MEMORIA DEL PROYECTO

ANÁLISIS DEL FLUJO DE TRÁFICO MEDIANTE EL RECONOCIMIENTO DE IMÁGENES

Trabajo de Fin de Grado

INGENIERÍA INFORMÁTICA

VNiVERSiDAD D SALAMANCA

Julio de 2021

AUTOR

Jorge de Huerta Merino

TUTORES

Juan Francisco De Paz Santana

Xuzeng Mao

Gabriel Villarrubia González

CERTIFICADO DE LOS TUTORES

D. Juan Francisco de Paz Santana y D. Gabriel Villarrubia González, profesores del Departamento de Informática y Automática de la Universidad de Salamanca y D. Xuzeng Mao, personal investigador de la universidad.

HACEN CONSTAR:

Que el trabajo titulado “Análisis del flujo de tráfico mediante el reconocimiento de imágenes” ha sido realizado por D. Jorge de Huerta Merino, con el número de documento 44919294-H y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado de la Titulación Grado de Ingeniería Informática de esta Universidad.

Y para que así conste a todos los efectos oportunos.

En Salamanca, a 5 de julio de 2021.

DE PAZ SANTANA, JUAN FRANCISCO (AUTENTICACIÓN)
Firmado digitalmente por DE PAZ SANTANA, JUAN FRANCISCO (AUTENTICACIÓN)
Nombre de reconocimiento (DN): c=ES, serialNumber=76125754D, sn=DE PAZ, givenName=JUAN FRANCISCO, cn=DE PAZ SANTANA, JUAN FRANCISCO (AUTENTICACIÓN)
Fecha: 2021.07.06 09:46:06 +02'00'

D. Juan Francisco de Paz Santana

MAO XUZENG - X2834542E
Firmado digitalmente por MAO XUZENG - X2834542E
Fecha: 2021.07.06 09:50:22 +02'00'

D. Xuzeng Mao

VILLARRUBIA GONZALEZ GABRIEL - 71932845T
Firmado digitalmente por VILLARRUBIA GONZALEZ GABRIEL - 71932845T
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-71932845T, givenName=GABRIEL, sn=VILLARRUBIA GONZALEZ, cn=VILLARRUBIA GONZALEZ GABRIEL - 71932845T
Fecha: 2021.07.06 11:26:02 +02'00'

D. Gabriel Villarrubia González

RESUMEN

En la actualidad, el uso de la visión artificial se ha expandido en multitud de sectores como la medicina, la agricultura y el transporte. Muchos de nosotros cuando pensamos en el transporte, únicamente pensamos en coches autónomos, pero estos no son los únicos avances en el campo del transporte, sino que la propia videovigilancia de las carreteras también se encuentra en continua evolución. Esta evolución se basa en la utilización de cámaras con inteligencia artificial. Según un artículo [38], se ha realizado un estudio el cual apunta que, en 2025, nos encontraremos con un volumen de 155.000 dispositivos de videovigilancia en las carreteras, los cuales utilizan un modelo inteligencia artificial.

El proyecto realizado como Trabajo Fin de Grado consiste en el diseño e implementación de un modelo de visión artificial capaz de identificar diversos tipos de vehículos (Coches, Camiones, Furgonetas), el número de estos vehículos y la velocidad aproximada a la que están circulando. Para obtener una información más fácilmente interpretable por el usuario, se llevará a cabo el procesamiento y presentación de estos datos en unas estadísticas en tiempo real.

Además, a parte de la implementación del modelo de visión artificial, también se ofrece una aplicación web a través de la cual los usuarios podrán ver en tiempo real las imágenes resultantes del análisis, así como las estadísticas anteriormente comentadas. También, en esta aplicación web podremos encontrar una diferenciación clara entre los diferentes roles de los usuarios del sistema.

Podemos concluir, que el objetivo del proyecto es desarrollar un sistema de videovigilancia que facilite la control y seguimiento de las diferentes carreteras, y de esta manera proveer a las autoridades competentes de una herramienta de gran utilidad.

Durante el desarrollo del proyecto se han seguido las prácticas de desarrollo del software indicadas en el proceso unificado. Todo este desarrollo se encontrará detalladamente explicado en los diferentes anexos.

Palabras clave: Yolov4, Visión artificial, Inteligencia artificial, OpenCV, Detección de objetos.

SUMMARY

Nowadays, the use of computer vision has spread to many sectors such as medicine, agriculture and transportation. Many of us when we think of transport, we only think of autonomous cars, but these are not the only advances in the field of transport, but the video surveillance of the roads itself is also in continuous evolution. This evolution is based on the use of cameras with artificial intelligence. According to an article [38], a study has been carried out which indicates that, in 2025, we will find a volume of 155,000 video surveillance devices on the roads, which use an artificial intelligence model.

The project carried out as Final Degree Project consists of the design and implementation of an artificial vision model capable of identifying various types of vehicles (Cars, Trucks, Vans), the number of these vehicles and the approximate speed at which they are circulating. To obtain information that is more easily interpretable by the user, the processing and presentation of these data will be carried out in real time statistics.

In addition, apart from the implementation of the artificial vision model, a web application is also offered through which users can see in real time the images resulting from the analysis, as well as the previously mentioned statistics. Also, in this web application we can find a clear differentiation between the different roles of the users of the system.

We can conclude that the objective of the project is to develop a video surveillance system that facilitates the control and monitoring of the different roads, and in this way provide the competent authorities with a very useful tool.

During the development of the project, the software development practices indicated in the unified process have been followed. All this development will be explained in detail in the different annexes.

Keywords: Yolov4, Computer Vision, Artificial Intelligence, OpenCV, Object Detection.

CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

1.INTRODUCCIÓN

En la actualidad, el uso de la visión artificial se ha esparcido en multitud de sectores como la medicina, la agricultura y el transporte. Muchos de nosotros cuando pensamos en el transporte, únicamente pensamos en coches autónomos, pero estos no son los únicos avances en el campo del transporte, sino que la propia videovigilancia de las carreteras también se encuentra en continua evolución. En esta evolución se encuentra la utilización de cámaras con inteligencia artificial. Según un artículo [38], se ha realizado un estudio el cual apunta que, en 2025, nos encontraremos con un volumen de 155.000 dispositivos de videovigilancia en las carreteras, los cuales utilizan un modelo inteligencia artificial.

El proyecto realizado como Trabajo Fin de Grado consiste en el diseño e implementación de un modelo de visión artificial capaz de identificar diversos tipos de vehículos (Coches, Camiones, Furgonetas), el número de estos vehículos y la velocidad aproximada a la que están circulando. Para obtener una información más fácilmente interpretable por el usuario, se llevará a cabo el procesamiento y presentación de estos datos en unas estadísticas en tiempo real.

Además, a parte de la implementación del modelo de visión artificial, también se ofrece una aplicación web a través los usuarios podrán ver en tiempo real las imágenes resultantes del análisis, así como las estadísticas anteriormente comentadas. También, en esta aplicación web podremos encontrar una diferenciación clara entre los diferentes roles de los usuarios del sistema.

Podemos concluir, que el objetivo del proyecto es desarrollar un sistema de videovigilancia que facilite la control y seguimiento de las diferentes carreteras, y de esta manera proveer a las autoridades competentes de una herramienta de gran utilidad.

El presente documento tiene como objetivo explicar los aspectos más relevantes del desarrollo del proyecto como Trabajo de Fin de Grado.

- **Estado del arte**: Se describen algunos de las tecnologías similares disponibles actualmente en el mercado.
- **Objetivos**: Se especifican los objetivos que persiguen la realización del proyecto.
- **Conceptos teóricos**: Se describen los conceptos teóricos necesarios para la compresión y entendimiento del proyecto.
- **Técnicas y herramientas**: Documentación de las técnicas y herramientas utilizadas durante el desarrollo.
- **Aspectos relevantes del desarrollo**: Aspectos más relevantes durante el desarrollo del proyecto.
- **Limitaciones del prototipo**: Se especifican las limitaciones encontraras durante el desarrollo del proyecto.
- **Conclusiones y líneas de trabajo futuras**: Se recogen las conclusiones como consecuencia del desarrollo del proyecto, así como las posibles ampliaciones y mejoras de este.
- **Bibliografía**: Enumeración de los diferentes recursos accedidos durante el desarrollo.

Este documento se encuentra cumplimentado con los siguientes anexos:

- **Anexo I** - **Planificación temporal**: Aspectos relacionados con la planificación temporal y la estimación de esfuerzos.
- **Anexo II** – **Especificación de requisitos**: Especificación de los requisitos del sistema a desarrollar.
- **Anexo III** – **Análisis de requisitos del software**: Documentación relacionada con la fase de análisis del proyecto.
- **Anexo IV** – **Diseño del sistema software**: Documentación de la fase de diseño del proyecto.
- **Anexo V** – **Documentación técnica**: Documentación técnica del código desarrollado.
- **Anexo VI** – **Manual de usuario**: Explicación de forma detallada de todas las funcionalidades desarrolladas, con el objetivo de guiar a los usuarios durante su proceso de aprendizaje.

2.ESTADO DEL ARTE

Actualmente podemos encontrar diferentes aplicaciones de inteligencia artificial en el sector de la seguridad vial y del transporte. En este apartado se van a comentar algunas de las ideas más relevantes disponibles en el mercado.

2.1 CÁMARAS DE DETECCIÓN DE TELÉFONOS MÓVILES

[9] En Nueva Gales del Sur (Australia) se ha implantado un sistema de videovigilancia capaz de detectar si un conductor está utilizando de manera ilegal el móvil mientras conduce.

Este novedoso sistema de vigilancia opera independientemente de la hora del día o de las condiciones del tiempo. Además, a través de estas cámaras también se está comenzando a realizar la detección de las infracciones relacionadas con el cinturón de seguridad. En este último caso los motoristas serian una excepción.

Figura 1. Detección de un conductor con el móvil.

El modelo de inteligencia artificial encargado de la detección de los teléfonos móviles revisa cada una de las imágenes recibidas por las cámaras y detecta los posibles conductores, los cuales se encuentren utilizando algún tipo de dispositivo móvil durante la conducción, esto se puede observar en la *Figura 1*. El resto de las imágenes en la que el modelo de inteligencia artificial no ha detectado ningún tipo de infracción serán completamente descartadas.

Hay que destacar, que las imágenes que contengan algún tipo de infracción serán revisadas por personal autorizado.

2.2 TESLA AUTOPILOT

[10] Tesla, una de las principales empresas centradas en la venta de vehículos completamente eléctricos, actualmente vende vehículos con la capacidad de autoconducción. Para realizar esta hazaña ha sido necesaria la utilización de diferentes técnicas y tecnologías.

Figura 2. Cámaras de un coche tesla.

Construido sobre una red neuronal, el sistema de autoconducción utiliza cámaras y sensores de ultrasonidos (*Figura 2*) para detectar todos los elementos y cambios que puedan ocurrir alrededor del vehículo. El uso de esta tecnología permite al conductor tener un mayor conocimiento de su entorno, que en otros casos no tendría. En la *Figura 3* se puede observar el análisis que realiza el autopilot de tesla.

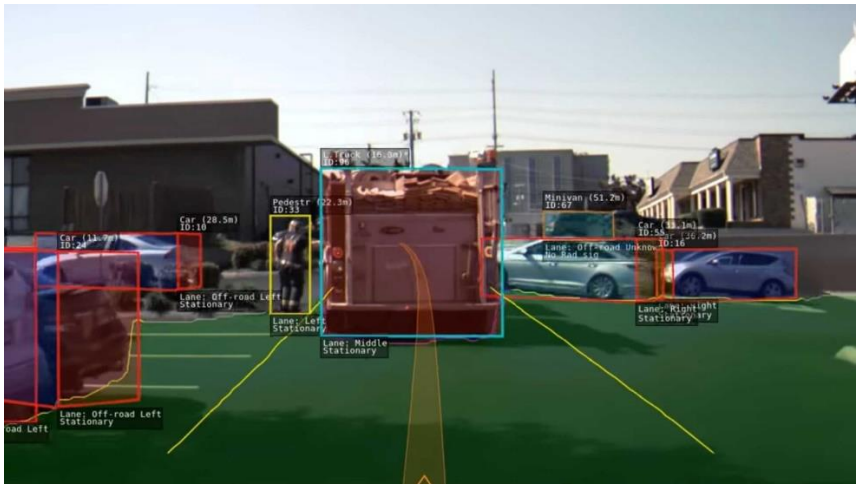

Figura 3. Funcionamiento del sistema autoconducción de Tesla.

Para realizar este proceso, Tesla incorpora en todos sus vehículos un ordenador el cual provee toda la potencia computacional necesaria para poder realizar este tipo de análisis en tiempo real. Dicho ordenador utiliza una arquitectura específica denominada “Hydranets”.

2.3 TRAFICAM AI

[49] TrafiCam AI es una de las soluciones ofertadas por la empresa Teledyne FLIR. Este es un sistema diseñado para detectar de forma fiable a los usuarios de las carreteras. Estas detecciones se realizan a través de un sensor HD e inteligencia artificial, permitiendo de esta manera la supervisión del tráfico en entornos más complicados.

Esta tecnología es capaz de realizar el seguimiento a múltiples objetos permitiendo el control preciso del tráfico en los diferentes cruces, recopilando datos de tráfico detallados con el fin de tomar las mejores decisiones de planificación urbana.

En la *Figura 4,* se puede observar un ejemplo del funcionamiento de uno de los dispositivos.

Figura 4. Funcionamiento de TrafiCam AI

Como se observa en la anterior figura, esta tecnología permite la detección y clasificación de los diferentes vehículos, así como el espacio que estos ocupan y la velocidad a la que se mueven.

3.OBJETIVOS

En este apartado se van a describir los diferentes objetivos que van a ser necesarios cumplir para la correcta realización del Trabajo de Fin de Grado. Se va a realizar una breve descripción de los objetivos que debe cumplir el sistema, así como una descripción de los objetivos personales perseguidos durante el desarrollo.

3.1 OBJETIVOS DEL SISTEMA

El principal objetivo del sistema es el diseño e implementación de una aplicación web a través de la cual poder visualizar en tiempo real las imágenes de diferentes cámaras de tráfico después de ser analizadas por un modelo de inteligencia artificial.

El desarrollo del proyecto se considerará un éxito una vez que se satisfagan los siguientes objetivos:

- **Gestión de usuarios**: Se deberá permitir la gestión de los diferentes usuarios que interactúen con el sistema. Pudiendo diferenciar el tipo de usuarios en función de los permisos en el sistema.
- **Gestión de cámaras:** El sistema permitirá llevar a cabo la gestión de la información de las diferentes cámaras de tráfico utilizadas por este, así como la información que estas envíen.
- **Información en tiempo real:** El sistema deberá permitir la visualización de las imágenes tomadas por las diferentes cámaras de tráfico en tiempo real.
- **Gestión de estadísticas**: El sistema deberá permitir a los usuarios, la observación de la información generada como consecuencia del análisis de las imágenes recibidas de las cámaras, en forma de gráficos y estadísticas.
- **Geolocalización de las cámaras**: La aplicación web deberá permitir la visualización de la localización aproximada de cada una de las cámaras utilizadas por el sistema.
- **Gestión de notificaciones**: El sistema deberá permitir el envío y visualización de diferentes notificaciones del sistema.

3.2 OBJETIVOS PERSONALES

En este apartado se van a describir los diferentes objetivos personales que me han llevado a realizar este proyecto.

Uno de los objetivos que me ha llevado a realizar este tipo de proyecto, es la posibilidad de poder entrar en contacto y conocer las diferentes técnicas necesarias para el entrenamiento de un modelo capaz de detectar diferentes elementos en una imagen. Otra de las razones para realizar este proyecto es debido a que, desde mi punto de vista, la inteligencia artificial me parece uno de los campos más fascinantes dentro de la informática.

Como último punto, la realización de este proyecto me ha permitido poner a prueba todos los conocimientos adquiridos a lo largo del Grado de Ingeniería Informática. También me ha permitido experimentar el desarrollo de un producto software desde sus inicios.

4. CONCEPTOS TEÓRICOS

En este apartado se detallarán alguno de los conceptos teóricos utilizados y citados a lo largo del desarrollo del proyecto.

4.1 VISIÓN ARTIFICIAL

La visión artificial [31] es uno de los subcampos de la Inteligencia artificial que incluye métodos encargados del procesamiento, análisis e interpretación de imágenes, con el fin de obtener información útil que pueda ser utilizada por un ordenador. Con este análisis se no se trata únicamente de imitar la capacidad de la visión humana, sino que también se intenta realizar una aproximación al comportamiento del cerebro humano.

Por lo tanto, el concepto de visión artificial se basa en enseñar a un ordenador o sistema a analizar una imagen píxel a píxel, y determinar y clasificar los elementos que se encuentran en la misma.

Hasta no hace mucho la capacidad de la visión artificial era muy limitada, hasta que los métodos de Deep Learning y los avances en campos de la Inteligencia artificial han permitido un aumento de la capacidad que la visión artificial anteriormente no poseía. En especial, lo que ha generado este aumento ha sido la utilización de Redes Neuronales Convolucionales o CNN.

En la actualidad, hay una gran cantidad de campos que utilizan tecnologías de reconocimiento de imágenes, entre las destacables se encuentra la Robótica. Esto ocurre debido a que para que un robot o autómata sea, dentro de sus posibilidades, autónomo o independiente este necesitará la capacidad de poder percibir y clasificar la información de su entorno. Por ejemplo, algunos robots implementan esta tecnología para poder evitar accidentes o choques innecesarios con elementos de su entorno.

Entre las diferentes tareas que se puede realizar con la visión artificial se van a destacar las siguientes:

- **Clasificación de imágenes**: Clasificación de imágenes dentro de un conjunto finitos de casos. Por ejemplo, clasificar una foto en función de si contiene un perro o un gato.
- **Detección de objetos**: Similar a la clasificación, pero en este caso permite detectar los objetos y su localización en la imagen.
- **Seguimiento de objetos en vídeo**: Seguimiento de los objetos que se detectan en cada fotograma del vídeo analizado.

Como se ha podido observar, la visión artificial tiene una gran versatilidad en cuenta a su uso y aplicación en diferentes campos.

4.2 DEEP LEARNING

Deep learning [1][32] es un subcampo de aprendizaje automático el cual intenta aprender abstracciones de alto nivel en los datos proporcionado a través de la utilización de arquitecturas jerárquicas. Este enfoque utiliza estructuras que se asemejan al cerebro de los mamíferos, presentando capas de unidades de procesos (Neuronas artificiales) que se especializan en la detección y diferenciación de patrones en las imágenes. En el nivel inicial de la estructura jerárquica se aprende algo simple y se lo pasa al siguiente nivel jerárquico. El siguiente nivel jerárquico recibirá la información del anterior nivel para poder formar unos datos más complejos, una vez finalizado se lo envía al siguiente nivel, y así sucesivamente. Esta estructura jerárquica se puede observar en la *Figura 5.*

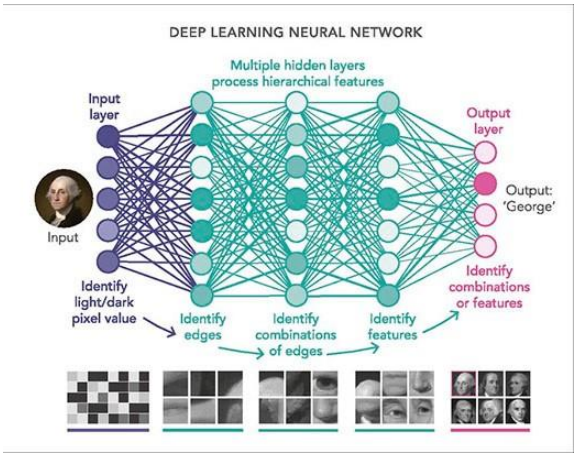

Figura 5. Ejemplo del funcionamiento interno de una red neuronal de Deep Learning.

El Deep learning ha sido ampliamente utilizados en múltiples campos de la inteligencia artificial como [4] visión artificial, [2,3] procesamiento del lenguaje natural y muchos más.

Este aumento de la utilización del Deep Learning se debe principalmente a las siguientes razones:

- **Aumento de la capacidad de procesamiento de los chips actuales**: Nacimiento de bibliotecas y librerías como CUDA, las cuales permiten el aprovechamiento de las capacidades de computación que nos proveen las tarjetas gráficas de Nvidia.
- **Disminución de los costes del hardware**: El precio de un hardware potente se ha reducido considerablemente a lo largo de los años.
- **Nuevos algoritmos de aprendizaje automático.**

4.2.1 RED NEURONAL CONVOLUCIONAL (CNN)

Gran parte de la revolución que se ha vivido en los últimos años dentro del Deep learning, en especial dentro de la visión artificial, se lo debemos principalmente al desarrollo de un tipo de red neuronal especializada en la identificación de patrones y

elementos característicos de una imagen, a esta red se la denomina Red Neuronal Convolucional o CNN.

Las redes neuronales convolucionales se encuentran inspiradas por el sistema de visión del que disponemos los seres vivos, a través del cual al ver una imagen podemos identificar y clasificar el contenido de dicha imagen. Este mismo mecanismo se intenta representar mediante las CNN. [1] El flujo de procesamiento que se sigue por una CNN se puede observar en la *Figura 6.*

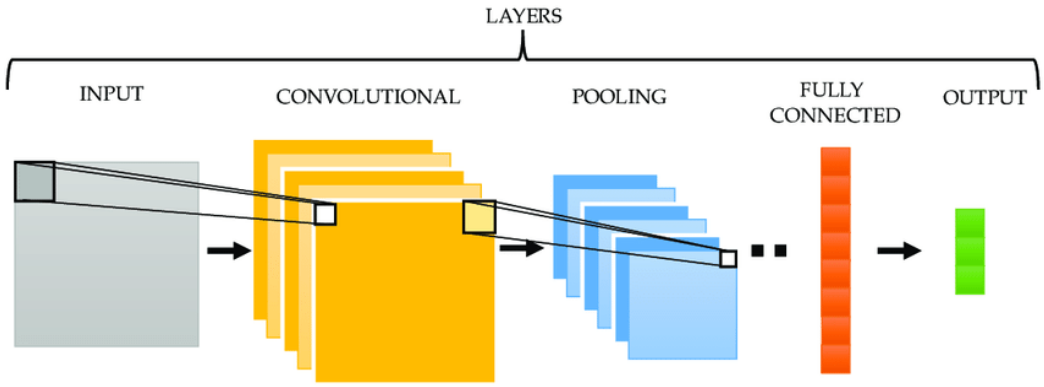

Figura 6. Funcionamiento de una Red Neuronal Convolucional.

[5] Una de las principales diferencias de la red neuronal convolucional con una red multicapa es la utilización de su estructura espacial. En el caso de las redes multicapa estas irán píxel por píxel analizando la imagen de entrada, mientras que en las redes convolucionales no se tratan los píxeles de manera independiente, sino que se tiene en cuenta la posición del píxel y las de sus vecinos más próximos. Esta es la idea en la que se basan las diferentes redes neuronales convolucionales.

Una red convolucional se encuentra caracterizada por la utilización de un tipo de capa denominada convolucional, donde se realizan un determinado tipo de operaciones matemáticas, a este proceso se le denomina convolución. Este proceso se puede observar en la *Figura 7.*

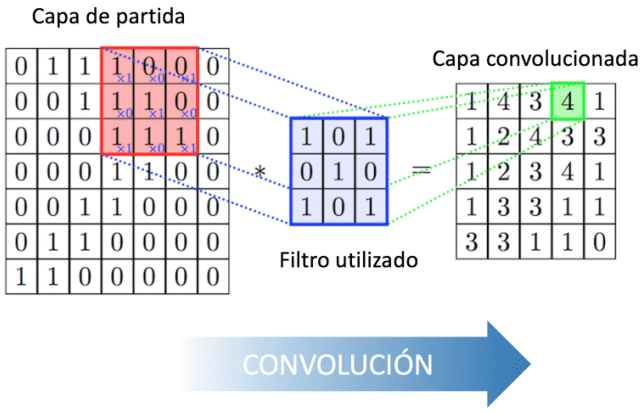

Figura 7. Proceso de convolución.

La convolución consiste básicamente en la aplicación de un filtro o kernel sobre la imagen original, una vez aplicado se multiplicará y sumará los valores de la imagen original con los valores del filtro seleccionado, dando como resultado el nuevo valor de la capa convolucionada. Este proceso dará como resultado una nueva imagen diferente a la original.

Para poder realizar el proceso de detección de ciertos objetos en una imagen, la red neuronal necesitará atravesar un proceso de aprendizaje de los diferentes filtros a aplicar, este aprendizaje permitirá a la red neuronal la identificación de los diferentes patrones de una imagen. Este proceso de detección se puede observar en la *Figura 8.*

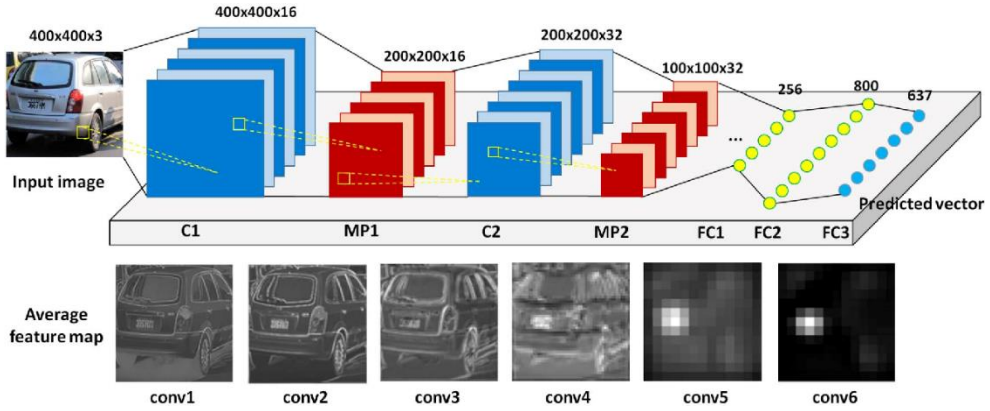

Figura 8. Proceso de detección de matrículas de coches.

A la imagen recibida por la red neuronal se le aplicarán sucesivas operaciones de convolución, dando como resultado un mapa de características. Este mapa de características nos indicará en que parte de la imagen se ha detectado la característica buscada por el filtro aplicado, indicando la activación de este mediante la utilización de pixeles blancos. En el caso de la detección de la matrícula del coche, se puede observar cómo los pixeles pertenecientes a la zona en la que se encuentra la matricula presentarán ciertos tonos blancos.

4.3 YOLO (YOU ONLY LOOK ONCE)

[6] Los sistemas de detección actual reutilizan clasificadores para poder realizar el proceso de detección. Estos sistemas, para poder detectar un objeto, utilizan un clasificador específico para dicho objeto y lo evalúan en diferentes localizaciones y escalas de la imagen a analizar. Sistemas como DPM utilizan un enfoque de ventana deslizante, donde el clasificador analiza de manera uniforme toda la imagen [7].

Los enfoques más recientes, como R-CNN, utilizan un método en el que, en la imagen a analizar, se proponen unas cajas delimitadoras donde se va a ejecutar el clasificador. Después de realizar el proceso de detección, se realizar un proceso de post procesamiento para eliminar posibles detecciones duplicadas [8]. Este tipo de procesos son lentos y difíciles de optimizar debido a que cada elemento independiente hay que entrenarlo de forma separada.

En el caso de YOLO, una única red neuronal convolucional se encarga de predecir múltiples cajas delimitadoras y las probabilidades de cada clase para dichas cajas. Este proceso se puede observar en la *Figura 9.*

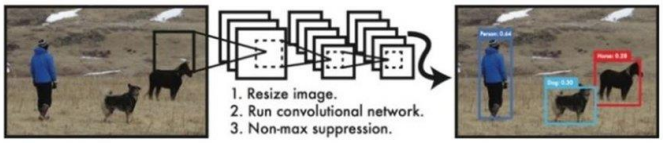

Figura 9. Sistema de detección utilizado por YOLO.

Este modelo unificado presenta diferentes ventajas frente a los modelos tradiciones de detección de objetos.

- **YOLO es extremadamente rápido**: YOLO trata la detección de cada frame como si de un problema de regresión se tratara, evitando de esta manera la realización un procesamiento complejo. Este factor permite analizar videos en tiempo real con una lactancia inferior a 25 milisegundos. Además, YOLO alcanza más del doble de la precisión promedio en comparación con otros sistemas de detección en tiempo real.
- **YOLO razona globalmente sobre la imagen cuando realiza predicciones**: A diferencia de los métodos de ventana deslizante y de propuesta de regiones, durante el entrenamiento y el tiempo de prueba YOLO añade implícitamente información contextual sobre las clases y sobre sus apariencias. Este hecho hace que YOLO realice menos de la mitad de los fallos que realizan otros modelos, como Faster R-CNN.
- **YOLO aprende las representaciones generales de los objetos**: Cuando se entrena con imágenes naturales y se prueba con imágenes reales, YOLO supera con un gran margen a los mejores métodos de detección de objetos, como son R-CNN o DPM.

A continuación, se va a realizar una pequeña descripción del proceso que realiza YOLO para analizar una imagen.

En primer lugar, YOLO divide la imagen de entrada en una matriz de celdas de S x S. En el caso de que el centro de un objeto caiga en una de las celdas de dicha matriz, dicha celda se convertirá en la responsable de la detección de dicho objeto.

Cada una de las celdas de la matriz calcula B cajas delimitadoras y la confianza de cada una de estas. El valor de la confianza refleja como de confiado está el modelo de que en dicha caja se encuentra un objeto y como de precisa es la que caja que se ha predicho. Formalmente la confianza se encuentra definida por la función:

$$\Pr(Objeto) \times IOU_{Pred}^{Truth}$$

En el caso de que no exista un objeto, esta fórmula dará como resultado 0. En caso contrario para la confianza se tendrá en cuenta la intersección sobre unión (IOU) entre la caja que ha predicho el modelo de YOLO y la caja que verdaderamente contiene al objeto.

Cabe destacar, que cada celda también predice C probabilidades condicionales de cada una de las clases. Esta probabilidad se representa de la siguiente manera:

$$\Pr(Clase_i|Objeto)$$

Estas probabilidades se encuentran condicionadas en la celda de la matriz que contiene dicho objeto. Únicamente se calcula un conjunto de probabilidades de las diferentes clases por cada celda de la matriz, independientemente del número de cajas.

Durante el tiempo de prueba, se realizará la multiplicación de las probabilidades condicionales de cada una de las clases y la confianza de la caja predicha. La fórmula por lo tanto será la siguiente:

$$\Pr(Clase_i|Objeto) \times \Pr(Objeto) \times IOU_{Pred}^{Truth}$$

El resultado de esta operación mostrara la probabilidad de que dicha clase aparezca en la caja y como precisa sea la caja predicha. La *Figura 10* muestra un ejemplo de este proceso.

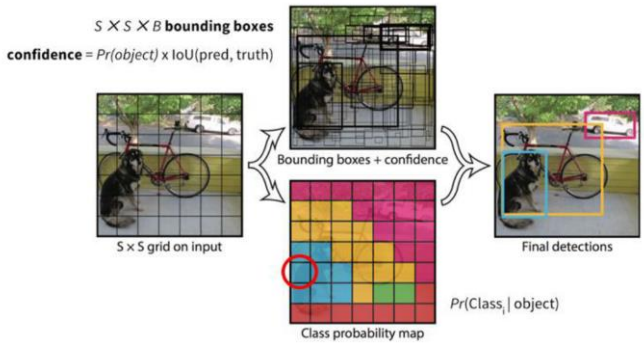

Figura 10. YOLO

Respecto al diseño de la red, este se encuentra inspirada en modelo de detección de imágenes de GoogLeNet [39]. En el caso de YOLO, la red está compuesta por 24 capas convolucionales seguidas por 2 capas completamente conectadas. Este diseño se puede observar en la *Figura 11.*

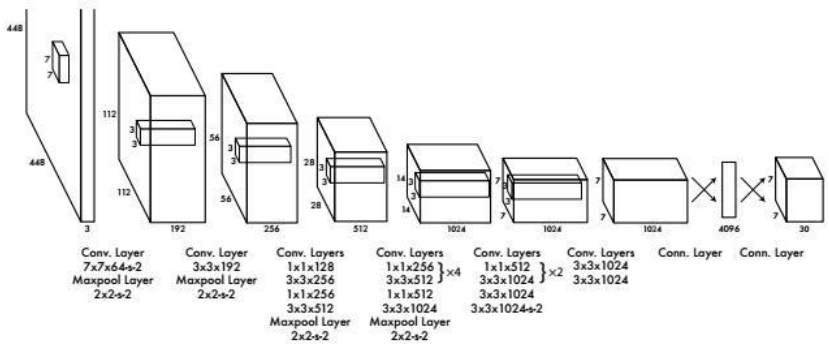

Figura 11. Arquitectura de la red empleada en YOLO.

Cabe destacar, que existe una versión de YOLO más rápida diseñada para romper los límites de la detección de objetos a una mayor velocidad. En este caso esta versión de YOLO más ligera utiliza un número menor de capas convolucionales (9 capas convolucionales en lugar de las 24 capas anteriormente comentadas) en comparación con la versión más lenta.

YOLO no es un modelo perfecto de detección de objetos, ya que este también presenta ciertas limitaciones a la hora de detectar objetos pequeños que aparecen juntos, como es el caso de los rebaños de animales. Yolo también sufre a la hora de detectar ciertos elementos que presenten nuevos e inusuales aspectos.

4.3.1 YOLOV4

Para este proyecto, se ha utilizado el algoritmo de detección de objetos YOLOv4, el cual es una evolución de su predecesor YOLOv3. [40] Este algoritmo mejora la precisión media y los FPS de YOLOv3 en un 10% y 13%, respectivamente. Esta mejora del rendimiento en comparación con otros algoritmos similares se puede observar en la *Figura 12.*

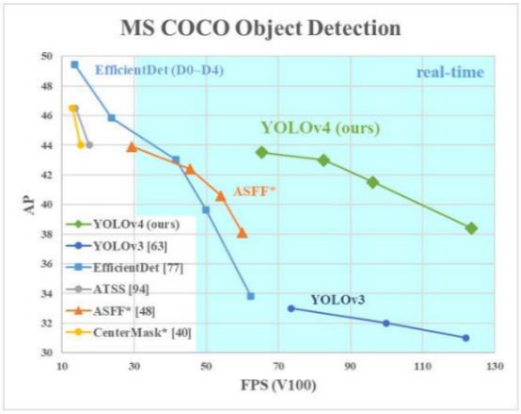

Figura 12. Comparación de YOLOv4 con otros algoritmos.

Esta última versión de YOLO, aparte de presentar un modelo detección de objetos eficiente y potente, permite que cualquier persona con una tarjeta gráfica con una capacidad de computación considerable (1080 ti o 2080 ti) pueda entrenar un preciso y potente modelo de detección.

YOLOv4 presenta la siguiente arquitectura:

- Backbone (Columna vertebral): CSPDarknet53
- Cuello (Neck): SPP y PAN
- Cabeza (Head): Utiliza la de su predecesor YOLOv3.

4.3 DEEPSORT

SORT (Simple Online and Realtime Tracking) es un simple y eficaz enfoque que realiza el seguimiento de múltiples objetos, para realizar este seguimiento utiliza el filtrado de Kalman y el algoritmo húngaro. DeepSORT permite el aumento del rendimiento del algoritmo SORT mediante la obtención de la apariencia de los objetos que se están siguiendo. Esta nueva extensión permite el rastreo de objetos a través de periodos más largos de ocultamiento, evitando de esta manera el cambio continuo de identificadores. Las pruebas experimentales de la extensión de DeepSORT muestran una reducción del 45% de los cambios de identificadores en comparación con el algoritmo original de SORT. A continuación, se va a comentar los diferentes componentes que forman el sistema [44].

- Estimación del estado. El escenario para el seguimiento se encuentra definido en un espacio de estados de ochos dimensiones (u,v,x,h,u`,v`,x`,h`) donde (u,v) contienen el punto central de la caja delimitadora, x es su relación de aspecto, h su altura, y sus respectivas velocidades en términos de coordenadas de imágenes. De esta manera, el componente original de DeepSORT, utiliza un filtro de Kelman para poder predecir la localización de las cajas delimitadoras de los objetos.
- La descripción de la apariencia constituye el nuevo componente de DeepSORT. La información de la apariencia de un objeto detectado es extraída mediante una red neuronal convolucional dando como resultado la diferenciación entre las características de una clase respecto a otra.
- Manejo del seguimiento. Por cada objeto sobre el que se está realizando un seguimiento se contabiliza el número de frames que han pasado desde la última asociación de medición exitosa. Este contador es incrementado durante la predicción del filtro de kelman, restableciéndose a 0 cuando se produce correctamente la asociación de una medida. En el caso de que dicho contador exceda un límite máximo, se considerara que el objeto que se estaba siguiendo ha salido de la imagen. El seguimiento de un objeto se mantiene en estado tentativo durante los primeros tres frames, tiempo en que se espera que se realice una asociación de medición exitosa. En el caso que no se realice una asociación exitosa el seguimiento tentativo será eliminado.
- A cada caja delimitadora sobre la que se esté realizando un seguimiento se le asignara un identificador único. Este identificador dependerá de las medidas de localización y apariencia que se estén llevando a cabo.

4.4 FÓRMULA DE SEMIVERSENO

La fórmula del semiverseno [33] es una importante ecuación que permite el cálculo del camino más corto entre dos puntos del globo sabiendo las coordenados de ambos puntos.

Dados dos puntos cualesquiera sobre una esfera.

$$\text{haversine}(\theta) = \text{haversine}(long_2 - long_1) + \cos(lat_1) \times \cos(lat_2) \times \text{haversine}(lat_2 - lat_1)$$

Las variables de long y lant, corresponde a las coordenadas en radianes de cada uno de los puntos.

Para resolver la distancia entre los dos puntos escogidos, se aplicará la inversa de la semiverseno, donde h es el resultado de la ecuación anterior.

$$d = r \times archav(h) = 2 \times r \times \arcsin(\sqrt{h})$$

R corresponderá al radio de la esfera, en el caso de nuestro proyecto este corresponderá al radio de la tierra en kilómetros, dato necesario para poder calcular la velocidad de un vehículo en kilometros/hora.

5. TÉCNICAS Y HERRAMIENTAS

En este apartado se van a describir las diferentes técnicas y herramientas utilizadas a lo largo del desarrollo. En este apartado podemos encontrar tres tipos de subapartados:

- Técnicas y herramientas empleadas en el Servidor
- Técnicas y herramientas empleadas en la aplicación web
- Herramientas CASE

5.1 TÉCNICAS Y HERRAMIENTAS: SERVIDOR

En este apartado se van a especificar las diferentes herramientas utilizadas en la parte del servidor.

5.1.1 PYTHON

Python [11] es un lenguaje de programación de alto nivel, orientado a objetos y con semántica dinámica. Cabe destacar que este lenguaje tiene una sintaxis simple y fácil que enfatiza en la legibilidad y, por lo tanto, reduce el costo del mantenimiento. La división en diferentes módulos provee a este lenguaje una mayor modularidad del programa y, por tanto, una mayor reusabilidad.

En el caso del proyecto a desarrollar este lenguaje ha sido elegido principalmente por dos razones. La primera es debido a que Python puede ser utilizado como servidor para una aplicación web. La segunda es debido a que este lenguaje es simple y fácil de entender.

5.1.2 PIP

Pip [12] es un gestor de paquetes del lenguaje de programación Python. Este gestor permite la instalación de paquetes y librerías que no se encuentran de manera estándar en las librerías de Python.

Este gestor de paquetes se encuentra incluido en el instalador de Python desde la versión 3.4.

Durante el desarrollo del proyecto este gestor de paquetes ha sido utilizado para la instalación de diferentes paquetes y librerías en Python.

5.1.3 PYCHARM

PyCharm [14] es un IDE utilizado principalmente para la programación en el lenguaje Python. Este IDE ha sido desarrollado por JetBrains. Entre las características de este IDE cabe destacar el análisis del código, la depuración gráfica, integración son sistemas de control de versiones y el desarrollo web con django.

A lo largo del desarrollo del proyecto, Pycharm ha sido el IDE utilizado para las partes que necesitaban ser programadas en Python.

5.1.4 FLASK

Flask [13] es un Framework creado en Python, el cual permite la creación rápida y sencilla de una aplicación web. Este Framework depende del motor de plantillas de Jinja y del conjunto de herramientas Werkszeug WSGI.

Flask permite definir las diferentes rutas que se utilizan para realizar las distintas acciones mediante peticiones HTTP.

A continuación, en la *Figura 13* se puede observar un código de ejemplo, el cual crea una aplicación web que imprime Hola Mundo.

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def holamundo():
    return "¡Hola Mundo!"

app.run(port=5000)
```

Figura 13. Ejemplo Hola Mundo con Flask.

5.1.5 FLASK SOCKETIO

Flask-SocketIO [15] proporciona a la aplicación de Flask un mecanismo de comunicación bidireccional de escasa latencia entre servidor y cliente. Se pueden utilizar cualquier tipo de librerías destinadas para los clientes de SocketIO, independientemente del lenguaje, para establecer una conexión permanente con el servidor.

5.1.6 FIREBASE

Firebase [17] es una plataforma para el desarrollo de aplicaciones móviles y aplicaciones web adquirida por Google en 2014.

Es una plataforma ubicada en la nube, integrada con Google Cloud Plataform, la cual usa un conjunto de herramientas para la creación y sincronización de proyectos para dotarlos de una mayor calidad. Podemos encontrar una plataforma similar a la ofrecida por Google en Amazon, con Amazon Web Services Amplify, al igual que Firebase está también permite el desarrollo de aplicaciones web y móvil.

Dentro de los servicios ofrecidos por Firebase, únicamente se van a definir los utilizados durante el desarrollo.

- **Real time Database**: Es una base de datos NoSQL alojada en la nube. Los datos de esta base de datos son compartidos por todos los usuarios que tengan una instancia de esta, en caso de pérdida de conexión los datos aún se encontraran disponibles. Además, cabe destacar que los datos de la base de datos se almacenan en formato JSON.
- **Storage**: Servicio de almacenamiento de datos en la nube que permite el almacenamiento y descarga de archivos pesados, como pueden ser fotos o videos.

Únicamente se han comentado dos de las funciones de esta plataforma, pero esta contiene un número mayor de funcionalidades las cuales no se van a comentar debido a que no han sido utilizadas. El uso de Firebase provee a los desarrolladores de software una serie de ventajas:

- Integración fácil con diferentes aplicaciones web y móviles, con soporte para casi cualquier Framwork actual. Para la integración en una aplicación únicamente será necesario importar las claves e identificadores que la plataforma te provee cuando creas un nuevo proyecto. Una vez añadida dicha información se tendrá total acceso a los diferentes servicios ofertados por la plataforma.
- No es necesario la creación ni mantenimiento de servidores propios. Por lo tanto, no será necesaria la instalación y utilización de software dedicado para esta función.
- Firebase dota a los usuarios de su plataforma con una gran y detallada documentación de cada uno de los servicios, permitiendo de esta manera el desarrollo de aplicaciones de una mayor calidad y envergadura.

5.1.7 POSTMAN

Postman [16] es una herramienta que ayuda al desarrollo de APIs y simplifica los pasos necesarios para crear las mismas. Con esta herramienta se pueden realizar solicitudes HTTP a cualquier API, permitiendo de esta manera su comprobación y testeo.

Esta herramienta ha sido utilizada para realizar las pruebas sobre la API desarrollada.

5.1.8 OPEN CV

OpenCV (Open Source Computer Vision Library) [41] es una biblioteca de software de visión artificial y aprendizaje automático de código abierto. OpenCV se creó para proporcionar una infraestructura común para aplicaciones de visión por computadora.

La biblioteca tiene más de 2500 algoritmos optimizados, que incluyen un conjunto completo de algoritmos de aprendizaje automático y visión por computadora clásicos y de última generación. Estos algoritmos se pueden utilizar para detectar y reconocer rostros, identificar objetos, clasificar acciones humanas en videos, etc.

5.1.9 GOOGLE COLAB

Google Colaboratory, también llamado "Colab", [51] te permite ejecutar y programar en Python en tu navegador con las siguientes ventajas:

- No requiere ningún tipo de configuración.
- Esta plataforma da acceso gratuito a GPUs.
- Permite compartir contenido gratuitamente.

Colab, en general, puede facilitar el trabajo, ya seas estudiante, profesor o investigador de IA.

Esta plataforma se ha utilizado para realizar el entrenamiento de un modelo de visión artificial personalizado.

5.2 TÉCNICAS Y HERRAMIENTAS: APLICACIÓN WEB

En este apartado se van a especificar las diferentes herramientas utilizadas en la parte de la aplicación web.

5.2.1 VUE

Vue [20] es un Framework de JavaScript de código abierto para la construcción de interfaces de usuario, este Framework está basado en el modelo arquitectónico MVVM o Modelo-Vista-Vista Modelo.

Los ficheros utilizados por Vue se encuentran formados por una combinación de los lenguajes más comunes del desarrollo web (HTML, CSS, JavaScript), lo que permite que este framework sea más accesible a los desarrolladores. Toda la información referente a los distintos paquetes y plugins de Vuejs se encuentra bien documentada. Estas son algunas de las razones por las que este Framework ha comenzado a ser más utilizado.

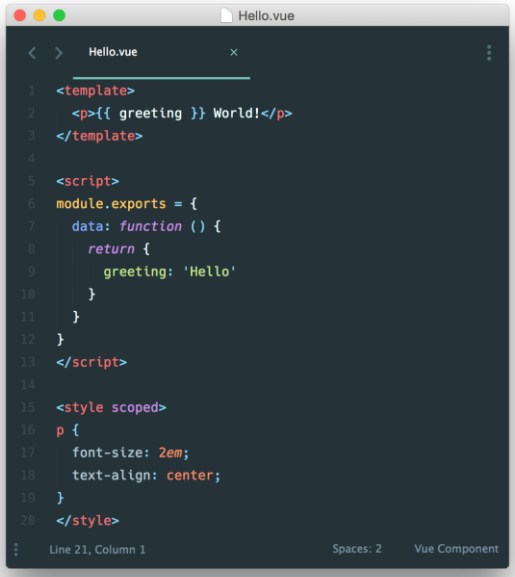

Figura 14. Hola mundo Vuejs

En la *Figura 14* se pueden diferenciar claramente 3 partes. El lenguaje HTML, encargado de la estructura de la página web, se encontrará entre las etiquetas <template>, mientras que el código CSS, encargado del diseño gráfico de la página web, se encontrará entre las etiquetas <style>. Para la definición de las variables y el comportamiento presente en la página web se utilizará JavaScript, este se encontrará entre las etiquetas <script>.

5.2.2 VUEX

Vuex [18] es una librería de Vuejs que sigue un patrón de gestión de estados. Este patrón proporciona a la aplicación de Vuejs un almacén con variables, las cuales pueden ser accedidas por todos los componentes. Estos componentes hacen referencia al State que se puede observar en la *Figura 15.*

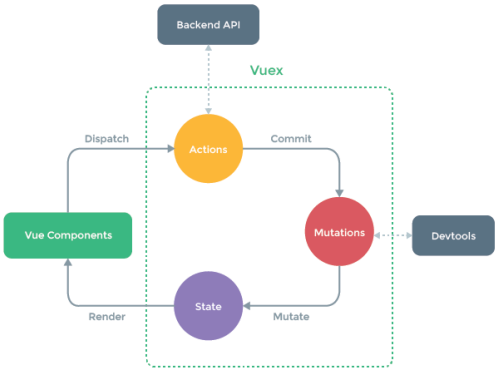

Figura 15. Vuex

Para modificar las variables del almacén será necesario hacer uso de funciones asíncronas o actions. La única forma de guardar los cambios realizados sobre el State en el almacén de Vuex será realizando un commit sobre una mutación o mutation. Este flujo de acciones se puede observar en la *Figura 15.*

5.2.3 VUETIFY

Vuetify [21] es un Framwork utilizado para crear de una manera más fácil y sencilla interfaces graficas de usuario. Este posee una gran cantidad de recursos que permiten que sea fácil de aprender. Todos estos recursos se encuentran debidamente documentados.

5.2.4 AXIOS

Axios es un cliente HTTP [22] basado en promesas, a través del cual se puede consumir los servicios de una API. Las solicitudes pueden ser de diferentes tipos como POST, GET, DELETE Y PATCH.

5.2.5 PLOTLYJS

Plotly [23] es una librería de gráficos de JavaScript con más de 40 tipos diferentes de gráficas disponibles.

5.2.6 VUE GOOGLE MAPS

Plugin que permite la incorporación del servicio de Google Maps [24] a la aplicación de Vuejs. Para el correcto funcionamiento de esta API será necesario poseer una key o clave de Google.

5.2.7 NPM

NPM [25] (Node Package Manager) es el sistema gestor de paquetes utilizado por Node.js. NPM nos permite instalar de forma fácil y sencilla cualquiera de las librerías disponibles.

Durante el desarrollo de la aplicación web, este gestor de paquetes ha sido utilizado para la instalación de diferentes paquetes y librerías en Vuejs.

5.2.8 VISUAL STUDIO

Visual Studio code [26] es un IDE desarrollado por Microsoft para la gran parte de sistemas operativos. Este IDE tiene soporte para una gran cantidad de lenguajes de programación, entre los que se encuentra Javascript. Este entorno de desarrollo presente ventajas como:

- **Corrección de errores**: Visual Studio proporciona sugerencias de rellenado, así como soluciones a errores comunes de código.
- **Destaca palabras clave**: Visual Studio resalta las palabras clave en diferentes colores, para que de esta manera sean más fácilmente identificables.
- **Temas y colores personalizados**: Visual Studio permite la selección de fuentes y colores de múltiples fuentes.
- **Compara cambios en el código**: Visual Studio implementa un control de versiones para evitar pérdidas de progreso.
- **Programación en cuadernos**: Permite la utilización de Jupyter Notebooks dentro del propio Visual Studio.

Durante el desarrollo del proyecto este IDE ha sido utilizado para la creación de la aplicación web de Vuejs.

5.2.9 CSS

CSS (siglas en inglés de Cascading Style Sheets) [46] es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Este lenguaje es muy usado para establecer el diseño visual de los documentos web y de las interfaces de usuario escritas en HTML, además puede ser aplicado a cualquier documento XML.

Junto con HTML y JavaScript, CSS es una tecnología usada por muchos sitios web para crear páginas visualmente atractivas, interfaces de usuario para aplicaciones web y GUIs para muchas aplicaciones móviles.

CSS es mantenido a través del grupo de trabajo de CSS, grupo perteneciente a W3C.

5.2.10 JAVASCRIPT

JavaScript [47] es un robusto lenguaje de programación que se puede aplicar a un documento HTML y usarse para crear interactividad dinámica en los sitios web. Fue inventado por Brendan Eich, cofundador del proyecto Mozilla, Mozilla Foundation y la Corporación Mozilla.

También es necesario comentar que JavaScript es un lenguaje de programación interpretado, el cual se define como orientado a objetos, basado en prototipos y

débilmente tipado. Cabe destacar que este lenguaje se encuentra basado en ECMAScript.

Toda la funcionalidad de la aplicación web se encuentra implementada en este lenguaje de programación.

5.2.11 HTML

HTML (Hyper Text Markup Language) [48] es un lenguaje de marcado estándar para la creación de páginas web. Los elementos que utiliza este lenguaje permiten especificar las posiciones y estructura que va a seguir la página, el texto, las imágenes, los vídeos, etc.…

HTML se encuentra se encuentra gestionado por el World Wide Web Consortium (W3C) o Consorcio WWW, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación.

En el caso de querer añadir un elemento externo, como poder ser una imagen o video, no será necesario su incrustación directa en el código, sino que simplemente será necesario la adición de su referencia. De este modo, será el navegador web el encargado de interpretar dicho código, permitiendo de esta manera su visualización.

5.3 HERRAMIENTAS CASE

En este apartado se van a especificar las diferentes herramientas CASE [27] utilizadas durante el desarrollo del proyecto. Estas herramientas permiten el aumento de la productividad en el desarrollo software, reduciendo de esta manera los recursos empleados en el mismo.

5.3.1 MICROSOFT PROJECT

Microsoft Project [28] es un software de administración de proyectos desarrollado y comercializado por Microsoft. Esta herramienta permite la planificación temporal de un proyecto, la asignación de los diferentes recursos a las tareas, administración del presupuesto necesario para la finalización del proyecto y el análisis de las cargas de trabajo de los recursos que intervienen en el desarrollo.

Esta herramienta ha sido utilizada durante el desarrollo del *Anexo I: Planificación del proyecto*. Un ejemplo de su utilización se muestra en la *Figura 16.*

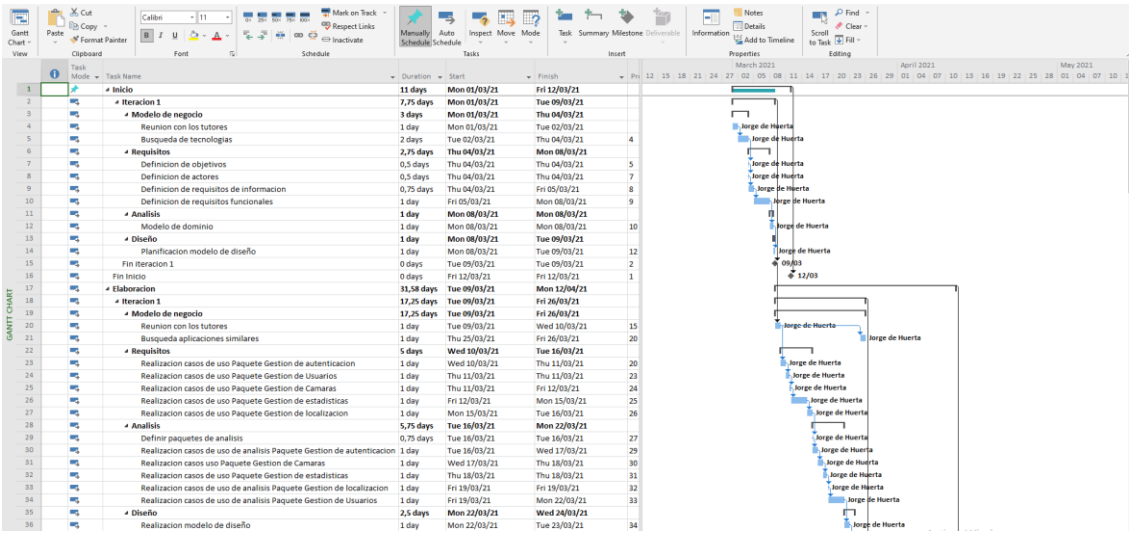

Figura 16. Ejemplo utilización Microsoft Project.

5.3.2 DRAW.IO

Draw.io [29] es una aplicación web para la creación de diagramas y gráficos. Esta aplicación nos permite la selección de diferentes plantillas predeterminadas o la creación de una plantilla propia. Las diferentes plantillas ofertadas por Draw.io se pueden observar en la *Figura 17.*

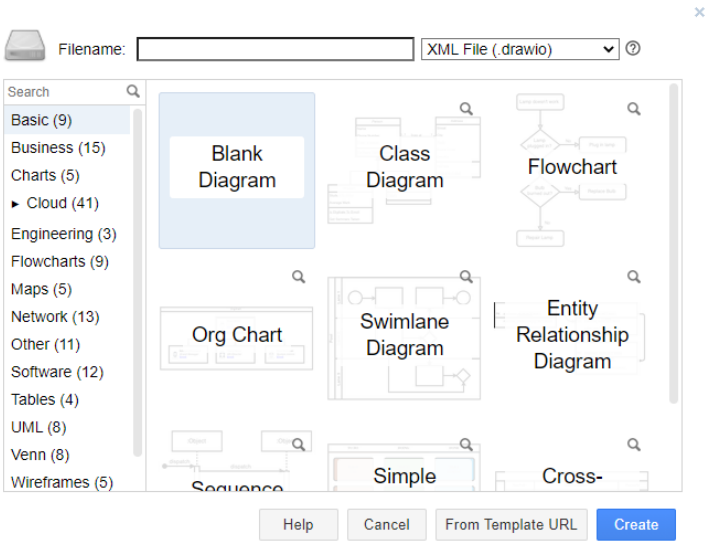

Figura 17. Diagramas disponibles en Draw.io

Los diagramas presentes en los diferentes anexos han sido realizados con esta herramienta.

5.3.3 EZESTIMATE

Herramienta destinada al cálculo de la estimación de esfuerzo de desarrollo de software utilizando el método de puntos de casos de uso (UCP). Esta herramienta ha sido utilizada en el *Anexo I: Planificación del proyecto.* En la *Figura 18* se puede observar la interfaz del programa EZEstimate.

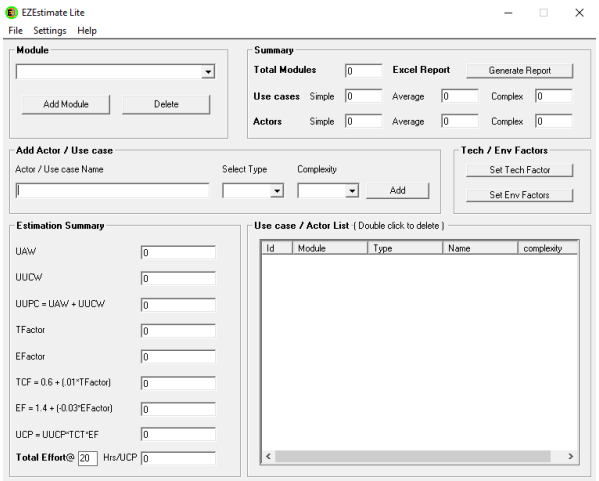

Figura 18. Interfaz EZEstimate

5.3.4 GIT

Git [30] es un sistema de control de versiones distribuido gratuito y de código abierto diseñado para manejar proyectos independientemente de su tamaño. Entre las ventajas de Git se pueden destacar:

- Fácil de aprender y utilizar
- Requiere poco espacio
- Gran rendimiento
- Permite múltiples flujos de trabajo

5.3.5 VUE STYLEGUIDIST

Herramienta de Vue utilizada para la documentación tecnica de los diferentes archivos del proyecto de Vuejs.

En este caso, esta herramienta se ha utilizada para la documentación de los diferentes archivos de Vuejs.

5.3.6 JSDOC

JSDoc permite agregar documentación al código fuente de JavaScript. Esta herramienta presenta una sintaxis similar a la de JavaDoc.

5.3.7 PYDOCTOR

Herramienta utilizada para la documentación del lenguaje de programación Python.

En este caso, esta herramienta se ha utilizada para la documentación de los códigos generados en el servidor.

6. ASPECTOS RELEVANTES DEL DESARROLLO

En este apartado se van a comentar cada de una de las etapas seguidas durante el desarrollo del proyecto.

6.1 MARCO DE TRABAJO

[35] Para el desarrollo del proyecto se han seguido las directrices del proceso unificado. El proceso unificado se caracteriza por estar conducido por casos de uso, por centrarse en la arquitectura y por seguir un proceso iterativo e incremental.

- **Conducido por casos de uso**: Desde el comienzo del desarrollo, se van a definir una seria de casos de usos, los cuales van a tener una gran importancia desde las primeas etapas de planificación del proyecto hasta el propio despliegue del sistema. De esta manera, se puede encontrar un gran enlace de las diferentes disciplinas a través de estos.
- **Centrado en la arquitectura**: El proceso unificado presenta múltiples vistas para poder cubrir todos los aspectos del desarrollo. Podremos encontrar diferentes modelos para tratar cada una de las vistas del sistema. Estos modelos son los mecanismos a través de los cuales se va a poder especificar la arquitectura.
- **Proceso iterativo e incremental**: Esta es una de las principales características del proceso unificado. El desarrollo se encuentra compuesto por cuatro fases principales denominadas inicio, elaboración, construcción y transición. Al finalizar cada una de las fases se realiza una entrega, permitiendo de esta manera una desarrollo y avance de manera evolutiva e incremental. Estas entregas se pueden realizar en el momento en que se finaliza una etapa, una fase o una iteración.

El proceso unificado propone un marco flexible y adaptable para el desarrollo de diferentes proyectos software.

Como se ha comentado anteriormente se van a diferenciar claramente cuatro diferentes fases (*Figura 19*), a continuación, se van a describir brevemente cada una de ellas:

- **Inicio**: En la fase inicial se especifican los recursos a utilizar y se desarrollan los diferentes casos de uso del sistema.
- **Elaboración**: Se refinan los diferentes casos de uso que intervienen en el sistema y se determina la arquitectura que se va a utilizar.
- **Construcción**: Se implementan y prueban los diferentes componentes que forman el sistema.
- **Transición**: En esta última fase ya se tiene una versión beta del producto desarrollado, únicamente será necesario su refinamiento y corrección de posibles bugs o errores.

Por último, también se podrán diferenciar múltiples disciplinas de desarrollo y gestión del proyecto (*Figura 19*), en este caso únicamente se tendrán en cuenta las siguientes disciplinas:

- Modelo de dominio
- Requisitos
- Análisis y diseño
- Implementación
- Prueba

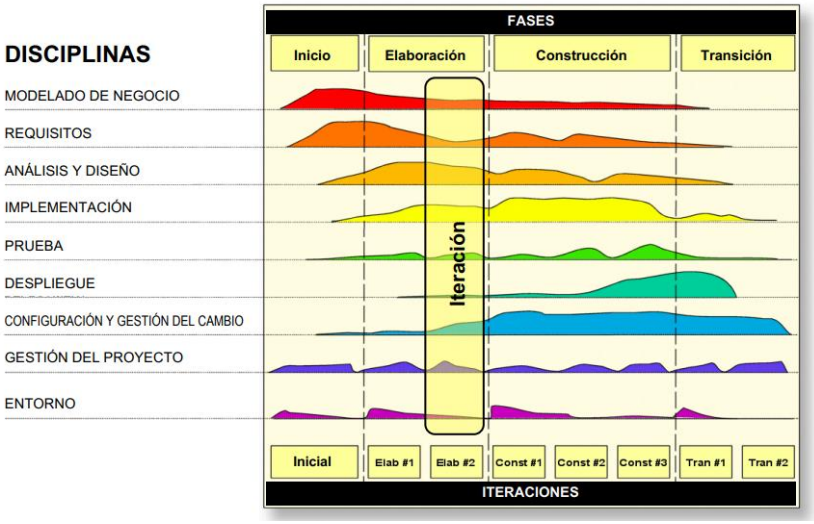

Figura 19. Fases del proceso unificado.

6.2 ESTIMACIÓN DE ESFUERZOS

En el *Anexo I: Planificación temporal* se ha realizado una estimación de esfuerzos mediante el método de puntos de casos de uso (UCP), este método tiene en cuenta para estimar el esfuerzo de desarrollo los actores que intervienen en el sistema, los diferentes casos de uso, los factores técnicos y los factores de entorno.

En el caso de los actores y los casos de uso, se les van a asignar unos valores en función de la complejidad de estos, mientras que los factores técnicos y de entorno se le va a asignar un peso (W) en función del impacto de dicho factor, así como una complejidad percibida por los desarrolladores.

Para la estimación de esfuerzos se ha utilizado la herramienta EZEstimate. En la *Figura 20,* se puede observar el proyecto de EZEstimate utilizado para la estimación realizada con los casos de uso del Anexo II.

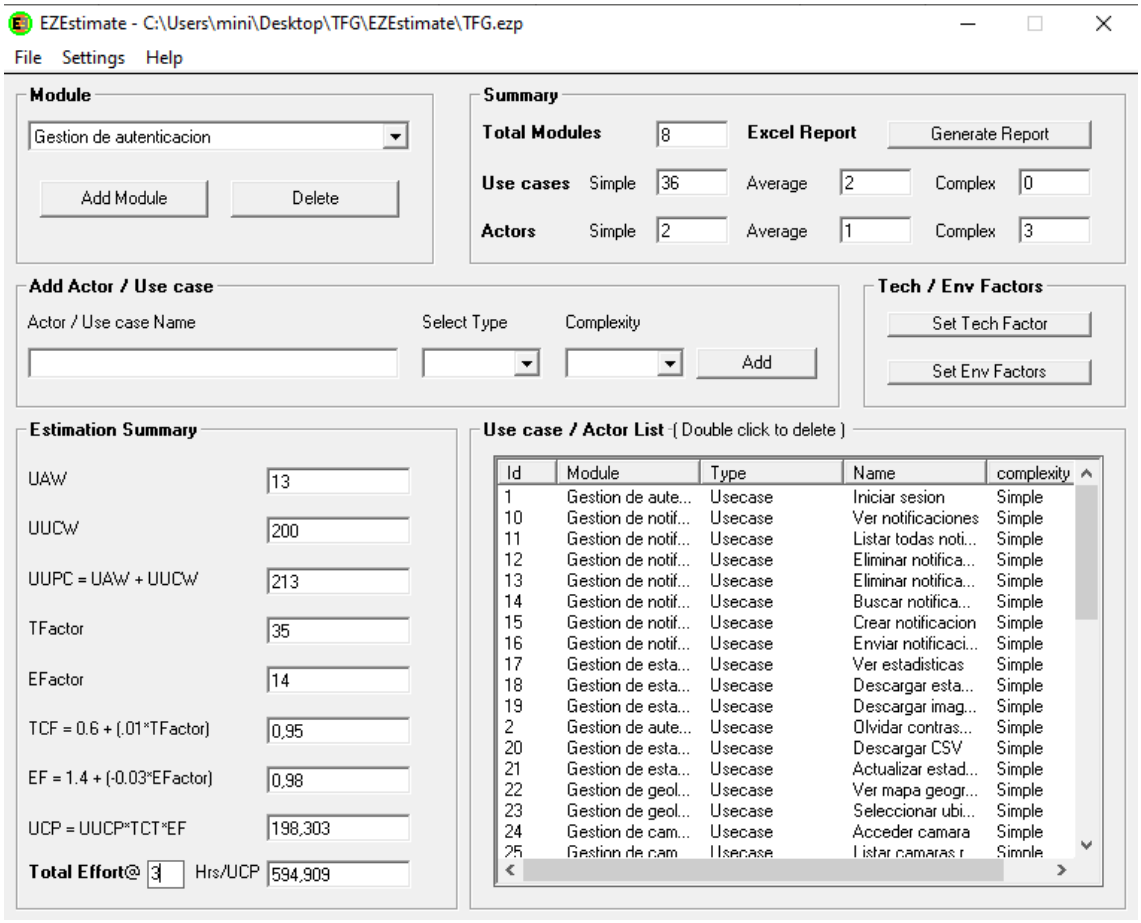

Figura 20. EZEstimate

Cabe destacar que el valor predeterminado del esfuerzo por caso de uso (20 horas/caso de uso) ha sido reducido a 3 horas/caso de uso debido a que la mayoría de los casos de uso tiene una complejidad simple y una funcionalidad parcialmente compartida.

6.3 PLANIFICACIÓN TEMPORAL

[35] Después de realizar la estimación de esfuerzos, se ha realizado una planificación temporal, para poder establecer las tareas a realizar, así como la duración de estas. Esta planificación nos permite conocer de manera estimada la duración del desarrollo del proyecto a realizar.

Esta planificación temporal se ha realizado teniendo en cuenta el proceso unificado, por lo que se podrán diferenciar claramente las diferentes cuatro fases y las disciplinas correspondientes. Además, para esta planificación se va a utilizar un diagrama de Gantt. Este diagrama nos permitirá diferenciar de manera clara la duración entre las diferentes tareas a realizar, y las relaciones entre las mismas.

Para el desarrollo de la planificación temporal ha sido necesario la utilización de la herramienta Microsoft Project, este se puede observar en la *Figura 21.*

	Inicio	**11 days**	**Mon 01/03/21**	**Fri 12/03/21**		
	Iteracion 1	**7,75 days**	**Mon 01/03/21**	**Tue 09/03/21**		
	Fin iteracion 1	0 days	Tue 09/03/21	Tue 09/03/21	2	Jorge de Huerta
	Fin Inicio	0 days	Fri 12/03/21	Fri 12/03/21	1	Jorge de Huerta
	Elaboracion	**31,58 days**	**Tue 09/03/21**	**Mon 12/04/21**		
	Iteracion 1	**17,25 days**	**Tue 09/03/21**	**Fri 26/03/21**		
	Fin Hito iteracion 1	0 days	Fri 26/03/21	Fri 26/03/21	18	Jorge de Huerta
	Iteracion 2	**14,33 days**	**Fri 26/03/21**	**Mon 12/04/21**		
	Fin Hito iteracion 2	0 days	Mon 12/04/21	Mon 12/04/21	42	Jorge de Huerta
	Fin Elaboracion	0 days	Mon 12/04/21	Mon 12/04/21	17	Jorge de Huerta
	Construccion	**53 days**	**Mon 12/04/21**	**Mon 07/06/21**		
	Iteracion 1	**20 days**	**Mon 12/04/21**	**Mon 03/05/21**		
	Fin Hito iteracion 1	0 days	Mon 03/05/21	Mon 03/05/21	65	Jorge de Huerta
	Iteracion 2	**20 days**	**Mon 03/05/21**	**Mon 24/05/21**		
	Fin Hito iteracion 2	0 days	Mon 24/05/21	Mon 24/05/21	84	Jorge de Huerta
	Iteracion 3	**13 days**	**Mon 24/05/21**	**Mon 07/06/21**		
	Fin Hito iteracion 3	0 days	Mon 07/06/21	Mon 07/06/21	104	Jorge de Huerta
	Fin Construccion	0 days	Mon 07/06/21	Mon 07/06/21	64	Jorge de Huerta
	Transicion	**12 days**	**Mon 07/06/21**	**Fri 18/06/21**		
	Iteracion 1	**7 days**	**Mon 07/06/21**	**Mon 14/06/21**		
	Fin Hito interacion 1	0 days	Mon 14/06/21	Mon 14/06/21	122	Jorge de Huerta
	Iteracion 2	**5 days**	**Mon 14/06/21**	**Fri 18/06/21**		
	Fin Hito iteracion 2	0 days	Fri 18/06/21	Fri 18/06/21	136	Jorge de Huerta
	Fin Transicion	0 days	Fri 18/06/21	Fri 18/06/21	121	Jorge de Huerta

Figura 21. Planificación temporal realizada en Microsoft Project.

Dentro de la planificación temporal cabe destacar cuatro hitos o entregas principales:

- **Fin Inicio**: Finalización de la primera etapa del proceso unificado, los casos de uso deben encontrarse definidos.
- **Fin elaboración**: Realización de todos los casos de uso de la disciplina de requisitos y de análisis.
- **Fin construcción**: Implementación y comprobación de todos los componentes relevantes del sistema a desarrollar.
- **Fin transición**: Creación de un programa funcional, con escasos o ningún error en el código. Finalización de la documentación del proyecto.

6.4 ESPECIFICACIÓN DE REQUISITOS

[36] En esta fase se va a realizar la especificación y documentación de los diferentes requisitos y objetivos que el sistema debe cumplir. Por lo tanto, se va a realizar la documentación de los objetivos, los actores que intervienen, los requisitos de información, los requisitos funcionales y los no funcionales. Toda esta documentación se puede consultar en el *Anexo II: Especificación de requisitos del software.*

Para la realización del *Anexo II: Especificación de requisitos del software*, se llevará a cabo la especificación de requisitos teniendo en cuenta el método de Durán y Bernárdez.

6.4.1 PARTICIPANTES

El proyecto cuenta con 4 participantes, entre ellos se encuentran 3 tutores y un alumno del Grado en Ingeniería informática. Todos los participantes pertenecen a la misma organización, la Universidad de Salamanca.

6.4.2 OBJETIVOS DEL SISTEMA

A continuación, se describen los objetivos necesarios que debe cumplir el sistema para poder satisfacer los requisitos planteados.

6.4.3 REQUISITOS DE INFROMACIÓN

A continuación, se define la información mínima que el sistema debe permitir almacenar para su correcto funcionamiento.

6.4.4 REQUISITOS FUNCIONALES

En primer lugar, en la *Figura 22* se pueden observar los diferentes paquetes en los que ha sido divido el sistema desarrollado.

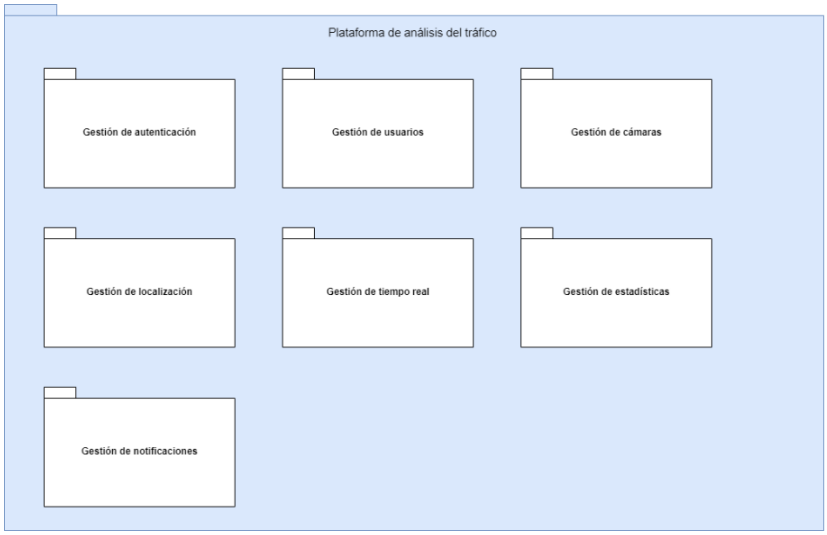

Figura 22. Diagrama de paquetes.

Posteriormente, se han descrito y especificado los diferentes actores que interactúan con el sistema de diferentes formas. Esto se podrán identificar en la *Figura 23.*

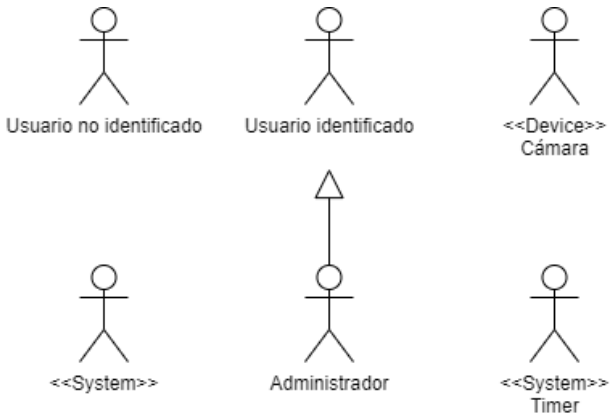

Figura 23. Actores del sistema.

También se podrán encontrar la descripción de los diferentes 38 casos de uso que se han establecido para el desarrollo de este proyecto.

Se encontrarán documentados de la siguiente manera:

UC-0001	**Iniciar sesión**	
Versión	1.0	
Autores	• Jorge de Huerta Merino	
Fuentes	• Juan Francisco De Paz Santana • Xuzeng Mao • Gabriel Villarrubia González	
Dependencias	[OBJ-0001] Gestión de usuarios [IRQ-0001] Información sobre usuarios del sistema	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *el Usuario no Identificado [ACT-0001] solicita el acceso al sistema.*	
Precondición	El actor *Usuario no Identificado [ACT-0001]* no debe encontrarse identificado en el sistema en ese momento.	
Secuencia normal	**Paso**	**Acción**
	1	El actor *Usuario no Identificado [ACT-0001]* solicita al sistema el inicio de su sesión.
	2	El sistema solicita al actor *Usuario no Identificado [ACT-0001]* los datos de acceso al sistema.
	3	El actor *Usuario no Identificado [ACT-0001]* rellena los datos necesarios.
	4	El sistema comprueba los datos rellenados. Si los datos proporcionados coinciden con algún usuario registrado en el sistema, se le dará acceso al sistema.
Postcondición	El actor *Usuario no Identificado [ACT-0001]* se ha identificado correctamente en el sistema.	
Excepciones	**Paso**	**Acción**
	4	Si los datos proporcionados, no coinciden con algún usuario registrado en el sistema, este notificara al actor *Usuario no Identificado [ACT-0001]* dicho error, a continuación este caso de uso termina.
Rendimiento	**Paso**	**Acción**
	2	3 segundos
	4	3 segundos
Frecuencia esperada	12 veces por día	
Importancia	Vital	
Urgencia	Hay presión	

Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 1. UC-0001 Iniciar sesión.

A continuación, se puede encontrar en la *Figura 24,* uno de los diagramas de casos de uso realizados.

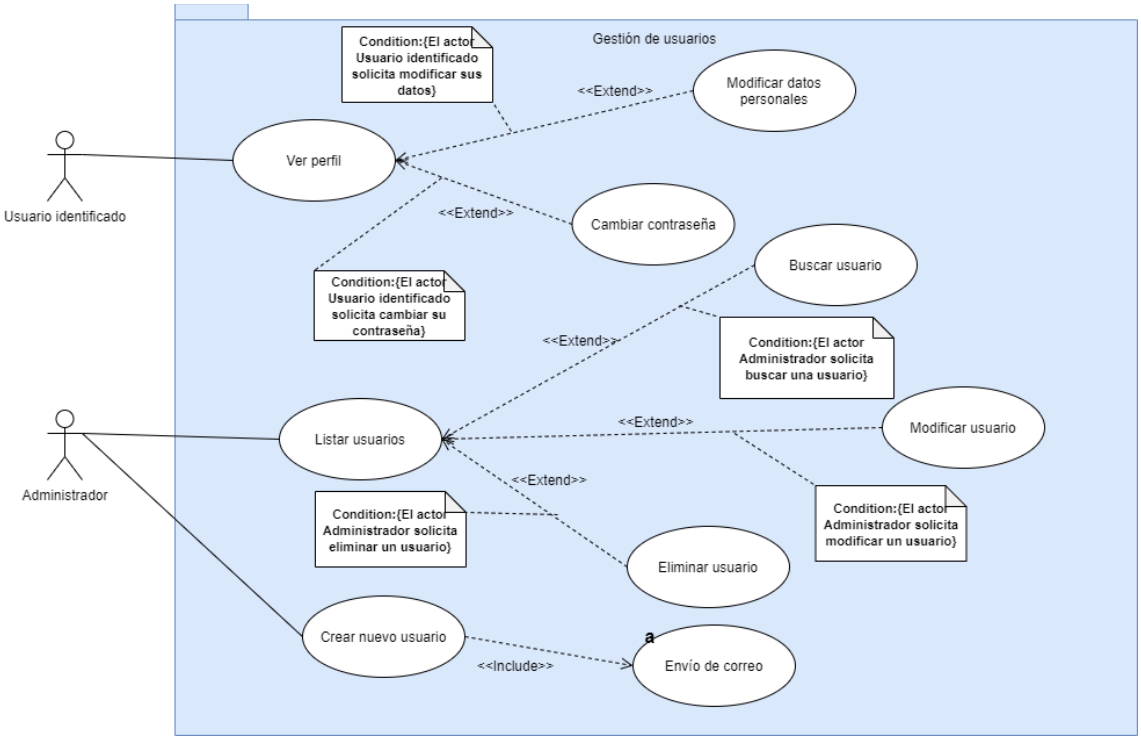

Figura 24. Paquete de Gestión de usuarios

6.4.5 REQUISITOS NO FUNCIONALES

Además de los requisitos funcionales, también se podrán encontrar los diferentes requisitos no funcionales, los cuales va a ser necesario que el sistema los cumpla.

6.5 ANÁLISIS DEL SISTEMA SOFTWARE

En la fase de análisis se necesario el análisis profundo y exhaustivo de los casos de uso documentados en el *Anexo II: Especificación de los requisitos del software*.

Toda la documentación de la fase de análisis se puede encontrar de forma más detallada en el *Anexo III: Análisis del sistema software.*

6.5.1 MODELO DE DOMINIO

El modelo de dominio es una representación de las clases conceptuales del mundo real, no de componentes software, con una gran importancia para el dominio del problema. Estas clases conceptuales irán acompañadas con los datos más relevantes de cada una de ellas, así como las posibles relaciones que puedan presentar entre ellas.

Este modelo de dominio se puede observar en la *Figura 25.*

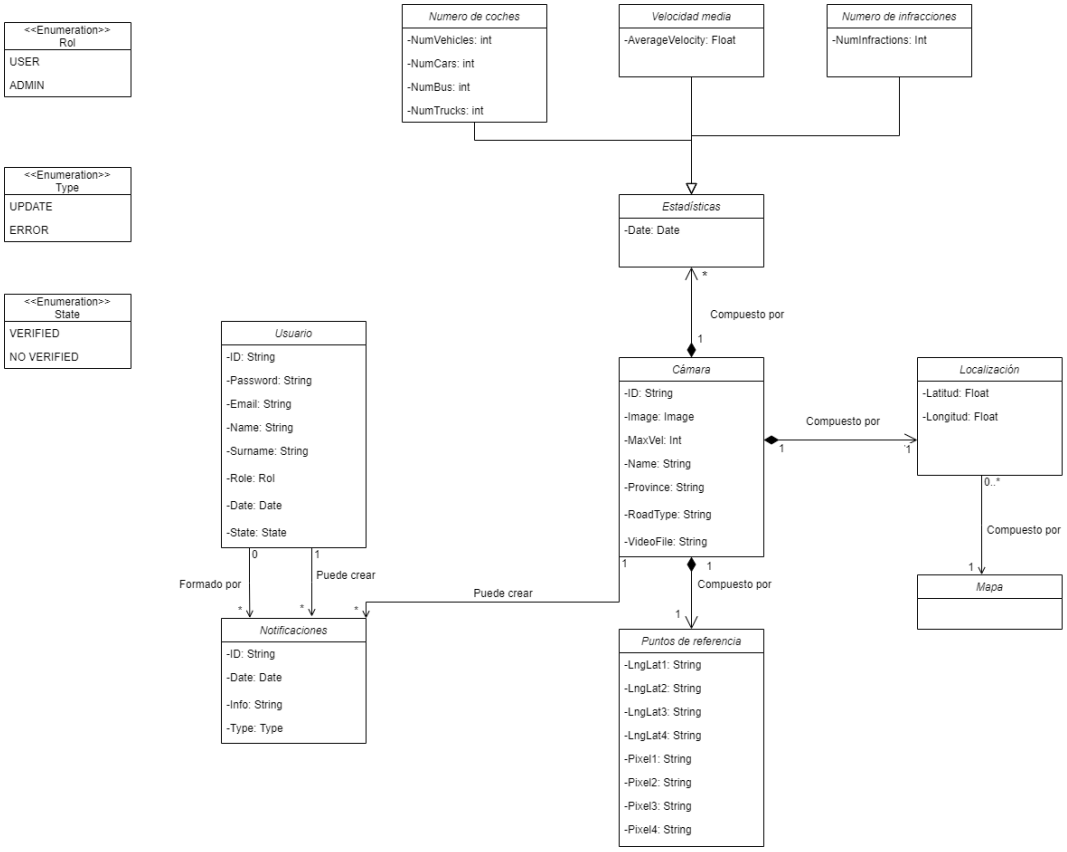

Figura 25. Modelo de dominio.

6.5.2 PAQUETE DE ANÁLISIS

En este apartado se va a realizar la descomposición del sistema, con el fin de organizar los artefactos del modelo del análisis en piezas más manejables. La disposición realizada se puede observar en la *Figura 26.*

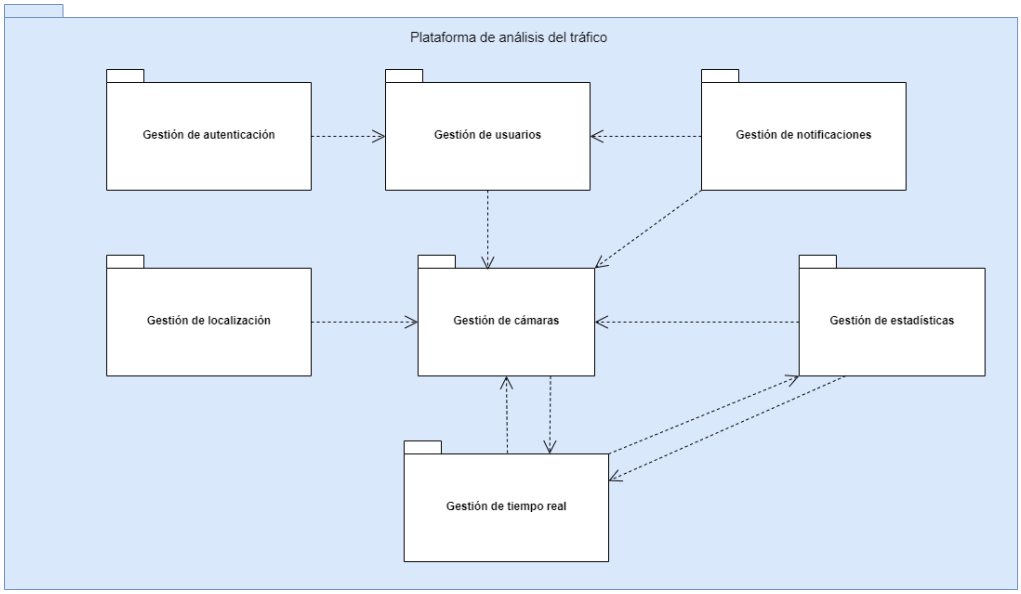

Figura 26. Paquete de análisis.

6.5.3 REALIZACIÓN DE LOS CASOS DE USO DEL ANÁLISIS

En este apartado se van a describir el funcionamiento de los casos de uso en termino de las clases de análisis y de sus objetos. La *Figura 27* muestra uno de los casos de uso realizados.

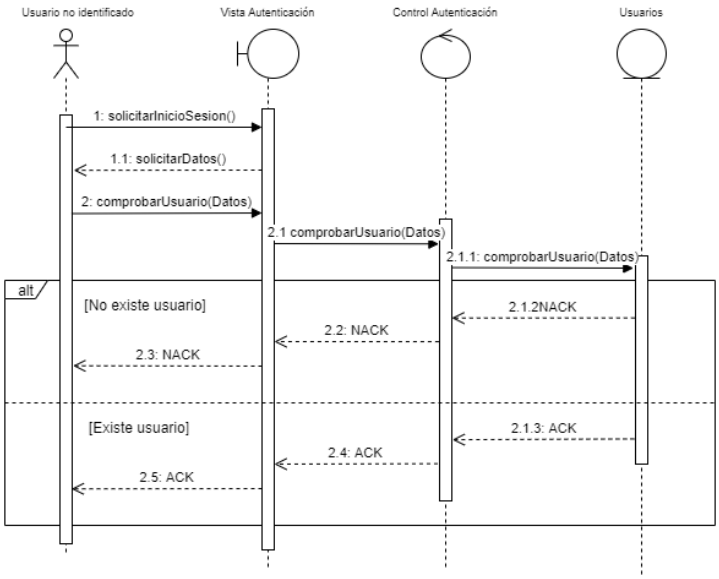

Figura 27. UC-0001 Inicio de sesión.

6.6 DISEÑO DEL SISTEMA SOFTWARE

En esta fase de diseño, a diferencia de la anterior fase de análisis, se va a abordar el dominio de la solución. Esta fase de diseño va a presentar una aproximación

a la futura implementación, por lo tanto, en esta fase se podrán encontrar los métodos y atributos que se observarán posteriormente en el desarrollo del sistema.

Toda la documentación relacionada con esta fase se puede encontrar en el *Anexo IV: Diseño del sistema software.*

6.6.1 MODELO DE DISEÑO

[37] En este apartado se van a especificar los diferentes patrones arquitectónicos utilizados. Para el modelo del dominio se considerarán las diferentes tecnologías utilizadas.

- Utilización del Framework **Vuejs**, con los lenguajes de programación JavaScript, HTML, CSS, para la presentación de la información al usuario.
- Utilización de **Python** para la gestión de los diferentes datos utilizados por el sistema, entre las posibles librerías a utilizar encontramos Flask.

6.6.1.1 PATRONES ARQUITECTÓNICOS

6.6.1.1.1 PATRÓN DE CAPAS

El patrón de capas tiene como objetivo la separación de las partes que componen el sistema software en capas con responsabilidades distintas, pero relacionadas entre sí, entre las capas podemos encontrar 3 diferentes tipos (*Figura 28*).

- **Capa de Interfaz de usuario**: Como su nombre indica, este para será la encargada de mostrar los datos al usuario, así como de interactuar con el mismo para la recogida de estos.
- **Capa de negocio**: En esta capa se encontrarán implementadas todas las operaciones y funcionalidades las cuales van a ser solicitadas desde la capa de interfaz de usuario.
- **Capa de datos**: En esta capa se llevará a cabo toda la gestión de los datos persistentes en el sistema.

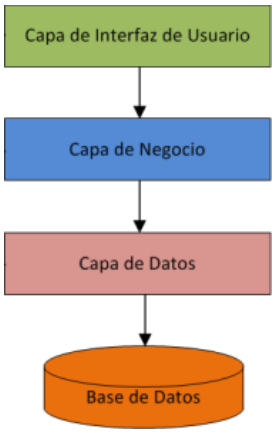

Figura 28. Patrón de capas

6.6.1.1.2 MVVM

El patrón MVVM, es un patrón arquitectónico con el que se pretende desacoplar lo máximo posible la interfaz del usuario de la lógica de la aplicación. La diferencia de este patrón con el patrón MVC es que este patrón sincroniza toda la información en el ViewModel en lugar de en el controlador como hace MVC.

Como se observa en la *Figura 29,* el ViewModel recibe comandos desde la capa de la View, para posteriormente realizar los cambios pertinentes en el modelo. La comunicación en sentido contrario se realizará mediante notificaciones.

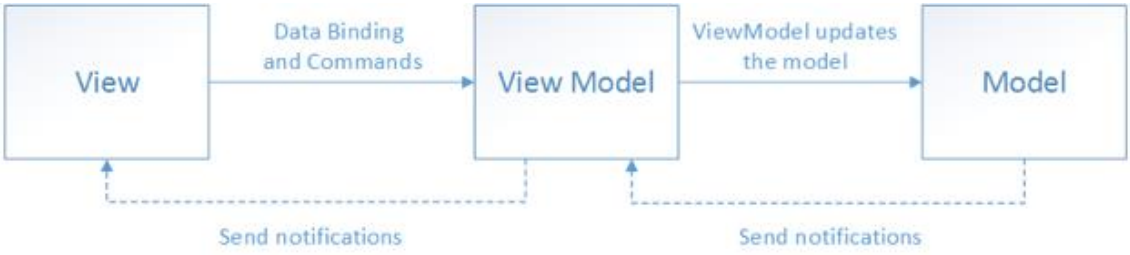

Figura 29. Patrón MVVM

La utilización de este patrón permitirá crear un código más limpio e estructurado, permitiendo de esta manera un mantenimiento más rápido. Estas son algunas de las razones por la que este patrón es muy utilizado en el desarrollo de aplicaciones web. Este patrón se encuentra presente en el Framework de Vuejs.

6.6.1.2 PATRONES DE DISEÑO

6.6.1.2.1 PATRÓN PUBLISH/SUSCRIBE

Publish/Suscribe es un patrón de mensajería en el cual el Publisher envía información de los Suscribers. Este tipo de arquitectura provee de un mecanismo de notificaciones instantáneo. En la *Figura 30,* se puede observar un ejemplo de la estructura que presentaría este patrón.

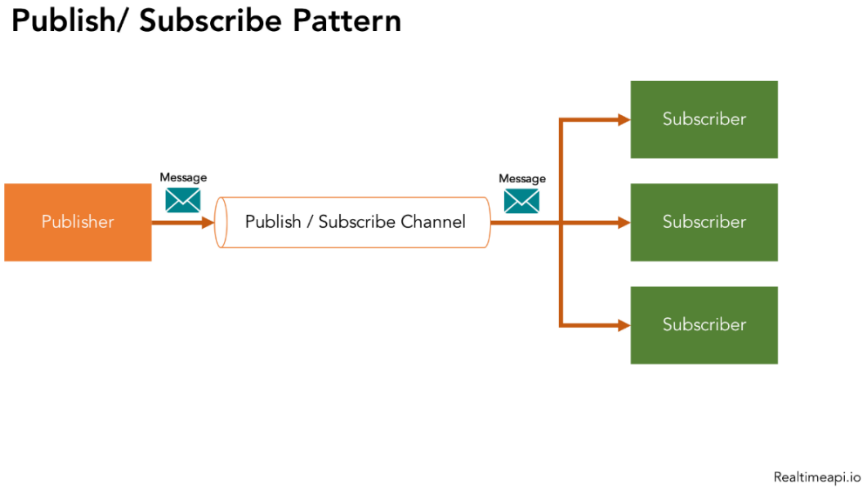

Figura 30. Patrón Publish-Suscribe

En este tipo de arquitectura intervienen tres tipos de actores:

- **Publisher**: Actor encargado de publicar los mensajes en el canal de comunicación establecido.
- **Canal de comunicación**: Canal de comunicación utilizado para la comunicación directa entre el Publisher y el Suscriber.
- **Suscriber**: Actor el cual va a recibir la información del Publisher.

Para la utilización de este tipo de arquitectura se utilizarán librerías como Flask_socketio, librería compatible con Flask. Esta librería permitirá que los usuarios que deseen ver la transmisión en vivo se suscriban a una room o habitación, a través de la cual se van a realizar el envío de las imágenes de la cámara después de ser procesadas.

6.6.1.2.2 PATRÓN DE GESTION DE ESTADOS

El patrón de gestión de estados o estate management (*Figura 31*) permite gestionar los diferentes recursos del sistema en stores o tiendas, permitiendo de esta manera el acceso a datos desde cualquier componente sin necesidad de generar datos duplicados.

Otra de las ventajas de definir esta separación entre las vistas y los estados de los datos, es la generación de un código más estructurado y de fácil mantenimiento.

En el caso de la biblioteca Vuex encontramos diferentes tipos de interfaces las cuales pueden interactuar con los diferentes componentes del sistema:

- **Actions**: Interfaz que permite la ejecución de las diferentes funciones asíncronas desde los diversos componentes del sistema.
- **Mutations**: Los métodos encontrados en los actions pueden realizar cambios en el estado de los diferentes objetos del store mediante commits.

- **State**: Contiene los estados de los diferentes objetos guardados en el store.

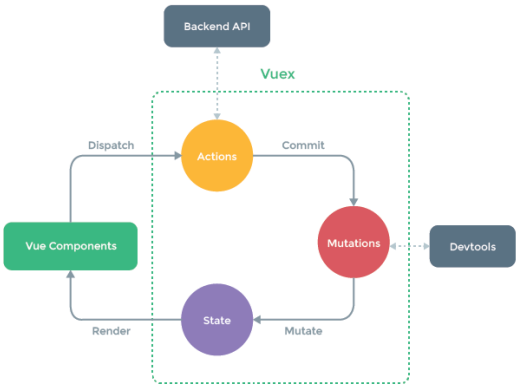

Figura 31. Patrón de gestión de estados.

6.6.1.3 SUBSISTEMAS DE DISEÑO

En este apartado se va a realizar la especificación del sistema en diferentes subsistemas. Entre los subsistemas podemos diferenciar el subsistema encargado de todas las funcionalidades del backend y el subsistema encargado de todas las funcionalidades del frontend. En la *Figura 32* se puede observar el subsistema de diseño implementado.

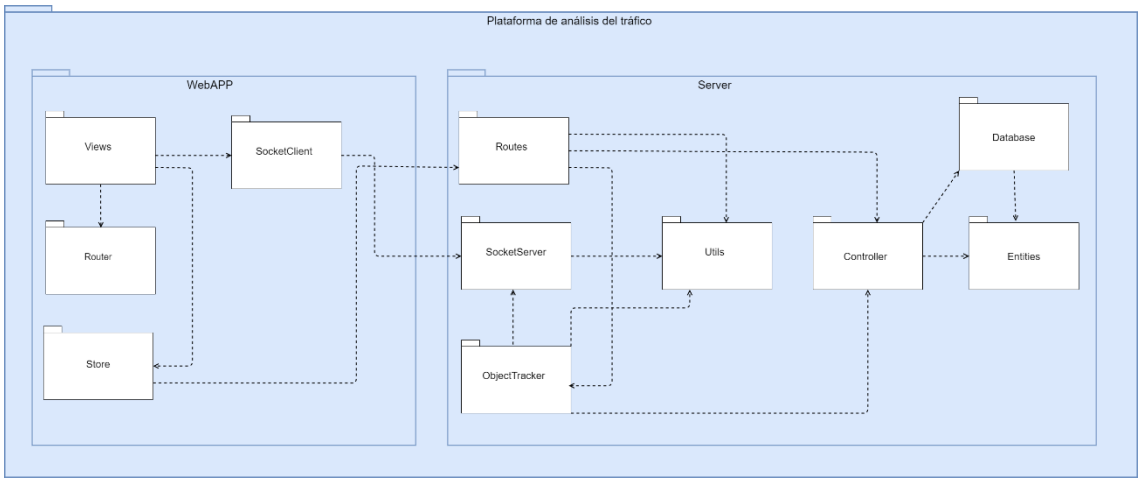

Figura 32. Subsistema de diseño.

6.6.1.4 CLASES DE DISEÑO

Este apartado tiene como objetivo especificar los contenidos de cada una de las clases de diseño especificadas en los anteriores subsistemas.

6.6.1.4.1 SUBSISTEMA WEBAPP

En este apartado se van a definir las diferentes clases de diseño del subsistema encargado de la aplicación web. Se pueden encontrar las siguientes clases:

- **Views**
- **Router**
- **Store**
- **SocketClient**

En la *Figura 33,* se puede observar el diagrama realizado para una de las clases de diseño anteriormente comentadas.

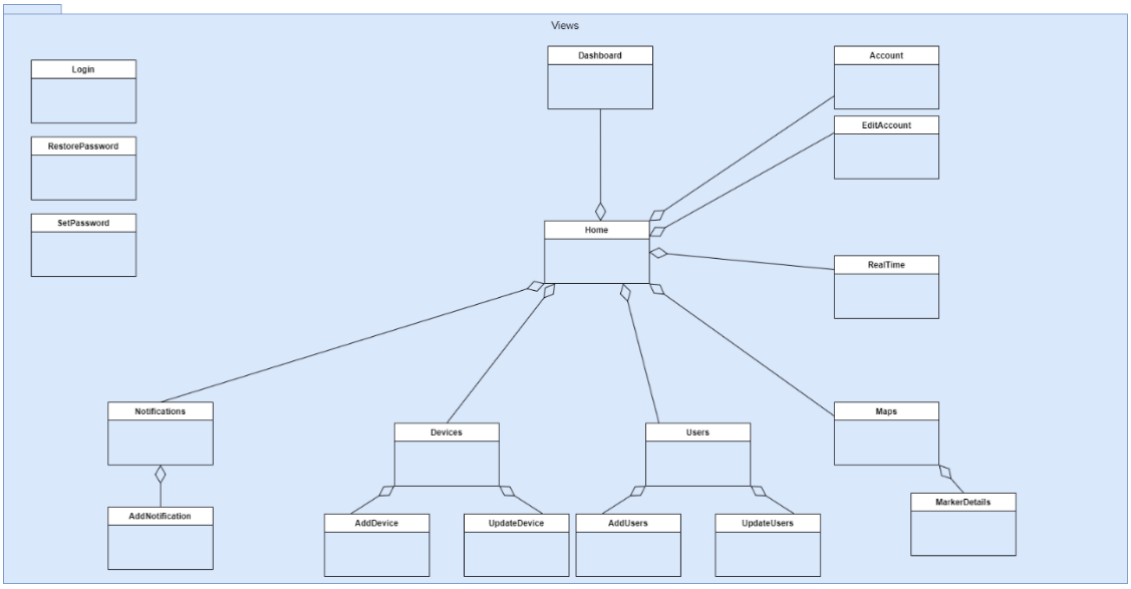

Figura 33. Views

6.6.1.4.2 SUBSISTEMA SERVER

En este apartado se van a describir las diferentes clases de diseño del subsistema servidor. Se pueden encontrar las siguientes clases:

- **Routes**
- **SocketServer**
- **Utils**
- **ObjectTracker**
- **Database**
- **Entities**
- **Controller**

En la *Figura 34,* se puede observar el diagrama realizado para una de las clases de diseño anteriormente comentadas.

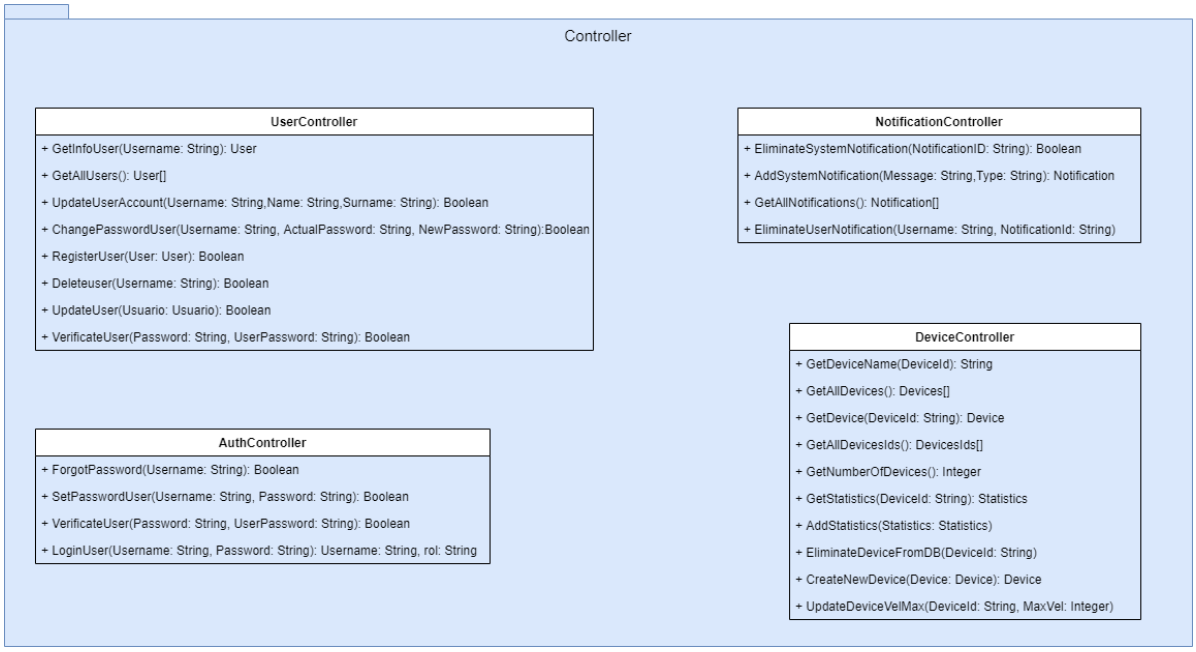

Figura 34. Controller

6.6.1.5 VISTA ARQUITECTÓNICA

En la *Figura 35,* se muestra la vista arquitectónica del modelo del diseño, dividida según las diferentes capas del patrón de capas anteriormente mencionado.

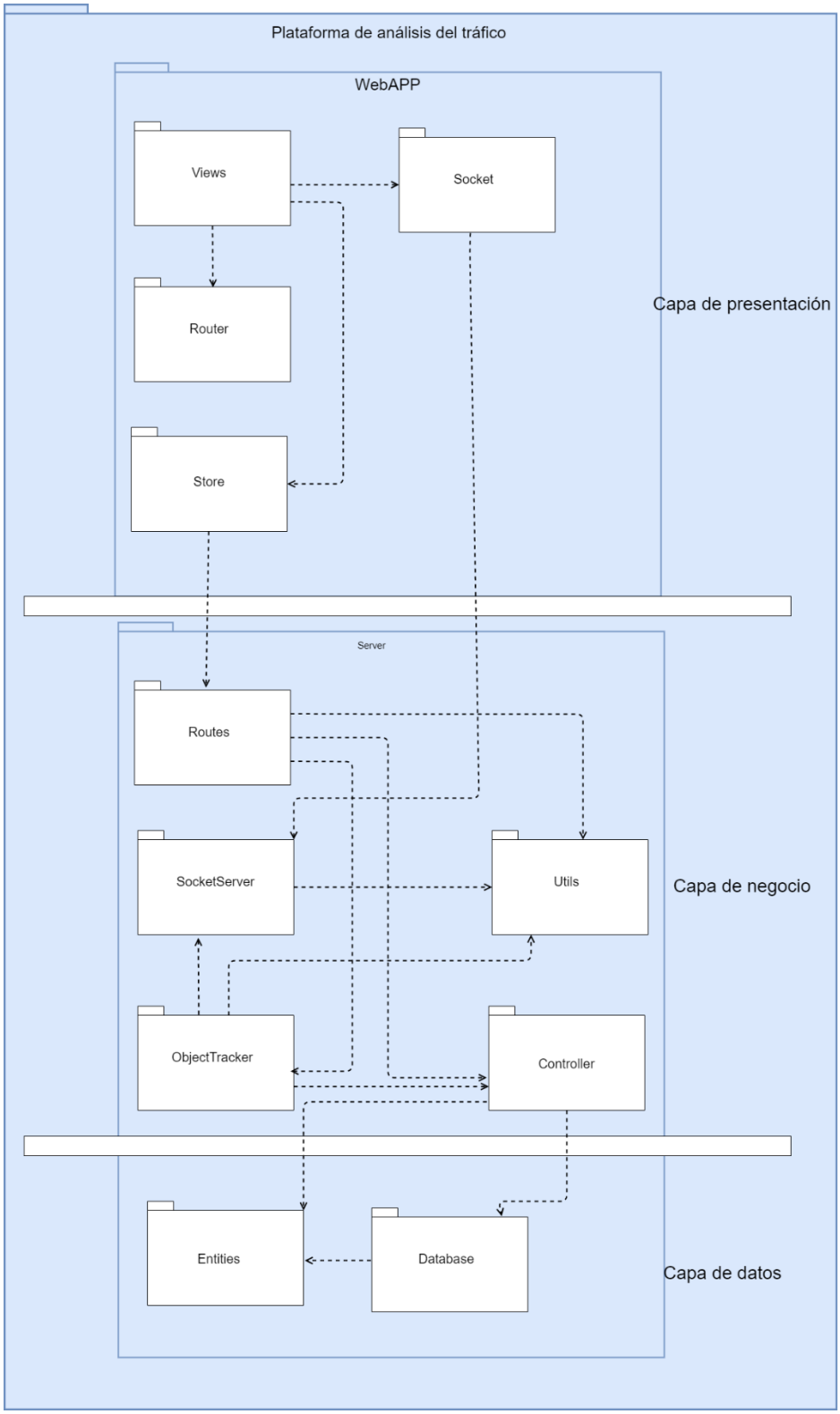

Figura 35. Vista arquitectónica

6.6.1.6 CASOS DE USO DEL MODELO DEL DISEÑO

En este apartado, se van a desarrollar los diferentes diagramas de secuencia de los casos de uso del sistema. En esta fase, los diagramas de secuencia presentarán un grado más alto de detalle.

En la *Figura 36,* se pueda observar el diagrama de secuencia realizado para uno de los casos de uso del sistema.

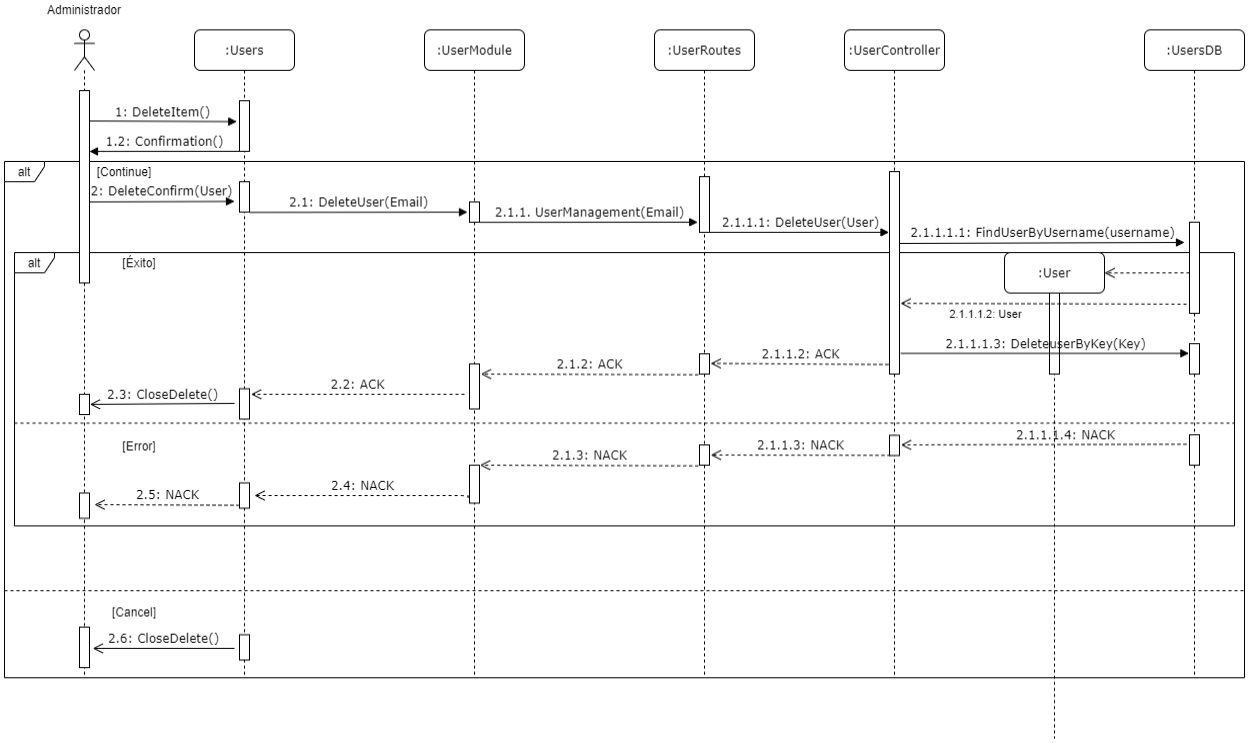

Figura 36. UC-0012 Eliminar usuario

6.6.2 DISEÑO DE LA BASE DE DATOS

Para el correcto funcionamiento del sistema, se necesitará que este almacene y gestione la información necesaria. Para este almacenamiento se utilizará RealTime Database, la cual es una de las funcionalidades que se pueden encontrar en Firebase. Esta base datos ofrecida por Google tiene un funcionamiento no relacional. El diseño de esta se puede observar en la *Figura 37.*

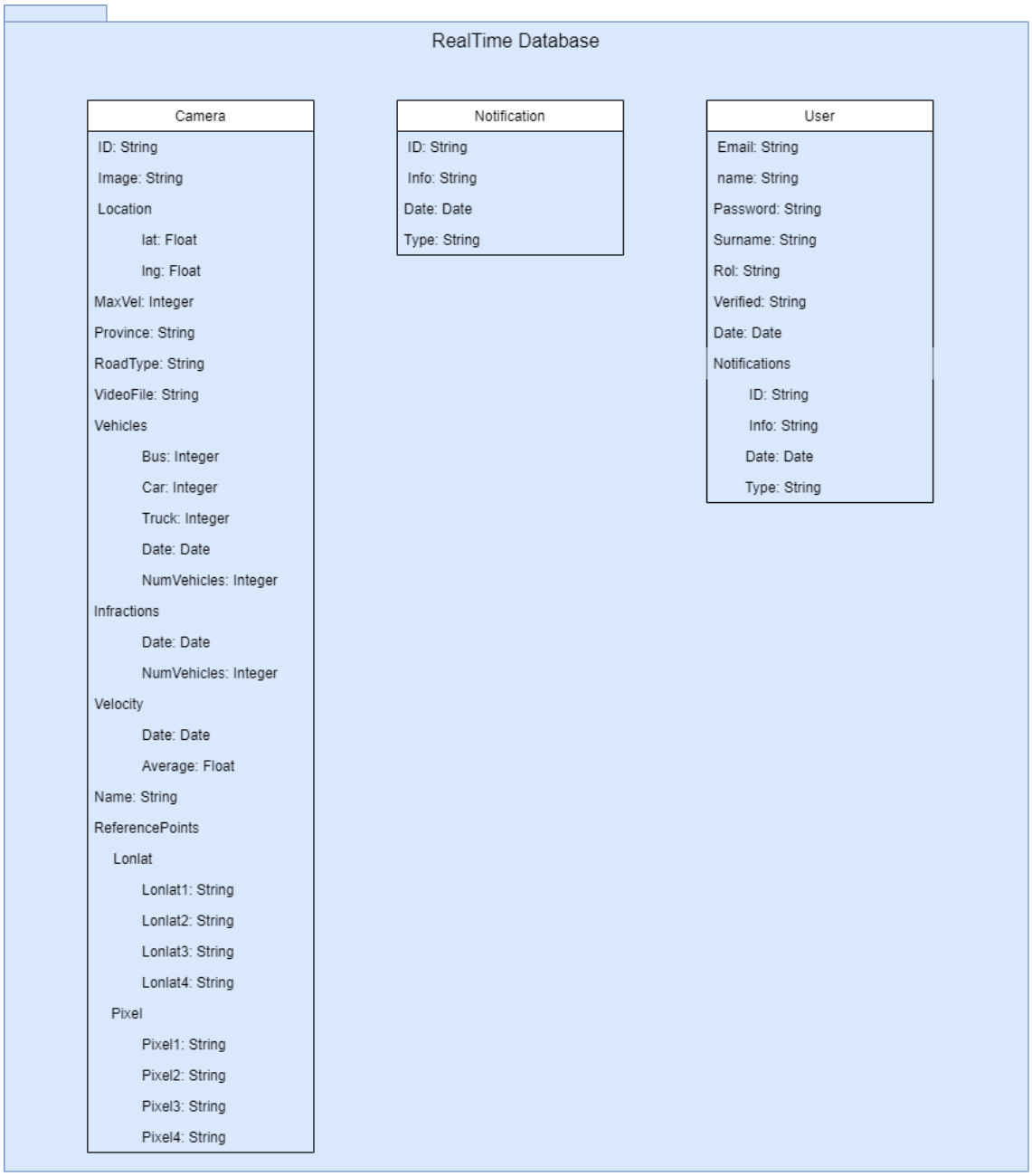

Figura 37. Diseño de la base de datos

6.6.3 MODELO DE DESPLIEGUE

El modelo de despliegue muestra de forma gráfica los diferentes nodos implementados durante el desarrollo del sistema, así como la forma de comunicarse entre ellos.

En el modelo de despliegue se muestran los siguientes nodos:

- **ServidorWeb:** Servidor en el que se encuentran los diferentes componentes necesarios para el correcto funcionamiento de la aplicación web y de todos sus servicios.
- **Cliente:** Este nodo representará la página web en el artefacto navegador al que pueden acceder los diferentes usuarios del sistema. Este nodo presentara tantas instancias como usuarios se encuentren conectados al sistema.
- **Servidor:** Este nodo representará al servidor encargado de recibir y gestionar las diferentes peticiones de los clientes conectados. Este se comunicará con Firebase para acceder a la base de datos del sistema.
- **Servidor Firebase:** Este nodo integra diversos servicios, como puede ser la base de datos utilizada para el almacenamiento de los diferentes elementos necesarios en el sistema o el almacén utilizado para el guardado de archivos más pesados, como imágenes.
- **Cámara:** Nodo encargado del envío de las imágenes**.**

El modelo de despliegue resultante se puede observar en la *Figura 38.*

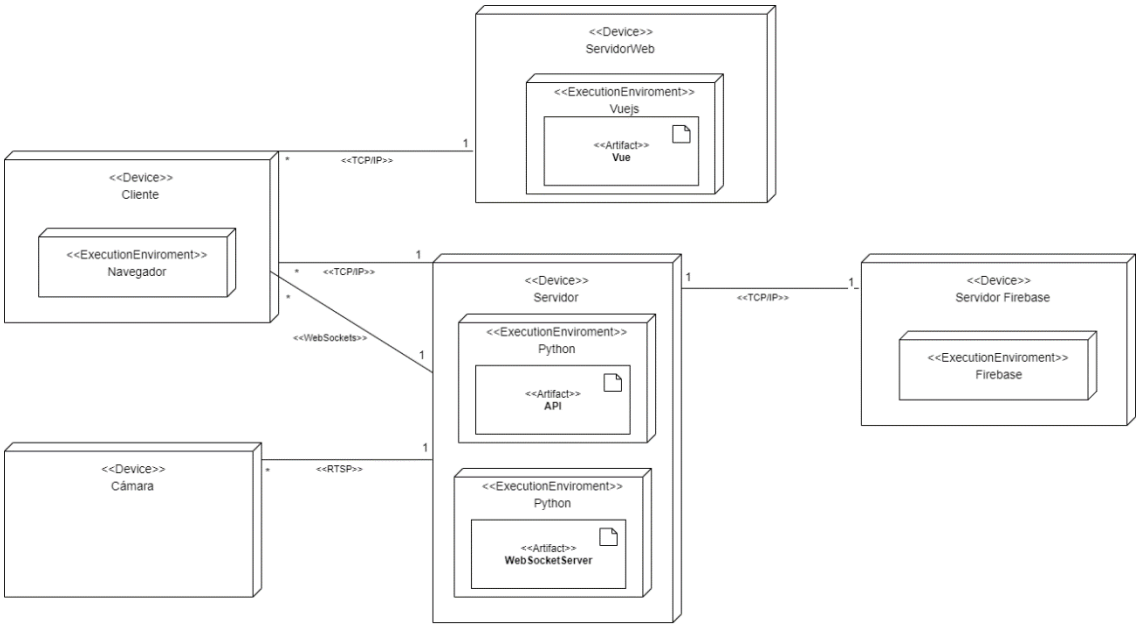

Figura 38. Modelo de despliegue.

6.7 IMPLEMENTACIÓN

En este apartado se va a detallar el proceso seguido en la fase de desarrollo del software. Esta fase debe cumplir con el dominio de la solución planteado en el *Anexo IV: Diseño del sistema software.*

El desarrollo se ha separado en dos subsistemas claramente diferenciados, uno es el subsistema del servidor y otro es el subsistema responsable de la aplicación web.

La información de los diferentes paquetes desarrollados se encuentra explicada y documentada en el correspondiente *Anexo V: Documentación técnica.*

6.7.1 SERVIDOR

En este apartado se va a realizar la enumeración de las diferentes tecnologías y bibliotecas empleadas durante el desarrollo del servidor.

Respecto a las tecnologías empleadas en la parte del servidor, se pueden diferenciar las siguientes:

- **Python**: Lenguaje de programación utilizado en todo el servidor. Se ha elegido este lenguaje de programación debido a la sencillez y facilidad de uso que presenta. Además, gracias a que Python es un lenguaje de programación de código abierto, la mayoría de los paquetes de este se encuentran documentados por su gran comunidad [11].
- **Firebase**: [17] Base de datos no relacional empleada para el almacenamiento de datos. Los motivos de su elección han sido los siguientes:
    - La integración de esta base de datos en Python es relativamente simple, ya que únicamente ha sido necesario importar las credenciales que Firebase provee en su aplicación web.
    - Al utilizar Firabase, no ha sido necesario la creación y mantenimiento de una base de datos propia, por lo tanto, no ha sido necesaria la instalación de ningún tipo de software específico para la gestión de bases de datos.
    - Firabase presenta la funcionalidad de Real Time Database, útil en nuestro caso debido a que permite la compartición de los datos en tiempo real, factor indispensable para poder cumplir los objetivos propuestos.
    - Firebase presenta un panel donde se puede visualizar y modificar fácilmente el contenido de la base de datos. Este panel se localiza en su aplicación web.

A continuación, se van a nombrar las principales bibliotecas utilizadas durante el desarrollo del servidor:

- **Flask**: Utilizada para el despliegue de la aplicación web.

- **Firebase_Admin**: Biblioteca necesaria para el acceso y utilización de la base de datos de Firebase. Esta biblioteca requerirá de una clave privada, la cual se obtiene directamente de Firebase.
- **Flask_SocketIO**: Biblioteca encargada de los eventos necesarios utilizados por los WebSockets.
- **OpenCV**: Biblioteca de código abierto utilizada para la visión artificial. En este caso será utilizada para la recogida de las imágenes de las cámaras.
- **Tensorflow**: Biblioteca destinada al aprendizaje automático.

6.7.1.1 RESULTADOS DEL PROCESO DE ENTRENAMIENTO

Debido a la falta de recursos y potencia computacional, se decidió realizar el proceso de entrenamiento del modelo de YOLOv4 a través de la herramienta gratuita de Google Colab, herramienta que se ha comentado anteriormente. Antes de realizar cualquier entrenamiento, fue necesario la configuración de la librería de cuDNN, esta es una biblioteca de primitivas acelerada por GPU utilizada para redes neuronales.

Respecto al conjunto de datos utilizado para el entrenamiento del modelo, se ha empleado el conjunto de imágenes de UA-DETRAC [42], este conjunto de imágenes se encuentra formado por 10 horas de vídeos de carreteras con diferentes características atmosféricas y en diferentes localizaciones. Este conjunto de datos se encuentra dividido en dos, un 60% de los datos para el entrenamiento y el otro 40% para las pruebas.

En el caso del conjunto de datos de UA-DETRAC, únicamente tiene identificadas tres tipos de clases. Entre ellas encontramos: Car, Truck y Bus. Estos tipos se encuentran definidos en un documento llamado **obj.names**. El contenido de este fichero se puede encontrar en la *Figura 39*.

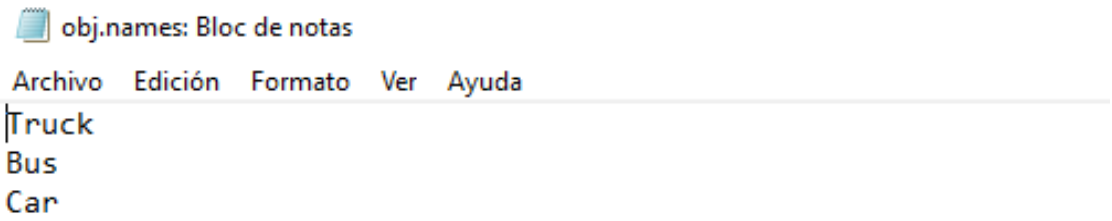

Figura 39. Clases de UA-DETRAC.

Cada una de las imágenes presentes en UA-DETRAC, tiene asociado un documento de texto o XML con el mismo nombre. Este fichero contendrá en cada línea un número de objeto, el cual representa la posición del objeto en el fichero **obj.names**, y las coordenadas de dicho objetos en la imagen, el formato es el siguiente: <clase del objeto> <x> <y> <anchura> <altura>. Un ejemplo de estas anotaciones se puede observar en la *Figura 40.*

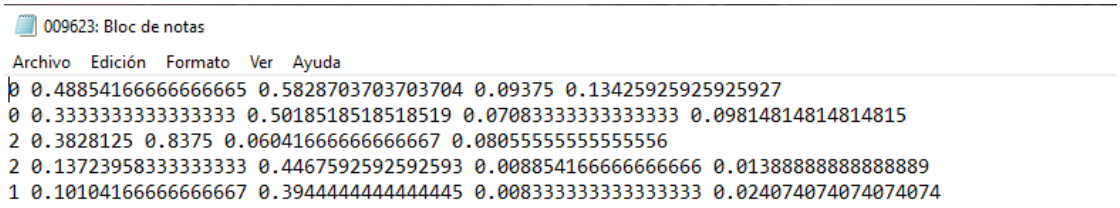

Figura 40. Anotación de una de las imágenes de UA-DETRAC.

Para realizar el entrenamiento, se necesitó crear previamente dos ficheros de texto, a los que llamamos traint.txt y test.txt. Estos ficheros funcionarán como una lista de las diferentes ubicaciones de las imágenes que van a ser utilizadas para el proceso de entrenamiento y pruebas.

Por último, será necesario la creación de un archivo **obj.data**, donde se indicará la localización de los archivos previamente creados. Este archivo (*Figura 41*) será completamente necesario para poder realizar el entrenamiento del modelo de YOLOv4.

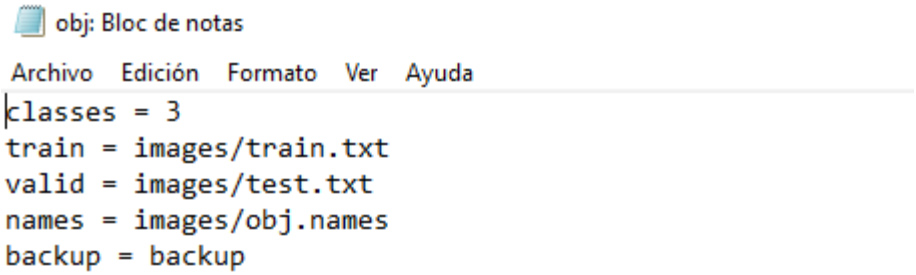

Figura 41. Fichero consultado durante el entrenamiento.

Una vez seleccionada la plataforma en la que se va a realizar el entrenamiento y el conjunto de datos con el que se va a realizar dicho proceso, únicamente ha sido necesario escoger el Framework sobre el que realizar el entrenamiento. En este caso se ha utilizado Darknet, el cual es un framework de redes neuronales de código abierto escrito en C y CUDA. Este se caracteriza por ser rápido, fácil de utilizar y con soporte de GPU.

Antes de realizar el entrenamiento de un modelo de YOLOv4, se necesitará la modificación del archivo de configuración **yolov4.cfg** [43]**.** Este archivo contendrá los diferentes parámetros de configuración que se tendrán en cuenta durante el proceso de aprendizaje. Cabe destacar que este archivo se encuentra dividido en diferentes capas, en nuestro caso únicamente va a ser necesaria la modificación de alguna de ellas.

A continuación, se van a comentar algunas de las capas más importantes:

```
[net]
batch=64
subdivisions=16
width=416
height=416
channels=3
momentum=0.949
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 6000
policy=steps
steps=4800,5400
scales=.1,.1
```

Figura 42. Capa de red.

En la *Figura 42,* se pueden observar los diferentes parámetros dentro de la capa de red, entre dichos parámetros cabe destacar los siguientes debido a su importancia en el entrenamiento:

- **Batch**: Este parámetro indica el número de muestras que van a ser evaluadas en un lote.
- **Subdivisions**: Este parámetro indicará el número de mini lotes que se van a realizar por cada lote. La GPU será la encargada de procesar estos mini lotes. Por lo que, en caso de tener una GPU no muy potente, este valor debería ser menor.
- **Width**: Tamaño de la red (ancho). Todas las imágenes serán redimensionadas al tamaño establecido en este parámetro. Al igual que en el caso anterior, en caso de tener una GPU no muy potente, este valor debería ser menor.
- **Height**: Tamaño de la red (alto). Todas las imágenes serán redimensionadas al tamaño establecido en este parámetro. En este parámetro ocurre lo mismo.
- **Max_batches**: Número de batches que va a ser necesario realizar. Este número será igual a 2000 * el número de clases que se desea entrenar. En este caso se ha establecido a 6000.

```
[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=mish
```

Figura 43. Parámetros correspondientes a la capa de red

En la *Figura 43,* se puede observar una de las múltiples capas encargadas de realizar el proceso de convolución, dentro de esta habría que destacar:

- **Filters**: Se indicará el número de filtros que se van a aplicar durante la convolución. Proceso que se encuentra definido en los conceptos teóricos.

```
[yolo]
mask = 6,7,8
anchors = 12, 16, 19, 36, 40, 28, 36, 75, 76, 55, 72, 146, 142, 110, 192, 243, 459, 401
classes=3
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1
```

Figura 44. Capa de YOLO.

En la *Figura 44,* se puede observar una de las tres capas encargadas de la detección, en nuestro caso será necesario modificar el siguiente parámetro:

- **Classes**: Para el entrenamiento de un número determinado de objetos, este parámetro debe ser cambiado acorde al número de las clases que se deseen detectar. Este parámetro se encontrará dentro de las 3 capas de YOLO.

Después de configurar los parámetros necesarios, se comienza con el proceso de entrenamiento, el cual una vez finalizado muestra la precisión media que se ha obtenido por cada una de las clases.

```
 detections_count = 27631, unique_truth_count = 8692
class_id = 0, name = Truck, ap = 86.64%          (TP = 2841, FP = 868)
class_id = 1, name = Bus, ap = 96.26%            (TP = 1443, FP = 72)
class_id = 2, name = Car, ap = 77.23%            (TP = 2763, FP = 712)
```

Figura 45. Resultado del entrenamiento.

Como se observar en la *Figura 45*, tendremos una precisión del 86%, 96% y 77% para la detección de camiones, buses y coches respectivamente.

Cabe destacar que debido a los fallos que el modelo entrenado generaba a la hora de detectar los coches, se ha decidido utilizar el modelo pre entrenado de YOLOv4. Este modelo se encuentra entrenado para las diferentes 80 clases de Microsoft COCO (Common Object in Context).

En la *Figura 46,* se muestra las imágenes resultantes después de ser procesada por el modelo YOLOv4. En este caso se puede observar el tipo de objeto detectado y la velocidad en Kilómetros/hora a la que se mueve dicho objeto.

Figura 46. Imágenes resultantes del proceso de análisis con YOLOv4.

6.7.1.2 CÁLCULO DE VELOCIDADES

[50] Después de haber realizado el proceso de detección y seguimiento de los objetos, se ha necesitado la implementación de un mecanismo a través del cual poder obtener la velocidad estimada de los diferentes objetos detectados. Para ellos, una vez obtenidas las ubicaciones de los objetos (punto medio de la caja delimitadora) será necesario transformar dichas coordenadas de la imagen a otro sistema de coordenadas, con las cuales sea posible estimar la distancia recorrida, estas coordenadas corresponderían a las empleadas en la superficie terrestre. Para esta transformación se ha utilizado un cambio de perspectivas, cambio mediante el cual se pueden intercambiar las coordenadas de un plano a otro. Esta transformación se puede observar en la *Figura 47.*

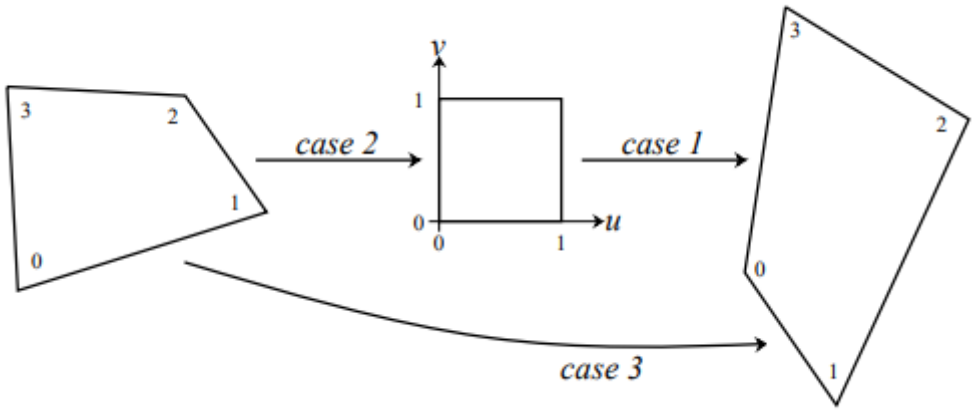

Figura 47. Transformación de un conjunto de coordenadas a otro.

Para la transformación de los diferentes puntos, se ha empleado una de las funciones incorporadas en la biblioteca OpenCV, librería anteriormente comentada.

Por último, cabe destacar que, para realizar las transformaciones anteriormente comentadas, ha sido necesaria la obtención previa de los cuatro puntos de referencia presentes en ambas perspectivas.

Una vez realizada dicha operación, será posible la obtención de la velocidad estimada de cada uno de los objetos presentes en la imagen. El cálculo de la velocidad se realizará mediante la aplicación de la formula del semiverseno. Una vez aplicada podremos obtener la velocidad en Kilómetros/hora de cada uno de los vehículos. Esta fórmula se puede observar en la *Figura 48.*

```
lon1, lat1, lon2, lat2 = map(radians, [lat_mid[0][0], lat_mid[0][1], lat_prevmid[0][0], lat_prevmid[0][1]])
dlon = lon2 - lon1
dlat = lat2 - lat1
a = sin(dlat / 2) ** 2 + cos(lat1) * cos(lat2) * sin(dlon / 2) ** 2
c = 2 * asin(sqrt(a))
r = 6371  # Radio de la tierra en Km
espacio =c * r
tiempo = 1/3600
velocity = espacio/tiempo
```

Figura 48. Formula del semiverseno en Python.

6.7.2 APLICACIÓN WEB

En este apartado se va a realizar la enumeración de los diferentes tecnologías y bibliotecas empleadas durante el desarrollo de la aplicación web de Vuejs.

Respecto a las tecnologías empleadas en la parte, se pueden diferenciar las siguientes:

- **Vuejs**: Framework utilizado para el desarrollo de la aplicación web. Se utilizado el framework de Vuejs debido a la flexibilidad y facilidad de uso que este presenta.
- **CSS**: Lenguaje de diseño gráfico.
- **JavaScript**: Como lenguaje de programación encargado de la funcionalidad de la aplicación.
- **HTML:** Lenguaje de marcas.

A continuación, se van a nombrar las principales bibliotecas utilizadas durante el desarrollo del servidor, algunas de estas bibliotecas ya se han definido en el apartado de técnicas y herramientas:

- **Axios**: Biblioteca utilizada para realizar las peticiones HTTP a una API. En nuestro caso esta biblioteca ha sido utilizada para realizar las peticiones al servidor Python.
- **Bcrypt**: Biblioteca utilizada para el encriptado de la contraseña de los usuarios del sistema.
- **Vuetify**: Biblioteca de Vue que concede al programar componentes con los que poder crear una interfaz de usuario equilibrada.

- **Vuex**: Biblioteca de Vue que permite la compartición de información entro los diferentes componentes que forman la aplicación web. Ha sido utilizada para poder evitar posibles casos de duplicidad de información o de peticiones al servidor repetidas.
- **Vue2-Google-Maps**: Componentes de Vue que permite la utilización del servicio de Google maps en la aplicación. Como ocurre con otros servicios de Google, para su utilización ha sido necesario la generación de una clave privada.
- **Plotly**: Biblioteca de Vuejs utilizada para la representación, en forma de graficas de distinto tipo, de los datos obtenidos durante el proceso análisis con YOLO.
- **Socketio**: Biblioteca de Vue a través de la cual se puede realizar la recepción y envío de eventos con una latencia baja.

6.7.2.1 EJEMPLO DEL CÓDIGO

A continuación, se van a mostrar algunas partes del código empleado en el desarrollo de la aplicación web.

En la *Figura 49,* se muestra el código empleado para el envío de una petición HTTP al servidor de Python. En este caso, la petición realizada corresponde a un POST, a través del cual se va a realizar el envío de un nombre de usuario.

```
return axios.post('/forgotPassword',{
  username:data
}).then(response => {
  let success = response.data.success;
  if(success === true) {
    return {'status':response.status}
  }else {
    return {'status':response.status,'message':response.data.message}
  }
}).catch(error => {
  return {'status':error.response.status,'message':error.response.data.message}
})
```

Figura 49. Ejemplo de una petición POST al servidor de Python.

En la *Figura 50,* se muestra el fichero utilizado para la creación de los diferentes endpoints utilizados en las llamadas a las API de Python. Como se puede observar, las llamadas realizadas tendrán adjunto un token, el cual se comprobará posteriormente en la API.

```
import axios from 'axios'

const instance = axios.create({
    baseURL: process.env.VUE_APP_API_URL || 'https://127.0.0.1:5000',
    headers: {
        'Authorization': {
            toString() {
                return `Bearer ${localStorage.getItem('token')}`
            }
        }
    }
});

export default instance
```

Figura 50. Endpoint utilizado en las llamadas a la API.

6.7.2.2 DESARROLLO DE LA APLICACIÓN WEB

Antes de realizar ninguna implementación de la aplicación web, se han realizado una serie de bocetos en papel de cada una de las interfaces que los usuarios van a encontrar en la aplicación web. La *Figura 51* muestra el boceto realizado para la interfaz que se encontraría un usuario al entrar en la aplicación.

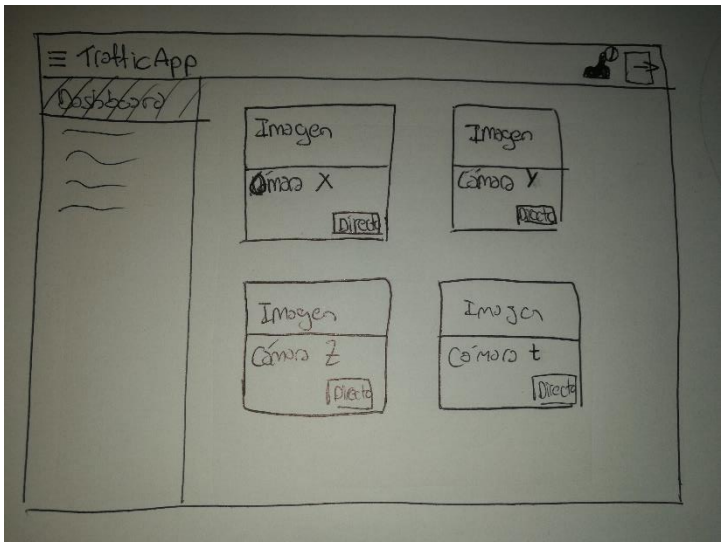

Figura 51. Boceto en papel de la página de inicio de la aplicación web.

A continuación, se muestra en la *Figura 52* la interfaz de la página de inicio implementada en Vuejs.

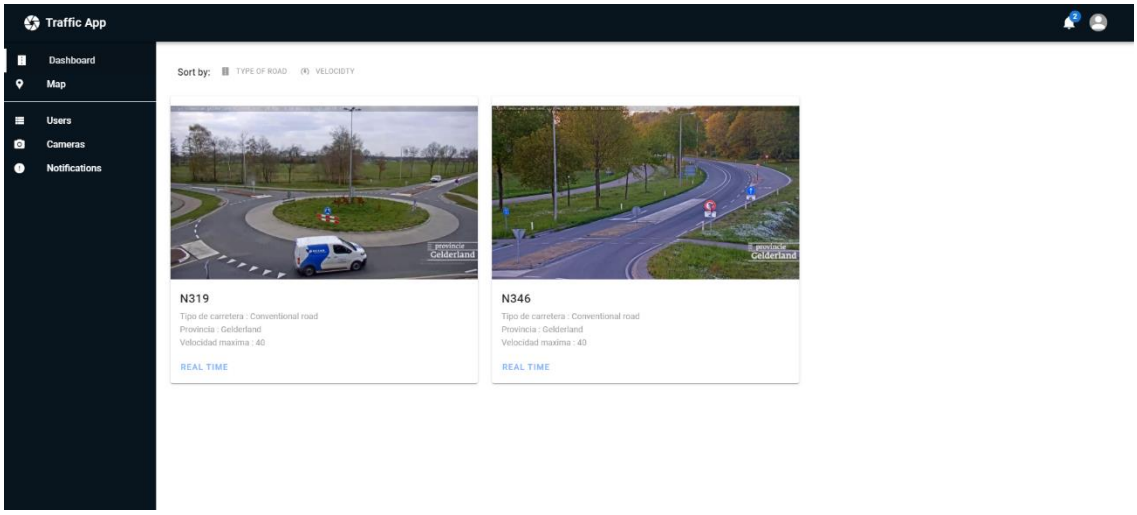

Figura 52. Página de inicio implementada en Vuejs.

Esto mismo ocurre con el boceto realizado de la página encargada de la visualización en directo de las imágenes de la cámara. En la *Figura 53*, se puede observar el boceto realizado.

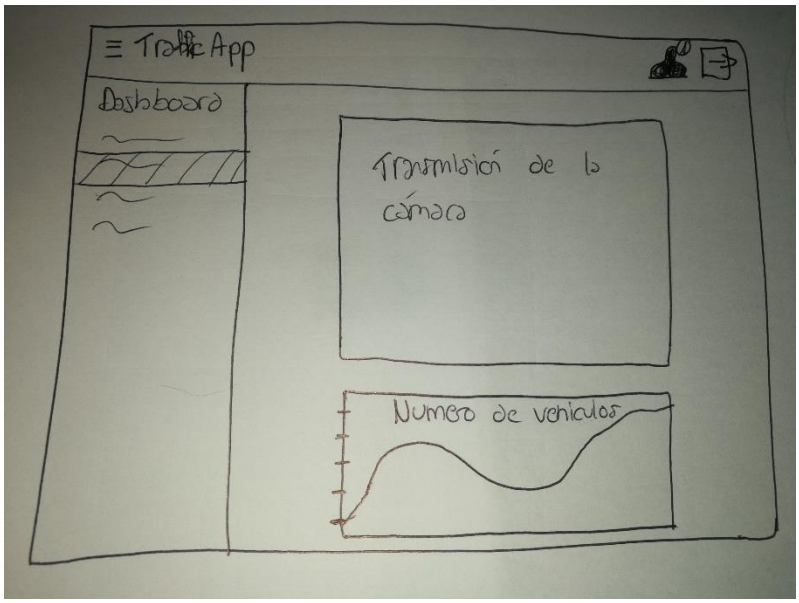

Figura 53. Boceto en papel de la página donde se realiza la transmisión de las imágenes de la cámara.

A continuación, se puede observar en la *Figura 54* la interfaz desarrollada para la página encargada de la retransmisión de las imágenes de la cámara.

Figura 54. Página encargada de la transmisión en tiempo real en Vuejs.

6.8 PRUEBAS

La realización de pruebas es una de las partes importantes del desarrollo, ya que a través de estas pruebas se puede comprobar el correcto funcionamiento del sistema que se está implementando.

Durante el desarrollo del sistema, se han ido realizando pruebas independientes de cada uno de los paquetes implementados, con el objetivo de comprobar que cada uno de ellos funcione correctamente.

Una vez finalizadas las pruebas anteriormente comentadas, se han realizado pruebas generalizadas del sistema, con el objetivo de comprobar que la interacción entre los módulos implementados es correcta y no presenta ningún fallo de gravedad.

6.9 FUNCIONALIDAD DEL SISTEMA

En este aparatado se van a poder observar las diferentes funcionalidades disponibles en la aplicación web desarrollada. Hay que destacar, que en función de los permisos que posea el usuario que ha accedido a al sistema, este tendrá una mayor o menor funcionalidad.

El *Anexo VI: Manual de usuario* muestra de forma más detallada la funcionalidad del sistema.

6.9.1 USUARIO NO IDENTIFICADO

Este usuario del sistema va a encontrar escasas funcionalidades en la aplicación web, ya que para poder ver información del sistema y tener una mayor funcionalidad será necesario identificarse.

6.9.1.1 INICIO DE SESIÓN

En la *Figura 55* se muestra la interfaz, la cual va a ser utilizada por el usuario no identificado, para acceder al sistema.

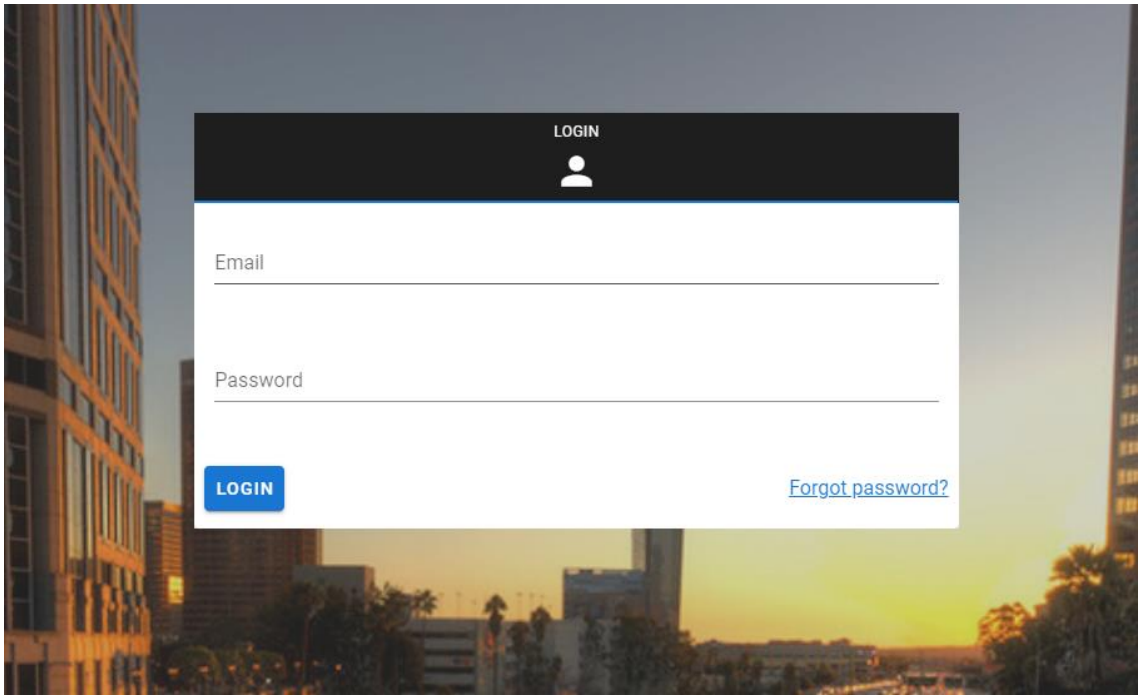

Figura 55. Inicio de sesión.

En este interfaz se podrán observar diferentes mensajes de error o mensajes informativos del sistema en función de las acciones que realice el usuario no identificado. Por ejemplo, en el caso de que el usuario no introduzca datos o sean erróneos, se le notificará dicho error. Este tipo de comprobaciones se realizarán en los casos que sean necesarios.

6.9.1.2 RECUPERAR CONTRASEÑA

Para acceder al procedimiento de recuperación de la contraseña será necesario pulsar el enlace correspondiente a "*Forgot password*". Este enlace se observa en la *Figura 56.*

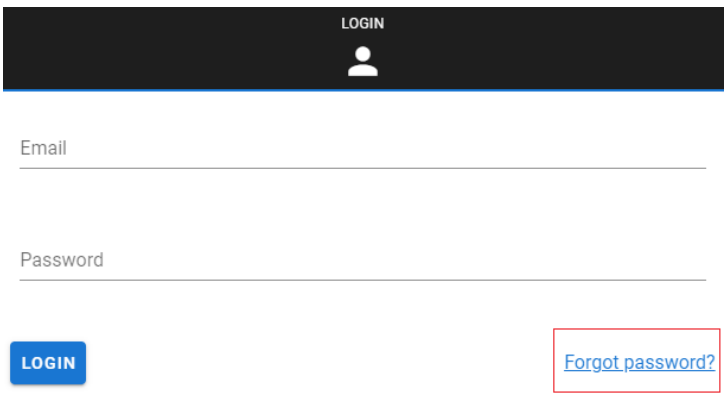

Figura 56. Enlace para acceder a recuperar contraseña.

En la *Figura 57,* se muestra la página destinada para el comienzo de la recuperación de la contraseña. En el caso de que el usuario no quiera continuar con el proceso de recuperación de la contraseña, simplemente necesitará dar al botón de "*Login*" para volver a la página de inicio de sesión.

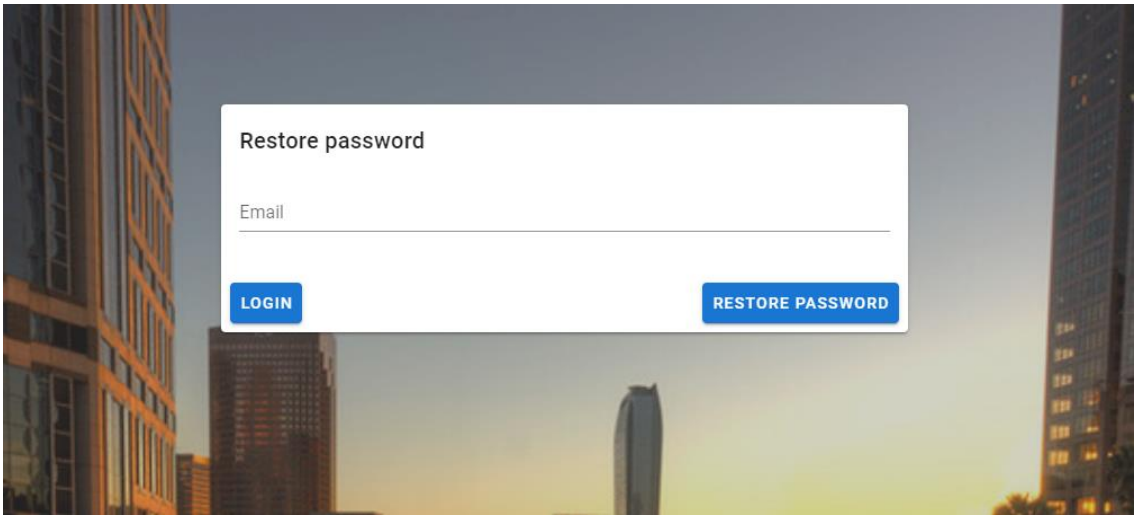

Figura 57. Página destinada a la recuperación de la contraseña.

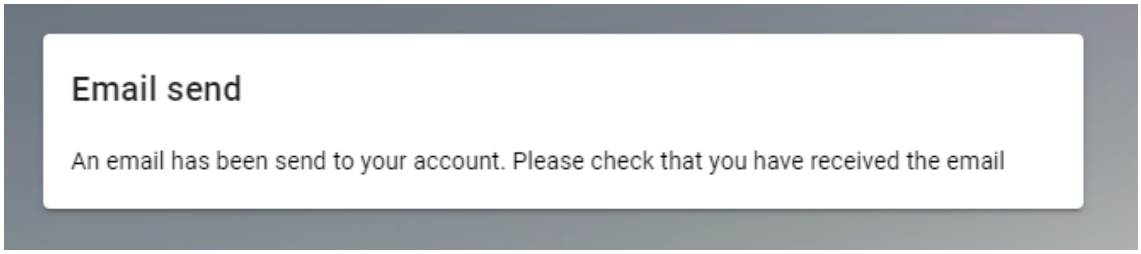

Figura 58. Mensaje informativo de envío de correo.

Una vez realizado el proceso anterior, se mostrará el mensaje informativo que se puede observar en la *Figura 58.*

6.9.1.3 ESTABLECER NUEVA CONTRASEÑA

En la *Figura 59,* se muestra la interfaz con la que se va a encontrar el usuario no identificado en el caso de querer establecer una nueva contraseña. Este únicamente tendrá que rellenar los campos con la nueva contraseña y su repetición. Una vez rellenado correctamente los datos, su contraseña se actualizará y se redireccionará al inicio de sesión.

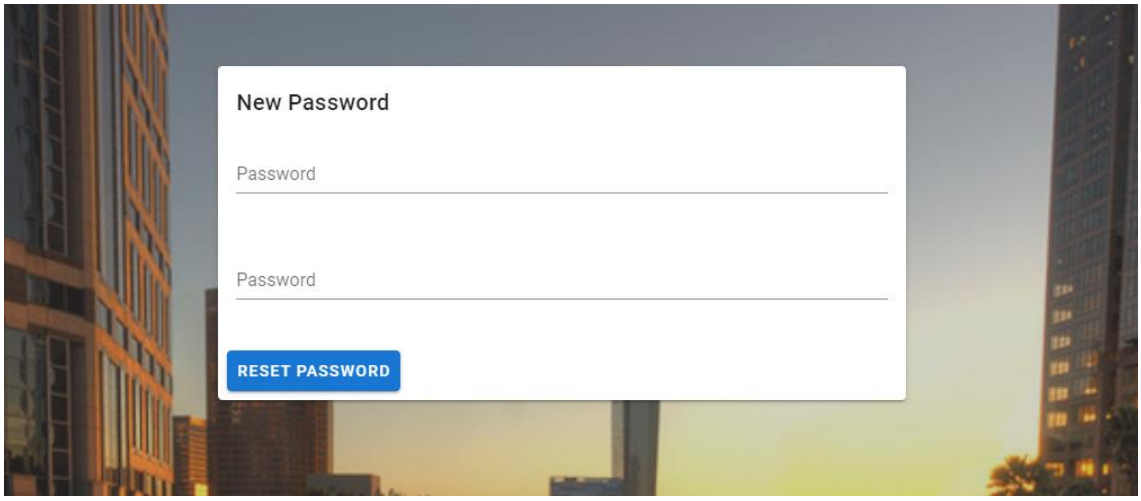

Figura 59. Página para establecer una nueva contraseña.

6.9.2 USUARIO IDENTIFICADO

El usuario identificado corresponde al usuario del sistema con los permisos más bajos. Este usuario únicamente tendrá un acceso limitado a la información del sistema, ya que parte de ella se encontrará restringida por la falta de permisos.

6.9.2.1 DASHBOARD

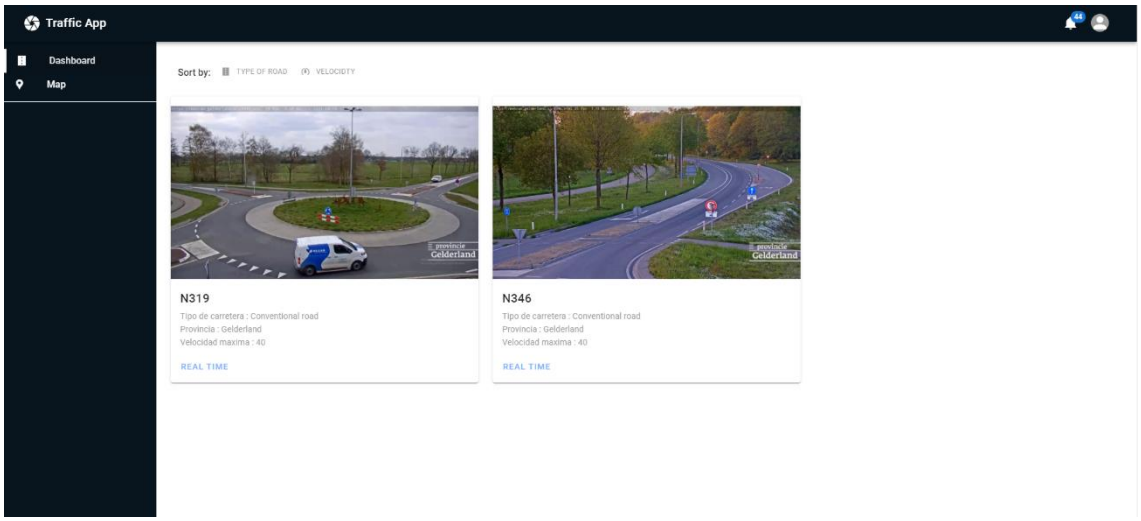

Figura 60. Dashboard.

La *Figura 60* muestra la página principal, en esta aparecerán las diferentes cámaras registradas en el sistema hasta ese momento. Sobre cada una de ellas podremos observar parte de su información, como puede ser el nombre de la carretera en la que se encuentra el dispositivo, el tipo de carretera, la velocidad máxima y la provincia en la que se encuentra.

6.9.2.2 NOTIFICACIONES

En la parte superior de la página podremos observar las diferentes notificaciones que han sido recibidas por el usuario. Estas notificaciones se pueden observar en la *Figura 61.*

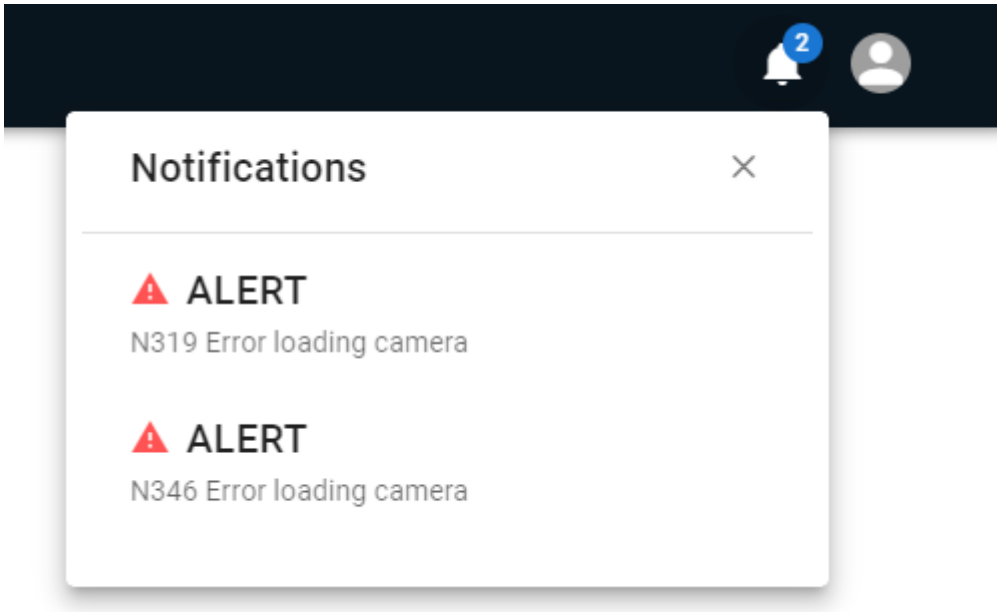

Figura 61. Notificaciones del usuario.

6.9.2.3 CERRAR SESIÓN

Para cerrar la sesión de un usuario identificado, se necesitará ir a la parte superior derecha de la página. Este se muestra en la *Figura 62.*

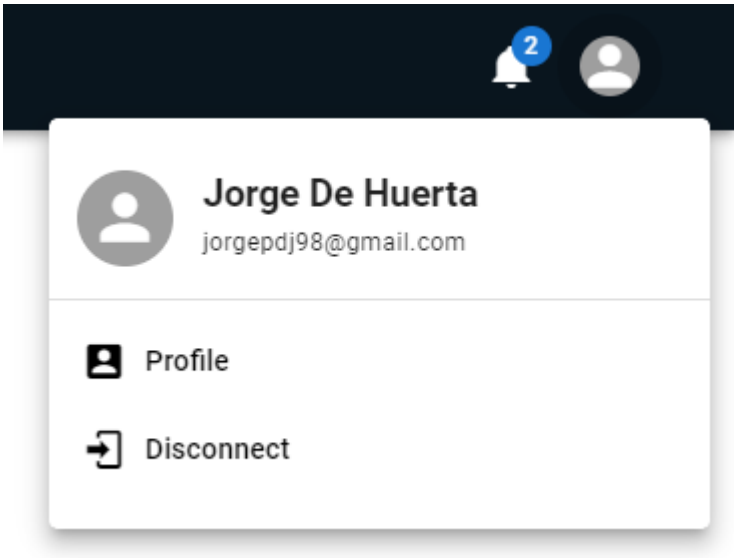

Figura 62. Cerrar sesión.

6.9.2.4 NAVEGACIÓN

Para poder navegar a través de las diferentes páginas, se va a utilizar el menú lateral izquierdo. En este caso dicho menú se encontrará de forma permanentemente en todas la paginas de la aplicación web. Este menú de navegación lateral se puede observar en la *Figura 63.*

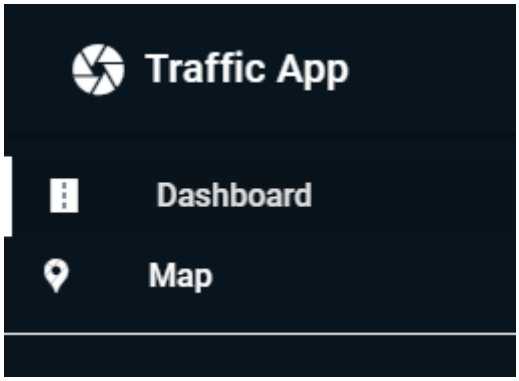

Figura 63. Menú de navegación lateral.

6.9.2.5 MAPA

En la *Figura 64,* se puede observar la página del mapa geográfico. Esta página ofrece imágenes de mapas desplazables, en las cuales se podrán observar las localizaciones de cada uno de los dispositivos registrados en el sistema.

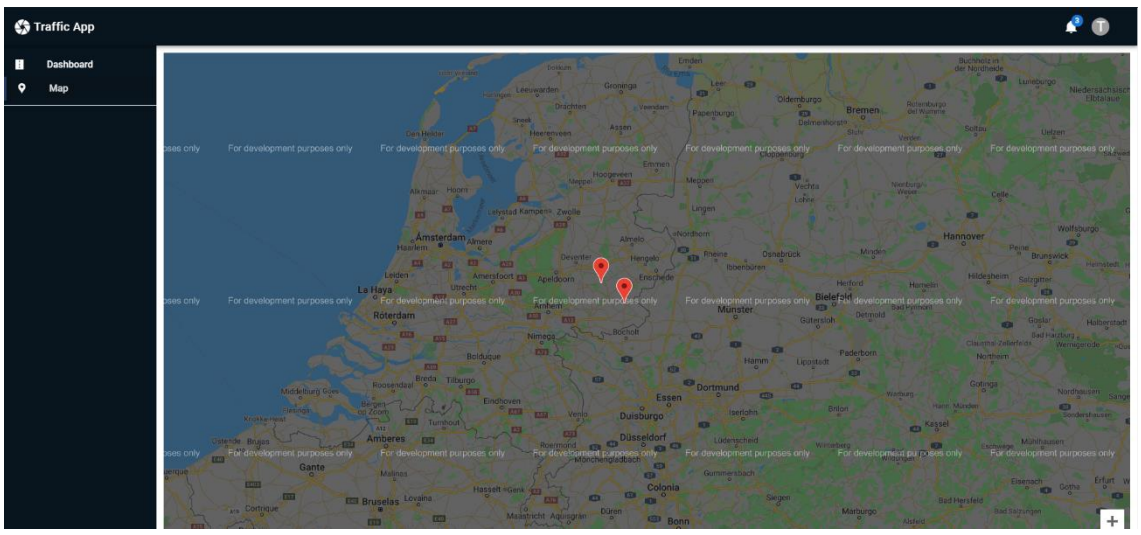

Figura 64. Mapa.

6.9.2.6 PERFIL DE USUARIO

Para acceder al perfil, se necesitará pulsar el botón de account, el cual se encuentra en el mismo desplegable donde se cierra la sesión de un usuario. En el perfil del usuario se podrán editar el propio perfil del usuario y su contraseña. En la *Figura 65* se puede observar la página encargada de mostrar la información del usuario identificado.

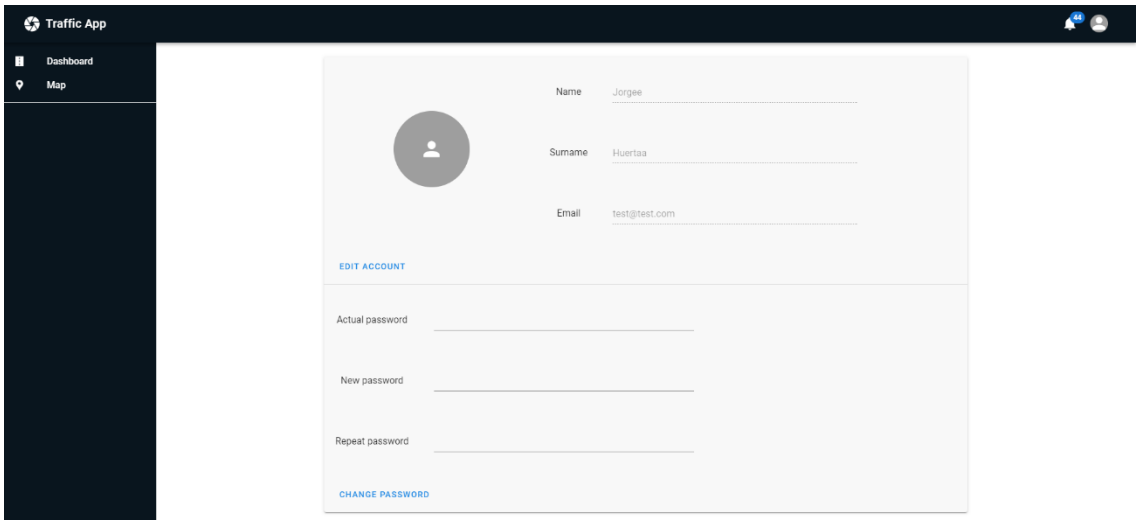

Figura 65. Perfil del usuario identificado.

6.9.2.7 TRANSMISIÓN EN TIEMPO REAL

En esta página se podrá observar en tiempo real las imágenes que la cámara captura, así como las estadísticas que se generan como consecuencia del análisis de las mismas. Todo esto se puede observar en la *Figura 66, Figura 67, Figura 68 y Figura 69.*

Figura 66. Transmisión en tiempo real 1.

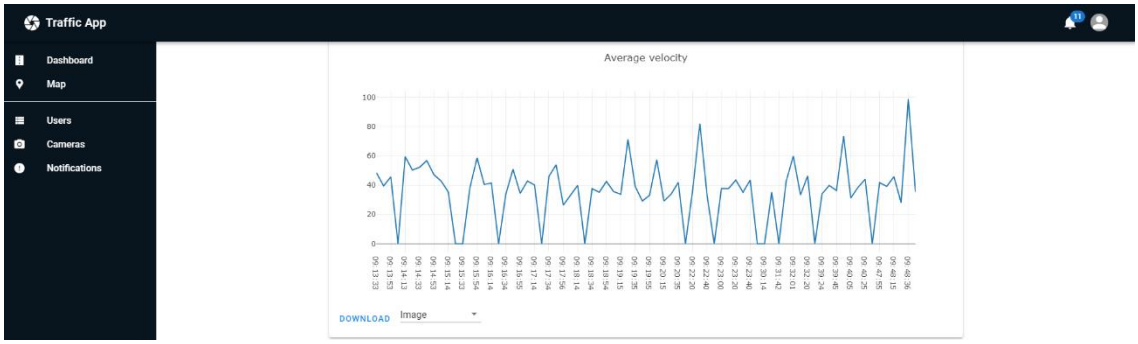

Figura 67. Transmisión en tiempo real 2.

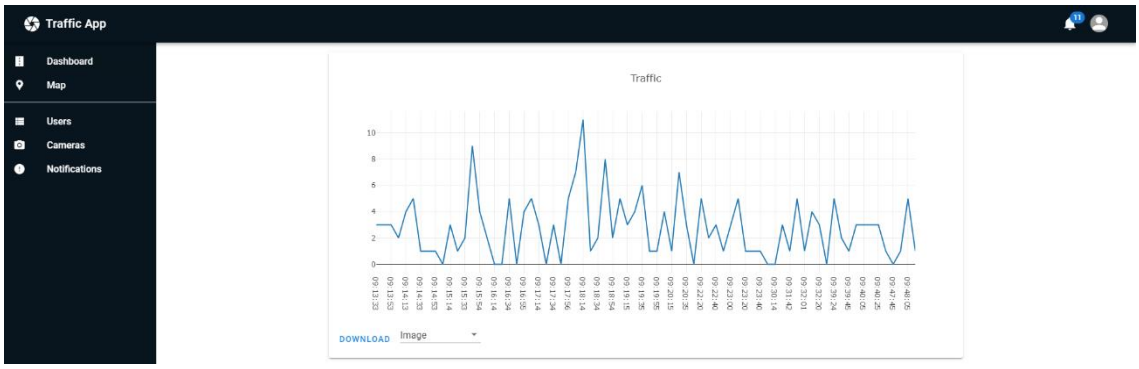

Figura 68. Transmisión en tiempo real 3.

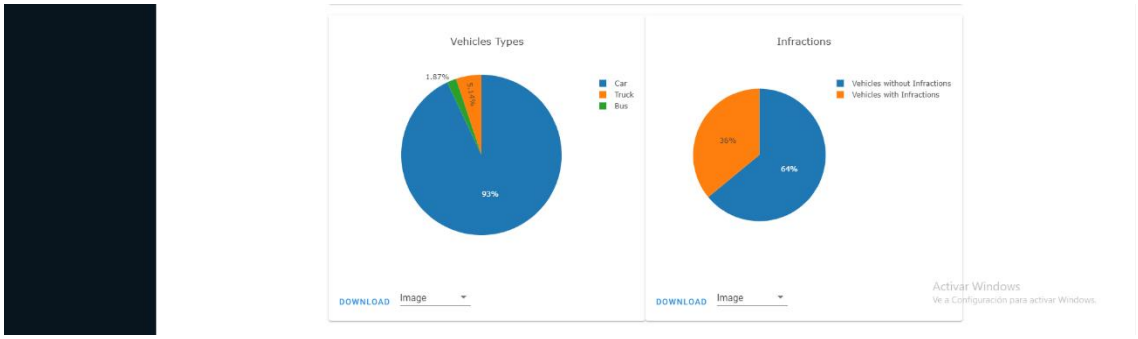

Figura 69. Transmisión en tiempo real 4.

6.9.3 ADMINISTRADOR

El administrador, corresponde al usuario con los permisos más altos del sistema. Este usuario podrá realizar modificaciones sobre los diferentes datos mostrados en la aplicación web.

6.9.3.1 NAVEGACIÓN

En el caso del usuario administrador, el menú de navegación lateral mostrara un número mayor de opciones a las que navegar. Estas opciones se pueden ver en la *Figura 70.*

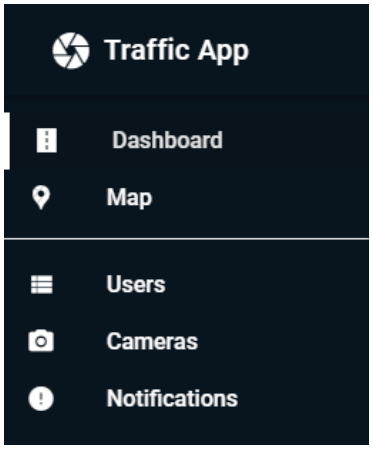

Figura 70. Menú de navegación del administrador.

6.9.3.2 LISTADO DE USUARIOS

Para acceder al listado de usuarios serán necesarios los permisos de administrador. A este listado únicamente se va a poder acceder desde el menú de navegación lateral.

En esta página encontraremos un listado de los diferentes usuarios registrados del sistema, el administrador podrá añadir nuevos usuarios, eliminarlos o modificarlos. Esta página se puede observar en la *Figura 71.*

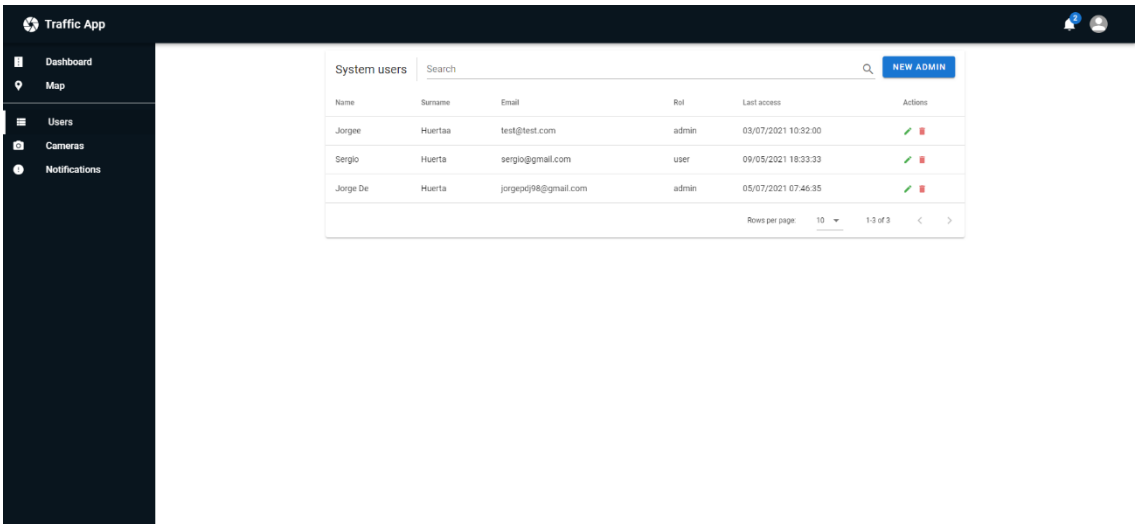

Figura 71. Listado de usuarios del sistema.

6.9.3.3 LISTADO DE NOTIFICACIONES

Al igual que en el caso anterior, para acceder al listado de las notificaciones del sistema se necesitarán permisos de administrador, y la única forma de llegar a esta página es a través del menú de navegación lateral. Este listado de notificaciones se puede observar en la *Figura 72.*

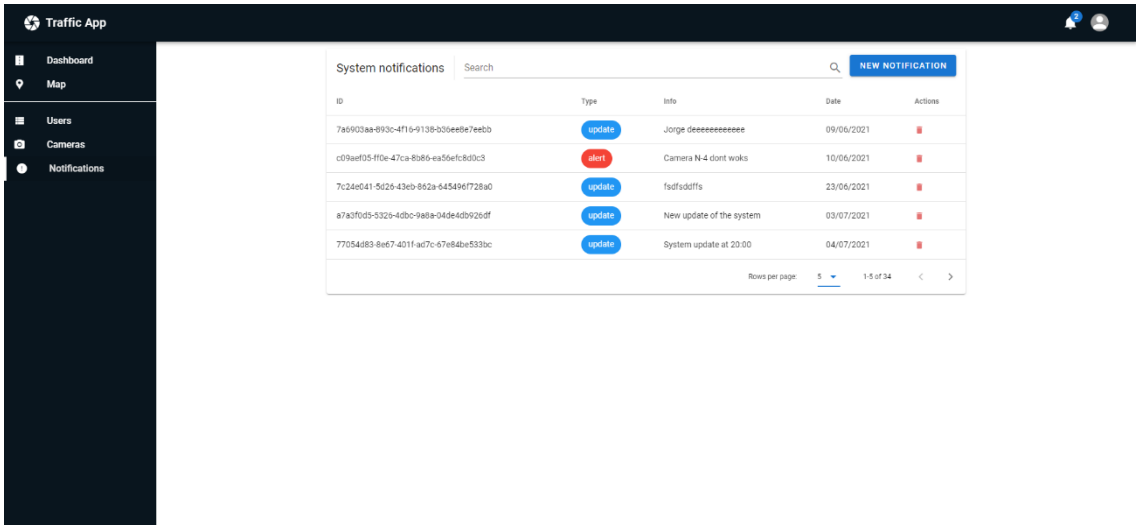

Figura 72. Listado de notificaciones.

En este caso, el administrador podrá añadir nuevas notificaciones, pudendo seleccionar el tipo de notificación que desea crear. También podrá realizar el proceso de eliminación de estas.

6.9.3.4 LISTADO DE CÁMARAS

Los administradores también dispondrán de un listado de las diferentes cámaras registradas, este listado es ligeramente diferente al que pueden ver los usuarios identificados. Al igual que en los casos anteriores, el administrador podrá registrar una nueva cámara, eliminar una de las cámaras registradas o modificar su velocidad máxima. Esta página se puede observar en la *Figura 73.*

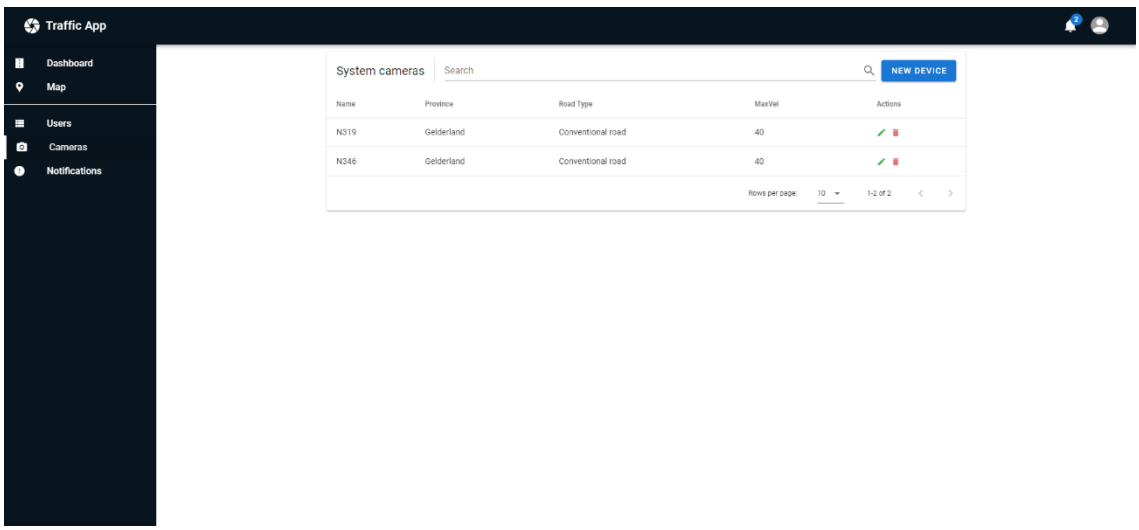

Figura 73. Listado de cámaras.

7. LIMITACIONES DEL PROTOTIPO

Durante el desarrollo del proyecto han surgido diversos problemas debido a factores como el hardware o el alto precio de los componentes necesarios para prestar un servicio con un mayor rendimiento.

- **Capacidad computacional limitada**: Para la realización del proyecto de fin de grado se ha utilizado una tarjeta gráfica 1650 de Nvidia. Esta gráfica, considerada actualmente como una gráfica de gama media-baja, no posee una gran capacidad computacional. Esta capacidad ha supuesto una limitación durante el desarrollo debido a que únicamente ha sido posible el análisis simultaneo de dos videos.
- **Hardware costoso**: Relacionado con el punto anterior, actualmente para poder realizar un análisis en tiempo real se necesita de un hardware con una gran capacidad de computación, pero a medida que aumenta la potencia computacional de un hardware también aumenta el precio de este. Además, cabe comentar que actualmente el mercado de las tarjetas gráficas, hardware necesario para realizar el análisis de una imagen, se ha visto afectado por el aumento del precio de las criptomonedas, generando de esta manera el aumento desorbitado de sus precios.
- **Características atmosféricas**: Las características atmosféricas afectan negativamente a nuestro prototipo por diversas razones. En caso de niebla espesa o lluvias torrenciales el modelo de visión artificial puede generar ciertos fallos de detección. Otro de los problemas puede surgir en el caso de fuertes vientos, ya que, si la cámara se mueve, esto puede generar fallos a la hora de calcular la velocidad de los vehículos.
- **Limitaciones geográficas**: Al igual que ocurre con los radares actuales, en el caso de que algún objeto obstaculice la visión de la cámara esta tendrá ciertas limitaciones a la hora de realizar las detecciones de vehículos y de infracciones.

8. CONCLUSIONES Y LÍNEAS DE TRABAJO FUTURAS

Por último, en este apartado se van a comentar las conclusiones obtenidas tras realizar el desarrollo del sistema y las posibles líneas de trabajo futuras que podrían permitir en un futuro la mejora y posible comercialización del nuestro producto.

8.1 CONCLUSIONES

Una vez alcanzado todos los objetivos y requisitos planteados al inicio del desarrollo, podremos considerar que el proyecto se ha finalizado satisfactoriamente. A continuación, se van a enumerar las diferentes conclusiones que se han podido sacar como consecuencia del desarrollo.

- Se ha logrado el objetivo de gestión de usuarios, ya que se ha desarrollado satisfactoriamente un sistema de control de usuarios a través de roles.
- Se ha logrado el objetivo de gestión de cámaras, ya que el sistema permite la modificación, eliminación y creación de estas.
- Se ha logrado el objetivo de información en tiempo real, ya que el sistema permite la visualización de las imágenes analizadas de la cámara en tiempo real.
- Se ha logrado el objetivo de gestión de estadísticas, ya que el sistema permite la visualización de las diferentes estadísticas generados como resultado del análisis de las imágenes.
- Se ha logrado el objetivo de gestión de geolocalización de las cámaras, ya que el sistema permite la geolocalización de las diferentes cámaras a través de sus coordenadas.
- Se ha logrado el objetivo de gestión de notificaciones, ya que el sistema permite la visualización, creación y eliminación de las diferentes notificaciones del sistema.
- Se ha logrado un desarrollo completo desde la etapa de conceptualización de la idea hasta la propia implantación y desarrollo de un prototipo completamente funcional.
- La implementación de este proyecto me ha permitido entrar en contacto con algunas de las tecnologías actualmente más punteras en temas de detección de objetos a través de una red neuronal convolucional, en este caso estoy hablando de YOLO.
- El desarrollo del proyecto me ha permitido aplicar diferentes conocimientos adquiridos en las asignaturas de ingeniería del software y gestión de proyectos. Además, el desarrollo del proyecto me ha permitido darme cuenta de la importancia que tienen la planificación del mismo antes de empezar cualquier tipo de desarrollo.
- El desarrollo del proyecto me ha facilitado entrar en contacto con nuevas herramientas y lenguajes de programación que en un futuro laboral me van a ser de utilidad. También se ha podido observar la importancia que tiene la selección de herramientas específicas en algunas etapas de la implementación, como es el caso de la utilización del framework Vuejs para la creación de la aplicación web.

8.2 LÍNEAS DE TRABAJO FUTURAS

A continuación, se van a comentar algunas de las posibles mejoras que se pueden aplicar al prototipo desarrollado. Entre las posibles líneas de trabajo futuras se encuentran:

- Utilización de placas Jetson Xavier para realización del análisis de las imágenes captadas por la cámara, de esta manera se podría evitar que todas las imágenes se analicen de manera centralizada por un único equipo. Esta mejora permitiría un aumento considerable del rendimiento de la aplicación, evitando además cuellos de botella innecesarios.
- Añadir nuevas funcionalidades a la aplicación y refinar las que ya se encuentran implementadas. Entre las nuevas funcionalidades se podría encontrar, el crear un listado con las imágenes de los diferentes infractores, permitiendo de esta manera al personal competente castigar con multas a los conductores que se hayan saltado el límite establecido para dicha carretera.

9. BIBLIOGRAFÍA

[1] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S. Lew. Deep learning for visual understanding: A review, 2016

[2] T. Young, D. Hazarika, S. Poria and E. Cambria, "Recent Trends in Deep Learning Based Natural Language Processing [Review Article]," in IEEE Computational Intelligence Magazine, vol. 13, no. 3, pp. 55-75, Aug. 2018, doi: 10.1109/MCI.2018.2840738.

[3] arXiv:1808.05697 [cs.CL]

[4] S. M. S. Islam, S. Rahman, M. M. Rahman, E. K. Dey and M. Shoyaib, "Application of deep learning to computer vision: A comprehensive study," 2016 5th International Conference on Informatics, Electronics and Vision (ICIEV), 2016, pp. 592-597, doi: 10.1109/ICIEV.2016.7760071.

[5] DotCSV ¡APRENDE Qué son las Redes Neuronales CONVOLUCIONALES! (s. f.). Recuperado 29 de junio de 2021, de https://www.youtube.com/watch?v=V8j1oENVz00

[6] arXiv:1506.02640 [cs.CV]

[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(9):1627–1645, 2010.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 580–587. IEEE, 2014.

[9] Detección de móviles

- https://roadsafety.transport.nsw.gov.au/stayingsafe/mobilephones/technology.html#fa2
- https://www.bbc.com/news/technology-50630763

[10] Tesla autopilot

- https://www.tesla.com/es_ES/autopilot
- https://www.tesla.com/support/autopilot

[11] Python.

- https://www.python.org/doc/essays/blurb/
- https://www.w3schools.com/python/python_intro.asp

[12] Pip

- https://realpython.com/what-is-pip/

[13] Flask

- https://flask.palletsprojects.com/en/2.0.x/

- https://es.wikipedia.org/wiki/Flask

[14] Pycharm

- https://www.jetbrains.com/es-es/pycharm/
- https://en.wikipedia.org/wiki/PyCharm

[15] Flask-SocketIO

- https://flask-socketio.readthedocs.io/en/latest/

[16] Postman

- https://www.postman.com/

[17] Firebase

- https://es.wikipedia.org/wiki/Firebase
- https://firebase.google.com/docs
- https://blog.back4app.com/es/comparacion-entre-aws-amplify-y-google-firebase/
- https://firebase.google.com/docs/database?hl=es
- https://firebase.google.com/docs/storage?hl=es

[18] Vuex

- https://vuex.vuejs.org/
- https://vuex.vuejs.org/guide/mutations.html#commit-with-payload
- https://vuex.vuejs.org/guide/actions.html#dispatching-actions
- https://vuex.vuejs.org/guide/state.html

[19] MVVM

- https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93modelo_de_vista

[20] Vuejs

- https://es.vuejs.org/v2/guide/single-file-components.html
- https://vuejs.org/

[21] Vuetify

- https://vuetifyjs.com/en/introduction/why-vuetify/#getting-started

[22] Axios

- https://es.vuejs.org/v2/cookbook/using-axios-to-consume-apis.html

[23] Plotly

- https://plotly.com/javascript/

[24] Vue2-Google-Maps

- https://www.npmjs.com/package/vue2-google-maps

[25] NPM

- https://es.wikipedia.org/wiki/Npm

[26] Visual Studio Code

- https://es.wikipedia.org/wiki/Visual_Studio_Code
- https://code.visualstudio.com/learn

[27] Herramienta CASE

- https://es.wikipedia.org/wiki/Herramienta_CASE

[28] Microsoft Project

- https://es.wikipedia.org/wiki/Microsoft_Project
- https://www.microsoft.com/es-es/microsoft-365/project/project-management-software

[29] Draw.io

- https://www.computerhope.com/jargon/d/drawio.htm

[30] Git

- https://git-scm.com/

[31] Vision artificial

- https://es.wikipedia.org/wiki/Visi%C3%B3n_artificial
- https://aprendeia.com/vision-computacional/

[32] Deep learning

- https://www.xataka.com/robotica-e-ia/deep-learning-que-es-y-por-que-va-a-ser-una-tecnologia-clave-en-el-futuro-de-la-inteligencia-artificial
- https://www.indracompany.com/es/blogneo/deep-learning-sirve

[33] Formula del semiverseno

- https://en.wikipedia.org/wiki/Haversine_formula

[34] García Peñalvo, F. J., Moreno García, M. N. Diapositivas de Ingeniería del Software I, Tema 4.

[35] Jacobson, I., Booch, G., Runbaugh, J. The Unified Software Development Process.

[36] Durán Toro, A., Bernárdez Jiménez, B. (2002). Metodología para la Elicitación de Requisitos de Sistemas Software (versión 2.3).

[37] Moreno García, M. N. Transparencias de Ingeniería del Software II, Tema 3.

[38] https://www.ittrends.es/infraestructura/2020/12/camaras-inteligentes-para-gestionar-el-trafico-en-las-carreteras

[39] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed,D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. CoRR, abs/1409.4842,2014.

[40] arXiv:2004.10934 [cs.CV]

[41] https://opencv.org/about/

[42] L. Wen, D. Du, Z. Cai, Z. Lei, M. Chang, H. Qi, J. Lim, M. Yang, and S. Lyu. UA-DETRAC: A new benchmark and protocol for multi-object tracking. CoRR, abs/1511.04136,2015.

[43] https://github.com/AlexeyAB/darknet/wiki/CFG-Parameters-in-the-different-layers

[44] N. Wojke, A. Bewley and D. Paulus, Simple online and realtime tracking with a deep association metric, 2017 IEEE International Conference on Image Processing, pp.3645-3649, 2017.

[45]https://www.researchgate.net/publication/344099630_Object_Tracking_Using_Improved_Deep_Sort_YOLOv3_Architecture

[46] CSS

- https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada
- https://www.tutorialspoint.com/css/what_is_css.htm

[47] JavaScript

- https://es.wikipedia.org/wiki/JavaScript
- https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/JavaScript_basics

[48] HTML

- https://es.wikipedia.org/wiki/HTML
- https://www.w3schools.com/html/html_intro.asp

[49] TrafiCam AI

- https://www.flir.es/products/traficam-ai/

[50] Velocidad

- http://graphics.cs.cmu.edu/courses/15-463/2008_fall/Papers/proj.pdf
- https://medium.com/hal24k-techblog/how-to-track-objects-in-the-real-world-with-tensorflow-sort-and-opencv-a64d9564ccb1

[51] https://colab.research.google.com/notebooks/welcome.ipynb?hl=es