ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERIA DE SISTEMAS

UNIDAD DE TITULACIÓN

DESARROLLO DE UNA APLICACIÓN PARA DETECCIÓN DE PLACAS VEHICULARES DEL ECUADOR

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

JOSÉ LUIS LEÓN BAYAS

jose.leon01@epn.edu.ec

DIRECTOR: PhD. Sang Guun Yoo

sang.yoo@epn.edu.ec

Quito, Julio 2022

Certificación

Certifico que el presente trabajo fue desarrollado por José Luis León Bayas, bajo mi supervisión.

Firmado electrónicamente por:
SANG GUUN YOO .

PhD. Sang Guun Yoo

DIRECTOR

DECLARACIÓN DE AUTORÍA

Yo, José Luis León Bayas, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

José Luis León Bayas

DEDICATORIA

A mi familia, por ser el pilar de mi educación y la razón por la cual he logrado culminar con este trabajo de titulación, por todo el apoyo en cada momento de mi vida y por su infinito amor.

A mis profesores, que han sido los que me han brindado los conocimientos y valores necesario para mi vida profesional y me han hecho amar la carrera que escogí.

AGRADECIMIENTO

Agradezco a mis padres, por ser las personas que siempre me apoyan y confían plenamente en mí. Por haberme brindado todas las herramientas para poder culminar y cumplir mis objetivos.

A mi hermana, por ser un apoyo incondicional y darme ánimos para seguir adelante.

A mi tutor, PhD. Sang Guun Yoo, por su dedicación con el presente trabajo y su infinita paciencia con mi persona.

A mi familia, por ser el centro de mis alegrías y aventuras.

A mis profesores, por enseñarme todo lo que uso en mi vida diaria y profesional e inculcarme el amor por la carrera que escogí.

ÍNDICE DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

RESUMEN

En los últimos años, se ha dado una gran migración de las personas desde el campo hacia las ciudades, que hoy en día, son el centro de actividad de las civilizaciones. Sin embargo, el aumento en el número de habitantes genera un problema considerable debido a que la infraestructura se vuelve insuficiente para suplir las necesidades básicas.

Por ello, las administraciones buscan soluciones por medio de la tecnología, que ha ido evolucionando a la par del crecimiento de las ciudades, investigando la forma de poder brindar los servicios necesarios de una manera eficiente y con bajo costo.

Uno de los conceptos que se ha desarrollado, dentro del área de la movilidad, es el *Smart Parking,* que consiste en brindar la información necesaria al usuario para que su tiempo de búsqueda de estacionamientos se vea reducido, así como también, agilizar su entrada y salida a estos sitios mediante el uso de herramientas como: sensores, cámaras, inteligencia artificial, entre otros.

Una de las aplicaciones que se ha desarrollado para el *Smart Parking* es el reconocimiento de placas, tecnología que evita que el conductor tenga que esperar en el proceso de entrada o salida de un parqueadero. Sin embargo, el rendimiento de esta aplicación depende de las características de las placas de los países donde se desarrolla.

Es por ello que para este trabajo se ha desarrollado un sistema que permita el reconocimiento de placas vehiculares ecuatorianas, mediante la aplicación del framework de trabajo Scrum. Posteriormente, con el sistema desarrollado, se ha realizado un conjunto de pruebas que permitieron establecer la eficiencia de este, obteniendo que el reconocimiento de la localización de la placa se encuentra en un 85% y la lectura de los caracteres de esta alcanza un 100% de efectividad en la solución[1].

Palabras clave: placas vehiculares, lectura de placas, reconocimiento de caracteres, smart parking

[1] Resultado obtenido de las pruebas realizadas

ABSTRACT

Over the last few years, a large migration occurred from the countryside to the cities, which have become the center of activity of civilizations. However, the rising number of citizens has caused a significant problem because the existing infrastructure is not enough to cover the basic needs.

Because of that, the administrations look for solutions through technology, that has evolved along with the growth of the cities, researching how to provide the necessary services in an efficient way and at low cost.

One of the developed concepts, in the mobility area, is the Smart Parking which consists of giving the needed information to the user to reduce the time he/she spent looking for a place to park, additionally, it also helps to speed up the entry and exit to these sites by using tools like sensors, cameras, artificial intelligence, among others.

In this perspective, one of the applications developed for Smart Parking is the license plate recognition, technology that prevents the driver from having to wait in the process of entering and exiting a parking lot. Nevertheless, the performance of the application depends on the characteristics each countries' plates.

It is because of that, this work has developed a system that allows the detection and recognition of Ecuadorian vehicles plates, using the framework Scrum. Afterwards, with the developed system, a set of tests was performed to establish the efficiency, getting that the recognition of plate location is 85% and the recognition of the characters is about 100% with the solution[2].

Keywords: vehicular plates, plate reading, character recognition, smart parking

[2] Result from executed tests

1 INTRODUCCIÓN

1.1 Planteamiento del problema

Las ciudades se convirtieron en centros de concentración de la población debido a las facilidades y mejorías en el estilo de vida que pueden ofrecer a sus habitantes. Adicionalmente, con el aumento del desarrollo tecnológico en las últimas décadas, surgió el concepto de Smart City que busca brindar estas mejoras mediante el uso de la tecnología, lo que hizo que la concentración de las personas en las ciudades aumente considerablemente.

Estas mejoras se centran en hacer que las actividades cotidianas sean realizadas de manera eficiente, logrando una disminución en el tiempo de ejecución, y por ello, un bienestar en los ciudadanos.

Las aplicaciones más comunes se pueden identificar en el área de la salud, permitiendo que los ciudadanos sean capaces de agendar citas médicas de manera online y del mismo modo tener acceso a telemedicina; también es posible identificar cambios en servicios públicos en donde es posible realizar trámites de manera remota. Otra de las áreas críticas para el funcionamiento de una ciudad es la movilidad.

Sin embargo, ésta es un área extensa que abarca desde el funcionamiento de los semáforos hasta sistemas complejos de parqueo inteligente, con el objetivo de reducir los tiempos de movilización que se ven afectados por el aumento en el parque automotor dentro de las ciudades y todo lo que conlleva.

Una de las consecuencias de este aumento es que los espacios de circulación pueden resultar pequeños, sobre todo debido a zonas de parqueo sobre la calzada, por lo cual, zonas de parqueo especializadas son primordiales para permitir una mejor fluidez. Del mismo modo, es necesario que el tiempo para buscar un parqueadero dentro de dichas zonas se vea disminuido.

Con estos antecedentes, la tecnología ha enfocado sus esfuerzos para lograr convertir un parqueadero ordinario a lo que se denomina un Smart Parking, haciendo uso de aplicaciones que permitan el acceso y salida de los vehículos, como también aquellas que brinden información al usuario de los espacios libres. Para ello, es necesario el uso de sensores, pantallas, entre otros componentes; que permitan proveer dichos servicios.

Por ejemplo, en la actualidad, existen soluciones aplicadas que permiten el ingreso de los usuarios de manera rápida mediante el uso de cámaras y sistema de reconocimiento de placas.

Sin embargo, en la literatura, existe poca información acerca de la implementación de estas soluciones, además, muchos de ellos se encuentran enfocados en características propias de las placas de los países en los que se realizó la aplicación.
Por ello, surge la idea de trabajar en un sistema de lectura y reconocimiento de placas, con la finalidad de ajustarlo a las placas de Ecuador para obtener mejores resultados en la detección de estas y poder explicar su implementación.

1.2 Objetivo general

Desarrollar una aplicación para la detección de placas vehiculares de Ecuador mediante el uso de visión artificial.

1.3 Objetivos específicos

- Realizar un estudio de las herramientas usadas en el área de visión artificial con la finalidad de establecer la más adecuada para el éxito de la aplicación.
- Desarrollar la aplicación con la herramienta de visión artificial seleccionada.
- Probar la eficiencia de la aplicación mediante la realización de pruebas en tiempo real.

1.4 Alcance

El presente trabajo constará de tres partes: en la primera parte, el estudio del arte respecto al tema de visión artificial y detección de placas, con la finalidad de encontrar la herramienta óptima y hacer uso de ella; en la segunda parte se realizará las modificaciones necesarias a la herramienta para que pueda trabajar con el modelo de placa que se usa en Ecuador, debido a que en muchas de estas técnicas se basan en características diferentes a las placas nacionales en cuanto al color, cantidad y posición de caracteres, entre otros; finalmente, se ejecutarán las pruebas respectivas con el fin de establecer la eficiencia de reconocimiento de las placas, tomando en cuenta los cambios realizados a la herramienta.

1.5 Marco Teórico

1.5.1 Generalidades

Las Naciones Unidas (NN.UU.) presentaron entre 2018 y 2019 datos y estadísticas sobre la población mundial, obtenidos a través de censos y encuestas. Estos contienen datos históricos desde 1950 junto con proyecciones para los siguientes años hasta 2050. La

información presentada nos muestra que, en 1950, la ocupación de las ciudades de ese entonces se encontraba en un 29.5%, pasando hasta un 55.3% en el 2018, y con expectativas que para el año 2050 el 68.4% de la población mundial se encuentre asentada en las ciudades (United Nations, 2018a).

Este aumento en las cifras de población urbana se ha dado por la urbanización, tema que Terán ya tomaba en cuenta en 1969, definiendo a este proceso como: "[…] Crecimiento de las ciudades y el progreso y extensión de las formas de vida urbanas, con un paralelo abandono del campo y de las formas de vida rurales." (Terán, 1969).

Cabe resaltar que inicialmente, las grandes urbes como New York, recibieron una gran cantidad de migrantes. Pero en los últimos años, los migrantes han decidido moverse en las pequeñas y medianas ciudades (Bottino, 2009).

Sin embargo, según las NN.UU., para que una zona urbana pueda tener un desarrollo sostenible es necesario que se optimicen tres áreas fundamentales: económica, social y ambiental (United Nations, 2018b), con el fin de que el proceso de migración desde las zonas rurales se lleve a cabo sin una afectación en el estilo de vida.

Adicionalmente, el Banco Mundial menciona que es necesario realizar una urbanización responsable mediante una correcta gestión urbana que permita dar una solución a los desafíos que se presentan como: "[…] la creciente demanda de viviendas asequibles, de sistemas de transporte bien conectados y de otros tipos de infraestructuras y servicios básicos, así como de empleo […]" (Banco Mundial, 2020).

Estos desafíos se encuentran principalmente ligados a la calidad de vida de las personas, que también es una de las razones por la cual las personas migran a las ciudades, ya que allí se encuentran los trabajos mejor pagados y se puede acceder a un nivel de educación más alto (Khatoun & Zeadally, 2016).

No obstante, se crea una brecha, entre las expectativas de los nuevos habitantes urbanos y lo que en realidad las ciudades ofrecen, dada principalmente porque no se ha encontrado soluciones adecuadas a los desafíos como la infraestructura y los servicios, ya que estos fueron pensados para un menor tamaño de población, y están comenzando a llegar a sus límites y a superar el abastecimiento para el que fueron creados inicialmente (Khatoun & Zeadally, 2016).

A pesar de los contratiempos mencionados anteriormente, las urbes se han establecido como centros de innovación tecnológica desde los inicios de la sociedad, principalmente debido a las revoluciones industriales (Kozulj, 2003). Estas innovaciones han buscado resolver los problemas emergentes que ha tenido la humanidad, entre ellos, los desafíos que se presentan debido al rápido crecimiento de las urbes.

En medio de esta era de urbanización, surge el concepto de *Smart City*, que mediante el uso de la tecnología y la innovación busca la sostenibilidad del crecimiento y desarrollo de las urbes.

1.5.2 Ciudades Inteligentes

Varios autores han definido este concepto debido a la importancia que ha tomado en los últimos años; una de estas definiciones pertenece a la Asociación Española de Normalización y Certificación (AENOR) que indica:

> "Una Ciudad Inteligente es una ciudad justa y equitativa centrada en el ciudadano que mejora continuamente su sostenibilidad y resiliencia aprovechando el conocimiento y los recursos disponibles, especialmente las Tecnologías y Comunicación (TIC), para mejorar la calidad de vida, la eficiencia de los servicios urbanos, la innovación y la competitividad sin comprometer las necesidades futuras en aspectos económicos, de gobernanza, sociales y medioambientales." (Copaja-Alegre & Esponda-Alva, 2019)

El concepto antes mencionado define que el ciudadano es el eje central de las ciudades inteligentes, que de manera general buscan aumentar la productividad dentro de las ciudades y con ello también reducir los costos de administración de los recursos, enfocando los esfuerzos a mejorar la calidad de vida de las personas.

Sin embargo, y como se presentó anteriormente, aún existen desafíos que las grandes urbes deben superar para la transformación a una ciudad inteligente y estos retos son tratados por varios autores; una de las recopilaciones es presentada en (Yin et al., 2015), donde se establecen los siguientes:

1. *Infraestructura técnica:* para la implementación de una ciudad inteligente es necesario la implementación de tecnologías que se encuentren conectados a los sistemas actuales y futuros que permitan el movimiento de la información de una manera ágil y segura. Estas tecnologías están centradas principalmente en infraestructura de red, sin embargo, es necesario también la implementación de sensores que permitan extraer información (Yin et al., 2015).
2. *Aplicación de la tecnología:* es necesario que las tecnologías implementadas sean usadas en varias áreas que son importantes para el manejo de una ciudad, comenzando por la administración de esta, agregando áreas como: movilidad, consumo de energía, salud, economías, entre otros. Todo con el fin de que los servicios brindados sean inteligentes y eficientes (Yin et al., 2015).

3. *Integración de los sistemas:* los sistemas y subsistemas de una ciudad inteligente deben encontrarse conectados con el fin de brindar un mejor acceso a la información que finalmente se traduzca en la creación y mejoramiento de los servicios existentes (Yin et al., 2015).
4. *Procesamiento de datos:* la infraestructura levantada para una ciudad inteligente permitirá extraer datos de los distintos sensores y tecnologías implementadas, sin embargo, es necesario realizar un procesamiento de los datos para poder obtener información que permita mejorar los servicios existentes y volverlos inteligentes (Yin et al., 2015).

Del mismo modo, estos retos han sido atacados desde diferentes ángulos, entre ellos podemos encontrar el paradigma del *Internet of Things* (IoT) que hace referencia a todos los dispositivos electrónicos que tienen conexión a internet y que es encuentran captando y compartiendo datos (Ranger, 2020).

Adicionalmente, IoT enfoca sus esfuerzos en convertir un dispositivo de uso diario en un dispositivo inteligente, que se encontrará interconectado compartiendo información y permitiendo una interacción con el usuario desde cualquier lugar, volviendo a los dispositivos "parte integral del internet" (Zanella et al., 2014), esto forma una analogía con lo que se busca en una ciudad inteligente: sistemas conectados y accesibles.

Tomando en cuenta la similitud del paradigma de IoT con el concepto de *Smart City*, se puede indicar que ambos se complementan, debido a que IoT busca integrarse en áreas como: manejo de la basura, calidad del aire, monitorización del ruido, congestión del tráfico, consumo de energía, iluminación, entre otros (Zanella et al., 2014). Con ello cumple el segundo reto para el establecimiento de una ciudad inteligente.

Este paradigma tiene un abanico de aplicaciones centrados en la administración, control y automatización de servicios y actividades (Zanella et al., 2014). Uno de los servicios en los que se puede aplicar dicho paradigma es el control de la gestión vehicular que es un área de gran relevancia en una ciudad inteligente porque permite un mejoramiento en la calidad de vida al enfocar sus esfuerzos en disminuir la congestión vehicular que causa incomodidad debido a la alteración de los tiempos en la movilización (Federal Highway Administration: Office of Operations, 2020).

Un claro ejemplo se demuestra en el informe presentado por la Oficina de Operaciones de la Administración Federal de Autopistas de Estados Unidos en donde se establece como medida para la congestión, el tiempo de viaje y su fluctuación. Los datos mostrados muestran que con el paso del tiempo, el tráfico se ha vuelto más pesado y con ellos los tiempos de viaje se vuelven inestables y volátiles (Federal Highway Administration: Office of Operations, 2020).

Adicionalmente, el tráfico influye significativamente en la contaminación del aire que producen los vehículos, debido a que cuando existen cambios en la velocidad del vehículo mediante eventos de aceleración y desaceleración, las emisiones aumentan de CO2 aumentan, siendo el caso que hasta un cambio mínimo puede afectar (Barth & Boriboonsomsin, 2008), afectando a la calidad de vida de los ciudadanos debido a un aire contaminado.

Una causa de esta congestión, según el informe de la FHWA, son los embotellamientos (Federal Highway Administration: Office of Operations, 2020), y según estadísticas, el 30% de vehículos que están generando un embotellamiento pertenecen a personas que se encuentran en búsqueda de un lugar donde estacionar (Martínez, 2016). En promedio, según el estudio realizado por INRIX, una persona ocupa 17 horas al año en busca de un estacionamiento, lo que a su vez se traduce en un gasto aproximado de $345 dólares que se desglosan en: tiempo consumido, combustible y emisión de gases (McCoy, 2017).

1.5.3 Parqueaderos Inteligentes

Como se evidencia anteriormente, existe un problema al momento de encontrar un lugar donde estacionar, que a su vez genera afectación en la parte ambiental, económica y produce inconvenientes debido a la fluctuación en el tiempo de viaje entre dos puntos.

El concepto de *Smart City* intenta abarcar esta problemática mediante la aplicación del paradigma IoT, a través de los denominados *Smart Parkings* que, mediante el uso de sensores, buscan detectar la presencia de vehículos en los lugares de parqueo, con el fin de conocer la utilización de estos y poder brindar esta información al usuario (Arasteh et al., 2016), esto permite a la persona planificar de mejor manera su viaje para dirigirse directamente hacía un lugar de parqueo que se encuentre libre (Barriga et al., 2019).

Es necesario recalcar que los lugares de parqueo pueden ser de dos tipos: *on-street* o parqueaderos de calle, y *off-street* o también denominados estacionamientos (Silderhuis, 2013), y de ello dependerá que tecnología será usada con el fin de brindar el mejor servicio.

Para convertir un estacionamiento en un *Smart Parking* es necesario cumplir con una arquitectura que permitirá la aplicación de las tecnologías deseadas y a su vez cumplir con los objetivos de aplicar este concepto. Esta arquitectura simplificada debe constar de: sensores, protocolos de red y soluciones de software (Barriga et al., 2019).

Los sensores son el eje central de las soluciones, debido a que son los que recolectan y envían la información necesaria para que una solución sea capaz de cumplir con su objetivo; los protocolos de red se aplican desde un g*ateway* que implementará protocolos de IoT que permitan conectar los sensores con los sistemas; y finalmente la solución de

software que será la encargada de interpretar la información y hacerla visible de manera entendible con el fin de ponerla a disposición del usuario (Barriga et al., 2019).

Tal como se menciona, los sensores son la parte principal de las soluciones, por lo que cada solución buscara implementar la tecnología que le sea más conveniente, de esto surgen varios de tipos de sensores utilizados en las distintas soluciones, cada uno de ellos con sus principales características, ventajas y desventajas. Como se puede ver en (Barriga et al., 2019) algunos de estos sensores son:

- Cámaras: permite la visualización de uno o varios espacios de parqueo, así como también tienen un uso en las entradas y salidas de los estacionamientos permitiendo la entrada y salida de vehículos. Necesitan de un algoritmo que les permita procesar las imágenes.
- Sensores ultrasónicos: emiten una onda en una determinada frecuencia, mediante el rebote de esta en una superficie, es posible medir la distancia que existe con el objeto más próximo. Se aplican principalmente en soluciones que permiten conocer el estado de un espacio de parqueo (libre u ocupado), sin embargo, suelen resultar costosos debido a que es necesario un sensor por cada lugar.
- Sensores propios de un teléfono celular: los teléfonos inteligentes se convirtieron en una herramienta de uso diario para las personas, por lo que las soluciones han optado por usar los sensores integrados en ellos para lo cual es necesario la creación de una aplicación que recolete la información. Entre algunos de los sensores usados es posible encontrar: GPS, acelerómetro, giroscopio, entre otros.
- Sensores infrarrojos: trabajan de una manera similar a los sensores ultrasónicos, enviando una señal que rebota en una superficie. Del mismo modo, su uso extendido puede resultar costoso, porque es necesario un sensor por lugar de parqueo.
- Radar: estos sensores envían ondas que permiten conocer el estado del espacio a su alrededor permitiendo la generación de un mapa 3D, y con ayuda de inteligencia artificial, es posible usarlos para determinar la ocupación de lugares de parqueo. Necesitan de un software que permita procesar todos los datos provenientes del sensor y poder obtener información precisa.
- Magnetómetros: mediante el uso del campo magnético generado alrededor del sensor, es posible detectar la presencia de objetos metálicos grandes, en este caso específico, vehículos, evitando así falsos positivos con otras entidades, como personas. Es necesario un sensor por cada lugar de parqueo.

Estos sensores buscan cumplir con las necesidades de las distintas soluciones de acuerdo a sus necesidades, y como se menciona en (Barriga et al., 2019) los sensores más usados son los ultrasónicos y las cámaras, en ese orden. Los sensores ultrasónicos son precisos, por otro lado, las cámaras pueden detectar varios lugares de parqueo, por lo que se requiere de un menor número de ellas, adicionalmente, pueden ser usadas también para el control de las entradas y salidas de los estacionamientos.

Por ello, una forma de ayudar a una mejor gestión del tráfico es permitiendo al usuario conocer que estacionamientos o lugares de parqueo se encuentren disponibles. Adicionalmente, en los estacionamientos es necesario agilizar la entrada y salida de vehículos que se puede lograr mediante la apertura de las barreras presentes mediante un sistema de detección de placas como en (Buhus et al., 2016).

1.5.4 Detección de placas vehiculares

En esta solución se nos muestra un trabajo experimental que busca la identificación del vehículo mediante el uso de una cámara que captura la parte frontal del vehículo. Esta imagen es procesada por un equipo Raspberry para extraer el texto de la placa, con el fin de buscarlo dentro de una base de datos y permitir un acceso automático al parqueadero, pero con el conocimiento de la información del vehículo.

Es posible visualizar un trabajo similar en (Hermawati & Koesdijarto, 2010): mediante el uso de una cámara y un sensor, se detecta la cercanía y distancia de un carro, cuando las condiciones son las adecuadas (distancia del vehículo y posición de la cámara), el sistema realiza una captura y la procesa para obtener el texto de la placa; dicho texto es verificado en la base de datos y en caso de encontrarse registrados permitir el acceso.

Estos sistemas permiten que la entrada de los vehículos hacia los parqueaderos sea eficiente, disminuyendo los tiempos de acceso y permitiendo que se realice de manera automática por lo que el número de personas involucradas en el proceso se ve reducido.

1.5.5 Visión por computadora

Los sistemas mencionados anteriormente hacen uso de un recurso conocido como visión por computadora que forma parte de la rama de la Inteligencia Artificia. Este recurso enfoca sus esfuerzos a lograr que los sistemas computacionales sean capaces de identificar objetos y procesar imágenes y videos para obtener información útil, buscando emular la visión humana (Mihajlovic, 2021).

Esta rama ha ido evolucionando gracias a los avances que se han tenido en d*eep learning* y redes neuronales. Adicionalmente, la información que podemos encontrar en un minuto ha crecido de manera exponencial, alcanzando niveles exorbitantes, un ejemplo de ello es

que, actualmente, a lo largo de un minuto se transmiten alrededor de 694000 videos en Youtube (Telefónica, 2021). Este dato es relevante para la visión por computadora, debido a que permiten que los algoritmos y sistemas tengan mayor cantidad de información para poder entrenarse y mejorar los resultados obtenidos (Mihajlovic, 2021).

Y esta mejora de los resultados ha permitido que las aplicaciones para este recurso crezcan, encontrando entre ellas la detección y reconocimiento de objetos, que dentro de los sistemas de detección de placas vehiculares van a permitir que la placa sea encontrada (Mihajlovic, 2021).

Para lograrlo, se hace uso de algoritmos como: binarización de una imagen, que permite establecer una clara diferencia entre el *background* y el texto, permitiendo que las letras presentes sean más sencillas de identificar y leer (Jyotsna et al., 2016), también es posible aplicar filtros a la imagen con el fin de cambiar los valores de los pixeles en la imagen para lograr una imagen con mayor calidad o con bordes mejor identificados según sea la necesidad (Ganzorig, 2018).

1.5.6 Optical Character Recognition

Con el procesamiento se obtienen imágenes más claras y con características que son fácilmente reconocibles por un sistema, sin embargo, en el reconocimiento de placas vehiculares, es necesario aplicar un conjunto adicional de algoritmos que nos permitan identificar el texto presente.

Estos algoritmos se encuentran englobados en una rama denominada *Optical Character Recognition* (OCR), cuyo objetivo es trasladar a texto editable los caracteres o números que se encuentren dentro de la imagen, logrando de este modo una transformación desde una imagen a un documento, evitando la necesidad de ingresar el texto manualmente (IBM, 2022).

Es necesario establecer que dicho proceso tiene dos partes importantes: segmentación de caracteres, que consiste en realizar la separación de los caracteres que componen el texto; y el reconocimiento, que consiste en asignar un carácter a cada sección que se obtuvo en el paso anterior (Patel et al., 2013).

El resultado de aplicar estos algoritmos en el reconocimiento de placas es obtener el texto que pertenece a la placa, en el caso de Ecuador, obtener tres caracteres alfabéticos y de tres a cuatro numéricos.

1.5.7 Algoritmos de inteligencia artificial

Con el fin de brindar un mejor entendimiento de los distintos algoritmos para realizar el procesamiento de la imagen, necesario para poder extraer la placa vehicular, se definirán

y explicaran los que han sido usados a lo largo del proyecto, en el orden de aparición de estos.

1.5.7.1 Transformación de colores a escala de grises

Al momento de captar la imagen, cada uno de los pixeles será representada como una combinación de tres colores, denominado RGB (Red, Green, Blue por sus siglas en inglés), esto valores ocupan 8 bits cada uno, por lo que el número posible para cada uno va desde 0 a 255, sin embargo, al ser una combinación de estos, las combinaciones posibles ascienden hasta 16 millones de colores diferentes (Saravanan, 2010). Para visualizarlo de mejor manera, se establece que cada uno de los colores representa un eje en el espacio, como se denota en la Figura 1.

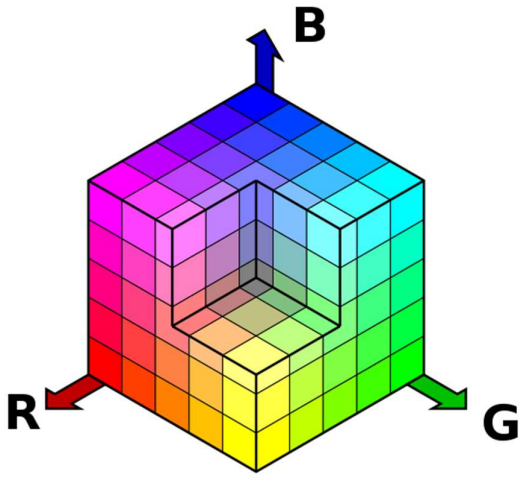

Figura 1. Cubo RGB

Debido a ello, procesar una imagen en RGB resulta de un mayor coste en tiempo y uso de recursos, por lo cual, en el procesamiento de imágenes es habitual realizar la conversión de dicha imagen a escala de grises, al realizarlo, cada píxel pasara a ocupar únicamente 8 bits, un valor de 0 a 255, que va a representar la cantidad de luz que posee el pixel con el fin de verlo en la escala (Saravanan, 2010). Es decir, si se establece que el píxel tiene un nivel de luz alto, el valor también será alto; del mismo modo, si se establece que el nivel de luz es bajo, el valor dado será bajo.

Adicionalmente, otras de las razones para realizar esta conversión es la reducción del ruido para el procesamiento, si el número de variables es mayor, los algoritmos pueden cometer errores al identificar características (Catherine Kouki, 2021). Finalmente, como se mencionó anteriormente, una imagen RGB hace uso de un número de recursos más alto, por lo que una imagen en escala de grises, con una menor cantidad de variables, necesitará

una menor cantidad de recurso y tomará menor tiempo en ser procesada (Catherine Kouki, 2021).

1.5.7.2 Bilateral Filter

Si el objetivo de realizar la transformación de una imagen a escala de grises es simplificar el procesamiento, también es necesario realizar lo que se denomina una “reducción de ruido” en la imagen, en el contexto del sistema es importante realizarlo debido a la finalidad que tiene el algoritmo, en este caso centra sus esfuerzos en obtener una imagen suavizada y en donde los bordes se encuentran mejor definidos.

Esto se logra mediante la aplicación de un filtro que evalúa el valor que posee cada píxel de una imagen, se compara con todos aquellos pixeles cercanos con el fin de establecer un valor que será un promedio de todos los pixeles evaluados, sin embargo, es necesario establecer que dentro del algoritmo existe también un sistema de pesos, en los cuales, los pixeles más lejanos, así como los que tienen un valor muy diferente, tendrán un menor peso y viceversa. De este modo se logra que los bordes no se pierdan pero que los valores que se tomarán en cada píxel sean similares con los cercanos (Shreyamsha Kumar, 2015).

Figura 2. Aplicación de filtro bilateral (Yang et al., 2009)

Para una mejor comprensión del algoritmo, se puede visualizar en la Figura 2 a pesar de que la imagen se encuentra a color, se puede notar un suavizado sobre todo en la parte del pómulo y los ojos. Sin embargo, no existe una perdida en los bordes presentes en la imagen, únicamente una homogenización de los colores presentes en partes extensas de la imagen.

1.5.7.3 Canny (Detección de Bordes)

El algoritmo Canny es usado para poder reconocer y extraer los bordes de una imagen. Consta de dos partes: primero, se aplica un operador Sobel sobre la imagen, dicho operador se enfoca en buscar todos los bordes existentes en la imagen (Fisher et al., 2003). El algoritmo aplica matrices como las de la Figura 3, que se usan para detectar bordes verticales y horizontales, respectivamente. Al aplicar las matrices dentro de la imagen, el valor que se obtiene por cada una de las aplicaciones va a depender del lugar donde se apliquen, de modo que cuando exista una mayor diferencia de colores el valor se alejara del cero, por el contrario, cuando el valor se acerque más al cero quiere decir que dentro del lugar de aplicación los colores son homogéneos o muy similares (Fisher et al., 2003).

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

Gx

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

Gy

Figura 3. Matrices Sobel

Las matrices se aplican en conjunto en busca de los dos bordes posibles, de modo que al final se realiza un cálculo del teorema de Pitágoras, tomando en cuenta que cada una de las matrices representa un eje, al obtener el valor, se aplica la misma lógica que se tenía anteriormente; adicionalmente, manteniendo la lógica de que las matrices forman un triángulo, es posible calcular la dirección que tiene el borde (Fisher et al., 2003).

Como segundo paso, se aplica el operador Canny, tomando en cuenta la salida del operador Sobel, mencionado con anterioridad, el objetivo de esta parte es disminuir el ancho de los bordes para que ocupen un solo píxel, adicionalmente, busca retirar aquellos bordes que no son relevantes y mantener aquellos que pueden brindar información (OpenCV Documentation, s/f-a).

Para lograr su objetivo, se separa en dos secciones: la primera consta la búsqueda de un máximo local donde se logró identificar un borde, es decir, al tomar uno conjunto de columnas o filas que forman un borde se evalúa cuál de ellas tiene el valor más alto, de ese modo encontramos el pixel o pixeles más representativo del borde dejándolo de ese modo en un solo pixel (OpenCV Documentation, s/f-a).

Posteriormente se aplica el denominado *hysteresis threshold,* el objetivo es mantener aquellos bordes que son significativos y eliminar aquellos que pueden ser ruido o no sean del todo representativos, para ello, se reciben dos valores, los cuales representan un valor mínimo y un valor máximo, estos, permiten evaluar los bordes de la siguiente manera: un borde cuyo gradiente o valor sea mayor al valor máximo siempre se va a mantener en la imagen, por el contrario el borde que se encuentre por debajo del valor mínimo dejará de ser tomado en cuenta (OpenCV Documentation, s/f-a).

Finalmente, aquellos bordes que se encuentren entre el valor mínimo y el máximo pertenecen a una zona de incertidumbre, por lo cual, se evalúan de manera distinta: si el borde no se encuentra conectado a ningún otro borde, y por ello, se encuentra completamente dentro de la zona de incertidumbre, será descartado; por el contrario, si dicho borde se encuentra conectado a uno que se encuentre por encima del valor máximo, no será excluido y formará parte del borde (OpenCV Documentation, s/f-a).

1.5.7.4 Cálculo de perímetros y aproximación de polígonos

A continuación, se aplica un algoritmo básico para el cálculo de perímetros, en donde se evalúan los puntos que conforman un borde y se mide la distancia entre puntos para darnos finalmente el perímetro (OpenCV Documentation, s/f-b).

Finalmente, se aplica una aproximación en la forma que posee el borde, en este caso se busca disminuir el número de vértices que posee el polígono formado por el borde (OpenCV Documentation, s/f-b).

1.6 Estudio del arte

El reconocimiento de placas vehiculares tiene un papel importante en aplicaciones de la vida diaria como: pago automático en estaciones de peaje, monitoreo del tráfico, entre otras (Du et al., 2013). Por ello, se han presentado trabajos que buscan obtener resultados óptimos en esta problemática enfocándose en las distintas etapas que forman parte del proceso de detección y reconocimiento de las placas.

A pesar de que las etapas pueden variar entre autores deben ser consecutivas. En la Figura 4 podemos encontrar las principales.

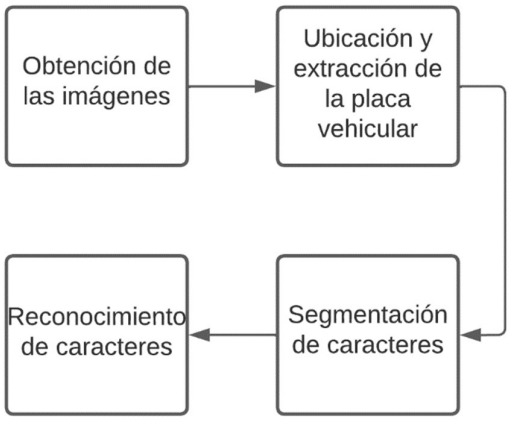

Figura 4. Etapas de reconocimiento de placas vehiculares

1. Obtención de la imagen o conjuntos de imágenes: en primer lugar, es necesario establecer el recurso que se utilizará para realizar la captura de las imágenes (Patel et al., 2013). Adicionalmente, es necesario tomar en cuenta factores como: calidad de la imagen, iluminación y en caso de ser en exteriores, el factor climático (Du et al., 2013).
2. Ubicación y extracción de placa vehicular: dentro de la imagen, es necesario escoger regiones que son candidatas para contener la placa vehicular. En esta etapa, se realiza el procesamiento de las imágenes con el fin de disminuir el ruido y los falsos positivos que puedan aparecer, junto con la selección de la región óptima para las etapas siguientes (Patel et al., 2013).
3. Segmentación de caracteres: luego de obtener la ubicación exacta de la placa vehicular dentro de la imagen, es necesario reconocer y separar la región de cada uno de los caracteres, evitando que una sola de ellas contenga más de un carácter (Patel et al., 2013).
4. Reconocimiento de caracteres: con las regiones separadas, se procede a reconocer los caracteres y números que nos permitirán leer finalmente la placa vehicular. Las complicaciones que pueden aparecer en este punto, se encuentran en el reconocimiento de caracteres que pueden parecer iguales como: B y 8, O y D (Patel et al., 2013).

Tomando en cuenta las cuatro fases implicadas en el reconocimiento de placas, es posible aplicar diferentes algoritmos para lograr el objetivo. Estos algoritmos han sido revisados en (Du et al., 2013) y (Patel et al., 2013) y se explican a continuación.

En la Tabla 1, se muestran los algoritmos de detección que se usan para detectar la región en la que se encuentra la placa. Por otro lado, en la Tabla 2, se muestran los métodos que permiten segmentar los caracteres. Y finalmente, en la Tabla 3, se muestran aquellos algoritmos que permiten reconocer que caracteres se encuentran presentes en la imagen.

Tabla 1. Algoritmo de detección

Método Principal	Descripción	Ventajas	Desventajas	Método Específico	Descripción	Exactitud	Artículo
Información del perímetro o bordes	Se busca todos los rectángulos que se puedan encontrar en la imagen mediante la identificación de sus bordes. Con la información de los bordes se puede hacer uso de ellos de tres maneras diferentes: solo bordes verticales u horizontales, o una mezcla de ambos	Son algoritmos simples y de rápida ejecución	La imagen debe ser clara para poder definir una continuidad en los bordes.	Filtro/Operador Sobel	En primer lugar, es necesario establecer la relación de aspecto que tiene la placa. El algoritmo busca enfatizar los bordes de la imagen, sobre todo por el cambio de color que puede existir entre la placa y el carro, por lo cual puede presentar problemas cuando el cambio no es notorio para el algoritmo.	Vertical: 96.2%	(Sarfraz et al., 2003)
				VEDA	Realiza una binarización de la imagen para encontrar el threshold de manera dinámica y aplicar el metodo de Bradley,	Similar a Sobel. No se encuentra definido en el paper	(Al-Ghaili et al., 2008)

					posteriormente aplica el algoritmo ULEA (Unwanted Lines Elimination Algorith) y finalmente aplica el algoritmo VEDA el cual aplica una máscara que se va trasladando a lo largo de la imagen y evalúa si el centro centro es de color negro, si es el caso, se evalúa los extremos derecho e izquierdo y se establece un borde.		
				Block based	Se aplica un bloque de tamaño específico que debe estar ajustado al tamaño y la resolución de la imagen a la que se va a aplicar. El bloque se va posicionando sobre la imagen en	92.5%	(Hsi-Jian Lee et al., 2004)

					busca de posibles regiones candidatas que posteriormente deben clasificarse para seleccionar la adecuada, eliminando falsos positivos.		
				Hough Transform	Detecta líneas rectas dentro de una imagen, incluso si poseen una inclinación de hasta 30°. Consume demasiado tiempo y necesita trabajar en conjunto con un algoritmo de contornos.	98.8%	(Duan et al., 2005)
Global Image Information	Mediante el uso de las características extraídas se busca disminuir la carga de trabajo que puede necesitarse para el escaneo de las			Connected Component analysis (CCA)	Funciona como reconocimiento debido a que etiqueta de manera similar aquellos pixeles que forman parte un conjunto, mediante el	96.62%	(Hsien-Huang P. Wu et al., 2006)

	imagenes. Busca describir el total de la imagen y el objeto que pueda encontrarse en ella mediante: contour representations, shape descriptors and texture features.				uso de las características globales de la imagen. Se usa principalmente medidas espaciales como: área y relación de aspecto.		
				2D cross relation	Mediante el uso de un template se busca a lo largo de toda la imagen para escoger aquella región que más se le parezca. Puede consumir tiempo de procesamiento. It is of the order of n4 for n×n pixels.	99.00%	(Miyamoto et al., 1991)
Características de las texturas	Depende de la presencia de caracteres en las placas, debido al uso que se le da en el cambio de la escala de grises entre las	Permite leer la placa aun si los bordes no se encuentran del todo definidos	Pueden resultar computacionalmente complejos en algunos escenarios, especialmente debido a los cambios de iluminación,	Scan-lines	El número de caracteres es definido por el número de picos que se encuentran en el escaneo de la línea. A partir de eso es	96%	(Hong-ke Xu et al., 2005)

	letras y el fondo de la placa		background o si existen demasiados bordes.		necesario definir el número de caracteres que se busca.		
				Vector Quantization	Similar a cluster pero con puntos random y un área que puede variar. La idea es mover el centroide con centro en el punto P de modo que se busca acomodarse y juntar la mayor cantidad de puntos similares.	98.00%	(Zunino & Rovetta, 2000)
				Sliding Concentric Windows (SCW)	Las placas se cuentan como irregularidades dentro de la imagen, por ello, si existe un cambio grande en las características locales se toma como un candidato para placa.	96.5%	(Anagnostopoulos et al., 2006)

Características de los colores	Consiste en usar la combinación de colores de los caracteres junto con la placa ya dicha combinación es casi exclusiva de la misma.	No se ve afectado por las placas giradas o deformadas	Susceptible a la iluminación (RGB) y al ruido (HLS)	Color Edge Detector	Detecta los lugares en donde existen los contrastes de color definidos.	97.90%	(Chang et al., 2004)
				Edge Image	Se busca de manera horizontal dentro de la imagen, en caso de que exista un pixel que se encuentre dentro del rango de colores admitido por la placa se busca en su vecino, si uno o más vecinos tienen el mismo rango de color, se toma en cuenta como un borde, finalmente, la nueva imagen con bordes es analizada en búsqueda de zonas que puedan ser candidatas a placas.	95.3%	(Yao-Quan Yang et al., 2005)

Tabla 2. Métodos de segmentación de caracteres

Método Principal	Descripción	Ventajas	Desventajas	Método Específico	Descripción	Exactitud	Artículo
Projection Profiles	Se aprovecha el contraste que existe entre los caracteres y el fondo de la placa.	Los métodos que usan las proyecciones verticales y horizontales son los más faciles y usados. Es independiente de la posición.	Cualquier ruido puede afectar al proceso, así como también es necesario tener el conocimiento del número de caracteres.	Vertical Projections	Mediante el uso de una imagen binaria de la placa extraída, los caracteres son proyectados de manera vertical con el fin de definir el inicio y el final de los caracteres. Se incluye *noise removing* y *character sequence analysis*.	99.20%	(Zhang Sanyuan et al., 2004)
				Projections	En contraposición con el anterior, se hace uso de una imagen junto con sus colores originales	91.25%	(Eun Ryung Lee et al., 1994)
Prior Knowledge of Characters	Se hace un escaneo dentro de la imagen binaria con el fin de encontrar el inicio y final de cada una de las letras. Cuando el			Método 1 (nombre no definido)	Se hace un escaneo dentro de la imagen binaria con el fin de encontrar el inicio y final de cada una de las letras. Cuando el	72%	(Busch et al., 1998)

	ratio que existe entre los pixeles del carácter y los pixeles del fondo supera un threshold se considera como el inicio de un carácter y para reconocer el final se hace lo contrario.				ratio que existe entre los pixeles del caracter y los pixeles del fondo supera un threshold se considera como el inicio de un caracter y para reconocer el final se hace lo contrario.		
				Neural Networks	La placa reconocida se redimensiona hasta ajustarse con un template en donde se conoce exactamente en donde se encuentran los caracteres. El problema se encuentra si existen cambios en la placa.	90%	(Paliy et al., 2004)

Tabla 3. Algoritmos de reconocimiento de caracteres

Método Principal	Descripción	Ventajas	Desventajas	Método Específico	Descripción	Exactitud	Artículo
Raw Data/ Template matching	Se redimensiona la imagen del caracter hasta un tamaño predeterminado y se compara con plantillas existentes de los caracteres.		Puede fallar si se da el caso de que la fuente del carácter es diferente, esta rotado, no es completo.	Varios	Se hace uso de una gran variedad de algoritmos de comparación de imagenes. Entre ellos encontramos: Hamming distance, Hausdorff distance, Jaccard value.	Hamming: 91.25% Hausdorff: 98% Jaccard: 95.24%	(Eun Ryung Lee et al., 1994) (Tang Shuang-tong & Li Wen-ju, 2005) (Sarfraz et al., 2003)
Extracted features	Aplicar una extracción de características a la imagen en escala de grises con el fin de compararlo posteriormente.	Permite reconocer los caracteres a pesar de las distorsiones. Menor tiempo que el template matching.		Varios	Existen varias formas de extracción de las características, entre las cuales podemos tener: dividir la imagen del carácter en bloques para extraer el número de pixeles negros por bloque, otra forma es tomar en cuenta el centro del carácter y escanear tanto hacia arriba como para		(Artículos varios)

					abajo de modo que se cuenta cuantas transiciones existen en ambos lados, también se puede usar el contorno del carácter para transformarlo en un onda que es independiente del tamaño o la fuente, se puede usar el Kirsch operator, entre otros. Finalmente es necesario hacer uso de un clasificador con las características extraídas.		

Debido a la eficacia de estos trabajos para el mejoramiento en los tiempos de acceso, este trabajo será enfocado a un objetivo similar a los mencionados anteriormente, con la diferencia de ser realizado para el reconocimiento y lectura de placas ecuatorianas, esto debido a que trabajos como (Hermawati & Koesdijarto, 2010) hacen uso de características específicas de las placas para poder completar el proceso exitosamente.

Adicionalmente, en los modelos propuestos existe escasa información sobre su implementación debido a que no se especifica tecnologías, lenguajes de programación utilizados o librerías, lo cual es otra de las razones para la realización del presente trabajo.

2 METODOLOGÍA

El presenta trabajo se centra en el desarrollo de una aplicación que permita la detección de las placas vehiculares, para luego ejecutar un conjunto de pruebas en tiempo real que permitirán evaluar la eficacia y velocidad de la detección.

Para logarlo, se hará uso de un framework de desarrollo de metodología ágil, denominado Scrum, cuyo principal objetivo es convertir las necesidades en un conjunto de tareas que al irse desarrollando permitan obtener un prototipo incremental, es decir, cada una de las fases planificadas tendrá como resultado un producto funcional.

Finalmente, las pruebas a realizarse tendrán como objetivo evaluar los resultados que se han obtenido de un conjunto de imágenes o video dentro de la aplicación, de modo que serán pruebas simples evaluando únicamente si se logró reconocer la placa y el texto que lleva escrito.

2.1 Scrum

Con el conjunto de herramientas establecidas, es necesario iniciar con el proceso de desarrollo de la aplicación con el fin de generar la solución al problema planteado en el presente trabajo.

Para ello, se tomó en cuenta un framework de desarrollo denominado Scrum, debido a que su principal enfoque es la agilidad y se establece al desarrollo como un proceso de cambio y mejora, de modo que se inicia con la menor cantidad de conocimiento y se lo consigue a lo largo del proyecto (Drumond, s/f) .

El uso de este framework implica la aparición de un conjunto de términos que deben ser definidos con el fin de tener un entendimiento del proceso a seguir. En este caso, los vamos a dividir en tres categorías: *Scrum artifacts,* representan las herramientas que vamos a usar para lograr el desarrollo; *Scrum events*, son las actividades que se realizan; y

finalmente los roles dentro de *Scrum,* que definirán las actividades que va a realizar la persona (Drumond, s/f).

Para poder comenzar con el trabajo de este framework es necesario que exista un *product owner*, que forma parte de los roles; la persona asignada a dicho rol será la misma que posee el conocimiento del negocio o proyecto, sabe cuáles son los objetivos que se buscan y define cuales son las características que debe tener el producto. Finalmente, es el encargado de generar y manejar el *product backlog*, que forma parte de los *Scrum artifacts,* el cual se define como la lista de pendientes del proyecto, contiene un conjunto de características, requerimientos, mejoras y arreglos que deben realizarse para poder cumplir con las exigencias del *product owner* (Drumond, s/f).

Los encargados de realizar las actividades propuestas en el *product backlog* va a ser equipo de desarrollo Scrum, categorizado como un rol dentro del proceso, que se encontrará conformado por personas con distintas habilidades que aporten al desarrollo y creación del nuevo producto (Drumond, s/f).

El trabajo será realizado dentro de un *Scrum event* denominado *Sprint,* que consiste en un periodo de tiempo dentro del cual el equipo de desarrollo cumplirá con las tareas asignadas para dicho *sprint* buscando brindar un valor incremental al producto, sin embargo, dicho trabajo debe ser planificado con anterioridad, con el fin de buscar las actividades que puedan aportar mayor valor y que puedan resultar críticas para el producto; esto se realiza en un *Scrum event* denominado *Sprint planning* (Drumond, s/f).

De esta fase de planificación se obtiene dos *Scrum artifact*, el primero se denomina *Increment* que es el objetivo del sprint, en este caso, es necesario que sea un producto usable en su totalidad, el segundo es el denominado *Sprint backlog* que define las actividades que se van a realizar a lo largo del *sprint*, aunque puede ser flexible, el objetivo con el que se inició el *sprint* y que se definió en la planificación no puede ser cambiado. Adicionalmente, durante este periodo de tiempo, es necesario que exista otro *Scrum event* que permitirá que el equipo se organice y pueda alinear las actividades que va a realizar con el objetivo del sprint, es una reunión corta denominada *daily scrum* que como lo dice su nombre es diaria (Drumond, s/f).

Para la finalización del sprint es necesario que se tomen en cuenta dos *Scrum events* que son importantes para poder revisar el trabajo realizado por el equipo. El primer evento es denominado *sprint review* que permite al equipo presentar los avances realizados en el producto, establecer cuales fueron las actividades que se completaron y se decide si el i*ncrement* será liberado o no. El segundo evento para considerar es el *sprint retrospective* que permite al equipo evaluar como fue el desempeño a lo largo del *sprint,* y buscar mejoras dentro de este (Drumond, s/f).

A lo largo de todo este proceso, existe un rol que se encarga de coordinar y manejar el *sprint* de modo que sea productivo y cumpla con los objetivos necesarios, este rol es denominado *scrum master,* la persona asignada a dicho rol debe trabajar tanto con el equipo de desarrollo como con el *product owner* (Drumond, s/f).

Con los conceptos claros, es necesario establecer el proceso de manera gráfica para que exista un mejor entendimiento y también definir las fases que se van a usar en el presente trabajo.

En la Figura 5 es posible visualizar el proceso en su totalidad y con ello se definirán fases necesarias para este trabajo:

- Fase 0: se definirá el *product backlog* en base a las herramientas definidas para su uso, tomando en cuenta el objetivo que se está buscando con este desarrollo y la necesidad que se busca solventar.
- Fase 1: se realizará el *sprint planning* con el fin de obtener un producto usable al final del *sprint* de modo que se definirá cual es el objetivo y que se espera con el *sprint* y se definirán las actividades a realizar.
- Fase 2: se realizarán las actividades establecidas durante un tiempo determinado, se desarrollará e implementará las distintas herramientas, por lo cual es un *sprint.*
- Fase 3: se realizará el *sprint review* de modo que se revisará un prototipo usable que podrá ser cambiado o mejorado en posteriores *sprint.*

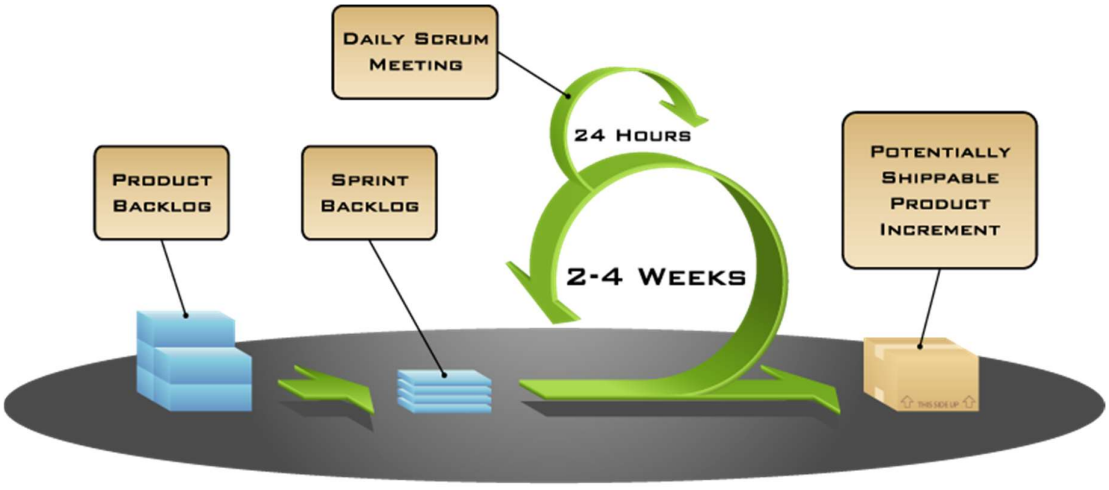

Figura 5. Proceso de Scrum

3 PLANIFICACIÓN

De acuerdo con la definición, las actividades deben ser desarrolladas a lo largo de 420 horas, las mismas que se dividirán de la siguiente manera:

- 60 horas para buscar, evaluar, clasificar y extraer la información más importante de artículos científicos.
- 20 horas asignadas al desarrollo de la arquitectura en base a otras soluciones
- 275 horas designadas para el desarrollo del proyecto
- 65 horas para realizar las pruebas correspondientes al sistema completo

Las necesidades que se van a cumplir en el proyecto se tradujeron en módulos obtenidos mediante la realización del estudio del arte.

Para el desarrollo del proyecto, se tomo en cuenta la realización de 3 sprints, con una duración de tres semanas cada uno. A lo largo del proyecto se fueron realizando cambios en las actividades programadas con el fin de ajustarse a las dificultades que se fueron encontrando y superando durante el desarrollo.

Finalmente, el tiempo para las pruebas se fue utilizando en todos los sprints con el fin de asegurar que el desarrollo realizado sea completamente funcional y permita continuar con el trabajo.

4 IMPLEMENTACIÓN

El objetivo principal del proyecto es el desarrollo de un sistema de detección de placas vehiculares del Ecuador, por lo que, con el fin de aplicar la metodología Scrum, se definieron tres componentes fundamentales que también permiten tener una mejor visión del problema que se busca solventar. El flujo de trabajo se puede ver en la Figura 6 donde se presentan de manera ordenada las fases que se van a ejecutar.

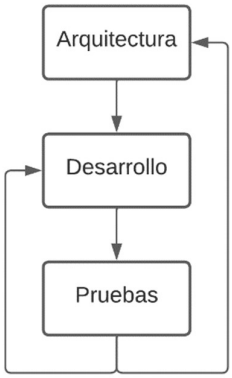

Figura 6. Proceso de generación del sistema

4.1 Arquitectura

De acuerdo a las fases presentadas en la Figura 6. Proceso de generación del sistema, es necesario iniciar con el diseño de la arquitectura del sistema en donde se definirá las partes y módulos que va a tener y la función de cada una de ellas.

Este diseño también permitirá definir las tareas que van a ser realizadas a lo largo de la fase de desarrollo, por ello, es importante iniciar con esto antes de producir el *product backlog* inicial.

La arquitectura del sistema va a mantener la estructura que se revisó en el estudio del arte, por lo tanto, constará de cuatro módulos fundamentales para su funcionamiento, que se van a ver reflejados en la Figura 7

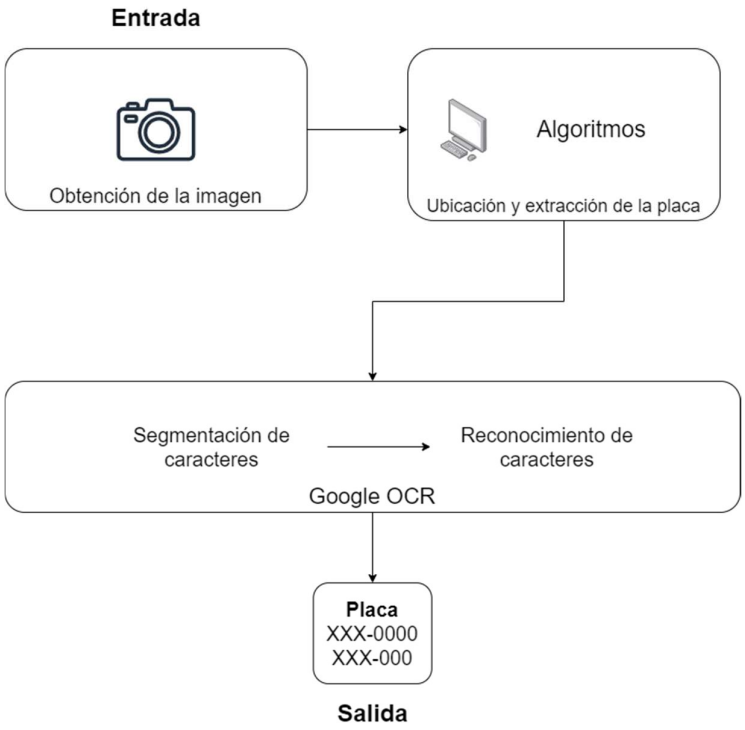

Figura 7. Arquitectura de la solución

En primer lugar, va a ser necesario establecer el mecanismo que va a permitir captar la información necesaria para la solución, en este caso, un video en donde se encuentre el vehículo y su placa, que servirá como entrada para el proceso a ejecutar. El video debe ser claro y tener las condiciones de luz adecuadas. Adicionalmente, es necesario denotar que un video es una secuencia de imágenes, con esta en consideración, el sistema deberá ser capaz de hacer uso de las imágenes obtenidas sin complicaciones adicionales.

Para el siguiente paso, existirán algoritmos definidos que realizarán aplicación de filtros y detección de bordes dentro de la imagen con el fin de: reducir el ruido existente, mejorar o definir los bordes y finalmente, mediante una detección de bordes y figuras específicas, lograr identificar la posición exacta donde se encuentra la placa, que posee una estructura como la mencionada en la Figura 8.

Posteriormente, y según lo revisado, es necesario que dentro de la placa se realice y se identifiquen las secciones que pertenecen a caracteres, en este caso, se denomina segmentación de caracteres, por lo tanto, el algoritmo deberá ser capaz de distinguir entre secciones que posean texto y aquellos que no, y adicionalmente sean capaces de separar y poner un carácter en una sola sección.

Finalmente, el siguiente proceso necesario a ejecutar consta de un último algoritmo que tome las secciones obtenidas anteriormente y defina el valor o letra de cada una, entregando finalmente un texto completo, adicional a ello, debido a la estructura que posee

la placa ecuatoriana, es necesario establecer una verificación en la estructura del texto obtenido.

Figura 8. Dimensiones y estructura de la placa vehicular ecuatoriana

Para la etapa final de verificación, es necesario aplicar una máscara mediante el uso de expresiones regulares, debido a que el algoritmo de la fase anterior puede darnos distintas líneas de texto, por lo cual se debe diferenciar la placa y el texto adicional. De este modo, la salida del sistema será la placa en texto legible.

4.2 Product Backlog

Con una arquitectura definida y de acuerdo con Scrum, se estableció un conjunto de tareas generales que se muestran en la Tabla 4. Tareas Generales

Tabla 4. Tareas Generales

ID	Nombre
R1	Revisión de Python: estructuras básicas y uso de bibliotecas externas
R2	Revisión de la documentación de la librería de OpenCV
R3	Revisión de ejemplos funcionales con OpenCV
R4	Revisión de las funciones de OpenCV más usadas en el reconocimiento de placas vehiculares
R5	Revisión de librerías para extracción de texto de una imagen
D1	Instalación del ambiente para desarrollo
D2	Desarrollo del preprocesamiento necesario que permita la identificación de una placa vehicular
D3	Desarrollo de la extracción de la placa vehicular en una imagen

D4	Desarrollo del reconocimiento del texto en la región de la placa vehicular de una imagen
D5	Búsqueda de imágenes de placas vehiculares para realizar las pruebas de los ejemplos y funciones aplicados
P1	Captura de imágenes y video de vehículos para poder probar la eficiencia de la solución desarrollada
P2	Pruebas de la extracción de la placa vehicular de una imagen
P3	Pruebas del reconocimiento del texto en una imagen
P4	Pruebas de la solución en imágenes tomadas
P5	Pruebas de la solución en un video

Para la evaluación y priorización de las actividades que van a realizarse en cada uno de los sprints, se estableció un conjunto de valores presentados en la Tabla 5. Valoraciones

Tabla 5. Valoraciones

Nombre	Valoración
Baja	1
Media	2
Alta	3

Con las valoraciones establecidas, es necesario que para cada una de las actividades se establezca la prioridad y dificultad, con el fin de poder organizar la realización de las tareas a lo largo de cada uno de los sprints. En este caso, debido a la arquitectura de la solución, la tarea de desarrollo está enfocado en realizar las actividades de manera ordenada, debido a que las funcionalidades son parte de un proceso, por lo cual la prioridad no aplica. Por otro lado, la dificultad fue establecida principalmente en base al conocimiento y experiencia que se tiene de cada una de las herramientas, adicionalmente, se tomó en cuenta la cantidad y claridad de la documentación existente y finalmente los ejemplos que se pueden encontrar y ejecutar.

A continuación, se muestran las actividades que van a realizarse separadas en las tareas generales junto con su complejidad y prioridad en la Tabla 6. Product Backlog

Tabla 6. Product Backlog

TG	Código	Actividad	Complejidad	Prioridad
R1	R11	Estructura básica de Python	1	1
	R12	Instalación y uso de bibliotecas	1	1
R2	R21	Términos habituales y funcionamiento básico	1	3
	R22	Tutoriales de la aplicación de las diferentes funciones	2	2

	R23	Aplicación de funciones revisadas en los tutoriales para entendimiento de su uso y cómo funcionan	2	2
R3	R31	Búsqueda de programas aplicados para la solución del problema para usarlos como ejemplo	1	1
	R32	Extracción de las funciones más usadas para lograr la extracción de una placa vehicular	2	2
R4	R41	Revisión de las funciones extraídas de los ejemplos (para que sirven, variables admitidas y valores retornados)	3	3
	R42	Aplicación de las funciones a imágenes de ejemplo para poder visualizar que función cumplen, cambio de parámetros para visualizar afectación a los resultados	3	3
R5	R51	Revisión de ejemplos funcionales en donde se realiza la extracción texto de una imagen	1	1
	R52	Instalación de las bibliotecas usadas en los ejemplos	1	1
	R53	Aplicación de las librerías a imágenes de ejemplo para comprobar la facilidad de uso y la eficacia	3	3
	R54	Revisión de la documentación de pytesseract	1	2
	R55	Revisión de ejemplos de uso en Python	1	1
D1	D11	Instalación de máquina virtual	1	3
	D12	Instalación de interprete de Python y programas necesarios	1	N/A
	D13	Establecimiento de un ambiente virtual dentro de la máquina	2	N/A
	D14	Instalación de OpenCV	2	N/A
	D15	Instalación de bibliotecas adicionales	1	N/A
	D16	Instalación de editor de texto	1	N/A
D2	D21	Aplicación de filtros y funciones usadas para el preprocesamiento de una imagen	2	N/A
	D22	Aplicación de filtros y funciones usadas para el preprocesamiento de una imagen que contenga una placa vehicular ecuatoriana	2	N/A
	D23	Cambio de parámetros para visualizar cuales brindan mejores resultados en base a la necesidad de que la placa sea más visible	3	N/A
D3	D31	Aplicación de funciones que permiten identificar los	2	N/A

		bordes de una imagen		
	D32	Aplicación de funciones que permiten extraer bordes de acuerdo a parámetros establecidos	2	N/A
	D33	Aplicación funciones para extraer bordes de una imagen que contenga una placa vehicular ecuatoriana	2	N/A
	D34	Cambio en los parámetros en las funciones que permiten la extracción de las placas y evaluación de los resultados en base a los parámetros	2	N/A
D4	D41	Desarrollo y aplicación de las funciones de pytesseract en una imagen	2	N/A
	D42	Aplicación de pytesseract en una imagen donde se puede visualizar una placa vehicular ecuatoriana	3	N/A
	D43	Cambio en los parámetros en las funciones que permiten la extracción de las placas y evaluación de los resultados en base a los parámetros	3	N/A
D5	D51	Búsqueda de imágenes en motores de búsqueda	1	N/A
P1	P11	Toma de imágenes de vehículos con placas vehiculares	2	2
	P12	Toma de un video de un vehículo con placas vehiculares para poder realizar la prueba	2	2
P2	P21	Ejecución de pruebas sobre las imágenes para evaluar la precisión de la extracción de la placa vehicular	3	3
P3	P31	Ejecución de pruebas sobre las imágenes para evaluar la precisión del reconocimiento de la placa vehicular del texto	3	3
P4	P41	Ejecución de pruebas sobre las imágenes descargadas para evaluar la precisión de la solución	3	3
	P42	Ejecución de pruebas sobre las imágenes tomadas para evaluar la precisión de la solución	3	3
P5	P51	Ejecución de pruebas sobre el video tomado para evaluar la precisión de la solución	3	3

A lo largo del desarrollo de las actividades antes mencionadas, existirá un proceso de retroalimentación, sustentado en las pruebas ejecutadas, que permitirá realizar cambios en

la arquitectura o modificaciones en el desarrollo en etapas tempranas, con el fin de lograr un mejoramiento continuo de la solución.

4.3 Planificación del proyecto

Para la fase de desarrollo se definió el trabajo en 3 sprints con una duración de 2 a 3 semanas cada uno, de acuerdo con la cantidad de actividades. Como se estableció con anterioridad, el *product backlog* puede ir cambiando de acuerdo con las necesidades que puedan presentarse. Las tareas se van a realizar de acuerdo con lo establecido en la Tabla 7.

Tabla 7. Planificación de Sprints

Sprint 1			**Sprint 2**		**Sprint 3**	
D11	R21	R52	D31	D43	R54	P31
D12	R22	R53	D32		R55	P41
D13	R23	R54	D33		D41	P42
D14	R31	R55	D34		D42	P51
D15	R32	D21	R54		D43	
D16	R41	D22	R55		D51	
R11	R42	D23	D41		P11	
R12	R51		D42		P21	

A pesar de que cada Sprint posee su propio detalle, se puede encontrar un resumen completo de cada uno de los sprints en los anexos del 1 al 6, en donde se detalla principalmente cual fue la cantidad de horas que se planearon y cuantas se ejecutaron por cada una de las actividades.

4.4 Sprint 1

En este Sprint se configuró el entorno virtual y una revisión de la documentación y ejemplos de las herramientas que se usaron a lo largo del desarrollo del sistema.

4.4.1 Objetivo del Sprint

- Tener un entorno virtual listo para comenzar con el desarrollo
- Revisar la documentación de las herramientas

4.4.2 Sprint Backlog

Tabla 8. Sprint Backlog del Sprint 1

Código	Actividad
D11	Instalación de máquina virtual
D12	Instalación de interprete de Python y programas necesarios
D13	Establecimiento de un ambiente virtual dentro de la máquina
D14	Instalación de OpenCV
D15	Instalación de bibliotecas adicionales
D16	Instalación de editor de texto
R11	Estructura básica de Python
R12	Instalación y uso de bibliotecas
R21	Términos habituales y funcionamiento básico
R22	Tutoriales de la aplicación de las diferentes funciones
R23	Aplicación de funciones revisadas en los tutoriales para entendimiento de su uso y cómo funcionan
R31	Búsqueda de programas aplicados para la solución del problema para usarlos como ejemplo
R32	Extracción de las funciones más usadas para lograr la extracción de una placa vehicular
R41	Revisión de las funciones extraídas de los ejemplos (para que sirven, variables admitidas y valores retornados)
R42	Aplicación de las funciones a imágenes de ejemplo para poder visualizar que función cumplen, cambio de parámetros para visualizar afectación a los resultados
R51	Revisión de ejemplos funcionales en donde se realiza la extracción texto de una imagen
R52	Instalación de las bibliotecas usadas en los ejemplos
R53	Aplicación de las librerías a imágenes de ejemplo para comprobar la facilidad de uso y la eficacia
D21	Aplicación de filtros y funciones usadas para el preprocesamiento de una imagen
D22	Aplicación de filtros y funciones usadas para el preprocesamiento de una imagen que contenga una placa vehicular ecuatoriana
D23	Cambio de parámetros para visualizar cuales brindan mejores resultados en base a la

	necesidad de que la placa sea más visible

4.4.3 Ejecución del Sprint

Para iniciar con la aplicación, fue necesario establecer el lenguaje de programación que fue usado en base a parámetros como: cantidad de bibliotecas y ejemplos disponibles y la rapidez de ejecución en el código.

Para el presente trabajo, se definió el uso de Python 3.10.4, debido a que la implementación es simple, existe una gran variedad de bibliotecas disponibles y tiene un bajo uso de recursos lo que conlleva una mayor velocidad de procesamiento.

Con el lenguaje seleccionado, se escogieron siguientes las bibliotecas para el desarrollo: OpenCV, debido a su amplio catálogo de algoritmos aplicados a visión por computadora, que van a permitir procesar las imágenes; adicionalmente, la librería 're' que permite la aplicación de expresiones regulares, con el fin de verificar que la estructura de la placa obtenida se encuentre acorde a lo definido como una placa vehiculas ecuatoriana.

Previamente al inicio del desarrollo, fue necesario realizar una serie de actividades en las que se incluye la revisión de la documentación y la generación del ambiente en el que se va a trabajar.

Para la creación del ambiente de desarrollo, se decidió utilizar una máquina virtual en el software Virtual Box versión 6.1.34 que puede ser descargado en https://www.virtualbox.org/wiki/Downloads.

El sistema operativo que se utilizó es Ubuntu 22.04, una distribución open source de GNU/Linux, que puede ser descargada en el siguiente enlace https://ubuntu.com/download/desktop. La máquina virtual posee las siguientes especificaciones: 60 GB de almacenamiento, 6 GB de memoria RAM.

Respecto a la instalación de Python, el sistema operativo viene preinstalado con la versión 3.10.4 junto con el instalador de librerías para Python denominado pip.

Para poder mantener aislada la instalación de paquetes, se configuró un ambiente virtual, de este modo, los paquetes se instalan únicamente para el proyecto, sin afectación a otros proyectos, para lo cual haremos uso del siguiente proceso.

- Abrimos un terminal
- Nos dirigimos a la carpeta que va a ser del proyecto

```
$ cd "{carpeta_deseada}"
```

- Generamos el ambiente virtual

```
$ python3 -m venv tesisv1
```

- Entramos en la carpeta del ambiente virtual para poder activarlo

```
$ cd .{nombre del ambiente}/bin
```

- Ejecutamos la inicialización del ambiente

```
$ source activate
```

- Cuando este activado, el terminal va a cambiar, en primer lugar, se mostrará el nombre del ambiente y luego una terminal normal

```
(tesisv1) $ ls
```

Con el ambiente listo, el siguiente paso fue instalar las librerías que se van a usar, principalmente, la librería OpenCV, de la cual se utilizó el paquete opencv-contrib-python que posee las funciones originales junto con módulos de la comunidad. Para ello simplemente se usó pip.

```
(tesisv1) $ pip install {paquete a instalar}
```

Con el ambiente listo, se procedió a la revisión de las funcionalidades que van a ser necesarias a lo largo del proyecto. En primer lugar, se realizó una revisión de la estructura básica para el uso de Python: importar librerías, variables, asignación de valores, uso de funciones, condicionales, bucles. Se lo realizó mediante la revisión de tutoriales y diferentes ejemplos presentados en estos.

Del mismo modo, para comenzar con el uso de OpenCV se revisó la documentación presente en https://docs.opencv.org/4.x/. En esta documentación se encuentra una introducción breve al uso de la librería y adicionalmente, se pueden encontrar ejemplos que van a permitir conocer cuáles son las formas comunes de realizar un procesamiento o trabajar con imágenes.

Posteriormente, mediante la revisión de ejemplos aplicados al reconocimiento de placas vehiculares, se establecieron funciones que son las más usadas, entre las cuales podemos encontrar el cambio de la imagen de RGB a escala de grises que se realiza mediante la siguiente función:

```
cv2.cvtColor({imagen}, cv2.COLOR_BGR2GRAY)
```

La principal idea de esta transformación es reducir la cantidad de información que posee la imagen, en este caso, cada uno de los pixeles tiene tres valores diferentes: azul, verde y rojo; al realizar la transformación, se convertirá en un solo valor que representa la luminosidad que tiene dicho píxel en base a los tres valores anteriores.

Un ejemplo de ello puede visualizarse en la Figura 9. Como puede apreciarse, en la parte inferior de la imagen original puede visualizarse los tres valores que componen un píxel, a diferencia de la escala de grises en donde existe un solo valor por cada píxel, que representa la luminosidad.

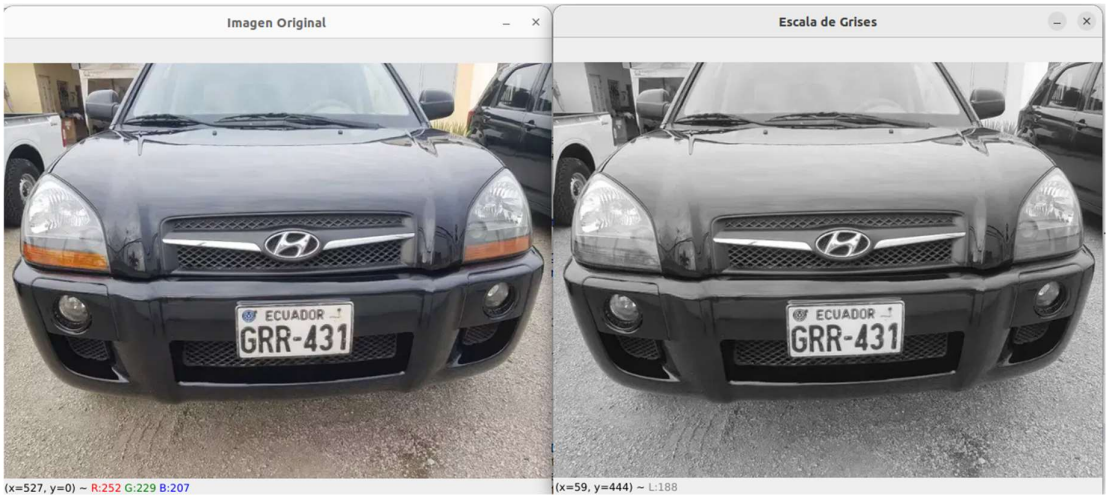

Figura 9. Izquierda: imagen original, en la parte inferior se puede visualizar los valores del píxel seleccionado. Derecha: imagen en escala de grises, en la parte inferior se puede visualizar el valor del píxel seleccionado

Antes de continuar, fue necesario mantener un contexto general de porque se realizó el trabajo y se usaron las siguientes funciones, principalmente, porque todas ellas están enfocadas en extraer la placa vehicular. Por ello, el procedimiento que se siguió buscó mejorar la calidad y el reconocimiento de los bordes, debido a que las placas siempre van a estar encerradas por una figura cuyos bordes deben estar claramente marcados, los pasos para realizar el preprocesamiento de una imagen son los siguientes:

- Realizar el cambio de la imagen de color a una escala de grises
- Aplicar un filtro o un suavizado de la imagen para reducir el ruido que pueda existir en la imagen
- Realizar una detección de bordes con el fin de definir cuáles son relevantes y no tomar en cuenta aquellos que no poseen información

Teniendo en cuenta está en información, el siguiente paso fue el suavizado. Para ello, se pueden usar funciones como aplicación de un umbral o filtro. Para este caso, se usó un filtro bilateral que busca disminuir la cantidad de valores diferentes entre pixeles vecinos, de este modo, las zonas lejos de un borde son más fáciles de reconocer y descartar de acuerdo con los parámetros establecidos.

Es necesario resaltar que el algoritmo debe ser aplicado sobre una imagen en escala de grises, caso contrario, los resultados pueden ser confusos o incorrectos. El algoritmo en acción se lo puede visualizar en la Figura 10.

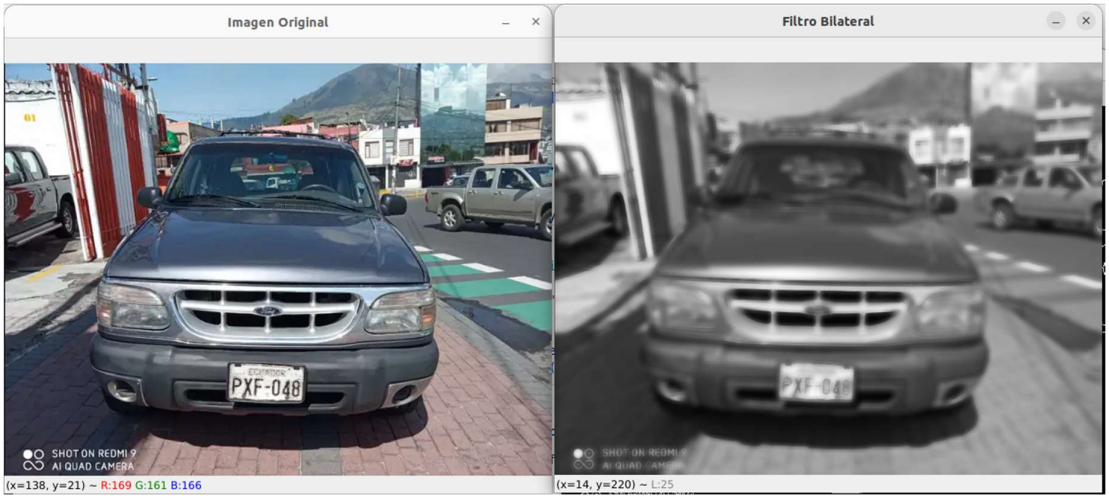

Figura 10. Izquierda: imagen original. Derecha: imagen con el filtro bilateral aplicado

En este ejemplo, el filtro busca acentuar las áreas grandes de luminosidad similar, por ejemplo, zona de la placa vehicular tiene una clara diferencia respecto a sus áreas aledañas que son de un color más oscuro. Del mismo modo, los bordes en donde existe un contraste alto respecto a sus vecinos se mantienen y permiten que el sistema sea capaz de diferenciarlos.

4.4.4 Revisión del Sprint

El entorno virtual fue correctamente instalado y puesto a disponibilidad a lo largo del sprint, debido a que era necesario para las siguientes tareas de desarrollo.

En el caso de las revisiones y desarrollo realizado en este sprint, se aplicó el preprocesamiento que busca eliminar gran parte del ruido presente en la imagen, por lo cual, el resultado final de esta fase sería la imagen en donde se aplica el suavizado.

4.5 Sprint 2

Este sprint está enfocado en el desarrollo de los módulos de detección de la placa vehicular y el reconocimiento de los caracteres que la componen.

4.5.1 Objetivo del sprint

Detectar la posición de la placa vehicular y reconocer el texto, separándole del resto del texto.

4.5.2 Product Backlog

Tabla 9. Product backlog del Sprint 2

Código	Nombre
D31	Aplicación de funciones que permiten identificar los bordes de una imagen
D32	Aplicación de funciones que permiten extraer bordes de acuerdo a parámetros establecidos
D33	Aplicación funciones para extraer bordes de una imagen que contenga una placa vehicular ecuatoriana
D34	Cambio en los parámetros en las funciones que permiten la extracción de las placas y evaluación de los resultados en base a los parámetros
R54	Revisión de la documentación de pytesseract
R55	Revisión de ejemplos de uso en Python
D41	Desarrollo y aplicación de las funciones de pytesseract en una imagen
D42	Aplicación de pytesseract en una imagen donde se puede visualizar una placa vehicular ecuatoriana
D43	Cambio en los parámetros en las funciones que permiten la extracción de las placas y evaluación de los resultados en base a los parámetros

4.5.3 Ejecución del Sprint

Continuando con las actividades, conforme a la arquitectura, se realizó la aplicación de un algoritmo que permita detectar los bordes presentes en la imagen y que excluya todos aquellos que no resultan relevantes para el sistema. Para lograr este objetivo existieron diferentes funciones que pueden aplicarse; sin embargo, evaluando la funcionalidad y la eficacia de cada se estableció que el algoritmo a usar será Canny.

Esta función se aplicó directamente sobre la imagen que pasó por el filtro bilateral. El funcionamiento de este algoritmo es la aplicación de dos operadores de manera secuencial: el primero es el operador Sobel, que busca establecer cuáles son los pixeles o regiones que conforman un borde dentro de la imagen; posteriormente, se realiza la aplicación del operador Canny, el cual hace uso de los bordes ya detectados y los transforma a una amplitud de un solo píxel, además, descarta los bordes que no resultan relevantes en base a los parámetros establecidos. La aplicación de este algoritmo se realiza de la siguiente manera en el código.

```
cv2.Canny(imagen, min_threshold, max_threshold)
```

En este caso, la función recibe tres parámetros principales:

- imagen: imagen que ha pasado por el filtro bilateral
- min_threshold: valor más bajo que puede tener un borde para ser considerado

- max_threshold: valor mínimo que debe tener un borde para ser relevante, marca también el valor más alto que puede tener el borde para ser considerado

Los resultados de la aplicación de este algoritmo se pueden visualizar en la Figura 11. Como se puede notar, la cantidad de bordes existentes es menor respecto a la imagen original, esto permitió que la cantidad de información que necesita ser procesada sea menor y por ello el sistema sea capaz de evaluar las imágenes con mayor rapidez.

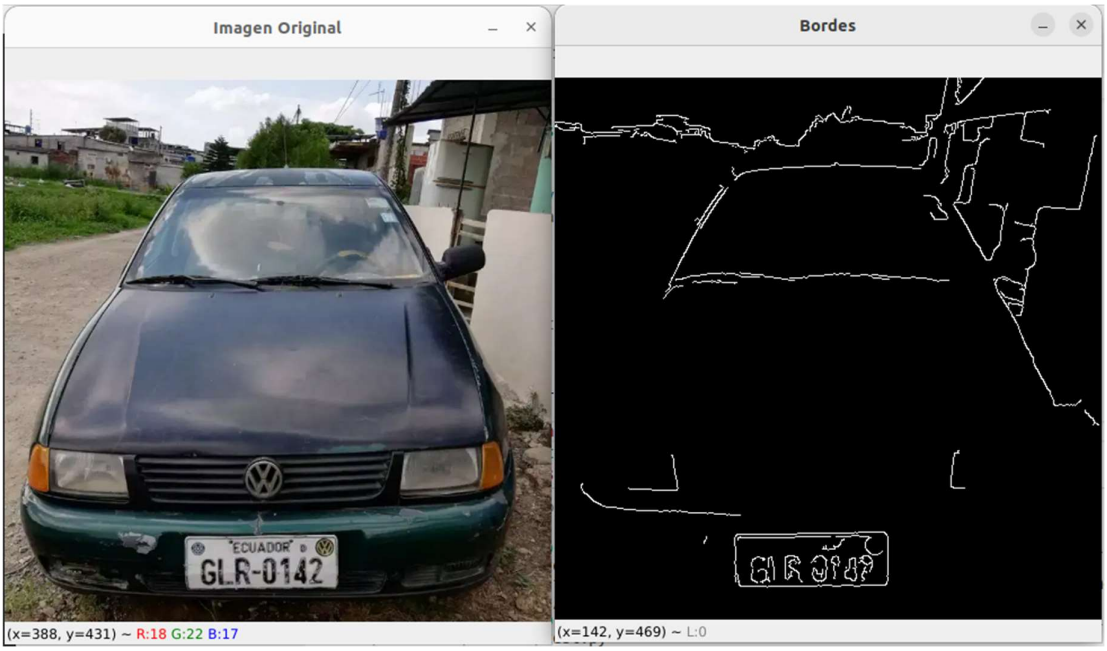

Figura 11. Izquierda: imagen original. Derecha: imagen con el algoritmo Canny ejecutado, bordes relevantes

Con los bordes relevantes seleccionados por el sistema, fue necesario establecer cuál de ellos es la placa vehicular. Para ello, fue importante tener en cuenta la estructura que posee, como podemos ver en la Figura 8, las dimensiones de la placa son: 15,4 cm de alto y 40,4 cm de largo, por lo cual, la placa tiene una forma rectangular (cuatro aristas y cuatro vértices); adicionalmente, es una figura que debería encontrarse cerrada, es decir brecha en los bordes.

Para encontrar la sección donde se encuentra la placa, se siguió un proceso que comienza por la aplicación de la siguiente función:

cv2.findContours(img, jerarquia, formato_contorno)

En este caso, los parámetros que recibe son los siguientes:

- img: imagen donde se aplica la función, en este caso, será aquella que se obtuvo de la función Canny

- jerarquia: debido a la naturaleza de los bordes, puede darse el caso que un borde se encuentre dentro de otro, lo que genera una jerarquía, sin embargo, el presente proyecto hace uso del parámetro cv2. RETR_LIST que retorna todos los contornos como una lista, sin una jerarquía
- formato_contorno: permite escoger la estructura que tendrá el contorno, para este caso, se hizo uso del parámetro cv2.CHAIN_APPROX_SIMPLE que devuelve únicamente los vértices del contorno

Dicha función retorna un *array* con todos los bordes que se encontraron en la imagen, el mismo que se ordenan con la siguiente función:

sorted(array, key, reverse)

donde los parámetros son los siguientes:

- array: array que contiene los valores que se van a ordenar
- key: permite definir bajo que criterio se van a ordenar los valores
- reverse: permite establecer si los valores van de menor a mayor o viceversa

En este caso, el parámetro key fue la función cv2.contourArea que calcula el área comprendida por los contornos, adicionalmente, se ordenan de mayor a menor.

Con el *array* ordenado, se evaluó cada uno de los bordes, el proceso comienza con el cálculo de la longitud del perímetro mediante la siguiente función:

cv2.arcLength(contorno, cerrado)

donde los parámetros son:

- contorno: *array* con los puntos que conforman el contorno
- cerrado: especifica si se va a calcular la longitud de una forma cerrada o simplemente de una curva

El perímetro de la forma nos servirá al momento de reducir la forma del contorno y aproximarlo a un polígono mediante la siguiente función:

cv2.approxPolyDP(contorno, epsilon, cerrado)

donde se recibe:

- contorno: conjunto de datos que va a usarse para realizar la aproximación
- epsilon: define la distancia máxima que puede existir entre la curva aproximada y la curva original
- cerrado: permite a la función definir si la curva tiene sus vértices conectados

La función reduce el número de vértices presentes en el conjunto de datos que conforma cada contorno, con el fin de obtener figuras geométricas con el menor número de lados posible. Resulta importante esta disminución en la información debido a que se usa en cada contorno para poder definir cuál es la placa vehicular.

De manera resumida, reducir el número de vértices de cada contorno permite conocer cuál es el candidato a una placa vehicular, validando que existan únicamente cuatro vértices. En la Figura 12, es posible visualizar la región donde se estableció que se encuentra la placa. La importancia de esta fase radica en que es necesario aislar esta sección para las siguientes etapas con el fin de evitar que caracteres que puedan encontrarse fuera de la placa sean leídos y reconocidos. Adicionalmente, extraer únicamente la sección de la imagen que posee la placa vehicular redujo de manera considerable la información que se envió a las fases siguientes.

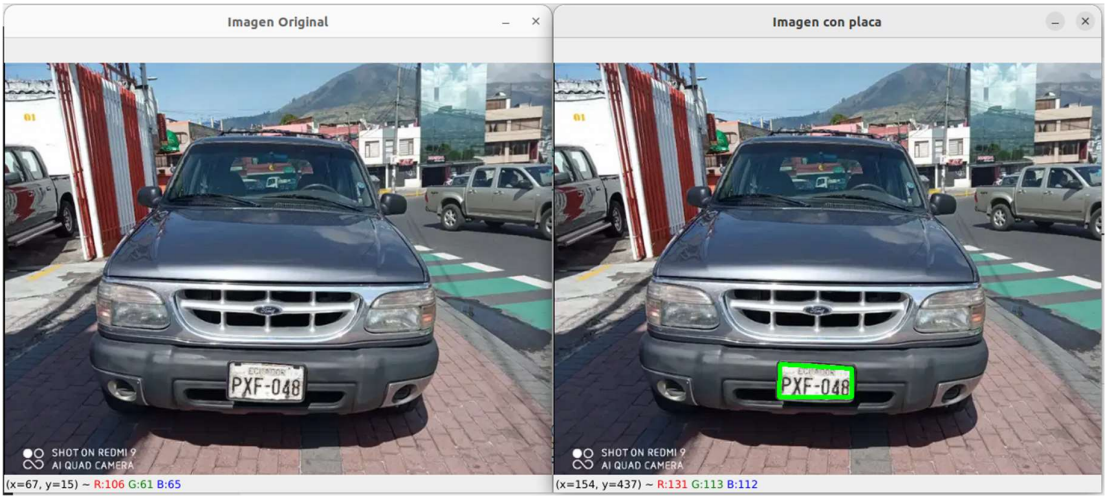

Figura 12. Izquierda: imagen original. Derecha: imagen con la región de la placa resaltada

Finalmente, el reconocimiento del texto consta de dos etapas distintas: segmentaciones de las regiones que poseen caracteres y la extracción del texto de cada una de estas regiones. Con el fin de facilitar el procesamiento, se hizo uso de una librería denominada pytesseract, que abarca las etapas mencionadas anteriormente.

4.5.4 Revisión del Sprint

La extracción de las placas vehiculares fue realizada mediante la detección de bordes y figuras geométricas, en este caso se logro realizar en las primeras imágenes de prueba. Para la extracción de los caracteres se hizo uso de la librería pytesseract, sin embargo, las pruebas realizadas con esta herramienta no dieron buenos resultados debido que la librería es un *wrapper*, una adaptación de la original en lenguaje C++, que permite tener las funcionalidades en Python. Por ello, y también debido a la complejidad que representó su instalación, uso y configuración, se buscó una herramienta cuya implementación sea más simple.

Luego de revisar las herramientas disponibles, se definió que se iba a usar Google Cloud visión como sustito para poder leer los caracteres de la placa vehicular, por lo cual se realizó modificación en el *product backlog.*

4.5.5 Adaptación del Product Backlog

Tomando en cuenta la nueva herramienta seleccionada, se tomarán en cuenta las nuevas actividades que se presentan en la Tabla 10. Nuevas actividades Sprint 2Tabla 10.

Tabla 10. Nuevas actividades Sprint 2

TG	Código	Actividad	Complejidad	Prioridad
R5	R54	Revisión de la documentación de Google Cloud Vision	1	2
	R55	Revisión de ejemplos de uso en Python de Google Cloud Vision	1	1
D4	D41	Desarrollo y aplicación de Google Cloud Vision en una imagen	2	N/A
	D42	Aplicación de Google Cloud Vision en una imagen donde se puede visualizar una placa vehicular ecuatoriana	3	N/A
	D43	Cambio en los parámetros en las funciones que permiten la extracción de las placas y evaluación de los resultados en base a los parámetros	3	N/A

4.6 Sprint 3

El último sprint del desarrollo fue enfocado en desarrollar los cambios realizados en el anterior sprint, hacer la búsqueda y toma de las imágenes y videos que van a ser utilizadas en el proceso de pruebas.

4.6.1 Objetivo del Sprint

Implementar los cambios que sufrió el *product backlog,* tomar las imágenes y videos de vehículos con placas vehiculares visibles, realizar las pruebas necesarias para establecer la eficiencia del sistema y finalmente, generar un producto final que sea instalable.

4.6.2 Sprint Backlog

Código	Actividad
R54	Revisión de la documentación de Google Cloud Vision
R55	Revisión de ejemplos de uso en Python de Google Cloud Vision
D41	Desarrollo y aplicación de Google Cloud Vision en una imagen
D42	Aplicación de Google Cloud Vision en una imagen donde se puede visualizar una placa vehicular ecuatoriana
D43	Cambio en los parámetros en las funciones que permiten la extracción de las placas y evaluación de los resultados en base a los parámetros
D51	Búsqueda de imágenes en motores de búsqueda
P11	Toma de imágenes de vehículos con placas vehiculares
P12	Toma de un video de un vehículo con placas vehiculares para poder realizar la prueba
P21	Ejecución de pruebas sobre las imágenes para evaluar la precisión de la extracción de la placa vehicular
P31	Ejecución de pruebas sobre las imágenes para evaluar la precisión del reconocimiento de la placa vehicular del texto
P41	Ejecución de pruebas sobre las imágenes descargadas para evaluar la precisión de la solución
P42	Ejecución de pruebas sobre las imágenes tomadas para evaluar la precisión de la solución
P51	Ejecución de pruebas sobre el video tomado para evaluar la precisión de la solución

4.6.3 Ejecución del Sprint

Para comenzar con el desarrollo, fue necesario revisar la documentación de la nueva herramientas Google Cloud Vision, la misma que puede ser encontrada en https://cloud.google.com/vision/docs.

Con las funcionalidades y ejemplos revisados fue necesario activar una cuenta en Google Cloud, permitir el uso del Cloud Vision API y finalmente generar unas credenciales para poder hacer el llamado correspondiente.

Con las credenciales creadas, fue posible realizar el llamado mediante código al API, de modo que se envía la imagen y se retorna todo el texto que aparece en ella. En la Figura 13 se muestra la imagen enviada al API de Google y el texto que se recibió como respuesta.

Figura 13. Placa enviada y respuesta del Cloud Vision API

Como se puede visualizar, en el texto obtenido también se lee la parte superior que tiene la palabra “ECUADOR”, sin embargo, esa sección no es necesaria para el sistema, por lo cual, se realizó una extracción del texto de la placa mediante el uso del siguiente comando:

```
re.search(expresion_regular, texto)
```

donde los parámetros recibidos son:

- expresion_regular: secuencia de caracteres que permiten buscar en un texto un patrón definido
- texto: donde se va a buscar el patrón

De este modo, el sistema excluyó cualquier texto adicional que pueda aparecer y se mantuvo únicamente la placa. En la Figura 14 se puede visualizar como el sistema excluyó texto que no pertenece a la placa.

Figura 14. Imagen enviada con extracción exclusiva de la placa

La expresión regular usada es: “[A-Z]{3}\s?\-?\s?\d{3,4}” de donde se obtiene que:

- [A-Z]{3}: deben existir tres letras mayúsculas
- \s?: puede existir un espacio en blanco

- \-?: puede existir un guion
- \d{3,4}: debe tener entre tres y cuatro números

La expresión regular se ajusta de acuerdo con los estándares presentes para las placas vehiculares ecuatorianas, tanto antiguos como nuevos modelos.

El proceso de verificación mediante la realización de pruebas va a ser explicado en una sección a parte debido a la extensión de la explicación necesaria.

Finalmente, para la generación de un instalador se utilizó el paquete *auto-py-to-exe*, el mismo que permite transformar los archivos de Python en ejecutables, con el fin de prescindir de un editor de texto. El funcionamiento de este paquete se da mediante una interfaz gráfica como la que se puede visualizar en la Figura 1.

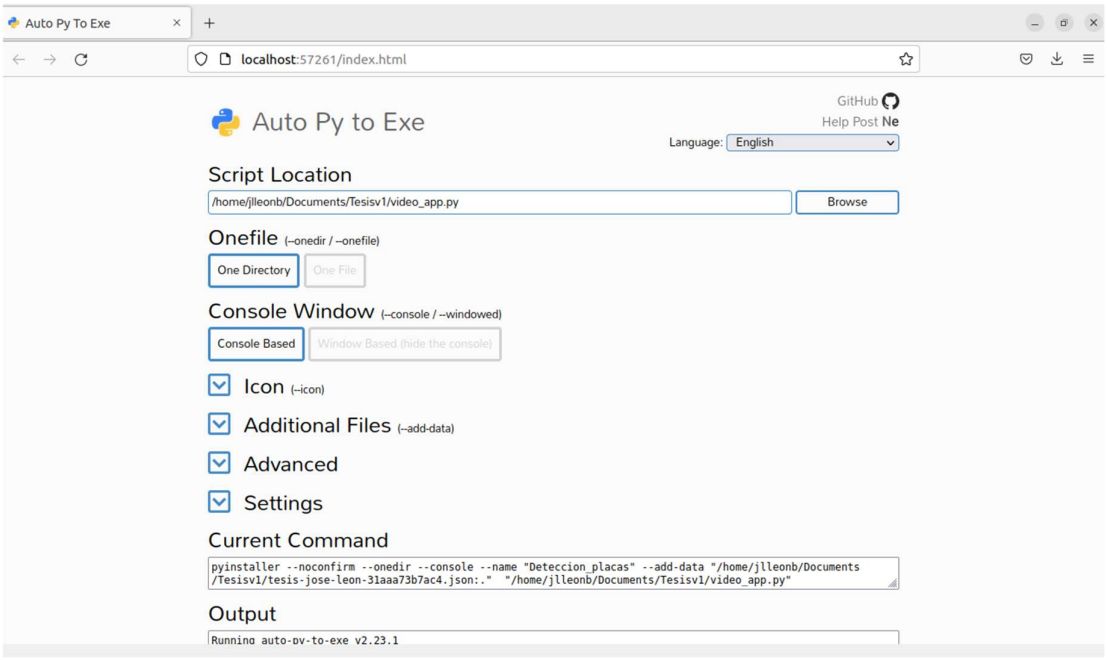

Figura 15. Pantalla de auto-py-to-exe

En primer lugar, se escogió el archivo Python que se deseaba transformar en ejecutable. Posteriormente, se mantuvo la opción de que se genere todo un directorio que sea propio de la aplicación, con el fin de poder agregar archivos adicionales, como la llave para el acceso al API de Google Cloud Vision. Dentro de este directorio, se generó un archivo que al ejecutarse aparecerá la imagen de la cámara y en el momento en el que se detecte que existe una placa, aparecerá un recuadro verde alrededor de la misma, como se puede visualizar en la Figura 16.

Figura 16. Visualización de la aplicación

Adicionalmente, para comprobación de la extracción del texto, se generó un archivo en donde se guardó las placas que han sido reconocidas. En este caso, como se mantuvo mucho tiempo la cámara sobre la placa, esta aparece leída varias veces como se puede ver en la Figura 17.

Figura 17. Archivo de texto con las placas

4.6.4 Revisión del Sprint

El resultado de este sprint ha sido la generación de un archivo ejecutable junto con todas las dependencias para que su ejecución no represente ningún problema para un usuario final.

Debido al paquete usado para poder realizar esta transformación de un archivo Python a un ejecutable, existe una limitante en los sistemas capaces de utilizarla, en este caso, sería posible únicamente desde otro computador Linux.

4.7 Pruebas y Resultados

Para poder comenzar con la realización de las pruebas, fue necesario obtener un conjunto de imágenes de prueba que contengan placas vehiculares ecuatorianas. Estas fueron conseguidas a través de dos métodos:

- **Imágenes propias:** se tomaron imágenes de vehículos de manera frontal en donde se pueda visualizar la placa vehicular en diferentes ángulos
- **Google Images:** se descargaron imágenes de vehículos con vista frontal en donde la placa sea claramente visible.

Este set de imágenes permitió que el proceso de pruebas se desarrolle a lo largo de los sprints, con el fin de evaluar los resultados obtenidos en cada uno de ellos; de este modo, las mejoras necesarias en el desarrollo se pudieron realizar de manera oportuna.

Inicialmente, las pruebas aplicadas fueron para poder evaluar cómo funciona el preprocesamiento de las imágenes y poder visualizar cómo funcionan los algoritmos, es por ello los resultados no pueden ser cuantificados únicamente permitieron o mejor conocimiento de las herramientas y funciones utilizadas.

La siguiente etapa de pruebas evaluó el funcionamiento de los algoritmos que permitían identificar los bordes dentro de la imagen, mantener aquellos que resultan relevantes y definir cuál era la sección en la que se encontraba la placa vehicular; para ello, se hizo uso de un set pruebas de 68 imágenes que fueron procesadas de la siguiente manera: se realizó un recorte de las imágenes para disminuir el área que se tiene que procesar y se cambia el tamaño de todas ellas para que los tamaños sean iguales, en este caso, el tamaño definido fue de 320x180. Esta transformación se puede visualizar en las Figura 18, 16 y 17.

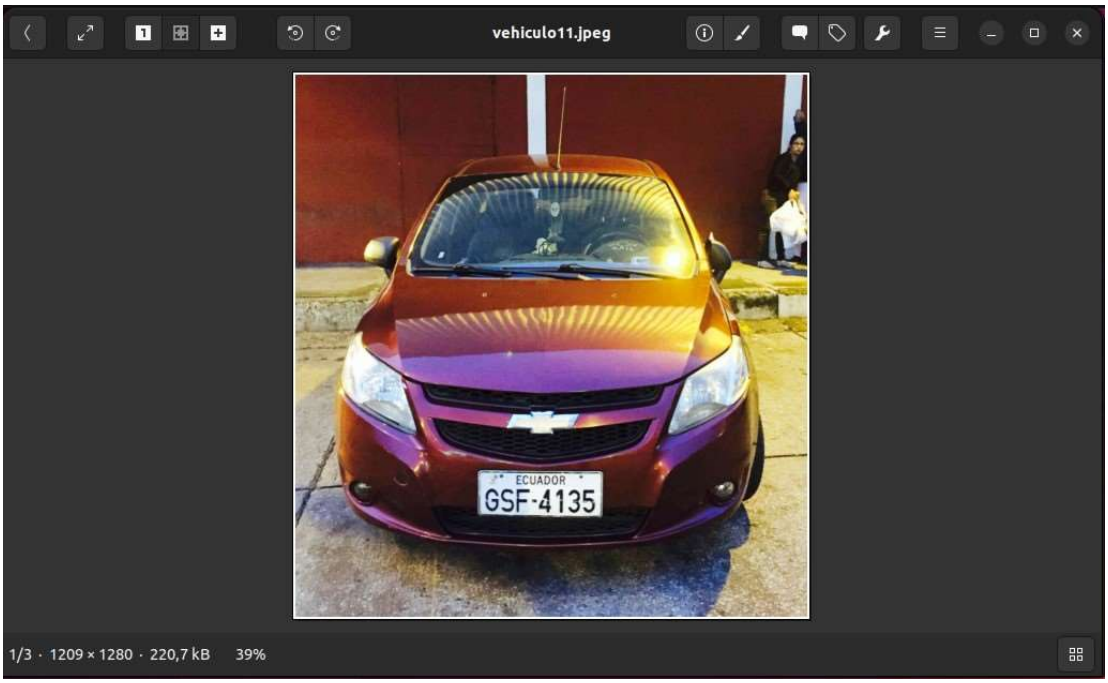

Figura 18. Imagen original con sus dimensiones en la parte inferior izquierda

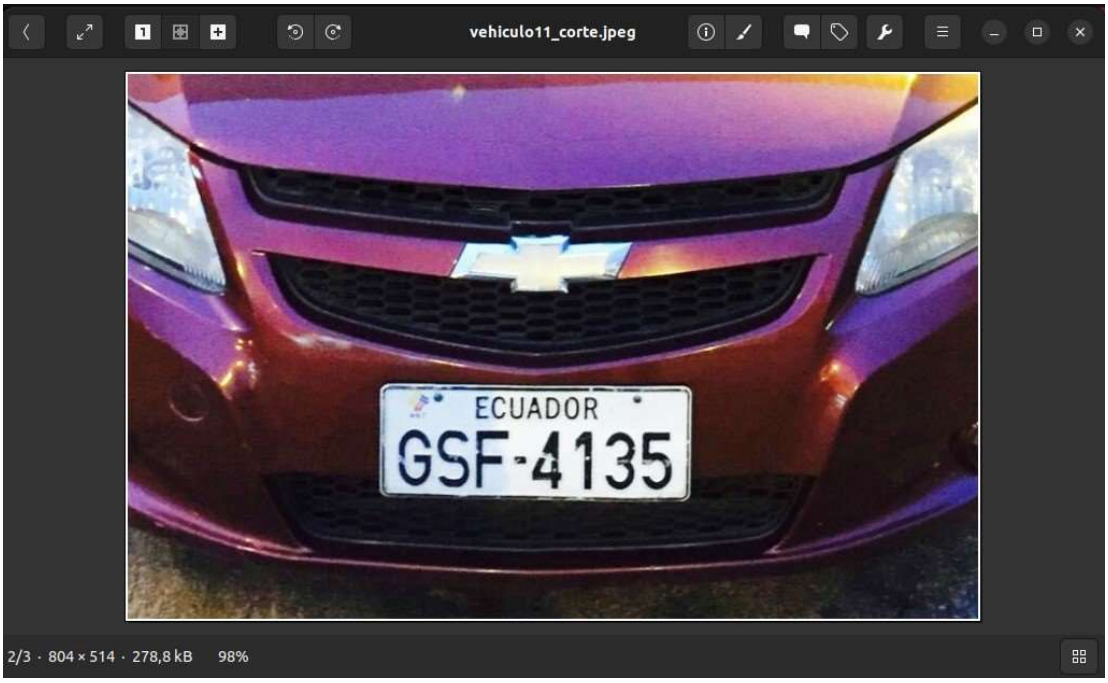

Figura 19. Imagen recortada con sus dimensiones en la parte inferior izquierda

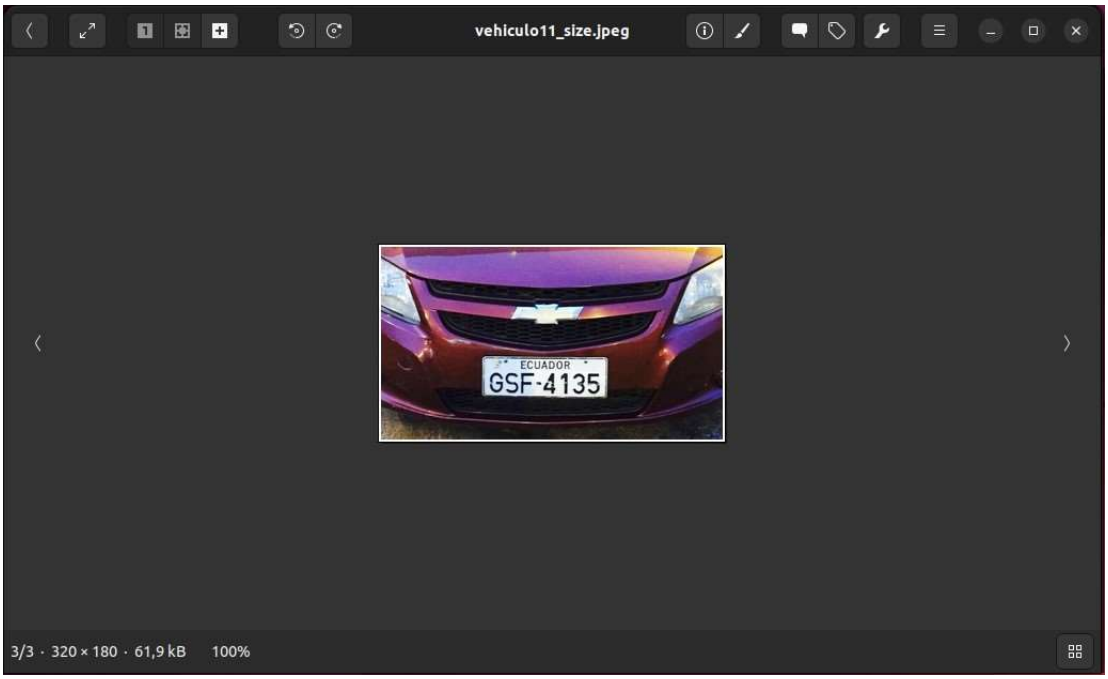

Figura 20. Imagen con tamaño reducido para pruebas con sus dimensiones en la parte inferior izquierda

El set de pruebas posee las siguientes características: las imágenes son de la parte frontal del vehículo, la placa es visible y se encuentra en la posición determinada por ley, no se tomaron en cuenta imágenes de vehículos que tengan la placa provisional brindada por la

autoridad, el estado de las placas es variado (existen placas nuevas, placas que tienen desgaste, placas cuyo nivel de deterioro es notable), las condiciones de luz son variadas, el color de los vehículos es diferente y el ángulo de las imágenes es diferente. Los ejemplos se pueden visualizar en la Figura 21.

Figura 21. Ejemplos de las imágenes usadas

Para esta etapa los resultados obtenidos Se muestran en la Tabla 11. Cómo se puede visualizar, el índice de aciertos de los algoritmos utilizados es del 85%, el 15% restante presenta fallos en la detección principalmente debido a que las condiciones de las imágenes son diferentes, como la iluminación y el ángulo de visión, y la parametrización debe ser diferente para poder hacer una correcta detección.

Tabla 11. Resultados de la herramienta

	Positivos	Negativos
Localización de placa vehicular	85%	15%
Reconocimiento del texto	98%	2%
Total	91.5%	8.5%

Sin embargo, también existen excepciones donde la posición y el estado de la placa no permiten que los algoritmos sean capaces de detectar correctamente los bordes y por ello se imposibilita el posicionamiento de la placa. Un ejemplo de hecho se lo puede visualizar en la Figura 22, donde se muestra los pasos seguidos por el sistema, no obstante, la figura que debería formar la placa no aparece debido a que uno o más bordes no están siendo considerados.

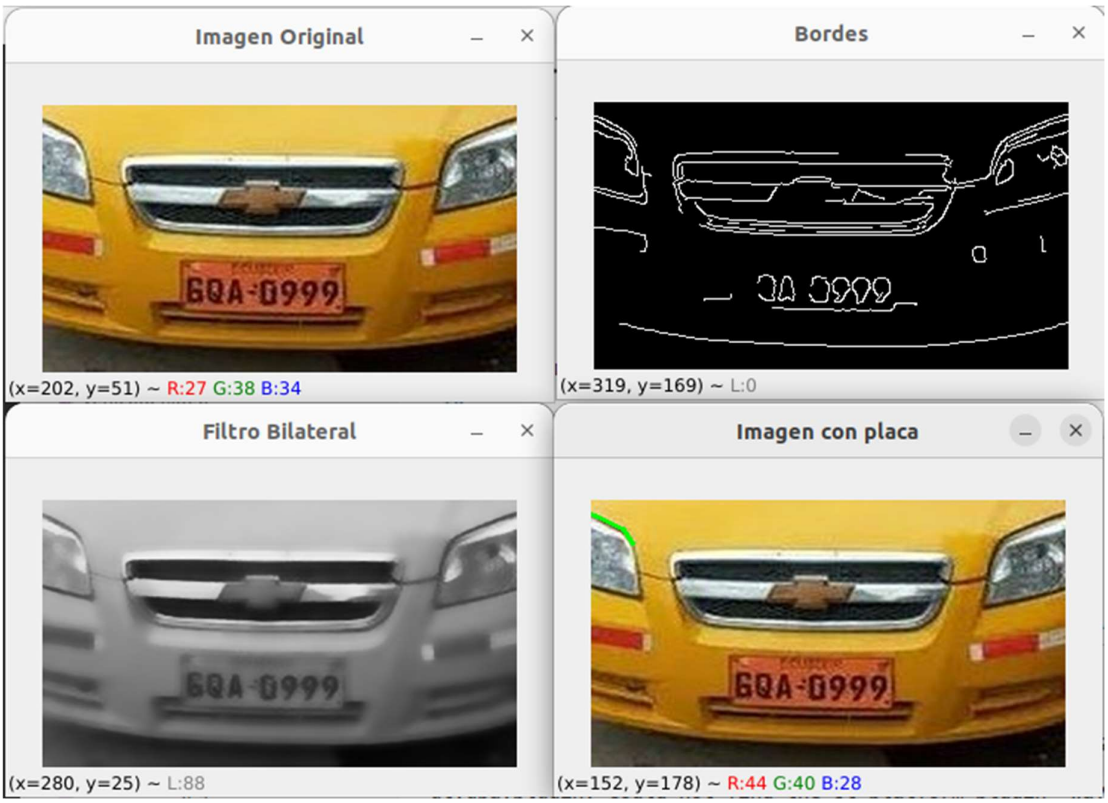

Figura 22. De arriba hacia abajo y derecha a izquierda. A) Imagen original. B) Aplicación del filtro bilateral. C) Bordes con mayor calidad. D) Imagen con placa seleccionada (errónea)

Esta problemática no puede ser resuelta realizando cambios en la parametrización porque al reducir los valores que permiten identificar los bordes relevantes se da lugar a la detección de bordes que anteriormente no se tenía y ello implica un aumento en la cantidad de información que necesita procesarse además de la posibilidad de que puedan aparecer falsos positivos.

Aquellas imágenes donde se pudo obtener la posición exacta de la placa fueron utilizadas en la siguiente etapa de pruebas en donde se iba a realizar la extracción del texto perteneciente a la placa vehicular.

En la última etapa, las pruebas se centraron en verificar que el texto que está siendo extraído de la placa sea el correcto. En primer lugar, se evaluó la precisión del sistema cuando se reciben las imágenes completas para luego realizar la prueba con las secciones seleccionadas de las placas. En ambos casos, la herramienta Google Cloud Vision obtuvo una precisión del 100% en las pruebas realizadas, no obstante, basándose en referencias encontradas, la precisión promedio es de un 98% en distintos escenarios (Dilmegani, 2022).

Es necesario resaltar que a pesar de que las placas pueden estar compuesta por números ceros y letras O, Google Cloud Vision fue capaz de diferenciar e identificar correctamente

las placas. Adicionalmente, en aquellas imágenes de baja calidad o en las que existía una placa difícilmente diferenciable, funcionó de manera correcta.

La Figura 23 muestra un ejemplo de la extracción y reconocimiento del texto de la placa

Figura 23. Derecha: imagen original. Izquierda: texto extraído de la imagen en recuadro rojo

La Figura 24 muestra la correcta extracción y reconocimiento de la letra O dentro del texto obtenido. Figura 25 muestra el mismo ejemplo, pero con el número 0.

Figura 24. Derecha: imagen original. Izquierda (recuadro en rojo): reconocimiento de la letra O

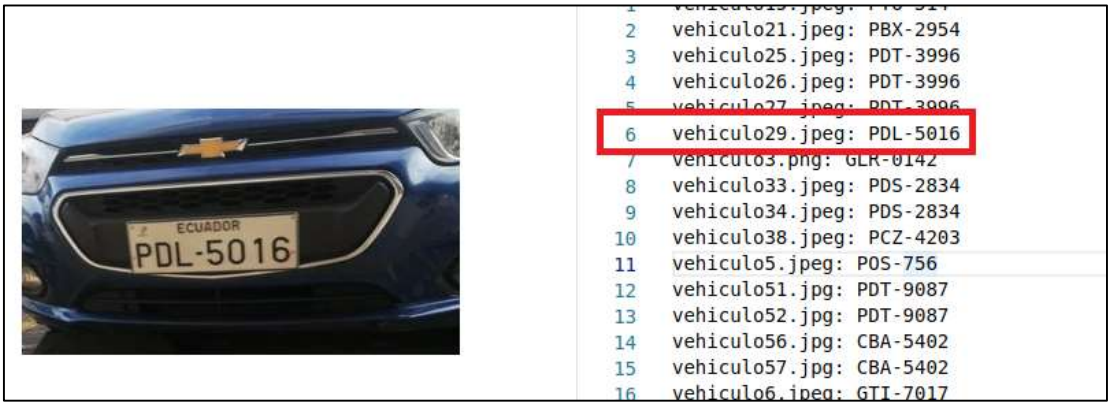

Figura 25. Derecha: imagen original. Izquierda (recuadro en rojo): reconocimiento del número 0

Los resultados se encuentran realizados dentro de un entorno controlado, por lo cual, los resultados poseen una tasa de éxito alta.

Con el fin de corroborar los resultados obtenidos, y mediante la aplicación generada al final del Sprint 3, se realizaron pruebas en tiempo real mediante el uso de una cámara web.

Las pruebas se ejecutaron con tres vehículos que se encontraban disponibles para poder ser grabados, para poder visualizar la factibilidad de la solución, se realizó el video en diferentes ángulos y la iluminación es propia de un parqueadero, debido a que es artificial y la luz natural es baja, principalmente en la parte frontal de los vehículos donde se va a leer la placa vehicular.

La forma de realización del video fue tener una exposición de la placa por aproximadamente 5 segundos para continuar con el siguiente, la idea principal es simular el detenimiento de un carro antes de la entrada de un parqueadero.

El video de la prueba realizada se puede ver en el Anexo VII.

Los resultados de las pruebas se pueden visualizar en la Tabla 12.

Tabla 12. Resultados de las pruebas en tiempo real

	Positivos	Negativos
Localización de placa vehicular	66%	33%
Reconocimiento del texto	100%	0%
Total	83%	16.5%

De acuerdo con los resultados, de 2 de los 3 vehículos se pudo establecer la posición y el texto de la placa, el otro vehículo no fue posible, principalmente porque posee accesorios en la parte frontal que tienen una figura geométrica de cuatro lados, por lo cual nunca logró detectarse la placa.

Tomando en cuenta el tiempo de exposición y la cantidad de un fotograma por segundo, el resultado sería que de 15 fotogramas procesados únicamente 2 fueron dieron un resultado positivo y se logró identificar correctamente la placa, como se puede visualizar en la Figura 26 donde se guardan los textos de las placas vehiculares detectadas.

Figura 26. Placas de pruebas en tiempo real

Finalmente, cabe recalcar que el objetivo del sistema es lograr el reconocimiento de la placa vehicular por lo cual, a pesar de un tiempo de exposición prologando de 5 segundos, únicamente es necesario que en uno de los fotogramas sea posible realizarlo, por lo cual las pruebas fueron exitosas.

5 CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

El desarrollo del presente trabajo fue enfocado en la creación de un sistema de detección de placas vehiculares ecuatorianas.

Mediante el estudio del arte, se pudo establecer que existe una gran variedad de soluciones aplicadas para la detección, localización y lectura de placas vehiculares; que hacen uso de una diversidad de algoritmos. Sin embargo, todos se centran en las especificaciones de las placas de los países en los que residen, por lo cual, es difícil establecer un sistema único y genérico para lograr el objetivo. No obstante, ciertos algoritmos tienen un uso extendido debido a su capacidad para trabajar en distintos escenarios, entre los cuales encontramos los que han sido seleccionados para el desarrollo del presente trabajo.

Asimismo, mediante la aplicación del framework de trabajo Scrum, fue posible trabajar de manera modular y en periodos de tiempo establecidos, debido a la naturaleza de los denominados sprint, parte esencial de este framework. Para el desarrollo, se trabajó en 3 sprints, cada uno con un objetivo establecido, que fue evaluado en la finalización de cada uno de estos, permitiendo que se logren detectar errores en etapas tempranas y corregirlos en el sprint en el que se detectó en el siguiente, de este modo se mantuvo un proceso de mejora continua a la par del desarrollo de las distintas partes que componen el sistema.

A lo largo del desarrollo, fue necesario la revisión de nuevos métodos que permitan la lectura de los caracteres presentes en la placa, debido a la complejidad y poca precisión de los que fueron evaluados y probados, por lo cual, se hizo uso de Google Cloud Vision, que permitió un aumento en la precisión de los resultados.

Finalmente, el sistema fue sometido a una serie de pruebas con distintos escenarios donde se obtuvo una precisión del 85% en la detección y localización de la placa. El 15% restante arrojó resultados erróneos debido a las condiciones que pueden afectar al reconocimiento (calidad de la imagen, condiciones de luz y climáticas, estado de la placa). Adicionalmente, de las imágenes donde fue posible la correcta localización de las placas, se obtuvo un 100% en el reconocimiento de los caracteres que componen el texto de la placa.

Para trabajos futuros, se recomienda el uso de un set ampliado de imágenes con el fin de corroborar los resultados obtenidos. También se recomienda establecer un tamaño específico para las imágenes que se van a trabajar, debido a que los parámetros establecidos funcionan de manera estáticas y al existir un cambio en el tamaño de las imágenes, puede afectar al correcto funcionamiento del sistema.

6 REFERENCIAS BIBLIOGRÁFICAS

Al-Ghaili, A. M., Mashohor, S., Ismail, A., & Ramli, A. R. (2008). A new vertical edge detection algorithm and its application. *2008 International Conference on Computer Engineering & Systems*, 204–209. https://doi.org/10.1109/ICCES.2008.4772997

Anagnostopoulos, C. N. E., Anagnostopoulos, I. E., Loumos, V., & Kayafas, E. (2006). A License Plate-Recognition Algorithm for Intelligent Transportation System Applications. *IEEE Transactions on Intelligent Transportation Systems*, *7*(3), 377–392. https://doi.org/10.1109/TITS.2006.880641

Arasteh, H., Hosseinnezhad, V., Loia, V., Tommasetti, A., Troisi, O., Shafie-khah, M., & Siano, P. (2016). Iot-based smart cities: A survey. *2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC)*, 1–6. https://doi.org/10.1109/EEEIC.2016.7555867

Banco Mundial. (2020, abril). *Desarrollo urbano*. Panorama general. https://www.bancomundial.org/es/topic/urbandevelopment/overview#1

Barriga, J. J., Sulca, J., León, J. L., Ulloa, A., Portero, D., Andrade, R., & Yoo, S. G. (2019). Smart Parking: A Literature Review from the Technological Perspective. *Applied Sciences*, *9*(21), 4569. https://doi.org/10.3390/app9214569

Barth, M., & Boriboonsomsin, K. (2008). Real-World Carbon Dioxide Impacts of Traffic Congestion. *Transportation Research Record: Journal of the Transportation Research Board*, *2058*(1), 163–171. https://doi.org/10.3141/2058-20

Bottino, R. (2009, agosto). *La Ciudad y la Urbanización*. *2*. https://estudioshistoricos.org/edicion_2/rosario_bottino.pdf

Buhus, E. R., Timis, D., & Apatean, A. (2016). Automatic Parking Access using OpenALPR on Raspberry Pi3. *Acta Technica Napocensis. Electronica-Telecomunicatii*, *57*(3), 10–15.

Busch, C., Domer, R., Freytag, C., & Ziegler, H. (1998). Feature based recognition of traffic video streams for online route tracing. *VTC '98. 48th IEEE Vehicular Technology Conference. Pathway to Global Wireless Revolution (Cat. No.98CH36151)*, 1790–1794 vol.3. https://doi.org/10.1109/VETEC.1998.686064

Catherine Kouki. (2021, mayo 7). *Why Should you Convert RGB to Grayscale? | The 6 Reasons – C4RE.GR*. https://c4re.gr/rgb-to-grayscale-conversion/

Chang, S.-L., Chen, L.-S., Chung, Y.-C., & Chen, S.-W. (2004). Automatic License Plate Recognition. *IEEE Transactions on Intelligent Transportation Systems*, *5*(1), 42–53. https://doi.org/10.1109/TITS.2004.825086

Copaja-Alegre, M., & Esponda-Alva, C. (2019). Tecnología e innovación hacia la ciudad inteligente. Avances, perspectivas y desafíos. *Bitácora Urbano Territorial*, *29*(2), 59–70. https://doi.org/10.15446/bitacora.v29n2.68333

Dilmegani, C. (2022, febrero 11). *Best OCR by Text Extraction Accuracy in 2022*. https://research.aimultiple.com/ocr-accuracy/

Drumond, C. (s/f). *Scrum—What it is, how it works, and why it's awesome*. Atlassian. https://www.atlassian.com/agile/scrum

Du, S., Ibrahim, M., Shehata, M., & Badawy, W. (2013). Automatic License Plate Recognition (ALPR): A State-of-the-Art Review. *IEEE Transactions on Circuits and Systems for Video Technology*, *23*(2), 311–325. https://doi.org/10.1109/TCSVT.2012.2203741

Duan, T. D., Tran, P. V., & Hoang, N. V. (2005, febrero). *Building an Automatic Vehicle License-Plate Recognition System*. Conference in Computer Science. https://www.researchgate.net/publication/283162801_Building_an_Automatic_Vehicle_License-Plate_Recognition_System

Eun Ryung Lee, Pyeoung Kee Kim, & Hang Joon Kim. (1994). Automatic recognition of a car license plate using color image processing. *Proceedings of 1st International Conference on Image Processing*, *2*, 301–305. https://doi.org/10.1109/ICIP.1994.413580

Federal Highway Administration: Office of Operations. (2020, marzo 23). *Traffic Congestion and Reliability: Trends and Advanced Strategies for Congestion Mitigation: Executive Summary*. https://ops.fhwa.dot.gov/congestion_report/executive_summary.htm

Fisher, R., Perkins, S., Walker, A., & Wolfart, E. (2003). *Feature Detectors—Sobel Edge Detector*. https://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm

Ganzorig, G. G. (2018, junio 28). How Does Image Filtering Work? *Medium*. https://medium.com/@ganchimeg10/how-does-image-filtering-work-e3ea5eb5b40

Hermawati, F. A., & Koesdijarto, R. (2010). A Real-Time License Plate Detection System for Parking Access. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, *8*(2), 97–106.

Hong-ke Xu, Fu-hua Yu, Jia-hua Jiao, & Huan-sheng Song. (2005). A New Approach of the Vehicle License Plate Location. *Sixth International Conference on Parallel and Distributed Computing Applications and Technologies (PDCAT'05)*, 1055–1057. https://doi.org/10.1109/PDCAT.2005.24

Hsien-Huang P. Wu, Hung-Hsiang Chen, Ruei-Jan Wu, & Day-Fann Shen. (2006). License Plate Extraction in Low Resolution Video. *18th International Conference on Pattern Recognition (ICPR'06)*, 824–827. https://doi.org/10.1109/ICPR.2006.761

Hsi-Jian Lee, Si-Yuan Chen, & Shen-Zheng Wang. (2004). Extraction and recognition of license plates of motorcycles and vehicles on highways. *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, 356-359 Vol.4. https://doi.org/10.1109/ICPR.2004.1333776

IBM. (2022, enero 5). *What Is Optical Character Recognition (OCR)?* https://www.ibm.com/cloud/blog/optical-character-recognition

Jyotsna, Chauhan, S., Sharma, E., & Doegar, A. (2016). Binarization techniques for degraded document images—A review. *2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 163–166. https://doi.org/10.1109/ICRITO.2016.7784945

Khatoun, R., & Zeadally, S. (2016). Smart cities: Concepts, architectures, research opportunities. *Communications of the ACM*, *59*(8), 46–57. https://doi.org/10.1145/2858789

Kozulj, R. (2003, enero). Urbanización, cambio tecnológico y sobrecapacidad estructural: De los años dorados a la globalización. *Banco Nacional de Comercio Exterior*, *53*(1). http://revistas.bancomext.gob.mx/rce/magazines/14/3/RCE.pdf

Martínez, C. (2016, noviembre 1). *Las 7 causas más típicas de congestión vial y las estrategias (exitosas) para enfrentarla, Plataforma Urbana*. https://www.plataformaurbana.cl/archive/2016/11/01/las-7-causas-mas-tipicas-de-congestion-vial-y-las-estrategias-exitosas-para-enfrentarla/

McCoy, K. (2017, julio 12). *Drivers spend an average of 17 hours a year searching for parking spots*. USA TODAY. https://www.usatoday.com/story/money/2017/07/12/parking-pain-causes-financial-and-personal-strain/467637001/

Mihajlovic, I. (2021, septiembre 24). *Everything You Ever Wanted To Know About Computer Vision. Here's A Look Why It's So Awesome.* Medium. https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e

Miyamoto, K., Nagano, K., Tamagawa, M., Fujita, I., & Yamamoto, M. (1991). Vehicle license-plate recognition by image analysis. *Proceedings IECON '91: 1991 International Conference on Industrial Electronics, Control and Instrumentation*, 1734–1738. https://doi.org/10.1109/IECON.1991.239253

OpenCV Documentation. (s/f-a). *OpenCV: Canny Edge Detection*. https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html

OpenCV Documentation. (s/f-b). *OpenCV: Contour Features*. https://docs.opencv.org/4.x/dd/d49/tutorial_py_contour_features.html

Paliy, I., Turchenko, V., Koval, V., Sachenko, A., & Markowsky, G. (2004). Approach to recognition of license plate numbers using neural networks. *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, *4*, 2965–2970. https://doi.org/10.1109/IJCNN.2004.1381137

Patel, C., Shah, D., & Patel, A. (2013). Automatic Number Plate Recognition System (ANPR): A Survey. *International Journal of Computer Applications*, *69*(9), 21–33. https://doi.org/10.5120/11871-7665

Ranger, S. (2020, febrero 3). *What is the IoT? Everything you need to know about the Internet of Things right now*. ZDNet. https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/

Saravanan, C. (2010). Color Image to Grayscale Image Conversion. *2010 Second International Conference on Computer Engineering and Applications*, 196–199. https://doi.org/10.1109/ICCEA.2010.192

Sarfraz, M., Ahmed, M. J., & Ghazi, S. A. (2003). Saudi Arabian license plate recognition system. *2003 International Conference on Geometric Modeling and Graphics, 2003. Proceedings*, 36–41. https://doi.org/10.1109/GMAG.2003.1219663

Shreyamsha Kumar, B. K. (2015). Image fusion based on pixel significance using cross bilateral filter. *Signal, Image and Video Processing*, *9*(5), 1193–1204. https://doi.org/10.1007/s11760-013-0556-9

Silderhuis, H. (2013, agosto 15). *On- and Off-street parking*. Parking Network. https://www.parking-net.com/about-parking/on-and-off-street-parking

Tang Shuang-tong & Li Wen-ju. (2005). Number and Letter Character Recognition of Vehicle License Plate Based on Edge Hausdorff Distance. *Sixth International Conference on Parallel and Distributed Computing Applications and Technologies (PDCAT'05)*, 850–852. https://doi.org/10.1109/PDCAT.2005.174

Telefónica. (2021, noviembre 16). ¿Qué pasa en un minuto en Internet en 2021? *Telefónica*. https://www.telefonica.com/es/sala-comunicacion/que-pasa-en-un-minuto-en-internet-en-2021/

Terán, F. de. (1969). *Ciudad y urbanización en el mundo actual*. Editorial Blume. https://oa.upm.es/11050/

United Nations. (2018a, 2019). *United Nations Population Division | Department of Economic and Social Affairs*. Population Databases. https://www.un.org/en/development/desa/population/publications/database/index.asp

United Nations. (2018b, mayo 16). *Las ciudades seguirán creciendo, sobre todo en los países en desarrollo*. https://www.un.org/development/desa/es/news/population/2018-world-urbanization-prospects.html

Yang, Q., Tan, K.-H., & Ahuja, N. (2009). Real-time O(1) bilateral filtering. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 557–564. https://doi.org/10.1109/CVPR.2009.5206542

Yao-Quan Yang, Jie Bai, Rui-Li Tian, & Na Liu. (2005). A vehicle license plate recognition system based on fixed color collocation. *2005 International Conference on Machine Learning and Cybernetics*, 5394-5397 Vol. 9. https://doi.org/10.1109/ICMLC.2005.1527897

Yin, C., Xiong, Z., Chen, H., Wang, J., Cooper, D., & David, B. (2015). A literature survey on smart cities. *Science China Information Sciences*, *58*(10), 1–18. https://doi.org/10.1007/s11432-015-5397-4

Zanella, A., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. (2014). Internet of Things for Smart Cities. *IEEE Internet of Things Journal*, *1*(1), 22–32. https://doi.org/10.1109/JIOT.2014.2306328

Zhang Sanyuan, Zhang Mingli, & Ye Xiuzi. (2004). Car plate character extraction under complicated environment. *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, *5*, 4722–4726. https://doi.org/10.1109/ICSMC.2004.1401277

Zunino, R., & Rovetta, S. (2000). Vector quantization for license-plate location and image coding. *IEEE Transactions on Industrial Electronics*, *47*(1), 159–167. https://doi.org/10.1109/41.824138

7 ANEXOS

Anexo I.- Resumen de Actividades – Sprint 1
Presenta las actividades que han sido planificadas para la ejecución del sprint 1 junto con las horas estimadas de trabajo, dificultad estimada de la tarea y la prioridad.

Sprint 1						
Responsable: José León						
Duración: 3 semanas						
Objetivo del Sprint: • Tener un entorno virtual listo para comenzar con el desarrollo • Revisar la documentación de las herramientas						
TG	Descripción	Codigo	Actividad	Prioridad	Dificultad	Tiempo
D1	Instalación del ambiente para desarrollo	D11	Instalación de máquina virtual	3	1	2
		D12	Instalación de interprete de Python y programas necesarios	1	1	2
		D13	Establecimiento de un ambiente virtual dentro de la máquina	2	2	1
		D14	Instalación de OpenCV	3	2	4
		D15	Instalación de bibliotecas adicionales	1	1	2
		D16	Instalación de editor de texto	1	1	1
R1	Revisión de Python: estructuras básicas y uso de bibliotecas externas	R11	Estructura básica de Python	1	1	1
		R12	Uso de bibliotecas instaladas	1	1	1
R2	Revisión de la documentación de la librería de OpenCV	R21	Términos habituales y funcionamiento básico	3	1	1
		R22	Tutoriales de la aplicación de las diferentes funciones	2	2	2

		R23	Aplicación de funciones vistas en los tutoriales para entendimiento de su uso y cómo funcionan	2	2	3
R3	Revisión de ejemplos funcionales con OpenCV	R31	Búsqueda de programas aplicados para la solución del problema para usarlos de ejemplo	2	1	2
		R32	Extracción de las funciones más usadas para lograr la extracción de una placa vehicular	3	2	5
R4	Revisión de las funciones de OpenCV más usadas en el reconocimiento de placas vehiculares	R41	Revisión de las funciones extraídas de los ejemplos (para que sirven, variables admitidas y valores retornados)	3	3	10
		R42	Aplicación de las funciones a imágenes de ejemplo para poder visualizar que función cumplen, cambio de parámetros para visualizar afectación a los resultados	3	3	15
R5	Revisión de librerías para extracción de texto de una imagen	R51	Revisión de ejemplos funcionales en donde se realiza la extracción texto de una imagen	2	1	5
		R52	Instalación de las bibliotecas usadas en los ejemplos	2	1	4
		R53	Aplicación de las librerías a imágenes de ejemplo para comprobar la facilidad de uso y la eficacia	3	3	15
D2	Desarrollo del preprocesamiento necesario que permita la	D21	Aplicación de filtros y funciones usadas para el preprocesamiento de una imagen	3	2	5

	identificación de una placa vehicular	D22	Aplicación de filtros y funciones usadas para el preprocesamiento de una imagen que contenga una placa vehicular ecuatoriana	3	2	2
		D23	Cambio de parámetros para visualizar cuales brindan mejores resultados en base a la necesidad de que la placa sea más visible	3	3	10
Total Horas						93

Anexo II.- Resumen de la evaluación – Sprint 1

Sprint 1						
Responsable: José León						
Duración: 3 semanas						
Objetivo del Sprint: • Tener un entorno virtual listo para comenzar con el desarrollo • Revisar la documentación de las herramientas						
TG	Descripción	Código	Actividad	Tiempo estimado	Tiempo ejecutado	Finalizado
D1	Instalación del ambiente para desarrollo	D11	Instalación de máquina virtual	2	3	Si
		D12	Instalación de interprete de Python y programas necesarios	2	2	Si
		D13	Establecimiento de un ambiente virtual dentro de la máquina	1	1	Si
		D14	Instalación de OpenCV	4	6	Si
		D15	Instalación de bibliotecas adicionales	2	2	Si
		D16	Instalación de editor de texto	1	1	Si
R1	Revisión de Python: estructuras básicas y uso de bibliotecas externas	R11	Estructura básica de Python	1	1	Si
		R12	Uso de bibliotecas instaladas	1	1	Si
R2		R21	Términos habituales y funcionamiento básico	1	1	Si

	Revisión de la documentación de la librería de OpenCV	R22	Tutoriales de la aplicación de las diferentes funciones	2	4	Si
		R23	Aplicación de funciones vistas en los tutoriales para entendimiento de su uso y cómo funcionan	3	5	Si
R3	Revisión de ejemplos funcionales con OpenCV	R31	Búsqueda de programas aplicados para la solución del problema para usarlos de ejemplo	2	3	Si
		R32	Extracción de las funciones más usadas para lograr la extracción de una placa vehicular	5	7	Si
R4	Revisión de las funciones de OpenCV más usadas en el reconocimiento de placas vehiculares	R41	Revisión de las funciones extraídas de los ejemplos (para que sirven, variables admitidas y valores retornados)	10	12	Si
		R42	Aplicación de las funciones a imágenes de ejemplo para poder visualizar que función cumplen, cambio de parámetros para visualizar afectación a los resultados	15	20	Si
R5	Revisión de librerías para extracción de texto de una imagen	R51	Revisión de ejemplos funcionales en donde se realiza la extracción texto de una imagen	5	0	No
		R52	Instalación de las bibliotecas usadas en los ejemplos	4	0	No
		R53	Aplicación de las librerías a imágenes de ejemplo para comprobar la facilidad de uso y la eficacia	15	0	No

D2	Desarrollo del preprocesamiento necesario que permita la identificación de una placa vehicular	D21	Aplicación de filtros y funciones usadas para el preprocesamiento de una imagen	5	7	Si
		D22	Aplicación de filtros y funciones usadas para el preprocesamiento de una imagen que contenga una placa vehicular ecuatoriana	2	5	Si
		D23	Cambio de parámetros para visualizar cuales brindan mejores resultados en base a la necesidad de que la placa sea más visible	10	10	Si
Total Horas				93	91	

Anexo III.- Resumen de actividades – Sprint 2

Sprint 2						
Responsable: José León						
Duración: 3 semanas						
Objetivo del Sprint: Detectar la posición de la placa vehicular y reconocer el texto, separándole del resto del texto						
TG	Descripción	Código	Actividad	Prioridad	Dificultad	Tiempo
D3	Desarrollo de la extracción de la placa vehicular en una imagen	D31	Aplicación de funciones que permiten identificar los bordes de una imagen	2	2	5
		D32	Aplicación de funciones que permiten extraer bordes de acuerdo a parámetros establecidos	3	2	10
		D33	Aplicación funciones para extraer bordes de una imagen que contenga una placa vehicular ecuatoriana	3	2	10
		D34	Cambio en los parámetros en las funciones que permiten la extracción de las placas y evaluación de los resultados en base a los parámetros	3	2	15
R5	Revisión de librerías para extracción de texto en una imagen	R54	Revisión de la documentación de pytesseract	2	1	5
		R55	Revisión de ejemplos de uso en Python	1	1	5
D4		D41	Desarrollo y aplicación de las funciones de pytesseract en una imagen	2	2	10

	Desarrollo del reconocimiento del texto en la región de la placa vehicular de una imagen	D42	Aplicación de pytesseract en una imagen donde se puede visualizar una placa vehicular ecuatoriana	3	2	15
		D43	Cambio en los parámetros en las funciones que permiten la extracción de las placas y evaluación de los resultados en base a los parámetros	3	3	15
Total Horas						90

Anexo IV.- Resumen de la evaluación – Sprint 2

Sprint 2						
Responsable: José León						
Duración: 3 semanas						
Objetivo del Sprint: Detectar la posición de la placa vehicular y reconocer el texto, separándole del resto del texto						
TG	Descripción	Código	Actividad	Tiempo estimado	Tiempo ejecutado	Finalizado
D3	Desarrollo de la extracción de la placa vehicular en una imagen	D31	Aplicación de funciones que permiten identificar los bordes de una imagen	5	5	Si
		D32	Aplicación de funciones que permiten extraer bordes de acuerdo a parámetros establecidos	10	11	Si
		D33	Aplicación funciones para extraer bordes de una imagen que contenga una placa vehicular ecuatoriana	10	8	Si
		D34	Cambio en los parámetros en las funciones que permiten la extracción de las placas y evaluación de los resultados en base a los parámetros	15	17	Si
R5	Revisión de librerías para extracción de texto en una imagen	R54	Revisión de la documentación de pytesseract	5	4	Si
		R55	Revisión de ejemplos de uso en Python	5	7	Si
D4		D41	Desarrollo y aplicación de las funciones de pytesseract en una imagen	10	12	Si

	Desarrollo del reconocimiento del texto en la región de la placa vehicular de una imagen	D42	Aplicación de pytesseract en una imagen donde se puede visualizar una placa vehicular ecuatoriana	15	15	Si
		D43	Cambio en los parámetros en las funciones que permiten la extracción de las placas y evaluación de los resultados en base a los parámetros	15	12	No
Total Horas				90	91	

Finalizado el sprint 2, se estableció que la librería pytesseract no obtiene resultados preciosos, en base a las pruebas realizadas, por lo cual se tomó la decisión de realizar un cambio en la herramienta que se va a usar para lograr el reconocimiento del texto de las placas vehiculares

Anexo V.- Resumen de actividades – Sprint 3

Para el sprint 3 se definió el uso de Google Cloud Vision como herramienta para la extracción del texto de la imagen

Sprint 3						
Responsable: José León						
Duración: 3 semanas						
Objetivo del Sprint: Implementar los cambios que sufrió el product backlog, tomar las imágenes y videos de vehículos con placas vehiculares visibles, realizar las pruebas necesarias para establecer la eficiencia del sistema y finalmente, generar un producto final que sea instalable						
TG	Descripción	Código	Actividad	Prioridad	Dificultad	Tiempo
R5	Revisión de librerías para extracción de texto en una imagen	R54	Revisión de la documentación de Google Cloud Vision	2	1	7
		R55	Revisión de ejemplos de uso en Python de Google Cloud Vision	1	1	7
D4	Desarrollo del reconocimiento del texto en la región de la placa vehicular de una imagen	D41	Desarrollo y aplicación de Google Cloud Vision en una imagen	2	2	12
		D42	Aplicación de Google Cloud Vision en una imagen donde se puede visualizar una placa vehicular ecuatoriana	3	2	15
		D43	Cambio en los parámetros en las funciones que permiten la extracción de las placas y evaluación de los resultados en base a los parámetros	3	3	15
D5	Búsqueda de imágenes de placas vehiculares para realizar las pruebas de los ejemplos y funciones aplicados	D51	Búsqueda de imágenes en motores de búsqueda	1	1	5

P1	Captura de imágenes y video de vehículos para poder probar la eficiencia de la solución	P11	Toma de imágenes de vehículos con placas vehiculares	2	1	5
		P12	Toma de un video de un vehículo con placas vehiculares para poder realizar la prueba	2	1	1
P2	Pruebas de la extracción de la placa vehicular de una imagen	P21	Ejecución de pruebas sobre las imágenes para evaluar la precisión de la extracción de la placa vehicular	3	2	5
P3	Pruebas del reconocimiento de la placa vehicular del texto en una imagen	P31	Ejecución de pruebas sobre las imágenes para evaluar la precisión del reconocimiento de la placa vehicular del texto	3	2	5
P4	Pruebas de la solución en imágenes tomadas	P41	Ejecución de pruebas sobre las imágenes descargadas para evaluar la precisión de la solución	3	2	5
		P42	Ejecución de pruebas sobre las imágenes tomadas para evaluar la precisión de la solución	3	2	5
P5	Pruebas de la solución en un video	P51	Ejecución de pruebas sobre el video tomado para evaluar la precisión de la solución	3	2	5
Total Horas						92

Anexo VI.- Resumen de la evaluación – Sprint 3

Sprint 3						
Responsable: José León						
Duración: 3 semanas						
Objetivo del Sprint: Implementar los cambios que sufrió el product backlog, tomar las imágenes y videos de vehículos con placas vehiculares visibles, realizar las pruebas necesarias para establecer la eficiencia del sistema y finalmente, generar un producto final que sea instalable						
TG	Descripción	Código	Tarea	Tiempo estimado	Tiempo ejecutado	Finalizado
R5	Revisión de librerías para extracción de texto en una imagen	R54	Revisión de la documentación de Google Cloud Vision	7	5	Si
		R55	Revisión de ejemplos de uso en Python de Google Cloud Vision	7	5	Si
D4	Desarrollo del reconocimiento del texto en la región de la placa vehicular de una imagen	D41	Desarrollo y aplicación de Google Cloud Vision en una imagen	12	14	Si
		D42	Aplicación de Google Cloud Vision en una imagen donde se puede visualizar una placa vehicular ecuatoriana	15	15	Si
		D43	Cambio en los parámetros en las funciones que permiten la extracción de las placas y evaluación de los resultados en base a los parámetros	15	10	Si

D5	Búsqueda de imágenes de placas vehiculares para realizar las pruebas de los ejemplos y funciones aplicados	D51	Búsqueda de imágenes en motores de búsqueda	5	5	Si
P1	Captura de imágenes y video de vehículos para poder probar la eficiencia de la solución	P11	Toma de imágenes de vehículos con placas vehiculares	5	5	Si
		P12	Toma de un video de un vehículo con placas vehiculares para poder realizar la prueba	1	3	Si
P2	Pruebas de la extracción de la placa vehicular de una imagen	P21	Ejecución de pruebas sobre las imágenes para evaluar la precisión de la extracción de la placa vehicular	5	5	Si
P3	Pruebas del reconocimiento de la placa vehicular del texto en una imagen	P31	Ejecución de pruebas sobre las imágenes para evaluar la precisión del reconocimiento de la placa vehicular del texto	5	5	Si
P4	Pruebas de la solución en imágenes tomadas	P41	Ejecución de pruebas sobre las imágenes descargadas para evaluar la precisión de la solución	5	5	Si
		P42	Ejecución de pruebas sobre las imágenes tomadas para evaluar la precisión de la solución	5	5	Si
P5	Pruebas de la solución en un video	P51	Ejecución de pruebas sobre el video tomado para evaluar la precisión de la solución	5	5	Si
Total Horas				92	87	

Anexo VII.- Link del video con las pruebas en tiempo real de la aplicación:
https://epnecuador-my.sharepoint.com/:v:/g/personal/jose_leon01_epn_edu_ec/EcB0cMbUhOxDhDfDovUrzaEBjiAeErvSOzQdPZEelGWYtg?e=1Cur8C