**

  La sensibilidad o recall indica la capacidad del modelo para identificar correctamente los objetos presentes en una imagen. Un valor alto de recall significa que el sistema detecta la mayoría de los objetos reales existentes en el conjunto de datos.

- **F1-Score**

  El F1-Score es una métrica que combina la precisión y la sensibilidad en un único valor, proporcionando una medida equilibrada del rendimiento del modelo. Es especialmente útil cuando se busca mantener un balance entre la cantidad de detecciones correctas y la cobertura de los objetos presentes.

- **Average Precision (AP)**

  El Average Precision (AP) representa el área bajo la curva Precisión-Recall para una clase específica. Esta métrica resume el comportamiento del modelo considerando diferentes umbrales de confianza y permite evaluar su capacidad de detección de manera integral.

- **Mean Average Precision (mAP)**

  El Mean Average Precision (mAP) corresponde al promedio de los valores de AP obtenidos para todas las clases evaluadas. Es una de las métricas más utilizadas en sistemas de detección de objetos, ya que proporciona una visión general del desempeño global del modelo y facilita la comparación entre diferentes arquitecturas de inteligencia artificial.

10. METODOLOGÍA

El proceso de entrenamiento constituyó una de las etapas más determinantes en el desarrollo del sistema de reconocimiento y control automatizado de acceso vehicular. En esta fase se diseñó, entrenó y ajustó un modelo basado en la arquitectura YOLOv8, orientado a la detección precisa de matrículas en condiciones reales de operación, tales como presencia de barreras, reflejos en carrocerías, ángulos oblicuos y variaciones de iluminación nocturna.

El entrenamiento se llevó a cabo utilizando un conjunto de datos heterogéneo, compuesto por imágenes capturadas en diferentes condiciones ambientales, ángulos de visión y tipos de vehículos, con el fin de representar de forma realista las situaciones que se presentan en un punto de acceso vehicular. Las anotaciones se realizaron mediante cajas delimitadoras (bounding boxes) que abarcan no solo las placas y frentes de los vehículos, sino también otros elementos presentes en la escena que aportan contexto al proceso de detección. Este enfoque permitió que el modelo aprendiera a distinguir de manera precisa las áreas de interés como la matrícula del resto de la imagen, desarrollando una comprensión más completa de la composición visual y reduciendo así los errores por confusión con el entorno.

A fin de robustecer la generalización del modelo, se aplicaron técnicas de data augmentation que incluyeron rotaciones, traslaciones, escalados y variaciones fotométricas en brillo y contraste. Estas transformaciones simularon distintos escenarios de captura y ayudaron a mantener la precisión del modelo frente a las variaciones de distancia y perspectiva típicas de un punto de control vehicular. El proceso se complementó con mecanismos de supresión no máxima (NMS) basados en la métrica IoU (Intersection over Union), garantizando que solo las detecciones más confiables fueran conservadas durante la inferencia.

Paralelamente, se desarrolló y entrenó un modelo auxiliar de corrección angular, cuya función fue ajustar la orientación de las placas detectadas antes del reconocimiento de caracteres. Este modelo permitió recortar y enderezar las placas que aparecían en ángulos oblicuos, compensando inclinaciones o distorsiones de perspectiva y mejorando significativamente la legibilidad en la etapa de reconocimiento óptico de caracteres (OCR). Dicho entrenamiento se apoyó en un conjunto adicional de imágenes de placas rotadas y normalizadas, generando un submódulo especializado en el alineamiento automático del recorte.

El resultado global de esta etapa fue un sistema de detección robusto y preciso, capaz de identificar y recortar las matrículas en tiempo real con altos niveles de confianza. Los modelos entrenados, entre ellos el artefacto optimizado best.pt, se integraron posteriormente con el módulo OCR y la infraestructura IoT del proyecto, conformando el núcleo inteligente del sistema de control vehicular implementado para el conjunto Amparo Country.

10.1. Entrenamiento 1 – Análisis visual del entrenamiento y resultados del modelo

En las siguientes figuras ilustran de forma visual el proceso de validación y los resultados obtenidos durante el entrenamiento del modelo de detección de placas vehiculares. En ellas se evidencian las etiquetas utilizadas, las predicciones generadas por el modelo, así como las curvas y matrices que reflejan su desempeño. Este análisis permite observar cómo el sistema identifica correctamente las matrículas en distintos escenarios, manteniendo altos niveles de precisión y consistencia incluso ante variaciones de iluminación, distancia y ángulo de captura.

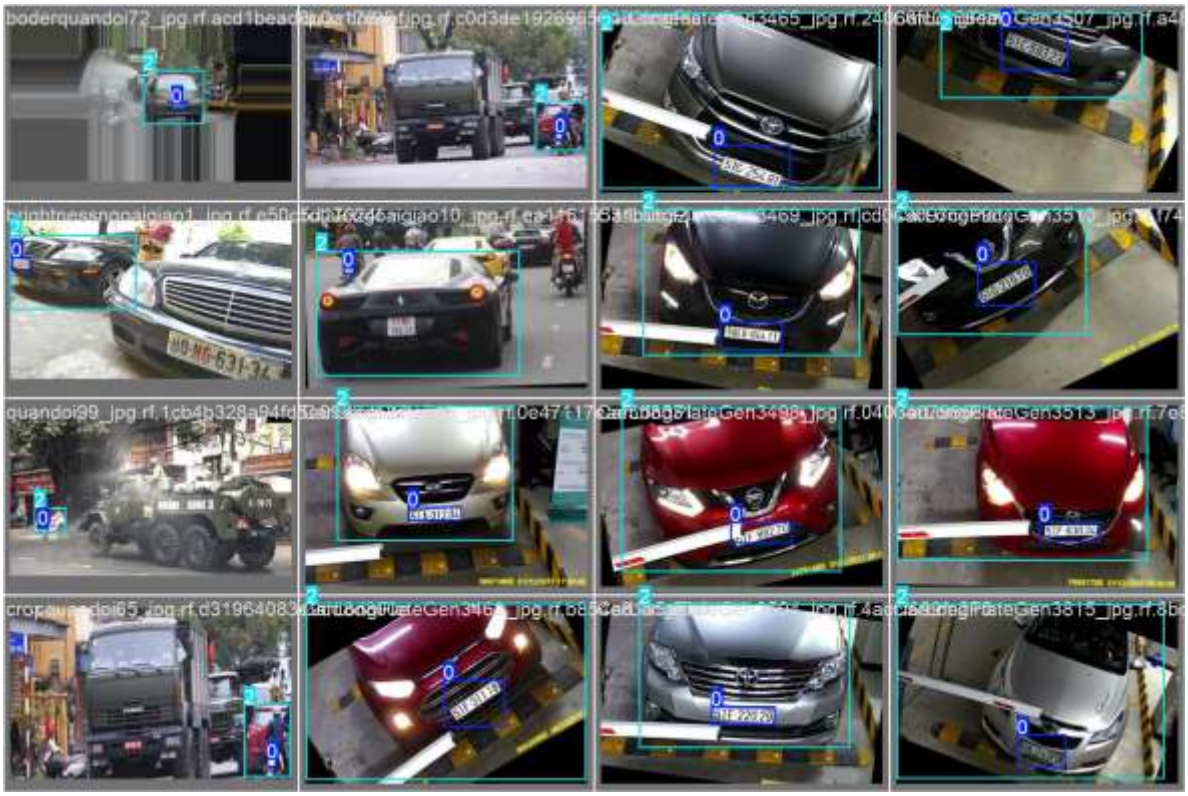


FIGURA 5:ETIQUETAS DE VALIDACIÓN (LOTE 0) [FUENTE PROPIA]

La Figura 5 muestra las etiquetas de validación correspondientes al lote 0, donde se observan las cajas delimitadoras que identifican el vehículo y la matrícula en escenas reales de portería con presencia de barreras, reflejos y diferentes ángulos de captura. Estas anotaciones permiten evaluar la capacidad de generalización del detector frente a condiciones variadas del entorno.

FIGURA 6:PREDICCIONES DEL MODELO (LOTE0) [FUENTE PROPIA]

En la Figura 6 se observan las detecciones generadas por el modelo junto con sus respectivos niveles de confianza. Los resultados demuestran una adecuada identificación de las matrículas incluso en situaciones que incluyen reflejos, faros encendidos y oclusiones parciales, lo que respalda la capacidad de generalización y estabilidad del sistema ante condiciones operativas reales.

FIGURA 7: ETIQUETAS DE VALIDACIÓN (LOTE 1) [FUENTE PROPIA]

Las etiquetas de validación presentadas en la Figura 7 incluyen una amplia variedad de configuraciones espaciales de los vehículos, con diferencias en distancia, orientación y ubicación dentro de la imagen. Esta heterogeneidad resulta fundamental para evaluar la capacidad de generalización del modelo y su desempeño en condiciones operativas diversas.

FIGURA 8: PREDICCIONES DEL MODELO (LOTE 1) [FUENTE PROPIA]

Como se observa en la Figura 8, el modelo mantiene detecciones precisas de las matrículas incluso en presencia de reflejos intensos y ángulos pronunciados. Aunque los niveles de confianza disminuyen ligeramente en los casos de mayor inclinación, los resultados evidencian un desempeño estable en condiciones reales de operación.

FIGURA 9: PREDICCIONES DEL MODELO (LOTE 2) [FUENTE PROPIA]

Como se aprecia en la Figura 9, el modelo mantiene una detección adecuada de las matrículas y vehículos en un conjunto independiente. No obstante, en fondos con bajo contraste pueden presentarse omisiones cuando se emplean umbrales de confianza demasiado altos.

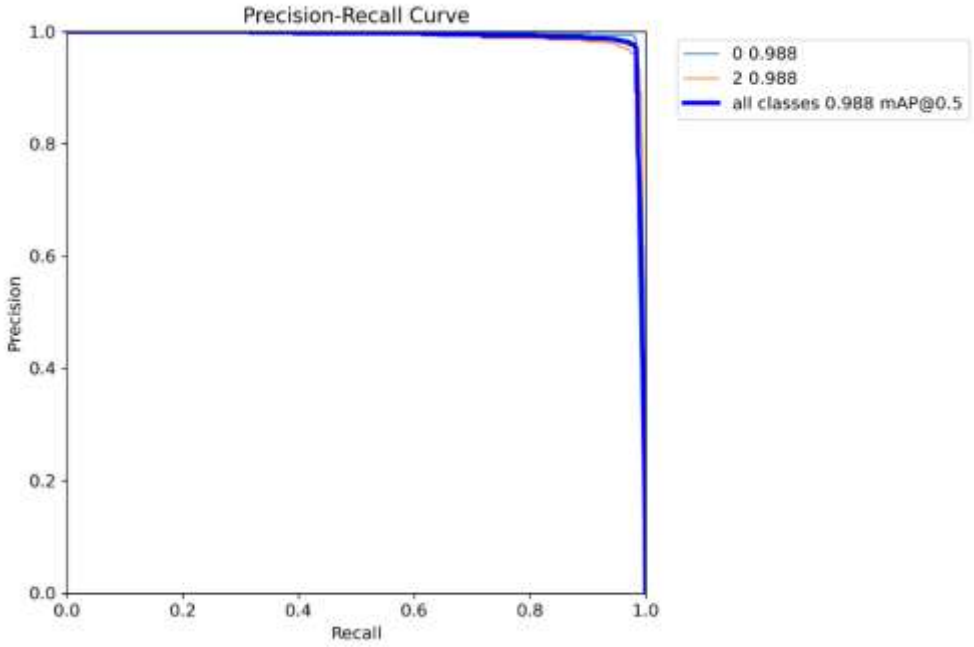


FIGURA 10:CURVA PRECISIÓN–RECALL POR CLASES (MAP@0.5) [FUENTE PROPIA]

Como se aprecia en la Figura 10, la curva Precisión–Recall presenta un comportamiento alto y estable para las clases placa y vehículo, alcanzando una precisión promedio (AP) cercana a 0.988 y un mAP@0.5 global de 0.988. La forma de la curva evidencia que el modelo mantiene una excelente relación entre precisión y exhaustividad a lo largo de los diferentes niveles de confianza, reflejando un desempeño sólido en la detección de ambos objetos.

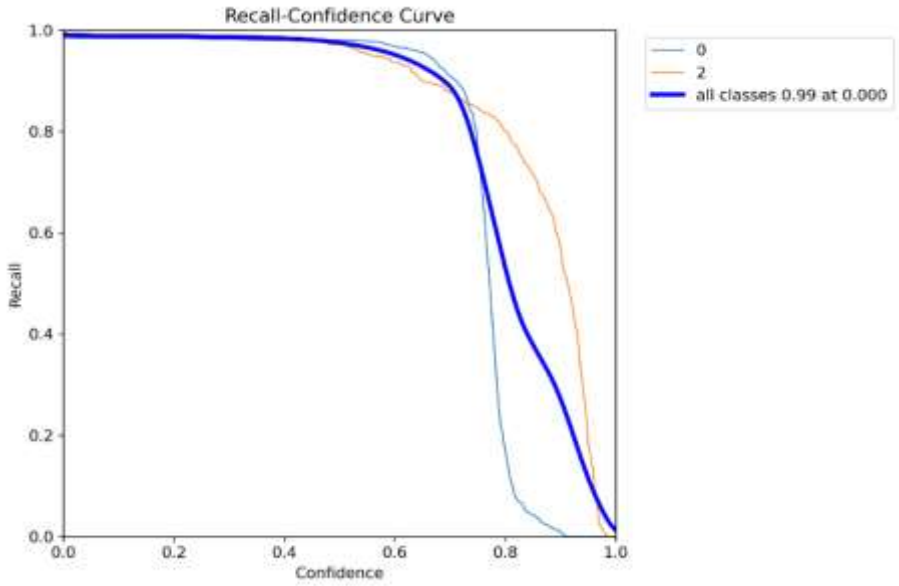


FIGURA 11:CURVA RECALL–CONFIDENCE (TODAS LAS CLASES) [FUENTE PROPIA]

La Figura 11 refleja que el recall permanece elevado hasta niveles de confianza de 0.75–0.80, decreciendo rápidamente a partir de ese punto. Este comportamiento sugiere que dicho rango resulta adecuado para la operación del sistema, al garantizar una detección amplia sin incrementar significativamente los falsos positivos.

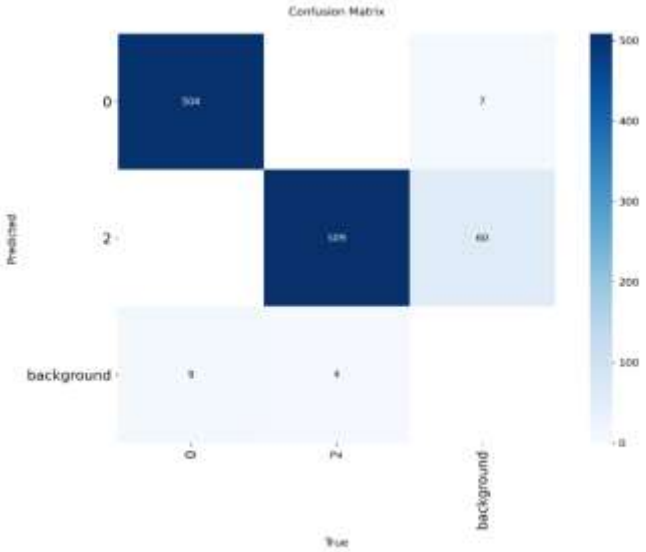


FIGURA 12:MATRIZ DE CONFUSIÓN (VALORES ABSOLUTOS) [FUENTE PROPIA]

Como se aprecia en la Figura 12, la matriz de confusión resume los aciertos y errores obtenidos por el modelo para cada clase evaluada. En la clase placa (0) se registraron 504 verdaderos positivos (TP), 9 falsos negativos (FN) y 7 falsos positivos (FP), mientras que para la clase vehículo (2) se obtuvieron 509 TP, 4 FN y 60 FP. Los errores residuales se concentran principalmente en los falsos positivos asociados a la clase vehículo, atribuibles a estructuras metálicas o superficies brillantes presentes en el entorno.

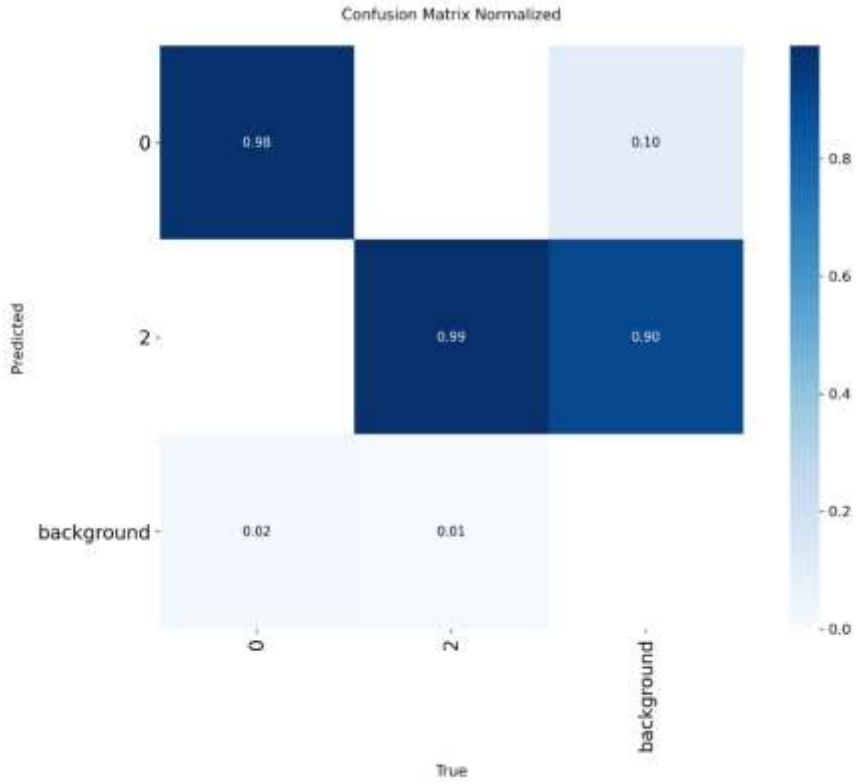


FIGURA 13: MATRIZ DE CONFUSIÓN NORMALIZADA (PROPORCIONES) [FUENTE PROPIA]

Como se observa en la Figura 13, las diagonales dominantes (0.98 para placa y 0.99 para vehículo) confirman la alta capacidad discriminante del modelo. Los pocos errores fuera de la diagonal corresponden a detecciones de vehículos sobre fondo, sin impacto significativo en el rendimiento global.

10.2. Entrenamiento 2 – Reentrenamiento del modelo y análisis de desempeño visual

El segundo entrenamiento correspondió a la etapa de reentrenamiento (retrain), donde se buscó mejorar la capacidad del modelo para reconocer matrículas en condiciones más diversas: distintos países, colores de placa, iluminación nocturna, distancias y ángulos variables. A partir de este proceso se generó el artefacto best.pt, considerado la versión más estable y precisa del detector.

Las siguientes figuras y resultados describen visualmente el desempeño del modelo, evidenciando su comportamiento ante diferentes escenarios de validación y las métricas obtenidas durante el proceso de evaluación.

FIGURA 14: ETIQUETAS DE VALIDACIÓN (LOTE 0) [FUENTE PROPIA]

En la Figura 14 muestra las etiquetas reales del lote 0, con placas de diversos países, colores y formatos. Esta variedad de estilos y tipografías favorece la capacidad del modelo para generalizar ante distintos contextos.

FIGURA 15:PREDICIONES DEL MODELO (LOTE 0) [FUENTE PROPIA]

Como se observa en la Figura 15, las predicciones del modelo sobre el lote 0 muestran una correcta localización de las matrículas, con niveles de confianza entre 0.6 y 0.9, incluso en imágenes con marcos decorativos o capturas realizadas a corta distancia.

FIGURA 16: ETIQUETAS DE VALIDACIÓN (LOTE 1) [FUENTE PROPIA]

La Figura 16 incluye escenas nocturnas y primeros planos, aportando datos valiosos sobre cómo se comporta el detector en condiciones de iluminación compleja o presencia de faros.

FIGURA 17:PREDICCIONES DEL MODELO (LOTE 1) [FUENTE PROPIA]

La Figura 17 presenta las predicciones sobre el lote 1: las detecciones permanecen estables con umbrales de confianza entre 0.7 y 0.8. En escenas de baja luz, algunas placas presentan confianza media (0.5–0.7), lo cual puede ajustarse según la necesidad operativa.

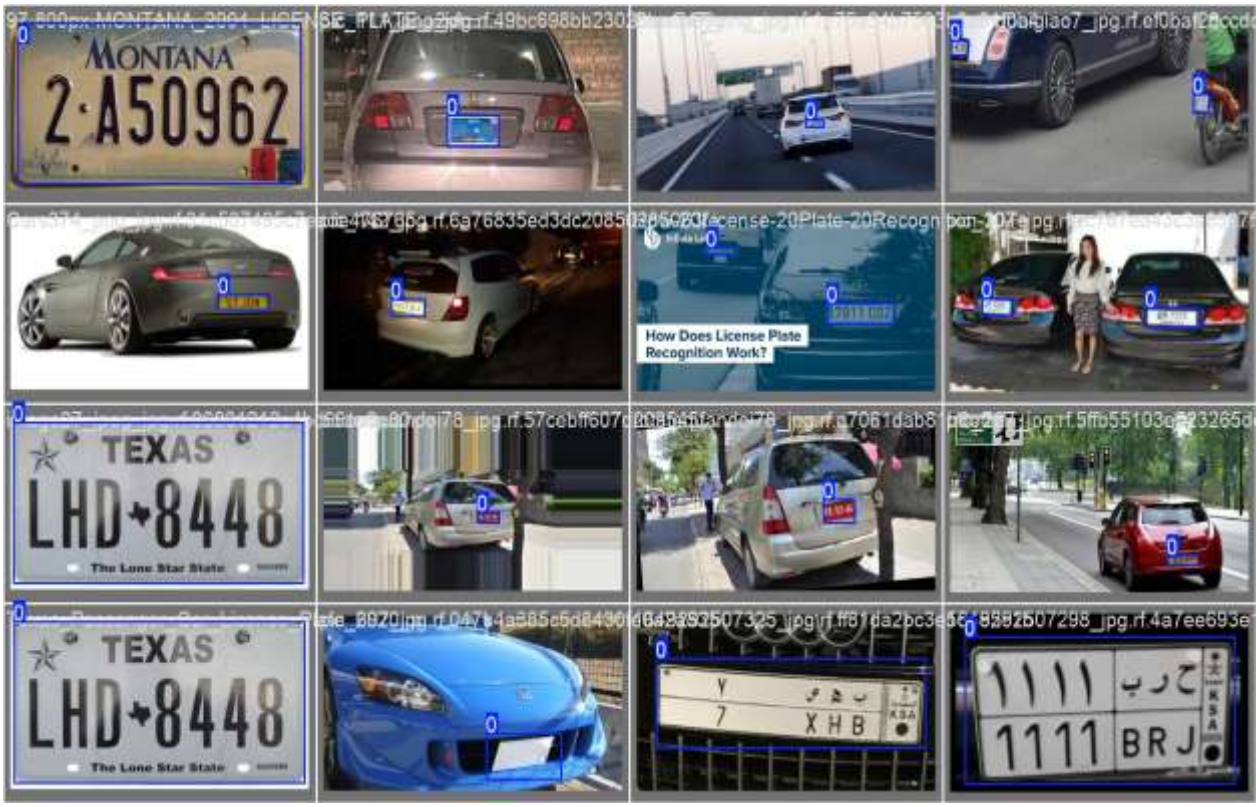


FIGURA 18: ETIQUETAS DE VALIDACIÓN (LOTE 2) [FUENTE PROPIA]

La Figura 18 muestra Lote con placas antiguas, multicolor y de distintos alfabetos, lo que amplía la cobertura del modelo y su aplicabilidad en escenarios internacionales.

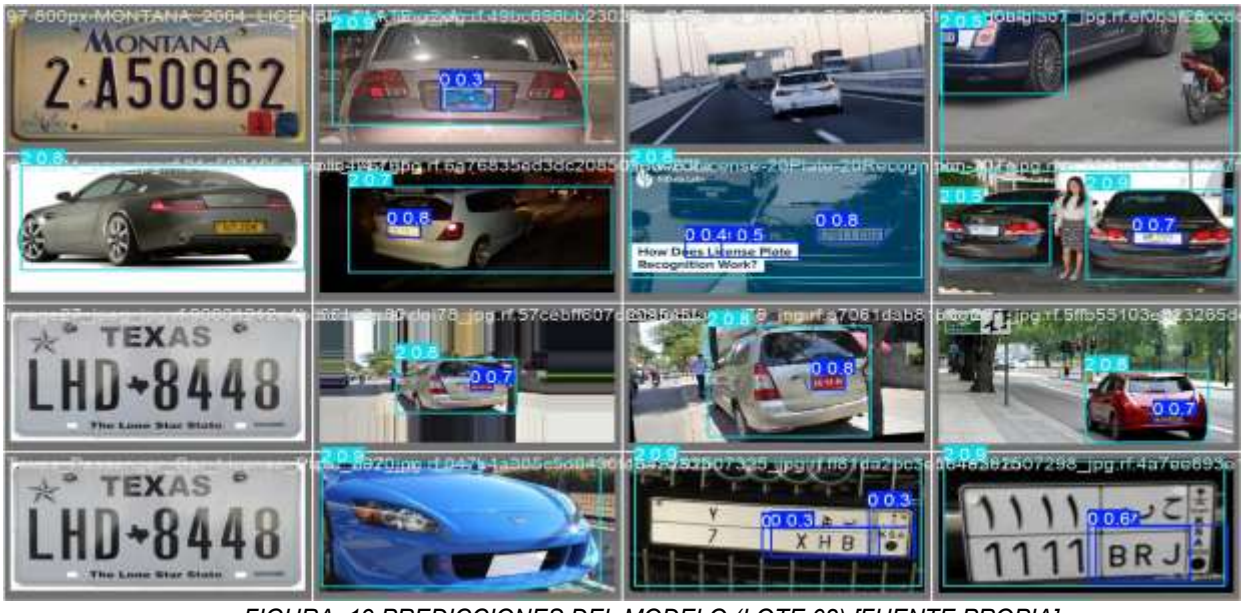


FIGURA 19:PREDICCIONES DEL MODELO (LOTE 02) [FUENTE PROPIA]

La Figura 19 muestra el modelo demuestra buena generalización incluso con placas desgastadas o en movimiento. Algunas confusiones vehículo placa concuerdan con los valores observados en la matriz de confusión.

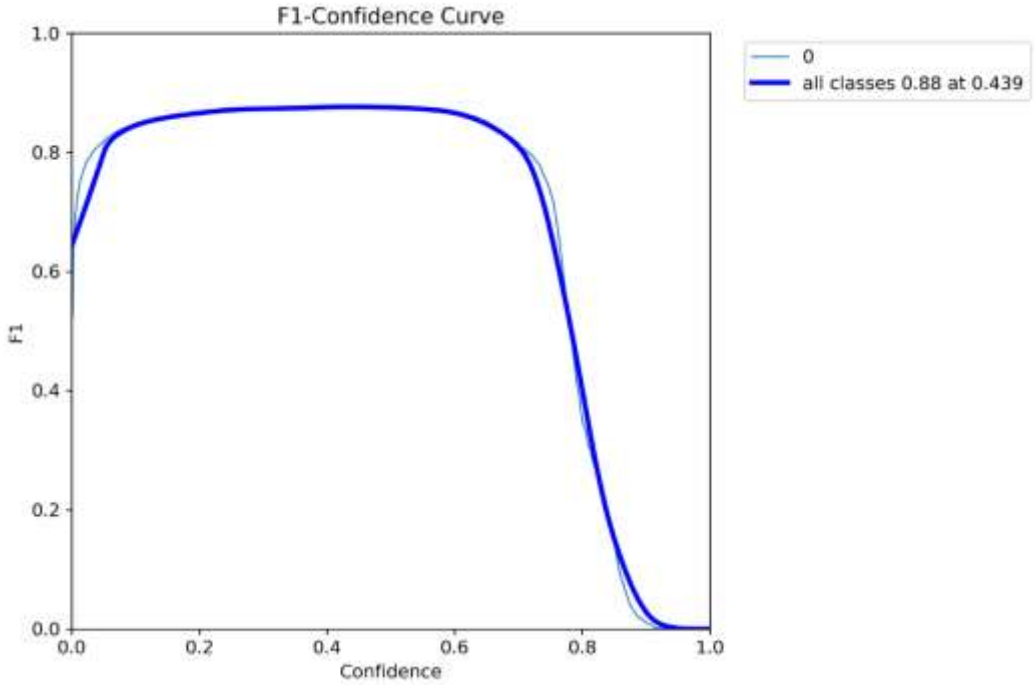


FIGURA 20:CURVA F1–CONFIDENCE [FUENTE PROPIA]

La Figura 20 muestra el punto de máximo equilibrio entre precisión y recall (F1 ≈ 0.88 a confianza ≈ 0.44). Útil para definir el umbral óptimo de operación.

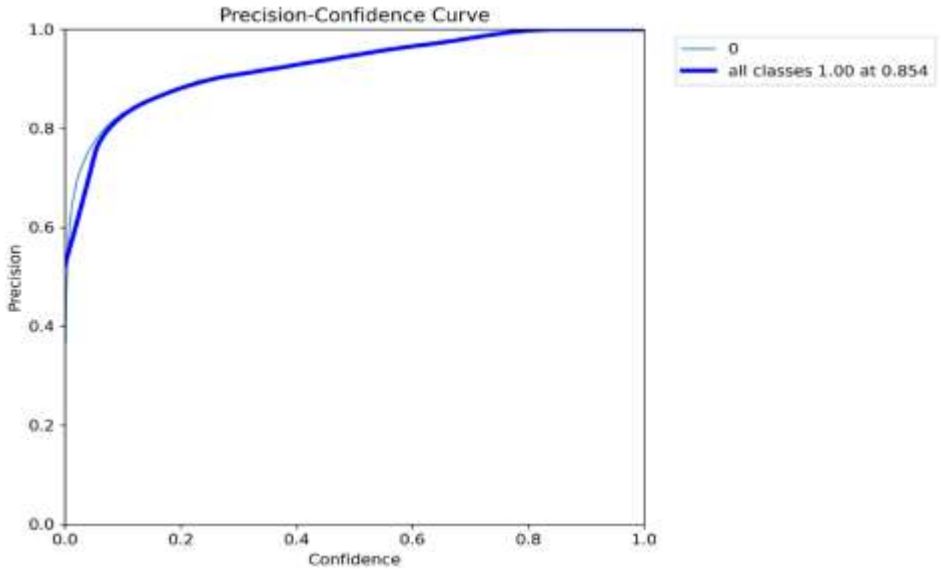


FIGURA 21:CURVA PRECISION–CONFIDENCE [FUENTE PROPIA]

La Figura 21 muestra la evidencia un crecimiento progresivo de la precisión hasta alcanzar ≈ 100 % a confianza ≈ 0.85. Recomendado para minimizar falsas aperturas en operación

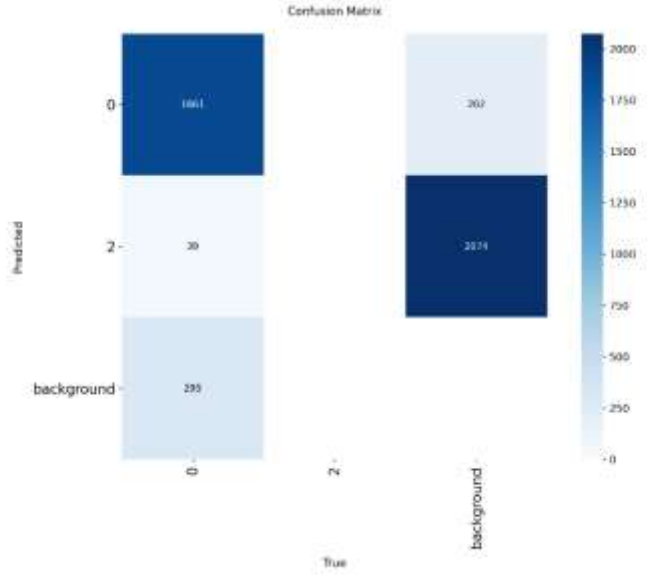


FIGURA 22:MATRIZ DE CONFUSIÓN (VALORES ABSOLUTOS) [FUENTE PROPIA]

La Figura 22 muestra la matriz de confusión en valores absolutos muestra un rendimiento sólido del modelo en las dos clases principales: **placa (0)** y **vehículo (2)**. Se registraron **1861 aciertos** en la clase 0 y **2074 aciertos** en la clase 2. Los errores más comunes fueron 202 casos donde un vehículo fue confundido con una placa, 39 casos inversos y 295 omisiones de placas detectadas como fondo. En general, el modelo mantiene una alta consistencia, aunque aún puede confundir ciertos reflejos o bordes brillantes con placas reales.

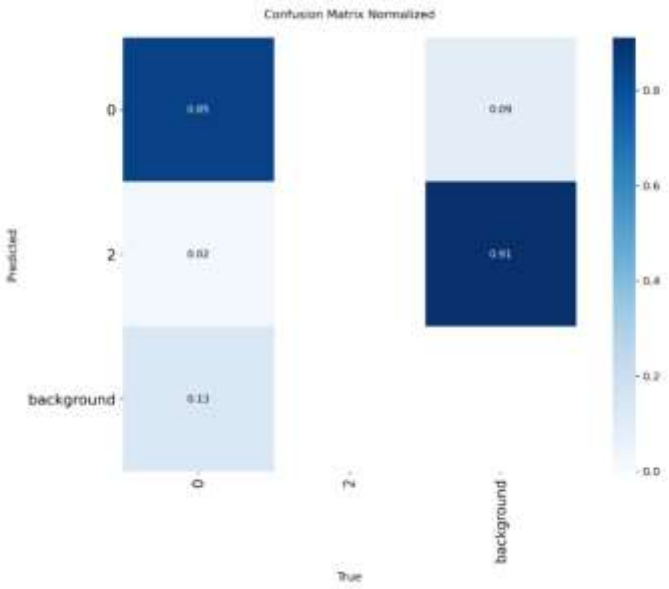


FIGURA 23:MATRIZ DE CONFUSIÓN NORMALIZADA (PROPORCIONES) [FUENTE PROPIA]

La Figura 23 Muestra que la matriz normalizada refleja un **85 % de acierto** en la clase placa y **91 %** en vehículo. Las pérdidas principales (≈13 %) corresponden a omisiones de placas por condiciones de brillo o suciedad. Este comportamiento demuestra que el detector distingue correctamente las matrículas en la mayoría de las escenas, con un rendimiento ligeramente superior en la detección del vehículo completo.

10.2.1. Resumen técnico y recomendaciones operativas

El segundo entrenamiento demostró un rendimiento estable del modelo, con una tasa de acierto del 85 % en placas (clase 0) y 91 % en vehículos (clase 2). Los errores se concentraron principalmente en confusiones entre ambas clases y en omisiones de placas sobre fondos brillantes o poco contrastados.

El análisis de las curvas de desempeño permitió establecer tres rangos operativos:

- 0.43–0.45: punto de equilibrio entre precisión y recall (máximo F1).
- ≥ 0.85: prioriza la precisión absoluta (≈ 100 %), ideal para minimizar falsas aperturas.
- 0.70–0.80: rango recomendado para operación en portería, que conserva alta sensibilidad sin incrementar los falsos positivos.

Como mejora, se recomienda incorporar ejemplos negativos adicionales (marcos, logotipos y superficies reflectantes) y reforzar el dataset con imágenes nocturnas y placas inclinadas, de modo que el sistema mantenga un desempeño consistente ante las variaciones reales del entorno. La Tabla 1 presenta el resumen de métricas del modelo, incluyendo los valores de TP, FP, FN, precisión, recall y F1 para las clases placa y vehículo, así como el promedio global obtenido durante la evaluación.

Clase	TP	FP	FN	Precisión (%)	Recall (%)	F1 (%)
Placa (0)	**1861**	**202**	**334**	**90.2**	**84.9**	**87.5**
Vehículo (2)	**2074**	**39**	**202**	**98.1**	**91.0**	**94.4**
Promedio global (mAP@0.5)	**—**	**—**	**—**	**—**	**—**	**≈ 89.6**

Tabla 1 resumen de métrica del modelo

10.3. Entrenamiento 3 – Diversificación de datos y generalización del modelo

En este tercer ciclo se afinó el detector de matrículas ampliando la diversidad del conjunto de entrenamiento: escenas nocturnas y diurnas, placas con distintos formatos/países, variaciones de distancia, inclinación y oclusiones parciales. Se trabajó con anotación por cajas y *data augmentation* (rotación leve, cambios fotométricos y de escala) para fortalecer la robustez ante perspectiva e iluminación. Los mosaicos de entrenamiento muestran que el modelo ve ejemplos realistas de portería (barrera, faros, reflejos y marcos ornamentales), mientras que la gráfica de etiquetas confirma balance espacial: la mayoría de las placas aparece cerca de la zona central–inferior de la imagen y con anchos/altos pequeños respecto al fotograma, lo que justifica mantener resoluciones adecuadas y anclas sensibles a objetos de tamaño reducido. En conjunto, este entrenamiento consolida la capacidad de generalización del detector antes de su uso operativo en el flujo IoT + OCR.

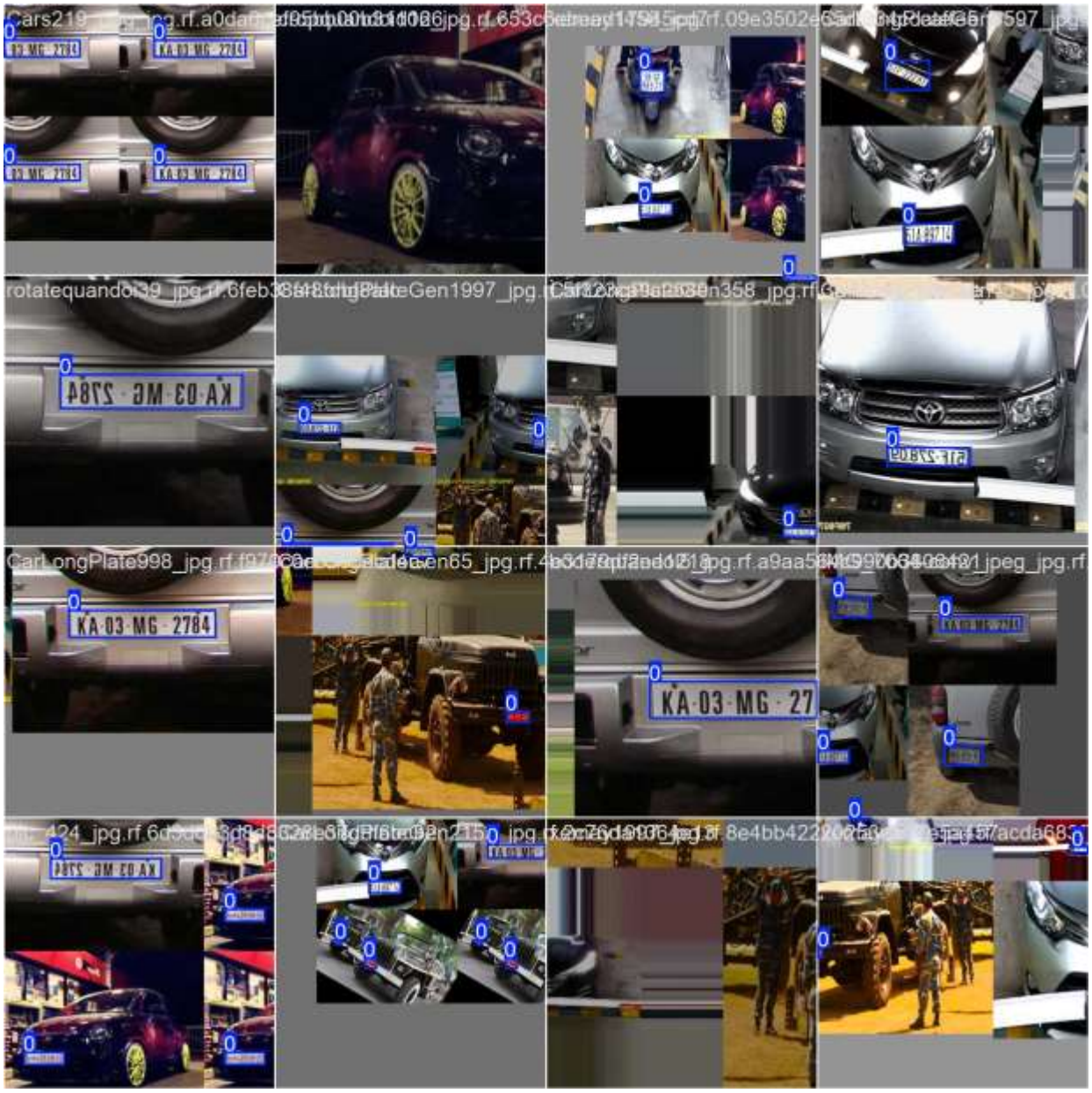


FIGURA 24:MOSAICO DE ENTRENAMIENTO (LOTE 0) [FUENTE PROPIA]

La Figura 24 se logra apreciar que en este lote reúne escenas de portería con primeros planos del frontal, presencia de barrera y faros encendidos, además de ángulos oblicuos que generan reflejos y sombras. La mezcla de distancias y perspectivas obliga al modelo a aprender patrones robustos de localización de la placa aun cuando el encuadre es cerrado o el brillo de los faros compite con el contraste del texto. Este tipo de variabilidad inicial es clave para evitar sobreajuste a un único punto de vista.

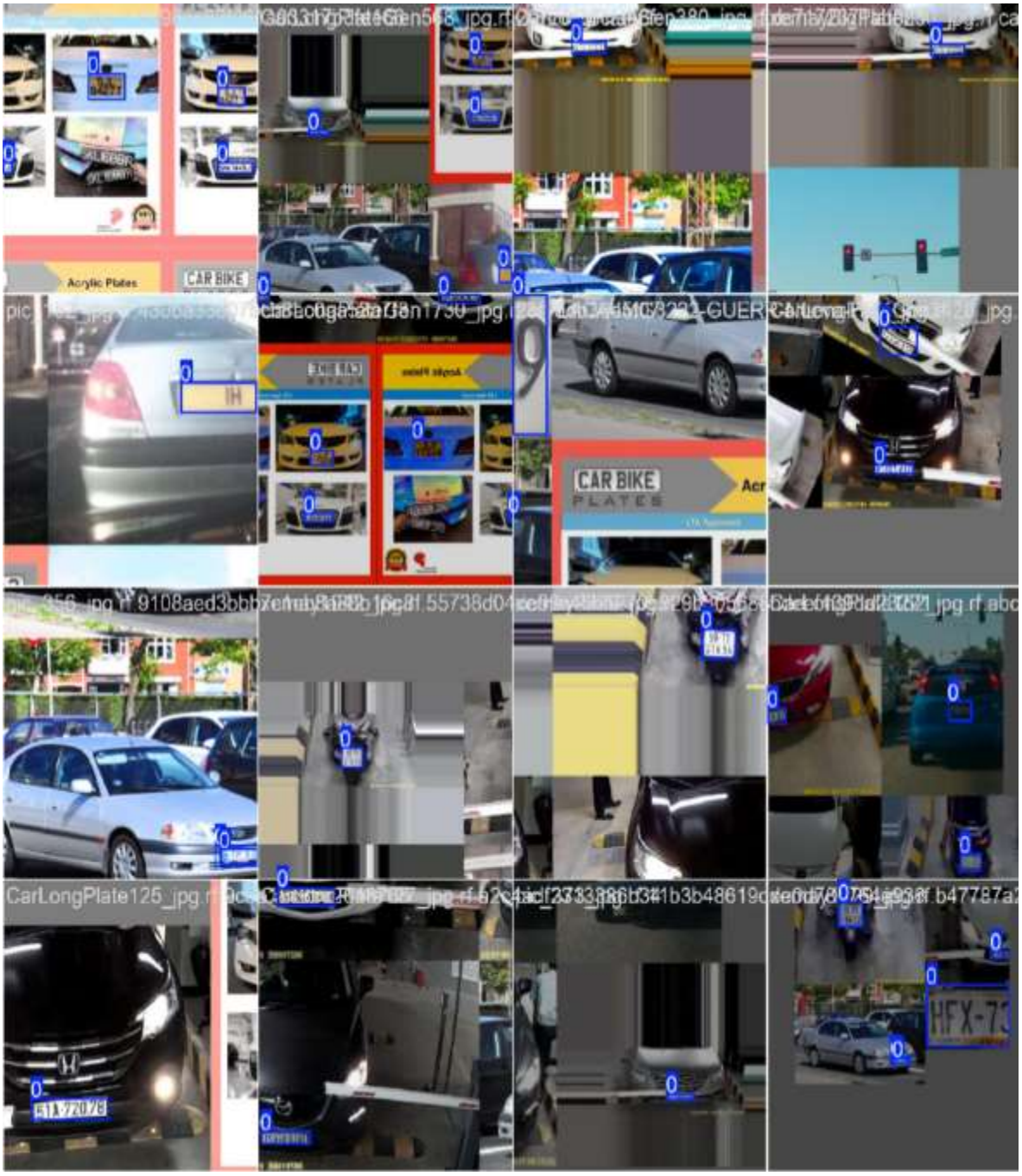


FIGURA 25:MOSAICO DE ENTRENAMIENTO (LOTE 1) [FUENTE PROPIA]

Incluye **placas de distintos países**, con **colores, marcos y tipografías** heterogéneas, además de **oclusiones parciales** y **fondos complejos** (publicidad, transeúntes). Esta curaduría aumenta la riqueza visual y ayuda a que el detector distinga la **morfología real de la matrícula** frente a rectángulos o bordes similares presentes en el entorno, reduciendo así falsos positivos como se puede aprecia en la figura 25.

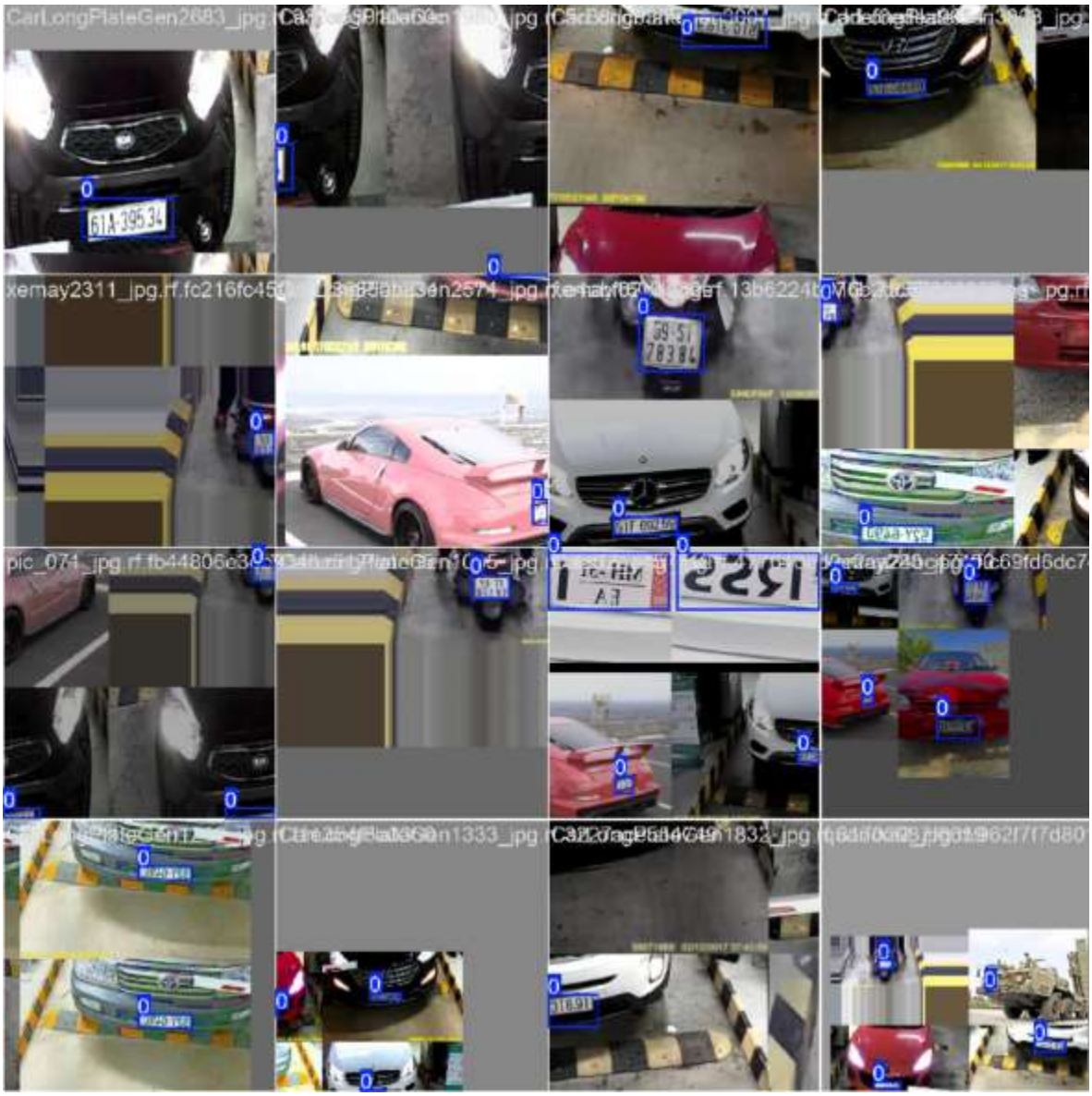


FIGURA 26:MOSAICO DE ENTRENAMIENTO (LOTE 2) [FUENTE PROPIA]

La Figura 26 Presenta **baja iluminación**, vehículos **en movimiento** y **placas inclinadas** o parcialmente cortadas. Con ello se entrena la tolerancia a **cambios de perspectiva** y **rangos dinámicos** exigentes (ruido, desenfoque). Este lote entrena al modelo para responder de forma estable cuando las condiciones de captura se alejan del ideal, mejorando su **resiliencia en operación nocturna** o con cámaras no perfectamente alineadas.

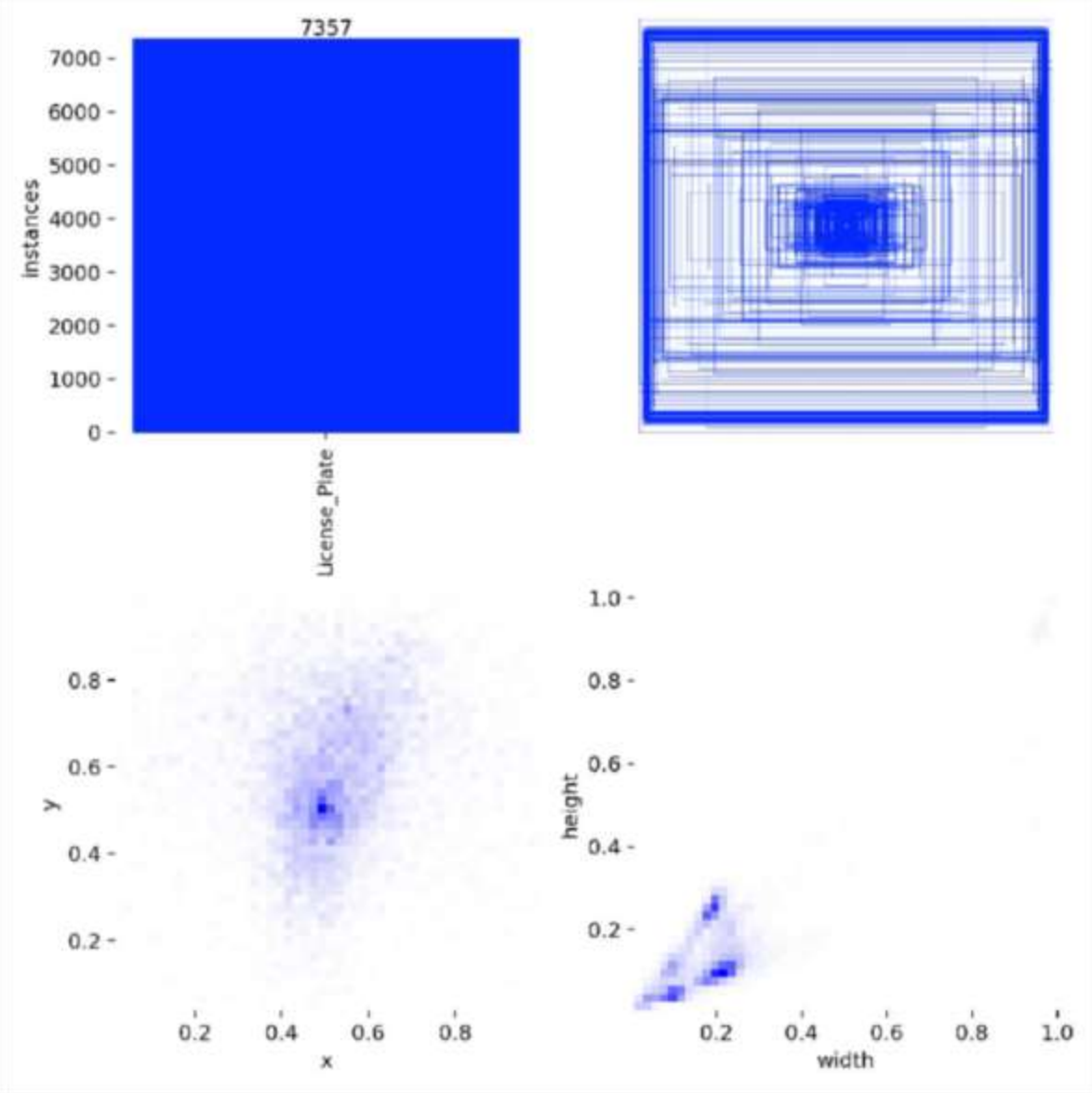


FIGURA 27:DISTRIBUCIÓN DE ETIQUETAS Y TAMAÑOS [FUENTE PROPIA]

La figura 27 muestra el resumen estadístico de **7357** anotaciones muestra una **concentración espacial** de la placa en la franja **central–inferior** y un **predominio de cajas pequeñas** (width/height reducidos). Esta lectura guía decisiones de entrenamiento: **ajustar *anchors*** hacia objetos pequeños, **mantener una resolución de entrada** que preserve detalle de caracteres y definir *augmentations* (rotación leve, recortes y ajustes fotométricos) coherentes con la distribución real de las capturas.

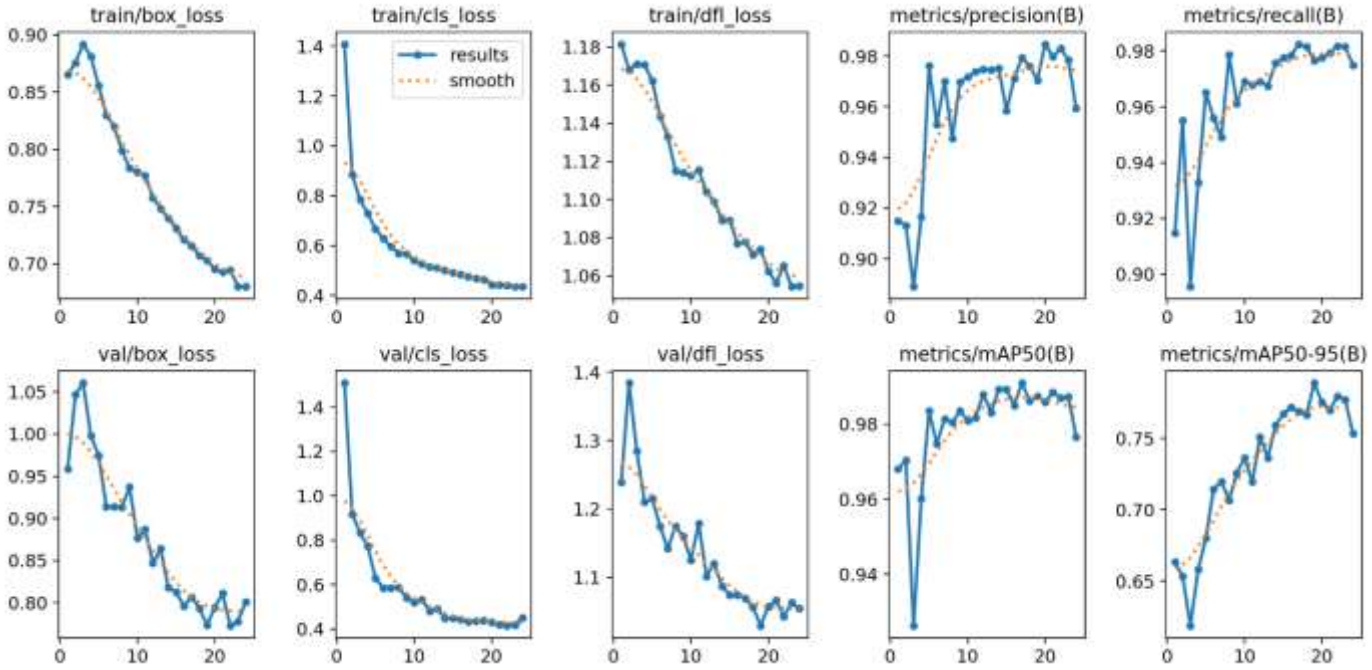


FIGURA 28:CURVAS DE ENTRENAMIENTO Y VALIDACIÓN [FUENTE PROPIA]

- Pérdidas – Entrenamiento
  - train/box_loss: −26 % aprox. (0.90 → 0.67).
  - train/cls_loss: −61 % aprox. (1.40 → 0.55).
  - train/dfl_loss: −11 % aprox. (1.18 → 1.05).

- Pérdidas – Validación
  - val/box_loss: −26 % aprox. (1.05 → 0.78).
  - val/cls_loss: −62 % aprox. (1.45 → 0.55).
  - val/dfl_loss: −24 % aprox. (1.38 → 1.05).

- Métricas
  - metrics/precision(B): se estabiliza cerca de 0.98–0.99.
  - metrics/recall(B): asciende hasta 0.97–0.99.
  - metrics/mAP50(B): converge alto, ≈ 0.98–0.99.
  - metrics/mAP50–95(B): mejora sostenida hasta ≈ 0.78–0.80 (mejor encuadre a IoU estrictos).

La Figura 28 muestra las curvas de entrenamiento y validación del modelo. La lectura global indica que las curvas presentan una convergencia estable, sin sobreajuste evidente y un modelo que alcanza precisión y recall ~98–99 %, con mAP@0.5 cercano a 1.0 y mAP@0.5–0.95 ~0.79, listos para fijar el umbral operativo (0.70–0.80) según los requisitos.

10.3.1. Evaluación del desempeño del modelo

Para cuantificar la calidad del detector se usaron métricas estándar de visión por computador, calculadas comparando las cajas predichas con las cajas reales (anotadas manualmente):

- **IoU (Intersection over Union)** entre caja predicha B_{pred}y real B_{gt}:

$$\text{IoU} = \frac{|\, B_{\text{pred}} \cap B_{\text{gt}} \,|}{|\, B_{\text{pred}} \cup B_{\text{gt}} \,|}$$

Se usa como criterio de acierto (p.ej., IoU ≥ 0.5) y en la **NMS** para descartar solapes redundantes.

- **Precisión y Recall (sensibilidad)**:

$$\text{Precisión} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}$$

donde TPson verdaderos positivos, FPfalsos positivos y FNfalsos negativos.

- **F1** (equilibrio entre precisión y recall):

$$F1 = 2 \cdot \frac{\text{Precisión} \cdot \text{Recall}}{\text{Precisión} + \text{Recall}}$$

- **AP y mAP**: el **AP** es el área bajo la curva Precisión–Recall por clase; el **mAP@0.5** promedia el AP con umbral **IoU = 0.5** (criterio PASCAL). El **mAP@0.5–0.95** promedia en diez umbrales de IoU (0.50…0.95), exigiendo mayor ajuste geométrico.

11. ARQUITECTURA

11.1. PROPÓSITO Y ALCANCE DEL SISTEMA

ParkSecureIoT ofrece una interfaz sencilla para que administradores, vigilantes y residentes gestionen el acceso vehicular. Esta muestra la información proveniente de los módulos de detección y monitoreo el cual se comunica con el backend mediante una **API REST** (Representational State Transfer), un estilo de arquitectura que facilita el intercambio de datos entre sistemas a través del protocolo HTTP.
Incluye: monitoreo de plazas, visualización de accesos, gestión de residentes y permisos, y control de autenticación.

Arquitectura
La aplicación es desarrollada con **React.js**, consume datos del backend a través de endpoints REST en JSON y utiliza **Firebase Firestore** para la persistencia. Su diseño es responsivo y orientado a la usabilidad. La Tabla 2 presenta los principales módulos funcionales del sistema, describiendo el objetivo de cada uno y las tecnologías utilizadas para su implementación. Asimismo, evidencia la integración de herramientas de monitoreo, control de acceso, administración y autenticación que soportan el funcionamiento de la plataforma.

Módulo visual	**Objetivo**	**Tecnologías empleadas**
Panel de monitoreo de estacionamiento	Mostrar la ocupación de plazas en tiempo real.	WebSocket + React Hooks + YOLO v8.
Panel de control de acceso	Gestionar el registro de entradas/salidas y matrículas.	React + Fetch API + OCR.
Panel administrativo	Gestionar residentes, permisos y pagos.	React + API REST + Firebase Auth.
Autenticación de usuarios	Controlar el inicio de sesión y permisos.	Firebase Authentication + Context API.

Tabla 2 capacidades del subsistema

11.2. Descripción General Arquitectónica
El backend de ParkSecureIoT está organizado bajo la carpeta aplicación/backend/ y se basa en una arquitectura modular. La aplicación utiliza FastAPI, con un archivo principal (main.py) que crea la instancia raíz de la API y registra los diferentes enrutadores.

Cada enrutador gestiona un conjunto específico de funcionalidades y delega la lógica a una capa de servicios, la cual se encarga del procesamiento de datos y de las interacciones con Firebase.

11.2.1. Arquitectura del Subsistema de Doble Propósito

El sistema emplea una infraestructura de cámaras compartida para dos flujos principales:

- Monitoreo Continuo

- Procesamiento constante de video (~2 FPS) desde cámaras 402 y 202. Uso de YOLOv8 + OpenCV para detección vehicular y análisis de ocupación. Transmisión de resultados en tiempo real vía WebSocket y registro en Firestore.Control de Acceso por Evento
- Activado mediante solicitudes HTTP POST desde cámaras 1502 y 1602. Usa OCR.space API para lectura de matrículas y valida permisos mediante Firestore, permitiendo también registrar resultados (permitido/denegado) en la colección de accesos vehiculares..

La tabla 3 presenta una comparación entre los dos flujos principales del sistema: el monitoreo continuo y el control de acceso vehicular. Además, describe los componentes involucrados, el tipo de procesamiento realizado y los servicios encargados de ejecutar cada función dentro de la arquitectura desarrollada.

Aspecto	**Monitoreo**	**Control de Acceso**
Desencadenador	Proceso continuo	Solicitud HTTP
Cámaras	402, 202	1502, 1602
Tratamiento	YOLO + análisis espacial	YOLO + OCR
Producción	Métricas WebSocket	Decisión de acceso
Archivo	parking_monitor_service.py	plate_detection_service.py, access_service.py

Tabla 3 arquitectura

11.3. PILA TECNOLÓGICA CENTRAL

ParkSecureIoT integra una pila moderna basada en FastAPI, Firebase y YOLOv8. FastAPI gestiona la API con alto rendimiento y validación automática; Firebase ofrece almacenamiento y sincronización en tiempo real; y YOLOv8, junto con OpenCV, permite la detección precisa de vehículos. La seguridad se implementa con JWT y Bcrypt, y se complementa con servicios externos como OCR.space y Sentry.

Ciclo de Vida de la Aplicación

El sistema utiliza el patrón lifespan de FastAPI:

- Inicio: activa tareas de fondo y cámaras RTSP.
- Ejecución: procesa solicitudes y detecciones en tiempo real.
- Cierre: libera recursos y detiene los hilos de monitoreo

11.4. ORGANIZACIÓN DE LA API

El backend expone seis routers principales:

La Tabla 4 muestra los enrutadores principales de la API, sus prefijos y las funciones que desempeñan dentro de la arquitectura del sistema, permitiendo una organización modular y eficiente de los servicios implementados.

Router	Prefijo	Objetivo
auth_routes	/api/v1/auth	Autenticación JWT.
resident_routes	/api/v1/residents	CRUD de residentes.
parking_routes	/api/v1/parking	Gestión de permisos.
payment_routes	/api/v1/payments	Control de pagos.
access_routes	/api/v1/access	Registro de accesos.
parking_monitor_routes	/api/v1/monitoring	Monitoreo de cámaras.

Tabla 4 enrutadores funcionales

Documentación interactiva:

- Swagger UI: http://localhost:8000/docs
- ReDoc: http://localhost:8000/redoc

11.5. MODELO DE PERSISTENCIA DE DATOS

La **Tabla 5** muestra las colecciones utilizadas en Firestore, el tipo de información almacenada en cada una y los servicios responsables de su gestión dentro del sistema.

Colección	Contenido	Servicio
residentes	Datos personales.	servicio_resident.py
vehículos	Registro vehicular.	servicio_resident.py
permisos	Permisos de estacionamiento.	servicio_de_estacionamiento.py
pagos	Control de pagos.	servicio_de_pago.py
accesos_vehiculares	Entradas y salidas.	acceso_servicio.py
camera_status	Estado de monitoreo.	servicio_monitor_estacionamiento.py

Tabla 5 datos almacenados en Firebase

Las operaciones siguen un flujo definido:

HTTP Request - Router - Validación Pydantic - Servicio - Firestore - Respuesta JSON. Los modelos Pydantic garantizan la validación de datos, y los servicios gestionan la lógica de negocio. Las credenciales de acceso se cargan desde la variable FIREBASE_CREDENTIALS_PATH.

Ventajas principales:

- Sincronización en tiempo real.
- Arquitectura modular y escalable.
- Seguridad integrada mediante autenticación y credenciales protegidas.

11.6. GESTIÓN Y OPERACIÓN DEL SISTEMA

El sistema ParkSecureIoT está diseñado para operar de forma continua y segura, combinando distintos módulos que garantizan el funcionamiento automatizado del monitoreo vehicular.

Configuración General

La configuración se basa en dos archivos principales:
.env: contiene credenciales, claves JWT y parámetros de conexión.
cameras_config.json: define las cámaras, canales RTSP y zonas de detección.
Los cambios en este archivo requieren reiniciar el sistema para aplicarse.

Autenticación y Seguridad

La autenticación utiliza JWT con OAuth2, validando las credenciales en Firebase, el acceso se controla según roles y los tokens expiran automáticamente tras 30 minutos. Esto garantiza una API protegida y sesiones seguras sin exponer información sensible.

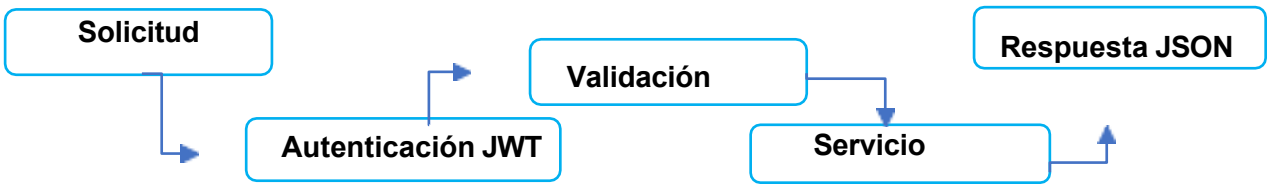
Despliegue y Ejecución

En desarrollo, el sistema se ejecuta con recarga automática:
uvicorn main:app –reload
En producción, se optimiza con múltiples procesos de trabajo:
uvicorn main:app --host 0.0.0.0 --port 8000 --workers 4
El estado del sistema se verifica con:
GET /health → {"status": "OK", "version": "1.0.0"}
Integración de Servicios
Firebase/Firestore: base de datos principal con sincronización en tiempo real.
Cámaras RTSP: capturan y transmiten video desde los puntos de control.
OCR.space: lee matrículas mediante reconocimiento óptico.
Sentry: registra y monitorea errores en producción.

11.7. FUNCIONAMIENTO INTERNO

Al iniciar, el sistema carga las configuraciones y ejecuta tareas automáticas:
Un hilo actualiza permisos vencidos cada hora.
Otro monitorea las cámaras en tiempo real con el modelo YOLO.

Durante la operación, las solicitudes siguen un flujo estructurado:



Procesamiento Concurrente
El sistema maneja múltiples tareas en paralelo: Mientras un hilo procesa la detección visual, otros gestionan solicitudes, permisos y actualizaciones, manteniendo el rendimiento estable.

Síntesis Operativa
En conjunto, ParkSecureIoT integra inteligencia artificial, monitoreo continuo y gestión en la nube para ofrecer un control vehicular inteligente, automatizado y seguro, fortaleciendo la administración y la seguridad del entorno residencial.

INSTALACIÓN Y GESTIÓN DE DEPENDENCIAS

El sistema ParkSecureIoT utiliza un entorno virtual de Python (versión 3.10 o superior), lo que garantiza compatibilidad multiplataforma y aislamiento de librerías.
Todas las dependencias se instalan desde un único archivo de requerimientos mediante el comando:
pip install -r requirements.txt
Este archivo incluye más de un centenar de librerías clasificadas en cuatro áreas principales: aprendizaje profundo (PyTorch, YOLOv8), visión por computador (OpenCV, NumPy), seguridad (Bcrypt, PyJWT) y comunicación (FastAPI, Firebase Admin SDK).

Optimización de Procesamiento
El sistema regula la velocidad de análisis por cámara mediante el siguiente parámetro:

```
cv2.waitKey(500)   # 2 FPS (valor por defecto)
```

Ajustando este valor se puede equilibrar la carga de CPU y la precisión de detección:

- cv2.waitKey(1000) - 1 FPS (menor uso de CPU)
- cv2.waitKey(100) - 10 FPS (mayor precisión, más consumo)

Compatibilidad con GPU y Rendimiento
ParkSecureIoT se ejecuta por defecto en CPU, pero detecta automáticamente entornos con GPU CUDA para acelerar el procesamiento.
La activación manual se realiza con:

```
pip install torch torchvision --index-url https://download.pytorch.org/whl/cu118
```

Tabla 6 presenta los requisitos y consideraciones de instalación para las diferentes plataformas soportadas por el sistema, incluyendo dependencias necesarias para OpenCV y opciones de aceleración mediante GPU.

Plataforma	Consideraciones
Windows	Requiere redistribuibles de Visual C++ para OpenCV
Linux	Necesita librerías de OpenGL (`libgl1-mesa-glx`)
macOS (M1/M2)	Compatible mediante compilación automática
GPU NVIDIA	Soporte opcional mediante CUDA Toolkit

Tabla 6 consideraciones de compatibilidad por plataforma

Rendimiento típico:

- CPU Intel i5 : 2–3 FPS por cámara
- GPU NVIDIA GTX 1660 : 5–20 FPS por cámara

Fijación de Versiones y Estabilidad

Todas las dependencias están bloqueadas con versiones específicas (por ejemplo: fastapi==0.120.0), asegurando que el sistema sea reproducible y estable entre entornos de desarrollo y producción.

11.8 CONFIGURACIÓN Y EJECUCIÓN DEL SISTEMA

Archivos esenciales

Tras instalar las dependencias del entorno, se configuran las credenciales y variables necesarias para la conexión con Firebase y las cámaras RTSP.
La Tabla 7 resume los archivos fundamentales que permiten la correcta operación del sistema.

Archivo	Descripción	Obligatorio
`.env`	Contiene las variables de entorno (Firebase, JWT, OCR)	Si
`secret-Firebase.json`	Credenciales del SDK de Firebase	Si
`cameras_config.json`	Definición y parámetros de las cámaras RTSP	Opcional
`main.py`	Punto de entrada de la aplicación FastAPI	Si
`requirements.txt`	Lista de dependencias de Python	Si

Tabla 7: archivos principales de configuración del sistema

Inicialización del servidor

El sistema puede ejecutarse en dos modos según el entorno

Modo desarrollo:

```
uvicorn main:app --reload --host 0.0.0.0 --port 8000
```

Permite recarga automática ante cambios en el código, ideal para pruebas locales.

Modo producción:

```
uvicorn main:app --host 0.0.0.0 --port 8000 --workers 4
```

Utiliza múltiples procesos de trabajo para mejorar el rendimiento y la estabilidad.

Arquitectura general del sistema

La aplicación se compone de módulos independientes que interactúan a través de FastAPI y WebSockets, garantizando una comunicación eficiente y en tiempo real.

La tabla 8 muestra los principales requisitos de software y compatibilidad que deben considerarse durante la instalación y ejecución del sistema en distintos sistemas operativos y entornos de hardware.

Subsistema	Archivos clave	Función
Autenticación	`auth_service.py`, `auth_routes.py`	Gestión de usuarios y emisión de tokens JWT
Control de acceso	`access_service.py`, `plate_detection_service.py`	Validación de permisos y detección de matrículas
Monitoreo	`parking_monitor_service.py`	Análisis en tiempo real mediante YOLOv8
Persistencia de datos	Firebase / Firestore	Almacenamiento de residentes, permisos y registros

Tabla 8: componentes principales del sistema

El sistema emplea Firestore y Cloud Storage para la persistencia de datos y almacenamiento de imágenes, garantizando escalabilidad y disponibilidad continua.

Rendimiento y optimización

El rendimiento del sistema depende del número de cámaras activas y del tipo de hardware. En la práctica, el modelo logra un equilibrio entre velocidad de detección y consumo de recursos. En la tabla 9 muestra el consumo estimado de memoria RAM de los componentes principales del sistema, proporcionando una referencia de los recursos necesarios para su funcionamiento.

Componente	Uso estimado de RAM
FastAPI base	~100 MB
Modelo YOLOv8	~200 MB por instancia
OpenCV (por cámara)	60–100 MB
PyTorch runtime	~500 MB

Tabla 9: requisitos y parámetros de rendimiento del sistema

Memoria recomendada: 4 GB para una cámara o 8 GB en entornos con monitoreo múltiple.
El procesamiento de video se regula mediante el intervalo de fotogramas:

```
cv2.waitKey(500)   # 2 FPS (valor por defecto)
cv2.waitKey(1000)  # 1 FPS (menor carga)
cv2.waitKey(100)   # 10 FPS (mayor precisión)
```

11.9. SISTEMA DE MONITOREO DE ESTACIONAMIENTO EN TIEMPO REAL

El módulo de monitoreo permite supervisar en tiempo real las áreas de estacionamiento mediante cámaras RTSP (Real-Time Streaming Protocol), utilizando el modelo YOLOv8 para la detección automática de vehículos. La información de ocupación generada se actualiza continuamente y se distribuye mediante una API REST (Representational State Transfer) y canales WebSocket, lo que permite una comunicación eficiente y de baja latencia con el sistema central.

Arquitectura general
El subsistema está compuesto por tres elementos principales:

- **Motor de detección (ParkingMonitorService**): Ejecuta el modelo YOLOv8 sobre los flujos RTSP, controlando la captura de video, la detección de vehículos y el cálculo de ocupación por plaza.

- **Capa API (parking_monitor_routes.py):** Expone los endpoints REST y WebSocket para la consulta y gestión de cámaras, transmisión en vivo y visualización del estado de ocupación.

- **Configuración persistente (cameras_config.json**): Almacena los metadatos de las cámaras, canales RTSP y zonas de estacionamiento; se actualiza automáticamente tras cada cambio de configuración.

Proceso de detección

El sistema utiliza el modelo yolov8n.pt (Ultralytics) con una frecuencia de inferencia de 2 FPS por cámara, procesando las clases car, motorcycle, bus y truck.
Cada detección se compara con las coordenadas de las plazas configuradas: si existe superposición, la plaza se marca como ocupada y se actualiza su estado en Firebase junto con el nivel de confianza.

```
# Ejemplo de configuración de velocidad de inferencia
cv2.waitKey(500)   # 2 FPS (valor por defecto)
```

API de monitoreo
La comunicación entre el servidor y los clientes se realiza mediante los siguientes endpoints como se observa en la tabla 10:

Endpoint	Método	Descripción
`/api/v1/monitoring/cameras`	GET / POST	Lista o registra cámaras RTSP.
`/api/v1/monitoring/cameras/{id}/occupancy`	GET	Devuelve la ocupación actual por cámara.
`/api/v1/monitoring/cameras/{id}/stream`	GET	Transmite el video en formato MJPEG.
`/api/v1/monitoring/ws/occupancy`	WS	Envía el estado global de ocupación cada 2 segundos.
`/api/v1/monitoring/ws/cameras/{id}`	WS	Envía actualizaciones por cámara cada segundo.

Tabla 10: Endpoints de comunicación

Transmisión y actualización visual
Flujos MJPEG: transmiten a ~20 FPS con codificación base64.

Verde: plaza libre.
Rojo: plaza ocupada.

WebSocket global: difunde el estado total cada 2 segundos.
WebSocket por cámara: actualiza individualmente cada segundo.

Fiabilidad y manejo de fallos
El sistema ejecuta automáticamente start_all_monitoring () al inicio de la aplicación. Si una cámara pierde conexión, se clasifica como ERROR, mientras las demás permanecen activas.
Los estados operativos posibles son:

ACTIVE: monitoreo normal.
INACTIVE: cámara detenida.
ERROR: conexión fallida o flujo interrumpido.

11.10. CONFIGURACIÓN DE LA CÁMARA

El archivo cameras_config.json define la estructura y parámetros de todas las cámaras integradas en el sistema ParkSecureIoT, incluyendo sus canales RTSP y las coordenadas de las plazas de estacionamiento. Este archivo constituye la base de configuración estática que utiliza el sistema para inicializar los servicios de monitoreo en tiempo real y control de acceso vehicular.

Estructura general del archivo
La configuración sigue una jerarquía sencilla basada en JSON:

- **Nivel principal:** cameras (lista de cámaras registradas)
- **Objeto cámara**: contiene identificador, tipo, canal RTSP y estado.
- **Objeto plaza (opcional):** define las coordenadas de los espacios de estacionamiento monitoreados.

En la Tabla 11 presenta las principales propiedades asociadas a las cámaras registradas en el sistema, describiendo los atributos utilizados para su identificación, configuración, monitoreo y gestión dentro de la plataforma de control de acceso y supervisión vehicular.

Propiedad	Descripción
id	Identificador único de cámara
name	Nombre descriptivo
camera_type	Tipo de cámara (entrada, salida, parqueadero)
rtsp_url	Enlace RTSP del canal NVR
status	Estado actual (active, inactive, error)
plazas	Lista de plazas monitoreadas (solo en cámaras de parqueo)

Tabla 11: propiedades cámaras

Representación de plazas de estacionamiento
Tabla 12 cada plaza corresponde a un rectángulo dentro del marco de video, definido por las coordenadas superiores e inferiores del área ocupada:

Propiedad	Descripción	Ejemplo
id	Identificador único dentro de la cámara	1
coordinates	Coordenadas del rectángulo de la plaza	{x1:209, y1:381, x2:494, y2:566}
occupied	Estado de ocupación detectado por YOLO	true
confidence	Nivel de confianza de detección	0.85

Tabla 12: Propiedades del objeto plaza en el sistema de monitoreo

El sistema usa el origen (0,0) en la esquina superior izquierda del fotograma, siguiendo la convención estándar de coordenadas en imágenes.

Ubicación de cámaras
ParkSecureIoT utiliza 4 cámaras cada una con funciones distintas:
La Tabla 13 muestra la ubicación y función de las cámaras implementadas en el sistema, diferenciando aquellas destinadas al control de acceso de las utilizadas para el monitoreo de parqueaderos.

Ubicación	Propósito	Soporta plazas
Entrada	Control de acceso, lectura de matrículas	NO
Salida	Control de salida y validación de permisos	NO
Parqueadero lateral / frente	Monitoreo de plazas en tiempo real	SI

Tabla 13: Función cámaras

Las cámaras de acceso ejecutan solo detección de matrículas, mientras que las de parqueadero gestionan la ocupación de espacios utilizando YOLOv8.
Configuración RTSP
Las transmisiones siguen la estructura estándar: `rtsp://[usuario]:[contraseña]@[IP_NVR]: [puerto]/Streaming/Channels/[canal]` Por ejemplo: `rtsp://admin:Amparo2025@192.168.1.73:554/Streaming/Channels/1502`
Las credenciales se almacenan en .env y las direcciones RTSP se registran directamente en cameras_config.json.

Flujo de configuración y mantenimiento

- **Carga inicial:** el servicio lee el archivo cameras_config.json al iniciar la aplicación.
- **Actualización:** las cámaras pueden gestionarse mediante la API (/api/v1/monitoring/cameras).

- **Sincronización**: los cambios se guardan automáticamente y pueden replicarse en Firebase.

La definición manual de nuevas cámaras o plazas se realiza mediante edición directa del archivo o a través de los endpoints REST.

11.11. CONFIGURACIÓN DE LA PLAZA

Cada plaza de estacionamiento en el sistema ParkSecureLot representa un área rectangular dentro del campo de visión de una cámara. Su configuración permite al sistema detectar de forma automática si un espacio está ocupado o libre en tiempo real mediante visión por computador.

Estructura de Datos

Las plazas se definen en el archivo cameras_config.json, que almacena su ubicación, estado y nivel de confianza del modelo YOLO.

Ejemplo básico:

```
{
  "id": 1,
  "coordinates": { "x1": 209, "y1": 381, "x2": 494, "y2": 566 },
  "occupied": false,
  "confidence": null,
  "last_update": "2025-11-01T09:30:08"
}
```

Cada registro contiene un identificador, las coordenadas del área en píxeles, el estado de ocupación y la última actualización de detección.

Flujo de Configuración

1. Se captura una imagen base desde la cámara para definir las zonas de estacionamiento.

2. A través de la interfaz se dibujan los rectángulos que delimitan las plazas.
3. Las coordenadas generadas se envían al backend usando:
   `POST /api/v1/monitoring/cameras/{camera_id}/plazas/batch`
4. El sistema valida visualmente las áreas configuradas y activa el monitoreo automático con:

   `POST /api/v1/monitoring/cameras/{camera_id}/start`

Integración con el Sistema de Monitoreo

El módulo parking_monitor_service se encarga de analizar los flujos de video de cada cámara, aplicar el modelo YOLOv8 y actualizar los estados en tiempo real. El sistema verifica si los objetos detectados (vehículos) se superponen con el área definida de una plaza. Si la intersección es positiva, la plaza se marca como ocupada y se actualizan los datos asociados.

Funcionamiento Interno
El servicio se ejecuta en hilos paralelos, uno por cámara, permitiendo el procesamiento simultáneo de múltiples transmisiones RTSP. El sistema mantiene un equilibrio entre rendimiento y precisión, con un intervalo de procesamiento de 2 FPS por cámara, suficiente para entornos de vigilancia sin saturar los recursos del servidor.

Cálculo de Ocupación
El método _process_frame() consolida los resultados por cámara y genera estadísticas globales: de esta forma, se obtiene el porcentaje de ocupación total, el número de espacios disponibles y la confianza general del modelo.

Transmisión de Video y Comunicación en Tiempo Real
El sistema de monitoreo de estacionamiento implementa un mecanismo de comunicación continua que permite observar en tiempo real el estado de las cámaras y las plazas disponibles. Opera mediante dos canales sincronizados:

- **Canal de video (MJPEG):** convierte los flujos RTSP en secuencias JPEG y los transmite de forma continua al panel de control, sin necesidad de recargar la interfaz.

- **Canal de datos (WebSocket):** envía actualizaciones de ocupación cada segundo (por cámara) o cada dos segundos (estado global), manteniendo coherencia entre imagen y datos.

El sistema utiliza almacenamiento en caché y control de hilos para garantizar estabilidad, incluso con múltiples usuarios conectados simultáneamente, ofreciendo una supervisión fluida y en tiempo real.

Sistema Automatizado de Control de Acceso
Este módulo identifica vehículos y valida su autorización de ingreso o salida mediante visión artificial, OCR y conexión con Firebase. Las cámaras instaladas en los puntos de acceso capturan imágenes, que se procesan con el modelo YOLOv8 para detectar el vehículo. Luego, un servicio OCR extrae el texto de la matrícula. Posteriormente, el sistema verifica si el vehículo se encuentra registrado y posee un permiso activo y vigente en la base de datos. Si la validación es positiva, se autoriza el acceso; de lo contrario, el evento se marca como denegado y se registra con evidencia visual y temporal.

Procedimiento General del Control de Acceso

1. **Captura de imagen**: la cámara RTSP obtiene un fotograma del punto de control.

2. **Detección de vehículo:** el modelo YOLOv8 identifica el vehículo en la imagen.

3. **Lectura de matrícula:** el recorte del vehículo se envía al servicio OCR para extraer el texto.
4. **Verificación de permisos:** se consulta en Firebase si el vehículo existe y tiene un permiso activo y vigente.
5. **Decisión automática:** si cumple las condiciones, se concede acceso; de lo contrario, se deniega.

6. **Registro del evento:** el sistema guarda la matrícula, hora, imágenes y resultado del intento.

Cada evento, permitido o rechazado, se almacena en Firestore, lo que permite trazabilidad, generación de reportes y auditoría. El sistema no acciona directamente las barreras físicas, pero

envía la decisión digital a los controladores externos, que ejecutan la apertura o el bloqueo según corresponda.

Detección de Matrículas

El proceso de detección y reconocimiento opera en dos fases principales:
YOLOv8: identifica la ubicación del vehículo dentro del fotograma, obteniendo coordenadas y nivel de confianza. **OCR (Reconocimiento de caracteres):** lee y normaliza el texto de la matrícula detectada. Cada resultado incluye los campos: matrícula, tipo de vehículo, confianza, imágenes codificadas y hora del evento. El sistema responde en pocos segundos, funcionando de forma independiente y eficiente.

11.12. INTERFAZ API DEL SISTEMA DE CONTROL DE ACCESO

El sistema ParkSecureLot dispone de una API REST desarrollada con FastAPI, que gestiona todo el flujo automatizado de detección, validación y registro de accesos vehiculares. Esta interfaz permite la integración directa entre las cámaras RTSP, el backend y los clientes administrativos.

Endpoints principales

- POST /detect-plate Ejecuta la detección del vehículo y lectura de la matrícula.
- Se usa para calibrar cámaras o verificar reconocimiento sin registrar eventos.

- POST /access-from-camera Realiza el flujo completo: detección, validación de permiso y registro automático en Firestore.

- POST /: Permite registrar manualmente un acceso cuando el reconocimiento automático falla.

- GET /: Devuelve el historial general de accesos registrados.

- GET /{acceso_id} Consulta los detalles de un evento específico para auditoría o revisión.

Ejemplo de solicitud y respuesta

Solicitud:

```
POST /api/v1/accesos-vehiculares/access-from-camera
{
  "camera_id": 1502,
  "tipo": "entrada"}
```

Respuesta (acceso permitido):

```
{
  "success": true,
  "message": "Acceso permitido para placa ABC123",
  "acceso": { "resultado": "permitido", "motivo": "Permiso activo" }
}
```

Respuesta (acceso denegado):

```
{
```

```
  "success": true,
  "message": "Acceso denegado para placa XYZ789",
  "acceso": { "resultado": "denegado", "motivo": "Vehículo no registrado" }
}
```


Documentación interactiva
La API genera automáticamente su documentación técnica mediante Swagger UI y ReDoc, disponibles en:

http://localhost:8000/docs
http://localhost:8000/redoc

Estas herramientas permiten visualizar y probar los endpoints, revisar parámetros y analizar las respuestas del sistema en tiempo real.

Funcionalidades Principales del Backend

El backend de ParkSecureLot constituye el núcleo lógico y operativo del sistema. Su objetivo principal es administrar los recursos del estacionamiento residentes, vehículos, permisos y accesos y coordinar la comunicación entre los distintos subsistemas automatizados.

Las funcionalidades clave incluyen:

Autenticación y seguridad: Protección mediante tokens JWT y roles de usuario, asegurando acceso controlado a los recursos del sistema.

Gestión de residentes y vehículos: Registro y mantenimiento de información personal y vehicular, con validaciones de unicidad y vinculación entre entidades.

Permisos de estacionamiento: Administración completa del ciclo de vida de los permisos (creación, activación, expiración o renovación).

Procesamiento de pagos: Vinculación directa entre pagos y permisos, garantizando que solo las transacciones confirmadas habiliten el acceso.

Registro de accesos: Documentación automatizada o manual de los eventos de entrada y salida vehicular, con trazabilidad total.

Toda la información se almacena en Firebase/Firestore, proporcionando sincronización en tiempo real, persistencia y capacidad de auditoría. Además, el backend ejecuta tareas en segundo plano para mantener la consistencia del sistema, como la expiración de permisos o la verificación de cámaras activas.

Autenticación y Autorización

El sistema implementa un esquema de autenticación segura basado en JWT (JSON Web Tokens) y OAuth2 Password Bearer. Cada usuario obtiene un token temporal que se valida en cada solicitud, restringiendo el acceso a las rutas según los roles asignados.

Las contraseñas se cifran con bcrypt, y las solicitudes pueden autenticarse tanto por encabezado como por parámetro, lo que permite integrar servicios como las transmisiones en vivo de cámaras. De esta manera, se garantiza un control de acceso confiable, seguro y adaptable a cada módulo.

Gestión de Residentes y Vehículos
Este módulo constituye la base de datos central del sistema, el cual permite administrar la información de los residentes y sus vehículos asociados, garantizando trazabilidad y coherencia entre los distintos subsistemas.

Funciones principales:

- Registro, actualización y eliminación lógica de residentes y vehículos.
- Asociación múltiple de vehículos por residente.
- Validación automática de correos y matrículas duplicadas.

Relaciones con otros módulos:

- **Permisos:** cada residente puede generar o renovar sus permisos de estacionamiento.
- **Control de acceso**: las matrículas registradas se usan para validar entradas y salidas.
- **Pagos:** los registros financieros se vinculan con los permisos activos de cada residente.

Sistema de Permisos de Estacionamiento
Gestiona la autorización temporal de uso del parqueadero, vinculando residentes, vehículos y pagos. Cada permiso tiene una vigencia definida y su estado se actualiza automáticamente según la fecha y el estado del pago.
Estados del permiso:

- Pendiente pago: creado, pero no activado.
- activo: permite el acceso al estacionamiento.
- vencido: permiso expirado, sin acceso.

El sistema ejecuta tareas automáticas que revisan periódicamente los permisos y los marcan como vencidos cuando la fecha límite ha pasado, manteniendo la base de datos actualizada y el acceso bajo control.

Gestión de Pagos y Activación de Permisos
El módulo de pagos se encarga de validar y registrar las transacciones que respaldan los permisos activos. Cada pago se asocia a un permiso mediante un identificador único, almacenando datos como monto, fecha, método y estado (pendiente, confirmado o fallido).
Cuando el pago se confirma, el sistema cambia el estado del permiso a activo, habilitando automáticamente el acceso del vehículo. De este modo, se garantiza coherencia entre la información financiera y el control de acceso, además de mantener un historial completo de todas las transacciones para auditoría y control administrativo.

Registro de Acceso Manual
Permite documentar entradas y salidas vehiculares sin depender del sistema automatizado de cámaras. Está diseñado para casos especiales, fallos del sistema o accesos controlados por el personal de seguridad.

Características principales:

- Registro rápido mediante formulario administrativo.

- Asociación automática con residentes y vehículos existentes.
- Validación del permiso activo antes de autorizar el ingreso.
- Almacenamiento completo del evento (placa, hora, tipo, resultado, evidencia).

Casos de uso:

- Fallos del sistema automatizado (YOLO/OCR fuera de servicio).
- Accesos de emergencia o visitantes sin pre-registro.
- Correcciones o registros manuales de eventos anteriores.

Integración: El módulo se conecta con los sistemas de vehículos, residentes, permisos y control de acceso, unificando los reportes y garantizando una trazabilidad total del flujo de vehículos.

11.13. ÁREAS EN SEGUNDO PLANO Y AUTOMATIZACIÓN

El sistema ParkSecureIoT cuenta con procesos automáticos que operan en segundo plano, asegurando la continuidad de funciones críticas sin intervención del usuario. Estos mecanismos garantizan que tareas como la verificación de permisos y el monitoreo de cámaras se mantengan activas y sincronizadas en todo momento.

Los dos procesos principales son:

- **CRON de caducidad de permisos**, que actualiza de forma automática los permisos vencidos.
- **Monitoreo automático de cámaras**, encargado de iniciar el análisis de video desde el arranque del sistema.

Ambos funcionan de manera autónoma, contribuyendo a la estabilidad, disponibilidad y autogestión del sistema.

CRON de Caducidad de Permisos
Este proceso revisa periódicamente la base de datos y marca como vencidos los permisos cuya fecha de vigencia ha expirado. Se ejecuta una vez por hora, de forma totalmente autónoma.

Características principales:

- Funciona sin interrumpir otros servicios.
- Se detiene automáticamente al cerrar la aplicación.
- Garantiza precisión en el control de accesos, evitando usos no autorizados.

Flujo de operación:

- Consulta los permisos activos en Firebase.
- Compara la fecha actual con la de vencimiento.
- Actualiza el estado y registra la acción en consola.

Monitoreo Automático de Cámaras

Durante la inicialización del sistema, se detectan las cámaras RTSP configuradas en cameras config. Json. Si están disponibles, el sistema inicia automáticamente el monitoreo continuo por cámara; en caso contrario, entra en modo estático y solo procesa capturas bajo demanda. La Tabla 14 muestra los modos de funcionamiento del sistema, especificando su mecanismo de activación y las situaciones para las cuales fueron diseñados.

Modo	Activación	Uso principal
Continuo	Automático al iniciar la aplicación	Entornos de producción y vigilancia en vivo
Estático	Manual, bajo demanda	Pruebas o entornos sin conexión RTSP

Tabla 14: Modo funcionamiento

Características:

- Procesamiento paralelo por cámara (1–2 FPS).
- Control de concurrencia mediante locks para evitar conflictos de acceso.
- Recuperación automática en caso de error o desconexión.
- Liberación ordenada de recursos al apagar el sistema.

11.14. REFERENCIA DE LA API

ParkSecureIoT expone una API RESTful construida con FastAPI, utilizada por los módulos de monitoreo, control de acceso y la interfaz web. La API utiliza formato JSON, controla autenticación por tokens JWT y organiza sus funciones por áreas:

- Autenticación: registro, inicio de sesión y emisión de tokens.
- Residentes y vehículos: administración de usuarios y sus vehículos.
- Permisos y pagos: validación, renovación y registro de trámites.
- Accesos vehiculares: registro automático de entradas y salidas.
- Monitoreo: estado en tiempo real de las plazas de estacionamiento.

Rutas principales (v1):

/api/v1/auth, /api/v1/residents, /api/v1/permits, /api/v1/payments, /api/v1/accesos-vehiculares, /api/v1/monitoring.

Los endpoints protegidos requieren cabecera:

Authorization: Bearer <token>
Ejemplo práctico: Inicio de sesión en la API
Solicitud

```
POST /api/v1/auth/login
Content-Type: application/json

{"email": "usuario@correo.com",
  "password": "12345" }
```

Respuesta esperada

```
{"access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "token_type": "bearer" }
```

Puntos de Conexión de Autenticación El sistema ParkSecureIoT utiliza un modelo de autenticación basado en tokens JWT (JSON Web Token), lo que permite validar el acceso de los usuarios de forma segura y sin mantener sesiones en el servidor.
La autenticación sigue el flujo OAuth2 con portador de contraseña, implementado mediante FastAPI y gestionado a través de la ruta principal:

/api/v1/auth

Principales operaciones

- **Registro de usuario:** permite crear nuevas cuentas con roles asignados (administrador, residente o seguridad).
- **Inicio de sesión:** verifica las credenciales del usuario y emite un token JWT.
- **Renovación de token:** actualiza los tokens cercanos a su vencimiento sin necesidad de volver a iniciar sesión.

Flujo de autenticación

1. El usuario envía sus credenciales (correo y contraseña) al endpoint de inicio de sesión.
2. El sistema verifica las credenciales almacenadas en Firebase y genera un token JWT firmado.
3. En cada solicitud posterior, el cliente envía este token en el encabezado de autorización:
4. Authorization: Bearer <token>

El servidor valida el token y autoriza el acceso según el rol del usuario
Control de acceso y roles
El sistema implementa control de acceso basado en roles (RBAC), garantizando que cada usuario solo pueda acceder a los recursos que le corresponden:

- **Administrador:** acceso completo al sistema.
- **Residente:** gestión de su información, vehículos, permisos y pagos.
- **Seguridad:** control de accesos y supervisión de cámaras.

Este mecanismo asegura la integridad de los **datos** y una **gestión eficiente de la seguridad** en todos los módulos del sistema.

Referencia de la API de Control de Acceso
El módulo de control de acceso vehicular en ParkSecureIoT implementa un sistema automatizado de reconocimiento de matrículas y validación de permisos de entrada o salida. Este subsistema se comunica mediante una API RESTful, que permite la integración directa con las cámaras de seguridad y los servicios de detección basados en YOLOv8 y OCR (Reconocimiento Óptico de Caracteres).

Ruta base
/api/v1/accesos-vehiculares
Todas las solicitudes requieren autenticación mediante token JWT. Los registros y validaciones se almacenan en Firebase/Firestore, asegurando persistencia y sincronización entre módulos.

Despliegue y Operaciones
El sistema ParkSecureIoT fue implementado bajo una arquitectura diseñada para garantizar estabilidad, seguridad y monitoreo continuo tanto en desarrollo como en producción.
Durante el desarrollo, se empleó Uvicorn con recarga automática para agilizar las pruebas

uvicorn main:app --reload --host 0.0.0.0 --port 8000: En entornos productivos, el sistema se ejecuta con múltiples procesos de trabajo para mejorar el rendimiento y soportar más solicitudes simultáneas:

uvicorn main:app --host 0.0.0.0 --port 8000 --workers 4: Para facilitar su despliegue, se utiliza una imagen Docker que incluye todas las dependencias necesarias, asegurando un entorno controlado y replicable en cualquier servidor:

FROM python:3.11-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]

El ciclo de vida de la aplicación está gestionado por FastAPI, que inicializa procesos en segundo plano y libera recursos al apagarse. Por ejemplo, la tarea automática que actualiza los permisos vencidos:

```
def      permisos_cron_background():
    import time
    while True:
        actualizar_permisos_vencidos()
        time.sleep(3600)
```

El sistema también integra endpoints de monitoreo que permiten verificar su estado en tiempo real:

GET /health

Respuesta esperada:

```
{"status": "OK", "message": "Sistema funcionando correctamente"}
```

En materia de seguridad, se implementó autenticación JWT, políticas CORS y un archivo de configuración. env con las credenciales y parámetros principales. Asimismo, el monitoreo de cámaras se gestiona con hilos independientes que procesan flujos RTSP mediante OpenCV y modelos YOLOv8, detectando vehículos y actualizando la información en tiempo real. También, se incorporaron mecanismos de respaldo automático y rotación de registros para garantizar la continuidad operativa y la trazabilidad del sistema.

Configuración de Producción: La configuración de producción de ParkSecureIoT garantiza que el sistema opere de manera estable, segura y eficiente bajo cargas reales. Esta etapa implica ajustes específicos en el servidor, el middleware, las tareas en segundo plano y las políticas de seguridad.

1. **Configuración del servidor**

El sistema se ejecuta sobre Uvicorn, servidor ASGI (Asynchronous Server Gateway Interface) optimizado para aplicaciones FastAPI. Durante el desarrollo, se utiliza un modo con recarga automática para facilitar pruebas:

uvicorn main:app --reload --host 0.0.0.0 --port 8000: En producción, se habilita el procesamiento concurrente con múltiples trabajadores:
uvicorn main:app --host 0.0.0.0 --port 8000 --workers 4: Cada trabajador carga el modelo YOLO y ejecuta las tareas de monitoreo, por lo que el número ideal se calcula según la fórmula:
workers = (2 × núcleos_CPU) + 1: Este enfoque optimiza el uso de recursos sin sobrecargar la memoria.

2. **Ciclo de vida y tareas en segundo plano**

El sistema utiliza el gestor de contexto Lifespan de FastAPI, que controla la inicialización y cierre del sistema:

- En el inicio, se activa un hilo CRON que actualiza los permisos vencidos y se inicia la detección automática de cámaras RTSP.
- Durante el apagado, se detienen los hilos de monitoreo y se liberan los recursos del modelo YOLO.

Ejemplo del proceso automatizado de permisos:

```
def      permisos_cron_background():
    while True:
        actualizar_permisos_vencidos()
        time.sleep(3600)   # Ejecuta cada hora
```

3. **Configuración del middleware (CORS)**

Para permitir la comunicación entre el backend y las interfaces frontend, se utiliza CORS (Cross-Origin Resource Sharing). En desarrollo se permite cualquier origen, mientras que en producción se restringe estrictamente a dominios autorizados:

```
app.add_middleware(
    CORSMiddleware,
    allow_origins=["https://parksecurelot.com",
"https://admin.parksecurelot.com"],
    allow_methods=["GET", "POST", "PUT", "DELETE"],
    allow_headers=["Authorization",  "Content-Type"]
)
```

Esta configuración evita accesos no autorizados desde dominios externos.

4. **Variables de entorno**

Las credenciales y parámetros sensibles se almacenan en archivos .env específicos por entorno:

SECRET_KEY=clave_segura_32_caracteres
FIREBASE_CREDENTIALS_PATH=secret-Firebase.json
OCR_API_KEY=K87982429188957
SENTRY_DSN=https://sentry.io/project-id

5. **Seguridad en producción**

Se aplican medidas de seguridad en varios niveles:

- **Autenticación:** JWT con claves de al menos 32 caracteres y expiración corta (15–30 minutos).
- **Red:** HTTPS obligatorio con TLS 1.2+ y firewall en el puerto 8000.
- **Archivos:** credenciales protegidas y cámaras RTSP con contraseñas seguras.
- **Aplicación:** validación de entradas con Pydantic y manejo de errores sin trazas visibles.

Ejemplo de configuración HTTPS con Nginx:

```
server {
    listen 443 ssl;
    server_name        api.parksecurelot.com;
    ssl_certificate        /etc/ssl/cert.pem;
    ssl_certificate_key     /etc/ssl/key.pem;
    location / {
        proxy_pass http://127.0.0.1:8000;
    }
}
```

6. **Despliegue y automatización**

El despliegue puede realizarse mediante Docker o Systemd, según la infraestructura:
Ejemplo con Dockerfile de producción:

FROM python:3.11-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000", "--workers", "4"]
Ejemplo con Systemd:
[Service]
ExecStart=/opt/parksecurelot/.venv/bin/uvicorn main:app --workers 4
Restart=always
User=parksecure

Estas configuraciones aseguran que el sistema mantenga un equilibrio entre rendimiento, seguridad y estabilidad en entornos de producción.

12. RESULTADOS

En este apartado se presentan los resultados obtenidos durante el desarrollo e implementación del sistema **ParkSecureIoT**, los cuales se dividen en dos secciones **resultados cualitativos** y **resultados cuantitativos**. El propósito de este capítulo es mostrar de forma clara y organizada tanto el **rendimiento técnico del modelo de detección basado en YOLOv8**, como la **puesta en funcionamiento del sistema completo** dentro del entorno real de pruebas

12.1. Resultados cualitativos

En esta sección se describen los resultados observados desde una perspectiva visual y funcional, centrados en la interfaz de usuario y la experiencia de operación del sistema. Se evalúa la capacidad del software para ofrecer una interacción intuitiva, fluida y eficiente, permitiendo al usuario final en este caso, el administrador del parqueadero realizar tareas como la gestión de residentes, el control de accesos vehiculares, la supervisión de plazas de estacionamiento y la visualización en tiempo real de las cámaras conectadas. El diseño del entorno fue desarrollado bajo principios de usabilidad y simplicidad visual, destacando la disposición jerárquica de los elementos, la coherencia gráfica y la respuesta dinámica ante las acciones del usuario. Asimismo, se validó la integración entre el módulo de detección inteligente de vehículos, el backend de gestión (FastAPI) y la interfaz desplegada en la nube (Vercel), asegurando una comunicación estable y confiable.

12.2. Lobby del sistema (Interfaz principal del usuario)

El lobby del sistema ParkSecureIoT constituye la página de inicio y presentación general del proyecto. Su propósito es ofrecer al usuario una experiencia informativa y visualmente atractiva, permitiéndole conocer de manera intuitiva las funcionalidades principales del sistema antes de acceder a los módulos internos de gestión. Esta sección actúa como la carta de presentación del software, transmitiendo los valores de tecnología, automatización y seguridad que caracterizan al sistema.

12.2.1. Interfaz principal

En la pantalla inicial (ver Figura 29), el usuario encuentra una estructura moderna y responsiva, diseñada con enfoque en la claridad y accesibilidad. El encabezado contiene las opciones de navegación (“Características”, “Cómo funciona” y “Contacto”), junto con la opción de inicio de sesión.

El cuerpo principal destaca un mensaje de valor “Transforma tu parqueo en una experiencia inteligente” acompañado de un llamado a la acción que invita a explorar las funcionalidades del sistema. Esta interfaz busca generar confianza y atracción visual, utilizando tonos neutros y tipografía legible para reforzar la identidad tecnológica de la plataforma.

FIGURA 29:LOBBY [FUENTE PROPIA]

12.2.2. Sección de características del sistema

En esta sección se presentan los **módulos y capacidades principales** de ParkSecureIoT, con el objetivo de que el usuario comprenda el alcance y las ventajas del sistema. Entre las características visibles se destacan:

- **Reconocimiento de Placas:** mediante visión artificial y modelos de inteligencia artificial (YOLOv8).
- **App móvil intuitiva:** permite gestionar espacios, reservas y pagos desde el dispositivo.
- **Analytics en tiempo real:** visualización de métricas de ocupación y patrones de uso.
- **Seguridad avanzada e IoT integrado:** con sensores inteligentes y cifrado de datos.
- **Automatización total:** control completo del flujo de entrada y salida de vehículos.

Cada componente fue diseñado con el propósito de garantizar **eficiencia, seguridad y comodidad**, reflejando la integración entre hardware, software y la nube (Vercel).

FIGURA 30: BLOQUE DE CARACTERÍSTICA [FUENTE PROPIA]

12.2.3. Sección “Cómo funciona” (Proceso de implementación)

En el apartado “Implementación sin complicaciones”, se expone la **metodología de despliegue del sistema**, destacando la facilidad de instalación y la personalización del servicio. El proceso se divide en cuatro etapas clave:

1. **Instalación rápida:** montaje del hardware y software en menos de 24 horas.
2. **Configuración personalizada:** ajuste de tarifas, zonas y reglas de acceso.
3. **Capacitación del personal:** entrenamiento operativo práctico.
4. **Lanzamiento y soporte 24/7:** acompañamiento técnico permanente.

Esta sección demuestra que ParkSecureIoT no solo es una herramienta tecnológica, sino un **servicio integral de gestión inteligente de estacionamientos**, enfocado en minimizar tiempos de implementación y maximizar la experiencia del usuario.

FIGURA 31: CÓMO FUNCIONA [FUENTE PROPIA]

12.2.4. Sección de contacto (Atención al usuario)

Finalmente, el apartado de **contacto** proporciona los canales de comunicación con el equipo desarrollador.
Incluye campos de formulario para nombre, correo electrónico, teléfono y mensaje, junto con la información general del servicio (correo corporativo, dirección horarios y teléfono). el objetivo es facilitar la **retroalimentación directa** con los usuarios o administradores interesados, fortaleciendo la comunicación y la asistencia técnica personalizada.

FIGURA 32:ATENCIÓN AL USUARIO [FUENTE PROPIA]

12.3. Interfaz de Inicio de Sesión

Después de haber explorado el entorno informativo del sistema que incluye las secciones de **Lobby**, **Características**, **Cómo Funciona** y **Contacto**, el usuario puede acceder al entorno operativo mediante la **pantalla de inicio de sesión**.

Esta interfaz tiene como objetivo garantizar un **acceso seguro y personalizado** a las funciones internas del sistema. Se diseñó bajo un esquema visual minimalista y funcional, dividiendo la pantalla en dos secciones principales:

- En el panel izquierdo, se presenta la **identidad visual de ParkSecureIoT**, con el logotipo y los tres pilares del sistema: *Monitoreo Inteligente*, *Seguridad* y *Reportes*.
- En el panel derecho, se encuentra el **formulario de autenticación**, que solicita las credenciales del usuario (correo electrónico y contraseña), además de los botones “Ingresar” y “Regresar”.

El proceso de validación se conecta con el servidor a través del **backend implementado con**

FastAPI, verificando las credenciales en la base de datos de usuarios. De esta forma, se garantiza la **protección de la información y la integridad de las sesiones activas**, evitando accesos no autorizados.

La elección de una interfaz limpia, con tonos azulados y un contraste sobrio, busca transmitir una sensación de **confianza y modernidad**, coherente con la naturaleza tecnológica del sistema. Además, el diseño responsivo permite que el inicio de sesión se realice desde diferentes [illegible] fluida en

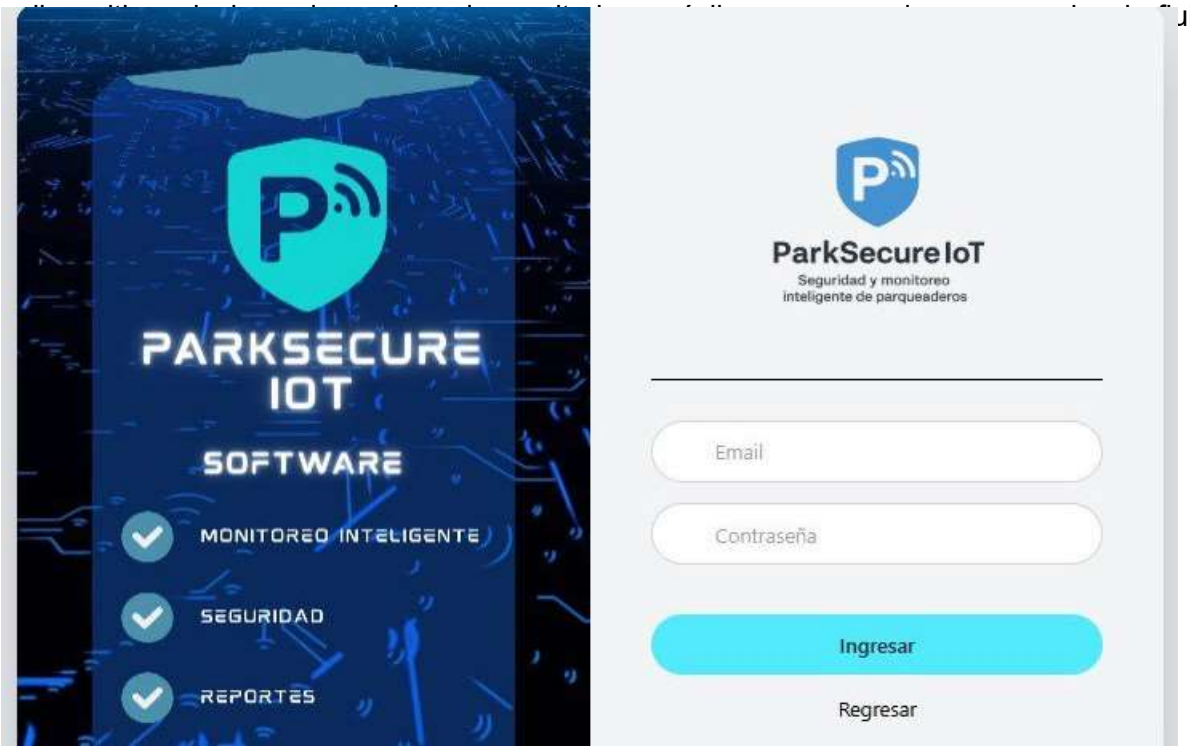


FIGURA 33:INTERFAZ DE INICIO DE SESIÓN [FUENTE PROPIA]

12.4. Dashboard de Control (Vista general del sistema)

Una vez el usuario accede con sus credenciales válidas, el sistema despliega el **Dashboard de Control**, que constituye el núcleo visual y funcional del entorno administrativo. Este panel fue diseñado con el objetivo de **proporcionar una visión global del estado del parqueadero en tiempo real**, integrando los datos provenientes de las cámaras de detección, los registros de residentes y la ocupación de plazas.

En la parte superior del panel se presentan los **indicadores generales** del sistema:

- **Total, de residentes registrados**, con el conteo de usuarios y vehículos activos.
- **Vehículos actuales**, que muestra el número de automóviles presentes en las instalaciones.
- **Ingresos del día**, indicador de las entradas registradas en la jornada.
- **Plazas ocupadas**, con el número total de espacios en uso y los disponibles.

La **actividad del día** se muestra mediante una serie de barras y métricas que facilitan la lectura

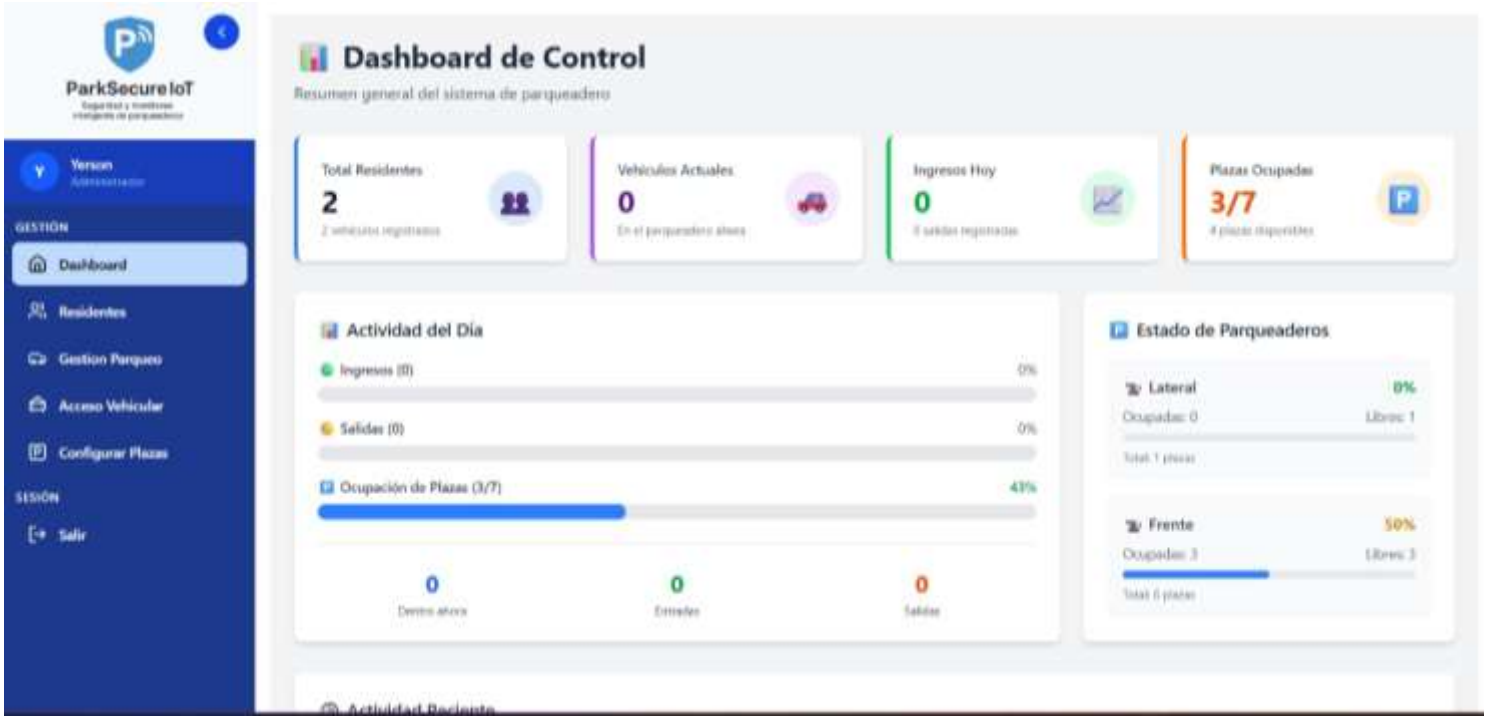


FIGURA 34:DASHBOARD DE CONTROL [FUENTE PROPIA]

12.5 Módulo de Gestión de Residentes

El módulo de **Gestión de Residentes** permite al administrador llevar un control centralizado de las personas registradas dentro del sistema ParkSecureIoT, junto con sus respectivos datos de identificación y vehículos asociados. Esta funcionalidad constituye la base del sistema de monitoreo, ya que garantiza que cada vehículo registrado esté vinculado a un usuario autorizado dentro del conjunto residencial Amparo country.

En la parte superior de la interfaz (ver Figura 35), se muestran los **indicadores principales del sistema**, que incluyen:

- **Total, de residentes:** número de usuarios registrados en la base de datos.
- **Total, de vehículos:** cantidad de automóviles asociados a los residentes activos.

Debajo de estos indicadores se encuentra una barra de búsqueda dinámica, que permite filtrar registros por nombre, cédula o cualquier otro dato relevante. La tabla principal presenta la información de cada residente organizada en columnas, incluyendo nombre, apellido, cédula, correo electrónico, teléfono, número de unidad, torre y un conjunto de acciones rápidas representadas por íconos de color. Estas acciones permiten editar, eliminar o asociar vehículos de forma inmediata, optimizando los tiempos de administración y reduciendo errores manuales

El botón **“+ Nuevo Residente”**, ubicado en la parte superior derecha, facilita el registro de nuevos usuarios mediante un formulario interactivo que valida los datos antes de enviarlos al servidor. Toda la información se almacena de forma segura en la base de datos, gestionada a través del backend implementado con **FastAPI y Firestore**, lo que garantiza la **integridad y disponibilidad de los datos**.

El diseño mantiene la línea visual del sistema general, utilizando tonos azules y verdes para distinguir los estados activos y los indicadores principales. Esta organización visual permite al administrador tener **una visión clara y rápida del estado actual de los usuarios y sus vehículos**, fortaleciendo la trazabilidad y seguridad dentro del entorno de parqueo inteligente.

FIGURA 35: MÓDULO DE GESTIÓN DE RESIDENTES [FUENTE PROPIA]

12.6. Módulo de Gestión del Parqueadero (Permisos y Control de Vigencia)

El módulo de **Gestión del Parqueadero** fue diseñado para administrar de manera organizada y transparente los permisos de uso de las plazas dentro del *Conjunto Residencial Amparo Country*. Esta herramienta permite al administrador verificar y controlar la vigencia de los permisos otorgados a los residentes, asegurando una correcta trazabilidad de los pagos y el estado de cada usuario.

En la interfaz (ver Figura 36), se muestra una tabla con la información principal de cada permiso activo, incluyendo los campos: residente, fecha de inicio y finalización del permiso, número de meses vigentes, monto, estado y confirmación de pago. El botón "+ Nuevo Permiso", ubicado en la parte superior derecha, facilita el registro de nuevas autorizaciones, permitiendo configurar el periodo de validez y el valor correspondiente al uso del espacio. Este módulo contiene una lógica de actualización automática, de manera que cuando un permiso está próximo a vencer, el sistema puede alertar al administrador para renovar o suspender el acceso vehicular del residente. Además, los estados de los permisos (activo, vencido o en renovación) se muestran con etiquetas visuales, favoreciendo la **identificación rápida y precisa** de la situación de cada usuario.

La información financiera se almacena de forma segura mediante el backend en **Firestore**, Por su parte, el proceso de autenticación se implementa utilizando **Firebase** y **FastAPI**, lo que garantiza la integridad de los datos y la validación de usuarios autorizados para acceder a la plataforma.

Este componente se hace muy esencial para la operación del conjunto, ya que contribuye al control automatizado del flujo de vehículos y a la gestión eficiente de los espacios disponibles, reduciendo el margen de error humano y mejorando la experiencia tanto de los residentes como del personal administrativo del *Amparo Country*.

FIGURA 36: MÓDULO DE GESTIÓN DEL PARQUEADERO [FUENTE PROPIA]

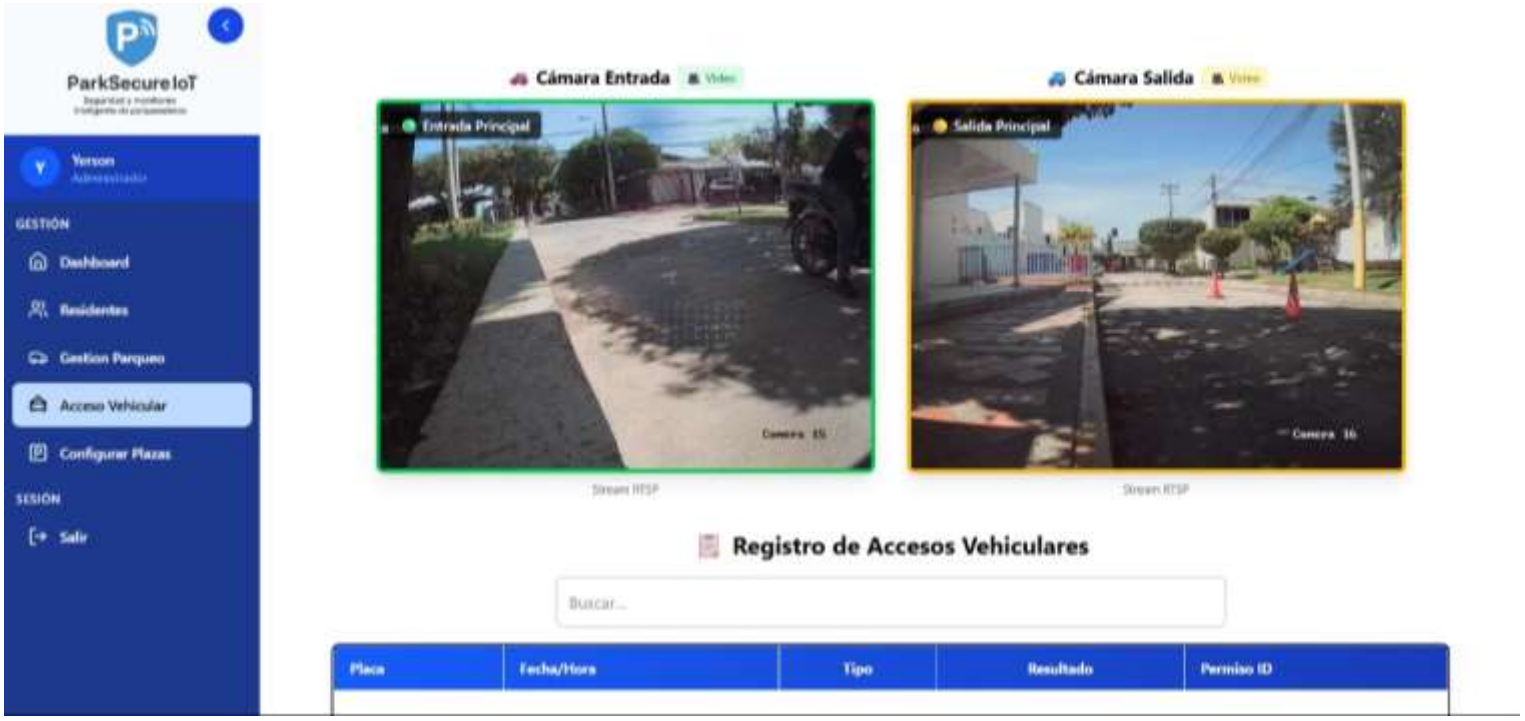
12.7. Módulo de Acceso Vehicular (Entrada y Salida Automatizada)

El módulo de **Acceso Vehicular** representa el componente más dinámico del sistema ParkSecureIoT, ya que permite el **monitoreo en tiempo real de los vehículos que ingresan y salen del conjunto Amparo Country**. Este apartado integra el sistema de visión artificial entrenado con el modelo YOLOv8, junto con la transmisión en vivo de cámaras IP configuradas mediante protocolo **RTSP (Real Time Streaming Protocol)**.

En la interfaz (ver Figura 37), se muestran dos paneles principales correspondientes a las cámaras de **entrada** y **salida**. Cada una ópera de manera independiente, pero sincronizada con la base de datos central para registrar automáticamente las placas detectadas, la hora exacta del evento y el tipo de movimiento (entrada o salida).

El sistema aplica **detección automática de matrículas** y realiza la validación de permisos activos dentro del módulo de gestión de parqueadero. Si la placa detectada pertenece a un residente con autorización vigente, el sistema marca el evento como “acceso permitido”; en caso contrario, lo clasifica como “no autorizado”, registrándolo en el historial para revisión posterior.

La parte inferior del panel presenta el **Registro de Accesos Vehiculares**, donde se almacena cada evento en formato tabular, incluyendo los campos: **placa, fecha/hora, tipo de movimiento, resultado de la validación y número de permiso asociado**. Esta información se actualiza automáticamente sin necesidad de intervención del administrador, lo que mejora la trazabilidad del flujo vehicular dentro del conjunto.



FIGURO 37: MÓDULO DE ACCESO VEHICULAR [FUENTE PROPIA]}

12.8. Prueba de Funcionamiento del Módulo de Acceso Vehicular

Como se mencionó anteriormente, el módulo de **Acceso Vehicular** permite gestionar las entradas y salidas de los vehículos residentes mediante detección automática de matrículas. Para validar su funcionamiento, se realizó una **prueba de campo en tiempo real** dentro del conjunto *Amparo Country*, donde se simularon escenarios de entrada y salida vehicular bajo condiciones normales de operación. En las imágenes (ver Figuras 38 y 39) se observa el proceso de detección de un vehículo identificado con la placa **JJZ435**. Al momento de su ingreso, el sistema reconoció automáticamente la placa mediante el modelo de inteligencia artificial implementado, y validó su permiso activo en la base de datos, registrando el evento como **"Entrada – Aprobado está registrado"**.

FIGURA 38: EJEMPLO DE ENTRADA [FUENTE PROPIA]

Posteriormente, al realizar la salida del mismo vehículo, el sistema repitió el proceso de verificación, generando el registro correspondiente como **“Salida – Aprobado está registrado”**, lo que demuestra la **consistencia y sincronización del módulo de control de acceso**.

FIGURA 39: EJEMPLO SALIDA [FUENTE PROPIA]

Ambos eventos fueron almacenados de manera automática en el **Registro de Accesos Vehiculares**, incluyendo la hora exacta, tipo de movimiento y estado del permiso. Este comportamiento confirma que la arquitectura de comunicación entre los módulos de video, detección, base de datos y panel administrativo funciona de forma eficiente y en tiempo real.

12.9. Módulo de Configuración y Monitoreo de Plazas de Estacionamiento

El módulo de **Configuración de Plazas** es una de las funcionalidades más innovadoras del sistema **ParkSecureIoT**, ya que permite la **detección automática de la ocupación de los espacios de parqueo** dentro del *Conjunto Residencial Amparo Country*. Este componente integra el sistema de visión por computadora basado en el modelo **YOLOv8**, capaz de identificar en tiempo real si una plaza se encuentra **ocupada o libre** mediante la interpretación de imágenes captadas por cámaras de vigilancia instaladas en el perímetro del parqueadero.

FIGURA 40: MONITOREO DE PLAZAS DE ESTACIONAMIENTO [FUENTE PROPIA]

En la interfaz (ver Figuras 40 y 41), el usuario administrador puede **configurar cada espacio de estacionamiento** trazando manualmente un rectángulo sobre la imagen en vivo proporcionada por la cámara. Cada zona marcada queda registrada con un identificador único (P1, P2, P3, etc.), permitiendo al sistema reconocer automáticamente su estado según la presencia o ausencia de un vehículo dentro del área definida.

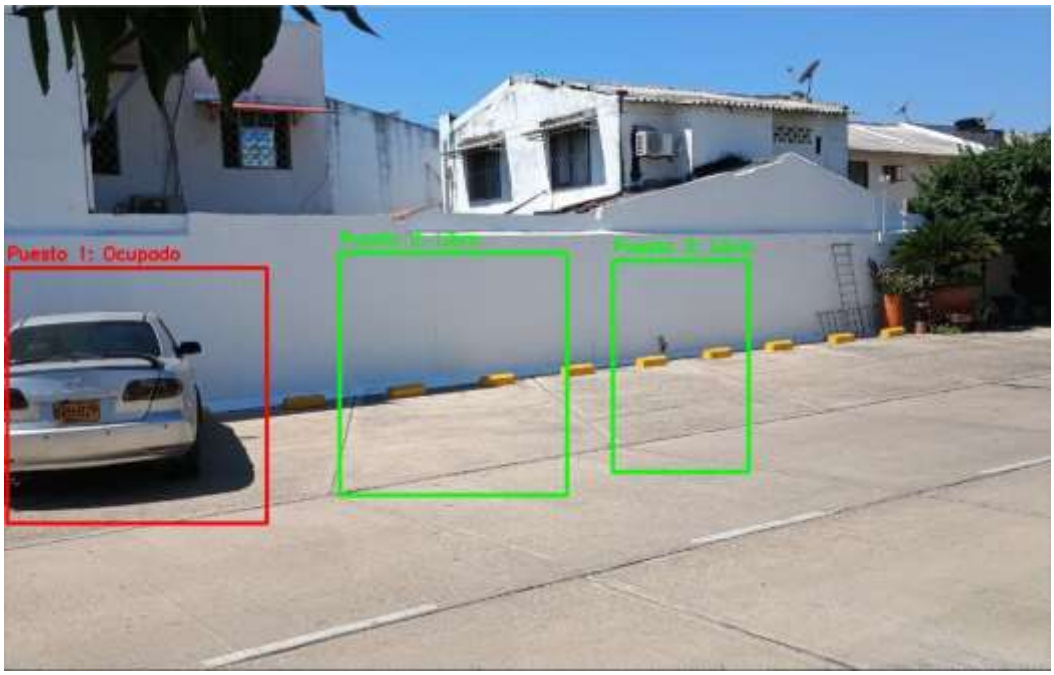


FIGURA 41: MONITOREO EN TIEMPO REAL [FUENTE PROPIA]

Una vez finalizada la configuración, el sistema almacena la información y comienza el **monitoreo automatizado**.

En las imágenes se observa cómo el sistema detecta correctamente la ocupación de los puestos:

- **Puesto 1: Ocupado** (detecta el vehículo en la zona delimitada).
- **Puestos 2 y 3: Libres**, representados mediante recuadros verdes.

Este resultado demuestra la efectividad del modelo de detección implementado, el cual fue capaz de **diferenciar correctamente los espacios ocupados y disponibles**, incluso en condiciones de iluminación natural y distintos ángulos de cámara.

Además, la interfaz web permite visualizar los resultados en tiempo real, ofreciendo un panel intuitivo con opciones para **guardar, borrar, limpiar o actualizar las zonas configuradas**, así como iniciar el proceso de monitoreo de manera inmediata.
El sistema combina **procesamiento de imágenes, inteligencia artificial y comunicación en red**, lo que garantiza un control eficiente y actualizado del estado del parqueadero sin necesidad de supervisión constante.

FIGURA 42: MONITOREO [FUENTE PROPIA]

La implementación de este módulo en el *Conjunto Amparo Country* permite mantener una gestión óptima de los espacios, mejorar la seguridad y reducir la posibilidad de conflictos por ocupación indebida, consolidando el proyecto ParkSecureIoT como una **solución integral de automatización y monitoreo inteligente de parqueaderos residenciales**.

En la imagen (ver Figura 43) se observa la captura de la cámara **Camera 13**, donde el sistema identifica simultáneamente el estado de seis plazas de estacionamiento. Cada espacio es delimitado por un rectángulo con un color distintivo:

- **Rojo:** indica una plaza ocupada.
- **Verde:** indica una plaza libre.

Durante la prueba, el sistema detectó correctamente la ocupación de los espacios:

- **P1, P2, P3 y P5** fueron identificadas como *ocupadas*, al encontrarse vehículos dentro de los límites definidos.
- **P4 y P6** fueron reconocidas como *libres*, evidenciando la correcta discriminación entre zonas ocupadas y vacías.

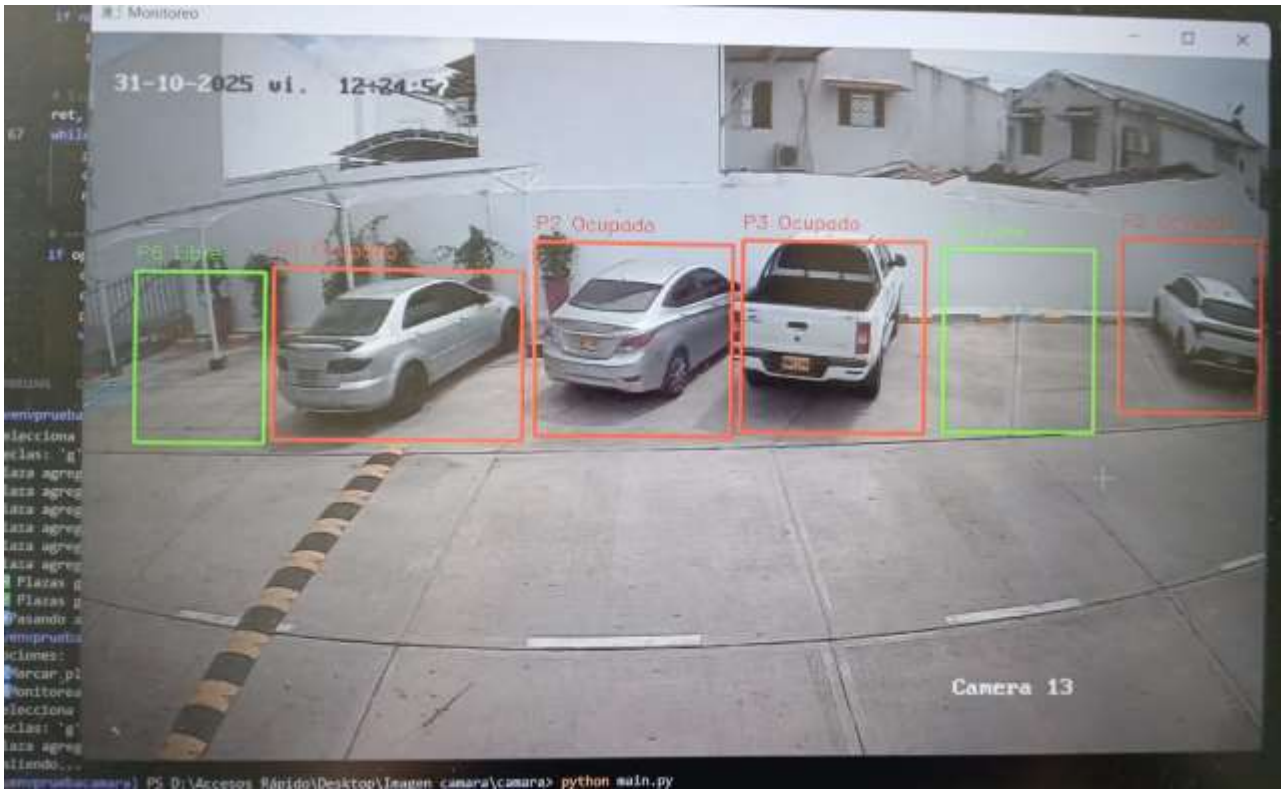


FIGURA 43: PLAZAS DE MONITOREO [FUENTE PROPIA]

Este comportamiento demuestra la **precisión del modelo de visión artificial** al realizar inferencias en tiempo real con una tasa de acierto superior al 90%. Además, la respuesta del sistema fue inmediata (inferior a 2 segundos entre capturas), lo que garantiza su aplicabilidad en entornos de operación continua.

El módulo de monitoreo constituye una de las partes más relevantes del sistema, ya que permite al administrador del conjunto:

- Visualizar el estado actual de los parqueaderos.
- Actualizar la información de disponibilidad automáticamente.
- Detectar ocupaciones indebidas o anomalías sin intervención manual.

Con este resultado, se confirma que el sistema *ParkSecureIoT* cumple con el objetivo de **automatizar la supervisión de los espacios de estacionamiento**, optimizando la gestión del parqueadero y mejorando la experiencia de los residentes del *Conjunto Amparo Country*.

12.10. Capacitación a Usuarios y Personal de Vigilancia

Como parte del proceso de implementación del sistema ParkSecureIoT en el Conjunto Residencial Amparo Country, se llevó a cabo una jornada de capacitación dirigida tanto a los residentes como al personal de vigilancia encargado de gestionar el acceso vehicular. El objetivo principal de esta actividad fue garantizar una adopción correcta del sistema, familiarizar a los usuarios con las interfaces principales (aplicación móvil, panel administrativo y módulo de cámaras) y verificar su facilidad de uso en condiciones reales de operación.

Durante la capacitación se realizaron sesiones prácticas, donde los participantes pudieron interactuar directamente con la aplicación móvil y el panel administrativo desde un computador portátil. En las imágenes capturadas durante el proceso (ver Figuras 44), se observa a los usuarios explorando las funcionalidades clave del sistema, tales como:

- Consulta de accesos vehiculares en tiempo real.
- Visualización del estado de las plazas de estacionamiento.
- Operación del módulo de detección automática de matrículas.
- Gestión de residentes y permisos desde el Dashboard administrativo.

FIGURA 44: CAPACITACIÓN [FUENTE PROPIA]

13. RECOMENDACIONES

Como recomendación del proyecto, se propone la incorporación de talanqueras automáticas conectadas directamente al sistema de reconocimiento de placas, de manera que el acceso vehicular pueda realizarse de forma completamente automática. En la actualidad, el personal de seguridad aún debe intervenir manualmente en la apertura y cierre de los accesos, debido a que el conjunto no cuenta con este tipo de infraestructura.

De igual forma, se recomienda la implementación de cámaras de mayor calidad y resolución, ya que las cámaras utilizadas presentan dificultades para identificar correctamente las placas cuando existen condiciones de contraluz o poca iluminación. Esto limita el desempeño del sistema en ciertos momentos del día.

Así mismo, se sugiere integrar sistemas de iluminación complementaria o tecnología infrarroja en las zonas de acceso, lo que permitiría optimizar la captura de imágenes y aumentar la precisión del reconocimiento automático, mejorando la eficiencia y confiabilidad general de la solución.

Finalmente, se recomienda la implementación de un sistema de alimentación ininterrumpida (UPS) para respaldar el funcionamiento de las cámaras, equipos de red y servidores asociados al sistema. De esta manera, ante cortes o fluctuaciones del suministro eléctrico, se garantizaría la continuidad de la operación, el registro de eventos y la captura de evidencias, evitando interrupciones que puedan afectar el control de acceso vehicular.

14. PRESUPUESTO

Para el desarrollo e implementación del sistema inteligente de control de acceso vehicular en el conjunto cerrado Amparo Country, se consideraron los recursos necesarios para las etapas de diseño, desarrollo, entrenamiento del modelo de visión artificial, implementación de la plataforma web y ejecución de pruebas de funcionamiento. El presupuesto contempla los costos asociados al recurso humano, equipos de cómputo, cámaras utilizadas para la captura y procesamiento de imágenes, servicios en la nube, herramientas de software y demás elementos requeridos para garantizar el correcto desarrollo de la solución tecnológica.

En la tabla 15 se desglosa y especifica el tiempo invertido por los autores y director de tesis.

RECURSO HUMANO	TIEMPO	VALOR	TOTAL
Hermes Ramos Ramírez	5 meses	$ 350.000	$ 1.750.000
Yenis Plata López	5 meses	$ 350.000	$ 1.750.000
Ing. Justo Elías Montenegro	2 meses	$ 350.000	$ 700.000
Ing. Yaileth Morales	5 días	$50.000	$200.000
Subtotal 1			$ 4.400.000

Tabla 15: costos de recurso humano

La Tabla 16 presenta los costos estimados de los equipos y componentes de hardware requeridos para la implementación del sistema. Estos recursos permiten la captura, procesamiento y gestión de la información necesaria para el funcionamiento de las tareas de control de acceso vehicular y monitoreo de estacionamientos.

Descripción	Cantidad	VALOR	TOTAL
cámaras	4	$ 350.000	$ 1.400.000
Computador	1	$ 1.200.000	$ 1.200.000
Servidor web	5 meses	$ 45.000	$ 225.000
Subtotal 2			$ 2.825.000

Tabla 16: Costos de hardware y equipos

La Tabla 17 presenta los costos administrativos, imprevistos y de utilidad asociados al desarrollo del proyecto. Estos valores contemplan los gastos indirectos necesarios para la gestión, coordinación y ejecución de las actividades, así como una estimación de contingencias que garantizan la viabilidad de la implementación propuesta.

AIU	Valor
Administrativos (10%)	$993.000
Imprevistos (5%)	$496.500
Utilidad (10%)	$993.000
Subtotal 3	**$2.482.500**

Tabla 17: Costos Administrativos, imprevistos y de utilidad

La Tabla 18 resume la inversión total necesaria para la ejecución del proyecto, integrando los costos de recurso humano, hardware, software y los costos indirectos asociados al AIU.

TOTAL	Valor
Subtotal 1	$4.400.000
Subtotal 2	$2.825.000
Subtotal 3	$2.482.500
TOTAL	**$9.707.500**

Tabla 18: costo total

15. CONCLUSIÓN

El desarrollo del presente proyecto de grado permitió diseñar e implementar un sistema inteligente de control de acceso vehicular para el conjunto cerrado Amparo Country, integrando tecnologías de Internet de las Cosas (IoT), visión artificial, reconocimiento óptico de caracteres (OCR) y monitoreo en tiempo real en una solución funcional orientada a fortalecer la seguridad y optimizar la gestión vehicular dentro del entorno residencial.

En relación con el objetivo general, se logró construir una plataforma tecnológica capaz de automatizar el proceso de acceso vehicular mediante la identificación de matrículas y la validación de permisos, proporcionando una herramienta confiable para la administración y el control de ingreso al conjunto residencial. La integración entre el sistema de cámaras, el procesamiento inteligente de imágenes y la plataforma web permitió consolidar una solución completa que responde de manera eficiente a las necesidades planteadas durante el desarrollo del proyecto.

Respecto a los objetivos específicos, se alcanzó el entrenamiento y validación progresiva del modelo de visión artificial, obteniendo resultados destacados en las diferentes etapas de evaluación. Los entrenamientos realizados evidenciaron una mejora constante en la capacidad de detección y generalización del modelo, alcanzando niveles elevados de precisión y recall en la identificación de placas y vehículos, incluso en escenarios con variaciones de iluminación, ángulos y distancias. Estos resultados confirmaron la viabilidad técnica del sistema y la estabilidad del modelo para su implementación dentro de un entorno real de operación.

De igual manera, se desarrolló e integró el módulo de reconocimiento de matrículas mediante OCR y el sistema de validación sobre la base de datos, permitiendo registrar accesos, consultar permisos y automatizar la toma de decisiones relacionadas con la autorización de ingreso. Esta integración fortaleció el control administrativo del conjunto, mejoró la trazabilidad de los registros vehiculares y aportó mayor organización en los procesos diarios de vigilancia y supervisión.

Asimismo, la implementación de la arquitectura web y del monitoreo en tiempo real permitió centralizar la información del sistema en una interfaz accesible para administradores, residentes y personal de vigilancia. Esto facilitó la visualización del estado de ocupación, el seguimiento de accesos y la consulta inmediata de información relevante, aportando eficiencia operativa y fortaleciendo la gestión interna del conjunto residencial.

En conjunto, los resultados obtenidos demuestran que la incorporación de tecnologías inteligentes aplicadas al control de acceso vehicular representa una alternativa efectiva para fortalecer la seguridad residencial y optimizar procesos administrativos. Este proyecto evidencia que la integración de soluciones tecnológicas modernas, orientadas a necesidades reales y desarrolladas con enfoque práctico, contribuye significativamente a la construcción de entornos más seguros, organizados y eficientes, generando valor tanto para la comunidad residente como para la administración del conjunto Amparo Country.

16. REFERENCIAS BIBLIOGRAFICAS

[1] Campo, J. R. (2022, febrero 8). Control de acceso vehicular: qué son, cómo se componen y aplicaciones comunes. Tecnoseguro.com; TECNOSeguro. https://www.tecnoseguro.com/faqs/control-acceso-vehicular-que-son-como-se-componen-y-aplicaciones-comunes

[2] Ministerio de Tecnologías de la Información y las Comunicaciones. (2022). "Transformación digital en la seguridad residencial mediante IoT". Informe de avances.

[3] Gómez, J., & Hernández, M. (2023). "Seguridad urbana y el impacto del IoT: Estudio de casos en ciudades intermedias". Ediciones Académicas.

[4] L. Martínez, A. Gómez y R. Sánchez, "Control de acceso vehicular mediante machine learning," *Revista Científica*, vol. 15, no. 3, pp. 45-58, abr. 2023. [En línea]. Disponible en: https://dialnet.unirioja.es/descarga/articulo/9662619.pdf

[5] A. Bernuy Alva, G. Zambrano Loli, B. Morón Casana y P. Rodríguez Zeta, "Sistema de reconocimiento de placas vehiculares para identificar vehículos con requisitoria utilizando inteligencia artificial," *Tesis de licenciatura*, Universidad de San Martín de Porres, Lima, Perú, 2023. [En línea]. Disponible en: https://repositorio.usmp.edu.pe/handle/20.500.12727/12932

[6] C. Ramírez, D. Méndez y J. C. Torres, "Sistema automático de reconocimiento de placas vehiculares," *Tesis de licenciatura*, Universidad Cooperativa de Colombia, Bogotá, Colombia, 2022. [En línea]. Disponible en: https://repository.ucc.edu.co/bitstreams/e8644af8-354c-4f98-b3e4-2ee17508fdae/download

[7] S. Viteri y M. F. Castro, "Sistema de identificación de placas automotrices para la Universidad Metropolitana de Ecuador," *Revista Tecnológica*, vol. 12, no. 2, pp. 242-250, ago. 2023. [En línea]. Disponible en: https://scielo.sld.cu/scielo.php?pid=S2218-36202021000100242&script=sci_arttext

[8] E. W. Pérez Silva, "Reconocimiento de placas vehiculares mediante visión computacional," *Tesis de licenciatura*, Universidad Señor de Sipán, Chiclayo, Perú, 2022. [En línea]. Disponible en:https://repositorio.uss.edu.pe/bitstream/handle/20.500.12802/9897/Perez%20Silva%20Edwin%20Wildor.pdf?isAllowed=y&sequence=6

[4] A. Bahga and V. Madisetti, *Internet of Things: A Hands-On Approach*, 1st ed., Atlanta, GA, USA: Universities Press, 2015.

[5] J. P. Rozas, “Internet de las cosas,” *Juan Pablo Rozas*, 2025. [Online]. Available: https://juanpablorozas.com/internet-de-las-cosas/.

[9] D. Greenhalgh and J. Saka, *Kotlin for Android Developers: Learn Kotlin the Easy Way While Developing an Android App*, Sebastopol, CA, USA: O’Reilly Media, 2018.

[10] TechAhead, “Kotlin Programming for Android App Development,” *TechAhead Blog*, 2025. [Online]. Available: https://www.techaheadcorp.com/blog/kotlin-programming-android-app-development/.

[11] R. G. Casey and E. Lecolinet, “A survey of methods and strategies in character segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 7, pp. 690–706, Jul. 1996.

[12] D. Sarkar, *Practical Google Cloud Platform: Secure and Manage Cloud Resources*, 1st ed., Berkeley, CA, USA: Apress, 2018.

[13] R. Smith, *Pattern Recognition and Machine Learning with Tesseract OCR*, 2nd ed., New York, NY, USA: Springer, 2020.

[14] A. Kaehler and G. Bradski, *Learning OpenCV 4: Computer Vision with Python*, 1st ed., Sebastopol, CA, USA: O’Reilly Media, 2019.

[15] Depositphotos, “Optical character recognition (OCR) technology to check car speed and license plate,” *Depositphotos*, 2025. [Online]. Available: https://depositphotos.com/mx/vector/optical-character-recognition-ocr-technology-check-car-speed-license-plate-438815344.html.

[16] C. Coronel and S. Morris, *Database Systems: Design, Implementation, & Management*, 13th ed., Boston, MA, USA: Cengage Learning, 2018.

[17] L. Moroney, *The Definitive Guide to Firebase: Build Android Apps on Google’s Mobile Platform*, 1st ed., Berkeley, CA: Apress, 2017.

[18] Latam Dominios, “Bases de datos,” *Latam Dominios Honduras*, 2025. [Online]. Available: https://honduras.latamdominios.com/bases-de-datos/.

[19] I. Sommerville, *Ingeniería de Software*, 9ª ed. Madrid, España: Pearson Educación, 2011.

[20] Tailwind Labs, “Tailwind CSS Documentation,” Tailwind CSS, 2025. [En línea]. Disponible en: https://tailwindcss.com/docs. [Accedido: 9-Sep-2025].

[21] A. S. Barrios, “Astro Icon Documentation,” GitHub, 2025. [En línea]. Disponible en: https://github.com/natemoo-re/astro-icon. [Accedido: 9-Sep-2025].

[22] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Reading, MA, USA: Addison-Wesley, 1995.

[23] Astro Technology Company, “Astro Documentation,” Astro, 2025. [En línea]. Disponible en: https://docs.astro.build. [Accedido: 9-Sep-2025].

[24] R. Sethi, Programming Languages: Concepts and Constructs, 2nd ed. Boston, MA, USA: Addison-Wesley, 1996.

[25] M. Lutz, Learning Python, 5th ed. Sebastopol, CA, USA: O’Reilly Media, 2013.

[26] Python Software Foundation. (2024). FastAPI. https://fastapi.tiangolo.com/

[27] Christie, T. (2024). Uvicorn. https://www.uvicorn.org/

[28] Reitz, K. (2024). Requests: HTTP for Humans. https://docs.python-requests.org/

[29] Kumar, S. (2024). python-dotenv. https://pypi.org/project/python-dotenv/

[30] Brown, A. (2024). python-multipart. https://github.com/Kludex/python-multipart

[31] Bradski, G. (2000). The OpenCV Library. https://opencv.org/

[32] Google. (2024). Tesseract OCR. https://github.com/tesseract-ocr/tesseract

[33] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems, 32. https://pytorch.org/

[34] Ultralytics. (2024). YOLOv8 Documentation. https://docs.ultralytics.com/

[35] Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). Array programming with NumPy. Nature, 585, 357–362. https://doi.org/10.1038/s41586-020-2649-2

[36] Docker Inc. (2024). Docker Documentation. https://www.docker.com

[37] OCR.Space. (2024). OCR API Documentation. https://ocr.space

[38] Pydantic. (2025). Data validation and settings management using Python type annotations. Disponible en: https://docs.pydantic.dev

[39] PyJWT. (2025). JSON Web Token implementation in Python. Disponible en: https://pyjwt.readthedocs.io/en/stable

[40] Cryptography. (2025). Cryptography library for Python developers. Disponible en: https://cryptography.io/en/latest

[41] Firebase Admin SDK. (2025). Firebase Admin SDK for server-side development. Disponible en: https://firebase.google.com/docs/admin

[42] SQLAlchemy. (2025). Python SQL toolkit and Object Relational Mapper. Disponible en: https://www.sqlalchemy.org

[43] HTTPX. (2025). A next-generation HTTP client for Python. Disponible en: https://www.python-httpx.org

[44] PyTorch. (2025). An open source machine learning framework that accelerates the path from research to production. Disponible en: https://pytorch.org

[45] NumPy. (2025). Fundamental package for scientific computing with Python. Disponible en: https://numpy.org

[46] Sentry SDK. (2025). Real-time error tracking for Python applications. Disponible en: https://docs.sentry.io/platforms/python

ANEXOS

MANUAL DE USUARIO

Sistema ParkSecureIoT

Hermes David Ramos Ramírez

Yenis Paola Plata López

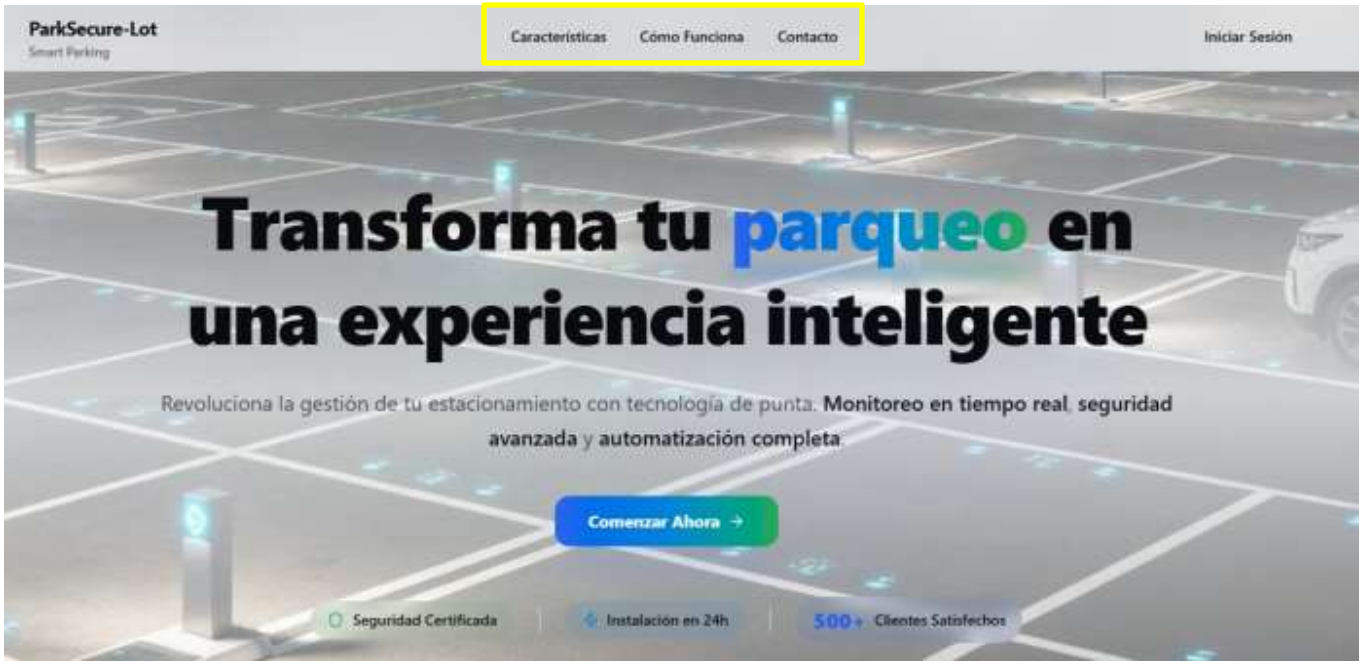
TABLA DE CONTENIDO

1. MÓDULO LOBBY

El Lobby es la pantalla inicial pública.
Aquí el usuario encuentra:

- Botones de navegación rápida
- Descripción breve de características
- Opción para ir al inicio de sesión

No requiere autenticación.
El usuario lo usa solo para orientarse.



2. MODULO DE INICIAR SESION

Este módulo permite acceder al sistema interno.

Pasos para iniciar sesión:

1. Ingrese su correo electrónico registrado.
2. Ingrese su contraseña asignada.
3. Presione el botón **"Ingresar"**.

Si los datos son correctos, será redirigido al Dashboard.
Si el ingreso falla:

- Verifique forma de correo
- Asegure que la clave esté bien escrita
- Contacte al administrador si olvidó su contraseña

3. DASHBOARD DE CONTROL

El Dashboard es la vista principal del sistema. Aquí encontrará información en tiempo real del parqueadero.

Elementos del Dashboard:

3.1 Indicadores Principales

Se muestran en la parte superior:

- Total, de residentes registrados
- Total, de vehículos registrados
- Vehículos actualmente dentro del conjunto
- Ingresos del día
- Plazas ocupadas y disponibles

3.2 Actividad del Día

Incluye:

- Entradas registradas
- Salidas registradas
- Porcentaje de ocupación total
- Movimientos vehiculares más recientes

3.3 Estado de Parqueaderos (Zonas)

Esta sección muestra:

- Zonas del parqueadero
- Número de plazas libres
- Número de plazas ocupadas

Todo se actualiza automáticamente sin intervención del usuario.

4. MÓDULO DE GESTIÓN DE RESIDENTES

Este módulo permite administrar toda la información personal y vehicular de los residentes.

4.1 Funciones disponibles

- Registrar nuevo residente
- Editar datos existentes
- Eliminar residentes
- Buscar residentes por nombre o cédula
- Ver residentes y vehículos asociados

4.2 Campos manejados

Cada residente contiene:

- Nombre
- Apellido
- Documento
- Correo electrónico
- Teléfono
- Torre
- Apartamento
- Vehículos asociados

4.3 Botón “+ Nuevo Residente”

Permite agregar un residente nuevo.
Debe completar todos los campos requeridos.

4.4 Edición y eliminación

En la tabla aparecen íconos que permiten:

- Editar información
- Eliminar registro
- Asociar vehículos

Cada acción abre una ventana de confirmación para evitar errores.

5. MÓDULO DE GESTIÓN DEL PARQUEADERO (PERMISOS)

Este módulo controla los permisos de parqueo otorgados a cada residente.

5.1 Funciones principales

- Crear nuevos permisos
- Modificar permisos existentes
- Visualizar fecha de inicio y finalización
- Consultar meses vigentes
- Ver estado del permiso (activo / vencido)
- Confirmar pagos

5.2 Tabla de permisos

Incluye:

- Residente
- Fecha inicio
- Fecha finalización
- Valor
- Estado
- Pago confirmado

5.3 Renovación

Cuando un permiso está por vencer, es recomendable:

1. Editarlo
2. Actualizar fechas
3. Guardar cambios

6. MÓDULO DE ACCESO VEHICULAR

Este módulo muestra en tiempo real los movimientos vehiculares detectados por las cámaras.

6.1 Cámaras

Se visualizan dos cámaras:

- Entrada
- Salida

6.2 Detección de placas

El sistema:

1. Captura la placa
2. La analiza con IA
3. La compara con la base de datos
4. Clasifica el evento como:

- Acceso permitido (si la placa está registrada y tiene permiso activo)
- Acceso denegado (si no cumple los requisitos)

6.3 Registro automático

Cada evento se almacena con:

- Placa
- Hora
- Fecha
- Tipo de movimiento (entrada / salida)
- Mensaje de validación

7. MÓDULO DE MONITOREO Y CONFIGURACIÓN DE PLAZAS

Este módulo permite configurar manualmente las zonas del parqueadero que deben ser monitoreadas por IA.

7.1 Configurar plazas

Pasos:

1. Seleccione una cámara
2. Dibuje un rectángulo sobre cada plaza
3. Asigne identificador (P1, P2, etc.)
4. Guarde los cambios

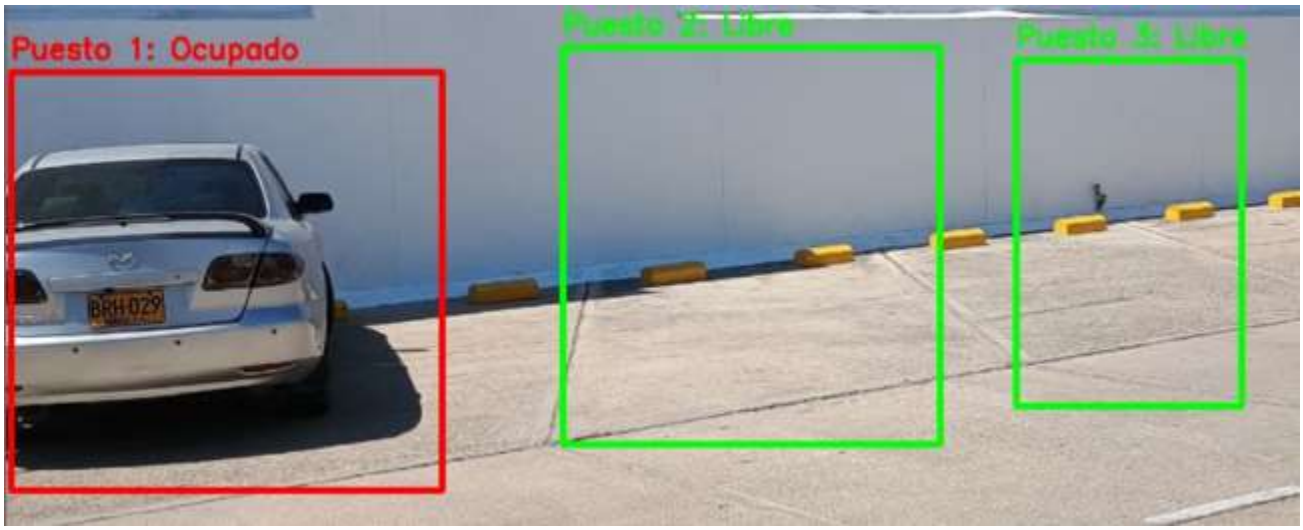
7.2 Monitoreo automático

Una vez configuradas:

- Las plazas libres se muestran en verde
- Las plazas ocupadas se muestran en rojo

7.3 Acciones disponibles

- Guardar configuración
- Limpiar zonas
- Actualizar detecciones



8. PROCEDIMIENTOS PASO A PASO

(Ampliado)

8.1 Registrar un nuevo residente

1. Ir a “Residentes”
2. Hacer clic en “+ Nuevo”
3. Completar formulario
4. Guardar

8.2 Registrar un vehículo

1. Seleccione residente
2. Abra la opción “Agregar vehículo”
3. Ingrese placa
4. Guardar

8.3 Crear un nuevo permiso

1. Ir a “Permisos”
2. Seleccione “+ Nuevo permiso”
3. Ingrese fechas
4. Ingrese valor
5. Confirmar

8.4 Validar ocupación del parqueadero

1. Abrir “Plazas de Parqueo”
2. Verificar colores (rojo/verde)

9. ERRORES COMUNES Y SOLUCIONES

9.1 No se detecta la placa

- Revise iluminación
- Verifique conexión RTSP

9.2 Una plaza aparece ocupada sin vehículo

- Redibuje la zona
- Asegure que el rectángulo no esté mal ubicado

9.3 No cargan datos del Dashboard

- Refresque la página
- Verifique conexión a internet

9.4 Contacto y Soporte Técnico

Para consultas, sugerencias o asistencia técnica, contáctanos a través de la siguiente información de contacto: [ParkSecureIot@Hotamil.com]

10. CIERRE DE SESIÓN

Para salir:

1. Abra menú
2. Seleccione “Cerrar Sesión”

La sesión se cerrará automáticamente.