Universidad Tecnológica Nacional

Proyecto Final

Sistema de control para playas de estacionamiento con reconocimiento automático de patentes.

Autores:

- *Churichi, Alan Xavier*
- *Gonzalez, Exequiel Gabriel*

Director:

- *Mg. Ing. Burgos, Sergio Enrique*

Proyecto final presentado para cumplimentar los requisitos académicos para acceder al título de Ingeniero en Electrónica

en la

Facultad Regional Paraná

Noviembre de 2021

Declaración de autoría:

Nosotros declaramos que el Proyecto Final “Sistema de control para playas de estacionamiento con reconocimiento automático de patentes” y el trabajo realizado son propio/s. Declaro/declaramos:

- Este trabajo fue realizado en su totalidad, o principalmente, para acceder al título de grado de Ingeniero en Electrónica en la Universidad Tecnológica Nacional, Regional Paraná.
- Se establece claramente que el desarrollo realizado y el informe que lo acompaña no han sido previamente utilizados para acceder a otro título de grado o pre-grado.
- Siempre que se ha utilizado trabajo de otros autores, el mismo ha sido correctamente citado. El resto del trabajo es de autoría propia.
- Se ha indicado y agradecido correctamente a todos aquellos que han colaborado con el presente trabajo.
- Cuando el trabajo forma parte de un trabajo de mayores dimensiones donde han participado otras personas, se ha indicado claramente el alcance del trabajo realizado.

Firmas:

- 
- 
-

Fecha: 03 / 11 / 2021

Agradecimientos:

Al ing. Sergio Burgos por habernos brindado su tiempo y conocimiento además de su predisposición en todo momento.

Estoy profundamente agradecido a mis padres y hermanos que me brindaron su ayuda y apoyo a lo largo del desarrollo de este proyecto. A mis amigos, compañeros de trabajo y entendidos del tema que brindaron su valioso tiempo para ayudarme, aportar con sus ideas y conocimiento.

Churichi, Alan Xavier

Me encuentro muy agradecido con mi familia, mis padres y hermanos que me brindaron su ayuda y apoyo incondicional a lo largo de toda la carrera. Agradezco a mis amigos, que siempre están cuando los necesito, a mi amigo Eric Beauchamps por ayudarnos con las pruebas de este proyecto y a mis compañeros de trabajo por brindar ideas nuevas y su experiencia en el área.

Gonzalez, Exequiel Gabriel

Universidad Tecnológica Nacional

Abstract

Facultad Regional Paraná

Ingeniero en Electrónica

Sistema de control para playas de estacionamiento con reconocimiento automático de patentes.

Churichi, Alan Xavier
Gonzalez, Exequiel Gabriel

Abstract:
It was designed a system capable of automating the entry and exit of a parking-type establishment. This allows to reduce operating times and costs, and to operate without human intervention unless a charge for the service is required.
The system has a management software which can calculate the cost of the stay and register payments, which can then be viewed on the history screen. In addition, with these records statistical graphs can be made. For this development, a microservices architecture was mainly used, in which each one fulfills a specific function. Among the technologies used are ESP8266, Mqtt, Machine Learning, Docker, and Postgres.
A scalable, modularized, economical product was obtained with a high degree of precision in the recognition of cars and trucks, and an average precision in the case of motorcycles.

Keywords: Deep Learning, ESP8266, License Plate Recognition, Microservice architecture, MQTT.

Resumen:

Se diseñó un sistema capaz de automatizar el ingreso y egreso de un establecimiento del tipo estacionamiento. Esto permite reducir tiempos y costos de operación, y funcionar sin intervención humana a menos que se requiera realizar un cobro por el servicio. El sistema cuenta con un software de gestión que tiene entre sus características la capacidad de calcular el costo de la estadía y registrar cobros, los cuales pueden ser visualizados luego en la pantalla de historial. Además, con estos registros se pueden realizar gráficos estadísticos.

Para este desarrolló se utilizó principalmente una arquitectura de microservicios, en los cuales cada uno cumple una función específica. Entre las tecnologías utilizadas se destacan ESP8266, Mqtt, Aprendizaje automático, Docker y Postgres.

Se obtuvo un producto escalable, modularizado, económico y con un elevado grado de precisión en el reconocimiento de automóviles y camionetas, y una precisión media en el caso de las motocicletas.

Palabras Clave: Aprendizaje profundo, ESP8266, Reconocimiento de patentes, Arquitectura de microservicios, MQTT.

Índice:

Capitulo 1: Introducción 1
1.1 Objetivo 2
1.2 Pruebas de Concepto 3
1.3 Pruebas de Producto 4
Capitulo 2: Desarrollo 5
2.1 Dispositivos 6
2.1.1 Cámara 7
2.1.2 Reflector Cámara 10
2.1.3 Sensor de Presencia 17
2.1.4 Semáforo 24
2.2 Análisis de microservicios. 30
2.2.1 Microservicio encargado de tomar fotografías 30
2.2.2 Backend-db 33
2.2.2.1 Base de datos 33
2.2.2.1.1 Vehicles 35
2.2.2.1.2 Entries 35
2.2.2.1.3 Payments 35
2.2.2.1.4 Passageway 36
2.2.2.1.5 Cameras 36
2.2.2.1.6 Spotlights 36
2.2.2.1.7 TrafficLights 37
2.2.2.1.8 Sensors 37
2.2.2.1.9 EntryDisplays 38
2.2.2.1.10 Costs 38
2.2.2.1.11 Alerts 39
2.2.2.1.12 PlacesCounters 39
2.2.2.1.13 Banners 40
2.2.2.1.14 Logos 40
2.2.3 Servicio de respaldo de base de datos 42
2.2.4 Servicio de detección de vehículos 42
2.2.4.1 Algoritmos de reconocimiento de tipo de vehículo y patente 42
2.2.4.1.1 Conceptos necesarios 43
2.2.4.2 Modelos utilizados 49
2.2.4.3 Reconocimiento del tipo de vehículo 49
2.2.4.4 Generación del conjunto de datos 51
2.2.4.4.1 Stanford Cars Dataset: 52
2.2.4.4.2 PASCAL VOC2012 Dataset: 52
2.2.4.4.3 Google Images: 52
2.2.4.5 Entrenamiento 53
2.2.4.6 Pruebas 54
2.2.4.7 Modelo de reconocimiento de patentes 54
2.2.4.7.1 Pruebas 59

2.2.4.7.2 Reconocimiento de texto en patentes. 59
2.2.5 Microservicio Central – MQTT 65
2.2.6 Cartel de entrada 70
2.2.7 HMI principal 72
2.3 Análisis completo 89
2.3.1 Entrada 89
2.3.2 Salida 90
2.3.3 Administración 92
2.3.4 Servidor Local 92
Capitulo 3: Resultados 96
Capitulo 4: Análisis de Costos 99
Capitulo 5: Discusión y Conclusión. 104
Capitulo 6: Trabajos citados 106

Lista de Figuras:

Figura 1: Diagrama simplificado del sistema. 5
Figura 2: Conexión de dispositivos con el servidor central. 6
Figura 3: Cámara utilizada para el reconocimiento de imágenes. 8
Figura 4: Fotografía de la cámara donde se pueden apreciar sus conexiones. 9
Figura 5: Reflector de 10 Watts de potencia. 10
Figura 6: Circuito esquemático del dispositivo reflector. 11
Figura 7: ESP8266 WeMos D1 Mini. 12
Figura 8: Fuente Step Down 220V AC a 5V DC. 13
Figura 9: Diagrama de flujo de la lógica de funcionamiento del dispositivo reflector. 14
Figura 10: Web server del reflector. 15
Figura 11: Imagen frontal del dispositivo reflector de prueba. 16
Figura 12: Imagen lateral del dispositivo reflector de prueba. 17
Figura 13: Dispositivo reflector por dentro. 17
Figura 14: Sensor de ultrasonido HC-SR04. 18
Figura 15: Circuito esquemático del Sensor de Presencia. 19
Figura 16: Diagrama de flujo de la lógica de funcionamiento del Sensor de Presencia. 20
Figura 17: Imagen frontal del Sensor de Presencia. 21
Figura 18: Sensor de Presencia por dentro. 22
Figura 19: Web server del sensor de proximidad. 23
Figura 20: Focos tipo gota de 2 Watts de potencia. 24
Figura 21: Módulo Relé. 25
Figura 22: Circuito esquemático del semáforo. 26
Figura 23: Web server del semáforo. 27
Figura 24: Diagrama de flujo de la lógica de funcionamiento del semáforo. 28
Figura 25: Semáforo utilizado durante las pruebas. 29
Figura 26: Parte interna del semáforo utilizado durante las pruebas. 30
Figura 27: Esquema Microservicio para toma de fotografías. 31
Figura 28: Esquema de la base de datos. 34
Figura 29: Neurona elemental. 43
Figura 30: Diagrama general de una red neuronal. 44
Figura 31: Proceso de convolución para matrices. Esta imagen fue tomada de [17]. 46
Figura 32: Detección de bordes verticales. 47
Figura 33: Detección de bordes horizontales. 47
Figura 34: Desenfoque de imagen. 48
Figura 35: Diagrama de una red neuronal convolucional genérica. 48
Figura 36: Imagen tomada con la cámara de prueba. 55
Figura 37: Imagen recortada del área de la patente. 55
Figura 38: Formatos de patentes para automóviles (izquierda) y motocicletas (derecha). Esta imagen fue tomada de [27]. 56
Figura 39: Detección de ubicación de patente en un automóvil. 57
Figura 40: Detección de ubicación de patente en una motocicleta. 57
Figura 41: Puntos de interés devueltos por el modelo. 58
Figura 42: Preprocesado de patente. 61
Figura 43: Diagrama de flujo de la lógica de funcionamiento del microservicio de detección de vehículos. 63
Figura 44: Esquema Microservicio central – parte 1. 67
Figura 45: Esquema Microservicio central – parte 2. 68
Figura 46: Esquema Microservicio central – parte 3. 69
Figura 47: Cartel de entrada cuando hay lugares disponibles en el estacionamiento. 70

Figura 48: Cartel de entrada cuando no hay lugares disponibles en el estacionamiento. 71
Figura 49: Pantalla: principal. 72
Figura 50: Filtrando tabla de ingresos con barra de búsqueda. 73
Figura 51: Tabla con los ingresos activos del establecimiento. 73
Figura 52: Fotografía de un ingreso exitoso. 75
Figura 53: Pantalla: Historial. 76
Figura 54: Reconociendo una camioneta de noche. 76
Figura 55: Reconociendo una motocicleta. 77
Figura 56: Pantalla: Gráficos. 77
Figura 57: Gráfico de recaudación, por día. 78
Figura 58: Gráfico de ocupación por día. 78
Figura 59: Gráfico de distribución de métodos de pagos por día. 78
Figura 60: Formatos para exportar gráfico. 79
Figura 61: Información de un gráfico como CSV. 79
Figura 62: Alerta: Error. 80
Figura 63: Alerta: Advertencia. 80
Figura 64: Alerta: Información. 81
Figura 65: ¿Esta seguro que desea eliminar un ingreso? 82
Figura 66: Confirmación de que una entrada fue eliminada correctamente. 83
Figura 67: Alerta de que un vehículo está listo para retirarse del establecimiento. 84
Figura 68: Menú de cobro. 85
Figura 69: Pantalla principal, desde un celular. 86
Figura 70: Fotografía de una de las entradas, desde un celular. 87
Figura 71: Vista de la pantalla principal, desde un celular inclinado. 88
Figura 72: Diagrama del sistema completo. 89
Figura 73: Diagrama del sistema de entrada. 90
Figura 74: Diagrama del sistema de salida. 91
Figura 75: Diagrama del sistema de administración. 92
Figura 76: Diagrama del servidor. 93
Figura 77: Fotografía tomada de día 97
Figura 78: Fotografía tomada de noche. 97
Figura 79: Distribución de las horas trabajadas. 101
Figura 80: Gráfico de ganancia, ingresos y costos totales según cantidad de ventas. 102

Lista de Tablas

Tabla 1: Esquema de la tabla “Vehicles”.....35
Tabla 2: Esquema de la tabla “Entries”.....35
Tabla 3: Esquema de la tabla “Payments”.....35
Tabla 4: Esquema de la tabla “Passageway”.....36
Tabla 5: Esquema de la tabla “Cameras”.....36
Tabla 6: Esquema de la tabla “Spotlights”.....37
Tabla 7: Esquema de la tabla “TrafficLights”.....37
Tabla 8: Esquema de la tabla “Sensors”.....38
Tabla 9: Esquema de la tabla “EntryDisplays”.....38
Tabla 10: Esquema de la tabla “Costs”.....39
Tabla 11: Esquema de la tabla “Alerts”.....39
Tabla 12: Esquema de la tabla “PlacesCounters”.....40
Tabla 13: Esquema de la tabla “Banners”.....40
Tabla 14: Esquema de la tabla “Logos”.....40
Tabla 15: Arquitectura del modelo VGG16 con capas personalizadas.....51
Tabla 16: Resultados del entrenamiento del modelo de detección de tipo de vehículo.....54
Tabla 17: Matriz de Confusión para el modelo de detección de tipo de vehículo.....54
Tabla 18: Primeras y últimas capas del modelo de redes neuronales convolucionales para detección de texto en patente.....60
Tabla 19: Protocolos de comunicación entre microservicios.....94
Tabla 20: Costos fijos.....100
Tabla 21: Costos variables.....100
Tabla 22: Resumen de costos.....101
Tabla 23: Análisis de amortización.....102

Lista de Abreviaciones y Símbolos

DB: Base de datos.

MQTT: Messaging Queue Telemetry Transport.

HTTP: Hypertext Transfer Protocol

REST: representational state transfer.

SSID: Service Set Identifier.

Trig: Trigger.

SQL: Structured Query Language.

HMI: Human Machine Interface.

OCR: Optical character recognition.

API: Application programming interface.

ID: Identificador único.

Dedicado a:

Todas las personas que nos ayudaron y confiaron en nosotros a lo largo de estos años, especialmente nuestras familias.

Capitulo 1: Introducción

En las grandes ciudades, municipios y regiones con un crecimiento poblacional exponencial, el factor común son sus limitaciones cuando deben responder de manera efectiva a la enorme demanda de estacionamiento de vehículos de transporte privado. A diferencia de otras décadas, la cantidad de vehículos está aumentando en cualquier parte del mundo. Las zonas urbanas se ven cada vez más afectadas por la falta de espacio para construir viviendas. Esto requiere que los estacionamientos tradicionales estén preparados no solo en su capacidad, sino también en una eficiente organización y gestión del tiempo. Reduciendo los tiempos de ingreso y egreso a las playas de estacionamiento, así como también asegurando la disponibilidad de lugar en el establecimiento previo al arribo del posible cliente, resultarían en una reducción la polución producida por los vehículos, generando así un impacto positivo en el medioambiente.

En un estacionamiento automatizado el personal requerido se reduce en comparación a uno clásico. Esto, podría implicar una reducción en los costos de operación, resultando así en una merma del coste por hora de la plaza de estacionamiento, hecho que sin duda pudiera beneficiar tanto a clientes como a propietarios de estacionamientos, ya que costos más baratos suponen un fuerte aumento en la demanda de plazas.

Por otra parte, la existencia de una aplicación de Administración trae consigo una variada gama de ventajas para las partes intervinientes en este sistema. Esta permitiría almacenar registros del tiempo de permanencia de los vehículos, horarios de ingreso y egreso, costos y un registro de clientes. Con todos estos datos se podrá realizar un análisis estadístico sobre el establecimiento que permita tomar decisiones objetivas para mejorar el servicio, rentabilidad y atención al cliente.

Se plantea la automatización de un estacionamiento, para que este sea completamente funcional con intervención humana mínima. Con las ventajas de tener un control preciso, costos más baratos y la posibilidad de funcionamiento 24hs.

El sistema cuenta con un conjunto de cámaras conectadas a un servidor, el cual realiza reconocimiento de patentes vehículos en tiempo real, con el fin de

detectar su número de matrícula y habilitar la entrada al estacionamiento. A su vez el sistema mantendrá un registro sobre el tiempo de permanencia de cada vehículo con el objetivo de calcular el monto a cobrar según el tiempo de permanencia (en caso de ser necesario). Además, estos registros permitirán conocer en tiempo real la ocupación del establecimiento y generar estadísticas de interés para la administración.

El procesamiento de la información de las cámaras y los diferentes sensores se realizará en un servidor local.

Finalmente, eliminando la opción de cobro, el sistema sería aplicable como sistema de ingresos-egresos automatizado en alguna organización pública o privada.

A continuación, se describe en detalle el paso a paso del ingreso, estadía y egreso de un vehículo utilizando el sistema automatizado.

1. En la entrada, mediante una pantalla se muestra si existen lugares disponibles.
2. En caso afirmativo, una cámara detecta la patente del vehículo y habilita el paso poniendo un semáforo en verde (podría reemplazarse por una barrera).
3. El conductor ingresa y elige una plaza disponible.
4. Cuando el cliente se retira, se detecta la patente en la salida, mediante una cámara, y se finaliza la facturación.
5. En caso de ser necesario, un empleado realizará el cobro en función del tiempo de permanencia en el establecimiento.
6. Se habilita la salida mediante un semáforo (o se abre la barrera) y el vehículo puede retirarse del establecimiento.

1.1 Objetivo

Como público objetivo se tiene a los propietarios de estacionamientos, ya sean playas de estacionamiento de tipo tradicional, donde se cobra la estadía, o estacionamientos de organizaciones públicas o privadas.
En una primera instancia se planea ingresar en el mercado local de la ciudad de Paraná, el cual cuenta con aproximadamente 60 estacionamientos convencionales.

Actualmente solo existen dos estacionamientos en la ciudad que poseen algún tipo de sistema semiautomático, aunque sus prestaciones son menores al producto que se propone. La idea podría ser adoptada por todos los demás estacionamientos tradicionales.

De acuerdo con los estudios realizados en la ciudad de Paraná, a la mayoría de posibles compradores les interesaría el sistema. Sin embargo, muchos no cuentan con el capital suficiente para realizar una inversión en este tipo de implementación.
Múltiples dueños de estacionamientos sugieren que su forma actual de administrar su establecimiento tiene como problema el congestionamiento de clientes y autos en horarios pico, y por lo tanto estarían interesados en alternativas que solventaran esta problemática.

Si bien varios propietarios de estacionamientos conocían la existencia de productos similares, desconocían el costo debido a que no es común la implementación de estos en la ciudad. Por este motivo los precios sugeridos rondan entre los $40000 y $100000. Estos precios no serían suficientes para financiar este proyecto, por lo tanto, se deberá trabajar en la parte de marketing para dar a conocer mejor las bondades del producto, y poder así alcanzar un precio que sea adecuado para ambas partes.

En general, los estacionamientos con menor cantidad de plazas de aparcamiento optarían por prescindir de muchas de las características que planteamos, dada su menor solvencia económica.

1.2 Pruebas de Concepto

El desarrollo tiene posibilidad de éxito dado que es innovador en la región, por lo tanto, favorecería a los estacionamientos, generando una ventaja competitiva respecto a los demás en los horarios pico y/o brindando una atención las 24hs. Sin mencionar la reducción de costos debido a la menor cantidad de trabajadores necesarios, factor sumamente importante para dueños de estos establecimientos. Debido a esto se supone que el producto tendrá una alta probabilidad de éxito en la región.

El cliente optará por el producto ofrecido sobre el de la competencia debido a que las alternativas de automatización para estacionamiento no son capaces de brindar todas las ventajas antes mencionadas.

El producto cumplirá con las expectativas del cliente siempre y cuando el funcionamiento sea correcto y presente un bajo porcentaje de fallas, en lo que a reconocimiento de patentes respecta.

Se tiene un amplio abanico de posibles clientes, como podrían ser estacionamientos tradicionales, empresas, supermercados, barrios privados, etc.

1.3 Pruebas de Producto

De acuerdo con investigaciones realizadas, se considera que la idea a desarrollar es buena, pero tal vez no se pueda explotar su máximo potencial en la zona, ya que hay un desconocimiento general del segmento, e incluso el costo podría ser elevado para estacionamientos pequeños. Debido a esto, el producto ofrecido podría no ser la mejor opción para este tipo de estacionamiento, quizá les sería conveniente optar por un sistema más simple y que requiera una inversión menor. No obstante, para estacionamientos grandes que cuenten con recursos suficientes, o aquellos casos en los que se permita el acceso solamente a vehículos específicos, por ejemplo, empresas o barrios privados, la solución sugerida permitirá una automatización completa.

Dentro de la región no existe una competencia directa que ofrezca un producto equivalente. Respecto a la competencia en el país, existen empresas que comercializan productos similares, pero ninguna ofrece un producto tan completo.

En Uruguay existe una empresa que comercializa una solución muy similar a la sugerida en este trabajo con un costo aproximado deUS$9000.

Capitulo 2: Desarrollo

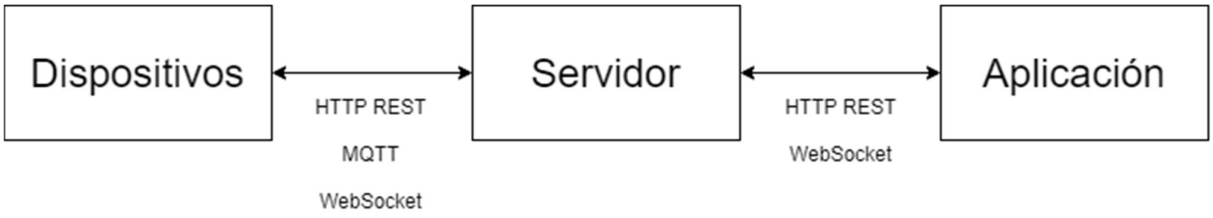


Figura 1: Diagrama simplificado del sistema.

Se denomina dispositivos a todo equipo de hardware encargado de recolectar y/o transmitir información al servidor. Dentro de ellos se encuentran las cámaras, reflectores, luces de entrada/salida, cartel de entrada, sensores de proximidad.

Estos dispositivos están compuestos por un equipo de hardware y software diseñado especialmente para la función que deberán desempeñar.

El Servidor es el equipo de cómputo encargado de la lógica de funcionamiento de la solución. Este tiene como objetivo brindar, procesar y almacenar los recursos obtenidos de los distintos dispositivos para luego poder mostrar al usuario de la aplicación la información pertinente.

En el servidor se utiliza una arquitectura de microservicios, es decir, pequeñas aplicaciones encargadas de tareas específicas que se comunican entre sí para lograr un funcionamiento completo y robusto del sistema.

A su vez se hace uso de la plataforma Docker [1] para dar soporte a los diferentes microservicios. Cada uno es montado en un container de Docker independiente y es capaz de comunicarse con los demás microservicios en una misma red virtual creada por Docker.

La comunicación entre los dispositivos y el servidor se lleva a cabo mediante diferentes protocolos, estos son: HTTP [2], MQTT [3] y WebSocket [4]. Con el objetivo de lograr una comunicación optima y fluida se realizó un estudio detallado para determinar que protocolo utilizar en cada paso del proceso de comunicación dispositivo-servidor.

Es necesario distinguir una tercera capa, la de aplicación. En esta capa se encuentra la página web a la que el administrador del estacionamiento tendrá acceso y mediante la cual podrá consultar datos y estadísticas, revisar y

administrar la configuración del sistema, o agregar entradas manualmente si así lo desease.

Cabe destacar que el microservicio de la aplicación en realidad se encuentra alojado en el servidor y un usuario con conexión a la misma red en la que se encuentra el servidor podrá acceder a ella mediante un navegador web. Sin embargo, se justifica diferenciar el servicio de la Aplicación como una capa independiente ya que es a la que el usuario final puede acceder y con la que podrá interactuar.

2.1 Dispositivos

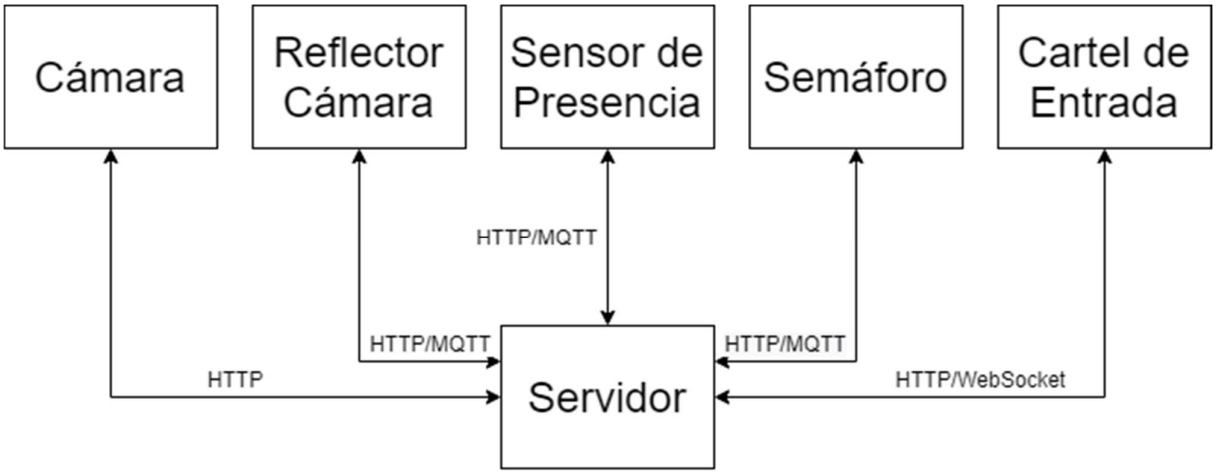


Figura 2: Conexión de dispositivos con el servidor central.

Los dispositivos son equipos de hardware con un papel fundamental en el sistema ya que son los encargados de obtener información (cámaras, sensores de presencia), o transmitirla (semáforo, cartel de entrada).

Se puede decir que cada dispositivo está compuesto por un conjunto de piezas fundamentales. La placa o circuito impreso donde se encuentran los componentes electrónicos o conexiones necesarias para componentes externos, el software o firmware de control para el dispositivo, y la caja plástica diseñada para brindar facilidad de conexión, robustez, protección y un estilo único al dispositivo.

Como se detallará luego, cada dispositivo es capaz de conectarse a una red Wifi con el objetivo de enviar o recibir datos de otros dispositivos o del servidor.

Los protocolos de comunicación a utilizar varían dependiendo del dispositivo y del objetivo de la comunicación.

En caso de que se desee consultar o guardar información en la base de datos del servidor se utiliza el protocolo HTTP que permite una manipulación y control adecuados de los datos.

Por otro lado, si el objetivo es enviar mensajes para informar estados o desencadenar acciones se utiliza el protocolo MQTT, el cual demuestra desempeñarse mejor en este tipo de tareas [5].

Finalmente, si es necesario realizar o comunicar cambios en tiempo real desde el servidor a los microservicios, se utiliza una conexión por WebSocket. Se ha comprobado que esta tecnología presenta una alta eficiencia para transmisión de datos en tiempo real [6].

A continuación, se analizarán las características de cada uno de los dispositivos diseñados y utilizados en la etapa de pruebas del proyecto.

2.1.1 Cámara

Se necesita de cámaras para lograr que el sistema sea capaz de reconocer los diferentes tipos de vehículos y sus patentes.

En un principio se planteó utilizar una cámara que tenga integrada la funcionalidad de detectar patentes. Sin embargo, el costo de un equipo con estas características es 20 veces más elevado que uno sin dicha funcionalidad. Debido a esto se decidió optar por una cámara de vigilancia estándar y realizar el procesamiento de la imagen en el servidor. Esto no supone solamente una disminución en el costo de desarrollo proyecto, sino que también le aporta un mayor grado de flexibilidad, ya que no es imperativo el uso de un modelo de cámara en particular.

Gracias a esto se podría utilizar cualquier cámara capaz de conectarse a la red y que permita obtener imágenes por medio del protocolo HTTP.

En un principio se sospechaba que una cámara de vigilancia convencional de bajo coste podría no ser suficiente para lograr un reconocimiento de patentes y clasificación de vehículos efectiva. Debido a esto se decidió investigar la existencia de antecedentes de casos exitosos de reconocimiento con imágenes de baja calidad. Los estudios “Automatic Number Plate Recognition in Low Quality Videos” [7] y “An Efficient Approach for Automatic Number Plate

Recognition for Low Resolution Images" [8] aseveran haber logrado una efectividad de reconocimiento de 90 y 83%, respectivamente, con imágenes de baja calidad.

Mediante la realización de un preprocesamiento meticuloso de las imágenes y utilizando algoritmos de aprendizaje automático en el proceso de detección es posible obtener buenos resultados sin la necesidad de tener hardware especializado.

Al momento de realizar las pruebas se utilizó la siguiente cámara de la marca ZS Vedio:

Figura 3: Cámara utilizada para el reconocimiento de imágenes.

Figura 4: Fotografía de la cámara donde se pueden apreciar sus conexiones.

Algunas de sus características son:

- Resolución Full HD 1920x1080px
- Sensor de lente 1/3" 2,0 MP CMOS
- Conexión TCP/IP HTTP
- Conexión Ethernet RJ45
- Comunicación por protocolo ONVIF
- Detección de movimiento
- Visión nocturna por infrarrojo
- Protección IP66
- Alimentación DC12V 2A
- Temperatura de funcionamiento -20℃ ~ +55℃

Se decidió hacer uso de este modelo debido a sus características, principalmente su resolución, tipo de lente, conexión IP y capacidad de visión nocturna. Además, en un principio se planteó la posibilidad usar la

característica de detección de movimiento con la que cuenta la cámara para detectar la presencia de vehículos. Lamentablemente esta característica fue diseñada para ser utilizada exclusivamente con el software propietario del fabricante. Por este motivo y debido a que el fabricante no proporciona demasiada información sobre su producto, el uso de esta funcionalidad quedó finalmente descartada.

2.1.2 Reflector Cámara

Se utiliza un reflector junto a la cámara para lograr una mejor iluminación de la patente a procesar.

Tras repetidas pruebas se determinó la necesidad de iluminar la patente para que el algoritmo de detección la pueda reconocer con mayor precisión.

Para realizar un dispositivo sencillo de realizar se decidió utilizar un reflector de 10W, debido a su potencia, precio, tamaño y disponibilidad en el mercado, además de que la mayoría de los modelos comerciales presentan una protección contra el polvo y el agua (certificación IP65).

Figura 5: Reflector de 10 Watts de potencia.

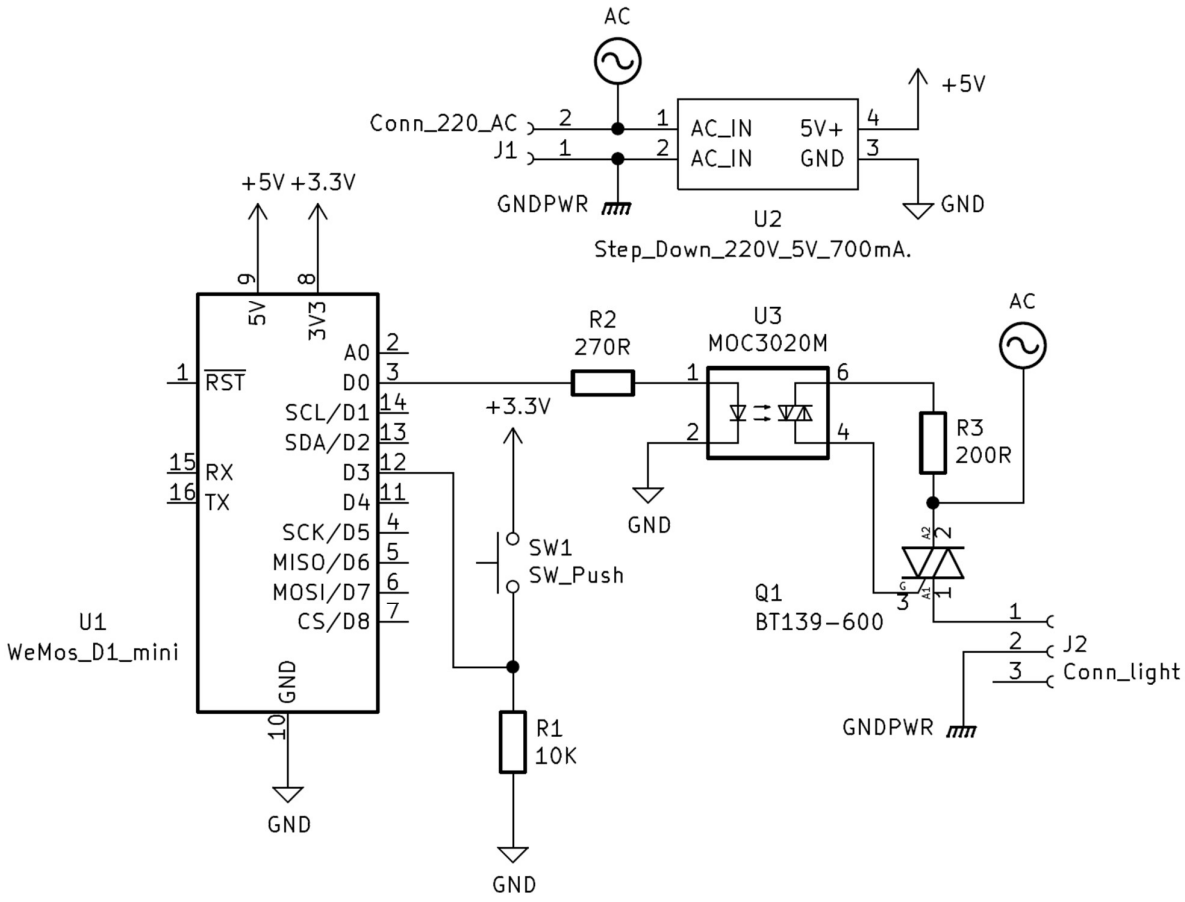


Figura 6: Circuito esquemático del dispositivo reflector.

El reflector se encuentra conectado a una etapa de control de potencia compuesta por un Triac, concretamente el modelo BT139 (elegido por su precio y disponibilidad en el mercado), el cual es capaz de soportar la tensión y corriente requerida por el reflector.

Se utilizó el microcontrolador ESP8266 para realizar la lógica de control y brindar al dispositivo la comunicación con el servidor. Se optó por este microcontrolador por su precio, disponibilidad en el mercado y capacidad de conexión inalámbrica Wifi. Durante las pruebas hizo uso de la placa de desarrollo WeMos D1 Mini que integra un ESP8266.

Además, este microcontrolador ofrece la posibilidad de ser utilizado como Web Server, para de esta forma poder realizar una configuración inicial de SSID, contraseña y nombre del dispositivo.

Para elegir en qué modo iniciar el dispositivo se añadió un botón en la placa, si este está presionado al momento del encendido, el dispositivo se inicia en

modo Access Point y espera la conexión de un usuario a su red Wifi para ser configurado, de lo contrario inicia en modo de funcionamiento normal.

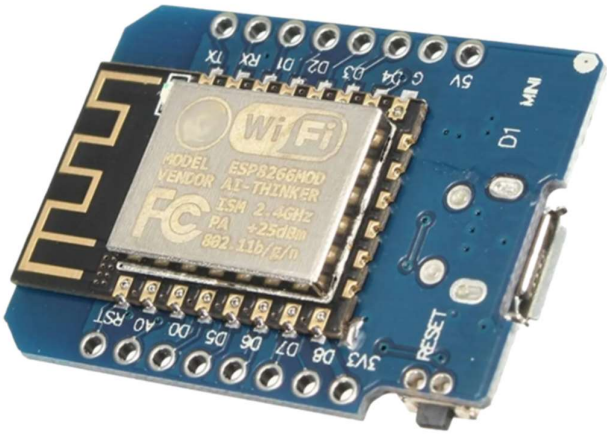


Figura 7: ESP8266 WeMos D1 Mini.

Como fuente de alimentación para el dispositivo se decidió utilizar un módulo pre armado debido a su amplia disponibilidad, bajo coste y la ventaja que estos brindan en cuanto a la reducción de tiempos de desarrollo del producto en general.
En este caso se hizo uso de una fuente Step Down 220V AC a 5V DC 700mA, que es capaz de suministrar los valores de tensión y corriente requeridos por el dispositivo.

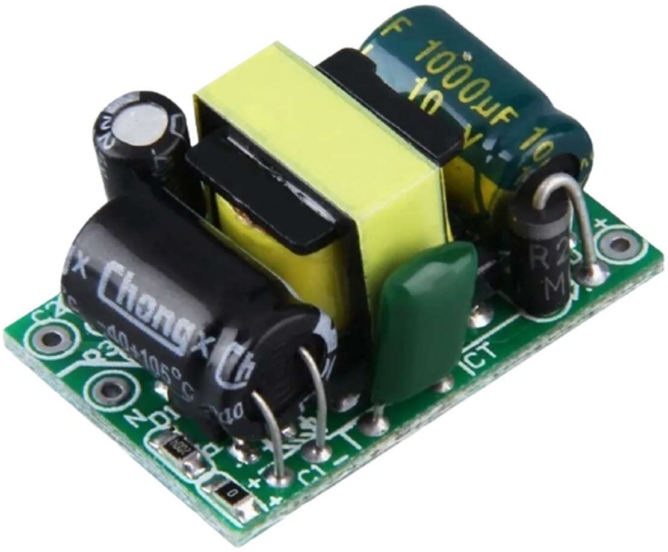


Figura 8: Fuente Step Down 220V AC a 5V DC.

El firmware desarrollado es parte fundamental del dispositivo, ya que, sin este el equipo carecería de la lógica necesaria para cumplir su función. A continuación, se puede apreciar un diagrama de flujo que intenta demostrar, de manera sencilla, el funcionamiento del equipo.

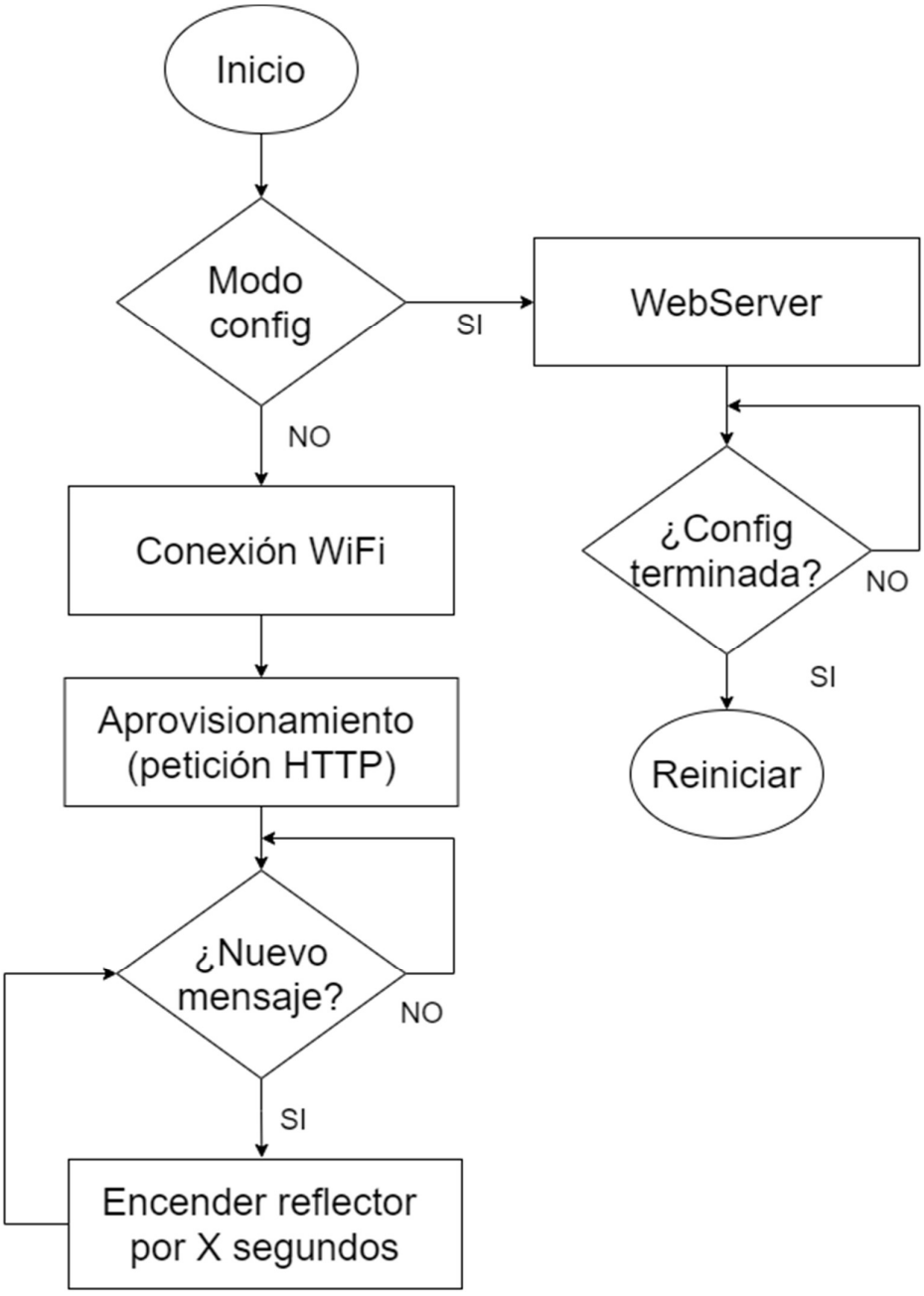


Figura 9: Diagrama de flujo de la lógica de funcionamiento del dispositivo reflector.

El funcionamiento es el siguiente:

Al encenderse el dispositivo se chequea si el pulsador de la placa está presionado, en caso afirmativo se inicia el microcontrolador como Web Server y expone una red Wifi, para poder ser configurado. Una vez hecha la configuración el equipo se reinicia automáticamente.

El web server mediante el cual se configura el dispositivo es el siguiente:

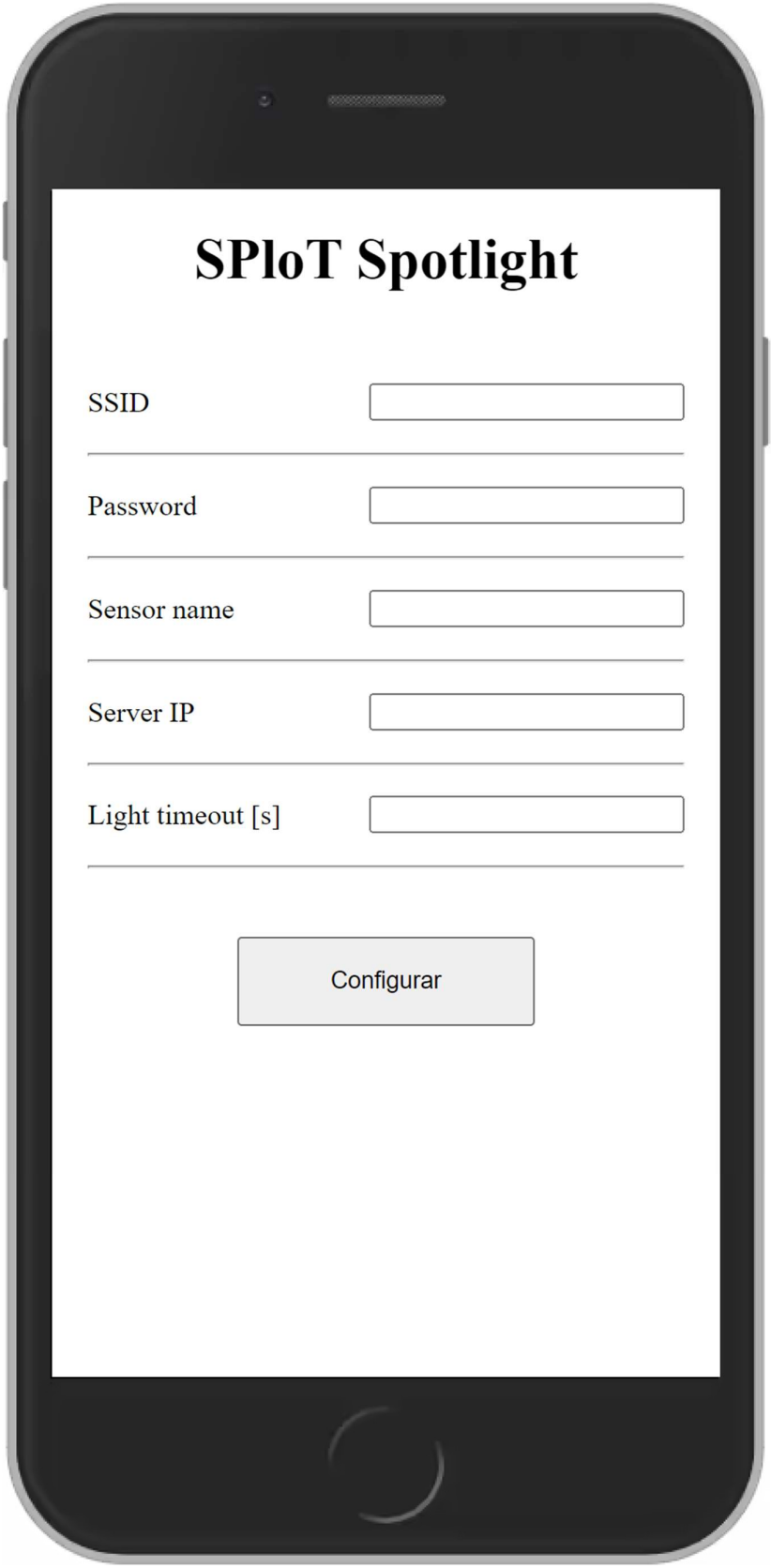


Figura 10: Web server del reflector.

Si por el contrario al encender el equipo no se encuentra presionado el botón, este se inicia el modo de funcionamiento por defecto. El dispositivo se conecta a la red Wifi con las credenciales previamente configuradas y hace una petición a uno de los microservicios alojados en el servidor, enviando el nombre con el cual se ha configurado el equipo.

El servidor responde con el id registrado para ese nombre en la base de datos. Una vez terminado el proceso de inicialización, el equipo utiliza su id para conectarse a un tópico de MQTT y queda a la espera de nuevos mensajes. Si en algún momento recibe un mensaje con valor “1”, enciende el reflector por la cantidad de segundos determinada en su configuración.

Luego de apagado el reflector el dispositivo queda la espera de nuevos mensajes para repetir el ciclo.

Figura 11: Imagen frontal del dispositivo reflector de prueba.

Figura 12: Imagen lateral del dispositivo reflector de prueba.

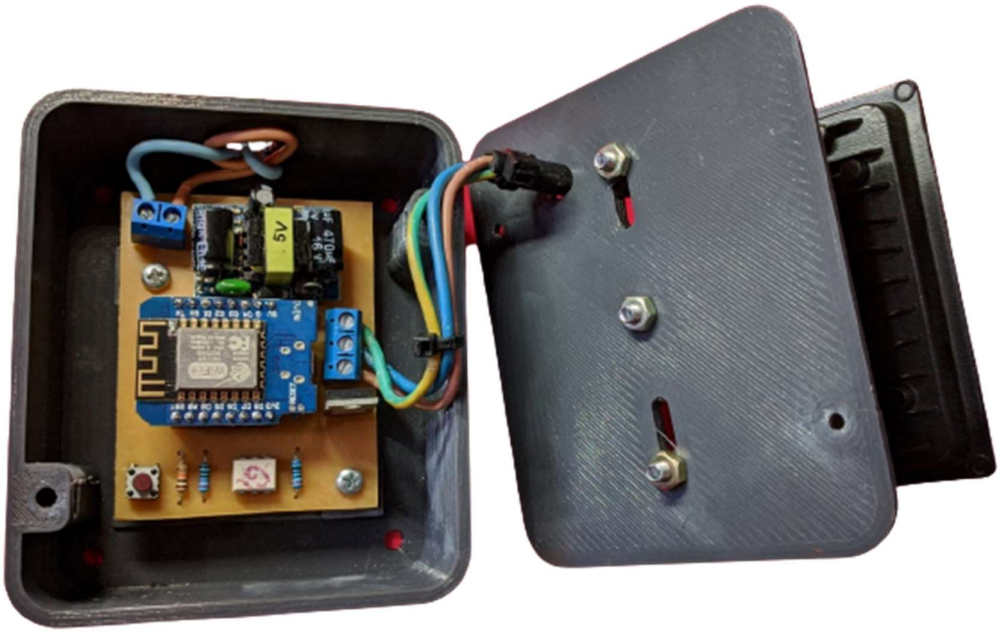


Figura 13: Dispositivo reflector por dentro.

2.1.3 Sensor de Presencia

Es necesario el uso sensores de presencia (medición de distancia) para poder indicar al sistema cuando se encuentra un vehículo en una entrada o salida. De esta manera no es necesario procesar constantemente las imágenes provenientes de la cámara para detectar la presencia de un vehículo o no. Esto

se traduce en una menor carga para el servidor y ganancia en la performance del sistema en su conjunto.

Se utiliza el sensor HC-SR04 por su precio y disponibilidad en el mercado. Este es capaz de medir distancias de hasta 4 metros con un gran nivel de precisión. Su principio de funcionamiento es sencillo, se debe aplicar un pulso de 10 microsegundos en el pin “Trig”, esto hace que el sensor emita un pulso de ultrasonido que rebota al encontrarse con una superficie y es registrado por el receptor ubicado en mismo sensor, al suceder esto el sensor emite un pulso en su pin “Echo”. Midiendo el tiempo transcurrido entre los pulsos “Trig” y “Echo” es posible determinar la distancia del objeto próximo al sensor. Para tener una mayor fiabilidad se decidió utilizar dos de estos sensores por dispositivo.

Figura 14: Sensor de ultrasonido HC-SR04.

El microcontrolador utilizado es el ESP8266, el cual puede ser iniciado como Web Server para configurar el dispositivo (de modo similar al funcionamiento del módulo reflector).

Se decidió optar por una fuente Step Down 220V AC a 5V DC 700mA.

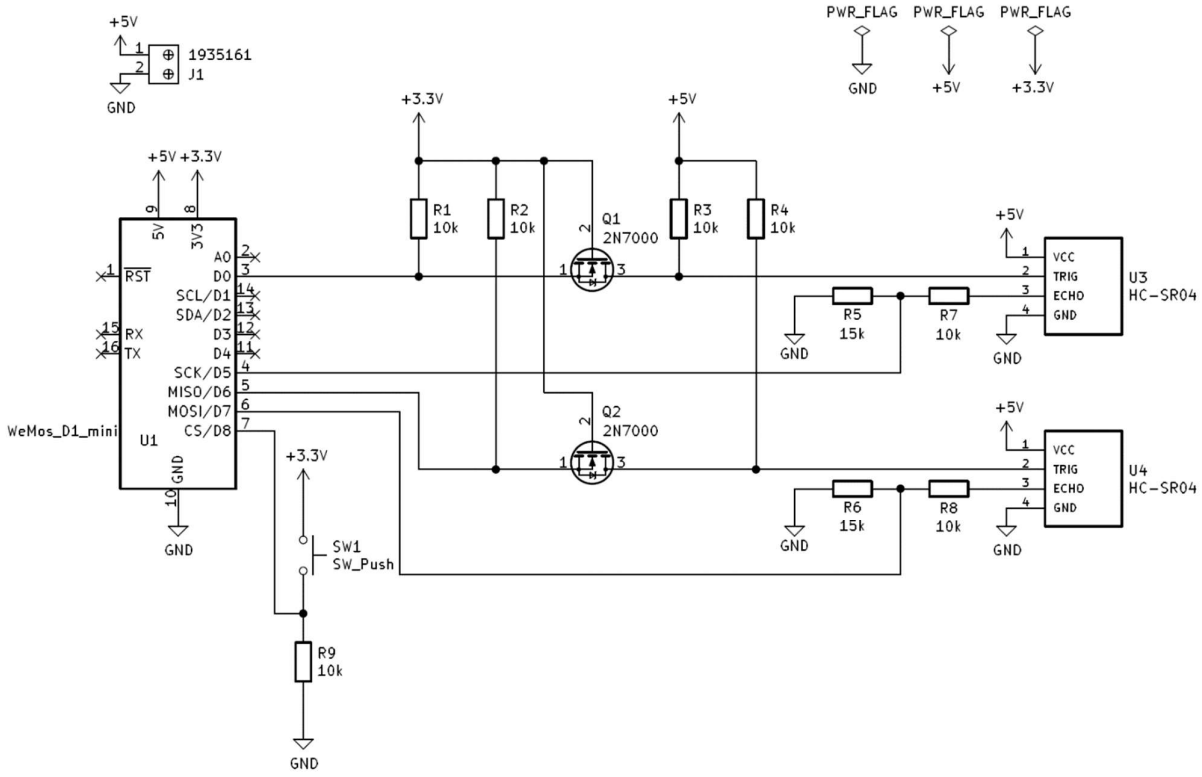


Figura 15: Circuito esquemático del Sensor de Presencia.

La comunicación entre el microcontrolador y el sensor no es directa, ya que el primero opera con una tensión lógica de 3.3V mientras que el segundo utiliza un voltaje de 5V. Es necesario adaptar los voltajes para los pines de “Trig” y “Echo”. En el caso del primero se utiliza un transistor con tecnología MOSFET, para adaptar los niveles lógicos [9]. Se utiliza el modelo 2N7000, debido a sus características y velocidad de respuesta.

Para el pin “Echo” solo hace falta utilizar un divisor resistivo para obtener 3.3V ya que la comunicación no es bidireccional.

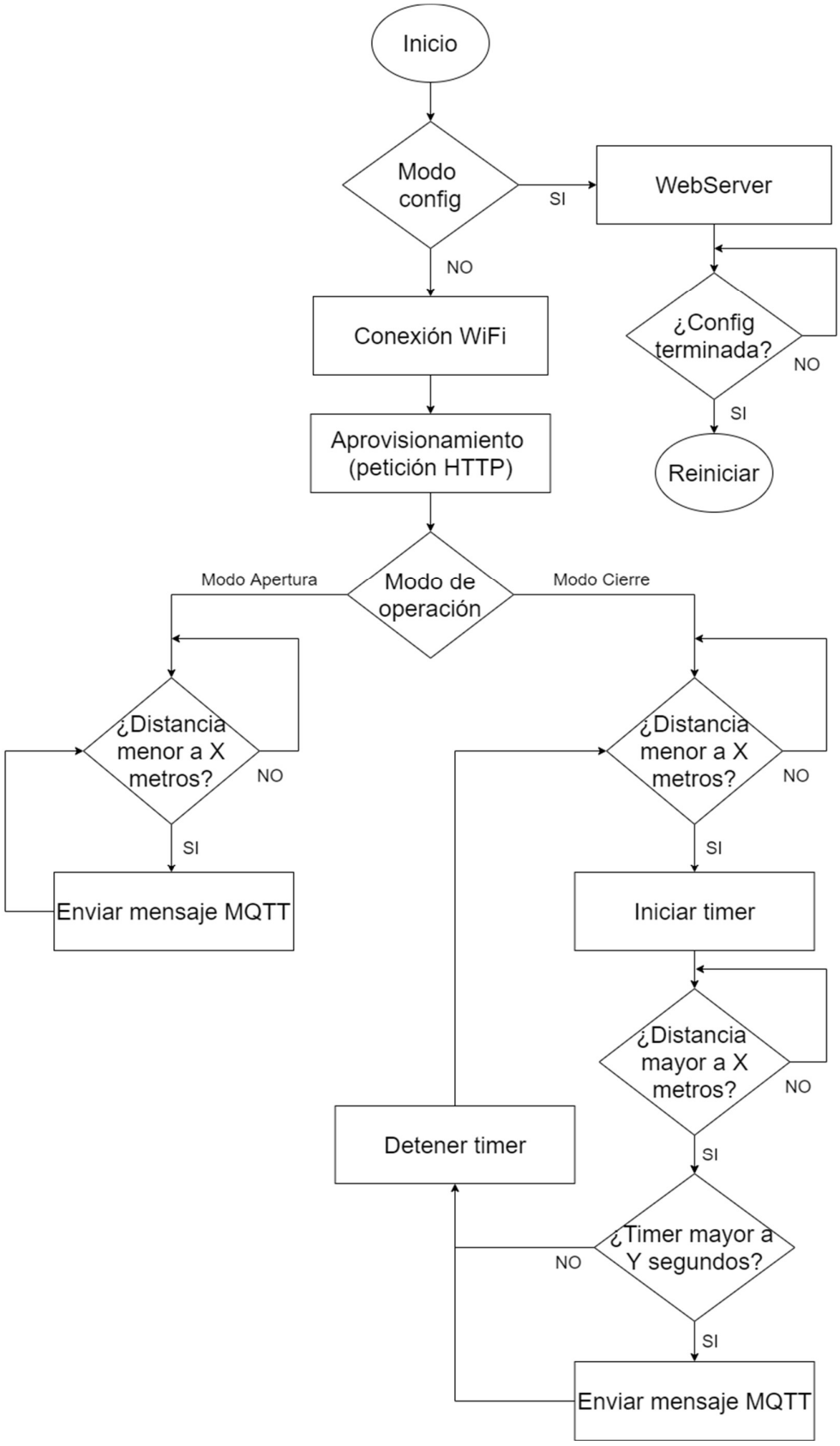


Figura 16: Diagrama de flujo de la lógica de funcionamiento del Sensor de Presencia.

Este dispositivo posee dos modos de funcionamiento, en ambos se deberá leer los sensores HC-SR04 para detectar la presencia de vehículos. Si el equipo está configurado en modo de “Apertura”, al detectar un vehículo (es decir, se mide una distancia menor al valor establecido en la configuración) se envía un mensaje por MQTT para desencadenar el evento de toma de fotografía y procesamiento de la imagen.

Un dispositivo configurado en modo de “Cierre” indica que este deberá detectar cuando un vehículo a ingresado o salido del estacionamiento (según se encuentre en una entrada o salida). En este caso el sensor enviará un mensaje por MQTT al semáforo para apagar la luz verde y encender la luz roja, indicando que está prohibido el paso.

En modo “Cierre”, cuando se detecta un vehículo se inicia un contador con el fin de conocer el tiempo transcurrido hasta que el sensor se libere. Una vez dado este acontecimiento se comprueba si el contador ha estado activo por una cantidad de tiempo mayor a la establecida por los parámetros de configuración. De ser así se envía un mensaje por MQTT hacía el semáforo correspondiente.

Es necesario llevar cuenta del tiempo transcurrido para poder evitar “falsos disparos”.

Figura 17: Imagen frontal del Sensor de Presencia.

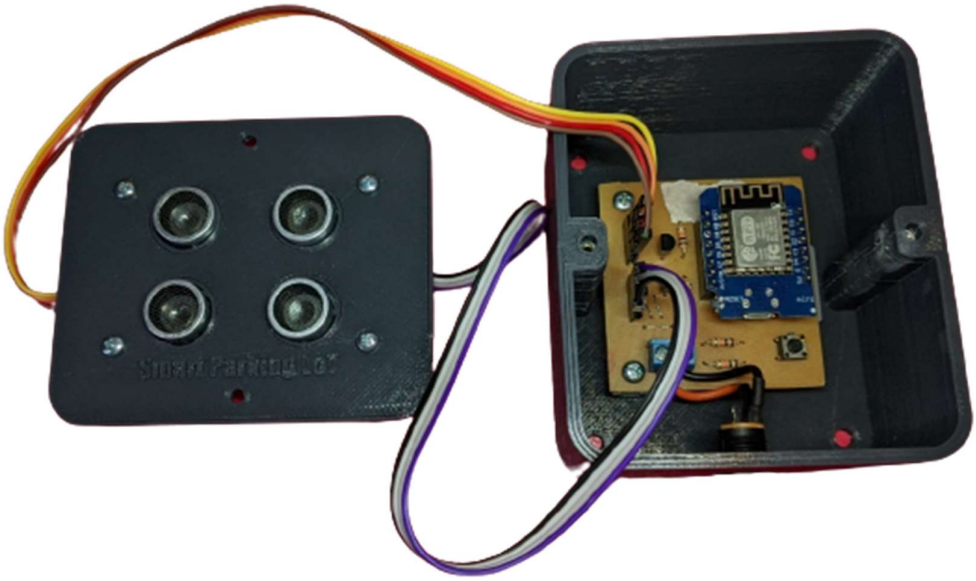


Figura 18: Sensor de Presencia por dentro.

El web server mediante el cual se configura este dispositivo es el siguiente:

Figura 19: Web server del sensor de proximidad.

2.1.4 Semáforo

Con el objetivo de indicar a los vehículos que se detengan o avancen en las entradas o salidas se decidió diseñar un semáforo. El semáforo cuenta con dos luces, una roja, indicando que no se debe avanzar y una luz verde para indicar que está permitido el paso.

Para el diseño de este dispositivo se utilizan focos led tipo gota de 2W en cada módulo. Estos son controlados por un relé, cuyo conector normal cerrado (NC) se encuentra conectado al foco rojo y el conector normal abierto (NA) se encuentra conectado al foco verde. Esto se decidió así ya que la luz roja será la que generalmente se encuentre encendida. Al hacerlo de esta forma el consumo de energía será menor ya que no se debe estar energizando constantemente la bobina del relé.

Figura 20: Focos tipo gota de 2 Watts de potencia.

Para mayor sencillez a la hora de realizar las pruebas se decidió utilizar un módulo relé opto acoplado. Gracias a esto, en caso de tener que rediseñar el circuito, se puede reutilizar el módulo y abaratar los costos de desarrollo. Para las pruebas se utilizaron relés capaces de manejar corrientes de hasta 10A, sin embargo, en un entorno producción se podrían elegir relés de menor corriente, ya que los focos leds requieren unos niveles corriente mucho menores.

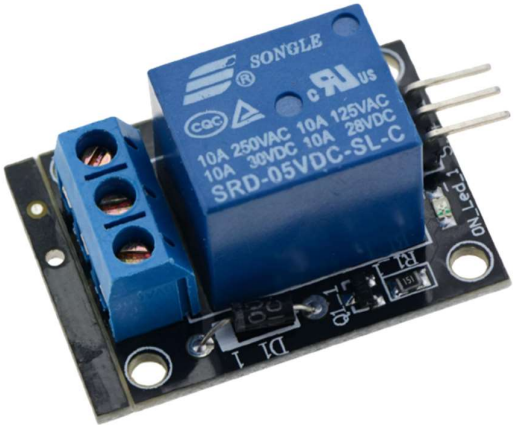


Figura 21: Módulo Relé.

Como microcontrolador se utiliza el ESP8266 y como alimentación una fuente Step Down de 5V 700mA.

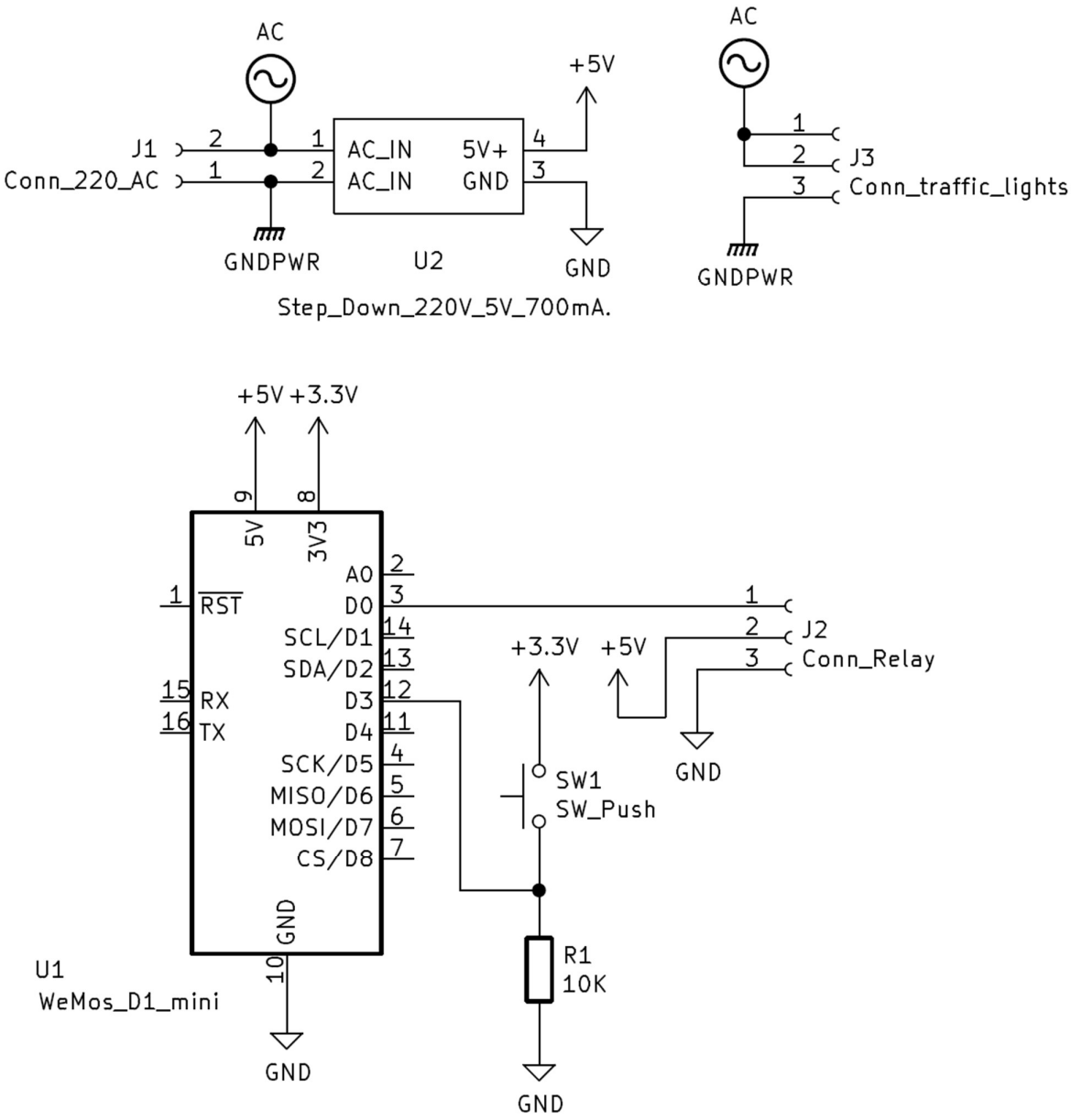


Figura 22: Circuito esquemático del semáforo.

El esquema es muy sencillo, uno de los pines digitales del ESP8266 es conectado a la entrada del módulo del relé para controlar su activación/desactivación.
Además, se realiza la conexión de un pulsador a otro pin digital del microcontrolador para poder inicializar el dispositivo en modo de Web Server en caso de ser necesario.

El web server mediante el cual se configura es el siguiente:

Figura 23: Web server del semáforo.

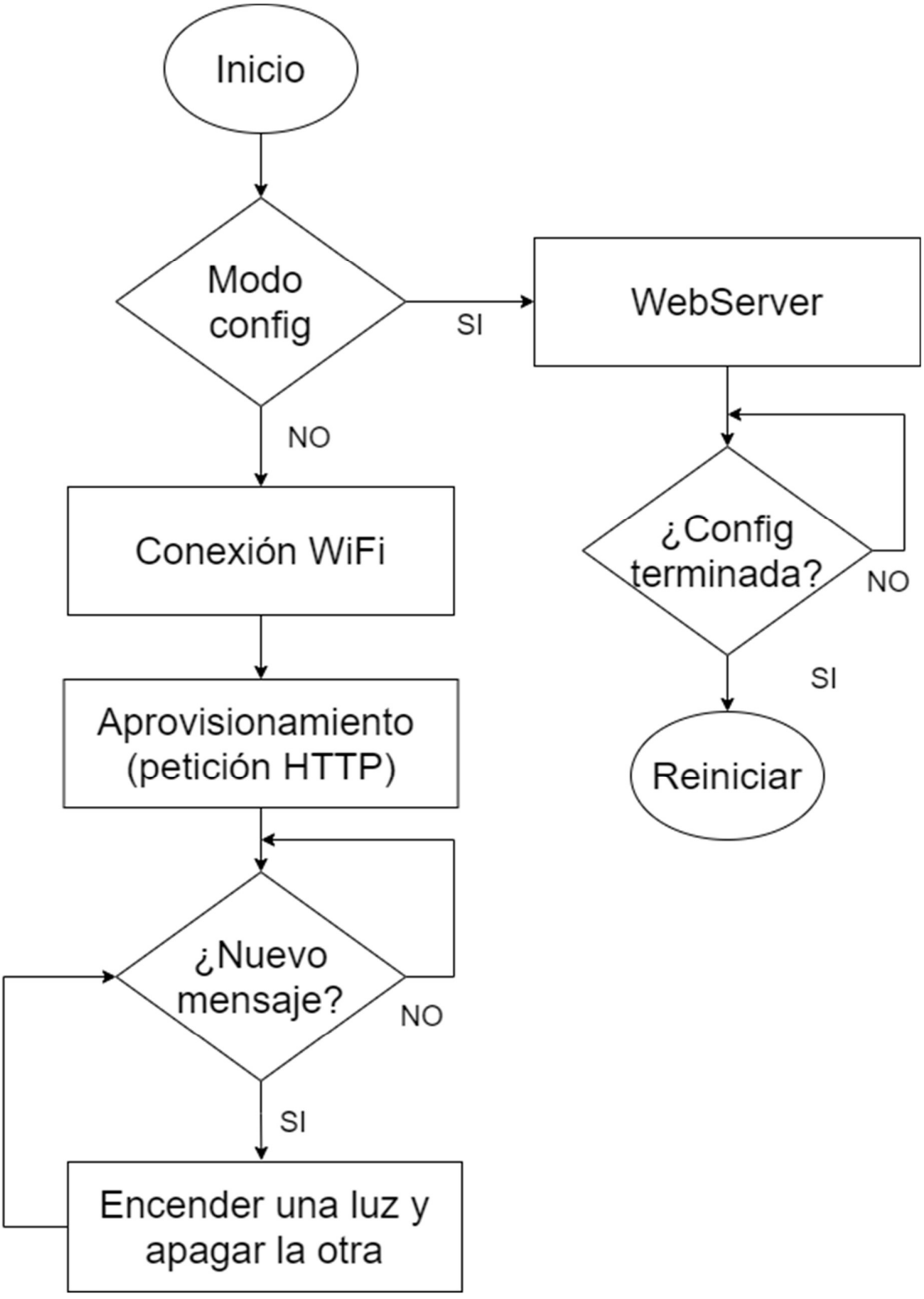


Figura 24: Diagrama de flujo de la lógica de funcionamiento del semáforo.

La lógica de funcionamiento del equipo es similar al del Dispositivo Reflector, con la excepción de que en lugar de encender un reflector por un tiempo determinado se activa o desactiva un relé. Esto ocasionará que se encienda la luz verde (si se activa) o la luz roja (si se desactiva).

Una vez recibido un mensaje el equipo permanecerá con el foco correspondiente encendido hasta registrar un nuevo mensaje indicando lo contrario.

Figura 25: Semáforo utilizado durante las pruebas.

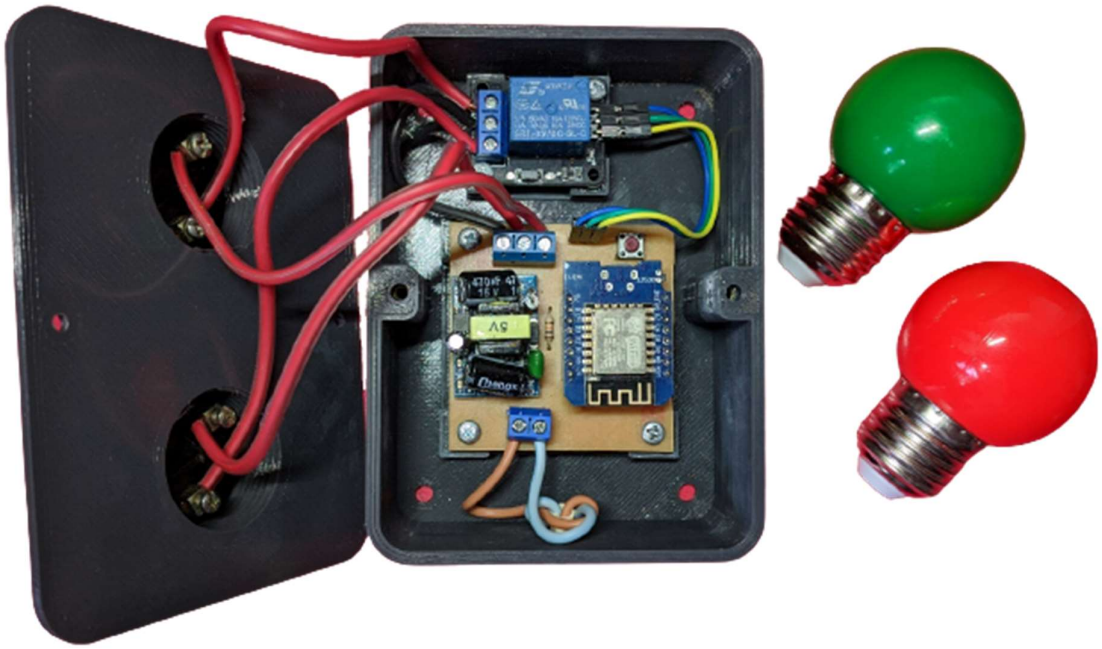


Figura 26: Parte interna del semáforo utilizado durante las pruebas.

2.2 Análisis de microservicios.

2.2.1 Microservicio encargado de tomar fotografías.

Este microservicio cumple un rol importante en el sistema, ya que es el encargado de realizar las peticiones HTTP a la cámara para obtener una fotografía. Para lo anterior es importante mencionar que la cámara debe ser compatible con este protocolo.

Utilizando el lenguaje Node Js[10], se diseñó este programa con dos objetivos en mente, que sea lo suficientemente especifico en su función, y genérico por si era necesario cambiar las cámaras. Ambos objetivos se abordaron en conjunto, resultando en un código compacto, escalable a N número de cámaras y contemplando la modularización del código para la conexión con la cámara. De este modo, en caso de utilizarse una cámara de un fabricante diferente, el sistema se podría adaptar con facilidad.

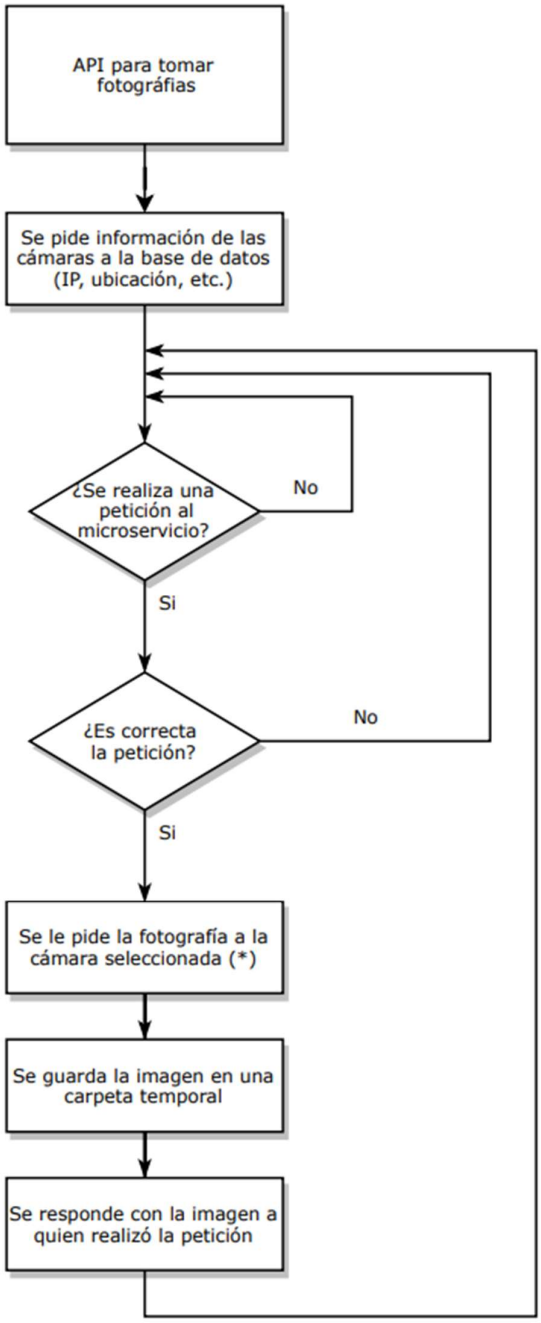


Figura 27: Esquema Microservicio para toma de fotografías.

El funcionamiento de este microservicio es el siguiente: En primer lugar, este microservicio realiza una petición a la base de datos con el objetivo de obtener la información de las cámaras conectadas al sistema. La información solicitada es la IP cada cámara, su ubicación en el establecimiento, y el nombre que esta

tiene asignado. Una vez que se dispone de la información anterior, el microservicio que en espera de alguna petición.

Cuando se realiza una petición al mismo, en primera instancia, se verifica que esta sea correcta y que incluya con que cámara de las disponibles se debe tomar la fotografía. Si es incorrecta, se descarta y se queda a la espera de una nueva. Si es correcta, se le solicita a la cámara correspondiente una fotografía. Esta es la única que debería modificarse en caso de realizar un cambio de cámaras. Una vez recibida la imagen, se la guarda en una carpeta temporal para su posterior utilización. Finalmente, la imagen es enviada de forma codificada a quien la solicitara y el microservicio queda a la espera de una nueva petición.

2.2.2 Backend-db

2.2.2.1 Base de datos

Es necesario persistir la información de los vehículos, entradas, cobros, costos, etc. Debido a esto se utilizará una base de datos, en este caso de optó por una base de datos SQL sobre una NoSQL [10], ya que la organización presenta cierta estructura concreta. Se utiliza el manejador de base de datos PostgreSQL (o simplemente Postgres) [11] por su seguridad, escalabilidad y alto grado de compatibilidad con el resto de las tecnologías a utilizar en el proyecto.

El esquema de base de datos es el siguiente:

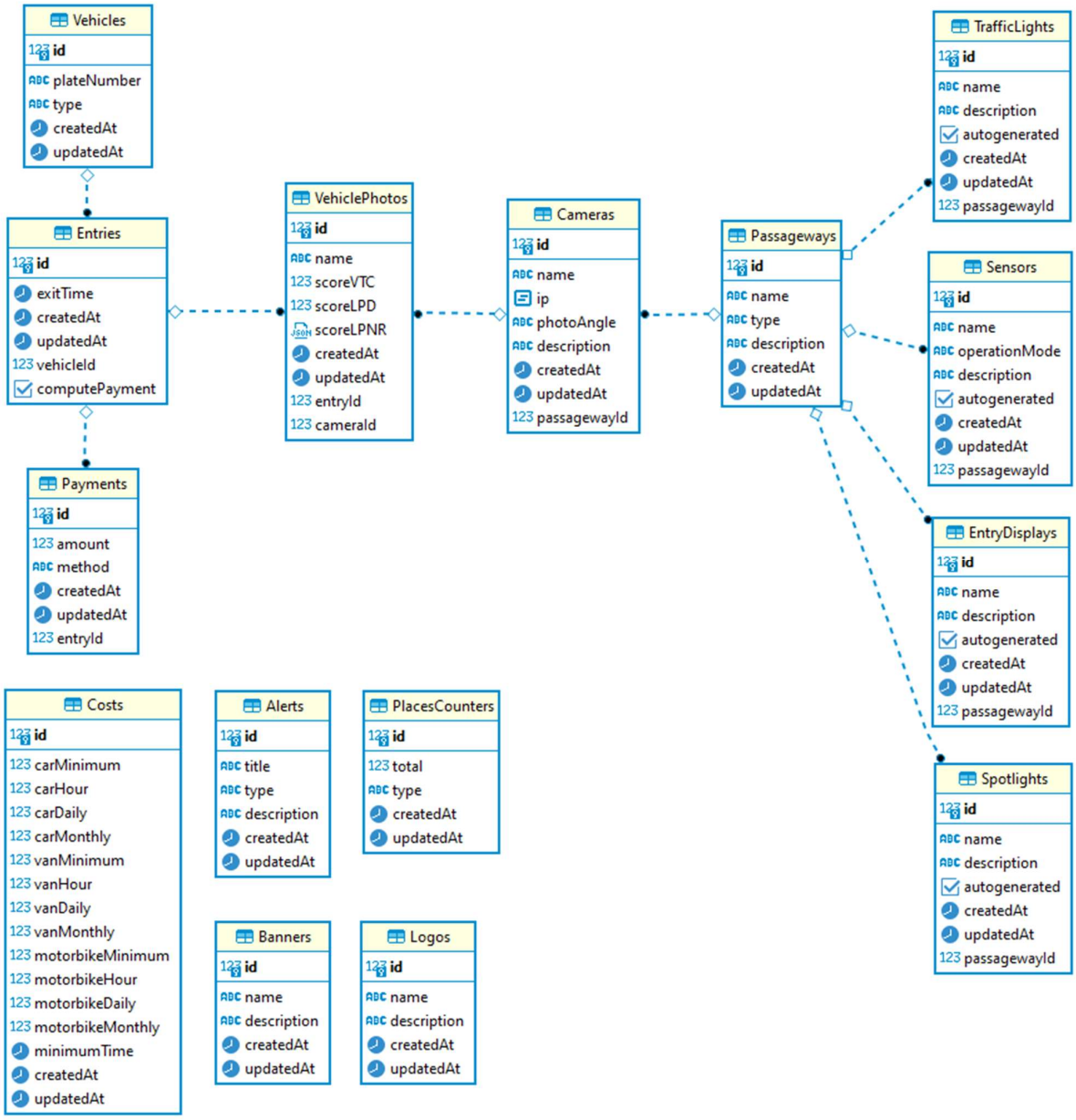


Figura 28: Esquema de la base de datos.

A continuación, se describe brevemente el objetivo de cada una de las tablas en el sistema.

El campo id corresponde a un valor entero que servirá como identificador único para cada elemento de una tabla. Los campos createdAt y updatedAt son de tipo Timestamp y representan, respectivamente, la fecha de creación y última modificación del elemento.

Ya que estas columnas son comunes a todas las tablas, serán omitidas en el siguiente análisis.

2.2.2.1.1 Vehicles

Tiene la función de registrar los vehículos que ingresan al estacionamiento, almacenando su número de patente y tipo. Solo deberá existir un elemento en la tabla por vehículo.

Tabla	Columna	Tipo de dato	Descripción
Vehicles	plateNumber	String	Número de patente
	type	Enum (Car, Van, Motorbike)	Tipo de vehículo

Tabla 1: Esquema de la tabla "Vehicles".

2.2.2.1.2 Entries

Se creará una entrada cada vez que se de acceso a un vehículo al estacionamiento. Cada entrada deberá estar asignada a un vehículo.

Tabla	Columna	Tipo de dato	Descripción
Entries	vehicleId	Integer (Foreign Key)	Relación con la tabla "Vehicles"
	exitTime	Timestamp	Fecha de salida
	computePayment	Boolean	Valor booleano para identificar si se debe procesar el pago

Tabla 2: Esquema de la tabla "Entries".

2.2.2.1.3 Payments

Al retirarse un vehículo se registra un pago, el cual se guarda en esta tabla. Debe especificarse la entrada a la cual corresponde el pago, el monto a cobrar y el método de pago utilizado.

Tabla	Columna	Tipo de dato	Descripción
Payments	entryId	Integer (Foreign Key)	Relación con la tabla "Entries"
	amount	Double	Monto a cobrar
	method	Enum (Cash, Credit Card, Debit Card)	Método de pago utilizado

Tabla 3: Esquema de la tabla "Payments".

2.2.2.1.4 Passageway

Es la entidad que nuclea a todos los dispositivos que se encontraran en una entrada o salida. Gracias a esto es fácil identificar que grupo de dispositivos (cámara, ultrasonidos, semáforo, etc.) se encuentran en un mismo punto de ingreso o egreso.

El nombre asignado debe ser único.

Tabla	Columna	Tipo de dato	Descripción
Passageways	name	String	Nombre
	type	Enum (Entry, Exit)	Determina si es una entrada o salida
	description	String	Descripción opcional

Tabla 4: Esquema de la tabla "Passageway".

2.2.2.1.5 Cameras

Representación de las cámaras en la base de datos. Se almacena información correspondiente a cada cámara en entradas y salidas.

El nombre y la ip deben ser únicos.

Tabla	Columna	Tipo de dato	Descripción
Cameras	passagewayId	Integer (Foreign Key)	Relación con la tabla "Passageways"
	name	String	Nombre
	ip	Inet	Ip de la cámara
	photoAngle	Enum (Front, Rear)	Ángulo desde el que se tomarán las fotografías
	description	String	Descripción opcional

Tabla 5: Esquema de la tabla "Cameras".

2.2.2.1.6 Spotlights

Representación de los reflectores en la base de datos. Se almacena información correspondiente a cada reflector que pudiese encontrarse en entradas o salidas.

El nombre asignado debe ser único.

Tabla	Columna	Tipo de dato	Descripción
Spotlights	passagewayId	Integer (Foreign Key)	Relación con la tabla "Passageways"
	name	String	Nombre
	autogenerated	Boolean	Permite saber si el dispositivo se registró en la base de datos automáticamente
	description	String	Descripción opcional

Tabla 6: Esquema de la tabla "Spotlights".

2.2.2.1.7 TrafficLights

Representación del semáforo en la base de datos. El objetivo de esta tabla es almacenar la información correspondiente a los semáforos del estacionamiento.

El nombre asignado debe ser único.

Tabla	Columna	Tipo de dato	Descripción
TrafficLights	passagewayId	Integer (Foreign Key)	Relación con la tabla "Passageways"
	name	String	Nombre
	autogenerated	Boolean	Permite saber si el dispositivo se registró en la base de datos automáticamente
	description	String	Descripción opcional

Tabla 7: Esquema de la tabla "TrafficLights".

2.2.2.1.8 Sensors

Representación de los dispositivos de ultrasonido en la base de datos. Se almacena información correspondiente a los sensores del estacionamiento ubicados en entradas y salidas.

El nombre asignado debe ser único.

Tabla	Columna	Tipo de dato	Descripción
Sensors	passagewayId	Integer (Foreign Key)	Relación con la tabla "Passageways"
	name	String	Nombre

	operationMode	Enum (Open, Close)	Permite identificar en qué modo deberá funcionar el sensor
	autogenerated	Boolean	Permite saber si el dispositivo se registró en la base de datos automáticamente
	description	String	Descripción opcional

Tabla 8: Esquema de la tabla "Sensors".

2.2.2.1.9 EntryDisplays

Representación de los carteles de entrada en la base de datos. Se almacena información correspondiente a cada cartel de entrada en el estacionamiento.

El nombre asignado debe ser único.

Tabla	Columna	Tipo de dato	Descripción
EntryDisplays	passagewayId	Integer (Foreign Key)	Relación con la tabla "Passageways"
	name	String	Nombre
	autogenerated	Boolean	Permite saber si el dispositivo se registró en la base de datos automáticamente
	description	String	Descripción opcional

Tabla 9: Esquema de la tabla "EntryDisplays".

2.2.2.1.10 Costs

Tabla donde se almacenan las tarifas de mínimas además de los costos por hora, día y mes para los diferentes tipos de vehículos.

Tabla	Columna	Tipo de dato	Descripción
Costs	carMinimum	Double	Costo mínimo para automóviles
	carHour	Double	Costo por hora para automóviles
	carDaily	Double	Costo diario para automóviles
	carMonthly	Double	Costo mensual para automóviles

	vanMinimum	Double	Costo mínimo para camionetas
	vanHour	Double	Costo por hora para camionetas
	vanDaily	Double	Costo diario para camionetas
	vanMonthly	Double	Costo mensual para camionetas
	motorbikeMinimum	Double	Costo mínimo para motocicletas
	motorbikeHour	Double	Costo por hora para motocicletas
	motorbikeDaily	Double	Costo diario para motocicletas
	motorbikeMonthly	Double	Costo mensual para motocicletas
	minimumTime	Timestamp	Tiempo a partir del cual se empieza a calcular el costo

Tabla 10: Esquema de la tabla "Costs".

2.2.2.1.11 Alerts

Tabla para guardar los distintos tipos de alerta que surgen durante la ejecución del sistema.

Tabla	Columna	Tipo de dato	Descripción
Alerts	title	String	Título de la alerta
	type	Enum (Error, Warning, Info)	Tipo de alerta
	description	String	Descripción opcional

Tabla 11: Esquema de la tabla "Alerts".

2.2.2.1.12 PlacesCounters

Se almacena la cantidad de plazas de estacionamiento existentes en el establecimiento.

Tabla	Columna	Tipo de dato	Descripción
PlacesCounters	total	Integer	Cantidad total de lugares en el estacionamiento

	type	Enum (CarAndVan, Motorbike)	Tipo de lugares

Tabla 12: Esquema de la tabla "PlacesCounters".

2.2.2.1.13 Banners

Tabla para almacenar banners personalizados.

Tabla	Columna	Tipo de dato	Descripción
Banners	name	String	Nombre de la imagen
	description	String	Descripción opcional

Tabla 13: Esquema de la tabla "Banners".

2.2.2.1.14 Logos

Tabla para almacenar logos personalizados.

Tabla	Columna	Tipo de dato	Descripción
Logos	name	String	Nombre de la imagen
	description	String	Descripción opcional

Tabla 14: Esquema de la tabla "Logos".

Para que los distintos microservicios puedan leer y escribir en la base de datos primero deben conectarse a esta. Si bien se puede implementar la lógica de conexión en cada microservicio, no se considera una buena práctica, ya que si se modifica el esquema de la base de datos será necesario realizar cambios en todos los microservicios. Es recomendable crear un único microservicio capaz de interactuar directamente con la base de datos.

Para este nuevo microservicio se utiliza Node js debido a que este ofrece una gran cantidad de paquetes (librerías) que permitirán realizar el manejo de la base de datos con facilidad y seguridad.

Para poder leer y escribir en la base de datos es recomendable utilizar un ORM (Object-relational mapping), ya que este hará más sencillo, seguro, mantenible y escalable el manejo de la base de datos desde nodejs.
Se utiliza Sequelize [12], un ORM basado en la funcionalidad de promesas

que ofrece JavaScript y es capaz de ser utilizado con diferentes bases de datos: Postgres, MySQL, MariaDB, SQLite y Microsoft SQL. Sequelize recibe constantes actualizaciones, mejorando su performance y seguridad, además de que cuenta con gran cantidad de funcionalidades que permiten crear modelos, que luego serán traducidos tablas en la base de datos. La librería también permite añadir validaciones para los diferentes campos en el mismo archivo donde se define modelo, de esta manera se evita ingresar a la base de datos valores no deseados.

Además, Sequelize brinda la posibilidad de realizar consultas complejas con sanitizado de datos.

Este microservicio se comunicará con los demás utilizando el protocolo HTTP. En adición, se hace uso de la librería Finale [13], de nodejs, con la finalidad de crear endpoints (puntos de acceso para otros microservicios) REST flexibles, estándares, reutilizables y escalables para los modelos creados previamente por Sequelize.

Además, en el microservicio se implementó un conjunto extra de endpoints, que permiten realizar consultas más complejas a la base de datos. Estos se detallan a continuación:

- Endpoint para obtener el número de vehículos en el establecimiento, donde se puede filtrar por número de patente.
- Endpoint para actualizar los registros en la tabla de la base de datos.
- Endpoint para consultar el historial, con paginación, donde se puede filtrar por número de patente.
- Endpoint para obtener fotografías de vehículos tomadas por la cámara.
- Endpoints para obtener imágenes de logo y banner.
- Endpoints para llenar los distintos gráficos.

Para añadir el manejo específico de diferentes verbos HTTP (GET, POST, DELETE, etc.) y gestionar diferentes URL se hace uso del framework web Express [14], que funciona bajo nodejs.

Por otro lado, en ciertos casos, es necesario que se informe lo más pronto posible el cambio de ciertas tablas en la base de datos al resto de microservicios. Por ejemplo, si hubiese un automóvil en la salida, se debe

comunicar lo antes posible al encargado del estacionamiento que hay un vehículo al cual se le debe cobrar.

Para estos casos particulares donde se desee una comunicación rápida y fluida, el microservicio hace uso del protocolo de comunicación WebSocket, implementado en la librería de nodejs, Socket.IO [15].

2.2.3 Servicio de respaldo de base de datos

Con la finalidad de que el administrador del estacionamiento pueda almacenar copias de la base de datos se creó un microservicio que implementa Cron, una aplicación de línea de comandos que permite planificar eventos para que sean ejecutados en una fecha y hora determinada.

Cron tiene programado un script que realiza una copia de seguridad de la base de datos y de las imágenes de patentes reconocidas. Se comprimen los directorios donde se encuentra esta información en un archivo *zip* cuyo nombre tiene el formato *"db-backup_YYYY-mm-dd_HH-MM-SS.zip"*. Por ejemplo, para una copia de seguridad realizada el 1 de noviembre de 2021 a las 3:00am el nombre del archivo generado será *"db-backup_2021-11-01_03-00-00.zip"*.

Por defecto la copia de seguridad se realiza el día 1 de cada mes, a las 3:00am.

Una mejora a futuro podría ser, permitir que el usuario pueda elegir la frecuencia de generación del archivo de respaldo desde la sección de configuración de la página web de administración.

2.2.4 Servicio de detección de vehículos

2.2.4.1 Algoritmos de reconocimiento de tipo de vehículo y patente

En el sistema es fundamental contar con reconocimiento de imágenes diferentes razones. Como primer motivo es necesario reconocer el tipo de vehículo para saber que tarifa se le debe asignar. El segundo motivo es el reconocimiento del código de la patente para poder automatizar el registro de entrada, salida y cálculo del monto a cobrar.

Se investigaron diferentes algoritmos que permitiesen llevar a cabo el reconocimiento de imágenes, y se llegó a la conclusión de que los métodos que actualmente dan los mejores resultados están basados en algoritmos de aprendizaje profundo mediante la utilización de redes neuronales convolucionales.

2.2.4.1.1 Conceptos necesarios

El aprendizaje automático tiene como finalidad que las computadoras puedan “aprender” de la experiencia, que es presentada en forma de datos, para así poder problemas en un entorno real.

El concepto de aprendizaje automático abarca diversas técnicas y algoritmos, una de ellas es el aprendizaje profundo (en inglés, deep learning), y ha demostrado, en los últimos años, ser de los algoritmos más eficientes para una amplia variedad de aplicaciones.

Un modelo de aprendizaje profundo se encuentra compuesto por una red neuronal, que no es más que un algoritmo cuya entrada es un conjunto de valores, a los cuales se les aplican funciones matemáticas no lineales con el fin de obtener un resultado.

Las neuronas (también conocidas como perceptrones o nodos) son las unidades fundamentales de toda red neuronal, cada una puede recibir uno o varios valores de entrada, y luego aplicarles una transformación.

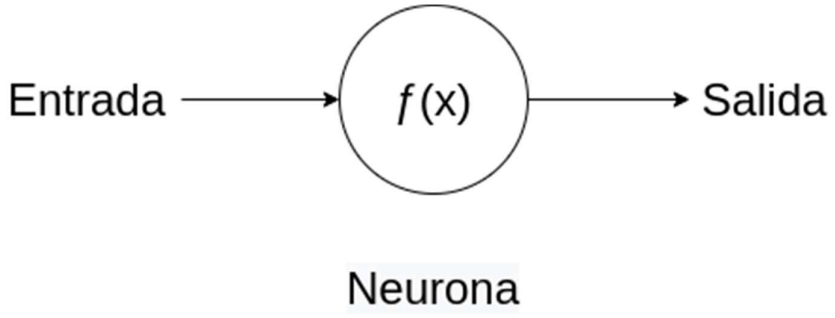


Figura 29: Neurona elemental.

Cada neurona es capaz de transformar el valor de su entrada mediante la aplicación de una función lineal de la forma:

$$y = a_1x_1 + a_2x_2 + \cdots + a_nx_n + b$$

Donde $x_1, x_2, \ldots, x_n$ son las entradas, $a_1, a_2, \ldots, a_n$ son valores constantes, conocidos como pesos que afectan a las entradas, b es una constante llamada sesgo (o *bias* en inglés) y y es la salida.

Con el objetivo de que la neurona pueda tener un conjunto más complejo de respuestas se le aplica una no linealidad a su salida, lo que resulta en la salida una función no lineal del tipo:

$$Y = \delta(y) = \delta(a_1x_1 + a_2x_2 + \cdots + a_nx_n + b)$$

Donde δ es el componente no lineal. Que podría ser, por ejemplo, una función exponencial, logarítmica, tangente hiperbólica, etc.

Debido al nivel de dificultad que conlleva en el procesamiento de imágenes, una red de una sola neurona es incapaz de procesar correctamente una imagen debido a su complejidad. Por este motivo se utilizan redes compuestas por diferentes capas, donde en cada una puede haber diferente número de neuronas.

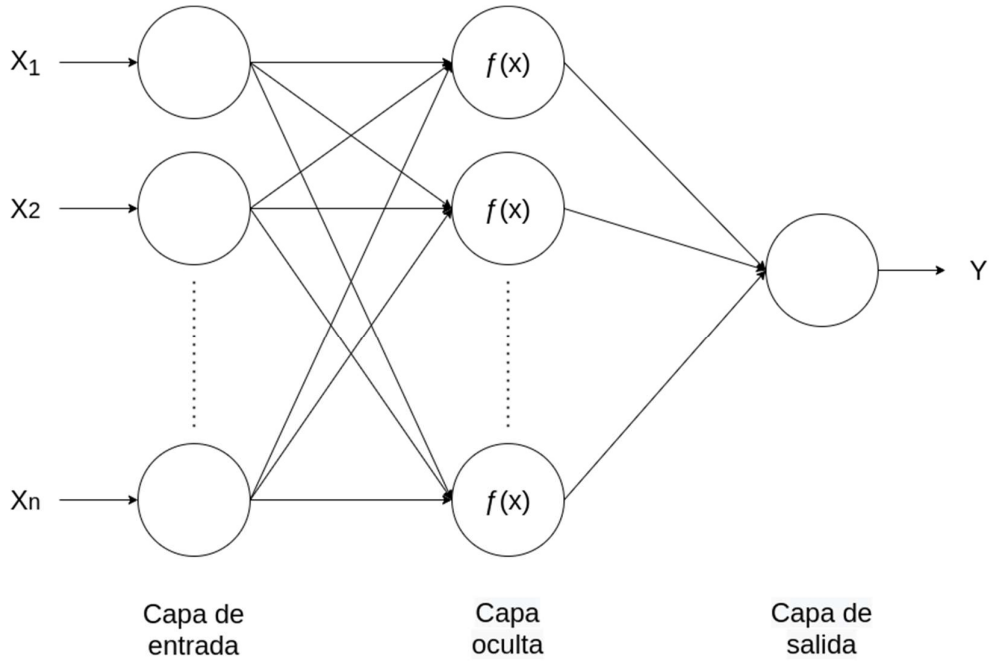


Figura 30: Diagrama general de una red neuronal.

Generalmente se pueden definir tres capas en toda red neuronal, la capa de entrada, donde se encuentran los valores de los datos, una o varias capas ocultas, las cuales se encargan de realizar las transformaciones de los valores de salida de la capa anterior, y la capa de salida, mediante la cual se obtienen los resultados de la predicción del algoritmo.

Las funciones no lineales para cada neurona son elegidas al momento de definir la arquitectura que tendrá la red. Estas funciones permanecen constantes durante la ejecución del algoritmo. Sin embargo, los valores de $a = a_1, a_2, \dots, a_n$ y b son ajustados durante el proceso de entrenamiento para obtener la salida esperada.

Se suele configurar el modelo para que este se inicialice con valores de pesos aleatorios y los valores de sesgo en cero, esto ocasiona que, para un modelo no entrenado, los resultados obtenidos para una predicción sean aleatorios. Durante el proceso de entrenamiento se utilizan algoritmos de retro-propagación, de esta manera cuando el resultado no es el esperado, mediante el proceso de retro-propagación, se determina el conjunto de neuronas que más influyó en la decisión, y mediante una función de minimización se cambia el valor de las constantes a y b, con la intención de que la próxima vez que se produzca una entrada asimilar el resultado sea más aproximado al esperado.

Este proceso se realiza repetidas veces para un gran conjunto de datos, con el objetivo de que, al finalizar la etapa de entrenamiento, el sistema sea capaz de predecir de manera correcta los valores de salida para entradas nunca antes vistas, pero que guarden cierto grado de similitud con los datos utilizados durante el entrenamiento.

Un tipo particular de red neuronal son las redes neuronales convolucionales, ampliamente utilizadas en el área de procesamiento de imágenes debido a que son capaces de extraer las relaciones espaciales y temporales entre los distintos valores que conforman los datos de entrada [16]. Para el caso del procesamiento de imágenes los datos de entrada serán los valores de los pixeles de las imágenes.

Una red neuronal convolucional se destaca por utilizar la operación matemática de convolución entre la imagen de entrada y una matriz conocida como filtro o kernel.

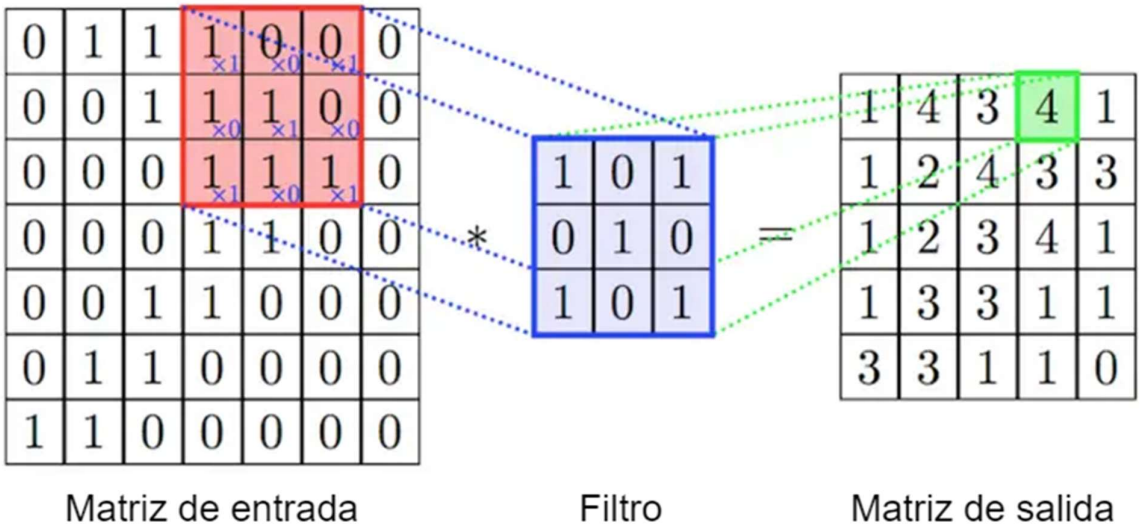


Figura 31: Proceso de convolución para matrices. Esta imagen fue tomada de [17].

La convolución se realiza sobre una matriz de entrada, que puede estar compuesta por los valores de los pixeles de una imagen, o bien puede ser el resultado de una convolución anterior.

Sobre la capa de partida se aplica una matriz de filtro, tomando una sub-matriz de la entrada (en rojo) del mismo tamaño que la del filtro y realizando una multiplicación componente a componente. Luego, se suman todos los valores resultantes y se obtiene el valor de un elemento (en verde) de la matriz convolucionada.

Este proceso se repite a lo largo de toda la matriz de entrada, con el fin de obtener también una matriz en la salida.

Los valores resultantes dependerán del filtro utilizado. Dependiendo de los valores que tomen las componentes de este, se pueden lograr diferentes resultados a la salida. Por ejemplo, se puede realizar una matriz para detectar bordes verticales, horizontales, diagonales, cambios de contraste, formas, texturas, etc.

A continuación, se muestran algunos ejemplos donde, para una misma imagen de entrada se aplican distintos tipos de filtro de tamaño 3x3. Los ejemplos fueron extraídos de [18].

Detección de bordes verticales.
Filtro:

-1	0	1

-2	0	2
-1	0	1

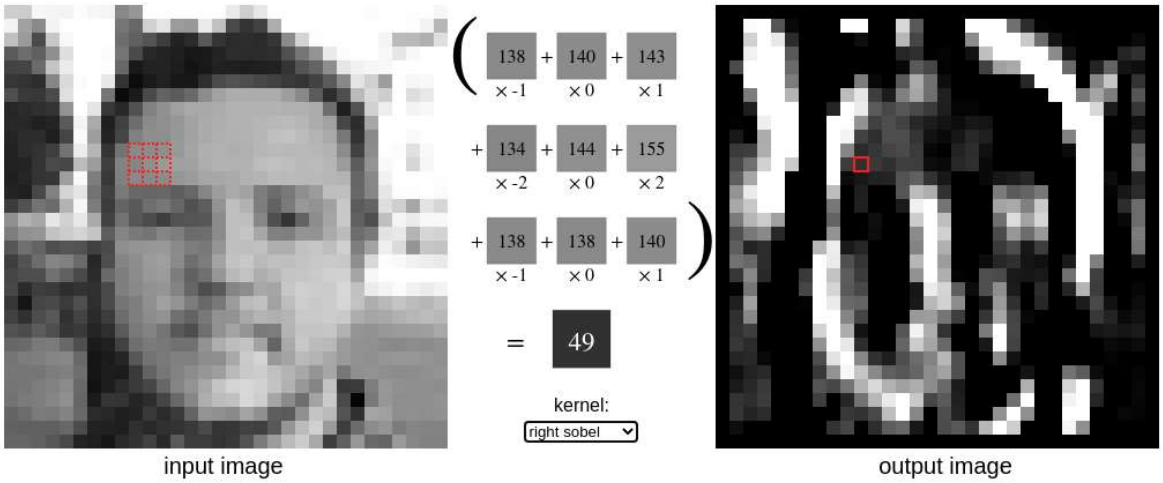


Figura 32: Detección de bordes verticales.

Detección de bordes horizontales.
Filtro:

1	2	1
0	0	0
-1	-2	-1

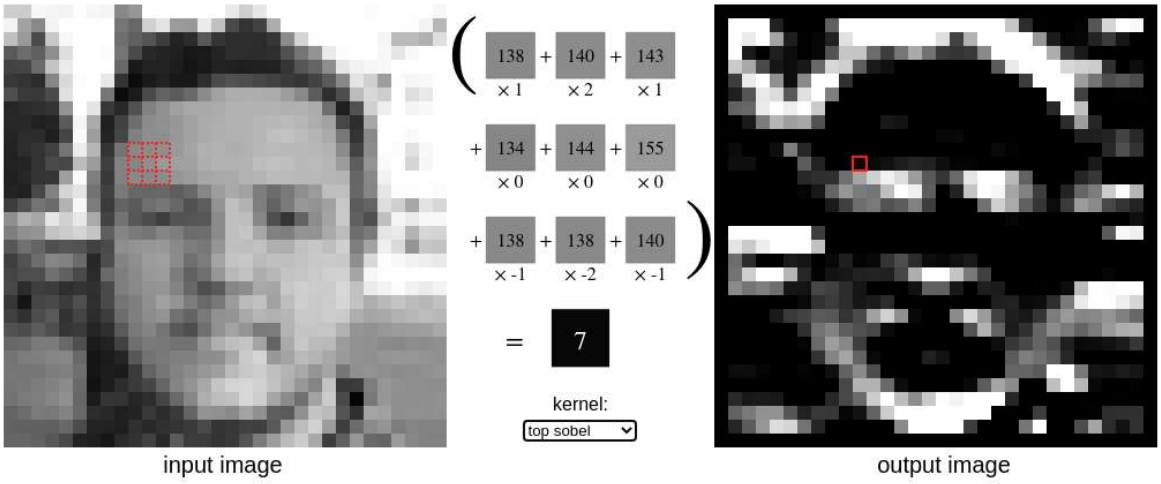


Figura 33: Detección de bordes horizontales.

Desenfoque de imagen.
Filtro:

0.0625	0.125	0.0625
0.125	0.25	0.125
0.0625	0.125	0.0625

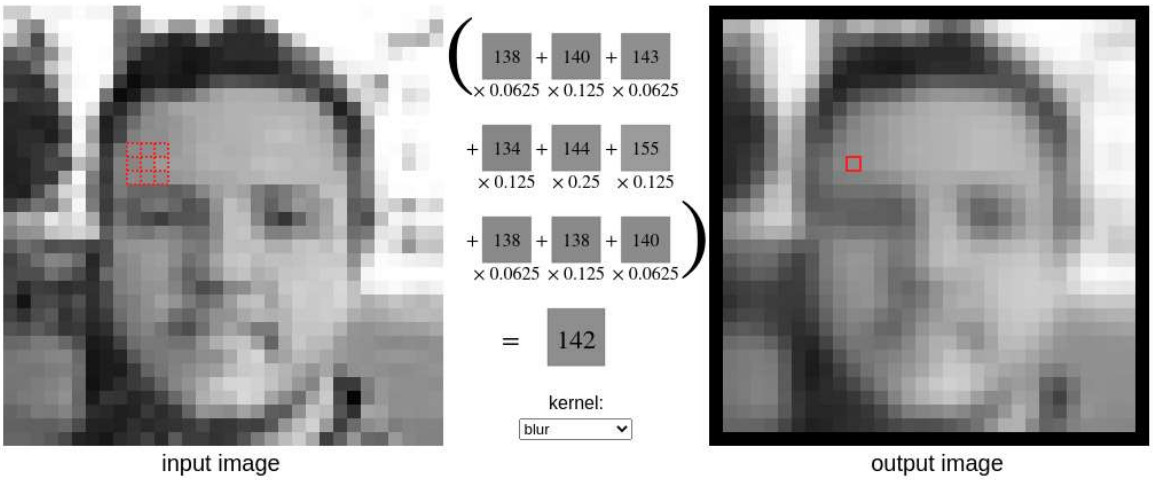


Figura 34: Desenfoque de imagen.

Durante el proceso de entrenamiento, los valores de los filtros serán modificados por el algoritmo de retro-propagación con el fin de encontrar el conjunto de valores que permita obtener el resultado que más se asemeje al esperado.

La estructura general de una red convolucional es la siguiente:

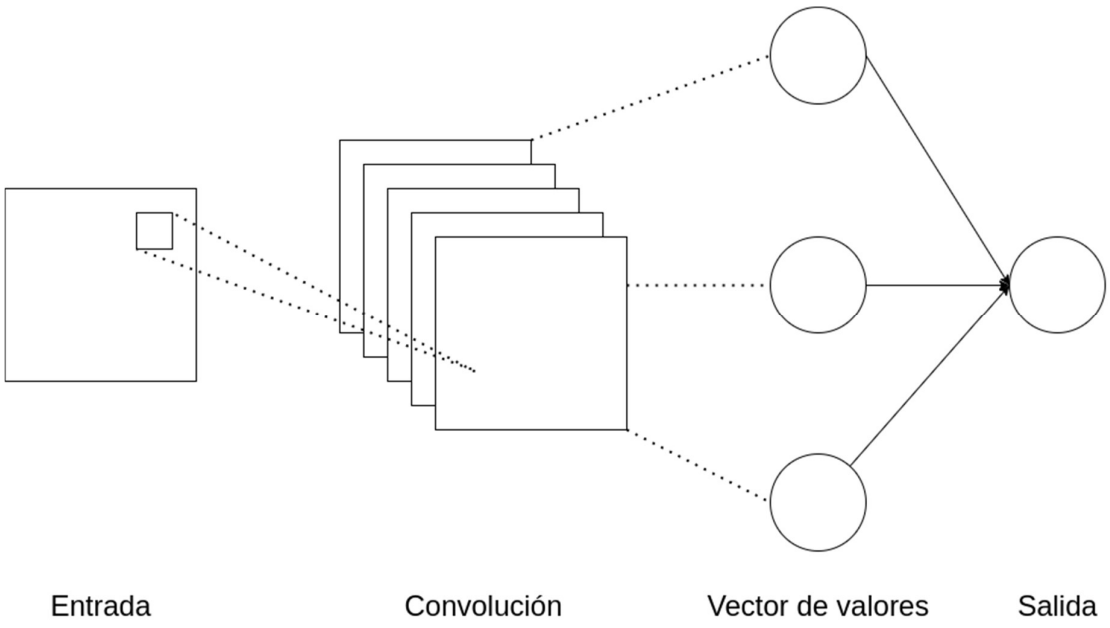


Figura 35: Diagrama de una red neuronal convolucional genérica.

A la matriz de entrada se le aplican diferentes filtros para lograr diferentes resultados que permitan identificar distintos patrones en la imagen.

Generalmente este proceso se repite varias veces con la finalidad de obtener patrones más complejos. Luego de repetidas capas de convolución, se toman todos los valores de las distintas matrices y se concatenan en un mismo vector, que finalmente es procesado para obtener uno o varios valores de salida, dependiendo del tipo de predicción que se desee realizar.

2.2.4.2 Modelos utilizados

En el sistema se utilizan redes neuronales convolucionales con el objetivo de clasificar el tipo de vehículo, reconocer la ubicación de la patente y detectar su código.

2.2.4.3 Reconocimiento del tipo de vehículo

Es necesario reconocer el tipo de vehículo en la entrada para que el sistema pueda saber que tarifa aplicar al momento del cómputo del costo de la estadía. Se requiere de un modelo de clasificación de tipo de vehículo, el cual permita distinguir entre automóviles, camionetas y motocicletas.

En un principio se planteó utilizar un modelo completamente entrenado (es decir, con los pesos de cada neurona ya definidos), pero esto no pudo ser así debido a que no se logró encontrar ningún modelo que clasificara imágenes exclusivamente en las tres categorías planteadas. Por lo tanto, se optó por utilizar una arquitectura definida, con las primeras capas entrenadas, y solo entrenar las capas finales del modelo.

Se utiliza VGG16 [19], creada por el Visual Geometry Group at Oxford. VGG16 cuenta con 16 capas pre-entrenadas con el conjunto de datos ImageNet [20]. El modelo original toma imágenes 224x224, pero en este caso al tener fotografías de diferentes tamaños se decidió utilizar un tamaño de entrada de 400x400. Además, la arquitectura VGG16 es capaz de clasificar imágenes en 1000 categorías distintas, pero como en el presente caso se desea realizar una clasificación entre solamente 3 categorías se modifican las ultimas capas para cumplir estos requisitos.

A continuación, se encuentra detallado el modelo final.

Capa	**Tamaño de salida**	**Parámetros**	**Obs.**
input_1 (InputLayer)	[(None, 400, 400, 3)]	0	
block1_conv1 (Conv2D)	(None, 400, 400, 64)	1792	
block1_conv2 (Conv2D)	(None, 400, 400, 64)	36928	
block1_pool (MaxPooling2D)	(None, 200, 200, 64)	0	
block2_conv1 (Conv2D)	(None, 200, 200, 128)	73856	
block2_conv2 (Conv2D	(None, 200, 200, 128)	147584	
block2_pool (MaxPooling2D)	(None, 100, 100, 128)	0	
block3_conv1 (Conv2D)	(None, 100, 100, 256)	295168	
block3_conv2 (Conv2D)	(None, 100, 100, 256)	590080	
block3_conv3 (Conv2D)	(None, 100, 100, 256)	590080	
block3_pool (MaxPooling2D)	(None, 50, 50, 256)	0	
block4_conv1 (Conv2D)	(None, 50, 50, 512)	1180160	
block4_conv2 (Conv2D)	(None, 50, 50, 512)	2359808	
block4_conv3 (Conv2D)	(None, 50, 50, 512)	2359808	
block4_pool (MaxPooling2D)	(None, 25, 25, 512)	0	
block5_conv1 (Conv2D)	(None, 25, 25, 512)	2359808	
block5_conv2 (Conv2D)	(None, 25, 25, 512)	2359808	
block5_conv3 (Conv2D)	(None, 25, 25, 512)	2359808	
block5_pool (MaxPooling2D)	(None, 12, 12, 512)	0	Personaliz.
conv2d (Conv2D)	(None, 12, 12, 512)	9437696	Personaliz.
conv2d_1 (Conv2D)	(None, 12, 12, 512)	9437696	Personaliz.

max_pooling2d (MaxPooling2D)	(None, 6, 6, 512)	0	Personaliz.
flatten (Flatten)	(None, 18432)	0	Personaliz.
dense (Dense)	(None, 4096)	75501568	Personaliz.
dense_1 (Dense)	(None, 1024)	4195328	Personaliz.
dense_2 (Dense)	(None, 128)	131200	Personaliz.
dense_3 (Dense)	(None, 3)	387	Personaliz.

Tabla 15: Arquitectura del modelo VGG16 con capas personalizadas.

Parámetros totales: 113.418.563
Parámetros entrenables: 98.703.875
Parámetros no entrenables: 14.714.688

Como se mencionó antes, durante el entrenamiento, solo se modificarán los parámetros correspondientes a las capas personalizadas añadidas al final. El modelo base VGG16 cuenta con una gran cantidad de parámetros entrenables, debido a esto, reentrenarlo conllevaría un gran costo computacional y de tiempo, el cual no se justificaría en cuanto ganancias en la performance del modelo para esta aplicación en específico.

La técnica de utilizar un modelo base pre-entrenado con un conjunto de datos diferente al que se utilizará en un entorno de producción se conoce como aprendizaje transferido (en inglés, transfer learning) y ha demostrado en los últimos años ser sumamente eficiente [21].

2.2.4.4 Generación del conjunto de datos

Para poder entrenar un modelo de aprendizaje profundo el primer paso es contar con un conjunto de datos (en inglés, dataset) similares a los que se pudiesen producir en situaciones reales. En un principio, mientras mayor sea la cantidad de datos, más opciones tendrá el modelo para aprender y será más efectivo en la tarea a desempeñar. Debido a esto se buscó tener la mayor cantidad posible de datos, en este caso imágenes de vehículos.

Se utilizaron diferentes conjuntos de datos de imágenes tomadas de internet, adaptadas según las necesidades para obtener las clases requeridas.
Las imágenes fueron tomadas de los siguientes sitios:

2.2.4.4.1 Stanford Cars Dataset:

Cuenta con 8144 imágenes de entrenamiento de automóviles y camionetas clasificadas en 196 clases dentro de las cuales se encuentra incluida el tipo de vehículo. Sin embargo, la categoría de tipo de vehículo presenta una clasificación diferente a la requerida, debido a esto se debieron volver a mapear las etiquetas para que correspondan a una de las siguientes tres: automóvil, camioneta o motocicleta.

2.2.4.4.2 PASCAL VOC2012 Dataset:

Cuenta con una gran cantidad de imágenes, clasificadas en cuatro grupos generales: persona, animal, vehículo e interior. Dentro de vehículo se tienen las subclasificaciones de: avión, bicicleta, bote, colectivo, automóvil, motocicleta y tren.

En este caso solo se toman las imágenes que contengan motocicletas, pero no haya colectivos o automóviles.

Se decidió no tomar las imágenes de la categoría automóvil ya que esta no diferencia entre camionetas y la convención de automóvil utilizada en este trabajo.
De este conjunto de datos se obtuvieron 459 imágenes.

2.2.4.4.3 Google Images:

Utilizando la herramienta *Chrome Image Scrapper* [22] fue posible obtener imágenes motocicletas y camionetas que se encuentren indexadas en el buscador de Google. Se consiguieron 1367 fotografías de motocicletas y 709 de camionetas.

Finalmente se almacenaron todas las imágenes en un mismo directorio y se creó un archivo de formato csv que contiene el nombre de cada fotografía junto con el tipo de vehículo que se encuentra en ella.

En total se obtuvieron 10679 imágenes, de las cuales 5871 corresponden a automóviles, 2982 a camionetas y 1826 a motocicletas.

2.2.4.5 Entrenamiento

Para el entrenamiento se utilizó el conjunto de datos generado, con una división 80/10/10 (80% de entrenamiento, 10% validación y 10% de prueba). Esto da como resultado 8543 imágenes para entrenamiento, 1068 imágenes para validación y la misma cantidad para pruebas. Se entrenó el modelo en épocas (una época corresponde a una iteración completa por el conjunto de datos de entrenamiento). Para que los datos no sean exactamente iguales (y simular un conjunto mayor de datos) en cada época, se realiza un proceso conocido como "datos aumentados" (data augmentation) [23], mediante el cual las imágenes de entrenamiento en cada época se verán afectadas aleatoriamente por los siguientes parámetros:

- Rotaciones de hasta 15 grados.
- Recortado de la imagen de hasta un 20%.
- Acercamiento de hasta 20%.
- Espejado horizontal.

El modelo se entrenó durante 40 épocas. Por limitaciones de hardware no es posible alimentar el modelo con el conjunto completo de datos de entrenamiento en una única iteración, debido a esto se realizan múltiples iteraciones (o pasos) con conjuntos de 64 imágenes hasta haber pasado por todos los datos de entrenamiento, momento en el cual se considera por terminada una época. Se realizan 133 pasos por época para los datos de entrenamiento y 16 pasos por época para los datos de validación. La métrica a utilizar para el entrenamiento será la precisión.

A continuación, se muestran los resultados obtenidos del entrenamiento para las primeras y últimas 5 épocas.

Época	Pérdida	Precisión	Precisión Validación	Pérdida Validación
1/40	13.9442	0.5807	0.4082	0.7939
2/40	0.7939	0.7867	0.3782	0.8193
3/40	0.4296	0.7962	0.3850	0.8213
4/40	0.3915	0.8149	0.3720	0.8154
5/40	0.3611	0.8337	0.3269	0.8389

...	...	...	...	...
36/40	0.1119	0.9573	0.3195	0.8994
37/40	0.1149	0.9558	0.3703	0.8916
38/40	0.1075	0.9581	0.2877	0.9004
39/40	0.1099	0.9586	0.3751	0.8955
40/40	0.0930	0.9652	0.2786	0.9053

Tabla 16: Resultados del entrenamiento del modelo de detección de tipo de vehículo.

2.2.4.6 Pruebas

Para las pruebas se utilizaron 1068 imágenes (tomadas del mismo conjunto de datos original que luego se dividió en entrenamiento, validación y pruebas), la precisión obtenida es del 92.32%.

A continuación, se muestra la Matriz de Confusión, donde las columnas representan la clase verdadera que le corresponde a cada imagen, mientras que en las filas se muestran las clases inferidas por el modelo.

	Automóvil	**Motocicleta**	**Camioneta**
Automóvil	566	4	35
Motocicleta	3	156	0
Camioneta	37	3	264

Tabla 17: Matriz de Confusión para el modelo de detección de tipo de vehículo.

Los resultados obtenidos son positivos, sin embargo, se espera que en un caso real la precisión sea un poco menor debido a que las imágenes reales a clasificar pueden diferir a las utilizadas en el entrenamiento, validación y prueba del modelo.

2.2.4.7 Modelo de reconocimiento de patentes

En un principio no se sabía si sería necesario utilizar un modelo para reconocer la ubicación de la patente, pero luego de la realización de pruebas con un modelo de reconocimiento del texto, se detectó que los resultados para la imagen completa de entrada tomada por la cámara generalmente fallaban. Se hicieron distintas pruebas en las que se ejecutaban el modelo de

reconocimiento de texto utilizando la misma fotografía, con la diferencia de que en el primer caso la entrada del modelo era la imagen completa y en el segundo un recorte del área donde se encontraba la patente. Al hacer estas pruebas fue evidente que el modelo tenía una mejor performance al recibir como entrada el área de la patente en lugar de la imagen completa.

Figura 36: Imagen tomada con la cámara de prueba.

Figura 37: Imagen recortada del área de la patente.

Para el primer caso, el de la imagen completa, la patente reconocida fue “ABW257”, mientras que para el caso de la patente recortada se reconoció “AC979IL”.

Debido a esto se planteó a utilización de un modelo para poder recortar el área de la patente en la imagen previo a la utilización del modelo de reconocimiento de texto.

En una primera instancia de planteó la utilización de la API de Detección de Objetos [24] provista por TensorFlow [25], una plataforma de código abierto mantenida por Google.

Se consideró utilizar un modelo pre-entrenado ofrecido por la API, en concreto con la arquitectura MobileNet [26]. Dada a conocer en 2017, es una arquitectura de redes neuronales convolucionales, diseñada para aplicación de visión en dispositivos móviles y sistemas embebidos.

Se obtuvieron un conjunto de datos de entrenamiento de 728 imágenes de automóviles y camionetas.

Uno de los principales problemas consistía en la diferencia en el factor de forma que presentan las patentes de motocicletas. Si bien las imágenes del conjunto de datos tenían un 40% de patentes argentinas, ninguna era de motocicletas. Por lo tanto, a pesar de obtenerse un 84.5% de acierto para patentes de automóviles y camionetas, el porcentaje de acierto para motocicletas no era superior al 30%

Figura 38: Formatos de patentes para automóviles (izquierda) y motocicletas (derecha). Esta imagen fue tomada de [27].

Finalmente se optó por utilizar otra arquitectura de red neuronal convolucional conocida como YOLO (por sus siglas en inglés, You Only Look Once) [28]. Este sistema ha demostrado ser sumamente efectivo en la detección de objetos. El algoritmo divide la imagen en regiones, examina cada una de ellas, predice múltiples cajas delimitadoras, y la probabilidad de detección cada una de las clases de entrenamiento, definidas a priori, para cada una de estas cajas.

Figura 39: Detección de ubicación de patente en un automóvil.

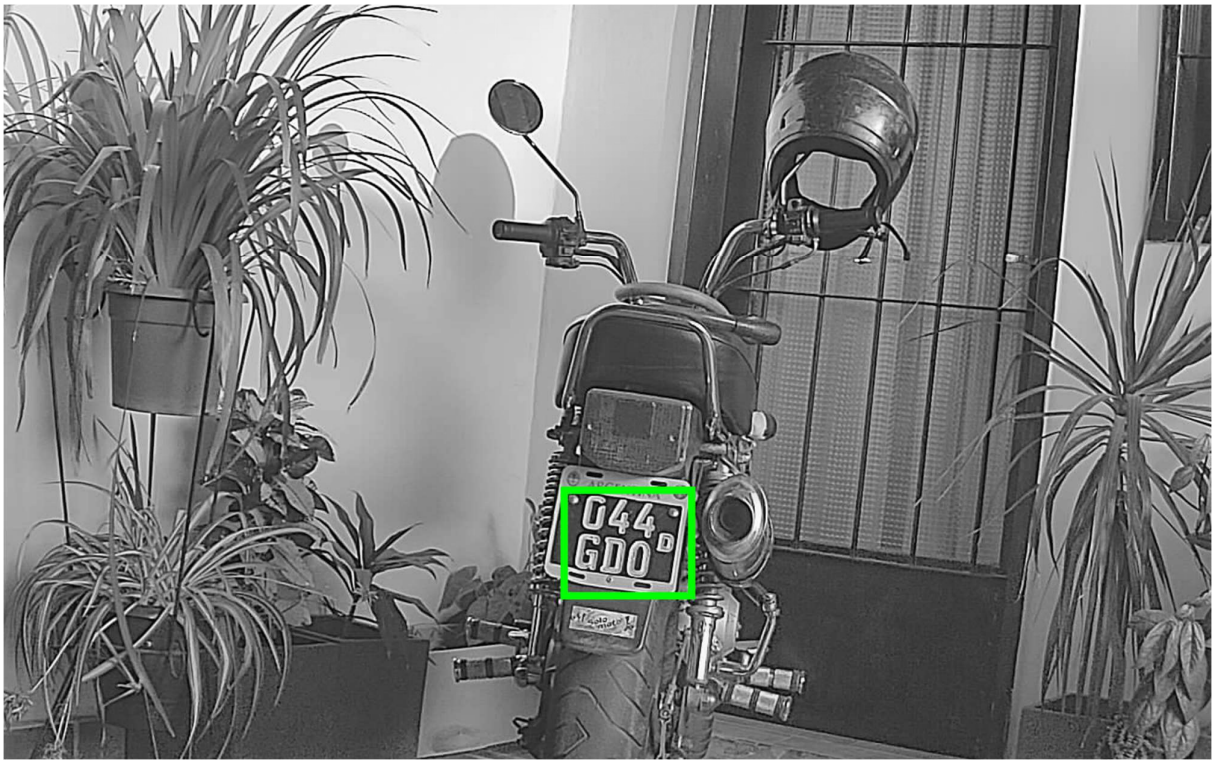


Figura 40: Detección de ubicación de patente en una motocicleta.

Se utilizó una implementación basada en YOLO v4 y entrenada específicamente para la detección de patentes de vehículos [29].

Antes de introducir la imagen en el modelo se realiza un preprocesamiento de esta que consiste en:

- Redimensionar el tamaño de la imagen para que sea apropiada al tamaño de entrada aceptado por el modelo.
- Normalizar los valores de los pixeles a una escala decimal entre 0 y 1 para que el preprocesamiento sea lo más parecido posible al utilizado con las imágenes de entrenamiento [30].

Luego de realizada la predicción el modelo devuelve un arreglo de valores normalizados de los puntos delimitadores *xmin*, *ymin*, *xmax*, *ymax* de la caja y el valor de la confianza en la detección también normalizado.

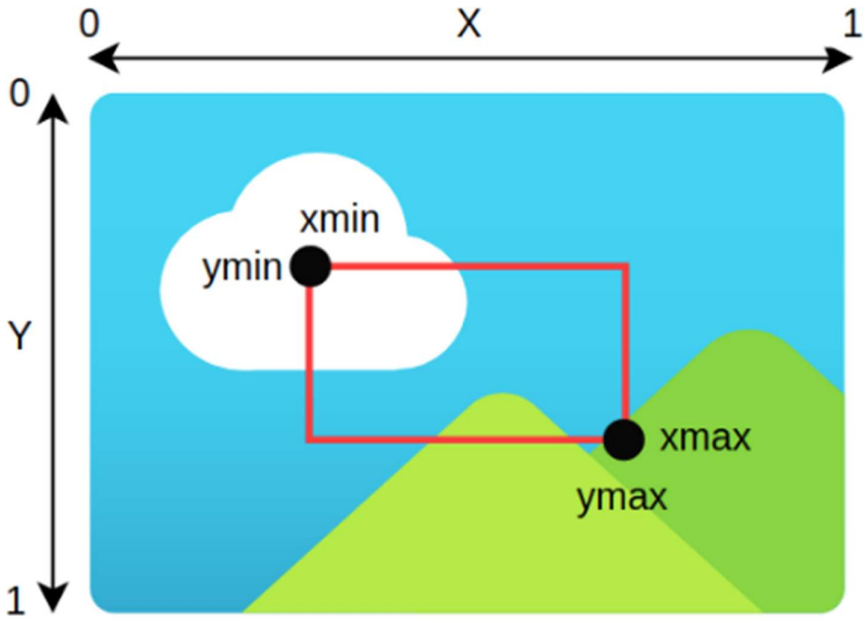


Figura 41: Puntos de interés devueltos por el modelo.

A este conjunto de resultados se le aplica una función para filtrar la cantidad máxima de cajas limitadoras que se desean obtener (en este caso se toma como máximo 1 ya que se supone solo una patente por imagen) y el valor mínimo de confianza necesario para tomar la predicción como válida.

Una vez aplicada la función de filtrado se puede obtener un único resultado, el de la confianza más alta, o ninguno, en caso de que no haya detecciones o no se supere el valor umbral de confianza.

Los valores de las cajas siguen estando normalizados, con el formato mencionado anteriormente.
Se utiliza una segunda función que convierte los valores normalizados de las esquinas de las cajas a valores de pixeles, de esta manera es posible obtener los valores *xmin*, *ymin*, *xmax*, *ymax* en pixeles.

2.2.4.7.1 Pruebas

Para comprobar la fiabilidad del modelo se realizó una prueba con 816 imágenes tomadas por las mismas cámaras que se utilizarán luego en el sistema. Se obtuvo un acierto en 775 imágenes, lo que da un porcentaje de acierto del 95%. Cabe destacar que de las 816 imágenes 85 eran de motocicletas, de estas solo se reconocieron correctamente 44, por lo tanto, el porcentaje de acierto para motocicletas está en torno al 51%.

2.2.4.7.2 Reconocimiento de texto en patentes.

Finalmente, es necesario reconocer los caracteres alfanuméricos de la patente para poder registrar efectivamente la entrada en el sistema y saber a qué vehículo corresponde.

Se pretendió utilizar, en un principio, un único modelo para el reconocimiento de patentes, en concreto uno diseñado con redes neuronales convolucionales y entrenado específicamente para reconocer patentes argentinas, tanto los modelos viejos como los nuevos.

A continuación, se muestra parte de la arquitectura del modelo. El esquema completo puede verse en [31].

Capa	**Tamaño de salida**	**Parámetros**
input_16 (InputLayer)	[(None, 70, 140, 1)]	0
conv2d_141 (Conv2D)	(None, 70, 140, 64)	576
batch_normalization_141 (BatchNormalization)	(None, 70, 140, 64)	256
activation_246 (Activation)	(None, 70, 140, 64)	0
conv2d_142 (Conv2D)	(None, 70, 140, 32)	18432
batch_normalization_142 (BatchNormalization)	(None, 70, 140, 32)	128
activation_247 (Activation)	(None, 70, 140, 32)	0
conv2d_143 (Conv2D)	(None, 70, 140, 32)	9216

batch_normalization_143 (BatchNormalization)	(None, 70, 140, 32)	128
activation_248 (Activation)	(None, 70, 140, 32)	0
conv2d_144 (Conv2D)	(None, 70, 140, 64)	18432
batch_normalization_144 (BatchNormalization)	(None, 70, 140, 64)	256
activation_249 (Activation)	(None, 70, 140, 64)	0
max_pooling2d_75 (MaxPooling2D)	(None, 35, 70, 64)	0
…	…	…
activation_263 (Activation)	(None, 37)	0
activation_264 (Activation)	(None, 37)	0
activation_265 (Activation)	(None, 37)	0
concatenate_15 (Concatenate)	(None, 259)	activation_259 activation_260 activation_261 activation_262 activation_263 activation_264 activation_265

Tabla 18: Primeras y últimas capas del modelo de redes neuronales convolucionales para detección de texto en patente.

Parámetros totales: 2.043.459
Parámetros entrenables: 2.037.955
Parámetros no entrenables: 5.504

Se hicieron pruebas utilizando 731 imágenes de patentes de tres automóviles distintos. De estas imágenes solamente 9 arrojaron resultados diferentes a una de las tres patentes utilizadas. Por lo tanto, para automóviles y camionetas el porcentaje a acierto es de 98%.

Respecto a las imágenes de motocicletas, se probaron 44 fotografías y ninguna fue reconocida con éxito.

Estos resultados se deben probablemente a que el modelo fue entrenado únicamente con patentes de automóviles, y no se tuvieron en cuenta la disposición de los números y letras en las motocicletas, por esta razón el porcentaje de acierto para las mismas es nulo.

Se intentó reentrenar el modelo con un nuevo conjunto de datos sin embargo solo el 10% de los datos eran correspondientes a motocicletas, por esta razón, los nuevos resultados no fueron mucho mejores que los obtenidos en la prueba con el modelo entrenado por defecto.

Ante esta situación y visto que el porcentaje de reconocimiento de patentes para automóviles y camionetas era del 98%, se optó por buscar un nuevo modelo para reconocer únicamente las patentes de motocicletas.

Para reconocer el texto en las patentes de motocicleta se utiliza la librería easyOCR [32], la cual proporciona un modelo de redes neuronales convolucionales recurrentes [33], que está compuesto por tres componentes fundamentales: extracción de características (Resnet [34] y VGG [19]), etiquetado secuencial (LSTM [35]) y decodificación (CTC [36]).

El proceso de reconocimiento para las imágenes de motocicletas cuenta con diferentes pasos.

Una vez cargada la imagen y recortada la patente por el modelo anterior, esta se debe convertir a escala de grises y luego a valores binarios con el fin de obtener una mejor performance del modelo.

Figura 42: Preprocesado de patente.

En este punto se ejecuta el modelo dos veces, uno dándole como entrada la imagen en escala de grises y otro la imagen binaria. Esto se decidió hacer así ya que durante las pruebas se observó que en algunos casos la imagen en escala de grises devolvía mejores resultados que la imagen binaria.

Para la imagen de la patente mostrada anteriormente se obtiene el siguiente resultado:

[([[73, 33], [283, 33], [283, 75], [73, 75]], 'RGENTINA', 0.9998589303763633),

([[79, 75], [297, 75], [297, 191], [79, 191]], '044', 0.998259055275871),
([[70, 186], [296, 186], [296, 310], [70, 310]], 'gdo', 0.23518455465154398)]

Donde los números enteros indican los lados de la caja (donde se encuentra el texto), el texto es lo reconocido dentro de la caja y el valor decimal indica la confianza.

Vemos que a diferencia del modelo anterior este modelo no fue específicamente diseñado para reconocimiento de patentes, por lo tanto, devuelve todo el texto que encuentra dentro de la imagen.

Es necesario filtrar el texto adicional que no es de intentes, para esto se pasa el resultado a una función encargada de computar el tamaño de las cajas para cada detección, se supone que las dos cajas de mayor tamaño son las correspondiente a la parte inicial y final de la patente. De esta manera se filtra el texto adicional. Una vez obtenido el texto de la patente se concatena, se convierte en letras mayúsculas y se corrobora que el texto en la patente sea alfanumérico.

Como se mencionó antes el modelo se ejecuta dos veces con las imágenes en escala de grises y binaria. Al obtener los resultados estos se comparan y solo se toma como correcto el resultado que tenga un número de patente válido. En caso de que ambas patentes sean igualmente válidas se tiene en cuenta únicamente aquella que tenga el mayor valor de confianza.

Los distintos algoritmos de inteligencia artificial deben funcionar en conjunto para poder reconocer una entrada efectiva. Para utilizar en conjunto los modelos de clasificación de vehículo, reconocimiento de ubicación de patentes y detección de texto se diseñó un microservicio independiente, que hace uso de una API y encapsula todo el proceso de reconocimiento de imágenes utilizando los distintos algoritmos de aprendizaje automático.

A continuación, se puede apreciar un diagrama sencillo que demuestra el proceso de reconocimiento.

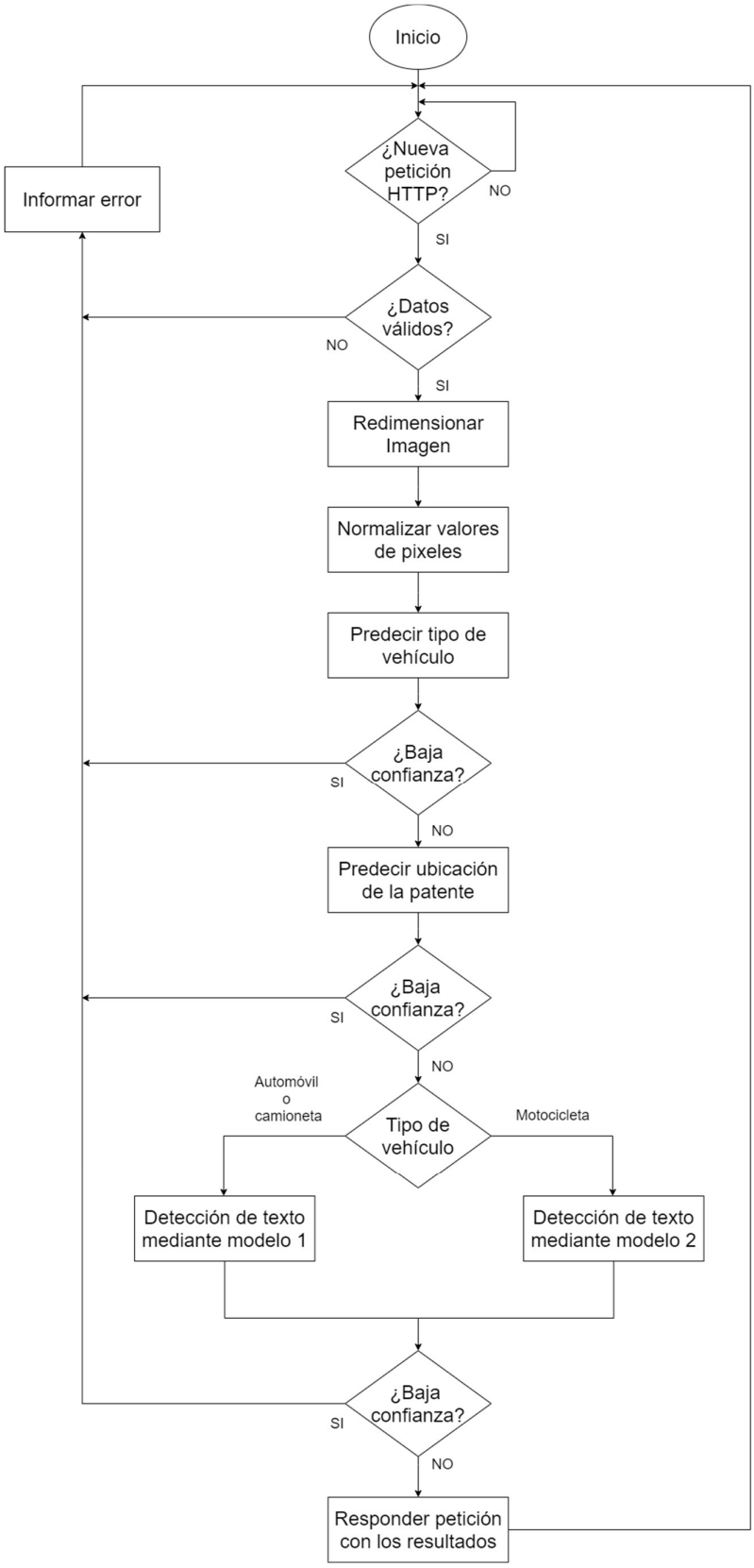


Figura 43: Diagrama de flujo de la lógica de funcionamiento del microservicio de detección de vehículos.

Al iniciar el servicio de detección son cargados en memoria los modelos de reconocimiento a utilizar y luego queda a la espera de peticiones HTTP tipo post para procesar imágenes. Al llegar un nuevo mensaje al servicio este chequea que el contenido de este sea válido. Se comprueba que la imagen pueda ser leída y que el formato sea el apropiado. Si se cumplen estos requisitos se procede a redimensionar y normalizar la imagen.

A continuación, se predice el tipo de vehículo, si la confianza del modelo es baja, no se puede estar seguro de que realmente haya un vehículo en la entrada, en este caso el sistema retorna la petición informando el error. En caso de que la confianza sea alta se puede proseguir con la predicción de la ubicación de la patente. Puede darse el caso de que el vehículo no posea patente o que el modelo no haya sido capaz de detectarla, en este caso se retorna un mensaje de error.

Si la patente pudo ser localizada se debe analizar a que tipo de vehículo corresponde, si pertenece a un automóvil o camioneta se utiliza el modelo de reconocimiento de texto en patentes argentinas [31] y si se tratase de una motocicleta se utiliza el modelo de reconocimiento de texto de easyOCR [32].

Finalmente, se mide el nivel de confianza, si es baja, se devuelve un error. De lo contrario se retorna una respuesta con los resultados de las predicciones y los niveles de confianza.

Al hacer pruebas de rendimientos en este modelo se descubrió que este microservicio tenía un consumo constante de 7.5GB de memoria RAM. Lo cual dificulta que pueda ser utilizado en un equipo compacto como una Raspberry Pi.

2.2.5 Microservicio Central – MQTT

Este servicio es el encargado de conectar todos los mencionados anteriormente. Aquí MQTT cumple un rol clave, ya que a través de este protocolo llega el mensaje que es el encargado de iniciar la secuencia de funcionamiento del sistema. El funcionamiento es como sigue:

Cuando algún sensor de proximidad detecta gracias a sus ultrasonidos que hay una presencia frente a él, de posiblemente un vehículo, este envía a través de MQTT su ID. Entonces el microservicio Central recibe este mensaje a través del Broker. Como el mensaje cuenta con la ID del dispositivo que lo envió, entonces se realiza una petición al servicio encargado de la base de datos para obtener toda la información asociada a dicha ID. El dato buscado es en que entrada/salida se encuentra ubicado.
Conociendo la ubicación de este, es posible obtener que cámara y que reflector se encuentran en el mismo lugar, y con estos datos en primer lugar se envía a través de MQTT la orden de encender el reflector, y una vez que esto sucedió, se solicita una fotografía al microservicio encargado.
Una vez que se dispone de la fotografía, esta es enviada a analizar al microservicio de reconocimiento de patentes. En caso de que la lectura de la patente fracase, independientemente del motivo, se incrementa un contador de errores interno del sistema. El objetivo de esto es llevar una cuenta de la cantidad de veces que se intentó leer una patente sin éxito y si se alcanzaron N errores consecutivos, se desactiva un tiempo T este microservicio, con el motivo de evitar procesamiento innecesario si se supone que estos van a fracasar y además se genera una alerta en la HMI para el que el operario conozca este problema. Cumplido el tiempo establecido, se reestablece el normal funcionamiento. Si la patente es leída correctamente, se continua con la secuencia.
Con la patente identificada, sus datos son enviados al servicio de base de datos para obtener información sobre el vehículo, y en caso de que no exista en la base de datos se lo crea. Luego se verifican dos cosas, si la fotografía se tomó en una entrada o una salida y si el vehículo ya se encontraba dentro del establecimiento. Esto abre cuatro posibilidades. Si el vehículo no se encuentra dentro, y la fotografía se tomó en una salida, entonces se descarta esta fotografía y finaliza la secuencia. Lo mismo sucede si el vehículo ya se encuentra dentro y la fotografía se tomó en una entrada.
En caso de que el vehículo no se encuentre dentro, y la fotografía se haya tomado en una entrada, entonces se está en el caso de que un vehículo intenta ingresar al establecimiento. En esta situación, se guarda en la base de datos el ingreso con la siguiente información:

- Fecha y hora,
- ID del vehículo

- Entrada por la cual ingresó.

Posteriormente, se guarda en la base de datos la fotografía asociada a este ingreso, junto con la patente leída y el tipo de vehículo, junto con la confianza obtenida en la lectura. Por último, se solicita a la base de datos la información del semáforo asociado a esta entrada, y se envía por MQTT la orden para colocarlo en color verde.

Por otra parte, en caso de que el vehículo se encuentre en una salida, y ya se encuentre dentro del establecimiento, entonces se realiza la lógica de salida. Aquí lo primero que se realiza es la verificación de si ya se indicó la orden de pago. En caso afirmativo, acá finaliza el programa y queda a la espera de un nuevo inicio. En caso negativo, se registra la orden de pago en la base de datos y se calcula el monto a pagar considerando el horario y fecha de ingreso. Finalmente se guarda en la base de datos el registro de salida con lo siguiente:

- ID de vehículo.
- ID de la salida.

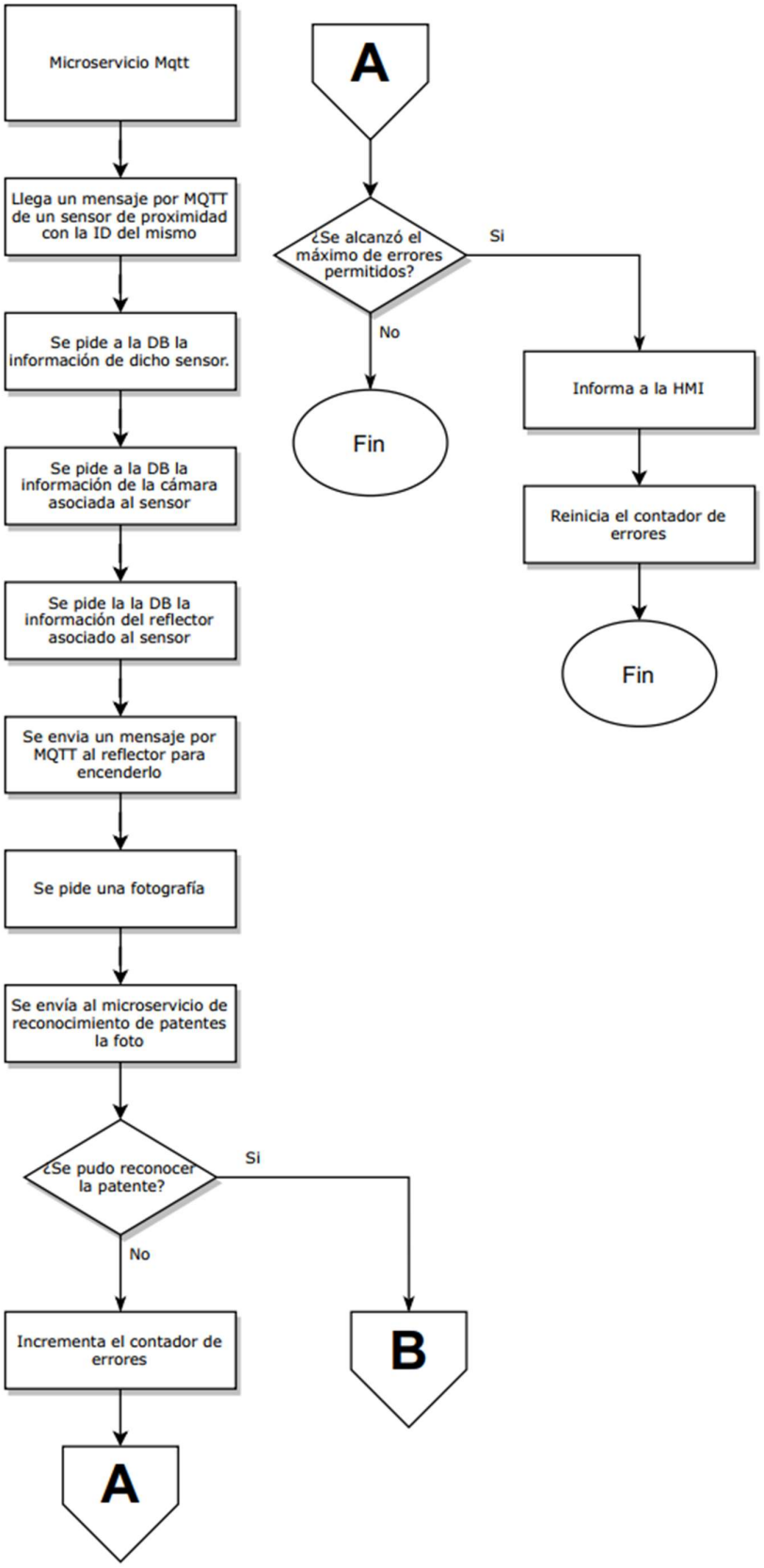


Figura 44: Esquema Microservicio central – parte 1.

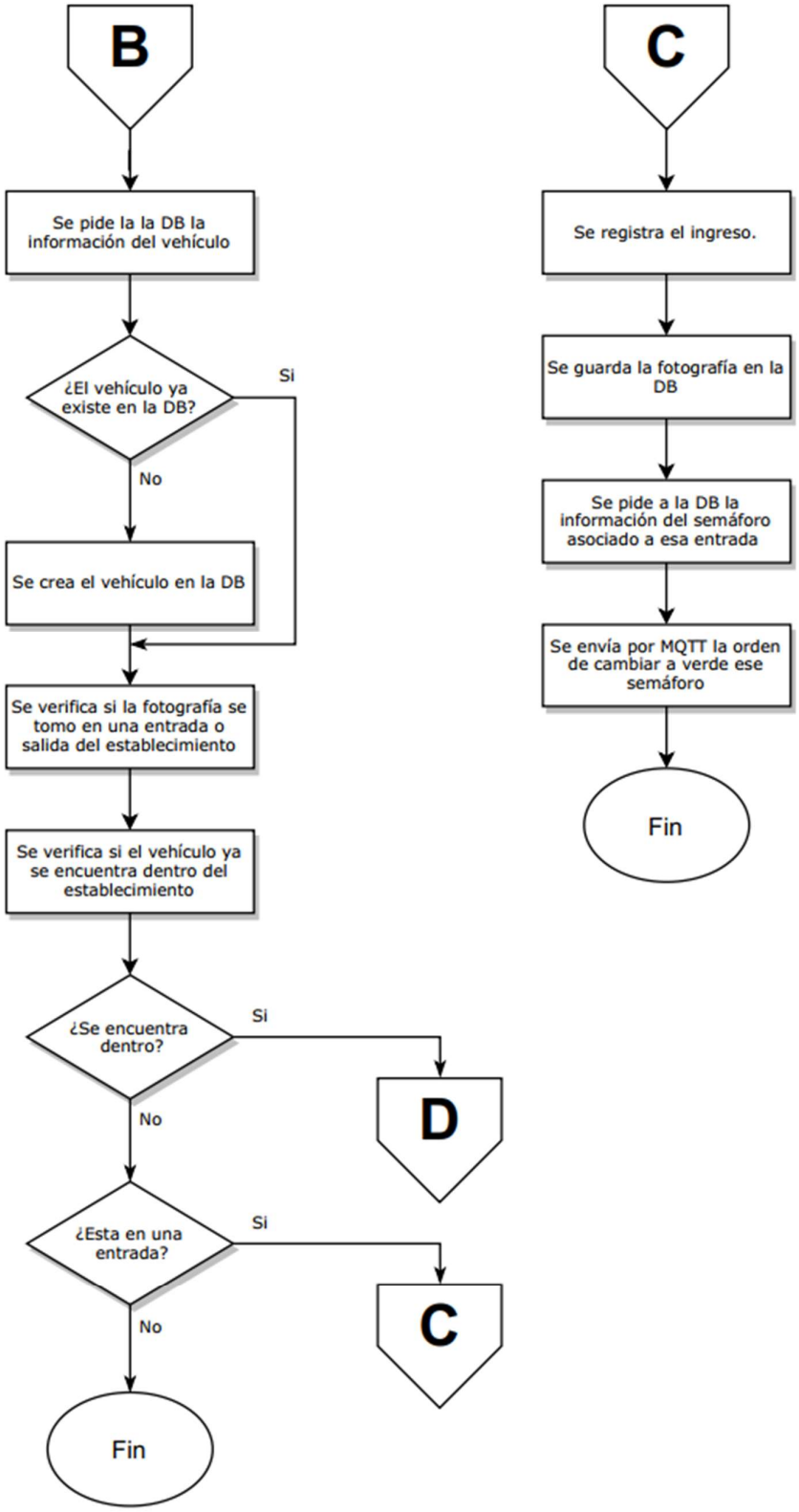


Figura 45: Esquema Microservicio central – parte 2.

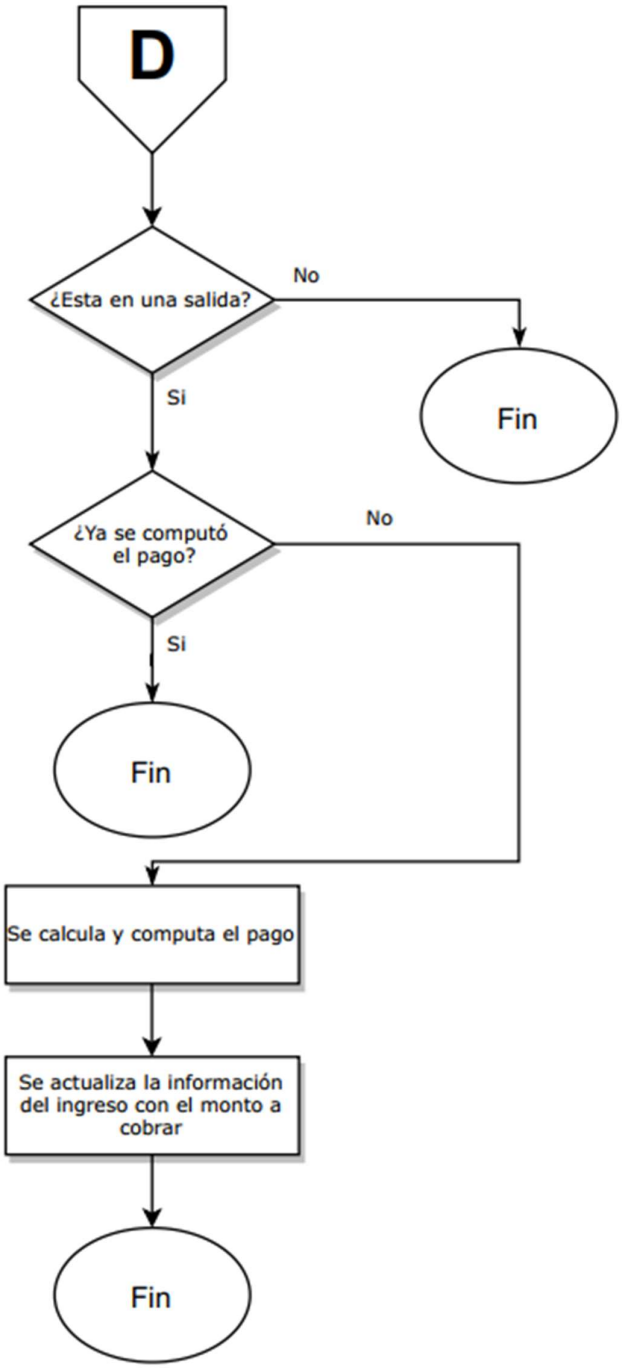


Figura 46: Esquema Microservicio central – parte 3.

2.2.6 Cartel de entrada

Este servicio tiene como objetivo servir de referencia visual para conductores que deseen ingresar al establecimiento. A través de letras y números grandes, se busca indicar rápidamente el estado de ocupación del establecimiento. Además, se utilizaron colores que normalmente se asocian con la idea de que hay lugar (verde) o que no se puede ingresar (rojo). Esta pantalla se diseñó para correr sobre una Raspberry Pi y se optimizó para ser mostrada en televisores de al menos 32”.

Diseñado con tecnologías web, más precisamente Vue.Js[38] con el Framework Quasar [38]. Se utiliza Web Sockets para recibir en tiempo real cuando hay un cambio en el nivel de ocupación del establecimiento.

Figura 47: Cartel de entrada cuando hay lugares disponibles en el estacionamiento.

Figura 48: Cartel de entrada cuando no hay lugares disponibles en el estacionamiento.

2.2.7 HMI principal

Esta se diseñó con el objetivo de resumir toda la información del sistema en un solo lugar y de una forma amigable e intuitiva. Programada en Vue Js con el framework Quasar y haciendo uso de los protocolos HTTP y WebSockets.
La pantalla principal es la siguiente:

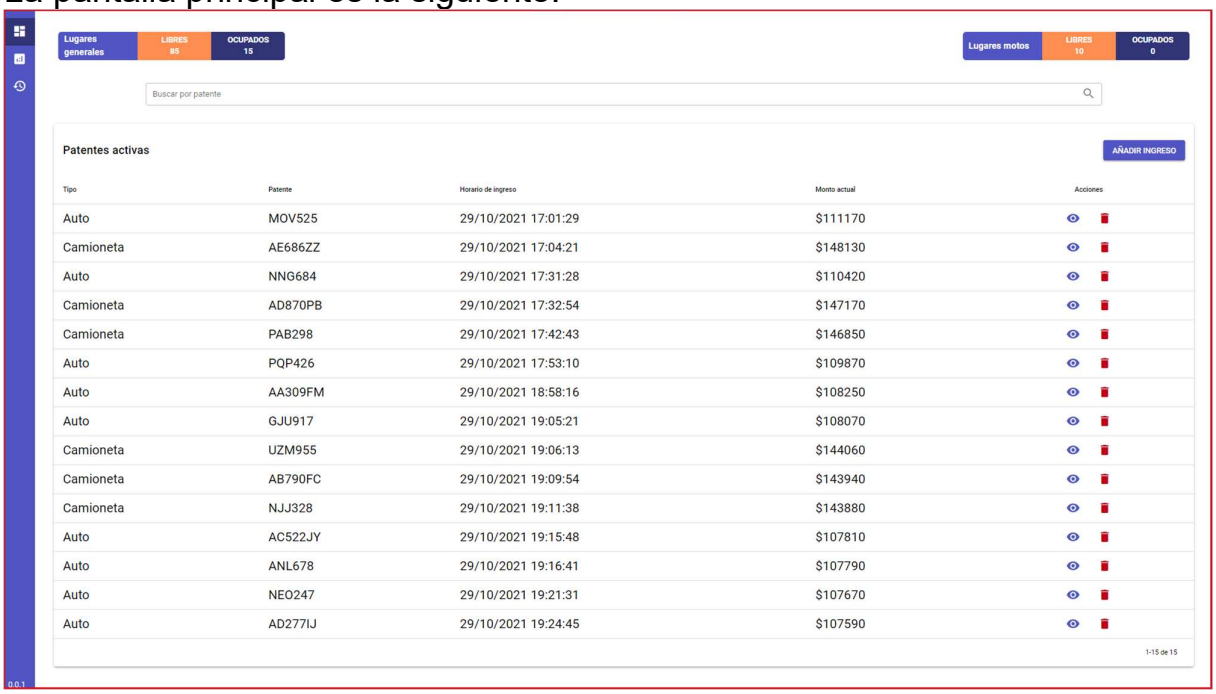


Figura 49: Pantalla: principal.

En la parte superior se observan los contadores de espacio en el establecimiento, donde a la izquierda se encuentra el contador para autos/camionetas y a la derecha se observa el contador de lugares para motocicletas. Se utilizó un esquema de colores que permite una rápida visualización de esta importante variable. Estos contadores se encuentran conectados por WebSockets al servicio de la base de datos, entonces cuando se registra un ingreso o egreso en el sistema, una señal es emitida por este protocolo para dar aviso de que debe realizarse un HTTP GET a la base de datos para acceder a la información actualizada.
Debajo se encuentra una barra de búsqueda que permite filtrar la información a mostrar por número de patente o bien por tipo de vehículo. Este componente es muy importante para la usabilidad ya que hace posible realizar una búsqueda rápida de cierto vehículo sin perderse entre todos los ingresos, que podrían llegar a ser un número elevado en función de las capacidades del establecimiento.

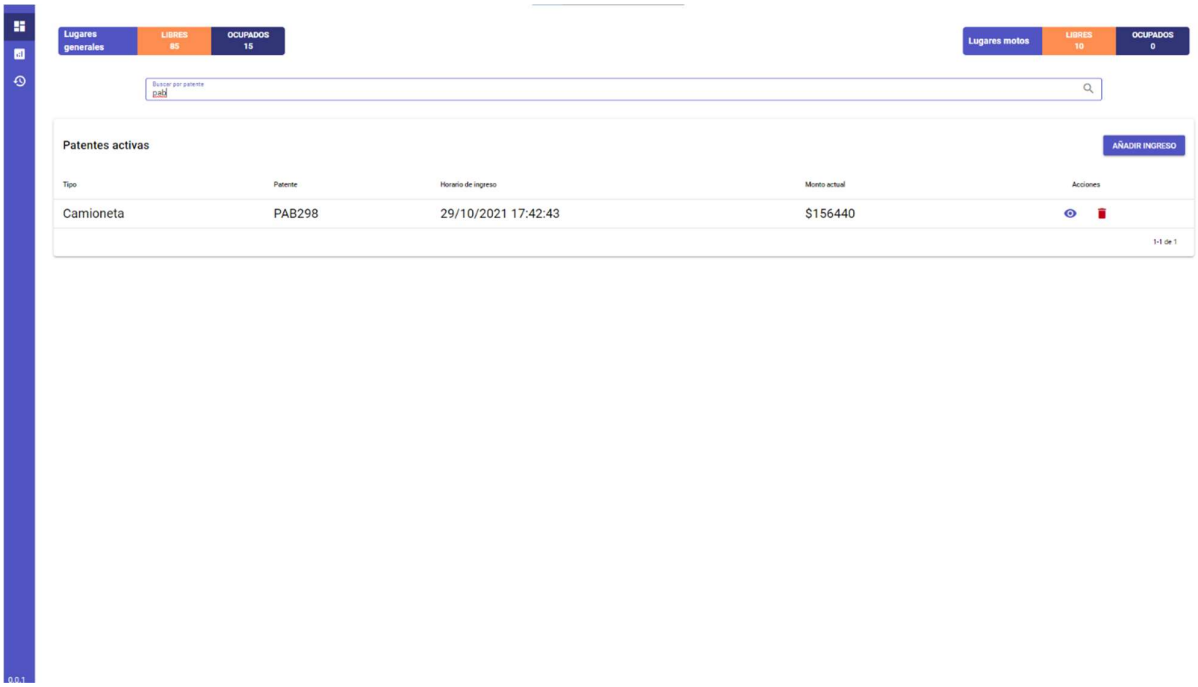


Figura 50: Filtrando tabla de ingresos con barra de búsqueda.

Posteriormente se tiene una tabla con la información principal del sistema. En ella se detallan todos los vehículos que actualmente están dentro del establecimiento, mostrando el tipo de vehículo (Auto, Camioneta o Motocicleta), el número de patente, la fecha y hora de ingreso, el monto a cobrar en función del tiempo de estadía, además de dos botones con acciones para cada registro, ver la foto con la que se ingresó al establecimiento y eliminar el registro.

Patentes activas

AÑADIR INGRESO

Tipo	Patente	Horario de ingreso	Monto actual	Acciones
Auto	MOV525	29/10/2021 17:01:29	$147040	
Camioneta	AE686ZZ	29/10/2021 17:04:21	$195950	
Auto	NNG684	29/10/2021 17:31:28	$146290	
Camioneta	AD870PB	29/10/2021 17:32:54	$195000	
Camioneta	PAB298	29/10/2021 17:42:43	$194670	
Auto	PQP426	29/10/2021 17:53:10	$145740	
Auto	AA309FM	29/10/2021 18:58:16	$144120	
Auto	GJU917	29/10/2021 19:05:21	$143940	
Camioneta	UZM955	29/10/2021 19:06:13	$191890	
Camioneta	AB790FC	29/10/2021 19:09:54	$191770	
Camioneta	NJJ328	29/10/2021 19:11:38	$191710	
Auto	AC522JY	29/10/2021 19:15:48	$143680	
Auto	ANL678	29/10/2021 19:16:41	$143660	
Auto	NEO247	29/10/2021 19:21:31	$143540	
Auto	AD277IJ	29/10/2021 19:24:45	$143450	

1-15 de 15

Figura 51: Tabla con los ingresos activos del establecimiento.

Si se selecciona para observar la fotografía del vehículo se despliega el siguiente componente.

Figura 52: Fotografía de un ingreso exitoso.

Otra página del sistema es el historial. En ella se muestran en formato de tabla todos los registros completos del sistema (ingreso y egreso con cobro). La información aquí presentada es:

- Tipo de vehículo.
- Número de patente.
- Fecha y horario de ingreso.
- Fecha y horario de egreso.
- Monto cobrado.
- Tipo de pago.
- Ver la foto de ingreso.

Además, se incluye un buscador por patente o tipo de vehículo, ya que puede resultar dificultoso encontrar un ingreso especifico entre todos los existentes.

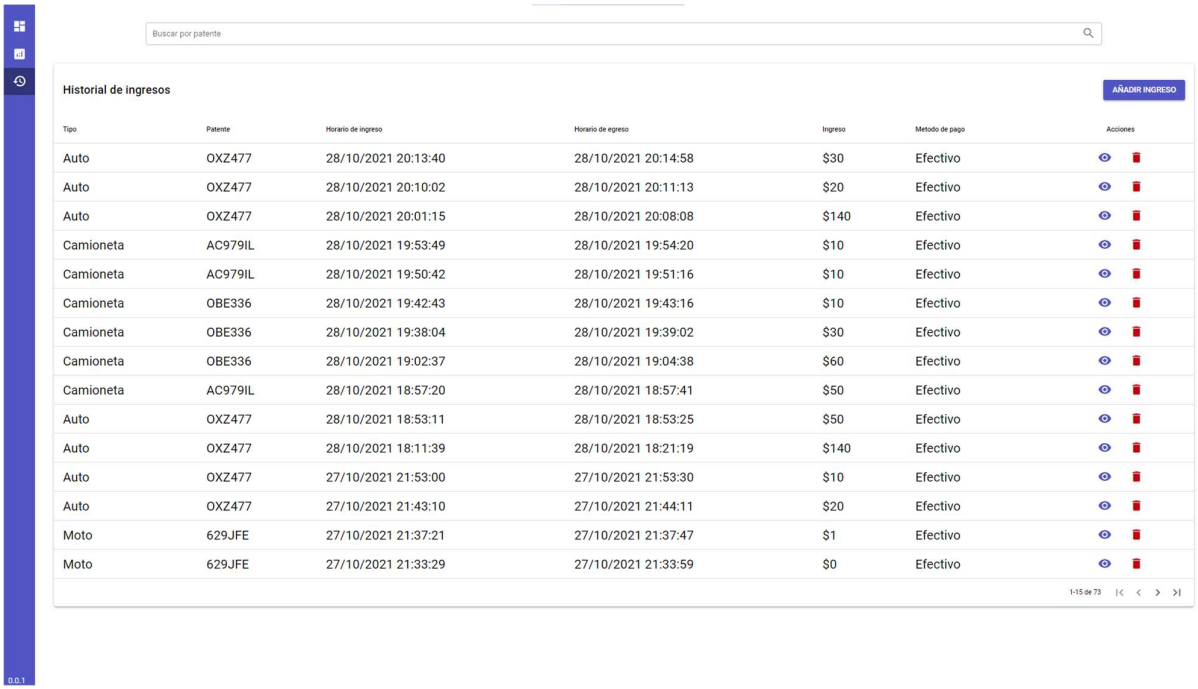


Figura 53: Pantalla: Historial.

La fotografía en esta pantalla es similar a la presentada en la pantalla principal.

Figura 54: Reconociendo una camioneta de noche.

Se puede observar cómo existen registros para distintos tipos de vehículos.

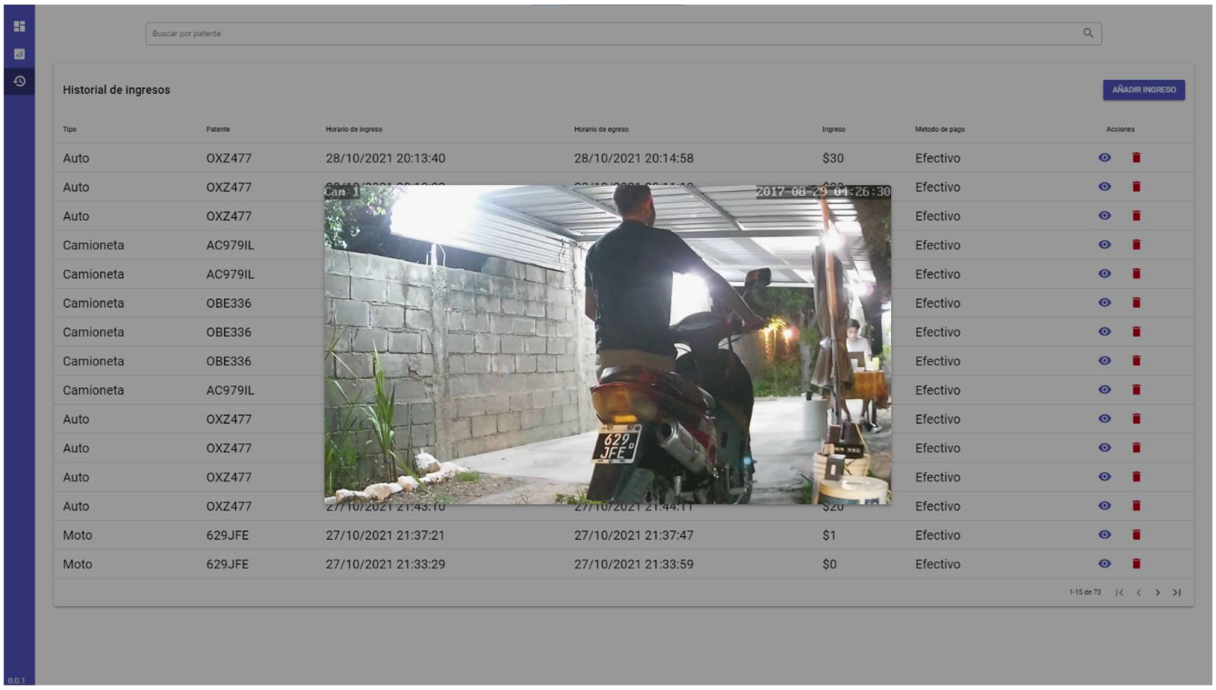


Figura 55: Reconociendo una motocicleta.

La siguiente pantalla es la de gráficos estadísticos. Estos son útiles para conocer como fue el desempeño del sistema y del establecimiento en un periodo determinado de tiempo.

Figura 56: Pantalla: Gráficos.

Se conformaron 3 histogramas con informaciones distintas. El primero muestra la recaudación total por día, considerando todos los tipos de pagos.

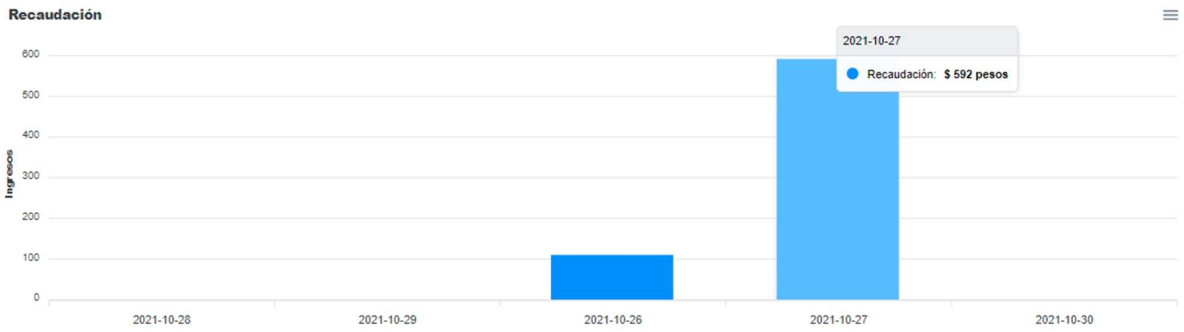


Figura 57: Gráfico de recaudación, por día.

El segundo gráfico muestra el nivel de ocupación por día, distinguiendo por tipo de vehículos, es decir, autos, camionetas y motocicletas.

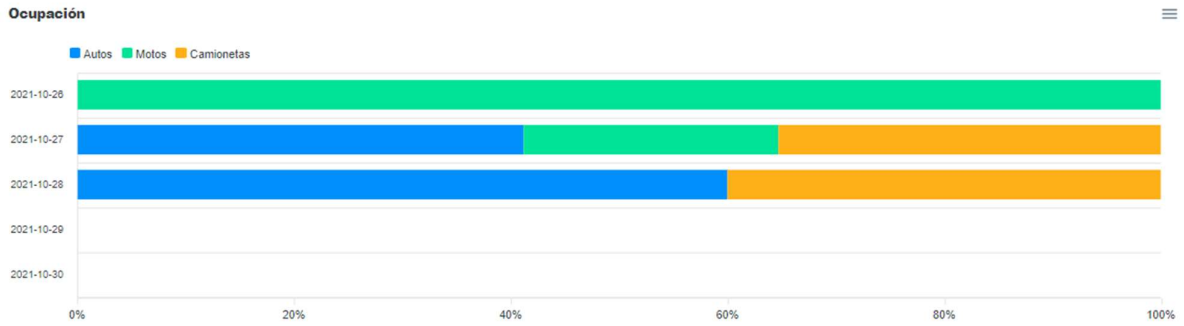


Figura 58: Gráfico de ocupación por día.

Por último, el tercer gráfico muestra la distribución de los tipos de pagos por día, distinguiendo entre efectivo, débito y tarjeta de crédito.

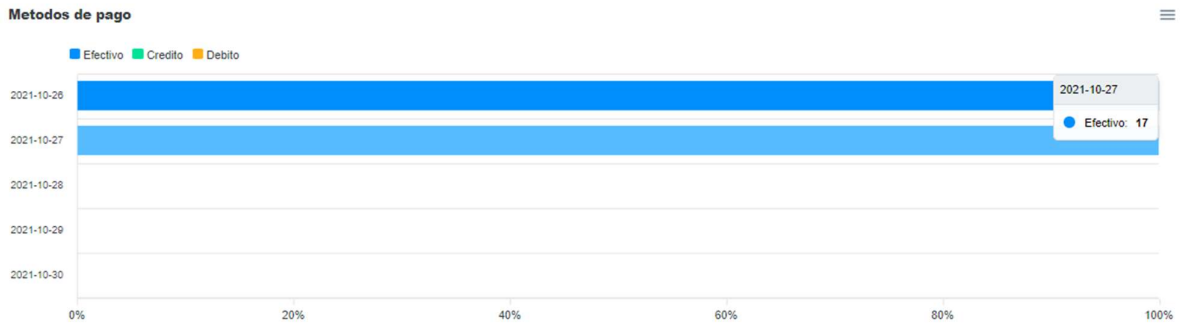


Figura 59: Gráfico de distribución de métodos de pagos por día.

Cabe mencionar que además de mostrar el valor acumulado de cada histograma, se muestra la distribución porcentual de los mismos.

Otra de las características de esta pantalla es que se permite exportar estos gráficos, en los formatos PNG, SVG y CSV.

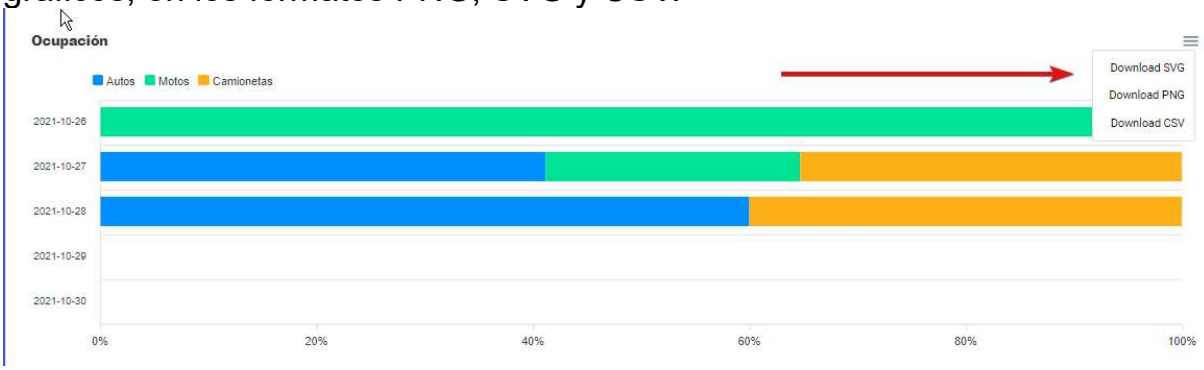


Figura 60: Formatos para exportar gráfico.

Eligiendo el formato CSV, es posible analizar posteriormente la información en un software de hoja de cálculo como Microsoft Excel.

	A	B	C	D	E
1	category	Autos	Motos	Camionetas	
2	26/10/2021	0	3	0	
3	27/10/2021	7	4	6	
4	28/10/2021	9	0	6	
5	29/10/2021	0	0	0	
6	30/10/2021	0	0	0	
7					
8					

Figura 61: Información de un gráfico como CSV.

Otra de las características del sistema es la capacidad de mostrar alarmas en la pantalla principal (que se espera que utilice la mayor parte del tiempo).
En las siguientes imágenes se observan los 3 niveles de alerta definidos:

- Error en color rojo.
- Advertencia en color amarillo.
- Información en color celeste.

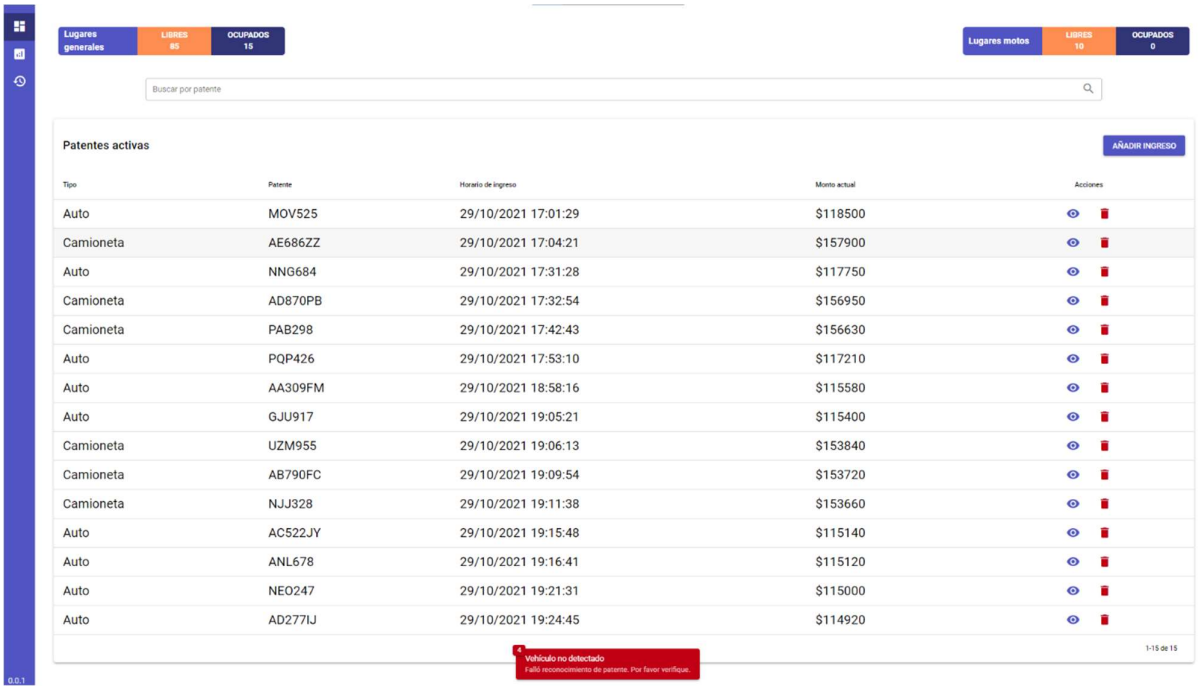


Figura 62: Alerta: Error.

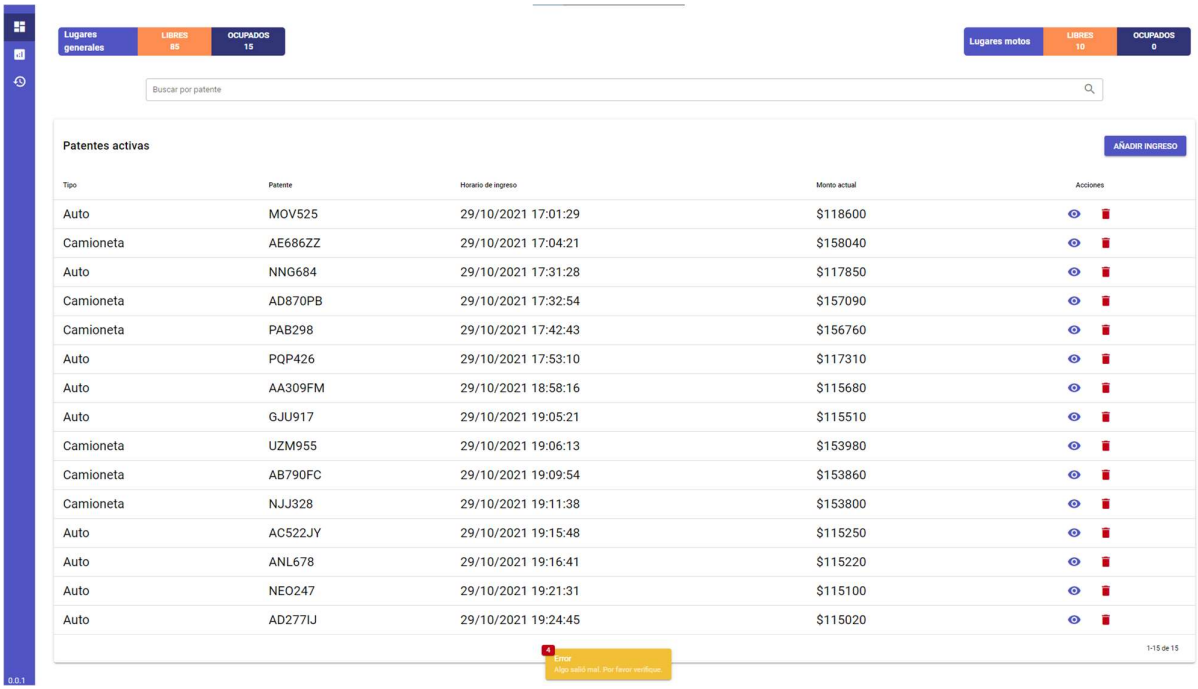


Figura 63: Alerta: Advertencia.

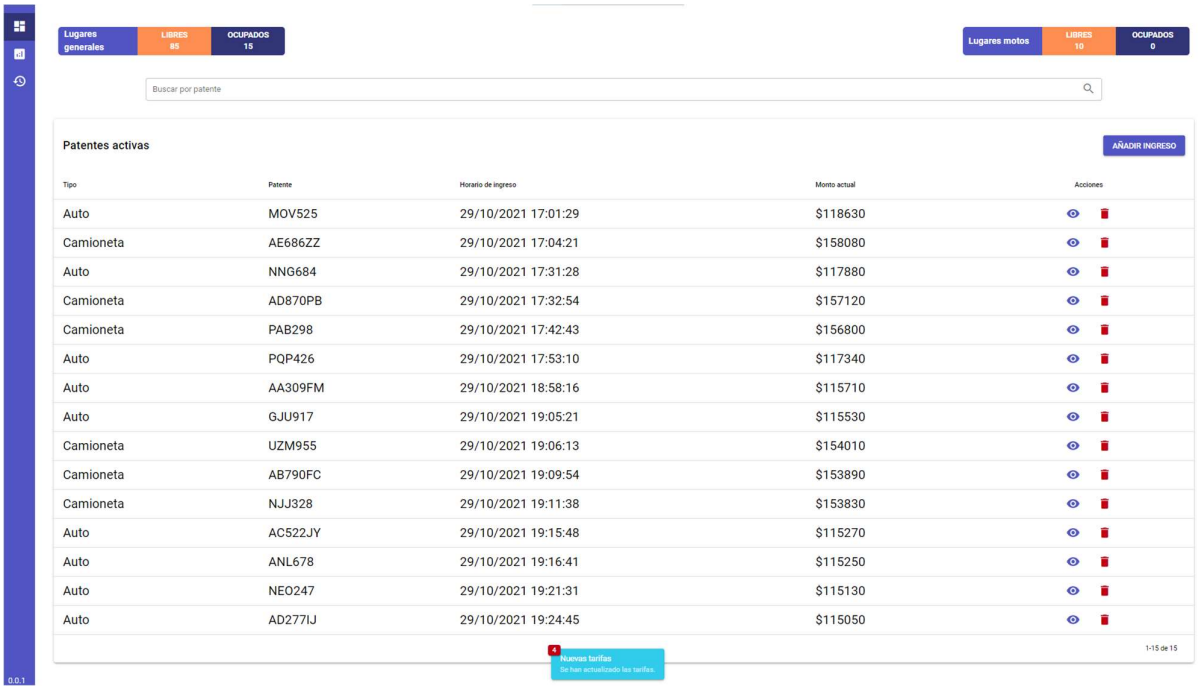


Figura 64: Alerta: Información.

En el caso de desear eliminar una entrada, independientemente del motivo, se muestran un cartel esperando la confirmación. Esto es para evitar eliminaciones erróneas.

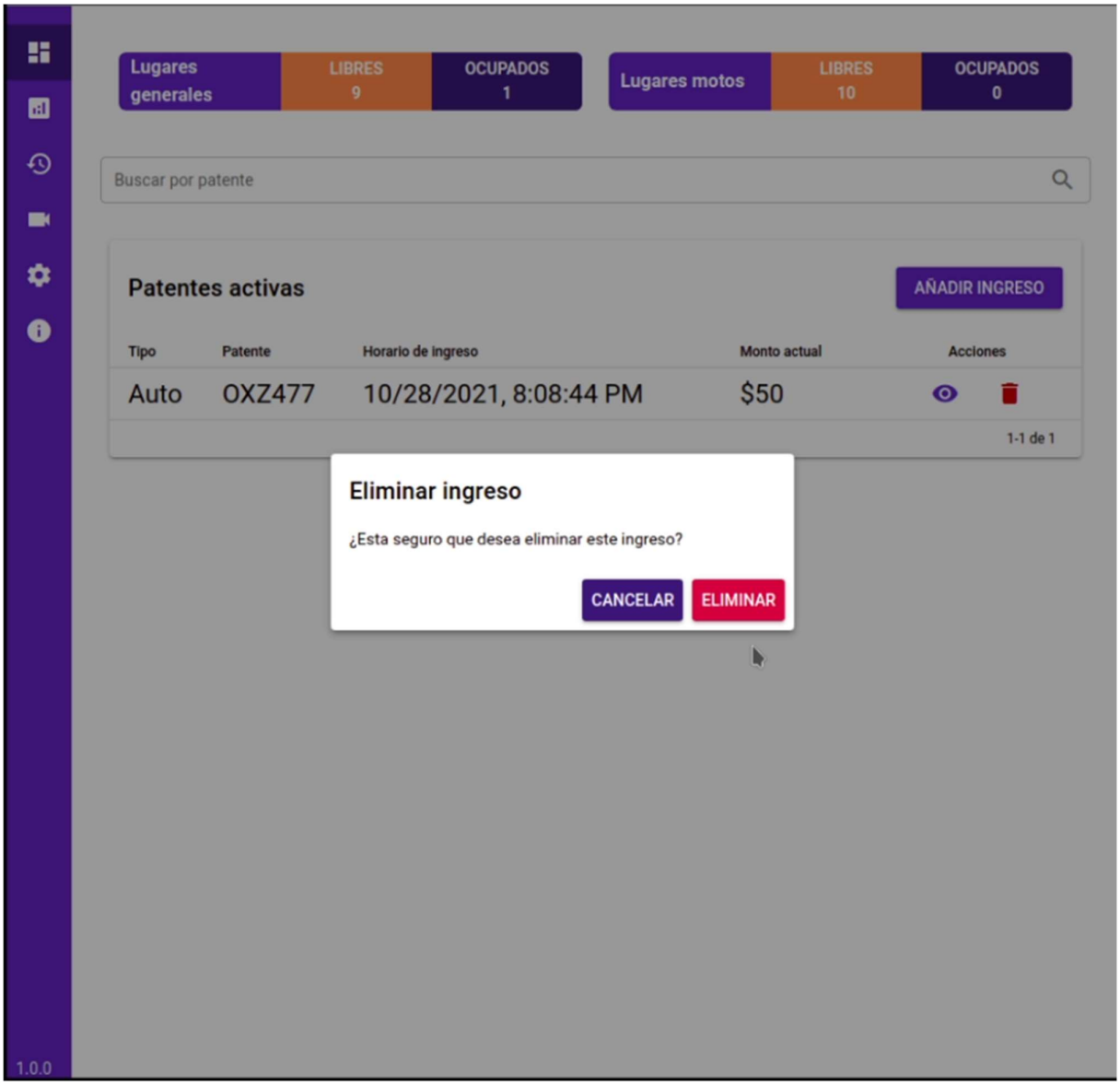


Figura 65: ¿Esta seguro que desea eliminar un ingreso?

Si se acepta la eliminación, un mensaje con la confirmación es mostrado.

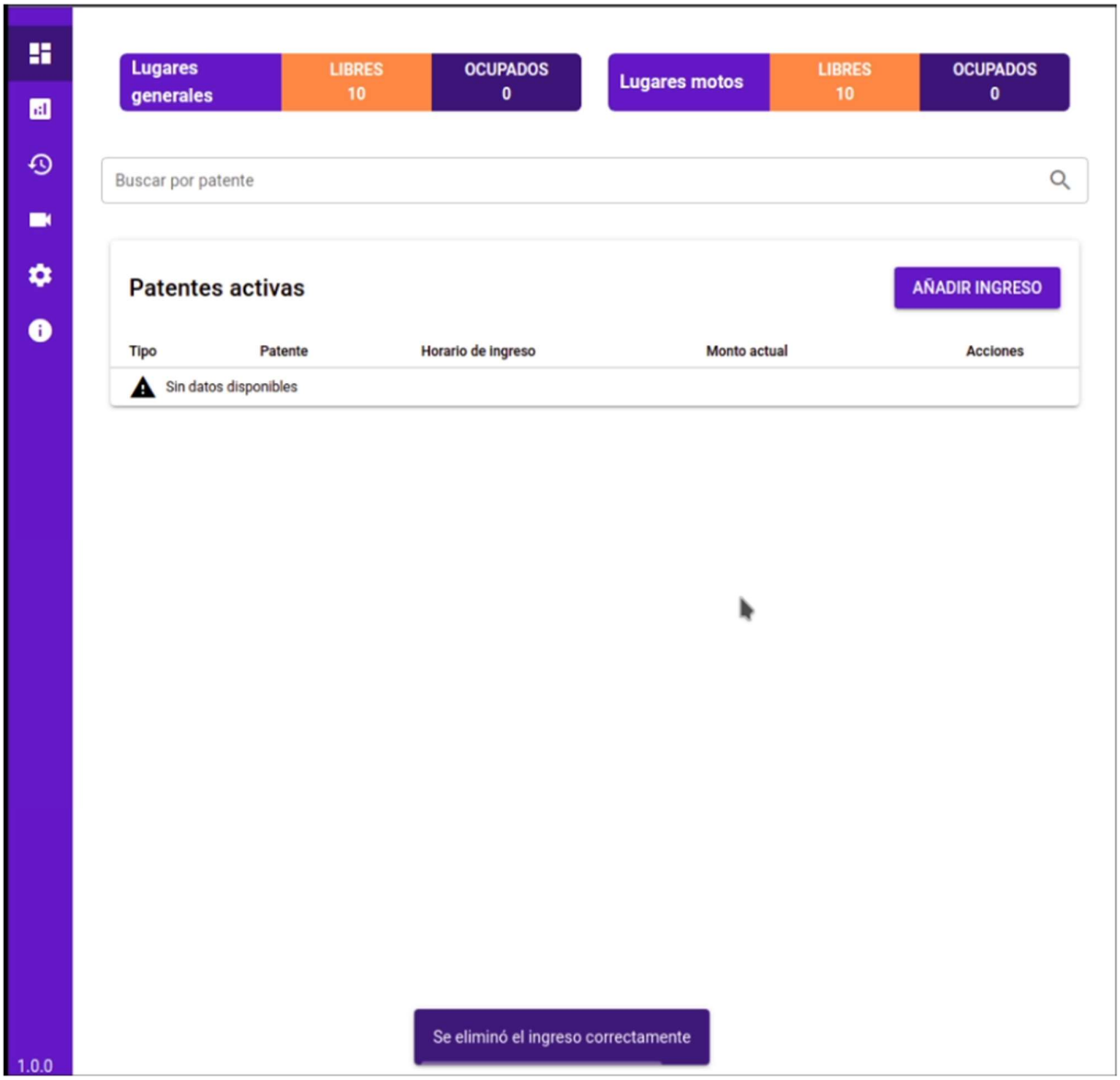


Figura 66: Confirmación de que una entrada fue eliminada correctamente.

Por último, se muestra la forma de ejecutar un cobro y realizar un egreso. Cuando un vehículo se posiciona en una salida, y sus datos son leídos correctamente, se envía a la HMI la señal para que un operario realice el cobro. A la imagen a continuación se ve la alerta que es emitida cuando sucede lo antes descripto.

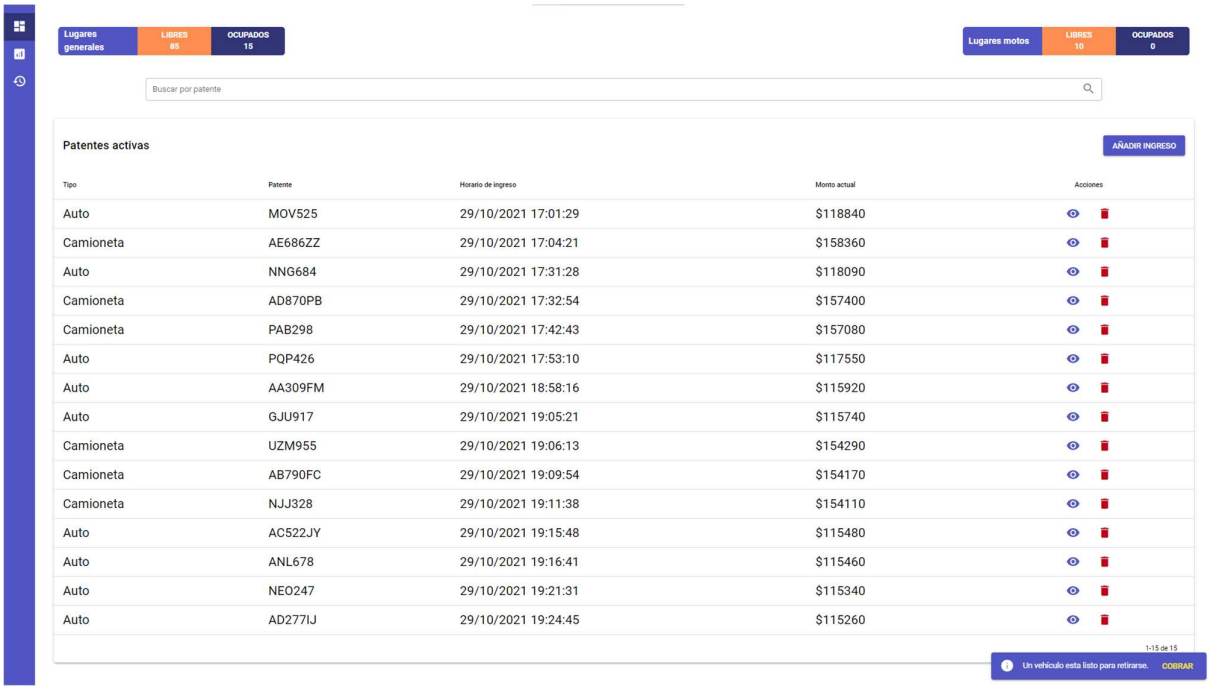


Figura 67: Alerta de que un vehículo está listo para retirarse del establecimiento.

Si se presiona sobre esta alerta, se abre el menú para finalizar el cobro. En este se muestra la imagen tomada cuando el vehículo ingresó al establecimiento, así como también los siguientes datos:
- Número de patente.
- Tipo de vehículo (auto, camioneta o motocicleta).
- Método de pago (efectivo, débito o tarjeta de crédito).
- Monto a cobrar.
- Monto con el que el usuario paga. Esto es por ejemplo si el monto a cobrar es $82, y el usuario paga con un billete de $100, en pantalla se mostrará que se debe entregar un vuelto de $18.

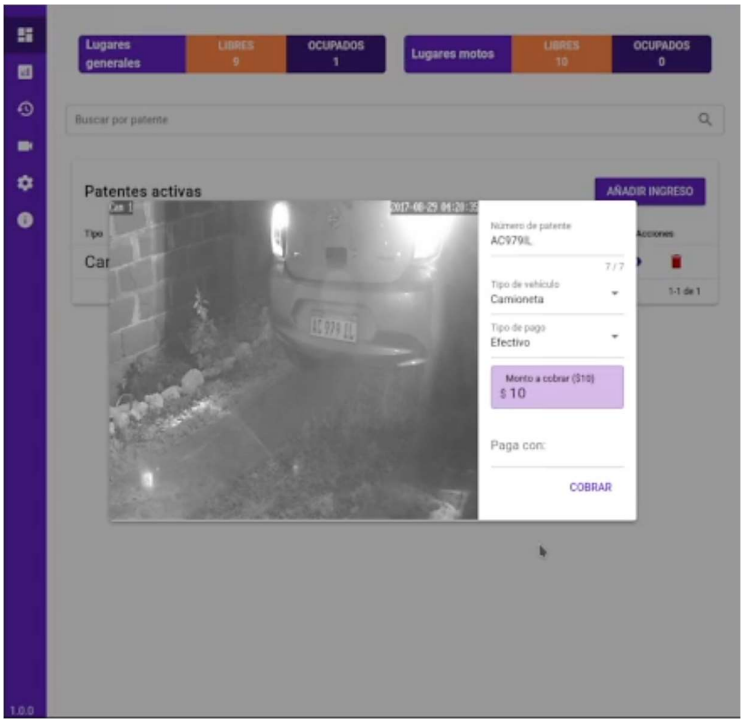


Figura 68: Menú de cobro.

Cabe destacar que todos los campos en este menú son editables, es decir, que, si bien el sistema sugiere un monto a cobrar, o un número de patente, el operario a cargo de la aplicación podría decidir alterar estos valores por unos más convenientes.

Para finalizar, se consideró la posibilidad de controlar esta HMI desde un dispositivo celular. Esto es muy importante por temas de accesibilidad y compatibilidad. A continuación, se muestran unas imágenes de la pantalla principal vista desde un celular.

Figura 69: Pantalla principal, desde un celular.

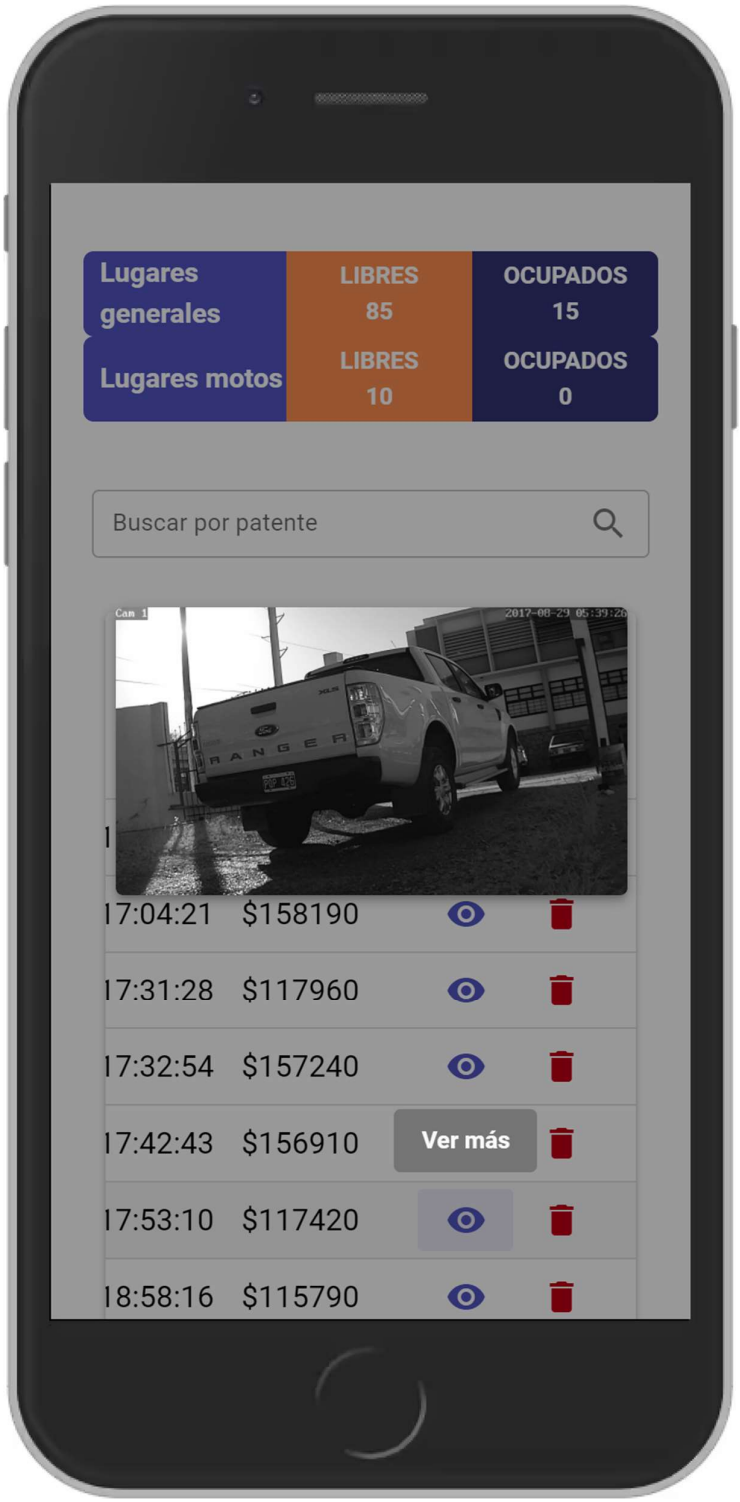


Figura 70: Fotografía de una de las entradas, desde un celular.

Figura 71: Vista de la pantalla principal, desde un celular inclinado.

2.3 Análisis completo

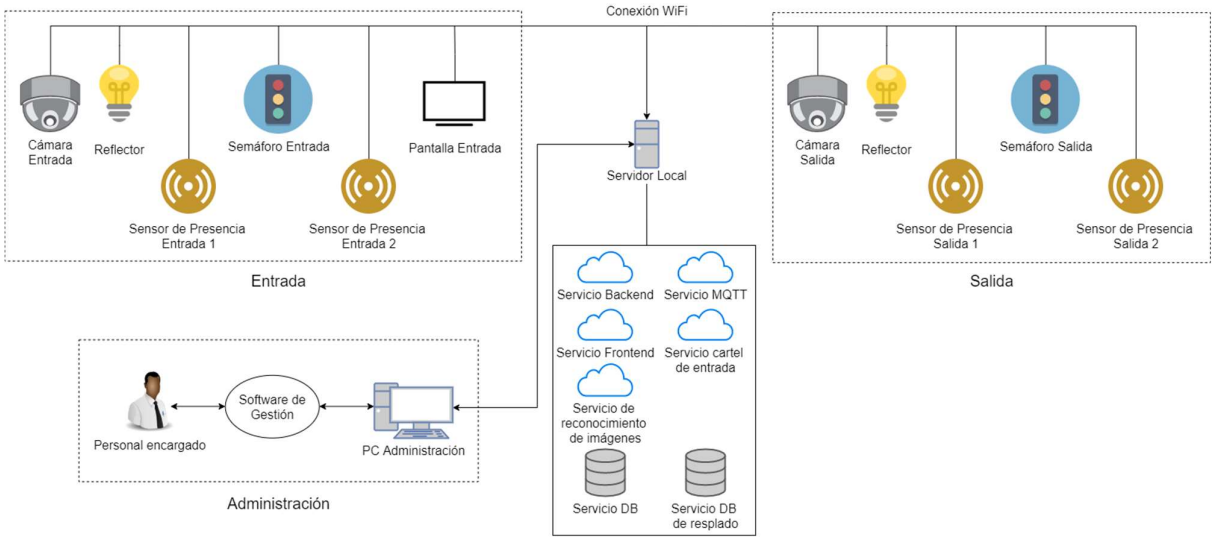


Figura 72: Diagrama del sistema completo.

En la imagen previa se puede apreciar el sistema en su totalidad.
A continuación, se explicarán los diferentes bloques del diagrama y como estos funcionan en conjunto.

2.3.1 Entrada

El grupo de entrada se encuentra formado por una cámara IP, un reflector para la cámara, un semáforo de entrada, dos sensores de presencia y una pantalla de entrada. Todos los anteriores, a excepción de la pantalla de entrada, son controlados por el sistema MQTT central.

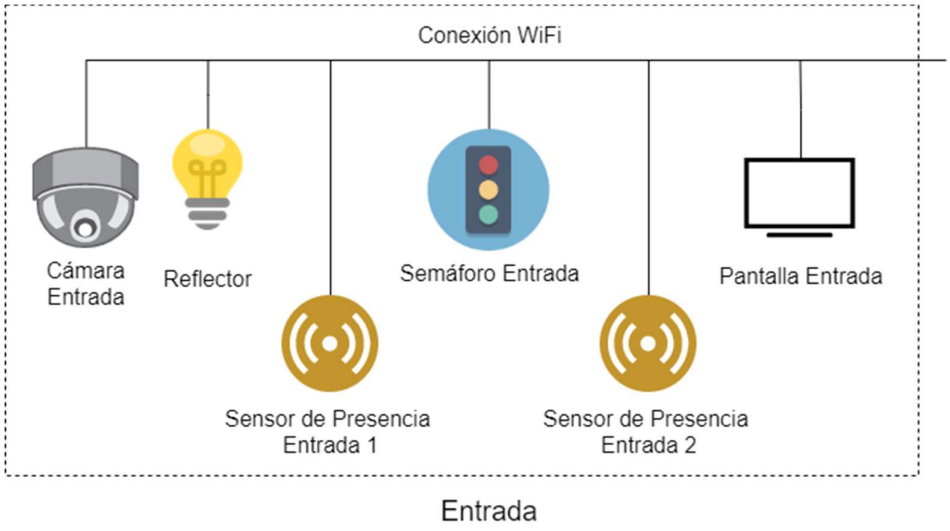
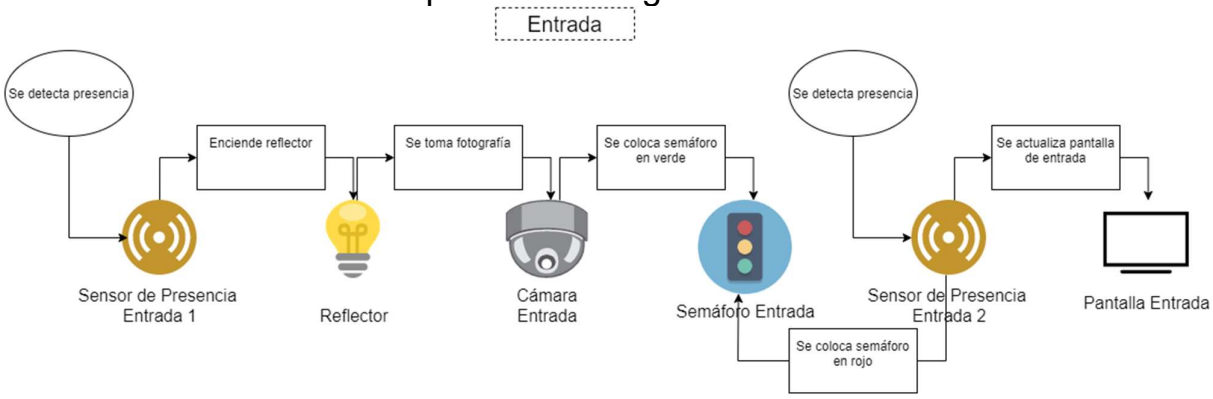


Figura 73: Diagrama del sistema de entrada.

Sin considerar la influencia del servicio MQTT central, la lógica de funcionamiento de esta etapa es como sigue:



Esta etapa puede analizarse en dos partes, definidas por la detección de presencia en los sensores de ultrasonido. Cuando el primer sensor de presencia de la entrada detecta algo, se envía la orden para encender el reflector. Una vez que sucede esto se toma una fotografía con la cámara asignada a este sector y si se reconoce con éxito la patente se coloca el semáforo en verde. La segunda parte comienza cuando el segundo sensor de presencia de la entrada detecta algo, entonces luego de un tiempo envía la orden para colocar el semáforo en rojo y se actualiza el contador de la pantalla de entrada en caso de que la entrada haya resultado exitosa.

2.3.2 Salida

El grupo de salida se encuentra formado por una cámara IP, un reflector para la cámara, un semáforo de salida y dos sensores de presencia. Todos los anteriores son controlados por el sistema MQTT central.

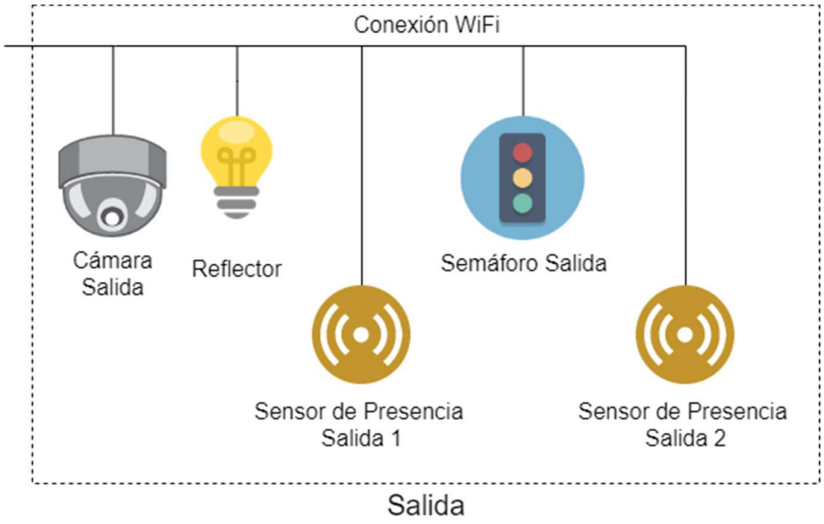
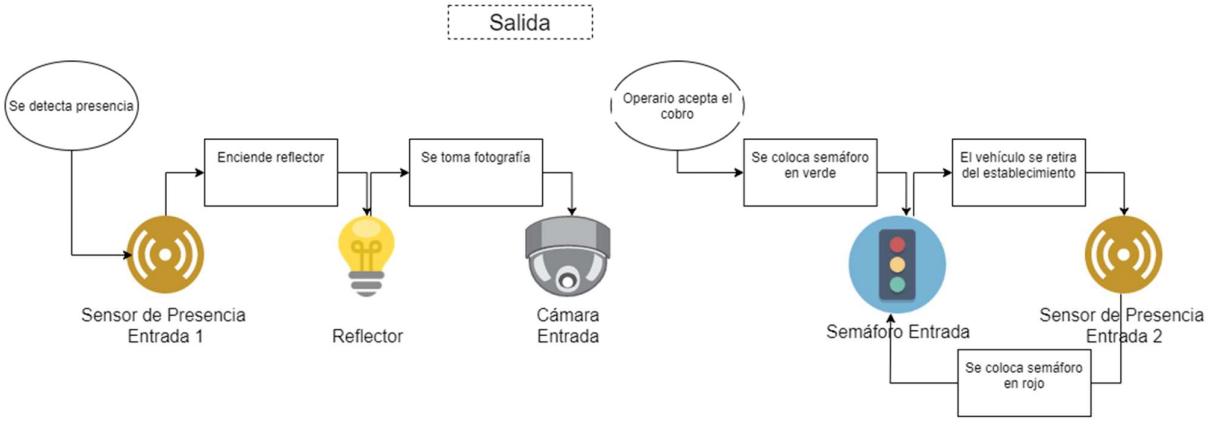


Figura 74: Diagrama del sistema de salida.

Sin considerar la influencia del servicio MQTT central, la lógica de funcionamiento de esta etapa es como sigue:



Esta etapa es similar a la de entrada, con leves diferencias. Esta etapa puede analizarse en dos partes, la primera definida por la detección de presencia en el sensor de ultrasonido y la segunda definida por el accionar del operario encargado de la HMI. Cuando el primer sensor de presencia de la salida detecta algo, se envía la orden para encender el reflector. Una vez que sucede esto se toma una fotografía con la cámara asignada a este sector y si se reconoce con éxito la patente se coloca una alerta en la HMI. La segunda parte comienza cuando el operario encargado de la aplicación acepta la petición de

cobro, entonces se coloca el semáforo en verde y se permite que el vehículo se retire del establecimiento. Luego, el segundo sensor de presencia de la salida detecta algo lo cual inicia un contador que luego de un tiempo envía la orden para colocar el semáforo en rojo.

2.3.3 Administración

Desde un punto de vista de la administración del sistema, el operario solo tiene una tarea que es la de realizar el cobro del monto acorde al tiempo que el vehículo permaneció en el establecimiento, el cual es calculado al momento de la salida.

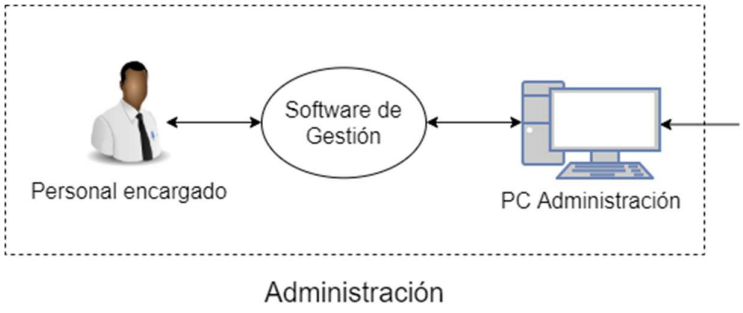


Figura 75: Diagrama del sistema de administración.

2.3.4 Servidor Local

Se considera como servidor local aquel que corre Docker y todos sus contenedores. Estos son todos los microservicios descritos en la sección anterior.

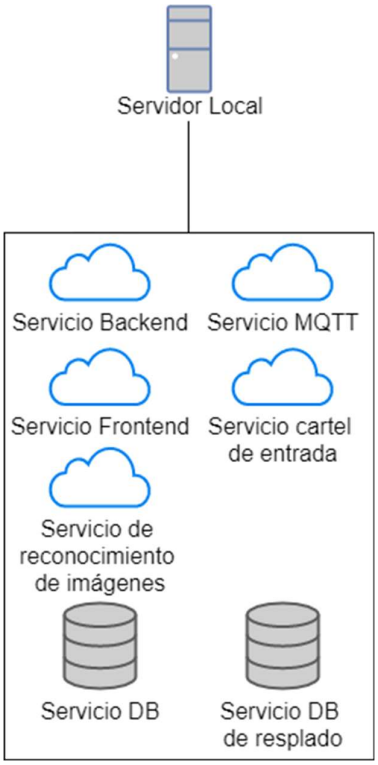


Figura 76: Diagrama del servidor.

Tal como ya se mencionó, Docker se encarga de realizar una red virtual que hace posible que cada microservicio se comunique entre sí. Para comunicar estos microservicios se utilizan los protocolos HTTP y WebSocket.

En la siguiente tabla se resume los protocolos de comunicación que se utilizan entre cada microservicio. Los cuadros negros implican que no hay comunicación directa.

	Servicio DB	Servicio MQTT	Servicio Reconocimiento de imágenes	Servicio Backend (foto)	Servicio Frontend	Servicio cartel de entrada
Servicio DB		WebSocket, HTTP	HTTP	HTPP	WebSocket, HTTP	WebSocket, HTTP
Servicio MQTT	WebSocket, HTTP		HTPP	HTPP	HTTP	
Servicio Reconocimiento de imágenes	HTTP	HTTP				
Servicio Backend (foto)	HTTP	HTTP				
Servicio Frontend	WebSocket, HTTP	HTTP				
Servicio cartel de entrada	WebSocket, HTTP					

Tabla 19: Protocolos de comunicación entre microservicios

Al comienzo, cada uno de los servicios le solicita al servicio de la DB la información que requiere para funcionar. Una vez que todos se aprovisionaron, todos quedan a la espera que de que se les solicite que se ejecuten, a excepción del servicio de la base de datos que a intervalos de tiempos regulares actualiza la tabla de costos y reporta esta información al servicio de Frontend (HMI) a través de WebSocket.

Luego, cada servicio comienza a realizar su función una vez que el servicio MQTT se lo solicita, y todo esto se desencadena una vez que alguno de los sensores de presencia detecta proximidad.

Una parte importante del sistema que permite que cada una de las partes se comuniquen es la adición de un router WiFi. Ya que todas las placas realizadas utilizan como microcontrolador al ESP8266, se utiliza WiFi para comunicarlas con el servidor. Lo mismo sucede con la pantalla de entrada que utiliza una Raspberry Pi.

Capitulo 3: Resultados

Sin duda se obtuvo un producto de gran calidad y prestaciones, competente con otras opciones disponibles en el mercado actual.
Se logró cumplir satisfactoriamente el objetivo principal del proyecto, diseñar un sistema moderno, escalable, modular y económico que permitiese registrar ingreso y egreso de vehículos en un establecimiento.

Uno de los puntos cruciales del proyecto fue el reconocimiento de imágenes, el cual se pudo realizar con un alto nivel de precisión para automóviles y camionetas. Sin embargo, en cuanto al reconocimiento de patentes de motocicletas, el porcentaje de aciertos fue menor al esperado. Esto se debió principalmente a dos motivos:

- El modelo de aprendizaje profundo encargado de detectar la ubicación de la patente tiene un porcentaje de error mayor para las motocicletas, debido a que estas tienen patentes con un factor de forma diferente.

- El algoritmo utilizado para reconocimiento de caracteres en las patentes de motocicletas no utiliza un modelo destinado específicamente para este fin, por lo tanto, en ocasiones detecta caracteres adicionales que provocan que no se puedan extraer correctamente los datos de la patente.

Una solución para ambos problemas podría ser obtener un gran conjunto de datos de patentes de automóviles y motocicletas argentinas (modelos nuevos y viejos) con el fin de utilizarlo para reentrenar los modelos de detección de ubicación y número de patente.

No obstante, el reconocimiento se desempeña de forma adecuada independientemente de los niveles de luz ambiente. En las siguientes imágenes se observa cómo pudo ser leída la patente de un vehículo con luz solar de frente a la cámara y otro ya sin luz solar.

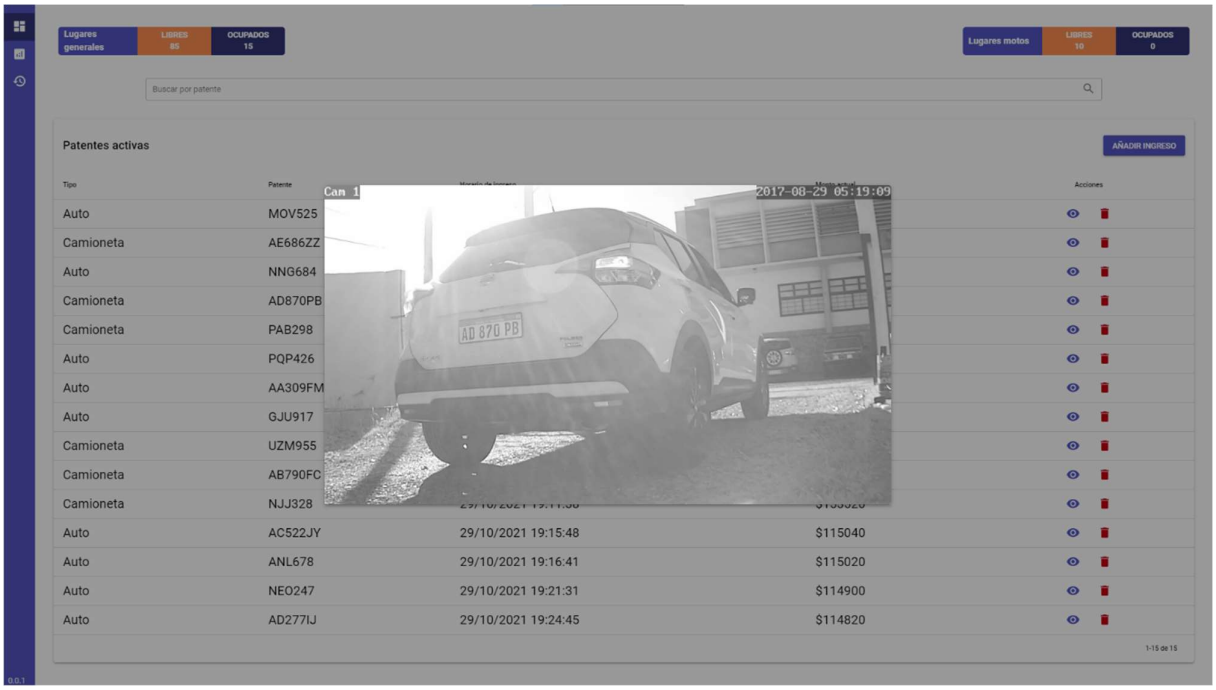


Figura 77: Fotografía tomada de día

Figura 78: Fotografía tomada de noche.

Los dispositivos detectores de presencia con ultrasonidos funcionan correctamente al momento de detectar vehículos cercanos. No obstante, la distancia máxima que se les puede asignar a estos no pude superar los 130cm, ya que, de hacerlo, es posible que sensores de diferentes dispositivos se

disparen entre si provocando falsos positivos. Para solventar este problema podrían utilizarse sensores ultrasónicos con un menor ángulo de apertura o un sensor que haga uso de otro tipo de tecnología, como por ejemplo uno láser.

Otra posibilidad que se evaluó fue implementación del servidor en un sistema embebido de alto rendimiento, como podría ser una Raspberry Pi. Sin embargo, solamente el consumo de memoria del microservicio de detección de vehículos descarta esta posibilidad en el sistema actual. Podría verse la posibilidad de adaptar los modelos de aprendizaje profundo en busca de un menor consumo de memoria.

Capitulo 4: Análisis de Costos

El siguiente análisis se realizó teniendo en cuenta teniendo en cuenta la disponibilidad de los componentes y buscando los mejores precios, por el hecho de que se plantea un equipo económico y accesible, que permita una rápida inserción en el mercado.

El costo aquí descripto es para la fabricación equipos con los mismos componentes utilizados en las pruebas. En el caso de un diseño final para un producto comercial podría reemplazarse algún componente por otro de mejor calidad, diseñar cajas y placas más pequeñas con componentes electrónicos de montaje superficial. Evidentemente estos factores podrían afectar el coste final de producción.

Para la implementación de un sistema idéntico al utilizado durante las pruebas se suponen los siguientes costos fijos:

						Precio dolar	99.7
Item	Sub Item	Cantidad	Descripcio	Precio unitario $ARS	subtotal $ARS	Precio unitario $USD	Subtotal $USD
Cámara		2		$ 5,000	$ 10,000	$ 50.15	$ 100.30
Sensor de presencia					$ -	$ -	$ -
	ESP8266	4		$ 650	$ 2,600	$ 6.52	$ 26.08
	Sensores ultrasónicos	8		$ 250	$ 2,000	$ 2.51	$ 20.06
	PCB	4		$ 65	$ 260	$ 0.65	$ 2.61
	Caja	4		$ 300	$ 1,200	$ 3.01	$ 12.04
	Fuente externa	4		$ 800	$ 3,200	$ 8.02	$ 32.10
	Bornera	4		$ 25	$ 100	$ 0.25	$ 1.00
	Conector Jack	4		$ 50	$ 200	$ 0.50	$ 2.01
	Mosfet 2N7000	8		$ 25	$ 200	$ 0.25	$ 2.01
	Resistencia	36		$ 5	$ 180	$ 0.05	$ 1.81
	Pulsador	4		$ 20	$ 80	$ 0.20	$ 0.80
	Tira de pines	2		$ 140	$ 280	$ 1.40	$ 2.81
	Cable Dupont x40	1		$ 200	$ 200	$ 2.01	$ 2.01
	Tornillo	32		$ 6	$ 192	$ 0.06	$ 1.93
	Tuerca	32		$ 4	$ 128	$ 0.04	$ 1.28
Reflector					$ -	$ -	$ -
	ESP8266	2		$ 650	$ 1,300	$ 6.52	$ 13.04
	Reflector	2		$ 650	$ 1,300	$ 6.52	$ 13.04
	PCB	2		$ 65	$ 130	$ 0.65	$ 1.30
	Caja	2		$ 300	$ 600	$ 3.01	$ 6.02
	MOC3020	2		$ 60	$ 120	$ 0.60	$ 1.20
	BT139	2		$ 125	$ 250	$ 1.25	$ 2.51
	Fuente	2		$ 465	$ 930	$ 4.66	$ 9.33
	Bornera	4		$ 25	$ 100	$ 0.25	$ 1.00
	Resistencia	6		$ 5	$ 30	$ 0.05	$ 0.30
	Pulsador	2		$ 20	$ 40	$ 0.20	$ 0.40
	Tornillo	14		$ 6	$ 84	$ 0.06	$ 0.84
	Tuerca	14		$ 4	$ 56	$ 0.04	$ 0.56
Semáforo					$ -	$ -	$ -
	ESP8266	2		$ 650	$ 1,300	$ 6.52	$ 13.04
	Foco Led 2W	4		$ 200	$ 800	$ 2.01	$ 8.02
	Portafoco	4		$ 100	$ 400	$ 1.00	$ 4.01
	PCB	2		$ 65	$ 130	$ 0.65	$ 1.30
	Caja	2		$ 300	$ 600	$ 3.01	$ 6.02
	Módulo Relé	2		$ 200	$ 400	$ 2.01	$ 4.01
	Fuente	2		$ 465	$ 930	$ 4.66	$ 9.33
	Bornera	4		$ 25	$ 100	$ 0.25	$ 1.00
	Resistencia	2		$ 5	$ 10	$ 0.05	$ 0.10
	Pulsador	2		$ 20	$ 40	$ 0.20	$ 0.40
	Cable	2		$ 30	$ 60	$ 0.30	$ 0.60
	Tornillo	10		$ 6	$ 60	$ 0.06	$ 0.60
	Tuerca	10		$ 4	$ 40	$ 0.04	$ 0.40
Cartel de Entrada					$ -	$ -	$ -
	Raspberry Pi	2		$ 14,000	$ 28,000	$ 140.42	$ 280.84
	TV 32 pulgadas	4		$ 30,000	$ 120,000	$ 300.90	$ 1,203.61
Router AP		1		$ 1,500	$ 1,500	$ 15.05	$ 15.05
PC Servidor	Intel i5, 16GB RAM, 1TB HDD	1	Solo CPU	$ 73,000	$ 73,000	$ 732.20	$ 732.20
PC Administración	Intel Pentium, 4GB RAM, 240GB SSD	1	CPU con monitor y periféricos	$ 65,000	$ 65,000	$ 651.96	$ 651.96
					$ 253,130		$ 2,538.92

Tabla 20: Costos fijos.

En las últimas dos columnas se muestran los precios en dólares americanos al valor de cotización oficial de $99.7 del día 31/10/2021.

Los costos variables se detallan a continuación:

						Precio dolar	99.7
Item	Sub Item	Cantidad	Descripció	Precio unitario $ARS	subtotal $ARS	Precio unitario $USD	Subtotal $USD
Cableado eléctrico			Por metro	$ 60	$ -	$ 0.60	$ -
Cable canal			Por metro	$ 100	$ -	$ 1.00	$ -
Trabajador		2	Por hora	$ 450	$ 900	$ 4.51	$ 9.03

Tabla 21: Costos variables.

En este caso se toman como costos variables los dados durante las pruebas y se descarta el costo del cableado. Se realizaron 400 horas de trabajo por persona, lo que da un total de 800 horas. Se consideraron las horas de planificación, diseño y pruebas. El precio de la hora se fijó en $450 pesos argentinos.

Se debe considerar que estos valores dependen de las dimensiones del establecimiento, ya que en función de esto será la cantidad de metros de cable a utilizar, por lo tanto, el costo real de implementación será ligeramente mayor.

Resumiendo, los costos fijos y variables en una misma tabla, se tiene lo siguiente:

						Precio dolar	99.7
Item	Sub Item	Cantidad	Horas	Precio unitario $ARS	subtotal $ARS	Precio unitario $USD	Subtotal $USD
Trabajador		2	400	$ 450	$ 360,000		$ 3,610.83
Costo fijo					$ 253,130		$ 2,538.92
					$ 613,130		$ 6,149.75

Tabla 22: Resumen de costos.

El conteo de las horas trabajadas y como se distribuyeron las mismas puede verse en el siguiente gráfico.

Figura 79: Distribución de las horas trabajadas.

En vista de lo anterior, se plantea un modelo de comercialización con un precio de venta fijo de US$4500. Con este valor, se necesitarán de al menos 4 ventas para generar ganancias. En la siguiente tabla se resume el análisis.

				Precio dolar	99.7
n	Costo fijo	Inversión	Costo total	Venta	Margen
1	$ 2,539	$ 6,150	$ 8,689	$ 4,500.00	$ -4,188.67
2	$ 5,078	$ 6,150	$ 11,228	$ 9,000.00	$ -2,227.58
3	$ 7,617	$ 6,150	$ 13,766	$ 13,500.00	$ -266.50
4	$ 10,156	$ 6,150	$ 16,305	$ 18,000.00	$ 1,694.58
5	$ 12,695	$ 6,150	$ 18,844	$ 22,500.00	$ 3,655.67
6	$ 15,234	$ 6,150	$ 21,383	$ 27,000.00	$ 5,616.75
7	$ 17,772	$ 6,150	$ 23,922	$ 31,500.00	$ 7,577.83
8	$ 20,311	$ 6,150	$ 26,461	$ 36,000.00	$ 9,538.92
9	$ 22,850	$ 6,150	$ 29,000	$ 40,500.00	$ 11,500.00
10	$ 25,389	$ 6,150	$ 31,539	$ 45,000.00	$ 13,461.08

Tabla 23: Análisis de amortización.

Para visualizarlo de forma más clara, se muestra el siguiente gráfico, en el cual se observa el punto de equilibrio entre costos e ingresos, así como también el margen de ganancia global.

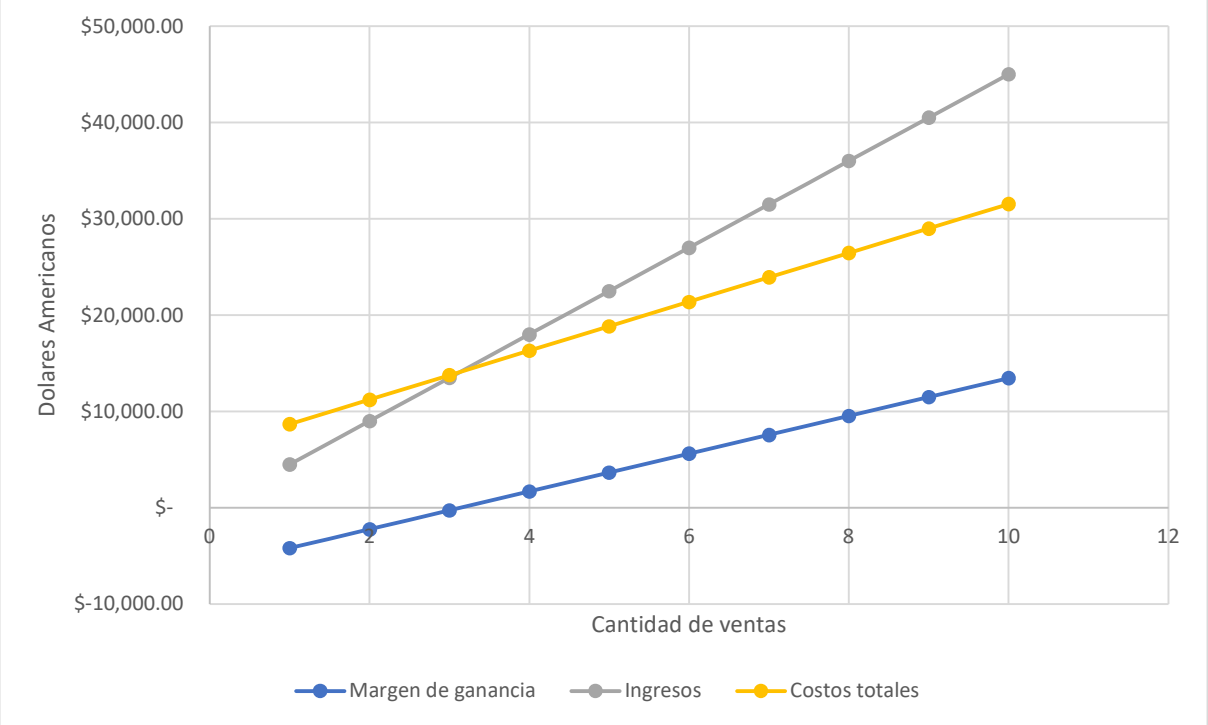


Figura 80: Gráfico de ganancia, ingresos y costos totales según cantidad de ventas.

Cabe mencionar que se tasó el producto en dólares dada la variación existente entre esta y el peso. Se sugiere cobrar el equivalente en pesos del precio antes fijado, con actualizaciones anuales atadas a la variación de las divisas.

Capitulo 5: Discusión y Conclusión.

Se considera que se cumplió con el objetivo de diseñar un sistema capaz de automatizar casi en su totalidad los ingresos y egresos a un establecimiento del tipo estacionamiento.

Se destacan como fortalezas la escalabilidad y modularidad del software implementado, dado mayormente por la arquitectura de microservicios y la utilización de Docker. Algo similar sucede con la electrónica realizada, ya que todas son inalámbricas y la información que requieren para funcionar (posición, configuraciones, IDs, etc.) son tomadas de la base datos. Esto permite que pueda haber múltiples entradas sin mayores modificaciones.

Otra fortaleza del sistema es precisión alcanzada en el reconocimiento de patentes en autos y camionetas. Incluso se logró reconocer vehículos en movimiento a velocidades no muy elevadas.

El sistema tiene algunos puntos débiles, que requerirían ser mejorados para alcanzar un producto más robusto. En primer lugar, el reconocimiento de patentes en motocicletas no está al nivel de lo alcanzado cuando se trata de autos y camionetas. Luego, los sensores de ultrasonidos utilizados en las placas detectoras de presencia no permitieron detectar distancias mayores a 130[cm] sin interferir unos con otros y generar falsos disparos. Esto es un problema ya que reduce mucho las posibilidades de aplicación de la solución actual. Estos sensores de distancia podrían ser reemplazados por otros que utilicen otra tecnología, por ejemplo, laser. Finalmente, se menciona como debilidad el resultado obtenido por el reconocimiento de patentes en situaciones nocturnas, que se debe principalmente a que los reflectores utilizados no son apropiados para la tarea que se les asignó. Si bien estos generan una luz intensa, esta es muy dispersa y no consiguen el resultado esperado. Estos podrían ser reemplazados por unos con menos dispersión del haz de luz, y del tipo infrarrojo para no molestar a los conductores de los vehículos.

El resultado obtenido es comparable a soluciones comerciales disponibles en el mercado, a un menor costo. Con la ventaja de que es modular y escalable, y que, gracias a Docker, el sistema puede funcionar tanto en un servidor local

(como es el caso de la implementación detallada en este informe), como así también en un servidor remoto, reduciendo así aún más los costos de implementación y adopción.

El sistema desarrollado podría ser mejorado en ciertos puntos. Con el añadido de una barrera que obligue a los vehículos a detenerse, se mejoraría el sistema de reconocimiento de imágenes, ya que podrían tomarse las fotos necesarias hasta que la patente sea leída correctamente. Otra mejora implementable es el reemplazo de MQTT en ciertos puntos donde la latencia es crítica (tiempo entre que un sensor de presencia detecta y una foto es tomada) por otra tecnología inalámbrica como Bluetooth, o incluso cablear los sensores que se encuentran próximos, como el grupo de entrada/s o salida.

Al haber utilizado tecnologías web para la HMI, es posible añadir una pasarela de pago sin muchas modificaciones, dotando al sistema de mayor versatilidad.

Si bien el trabajo aquí presentado se orientó al caso de un estacionamiento privado comercial, lo desarrollado es perfectamente aplicable a barrios privados o estacionamientos de empresas, siendo posible añadir otras funcionalidades como listas blancas/negras. Además, en estas situaciones no sería necesario contar con pagos, por lo tanto, el sistema podría funcionar de forma 100% autónoma, es decir, sin intervención humana.

Para finalizar, se destaca el rol fundamental de la planificación y organización en el desarrollo de un proyecto de estas características. Fueron de suma importancia herramientas como listas de tareas, diagramas de Gantt y el seguimiento de las horas dedicadas a cada una de las etapas planteadas. Esto nos permitió priorizar actividades y distribuir el tiempo de trabajo de forma ordenada.

Capitulo 6: Trabajos citados

[1] «Docker,» [En línea]. Available: https://www.docker.com/. [Último acceso: 24 6 2021].

[2] «HTTP,» Mozilla Developer Network, [En línea]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP. [Último acceso: 8 6 2021].

[3] «MQTT,» MQTT, [En línea]. Available: https://mqtt.org/. [Último acceso: 8 6 2021].

[4] «The Websocket API,» Mozilla Developer Network, [En línea]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API. [Último acceso: 12 6 2021].

[5] T. Yokotani y Y. Sasaki, *Comparison with HTTP and MQTT on Required Network Resources for IoT,* 2016.

[6] M. H. Skvorc y S. Srbljic, *Performance Evaluation of WebSocket Protocol for Implementation of Full-Duplex Web Streams,* Opatija, Croacia, 2014.

[7] A. T., K. P. y R. Rodrigo, *Automatic Number Plate Recognition in Low Quality Videos,* 2013.

[8] K. Tarun, G. Suraj y S. K. Dharmender, *An Efficient Approach for Automatic Number Plate Recognition for Low Resolution Images.*

[9] Philips, *Bi-directional level shifter for I²C-bus and other systems,* 1997.

[10] «Node,» [En línea]. Available: https://nodejs.org/es/. [Último acceso: 3 11 2021].

[11] M. Chan, «Thorn,» 5 3 2019. [En línea]. Available: https://www.thorntech.com/sql-vs-nosql/. [Último acceso: 10 6 2021].

[12] «PostgreSQL,» [En línea]. Available: https://www.postgresql.org/. [Último acceso: 10 7 2021].

[13] «Sequelize,» [En línea]. Available: https://sequelize.org/. [Último acceso: 14 7 2021].

[14] «Finale,» Github, [En línea]. Available: https://github.com/tommybananas/finale. [Último acceso: 15 7 2021].

[15] «Express,» [En línea]. Available: https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction. [Último acceso: 14 7 2021].

[16] «Socket.IO,» [En línea]. Available: https://socket.io/. [Último acceso: 25 7 2021].

[17] R. Chauhan, K. K. Ghanshala y R. Joshi, *Convolutional Neural Network (CNN) for Image Detection and Recognition,* 2018.

[18] D. Calvo, «Red Neuronal Convolucional CNN,» 20 7 2017. [En línea]. Available: https://www.diegocalvo.es/red-neuronal-convolucional/. [Último acceso: 3 9 2021].

[19] V. Powell, «Image Kernels Explained Visually,» [En línea]. Available: https://setosa.io/ev/image-kernels/. [Último acceso: 6 7 2021].

[20] K. Simonyan y A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition,* 2015.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li y L. Fei-Fei, *ImageNet A Large-Scale Hierarchical Image Database,* Princeton University, USA, 2009.

[22] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong y Q. He, *A Comprehensive Survey on Transfer Learning,* 2020.

[23] «Chrome Image Scrapper,» [En línea]. Available: https://github.com/akash-shastri/Chrome_image_scraper/blob/master/Chrome_Image_scraper.ipynb. [Último acceso: 10 8 2021].

[24] M. Grochowski y A. Mikołajczyk, *Data augmentation for improving deep learning in image classification problem,* Gdańsk, Polonia, 2018.

[25] «TensorFlow Object Detection API,» [En línea]. Available: https://github.com/tensorflow/models/tree/master/research/object_detection. [Último acceso: 10 8 2021].

[26] «TensorFlow,» [En línea]. Available: https://www.tensorflow.org/. [Último acceso: 26 7 2021].

[27] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto y A. Hartwig, *MobileNets Efficient Convolutional Neural Networks for Mobile Vision Applications,* 2017.

[28] «Patentes que rigen desde el 1 de Abril,» 3 4 2016. [En línea]. Available: https://www.16valvulas.com.ar/todo-lo-que-tenes-que-saber-de-las-nuevas-patentes-que-rigen-desde-el-1-de-abril/. [Último acceso: 9 7 2021].

[29] J. Redmon, S. Divvala, R. Girshick y A. Farhadi, *You Only Look Once: Unified, Real-Time Object Detection,* 2016.

[30] «YOLO v4 Custom Functions,» [En línea]. Available: https://github.com/theAIGuysCode/yolov4-custom-functions#license. [Último acceso: 1 9 2021].

[31] «Image Data Pre-Processing for Neural Networks,» 11 9 2017. [En línea]. Available: https://becominghuman.ai/image-data-pre-processing-for-neural-networks-498289068258. [Último acceso: 4 9 2021].

[32] «Reconocedor de Texto(OCR) para Patentes vehiculares de Argentina,» [En línea]. Available: https://github.com/ankandrew/cnn-ocr-lp. [Último acceso: 12 8 2021].

[33] «EasyOCR,» [En línea]. Available: https://github.com/JaidedAI/EasyOCR. [Último acceso: 12 8 2021].

[34] B. Shi, B. Xiang y Y. Cong, *An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition,* Wuhan, China, 2015.

[35] K. He, X. Zhang, S. Ren y J. Sun, *Deep Residual Learning for Image Recognition,* 2015.

[36] S. Hochreiter y J. Schmidhuber, *Long Short-Term Memory,* 1997.

[37] A. Graves, S. Fernández, F. Gomez y J. Schmidhuber, *Connectionist Temporal Classification Labelling Unsegmented Sequence Data with Recurrent Neural Networks.*

[38] «VueJs,» [En línea]. Available: https://vuejs.org/. [Último acceso: 3 11 2021].

[39] «Quasar,» [En línea]. Available: https://quasar.dev/. [Último acceso: 3 11 2021].