



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería  
Informática**

**Control de acceso a  
edificios de vehículos  
mediante visión artificial**



Presentado por Pablo Sainz Pino  
en Universidad de Burgos — 07 de junio  
de 2025

Tutores: José Manuel Galán Ordax y  
José Ignacio Santos Martín







UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



D. José Manuel Galán Ordax, profesor del departamento de Ingeniería de Organización, área de Organización de Empresas.

Expone:

Que el alumno D. Pablo Sainz Pino, con DNI 71314321S, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 07 de junio de 2025

Vº. Bº. del Tutor:

Vº. Bº. del co-tutor:

D. José Manuel  
Galán Ordax

D. José Ignacio Santos Martín





## **Resumen**

Este proyecto utiliza modelos de detección basados en visión por computador para crear una aplicación que permita detectar, contar y clasificar vehículos en vídeos y a su vez permita la gestión y explotación de datos mediante dashboards interactivos que muestran estadísticas, gráficos y tablas relacionados con la movilidad en la Universidad de Burgos.

## **Descriptores**

Visión por computador, YOLO, flask, aplicación web, detección de vehículos, redes neuronales, JavaScript, KPIs, explotación de datos.

## **Abstract**

This project uses a detection model based on computer vision to create an application to detect, count and classify vehicles in videos and in turn allows the management and exploitation of data through interactive dashboards that display statistics, graphs and tables related to mobility at the University of Burgos.

## **Keywords**

Computer vision, YOLO, flask, web application, vehicle detection, neural networks, JavaScript, KPIs, data mining.



---

# Índice general

---

Índice general.....	9
Índice de figuras.....	11
Índice de tablas.....	12
Índice de ecuaciones.....	13
1. Introducción.....	1
1.1. Estructura de la memoria .....	2
1.2. Estructura de los anexos.....	3
2. Objetivos del proyecto.....	4
2.1. Objetivos generales.....	4
2.2. Objetivos técnicos.....	5
3. Conceptos teóricos.....	6
3.1. Visión por computador .....	6
3.2. Redes neuronales convolucionales .....	8
3.3. You Only Look Once (YOLO).....	9
4. Técnicas y herramientas .....	11
4.1. Python .....	11
4.2. Visual Studio Code .....	11
4.3. Flask.....	12
4.4. MySQL.....	12
4.5. Github.....	12
4.6. Zube.....	13
4.7. Zotero.....	13
4.8. Ultralytics .....	13

4.9.	OpenCV .....	14
4.10.	PyMediaInfo.....	14
4.11.	Shapely.....	14
4.12.	GeoJSON .....	15
4.13.	Chart.js.....	15
4.14.	DataTables.....	15
5.	Aspectos relevantes del desarrollo del proyecto .....	16
6.	Trabajos relacionados.....	27
6.1	Plate Recognizer .....	27
6.2	Amazon Rekognition.....	28
7.	Conclusiones y Líneas de trabajo futuras.....	30
7.1	Conclusiones.....	30
7.2	Líneas de trabajo futuras .....	31
	Bibliografía.....	32

---

# Índice de figuras

---

Figura 1: Detección y clasificación de objetos [3].....	7
Figura 2: Ejemplo Max-Pooling [6] .....	9
Figura 3: Arquitectura YOLO [8] .....	10
Figura 4: Comparación FPS entre modelos [26].....	18
Figura 5: Conexión con MySQL.....	18
Figura 6: Función de conexión.....	19
Figura 7: Tabla de registros.....	19
Figura 8: Ventana usuario experto.....	20
Figura 9: Ventana usuario .....	21
Figura 10: Línea de cruce personalizad.....	21
Figura 11: Facultad detectada automáticamente.....	22
Figura 12: Facultad no detectada automáticamente.....	22
Figura 13: Gráfico de resultados .....	23
Figura 14: Estadísticas generales.....	23
Figura 15: Tabla de resultados.....	24
Figura 16: Resultados del análisis del código.....	25
Figura 17: Plate Recognizer resumen [28].....	28
Figura 18: ALPR that Works [28].....	28
Figura 19: Amazon Rekognition facial analysis[32].....	29
Figura 20: Amazon Rekognition bounding box [33].....	29

---

## Índice de tablas

---

---

# Índice de ecuaciones

---

Ecuación 1: Fórmula convolución para kernels 2D [5] .....	8
Ecuación 2: Función ReLU [4] .....	8
Ecuación 3: Función softmax [7] .....	9



---

# 1. Introducción

---

Estudiar la movilidad en los campus universitarios es fundamental para entender cómo se mueven las personas y cómo se utilizan estos espacios, para así lograr una gestión más eficiente y sostenible. Saber cuántos vehículos acceden a diario a las diferentes facultades de la universidad permite identificar patrones de desplazamiento y plantear iniciativas orientadas a la reducción y control de emisiones.

Gracias a los avances en visión por computador y aprendizaje automático, es posible automatizar tareas que hasta hace poco requerían intervención humana. Esto ha permitido que la detección y conteo de vehículos sea una solución practica y accesible sin necesidad de utilizar dispositivos físicos.

Rankings de sostenibilidad universitaria internacionales como el de GreenMetrics destacan la importancia de contar con herramientas que midan el impacto ambiental en centros de enseñanza superior.[1]

En este proyecto se propone el desarrollo de un sistema capaz de detectar y contar vehículos a partir de grabaciones de vídeo de manera automática. El objetivo es automatizar la recogida de datos sobre movilidad y generar informes a partir de ello. Esto podría utilizarse en iniciativas como la de GreenMetrics para contribuir a la evaluación de sostenibilidad en la Universidad de Burgos.

## 1.1. Estructura de la memoria

- **Introducción:** Presentación del tema central que se va a abordar en el proyecto, organización del contenido de la memoria y los anexos.
- **Objetivos del proyecto:** Definición de los temas y objetivos que se tratarán a lo largo del proyecto.
- **Conceptos teóricos:** Desarrollo de los conceptos necesarios que ayudan al entendimiento del proyecto.
- **Técnicas y herramientas:** Enumeración y descripción de las metodologías y herramientas empleadas en el proyecto.
- **Aspectos relevantes del desarrollo del proyecto:** Identificación de los aspectos más relevantes surgidos durante la realización del proyecto.
- **Trabajos relacionados:** Estudios y trabajos relacionados con el ámbito de detección de objetos.
- **Conclusiones y líneas de trabajo futuras:** Conclusiones obtenidas al final del proyecto y aspectos a mejorar e implementar en un futuro.



## 1.2. Estructura de los anexos

- **Plan de proyecto software:** Planificación temporal y estudio de viabilidad tanto económica como legal del proyecto.
- **Especificación de requisitos:** Objetivos y requisitos iniciales.
- **Especificación de diseño:** Diseño de datos, procedimental, arquitectónico.
- **Documentación técnica de programación:** Explicación de los conceptos del proyecto, como la instalación, estructura de carpetas y ejecución.
- **Documentación de usuario:** Guía acerca del uso de la aplicación
- **Anexo de sostenibilización curricular:** Aspectos acerca de la sostenibilidad del proyecto

---

## 2. Objetivos del proyecto

---

En este apartado se detallan los objetivos que se quieren alcanzar con la realización del proyecto. Se distinguen entre dos objetivos:

- **Generales:** Los cuales hacen referencia a los fines principales del proyecto.
- **Técnicos:** Engloban los aspectos necesarios a implementar.

### 2.1. Objetivos generales

Los objetivos generales abordados en este proyecto son:

- Detectar y contar los vehículos que acceden a los campus de la UBU.
- Almacenar los datos obtenidos.
- Contribuir a la monitorización de la movilidad en las facultades.
- Generar informes y representaciones visuales a partir de los datos obtenidos.
- Facilitar la visualización de los datos obtenidos mediante una interfaz web.
- Generar estadísticas a partir de datos obtenidos.

## **2.2. Objetivos técnicos**

Los objetivos técnicos del proyecto son los siguientes:

- Usar un modelo basado en visión por computador para la detección y conteo de vehículos.
- Desarrollar una aplicación web que permita subir vídeos previamente grabados para su procesamiento.
- Automatizar la detección de facultad mediante la extracción de metadatos, o permitir la selección manual si no contiene datos.
- Incorporar una funcionalidad que permita al usuario definir manualmente la línea de conteo, adaptándose a cualquier vídeo.
- Implementar base de datos que almacene los registros de cada vehículo detectado.
- Mostrar informes y gráficos de los datos obtenidos.
- Utilizar Zube como herramienta para la gestión del proyecto.
- Utilizar GitHub como sistema de control de versiones.
- Utilizar Zotero como gestor de referencias

---

## 3. Conceptos teóricos

---

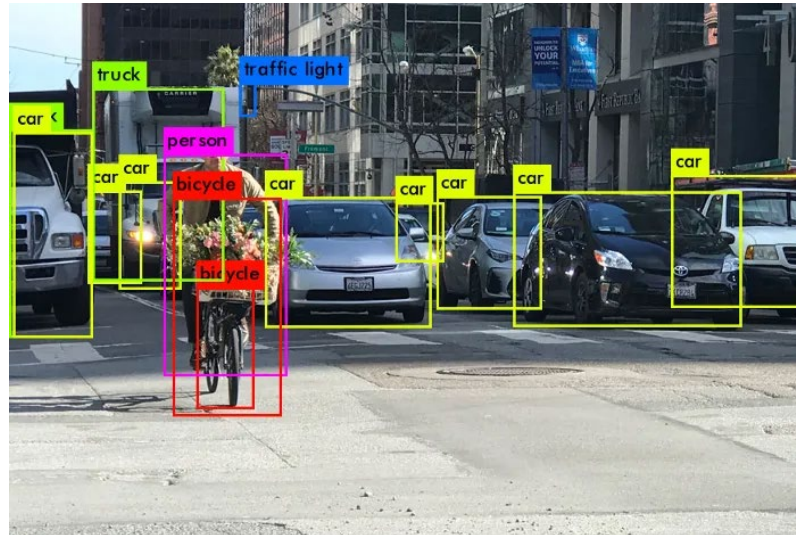
En este apartado se presentan los conceptos y aspectos teóricos del proyecto. El contenido se organiza en varios apartados, cada uno centrado en un tema relevante que aporta las bases necesarias para el desarrollo del proyecto.

### 3.1. Visión por computador

La visión por computador [2] es una rama de la inteligencia artificial cuyo objetivo es dotar a las máquinas de la capacidad de analizar e interpretar imágenes y secuencias de vídeo mediante diferentes algoritmos. Esta rama se apoya en modelos matemáticos, como análisis multiescala y ecuaciones diferenciales, para realizar tareas como la identificación de objetos o la segmentación de imágenes. (Figura 1)

La visión por computador abarca distintos niveles, entre los que se pueden encontrar:

- Detección de objetos: Identificación y localización de elementos en imágenes o vídeos.
- Clasificación de objetos: Asociación de cada objeto detectado a una categoría concreta.
- Seguimiento de objetos: Análisis del desplazamiento de elementos a lo largo del tiempo.



*Figura 1: Detección y clasificación de objetos [3]*

La figura 3.1 muestra un ejemplo de detección y clasificación de objetos, donde los elementos son categorizados e identificados con rectángulos. Estos procesos se apoyan en diferentes técnicas que facilitan el análisis preciso de la información visual. Los principales métodos son:

- Tradicionales: Se fundamentan en el cálculo matemático de características, como bordes o texturas. [2]
- Aprendizaje profundo: Utilizan redes neuronales convolucionales para detectar patrones automáticamente. [4]
- Híbridos: Combinan técnicas más clásicas con técnicas más avanzadas para mejorar la precisión y eficiencia. [4]

## 3.2 Redes neuronales convolucionales

Las redes neuronales convolucionales (CNN) son un tipo de red neuronal optimizada para procesar datos con una estructura espacial. Su diseño permite extraer características automáticamente, desde bordes o texturas hasta objetos completos, lo que las ha convertido en la herramienta por defecto para tareas de visión por computador[4], [5].

Están compuestas principalmente por:

- Capa de convolución: La operación de convolución es la que aplica filtros entrenables sobre la imagen para detectar patrones. Cada filtro se desliza por la imagen generando un mapa de activación que resalta características específicas.[5] Para un filtro  $K$  aplicado a una imagen  $I$  se define como:

$$S(i, j) = (K * I)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m, n)$$

*Ecuación 1: Fórmula convolución para kernels 2D [5]*

- Función de activación ReLU: Tras la convolución, se aplica la función Unidad Lineal Rectificada (ReLU, Rectified Linear Unit), la cual permite eliminar valores negativos. Esto facilita el entrenamiento y evita la saturación del gradiente. Se define como:

$$h(y) = \max(0, y)$$

*Ecuación 2: Función ReLU [4]*

- Capa de pooling: La operación de pooling reduce las dimensiones espaciales de los mapas de activación, disminuyendo el número de parámetros y el coste computacional. El más común es Max-Pooling, que selecciona el valor máximo de una región. [4]

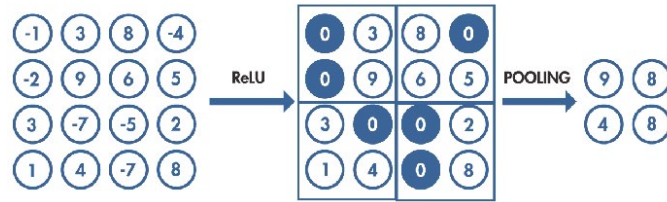


Figura 2: Ejemplo Max-Pooling [6]

- Capa de salida: Softmax: La última capa de una red neuronal convolucional suele ser la función softmax. Esta capa se encarga de convertir los valores obtenidos anteriormente en probabilidades, lo que permite asignar a cada imagen una clase concreta. [7] La función softmax se define como:

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}.$$

Ecuación 3: Función softmax [7]

### 3.3 You Only Look Once (YOLO)

YOLO (You Only Look Once) es un algoritmo de detección de objetos que se apoya en redes neuronales convolucionales. A diferencia de otros métodos más tradicionales, los cuales dividen el problema de detección en distintas fases, YOLO lo aborda como un único problema de regresión.

Este algoritmo divide la imagen de entrada en una cuadrícula, donde cada celda es responsable de predecir un número fijo de cajas delimitadoras junto a una puntuación de confianza. Esta puntuación refleja la probabilidad de presencia de un objeto y la clase a la que pertenece. Gracias a que este modelo realiza todo el proceso en una sola instancia, se consigue una detección rápida y eficiente, lo cual es ideal para aplicaciones en tiempo real. [8]

## Conceptos teóricos

La arquitectura general de YOLO, ver Figura 3, combina varias capas convolucionales para la extracción de características con capas completamente conectadas, las cuales generan las predicciones finales. A continuación, se muestra la arquitectura correspondiente a la primera versión de YOLO, la cual estaba compuesta por 24 capas convolucionales y 2 capas completamente conectadas. [8]

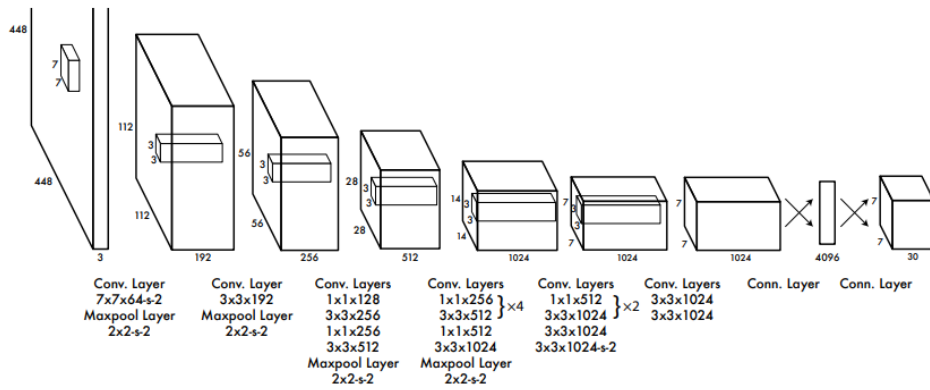


Figura 3: Arquitectura YOLO [8]



---

## 4. Técnicas y herramientas

---

En este apartado se describen las principales herramientas y aplicaciones utilizadas en el proyecto, justificando su elección y uso.

### 4.1. Python

Python [9] es un lenguaje de programación de alto nivel, multiplataforma e interpretado, por lo que no necesita compilarse para ser ejecutado. Se caracteriza por una sintaxis limpia, lo que facilita su aprendizaje.

Se ha utilizado como lenguaje principal, tanto para la detección de vehículos como para el servidor web y la conexión con la base de datos.

### 4.2. Visual Studio Code

Visual Studio Code [10] es un editor de código gratuito y multiplataforma desarrollado por Microsoft. Funciona con casi cualquier lenguaje de programación y cuenta con herramientas para depurar, control de versiones y una gran cantidad de extensiones.

En este proyecto se ha empleado para el completo desarrollo del proyecto, tanto la parte de backend como la de frontend. Además, se ha realizado el control de versiones desde VSCode gracias a su integración con Git.

### 4.3. Flask

Flask [11] es un microframework escrito en Python. Permite desarrollar aplicaciones web de forma sencilla con una instalación mínima, por lo que si se requieren más componentes cuenta con una gran cantidad de plugins.

Se ha empleado Flask sobre otras opciones gracias a recomendaciones y a la sencilla integración con la mayor parte librerías empleadas en el proyecto.

### 4.4. MySQL

MySQL [12] es un sistema de gestión de base de datos relacional de código abierto desarrollado por Oracle. Es ampliamente utilizada para aplicaciones web ya que es gratuito y de código abierto además de ofrecer:

- Escalabilidad
- Seguridad
- Alto rendimiento

Se ha optado por usar MySQL frente a otras opciones como SQLite o PostgreSQL principalmente por su seguridad, robustez y facilidad de configuración.

Se ha empleado para almacenar y gestionar los registros de los vehículos detectados, incluyendo la información relevante de cada uno. Se ha realizado la comunicación del proyecto con la base de datos mediante la librería `mysql-connector-python`. [13]

### 4.5. Github

Github [14] es una plataforma de desarrollo colaborativo que permite alojar proyectos utilizando Git como control de versiones. Ofrece herramientas muy útiles para el trabajo en equipo, como pueden ser:

- Wiki
- Visor de ramas
- Herramienta de revisión de código
- Sistema de seguimiento de problemas

## *Técnicas y herramientas*

GitHub es una de las herramientas más importantes del proyecto, ya que dentro se encuentran todos los archivos necesarios para su ejecución además del historial donde se refleja el progreso de cada sprint.

### **4.6. Zube**

Zube [15] es una herramienta de gestión de proyectos que se integra con Github. Nos permite controlar gestionar los issues, planificar tareas y organizar el trabajo mediante tableros visuales como Sprints o Kanban.

Se ha utilizado Zube para organizar las tareas de cada sprint, asignarles una dificultad y evaluar si los objetivos del sprint se han cumplido de forma adecuada.

### **4.7. Zotero**

Zotero [16] es un gestor de referencias bibliográficas gratuito y de código abierto. Permite recopilar, organizar y citar fuentes de información en diferentes formatos como APA, IEE, Vancouver.

En este proyecto, se ha empleado Zotero para generar las referencias bibliográficas de la memoria y los anexos.

### **4.8. Ultralytics**

Ultralytics [17] es una librería de Python que permite cargar modelos preentrenados como YOLO, pudiendo aplicar diferentes funcionalidades como detección, seguimiento y clasificación de objetos.

En el proyecto se ha utilizado esta librería para realizar la detección de vehículos cargando el modelo YOLOv11. La librería está integrada con PyTorch [18], por lo que se ha podido mejorar el rendimiento utilizando la aceleración por GPU.

## **4.9. OpenCV**

OpenCV [19] es una librería de visión por computador muy popular, ya que es libre, multiplataforma y cuenta con una amplia documentación. Se emplea principalmente en aplicaciones que requieren procesamiento de imágenes y vídeos.

La carga y lectura de los vídeos subidos por el usuario, así como el procesamiento y renderizado de los frames en tiempo real, se ha realizado con esta librería.

## **4.10. PyMediaInfo**

PyMediaInfo [20] es una librería de Python que actúa como interfaz de la librería MediaInfo, la cual permite extraer los metadatos de archivos multimedia.

En este proyecto se ha utilizado PyMediaInfo para obtener las coordenadas GPS de los vídeos subidos por el usuario. Esto ha permitido detectar automáticamente si el vídeo fue grabado en alguna de las facultades de la Universidad de Burgos.

## **4.11. Shapely**

Shapely [21] es una librería de Python para el análisis y manipulación de objetos geométricos planos. Permite realizar operaciones como el cálculo de distancias, intersecciones y la comprobación de pertenencia de puntos a determinadas áreas.

Se ha utilizado Shapely para determinar si las coordenadas extraídas con PyMediaInfo se encuentran dentro del área de alguna facultad.

## **4.12. GeoJSON**

GeoJSON [22] es un formato de datos basado en JSON diseñado para representar información geográfica como puntos, líneas y polígonos.

En el proyecto se empleó la herramienta `geojson.io` [23] para delimitar manualmente el área de cada facultad de la Universidad de Burgos y, posteriormente, poder verificar con `Shapely` si las coordenadas pertenecen a una de estas zonas.

## **4.13. Chart.js**

`Chart.js` [24] es una librería de JavaScript que permite generar gráficos de forma sencilla utilizando lienzos `HTML5`. Es ampliamente utilizada en aplicaciones que quieren representar grandes conjuntos de datos de forma visual. Los gráficos generados no se pueden estilar usando `CSS`, por lo que para personalizarlos se deben emplear solamente las opciones integradas en la librería.

En este proyecto se ha utilizado `Chart.js` para representar gráficamente la cantidad de vehículos detectados por tipo y día, en función del periodo seleccionado por el usuario.

## **4.14. DataTables**

`DataTables` [25] es una librería de JavaScript que permite mejorar las tablas `HTML` de una aplicación, añadiendo funcionalidades como ordenación paginación, búsqueda de forma sencilla.

Se ha utilizado `DataTables` para mostrar de forma interactiva una tabla con los resultados de la detección del periodo seleccionado.

---

## 5. Aspectos relevantes del desarrollo del proyecto

---

Este apartado recoge los aspectos más destacados del desarrollo del proyecto, incluyendo cómo surgió la idea, formación adquirida, ciclo de vida del desarrollo del proyecto y retos planteados.

### Inicio del proyecto

A la hora de buscar tema para realizar el Trabajo de Fin de Grado, tenía interés en desarrollar algo relacionado con gestión de la información o inteligencia artificial, ya que me parecieron asignaturas interesantes. Con esta idea, hable con José Manuel, profesor de la asignatura, para explorar posibles opciones.

José Manuel me propuso un tema enmarcado en las iniciativas de sostenibilidad promovidas por el ranking Greenmetrics [26]. Me resultó interesante ya que combinaba una parte de explotación de datos con otra de visión por computador, por lo que me decidí por este.

### Metodologías

La metodología empleada en el proyecto ha sido SCRUM, aprendida en la asignatura de gestión de proyectos. Esta metodología se basa en intervalos de tiempo cortos, de una o dos semanas, llamados sprints, donde se marcan unos objetivos concretos a realizar.

Se ha empleado Zube para poder gestionar fácilmente las tareas mediante tableros, asignarles *Story points* y ver el progreso.

## *Aspectos relevantes del desarrollo del proyecto*

Al final de cada sprint, se realizaba una reunión donde se presentaban los objetivos alcanzados, posibles problemas encontrados y se planificaba el siguiente sprint.

### **Formación**

Para realizar el proyecto, ha sido necesario investigar en áreas las cuales no se habían tratado durante el grado. Los principales temas fueron:

#### **Visión por computador**

Dado que el objetivo principal del proyecto es detectar y clasificar objetos en un vídeo, ha sido imprescindible aprender el funcionamiento general de modelos basado en redes convolucionales, en concreto YOLO.

Fue necesario consultar documentación y tutoriales sobre redes neuronales y modelos de detección para entender como configurar y utilizar este modelo.

#### **Flask**

Para crear la aplicación web, se eligió Flask por su sencillez para crear aplicaciones web en Python.

Utilicé tutoriales para comprender de forma adecuada como implementar el patrón MVC, el manejo de sesiones de usuario y la integración de plantillas HTML mediante Jinja.

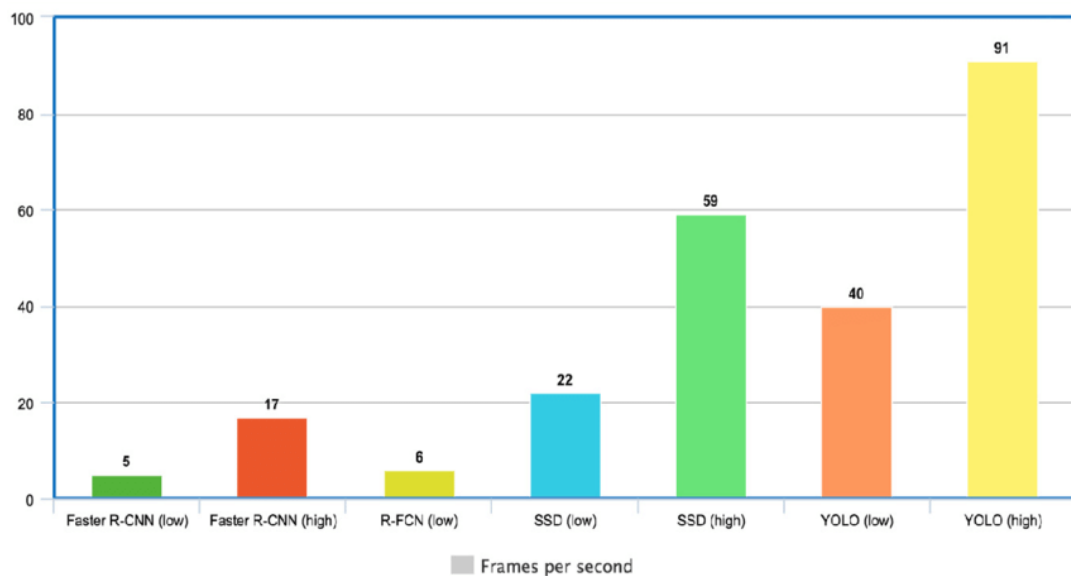
#### **JavaScript**

Para poder mostrar los resultados y métricas en la interfaz, ha sido necesario aprender JavaScript.

Se han empleado bibliotecas como DataTables y Chart.js que han facilitado en gran medida la explotación de los datos, permitiendo crear gráficos y tablas de forma sencilla.

## Desarrollo del proyecto

Durante las primeras semanas del desarrollo del proyecto, se eligió el entorno de desarrollo donde se iba a trabajar y se creó una interfaz simple para realizar las pruebas iniciales. A su vez, se investigó sobre trabajos relacionados y diferentes modelos de detección como R-CNN [27] y SSD [28]. Tras esto, finalmente se decidió implementar el modelo YOLO ya que es el que mejores resultados da en general.



*Figura 4: Comparación FPS entre modelos [29]*

Posteriormente, se desplegó la aplicación en un servidor flask. Se diseñó la nueva interfaz, se implementó YOLO y se realizó la conexión con la base de datos.

```
try:
    conn = get_connection()
    print("✅ Conexión a MySQL establecida.")
    conn.close()
except Exception as e:
    print("❌ Error conectando a MySQL:", e)
```

*Figura 5: Conexión con MySQL*



```
def get_connection():  
    return mysql.connector.connect(  
        host=os.environ['MYSQL_HOST'],  
        user=os.environ['MYSQL_USER'],  
        password=os.environ['MYSQL_PASS'],  
        database=os.environ['MYSQL_DB']  
    )
```

*Figura 6: Función de conexión*

Una vez comprobada la conexión, se creó la tabla donde se iban a almacenar las detecciones registradas por las diferentes versiones de YOLOv11. (Ver Figura 7) Se implementó una función donde, mediante un video de prueba, se realizaba el conteo de vehículos que cruzaban una línea definida en una posición determinada.

Teniendo un prototipo que cargaba un vídeo y realizaba el conteo correctamente, se comenzó con el almacenamiento de los logs para su posterior análisis.

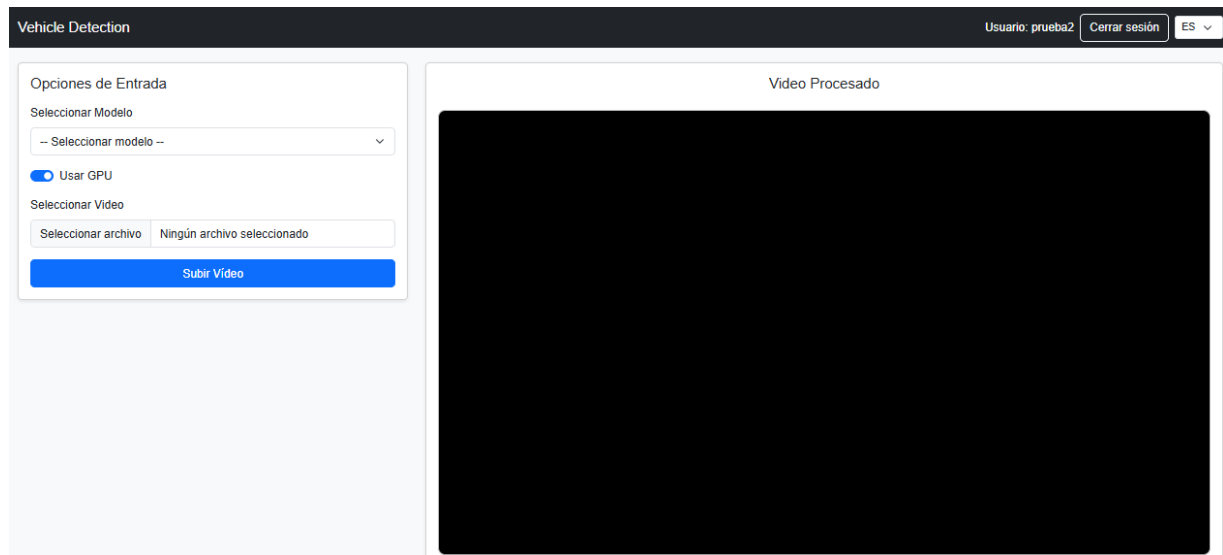
```
def create_vehicle_logs_table():  
    try:  
        conn = get_connection()  
        cursor = conn.cursor()  
        cursor.execute('''  
            CREATE TABLE IF NOT EXISTS vehicle_logs (  
                id INT AUTO_INCREMENT PRIMARY KEY,  
                timestamp DATETIME NOT NULL,  
                vehicle_type VARCHAR(50) NOT NULL,  
                model_used VARCHAR(20) NOT NULL,  
                facultad VARCHAR(255) DEFAULT 'Desconocida',  
                device_used VARCHAR(10) DEFAULT 'CPU',  
                video_filename VARCHAR(255) DEFAULT 'Desconocido'  
            );  
        ''')  
        conn.commit()  
        print("✅ Tabla 'vehicle_logs' verificada/creada.")  
    except Exception as e:  
        print("❌ Error creando la tabla 'vehicle_logs':", e)  
    finally:  
        cursor.close()  
        conn.close()
```

*Figura 7: Tabla de registros*

### *Aspectos relevantes del desarrollo del proyecto*

Tras tener la funcionalidad de detección prácticamente completa, se decidió crear un sistema de gestión de usuarios. Se implementaron dos usuarios:

- Expert: Realiza todo lo referente a la detección, desde la subida y configuración de opciones hasta la visualización completa del procesamiento que realiza YOLO. (Figura 8)
- User: Realiza todo lo relacionado con la gestión de la información y explotación de datos. Puede seleccionar el periodo de fechas deseado, y se le muestran estadísticas y gráficos de gran interés. (Figura 9)



*Figura 8: Ventana usuario experto*

## *Aspectos relevantes del desarrollo del proyecto*

Platoon'." data-bbox="125 190 879 429"/>

*Figura 9: Ventana usuario*

Se intentó automatizar en la medida de lo posible todos los aspectos relacionados con la detección de los vehículos. Para ello, se añadió una funcionalidad que permite al usuario dibujar la línea de cruce donde él desee. Fue necesario adaptar la función de conteo ya que anteriormente, al ser una línea horizontal, no se tenía en cuenta que la línea pudiese ser dibujada diagonalmente.

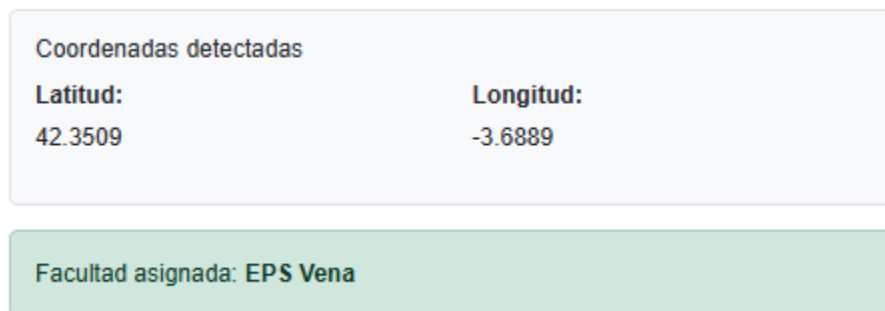


*Figura 10: Línea de cruce personalizada*

### *Aspectos relevantes del desarrollo del proyecto*

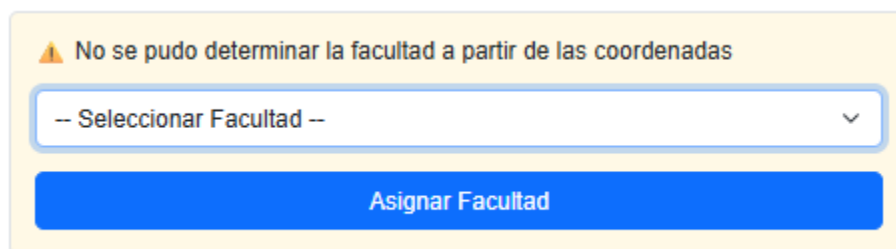
Además, se añadió otra funcionalidad para poder detectar automáticamente desde que facultad se ha grabado el vídeo. Con vídeos de prueba grabados desde diferentes modelos de móvil, se extrajeron los metadatos para ajustar la conversión a coordenadas, ya que difieren entre sistemas operativos. Comprobado que las coordenadas eran las correctas, se procedió a la asignación automática. (Figura 11)

Se tuvo en cuenta que, por diversos motivos, el vídeo subido puede no tener metadatos o estar en otro formato, por lo que, en estos casos, se muestra un formulario para que se seleccione manualmente la facultad. (Figura 12)



The screenshot shows two distinct UI elements. The top element is a light gray rectangular box containing the text 'Coordenadas detectadas' in a bold, dark font. Below this, there are two columns of data: 'Latitud:' followed by the value '42.3509' on the left, and 'Longitud:' followed by the value '-3.6889' on the right. The bottom element is a green rectangular box with the text 'Facultad asignada: EPS Vena' in a bold, dark font.

*Figura 11: Facultad detectada automáticamente*



The screenshot shows a yellow rectangular box with a warning icon (a triangle with an exclamation mark) and the text 'No se pudo determinar la facultad a partir de las coordenadas'. Below this message is a white dropdown menu with a blue border and the text '-- Seleccionar Facultad --'. At the bottom of the yellow box is a blue rectangular button with the text 'Asignar Facultad' in white.

*Figura 12: Facultad no detectada automáticamente*

### Aspectos relevantes del desarrollo del proyecto

Dando por finalizada la implementación completa del usuario experto, se continuó con el usuario estándar. Mediante librerías de JavaScript y consultas SQL, se añadieron gráficos, estadísticas y tablas que proporcionan al usuario un análisis detallado de la movilidad en las diferentes facultades de la Universidad de Burgos. También se pueden exportar los resultados a un archivo CSV para la explotación personalizada por parte del usuario.

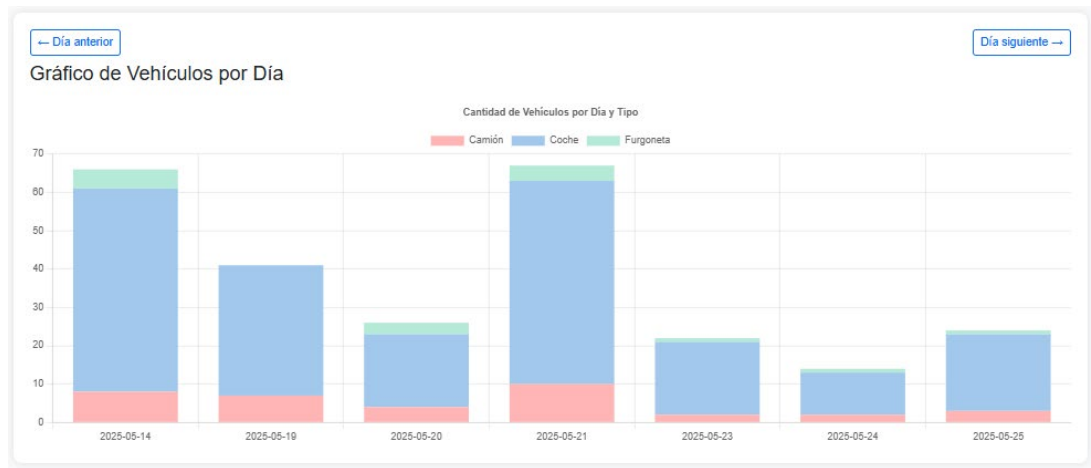


Figura 13: Gráfico de resultados

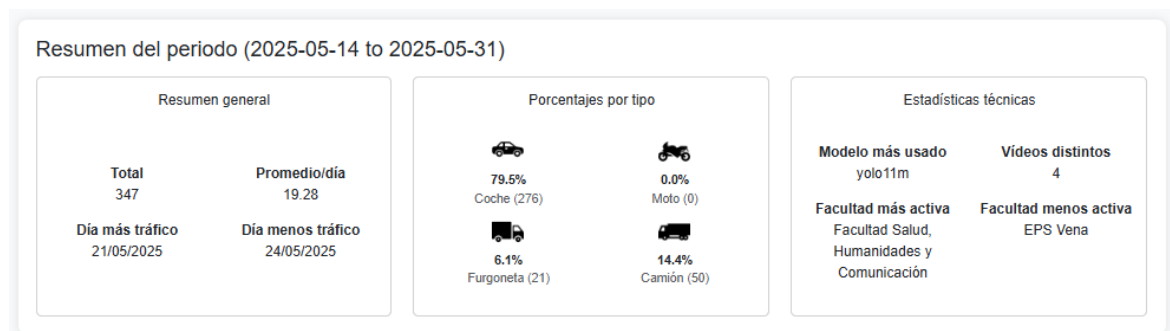


Figura 14: Estadísticas generales

### Aspectos relevantes del desarrollo del proyecto

Resumen por Facultad

Buscar:

Facultad	Total	🚗 Coche	🏍 Moto	🚚 Furgoneta	🚛 Camión	Videos
Facultad Salud, Humanidades y Comunicación	122	95 (77.9%)	0 (0.0%)	9 (7.4%)	18 (14.8%)	2
Facultad Económicas	98	80 (81.6%)	0 (0.0%)	3 (3.1%)	15 (15.3%)	1
Facultad Derecho	39	28 (71.8%)	0 (0.0%)	4 (10.3%)	7 (17.9%)	1
Facultad Educación	27	19 (70.4%)	0 (0.0%)	3 (11.1%)	5 (18.5%)	1
EPS Milanera	25	22 (88.0%)	0 (0.0%)	1 (4.0%)	2 (8.0%)	2
Facultad Ciencias	20	16 (80.0%)	0 (0.0%)	1 (5.0%)	3 (15.0%)	2
EPS Vena	16	16 (100.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	3

Mostrando 1 a 7 de 7 registros

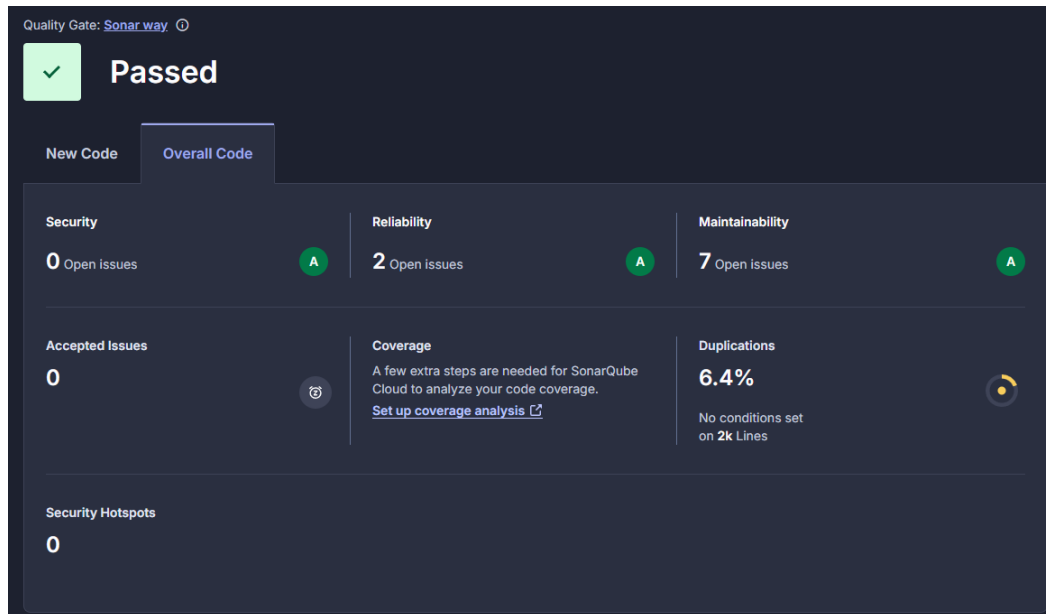
Exportar CSV

Figura 15: Tabla de resultados

Para finalizar, se implementó la internacionalización de la app mediante dos diccionarios con las traducciones, uno en español y otro en inglés.

Se intentó desplegar la aplicación en plataformas web de hosting gratuito, pero al estar diseñadas para aplicaciones ligeras no permitían instalar las librerías necesarias para ejecutar la aplicación. Además, presentan limitaciones en cuanto a recursos de memoria y tiempo de ejecución. Por ello, se optó por crear una máquina virtual donde se puede garantizar un entorno de ejecución controlado.

Terminado el proyecto, se realizó el análisis del código con la herramienta SonarCloud.[30] Una vez analizado el repositorio, se realizaron las mejoras de código correspondientes. Fue necesario aumentar la seguridad en cuanto al acceso de datos ya que anteriormente, al ser un prototipo, las claves de prueba estaban en el código. También se realizaron pequeñas mejoras en cuanto a duplicación y mantenimiento del código



*Figura 16: Resultados del análisis del código*

El análisis completo se puede encontrar en

[https://sonarcloud.io/project/overview?id=pabloosp\\_VehicleDetection](https://sonarcloud.io/project/overview?id=pabloosp_VehicleDetection)

## Resolución de problemas técnicos

A lo largo del desarrollo, se han presentado diferentes problemas que han requerido ser solventados antes de avanzar.

### Conexión segura con la base de datos

Inicialmente, las credenciales de la base de datos estaban almacenadas en texto dentro del código, lo que suponía un riesgo en la seguridad del proyecto.

Para ello, se extrajeron dichas credenciales a un archivo .env y que, al lanzar la app, estas credenciales se carguen automáticamente. Las adaptaciones necesarias para este cambio se detallan en los anexos.

## *Aspectos relevantes del desarrollo del proyecto*

### **Detección automática de la facultad**

Algunos vídeos no incluían metadatos o el formato en el que se presentaban no estaba contemplado. Por ello, se realizaron pruebas con diferentes vídeos para adaptar el procesamiento de metadatos según el dispositivo de grabación y proporcionar un selector manual cuando estos datos no se detecten.

Sin embargo, podría ser necesario añadir algún caso más en caso de no estar contemplado el formato de algún dispositivo.

### **Conteo personalizado de vehículos**

En un principio, la función de conteo de vehículos solo tenía en cuenta líneas horizontales fijas. Al implementar la mejora que permite al usuario dibujar la línea de forma libre, fue necesario adaptar la lógica de la función.

Se necesito reescalar la línea al pasar del frame donde se dibuja al recuadro de procesamiento, ya que si no la línea no se mostraría donde indicó el usuario. A su vez, se modificó la lógica de cruce para que solamente se registre una vez cada vehículo adaptando los puntos de intersección entre la línea y el vehículo.



---

## 6. Trabajos relacionados

---

En este apartado se detallan un par de herramientas que hacen uso de la visión por computador y análisis de imágenes y vídeos.

### 6.1 Plate Recognizer

Plate Recognizer [31] es un software basado en visión por computador que permite el reconocimiento automático de matrículas (ALPR). Está diseñado para poder trabajar con imágenes borrosas, condiciones climatológicas adversas, imágenes de baja resolución, etc. Puede integrarse fácilmente mediante API REST, cuenta con un panel de usuario para ver los resultados y configurar alertas, además de ofrecer kits de desarrollo software (SDK).

Ofrecen diferentes módulos específicos dependiendo del tipo de aplicación, como reconocimiento en imágenes fijas (Snapshot ALPR), vídeos en tiempo real (Stream ALPR), detección de barcos (Boat ID) ... También cuentan con módulos de gestión de aparcamientos y seguimiento de personas. [31]

El precio de este software varía dependiendo del módulo a contratar, pero todos cuentan con un plan gratuito que permite un número muy limitado de detecciones u ofrecen una prueba mensual, y planes de pago que comienzan a partir de los 20\$. También cuentan con planes empresariales personalizados. [32]







 <b>Try It Out</b> Our algo handles plates that are blurry, dark, angled, and <b>much more!</b> See our <a href="#">ALPR in India or USA</a> .	 <b>Images &amp; Videos</b> We handle both. <b>Snapshot</b> decodes plates from images. <b>Stream</b> processes live camera or video files.	 <b>Feature-Rich</b> Get vehicle <b>make, model, color, region</b> , direction of travel, Webhooks, <b>Dashboard</b> , and more! <b>Blur plates</b> too!
 <b>On-Premise &amp; Cloud</b> We provide both cloud and on-premise software (no Internet required) on a variety of hardware.	 <b>Fast, Nimble</b> Snapshot's <b>inference speed</b> is 50-100 ms and Stream processes 5-10 cameras on a mid-range CPU. No GPU needed!	 <b>Proven Globally</b> Our ALPR engine supports over <b>90 countries</b> and is tuned to your specific region.

Figura 17: Plate Recognizer resumen [31]

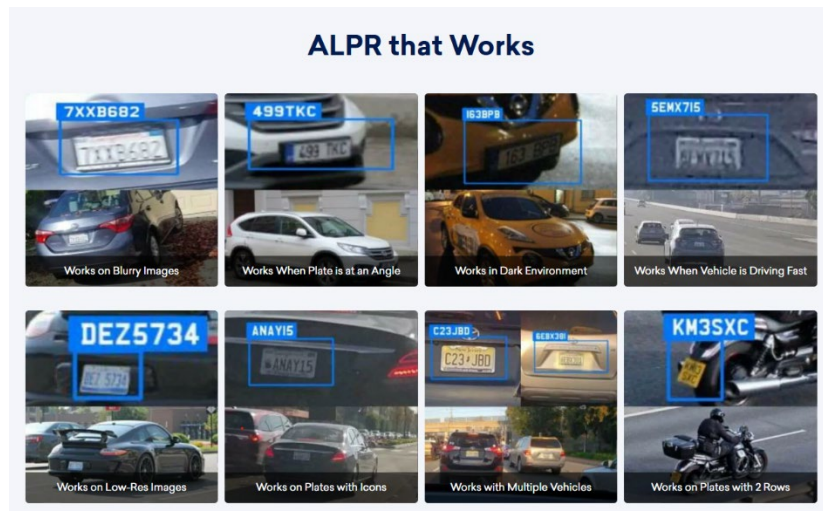


Figura 18: ALPR that Works [31]

## 6.2 Amazon Rekognition

Amazon Rekognition [33] es un servicio de análisis creado por AWS (AWS, Amazon Web Services) basado en la nube que permite analizar imágenes y vídeos gracias a la visión artificial. Se puede integrar en aplicaciones, móviles y dispositivos a través de la API.

Permite detectar objetos y escenas, verificar la edad de un rostro para evitar fraudes, detectar y filtrar contenido explícito o violento tanto en imágenes como vídeos, reconocer famosos, detectar medidas de seguridad en diferentes sectores, etc.

Este software se contrata dependiendo de la modalidad que queramos usar, contando con una suscripción gratuita de 12 meses. La modalidad de pago ofrece un coste fijo por imagen. [34]

## Trabajos relacionados

The screenshot shows the Amazon Rekognition console for the 'Facial analysis' service. The left sidebar contains navigation options: Custom Labels, Demos (with 'Face comparison' highlighted), Video Demos, Metrics, and Additional Resources. The main content area displays a sample image of a family with bounding boxes around their faces. Below the image are options to 'Choose a sample image' or 'Use your own image'. The 'Results' section on the right provides a detailed analysis of the selected face.

Attribute	Confidence Score
looks like a face	99.9 %
appears to be male	99.9 %
age range	56 - 64 years old
smiling	96.2 %
appears to be happy	99.9 %
not wearing glasses	97.6 %

Figura 19: Amazon Rekognition facial analysis[35]

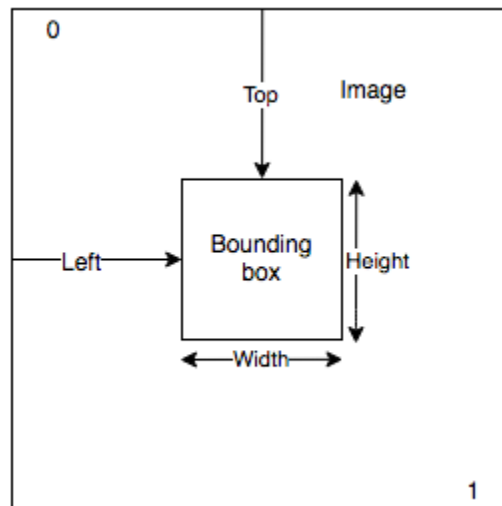


Figura 20: Amazon Rekognition bounding box [36]

---

## 7. Conclusiones y Líneas de trabajo futuras

---

En este apartado se detallan las conclusiones obtenidas tras la realización del proyecto y las posibles líneas

### 7.1 Conclusiones

Una vez finalizado el proyecto, puedo afirmar que el resultado cumple con los objetivos establecidos inicialmente. Además, se han incorporado funcionalidades no previstas en un primer momento como la detección automática de facultad, pero que dan un valor añadido al proyecto.

A nivel personal, este proyecto me ha permitido aprender y adquirir habilidades en diferentes tecnologías y herramientas en las que anteriormente no poseía conocimientos o manejaba de manera superficial.

He comprendido en gran medida el funcionamiento de los modelos basados en visión por computador y también Flask, los protocolos que emplea y como se realizan las peticiones para compartir los datos entre el servidor y el código. También he aprendido acerca de JavaScript, que era un lenguaje que desconocía, y he ampliado y mejorado mis conocimientos en Python y CSS.

Una de las mayores dificultades considero que ha sido entender completamente el funcionamiento de YOLO, ya que al comienzo del proyecto nunca había indagado en cómo se realizaba la detección de objetos. Otra ha sido implementar la lógica de conteo de vehículos, ya que tuve que realizar diferentes ajustes en los cálculos para asegurar que fuera preciso.

En cuanto a la metodología seguida, he utilizado Scrum. Ha sido imprescindible a la hora de tener una buena organización, ya que la división del proyecto en sprints y tareas me ha permitido planificar con antelación lo que iba a realizar, y así estimar lo que dedicaría a cada tarea.

En términos generales, considero que el balance del proyecto ha sido bueno dado que he adquirido conocimientos en diferentes áreas además de mejorar mis habilidades en gestión de proyectos.

## **7.2 Líneas de trabajo futuras**


- Mejorar seguridad de la aplicación.
- Implementar autenticación de doble factor.
- Implementar la recuperación de contraseña.
- Implementar la detección de matrículas para poder llevar a cabo un registro de entradas y salidas.
- Desarrollar soporte para streaming de vídeo en directo.
- Almacenar frame o fragmento de vídeo de cada detección.
- Mostrar un contador fuera del recuadro de procesamiento empleando websockets.

---

# Bibliografía

---

- [1] «GreenMetric World University Ranking | Unidad de Campus y Medioambiente». Accedido: 27 de abril de 2025. [En línea]. Disponible en: <https://www.ucm.es/sostenibilidad/green-metric-university-ranking>
- [2] L. Á. León, «Matemáticas y Visión por Ordenador», 2003, [En línea]. Disponible en: [https://www.researchgate.net/profile/Luis-Alvarez-37/publication/40670113\\_Matematicas\\_y\\_vision\\_por\\_elordenador/links/0c960517ae21d189c8000000/Matematicas-y-vision-por-elordenador.pdf](https://www.researchgate.net/profile/Luis-Alvarez-37/publication/40670113_Matematicas_y_vision_por_elordenador/links/0c960517ae21d189c8000000/Matematicas-y-vision-por-elordenador.pdf)
- [3] A. Aggarwal, «YOLO Explained», Analytics Vidhya. Accedido: 20 de abril de 2025. [En línea]. Disponible en: <https://medium.com/analytics-vidhya/yolo-explained-5b6f4564f31>
- [4] R. Szeliski, *Computer Vision: Algorithms and Applications*. en Texts in Computer Science. Cham: Springer International Publishing, 2022. doi: 10.1007/978-3-030-34372-9.
- [5] I. Goodfellow, Y. Bengio, y A. Courville, «Deep Learning CNN». Accedido: 20 de abril de 2025. [En línea]. Disponible en: <https://www.deeplearningbook.org/contents/convnets.html>
- [6] «Introduction to Deep Learning: What Are Convolutional Neural Networks?», Mathworks. Accedido: 20 de abril de 2025. [En línea]. Disponible en: <https://es.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html>
- [7] I. Goodfellow, Y. Bengio, y A. Courville, «Deep Learning Softmax». Accedido: 20 de abril de 2025. [En línea]. Disponible en: <https://www.deeplearningbook.org/contents/mlp.html>
- [8] J. Redmon, S. Divvala, R. Girshick, y A. Farhadi, «You Only Look Once: Unified, Real-Time Object Detection», en *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jun. 2016, pp. 779-788. doi: 10.1109/CVPR.2016.91.
- [9] «Python», *Wikipedia, la enciclopedia libre*. 19 de abril de 2025. Accedido: 15 de mayo de 2025. [En línea]. Disponible en: <https://es.wikipedia.org/w/index.php?title=Python&oldid=166937646>
- [10] «¿Qué es Visual Studio Code y cuáles son sus ventajas? | Blog de Arsys», Arsys. Accedido: 15 de mayo de 2025. [En línea]. Disponible en: <https://www.arsys.es/blog/que-es-visual-studio-code-y-cuales-son-sus-ventajas>
- [11] «Qué es Flask y ventajas que ofrece | OpenWebinars», OpenWebinars.net. Accedido: 15 de mayo de 2025. [En línea]. Disponible en: <https://openwebinars.net/blog/que-es-flask/>
- [12] «¿Qué es MySQL? Explicación y características | Blog de Arsys», Arsys. Accedido: 17 de mayo de 2025. [En línea]. Disponible en: <https://www.arsys.es/blog/mysql>
- [13] *mysql-connector-python: A self-contained Python driver for communicating with MySQL servers, using an API that is compliant with the Python Database API Specification v2.0 (PEP 249)*. Python. Accedido: 17 de mayo de 2025. [MacOS :: MacOS X, Microsoft :: Windows, POSIX :: Linux, Unix]. Disponible en: <https://dev.mysql.com/doc/connector-python/en/>
- [14] «Introducción — Conociendo GitHub 0.1 documentation». Accedido: 17 de mayo de

2025. [En línea]. Disponible en:  
<https://conociendogithub.readthedocs.io/en/latest/data/introduccion/>
- [15] «Zube, un sistema de gestión de proyectos para programadores». Accedido: 17 de mayo de 2025. [En línea]. Disponible en: <https://www.whatsnew.com/2015/10/04/zube-un-sistema-de-gestion-de-proyectos-para-programadores/>
- [16] «Zotero», *Wikipedia, la enciclopedia libre*. 16 de marzo de 2025. Accedido: 17 de mayo de 2025. [En línea]. Disponible en:  
<https://es.wikipedia.org/w/index.php?title=Zotero&oldid=166145269>
- [17] *ultralytics: Ultralytics YOLO  for SOTA object detection, multi-object tracking, instance segmentation, pose estimation and image classification*. Python. Accedido: 18 de mayo de 2025. [MacOS, Microsoft :: Windows, POSIX :: Linux]. Disponible en:  
<https://ultralytics.com>
- [18] «PyTorch», PyTorch. Accedido: 18 de mayo de 2025. [En línea]. Disponible en:  
<https://pytorch.org/>
- [19] «OpenCV», *Wikipedia, la enciclopedia libre*. 6 de mayo de 2025. Accedido: 18 de mayo de 2025. [En línea]. Disponible en:  
<https://es.wikipedia.org/w/index.php?title=OpenCV&oldid=167234905>
- [20] *pymediainfo: A Python wrapper for the MediaInfo library*. Python. Accedido: 18 de mayo de 2025. [MacOS :: MacOS X, Microsoft :: Windows, POSIX :: Linux]. Disponible en: <https://github.com/sbraz/pymediainfo>
- [21] *shapely: Manipulation and analysis of geometric objects*. Python.
- [22] «GeoJSON», *Wikipedia, la enciclopedia libre*. 27 de octubre de 2024. Accedido: 18 de mayo de 2025. [En línea]. Disponible en:  
<https://es.wikipedia.org/w/index.php?title=GeoJSON&oldid=163246303>
- [23] Mapbox, «geojson.io | powered by Mapbox», [geojson.io](https://geojson.io). Accedido: 18 de mayo de 2025. [En línea]. Disponible en: <https://geojson.io>
- [24] «Chart.js | Chart.js». Accedido: 18 de mayo de 2025. [En línea]. Disponible en:  
<https://www.chartjs.org/docs/latest/>
- [25] «DataTables | Javascript table library». Accedido: 20 de mayo de 2025. [En línea]. Disponible en: <https://datatables.net/>
- [26] «Overall Rankings 2024 - UI GreenMetric». Accedido: 4 de junio de 2025. [En línea]. Disponible en: <https://greenmetric.ui.ac.id/rankings/overall-rankings-2024>
- [27] «R-CNN - Region-Based Convolutional Neural Networks», GeeksforGeeks. Accedido: 7 de junio de 2025. [En línea]. Disponible en: <https://www.geeksforgeeks.org/r-cnn-region-based-cnns/>
- [28] «IBM Maximo Visual Inspection». Accedido: 7 de junio de 2025. [En línea]. Disponible en: <https://www.ibm.com/docs/es/visual-insights?topic=model-ssd>
- [29] H. Romero y A. Morales Acosta, «A review: Comparison of performance metrics of pretrained models for object detection using the TensorFlow framework», *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 844, p. 012024, jun. 2020, doi: 10.1088/1757-899X/844/1/012024.
- [30] «Login - SonarQube Cloud». Accedido: 31 de mayo de 2025. [En línea]. Disponible en:  
<https://sonarcloud.io/login>
- [31] «Automatic License Plate Recognition - High Accuracy ALPR», Plate Recognizer. Accedido: 14 de mayo de 2025. [En línea]. Disponible en: <https://platerrecognizer.com/>
- [32] «Pricing | Plate Recognizer ALPR», Plate Recognizer. Accedido: 14 de mayo de 2025.

- [En línea]. Disponible en: <https://platerecognizer.com/pricing/>
- [33] «¿Qué es Amazon Rekognition? - Amazon Rekognition». Accedido: 14 de mayo de 2025. [En línea]. Disponible en: [https://docs.aws.amazon.com/es\\_es/rekognition/latest/dg/what-is.html](https://docs.aws.amazon.com/es_es/rekognition/latest/dg/what-is.html)
- [34] «Amazon Rekognition – Precios – AWS», Amazon Web Services, Inc. Accedido: 14 de mayo de 2025. [En línea]. Disponible en: <https://aws.amazon.com/es/rekognition/pricing/>
- [35] «Detectar, analizar y comparar rostros con Amazon Rekognition», Amazon Web Services, Inc. Accedido: 14 de mayo de 2025. [En línea]. Disponible en: <https://aws.amazon.com/es/getting-started/hands-on/detect-analyze-compare-faces-rekognition/>
- [36] «Visualización de cuadros delimitadores - Amazon Rekognition». Accedido: 14 de mayo de 2025. [En línea]. Disponible en: [https://docs.aws.amazon.com/es\\_es/rekognition/latest/dg/images-displaying-bounding-boxes.html](https://docs.aws.amazon.com/es_es/rekognition/latest/dg/images-displaying-bounding-boxes.html)