Universidad de Alcalá
Escuela Politécnica Superior

GRADO DE SISTEMAS DE LA INFORMACIÓN

Trabajo Fin de Grado

Procesamiento de algoritmos de visión artificial en la nube - Microsoft Azure -

Autor: Álvaro Fernández-Taviro Gracia

Tutor: Alfredo Gardel Vicente

2021

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

GRADO DE SISTEMAS DE LA INFORMACIÓN

Trabajo Fin de Grado

Procesamiento de algoritmos de visión artificial en la nube – Microsoft Azure -

Autor: Álvaro Fernández-Taviro Gracia

Tutor: Alfredo Gardel Vicente

TRIBUNAL:

Presidente: Pedro Martín Sánchez

Primer vocal: Julio Pastor Mendoza

Segundo vocal: Alfredo Gardel Vicente

FECHA: 26 Marzo 2021

TABLA DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

GLOSARIO DE ACRÓNIMOS Y ABREVIATURAS

AIA	Asociación de Imágenes Automatizadas
AIaaS	Artificial Intelligence as a Service (Inteligencia artificial como servicio)
API	Application programming interface (Interfaz de programación de aplicaciones)
AWS	Amazon Web Services
CAPEX	Capital expenditure (gastos en capital)
CCD	Charge-coupled device (Dispositivo de carga acoplada)
CPU	Unidad central de procesamiento
HaaS	Hardware as a Service (Hardware como servicio)
HW	Hardware
IA	Inteligencia Artificial
IaaS	Infrastructure as a Service (Infraestructura como servicio)
IDPS	Sistemas de detección de intrusiones y prevención
OCR	Reconocimiento óptico de caracteres
OPEX	Operational expenditures (gastos de funcionamiento)
PaaS	Platform as a Service (Plataforma como servicio)
RGPD	Reglamento General de Protección de Datos
ROI	Región de interés de una imagen
SaaS	Software as a Service (Software como servicio)
SDK	Software Development Kit (Kit de desarrollo de software)
SW	Software
TI	Tecnologías de la Información
TIC	Tecnologías de la Información y la Comunicación
XaaS	X as a Service (X como servicio, se refiere a “todo” como servicio)

Procesamiento de algoritmos de Visión artificial en la nube – Microsoft Azure–

Resumen

El uso de la Inteligencia Artificial está cada día más extendido en diversos campos, entre los que se incluye la visión artificial.

Este trabajo muestra que es posible programar empleando las capacidades de visión artificial de Computer Vision, uno de los servicios de Azure, la nube pública de Microsoft, sin tener una experiencia previa en Inteligencia Artificial.

Esas capacidades permiten hacer diferentes análisis de imágenes, accesibles mediante el SDK del correspondiente lenguaje de programación o mediante llamadas directas a la API del servicio, siendo esta última opción la que se ha elegido para desarrollar el ejemplo práctico de desarrollo en PYTHON.

Palabras clave

Nube, Inteligencia Artificial, visión artificial , Azure, Programación.

Processing Computer Vision algorithms in the cloud – Microsoft Azure –

Abstract

The use of Artificial Intelligence is increasingly widespread in multiple fields, including computer vision.

This work shows how simple it is to program using the machine vision capabilities of Computer Vision, one of the multiple services provided by Azure, the Microsoft's public cloud, without experience in Artificial Intelligence.

These capabilities allow you to make different image analysis, accessible through the SDK of the corresponding programming language or through direct calls to the API service. The latter option has been chosen to develop a PYTHON program as an example of Computer Vision capabilities.

Keywords

Cloud, Artificial Intelligence, Computer Vision, Azure, Programming.

Resumen extendido

La visión artificial es una de las ramas de la Inteligencia Artificial. Con ella se puede obtener información de imágenes de forma similar a cómo lo haría una persona, yendo mucho más allá de la simple representación digital de los objetos que componen esas imágenes.

Mediante la visión artificial se analizan las imágenes y se obtiene información sobre ellas, consiguiendo catalogar los objetos que se encuentran en la imagen, así como detectar marcas, caras conocidas e incluso proporcionar una frase que la describa o reconocer texto dentro ella, entre otras capacidades.

Tradicionalmente los algoritmos que proporcionaban las capacidades propias de las funciones de cada lenguaje de programación estaban asociados “de forma estática” a los compiladores que se instalaban localmente en los ordenadores (o servidores) con los que se desarrollaban los programas, los cuales debían ser actualizados periódicamente para poder usar las mejoras que proporcionaban las sucesivas versiones del lenguaje y de sus compiladores.

Mediante la computación en la nube se simplifica enormemente todo lo relacionado con el mantenimiento del software, a la vez que se reducen los costes y, en principio, se consigue una mayor seguridad. En el ámbito de la Inteligencia Artificial, a esas ventajas hay que añadir la posibilidad de aprovechar el *aprendizaje* conseguido por los sistemas, del que se puede sacar provecho sin necesidad de participar en él.

Al hablar de computación en la nube nos encontramos con tres posibles contextos:

- *Nube pública* (disponible para cualquiera que pueda que pueda acceder a Internet).
- *Nube privada* (disponible para un grupo reducido de máquinas).
- *Nube híbrida* (una solución que combina servicios de las dos anteriores).

A su vez, cada uno de los tipos de nube puede ofrecer una variedad de servicios que se agrupan por diversos criterios, siendo los siguientes los tipos más frecuentes:

- *Infraestructura como Servicio* (IaaS).
- *Plataforma como Servicio* (PaaS).
- *Software como Servicio* (SaaS).

Existe una multitud de proveedores de computación en la nube, cada uno de los cuales puede, a su vez, ofrecer múltiples servicios, dando lugar a un entramado de proveedores y servicios en el que existe una gran competencia y que permite conseguir *soluciones cloud computing* para cualquier necesidad que se pueda plantear. En esa variedad de servicios, naturalmente, se pueden encontrar servicios de Inteligencia Artificial y, dentro de ellos, de visión artificial .

Entre los proveedores de servicios de computación en la nube se encuentran las grandes empresas tecnológicas, que ofrecen en su catálogo de productos los servicios disponibles en “su nube”. Así, nos encontramos, por ejemplo, con las nubes de Microsoft (Azure), de Google (Google Cloud), de Amazon (Amazon Web Services), de Alibaba (Alibaba Cloud) o de IBM (IBM Cloud).

Dado que el objeto del presente trabajo es mostrar el uso de los algoritmos de visión artificial disponibles en [una] nube, se ha optado por elegir la nube del proveedor que se considera más extendido en nuestro ámbito académico que, además, es el que más

facturó en 2020 [1], Microsoft Azure, que por sí solo facturó más que sus dos competidores directos juntos, Google Cloud y AWS.

Centrándonos por tanto en los servicios ofrecidos por Microsoft en su nube, Azure, nos encontramos con que estos incluyen tanto IaaS como PaaS y SaaS.

En el marco de los servicios PaaS de Azure se encuentran disponibles determinadas capacidades de Inteligencia Artificial, entre las cuales se encuentra el servicio que sirve para atender al objetivo de este trabajo: *Computer Vision*.

Computer Vision proporciona acceso a algoritmos que permiten procesar imágenes para obtener información basada en sus contenidos, los cuales están disponibles en una API REST que puede ser invocada por desarrollos propios de dos formas diferentes:

- A través del Software Development Kit (SDK) de una biblioteca cliente, disponible para los lenguajes de programación más comunes.
- A través de llamadas directas a la API.

La versión más reciente de la API permite:

- Describir la imagen.
- Analizar la imagen.
- Detectar objetos en la imagen.
- Obtener el área de interés de la imagen.
- Reconocimiento óptico de caracteres (OCR) en la imagen.
- Generar una miniatura de la imagen.
- Detectar marcas en la imagen.
- Detectar caras en la imagen.
- Clasificar una imagen.
- Detectar el tipo de imagen.
- Detectar contenido específico.
- Detectar la combinación de colores.
- Moderación del contenido de las imágenes.

Para alcanzar el objetivo de este trabajo se ha realizado un desarrollo en PYTHON que accede a las capacidades de *Computer Vision* con llamadas directas a su API, para hacer uso de las funcionalidades que permiten describir una imagen y detectar objetos en ella. Además, para tener una visión completa de las capacidades de *Computer Vision*, también se muestra la forma de proceder para acceder a la API a través del SDK para PYTHON, mostrando diferentes ejemplos en los que se puede ver el tipo de salidas que proporcionan todas y cada una de sus funcionalidades.

Procesamiento de algoritmos de *Visión artificial* en la nube – Microsoft Azure –

1 Campo de aplicación y objetivos

El campo de aplicación de este trabajo es la *Inteligencia Artificial* (IA) centrándose en el ámbito específico de la *visión artificial* .

El objetivo principal de este trabajo es realizar un "tutorial" y desarrollar una aplicación que muestre de forma práctica el funcionamiento de las herramientas de Inteligencia Artificial que ofrece Microsoft Azure *para el* análisis de imágenes*, aplicándolas en la detección de un tipo de objeto determinado, partiendo del estudio previo de las características de la visión artificial y del "cloud computing".*

De ese objetivo principal del trabajo se derivan los siguientes objetivos secundarios:

- Estudio de las características generales de la visión artificial .
- Estudio de los diferentes tipos de nube y de servicios de computación en la nube.
- Estudio de los servicios que ofrece Microsoft Azure.
- Estudio de la implantación y uso del servicio Computer Vision de Microsoft Azure.

Para presentar el trabajo realizado, esta memoria incluye diferentes secciones. En la sección 2 se hace un breve recorrido por la historia de la Inteligencia Artificial desde sus inicios hasta la actualidad. En la sección 3 se tratan las generalidades de la visión artificial y se hace una comparación de ésta con la visión humana. En la sección 4 se analiza la otra pata en la que se basa el servicio *Computer Vision* de *Microsoft Azure*: la computación en la nube. La sección 5 es la última del "bloque de teoría", en la que se tratan las características de *Microsoft Azure* y de sus servicios, especialmente el de *Computer Vision*. Las secciones 6, 7 y 8 se dedican al "bloque de la práctica" del trabajo, mostrando en la primera de ellas el entorno empleado para la realización de esa práctica,

en la segunda las particularidades del programa desarrollado en PYTHON como práctica, para mostrar las llamadas a la API de Computer Vision y la forma de tratar los datos devueltos por ésta, y en la tercera, la sección 8, se presentan los resultados obtenidos al ejecutar el citado programa. Por último, la sección 9 recoge las conclusiones personales obtenidas durante la elaboración del Trabajo.

La memoria finaliza con tres anexos:

- En el anexo 1 se muestran los resultados obtenidos del análisis de una serie de fotografías, en concreto las descripciones y relaciones de objetos que *Computer Vision* devuelve sobre ellas.
- En el anexo 2 se incluye en código PYTHON íntegro del programa desarrollado para la práctica.
- En el anexo 3 se presenta una serie de ejemplos en los que se muestran tanto las llamadas a todas las funcionalidades ofrecidas por Computer Vision como los resultados devueltos por Computer Vision a esas llamadas.

Al contenido expuesto hay que añadir los preceptivos índices, resúmenes y relación de bibliografía.

2 Introducción

Resulta difícil encontrar consenso a la hora de definir qué es la Inteligencia Artificial, máxime cuando ni siquiera existe un acuerdo sobre cómo definir la inteligencia, por lo que, obviando esa discusión, nos limitaremos a admitir que la IA es la inteligencia llevada a cabo por máquinas.

La idea de poder disponer de máquinas que, de alguna forma, pudieran imitar el razonamiento humano y alcanzar conclusiones de forma razonada es muy antigua, pudiendo remontarnos al filósofo griego *Aristóteles* (384-322 a. C.) y a sus pensamientos sobre la filosofía de las matemáticas [2]. Aristóteles sostenía que las matemáticas estudian propiedades como la simetría, la continuidad y el orden que pueden realizarse literalmente en el mundo físico, es decir, para él, los objetos matemáticos no existen en sí mismos, sino que son abstracciones de objetos que existen en el mundo físico, de donde cabe deducir que *los objetos del mundo físico pueden representarse mediante modelos matemático*s.

Mucho más recientemente nos encontramos con *Ramón Llull* (1232-1316), filósofo y teólogo laico, natural de la ciudad de Mallorca (actual Palma de Mallorca), que publicó un trabajo que, con nuestra percepción actual, podemos considerar próximo a la Inteligencia Artificial: *Ars Magna* [2]. En esta obra, publicada en el año 1315, Ramon Llull aseguraba que *es posible realizar razonamiento mediante un ingenio mecánico*. Mediante ese ingenio, Llull pretendía mostrar las verdades de la fe cristiana de forma indiscutible, es decir, pretendía construir una máquina, a la que llamó Ars Generalis Ultima o Ars Magna, que demostraría que el razonamiento cristiano era correcto y que los infieles erraban en el suyo, sirviéndole de apoyo en uno de los propósitos principales de su actividad literaria, que fue el de señalar los errores de los racionalistas (como Averroes).

Son varios los autores que de una u otra forma se han referido posteriormente a cuestiones que en nuestro días relacionamos con el concepto de la Inteligencia Artificial, como el propio Descartes, el filósofo francés por excelencia del siglo XVII, que pretendía desarrollar un método que permitiera alcanzar la verdad y hacer de la filosofía un conocimiento tan exacto y fiable como podía serlo la geometría. En el ámbito al que nos estamos refiriendo, *Descartes negaba la posibilidad de que una máquina llegase a ser inteligente*, debido a que los mecanismos son predecibles, inflexibles y limitados. Según él, en base a su teoría metafísica del dualismo de sustancias, jamás podrían recrearse mediante disposiciones de la *sustancia material* dos cualidades que son exclusivas de la *sustancia pensante*: la capacidad para el lenguaje natural y la flexibilidad para habérselas con problemas de todo tipo. Parece obvio que, al menos en esto, se equivocó.

No obstante, el término "*Inteligencia Artificial*" fue acuñado como tal por John McCarthy, creador del lenguaje de programación Lisp, junto a Marvin Minsky y Claude Shannon en 1956 durante la Conferencia de Darmouth [2].

En la década de 1960, el Departamento de Defensa de los Estados Unidos mostró interés en esta iniciativa y puso en marcha proyectos destinados a "entrenar" a ordenadores para que imitasen el razonamiento humano[1] [3].

[1] Como ejemplo, la Defense Advanced Research Projects Agency (DARPA, Agencia de Proyectos de Investigación Avanzada de Defensa) realizó proyectos de planimetría de calles en la década de 1970. También DARPA, produjo asistentes personales inteligentes en 2003, mucho tiempo antes que Siri, Alexa o Cortana.

Estos trabajos iniciaron un camino que obviamente aún no ha llegado a su meta, ni mucho menos, pero que ya ha propiciado multitud de capacidades que, consciente o inconscientemente, conviven con nosotros en la actualidad, entre las que se encuentra la "*visión artificial* " ("*computer vision",* en su acepción inglesa), cuyo objetivo es lograr que un sistema informático "entienda" una escena o "analice" las características de una imagen mediante acciones tales como la detección, segmentación, localización y reconocimiento de los objetos presentados en ella.

El Machine Learning y el Deep Learning están haciendo posible que la visión artificial opere como si se tratara de del sentido humano de la vista, por su capacidad para analizar, evaluar y tener en cuenta todas sus variables; pero con una mayor capacidad de respuesta por la posibilidad de procesar datos masivos -Big Data- y analizarlos en tiempo récord, gracias a que estas tecnologías dotan de mayor capacidad de procesamiento e inteligencia a los sistemas automatizados [4].

3 Características generales de la *visión artificial*

El proceso mediante el cual un equipo/sistema informático (cámara fotográfica, cámara de video, escáner, etc.) es capaz de reconocer una imagen se basa, básicamente, en un *sensor de imagen,* que consiste en una matriz de elementos fotosensibles (celdas) que detectan y capturan la información que compone dicha imagen, convirtiendo en señales eléctricas la atenuación que sufren las ondas de luz cuando atraviesan un cuerpo o son reflejadas por él. Esas señales eléctricas pueden ser convertidas, analizadas, almacenadas y representadas posteriormente como un patrón que se almacena en forma de archivo informático.

El archivo informático contiene datos referentes a los "píxeles" en los que se descompone la imagen, creando una *matriz numérica* que traslada al ámbito matemático una técnica que podría entenderse como la "versión digital" de la técnica pictórica conocida como "*puntillismo*" [5] (ver Ilustración 1), mediante la cual hubo artistas que elaboraron sus obras sobre lienzo.

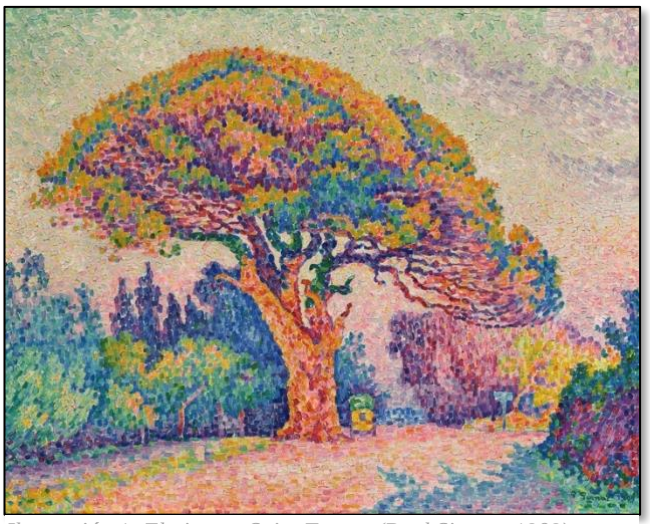


Ilustración 1: El pino en Saint Tropez (Paul Signac, 1909)

La información que se almacena de cada píxel varía en tamaño y forma, según el tipo de imagen del que se trate (blanco y negro, escala de grises, color, etc.)

Tanto los sensores de imagen como los archivos informáticos pueden utilizar diferentes tecnologías y formatos que no son objeto de este trabajo (p.ej. sensores CCD[2], ficheros .jpg[3] o ficheros .mp4[4]), dando lugar a una amplia variedad de equipos y tipos de ficheros.

> En comparación con la visión humana, la captura de imágenes con este tipo de equipos informáticos es mejor en el aspecto cuantitativo, por su capacidad de medición de escenas estructuradas con gran velocidad, pudiendo serlo también en lo cualitativos, dependiendo de la precisión del equipo. En cualquier caso, este es un paso previo a lo que entendemos por visión artificial y no debe confundirse con ella.

[2] Un dispositivo de carga acoplada (CCD) es un circuito integrado que contiene un número determinado de condensadores enlazados o acoplados.
[3] Formato de compresión de imágenes con alta calidad que fue acuñado por Joint Photographic Experts Group.
[4] Los archivos MPEG-4 parte 14 (MP4) emplean un formato que permite de almacenar contenido multimedia, como audio, vídeo y subtítulos.

Mediante la visión artificial se va más allá de la simple representación digital de las imágenes, llevando a cabo un análisis de las imágenes.

Con la *visión artificial* se analizan imágenes (bidimensionales) que normalmente se obtienen del mundo real (tridimensional), gracias a un modelo que proporciona información sobre aquello que está representado en esas imágenes analizadas.

La *visión artificial*, como se exponía anteriormente, es una parte de la Inteligencia Artificial y se centra en el desarrollo y el perfeccionamiento de técnicas que permiten a las máquinas ver, identificar y procesar imágenes mediante procedimientos análogos a los empleados por una persona con las capacidades del sentido humano de la vista.

Según la Asociación de Imágenes Automatizadas (AIA), la *visión artificial* incluye las aplicaciones (industriales y no industriales) que, mediante la combinación de hardware y software, hacen posible el *procesamiento de imágenes*, mejorando la calidad y proporcionando una sustancial mejora en la eficacia y la eficiencia de los procesos [6].

A modo de resumen de lo expuesto hasta ahora, la *visión artificial* proporciona información sobre objetos reales que fueron captados en una imagen digital, gracias a la aplicación de algoritmos que analizan el contenido de los ficheros en los que se almacena dicha imagen. Esos ficheros consisten, básicamente, en una matriz de datos que representan a los píxeles que componen la imagen digital.

A los efectos del presente trabajo se va a poner el foco en las características del *software de visión artificial*, aunque no se puede dejar de mencionar la importancia de los componentes físicos (incluyendo aspectos que pueden resultar determinantes tal y como podría ser la iluminación),

pero consideramos que son los algoritmos del sistema de visión artificial lo que realmente la diferencia de otros sistemas que se limiten a la captura de imágenes sin proceder a su análisis.

Así pues, centrándonos en el procesamiento de las imágenes, entendido como el mecanismo que permite extraer información de una imagen y ponerla a disposición de los usuarios mediante sistemas de información, éste requiere de un *software*, diseñado específicamente para comparar las características detectadas en la imagen digital con las especificaciones que corresponda, y de unas elevadas *capacidades de computación* para ejecutar los algoritmos que llevan a cabo ese procesamiento.

Nos encontramos con que, por ejemplo, un sistema de *Visión artificial* es capaz de inspeccionar con enorme exactitud un gran número de piezas por minuto, detectando aquellas que no cumplen determinadas medidas o estándares que el ojo humano difícilmente podría detectar a una velocidad mucho menor, sin embargo, la visión humana sigue siendo mejor que la *Visión artificial* cuando se trata de la interpretación cualitativa de escenas complejas. En la Tabla 1 se recogen algunas de las ventajas y de las desventajas tanto de la visión humana como de la *visión artificial* [7]:

Tabla 1: Ventajas y desventajas de la visión humana y artificial

Ventajas	Desventajas
Visión humana	
Se basa en el conocimiento previo, lo que permite completar la información que percibimos cuando no se dispone de las imágenes completas.	Poca exactitud para medir la intensidad de los colores.
Directamente relacionado con el punto anterior, dispone de una alta capacidad para reconocer y analizar objetos.	Sujeta al cansancio temporal o definitivo, lo que limita su tiempo de trabajo efectivo.
Se adapta a una gran variedad de situaciones sin necesidad de ningún tipo de cambio.	Sensible a efectos ópticos.
La resolución de captura es razonablemente alta	Incapaz de medir con exactitud.
visión artificial	
Los efectos ópticos no le afectan.	Susceptible a la pérdida de información dependiendo del hardware.
Puede medir magnitudes físicas con una alta precisión.	Dificultad para hacer mediciones cuando cambia la escala.
Permite el análisis objetivo de las imágenes, proporcionando información detallada de ella.	Susceptible a errores por cambios de iluminación.
Puede trabajar de forma continua mucho tiempo sin perder precisión en los trabajos.	No puede detectar movimientos directamente.

Veamos a continuación una serie de ejemplos que ilustren gráficamente algunas de esas ventajas e inconvenientes:

— Ejemplo 1: Magnitud de intensidad de colores. La visión humana no es capaz de distinguir con exactitud todos los colores del espectro visible y diferenciar los que se encuentran próximos (Ilustración 2), a diferencia de la *Visión artificial* que sí puede hacerlo con gran exactitud.

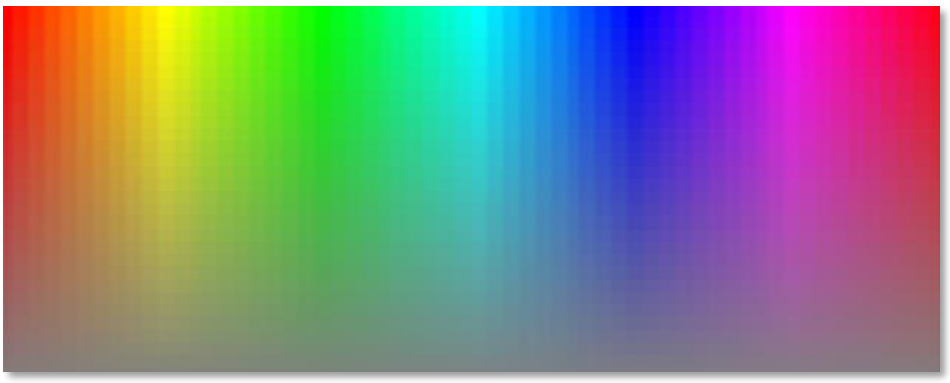


Ilustración 2: Espectro de colores visibles

— Ejemplo 2: Efectos ópticos. La visión humana a veces sufre "conflictos" por la interpretación que hace el cerebro de lo que percibe, por ejemplo, en la Ilustración 3 creemos percibir algunos puntos negros donde en realidad solamente hay puntos blancos. La *Visión artificial* está libre de este tipo de errores.

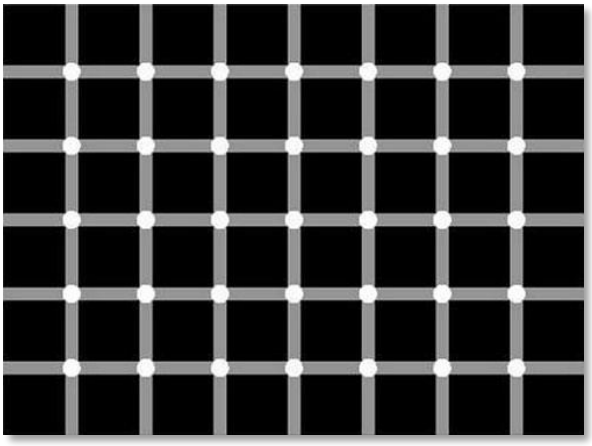


Ilustración 3: Efecto óptico. Puntos blancos/negros

— Ejemplo 3: Pérdida de información: Dependiendo del hardware, la *Visión artificial* puede perder eficacia por falta de calidad en las imágenes, por ejemplo, por una resolución baja (Ilustración 4).

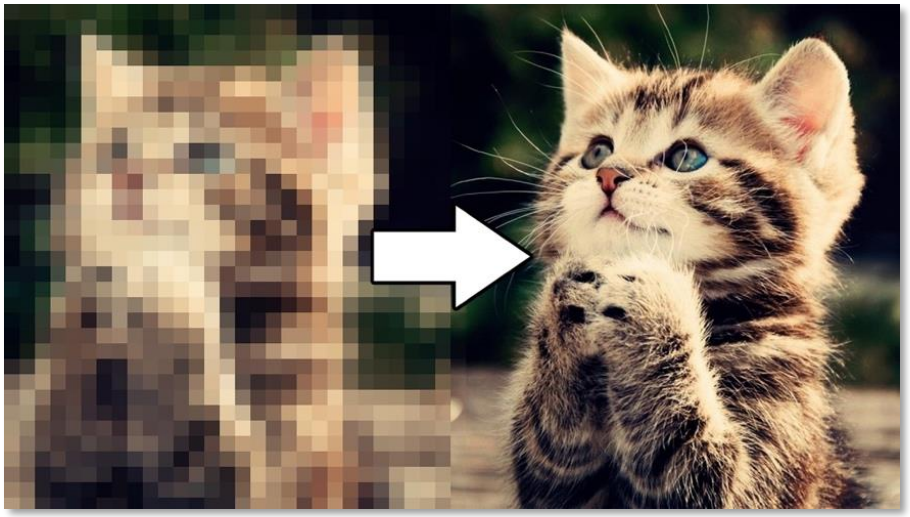


Ilustración 4: Imagen con resolución baja

— Ejemplo 4: Difícil reconocimiento de objetos ante cambio de escenarios. En la Ilustración 5, una persona reconocería todos los objetos como iguales (limones). Sin embargo, un sistema de *visión artificial* , sin entrenamiento previo, serán objetos no es capaz de inferir que el limón cortado es “igual de limón” que los limones sin cortar.

Ilustración 5: Limones mostrados de formas diferentes

La *Visión artificial* hace uso de diversas técnicas que nos permiten conseguir información inteligible procedente de esas matrices numéricas con las que se representan las imágenes. Entre esas técnicas podemos señalar:

— Matemáticas: Para eliminar el ruido de la imagen, detectar los bordes, los objetos o su color.
— Aprendizaje de máquina (machine learning): Normalmente los programas de *Visión artificial* parten de un escenario determinado, por lo que es necesario que nuestro programa tenga un proceso de aprendizaje que le permita adaptarse a otros escenarios.

4 Computación en la nube.

La computación en la nube ("cloud computing"), a menudo denominada simplemente "la nube" ("el cloud"), consiste en el suministro de recursos informáticos a petición, desde aplicaciones hasta centros de datos, a través de Internet y con un modelo de pago según uso.

4.1 Tipos de nube según la propiedad del servicio

La nube, dependiendo de quien sea el propietario de servicio, puede ser:

- Nube pública.
- Nube privada.
- Nube híbrida.

4.1.1 Nube pública

El concepto de uso de la nube pública es el mismo que el de uso del gas o de la electricidad, en el sentido de que, al igual que los propietarios e inquilinos de una vivienda no son dueños del agua que sale de sus tuberías ni se preocupan por las operaciones de la planta generadora de electricidad, los clientes de la nube pública no son dueños del almacenamiento que utilizan ni gestionan servidores o actualizan versiones de su software. La nube está disponible gracias a unos equipos y unas conexiones que los usuarios utilizan en base a unos acuerdos que establecen y en base a los cuales pagan por lo que usan.

Se conoce como nube pública un conjunto de servicios informáticos que ofrecen proveedores externos, a través de Internet, y que se ponen a disposición de quien desee utilizarlos en forma de plataformas y otras capacidades ajenas a la infraestructura de los clientes, evitándoles la necesidad de adquirir y mantener el hardware y/o software que requieren.

El hecho de que la nube sea pública no implica ni que sea gratuita ni que los datos disponibles en ella estén disponibles sin ningún tipo de limitaciones de acceso:

- Los servicios prestados pueden ser *gratuitos* o *de pago.* En el caso de ser de pago, los clientes pueden pagar exclusivamente por el uso que hacen de esos servicios (ciclos de CPU, almacenamiento o el ancho de banda que consumen, según el caso).

— Los clientes acceden a los servicios a través de una cuenta propia, que les permite disponer de las prestaciones que haya contratado, con las correspondientes medidas de seguridad.

La conexión de los usuarios con la nube pública se realiza a través de Internet, accediendo a través de una interfaz de usuario basada en el navegador, con un aspecto que puede variar desde un correo web hasta *sistemas ad hoc* con los que administrar aplicaciones web.

Las empresas proveedoras se encargan de la gestión del backend y suministran el hardware (servidores, unidades de almacenamiento, etc.), lo que incluye medidas de redundancia para impedir los cortes en el servicio.

La nube pública proporciona una alternativa muy atractiva para cualquier tipo de empresa, especialmente a los usuarios particulares, y a las empresas jóvenes y medianas, que ven en ellas una forma de reducir los gastos relacionados con las tecnologías de la información [8].

Entre las características de la nube pública se encuentran las siguientes [8]:

— **Costes según demanda**. La facturación de los costes en la nube pública normalmente se hace en base al uso hecho de los servicios contratados, proporcionando ahorro a los usuarios, ya que evita los costes de compras y mantenimiento de hardware y software, a la vez que proporciona las capacidades necesarias en cada momento, olvidándose de tener que hacer adquisiciones orientadas a disponer permanentemente de las capacidades necesarias para los momentos de mayor demanda.
— **Escalabilidad**. Es un factor muy importante, que garantiza una gran flexibilidad a la hora de adaptar los servicios contratados a las necesidades de cada momento.
— **Interfaz de usuario basada en la web**. Las transacciones normalmente se hacen a través del navegador, en un entorno web, tanto para el uso de los servicios contratados como para la propia relación con el proveedor de esos servicios (modificación de los servicios contratados, pagos, etc.), lo que hace esos accesos sencillos y especialmente económicos, pues el cliente solamente debe preocuparse de su dispositivo de acceso a internet, de su navegador y del acceso a internet propiamente dicho (*lean client*, cliente “liviano”), con la ventaja añadida de, en función de las medidas de seguridad que se apliquen, poder acceder a los servicios desde cualquier lugar del mundo con solo tener acceso a internet.
— **Fiabilidad**. El proveedor aporta una garantía de sus servicios que proporciona una garantía difícil de conseguir si se gestionan directamente con los medios de cliente, incluyendo redundancia de hardware que impide (o al menos dificulta) las caídas de los sistemas.
— **Eficiencia**. Los proveedores de computación en la nube tramitan los procesos con gran rapidez.
— **Protección de datos**. Los prestadores suelen ser especialmente sensibles a la hora de cumplir con la normativa vigente de protección de datos, en este sentido es aconsejable ser riguroso a la hora de elegir un proveedor, por las responsabilidades que se pueden derivar de la falta de observación de esa normativa, por eso, lo recomendable es un proveedor con sede en Europa, sujeto al derecho europeo, y que gestione sus servidores en suelo europeo (si la empresa se localiza en Europa está sujeta al Reglamento General de Protección

de Datos -RGPD-, el cual prohíbe depositar datos personales europeos fuera de su territorio, a excepción de ciertos terceros países considerados seguros).

— **Protección del medioambiente**. En dos niveles, en el global, porque se optimiza el uso de recursos gracias a su centralización, y en el particular, porque frecuentemente los proveedores hacen uso de energía producida por procedimientos sostenibles que quedan fuera del alcance de las pequeñas/medianas empresas que emplean sus servicios.

La nube pública es, de facto, un pilar fundamental de la transformación digital, porque permite a un amplio público acceder a los datos en cualquier momento y lugar, desde una diversidad de equipos que incluye los terminales móviles (tablets y teléfonos) y proporciona capacidades que de otra forma estarían restringidas a las grandes empresas.

Además de lo referente a los equipos informáticos, utilizar una nube pública también permite liberar personal de los departamentos TIC que puede emplearse en otros cometidos.

Frente a todas las ventajas que aporta este modelo, la nube pública conlleva los riesgos asociados a la descentralización y a la compartición de recursos que la caracterizan, si bien, los proveedores aseguran que la nube pública es tan segura como la nube privada, siempre que se implemente correctamente y se empleen métodos de seguridad adecuados (como sistemas IDPS).

En principio, cualquier empresa o persona puede ofrecer una nube pública, de hecho, hay miles de proveedores en todo el mundo (como ejemplo se puede consultar el número de colaboradores de Red hat [9]), siendo los siguientes los que se consideran entre los más importantes:

— Alibaba Cloud[5]
— Amazon Web Services (AWS)[6]
— Google Cloud[7],
— IBM Cloud[8].
— Microsoft Azure[9].

[5] https://eu.alibabacloud.com/en
[6] https://aws.amazon.com/es/
[7] https://cloud.google.com
[8] https://www.ibm.com/es-es/cloud
[9] https://azure.microsoft.com/es-es/

4.1.2 Nube privada

La nube privada es un modelo de "*cloud computing*" que está dirigido a una audiencia limitada (p. ej. a una empresa concreta), ofreciendo servicios informáticos a los miembros de esa audiencia, a través de Internet o de una red interna privada, y empleando equipos que son de su propiedad, no del proveedor de servicio.

La nube privada puede ubicarse físicamente en el propio centro de proceso de datos de la organización a la que sirve o puede hospedarse en un proveedor de servicios externo, pero, en cualquier caso, en una nube privada, los servicios y la infraestructura siempre se mantienen en una red privada y *el hardware y software se dedican únicamente a su organización.*

> La tecnología empleada en la nube privada es la misma que la de la nube pública, diferenciándose básicamente en el entorno al que prestan sus servicios, de modo que en la nube privada se trata de sistemas que están reservados a las personas de ese entorno (empresa) y que son gestionados por personas de este.

Este modelo aporta gran parte de las ventajas de la nube pública, si bien el hecho de que el control de sus recursos se haga mediante una infraestructura propia, a la vez que aumenta los gastos de personal, administración y mantenimiento, añade otras ventajas específicas:

- Más flexibilidad. El entorno se puede personalizar para adaptarlo a las necesidades específicas de la organización.
- Más control y privacidad. Como consecuencia de que no se compartan los recursos con nadie más.
- Más escalabilidad. Derivada de las otras ventajas.

Con respecto a las infraestructuras tradicionales (las que no emplean "*cloud computing*"), la nube privada sigue representando un ahorro de costes de capital (CAPEX) muy importante, porque permite reducir el número total de servidores, así como de costes operacionales (OPEX), reduciendo los gastos, por ejemplo, en equipos, electricidad y mantenimiento de hardware.

Los usuarios más frecuentes de las nubes privadas son organizaciones medianas o grandes que requieran de un mayor control y confidencialidad, tales como organismos estatales o instituciones financieras.

4.1.3 Nube híbrida

Como se ha visto en los dos puntos anteriores, lo que distingue fundamentalmente a la nube privada de la nube pública es, en esencia, lo mismo que separa a lo privado de lo público.

Por su parte, la nube híbrida es una solución que combina una nube privada con uno o más servicios de nube pública, con software exclusivo que permite la comunicación entre cada servicio distintivo. Una nube híbrida se apoya en un único plano de administración, a diferencia de una estrategia multinube donde los administradores deben gestionar cada entorno de nube por separado [10].

> Por ejemplo, una organización puede almacenar su información sensible en una nube privada (o un centro de datos local) a la vez que aprovecha las ventajas que proporcionan los servicios ofrecidos por una nube pública, reduciendo la entidad de la infraestructura propia.
>
> Así, una infraestructura de nube híbrida incluye frecuentemente una plataforma pública de infraestructura como servicio (IaaS), una nube privada o un centro de datos y acceso a una red segura.

La tendencia actual es inclinarse hacia soluciones híbridas que se adapten adecuadamente a las necesidades de cada usuario (empresa), disminuyendo la adopción de soluciones puramente públicas o privadas.

4.2 Modalidades de computación en la nube según los servicios ofrecidos

Los tipos de informática en la nube son modelos de implementación de servicios que le permiten elegir el nivel de control sobre su información y los tipos de servicios que tiene que proporcionar. Hay tres tipos principales de servicios de informática en la nube, a veces denominados "pila informática en la nube", porque se basan unos en otros [11]*:*

- Infraestructura como servicio (IaaS)
- Plataforma como servicio (PaaS)
- Software como servicio (SaaS)

El siguiente diagrama (Ilustración 6) muestra de forma gráfica y resumida los diferentes componentes que normalmente se integran en cada uno de los tres modelos (IaaS, PaaS y SaaS).

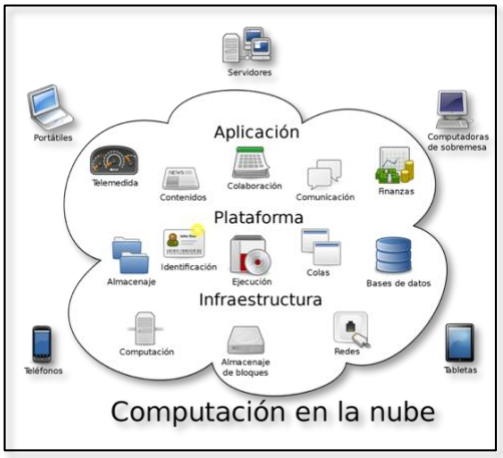


Ilustración 6: Visión esquemática de los componentes más frecuentes de la computación en la nube[10]

[10] https://commons.wikimedia.org/wiki/File:Cloud_computing.svg#/media/File:Cloud_computing-es.svg

4.2.1 Infrastructure as a Service (IaaS)

Es la categoría más básica que se utiliza para acceder, a través de Internet, a servicios de Infraestructura informática, básicamente almacenamiento, capacidad de computación e infraestructura de red (ancho de banda) y puede presentarse tanto en nubes públicas como en privadas.

Los usuarios, en lugar de tener que comprar hardware, compran *servicios IaaS* basado en el consumo, de forma similar a como se paga la electricidad que se consume o se factura otros servicios públicos.

La IaaS no debe confundirse con el concepto de Hardware as a Service (HaaS), ya que éste último es similar a un "leasing" mediante el que los equipos "alquilados" se alojan en instalaciones de quien adquiere el servicio (del cliente), mientras que en IaaS los equipos están en instalaciones del proveedor, que los maneja y mantiene. En IaaS en la nube pública, como se vio más arriba, los clientes pagan exclusivamente por el uso que hacen de sus servicios.

Sus características se derivan directamente de las características propias del *cloud computing*:

- **Escalabilidad y flexibilidad**: Se pueden modificar los recursos disponibles en función de las necesidades reales de cada momento.
- **Pago por uso**. Se paga en función de lo que realmente se utiliza.
- **Reducción de costes**. Elimina el coste de implementación y mantenimiento de hardware físico.
- **Independencia de localización**. Permite utilizar el servicio desde cualquier lugar con Internet.
- **Seguridad**. Proporciona unos altos estándares de seguridad que son difícilmente igualables por instalaciones propiedad de los clientes.
- **El cliente no realiza trabajos de administración**. Las tareas administrativas son ejecutadas por el proveedor

4.2.2 Platform as a Service (PaaS)

Para que pueda ejecutarse una aplicación o un programa, además de la infraestructura tecnológica, es necesaria una plataforma que lo soporte, que es lo que proporciona desde la nube la alternativa de "*Platform as a Service"* (PaaS) y que, al igual que IaaS, está disponible tanto en las nubes públicas como en las privadas.

PaaS ofrece herramientas que permiten crear y almacenar aplicaciones, dando el apoyo necesario para todo su ciclo de vida (compilación pruebas, implementación, administración y actualización) en un entorno seguro, además de los servicios de IaaS que se han mencionado en el punto anterior (por eso, la IaaS podría considerarse la capa inferior de los servicios disponibles en la nube).

En resumen, las funciones principales de PaaS son:

- Proporciona una plataforma con herramientas para probar, desarrollar y alojar aplicaciones en el mismo entorno.
- Permite a las organizaciones centrarse en el desarrollo, sin tener que preocuparse por la infraestructura subyacente.

- Los proveedores gestionan la seguridad, los sistemas operativos, el software de servidor y las copias de seguridad.
- Facilita la colaboración incluso si los equipos trabajan en remoto.

Los servicios de PaaS, complementando los de IaaS, permiten ejecutar el ciclo de vida completo de una aplicación, proporcionando las capacidades (hardware y software) y los entornos necesarios (desarrollo, pruebas y explotación) en una plataforma que, dependiendo de los proveedores, puede contar con unos niveles elevados de seguridad que incluyan la redundancia geográfica.

Por lo expuesto, PaaS permite reducir los costes, evitando procesos de compras y gastos y labores de administración de software y de infraestructura de aplicaciones, ofreciendo un entorno seguro y flexible, además de escalable.

4.2.3 Software as a Service (SaaS)

El *Software como Servicio* (SaaS) es la capa superior de la arquitectura de servicios del "cloud computing" ofreciendo la alternativa más completa, un software integral, en la que, además de los servicios de Infraestructura (IaaS) y plataforma (PaaS), proporciona aplicaciones basadas en la nube, disponibles a través de internet y listas para ser utilizadas directamente tal y como ocurre con el correo electrónico o con herramientas ofimáticas como, por ejemplo, *Microsoft Office 365*.

Los usuarios no tienen que instalar las aplicaciones en sus dispositivos locales, ya que residen la nube, lo que permite acceder a ellas desde cualquier dispositivo compatible (ordenador, tablet, móvil…) y desde cualquier lugar del mundo con acceso a internet, almacenando datos y permitiendo colaborar entre usuarios en proyectos.

Funciones principales

- Los proveedores de SaaS proporcionan a los usuarios el software y las aplicaciones a las que se suscriben según los acuerdos que contemple su suscripción.
- Los usuarios no tienen que gestionar de ninguna forma el software, olvidándose de todo lo que se refiera a su mantenimiento, desde la misma instalación hasta la actualización de versiones.
- Los datos se almacenan en la nube, de forma segura, estando a salvo de pérdidas de datos y ofreciendo opciones de rescate y backup.
- Los recursos, como en todas las opciones vistas hasta ahora, se pueden ampliar o reducir en función de las necesidades.
- Aportan opciones de movilidad prácticamente ilimitada, siendo suficiente con disponer de un dispositivo compatible y de acceso a internet.

4.2.4 X as a Service (XaaS)

Por ultimo, a modo de resumen y como orientación de las tendencias actuales nos referiremos al término (acrónimo) *XaaS*, derivado de "X as a Service", que viene a ser una forma de referirse a "*todo como un servicio*", algo así como que cualquier recurso informático estaría a disposición de quien lo necesite en forma de servicio accesible mediante la nube, reduciendo (eliminando) la necesidad de adquirir o mantener capacidades técnicas propias.

Por lo dicho, XaaS no es un tipo de servicio en sí mismo, sino más bien una filosofía que engloba todos los "*… as a Service*", entre los que se encuentran los ya mencionados IaaS, PaaS y SaaS, pero a los que habría que añadir una lista que no deja de crecer [12]:

- API as a Service (AaaS).

- Artificial intelligence as a Service (AIaaS)
- Business process as a Service (BPaaS)
- Content as a Service (CaaS)
- Database as a Service (DaaS)
- Desktop as a Service (DaaS)
- Energy storage as a Service (ESaaS)
- Malware as a Service (MaaS)
- Monitoring as a Service (MaaS)
- Network as a Service (NaaS)
- Payments as a Service (PaaS)
- Robot as a Service (RaaS)
- Storage as a Service (SaaS)
- Unified Communications as a Service (UCaaS)

Entre esta maraña de "*as a Service*", en lo referente al objetivo de este trabajo, es obligada la mención al "*Image Processing as a Service*" (IPaaS), que a su vez tendría sus particularidades tanto en lo referente a Infraestructura (IPIaaS) como a Plataforma (IPPaaS) y a Software (IPSaaS).

5 Microsoft Azure

Cuando se decide la puesta en marcha de un proyecto que requiera el uso de técnicas avanzadas de Inteligencia Artificial, una de las tareas más complejas es el desarrollo de los modelos, por eso, una alternativa a valorar siempre es el uso de modelos ya desarrollados, probados y validados.

Las características de la Inteligencia Artificial, debido a su permanente necesidad de actualización y "*aprendizaje*", la hacen un firme candidato a integrarse como servicios del tipo *PaaS* proporcionados en la nube. De hecho, se trata de un servicio que ofrecen las principales plataformas de cloud computing en forma de *AIaaS* (Inteligencia Artificial como servicio) [13].

En línea con los objetivos de este trabajo, vamos a pasar a estudiar con más detalle las características de *Azure,* la plataforma en la nube de Microsoft, y, dentro de este producto, los denominados servicios cognitivos (*Azure Cognitive Services*).

Microsoft Azure es una nube pública que, a día de hoy, forma una red con más de 100 data centers

Microsoft Azure no se centra exclusivamente en aplicaciones propias de Microsoft, por el contrario, soporta aplicaciones de otros fabricantes como SAP, Oracle, IBM… así como Sistemas Operativos como Linux.

Esta nube pública forma una de las redes más grandes del planeta, con más de 100 data centers ubicados por todo el mundo [14] (estando prevista la apertura próximamente de uno en Madrid), lo que significa un punto importante para tener en cuenta por aquellos clientes que tengan necesidad de acceder desde diversos continentes/países con baja latencia y cumpliendo las normativas locales.

Azure proporciona un conjunto de servicios (Ilustración 7) que permite compilar, administrar e implementar aplicaciones en un entorno de red global y seguro, siendo especialmente relevantes el *Portal de Azure* y el *Azure MarketPlace*, por proporcionar funcionalidades que facilitan el uso de todo el sistema:

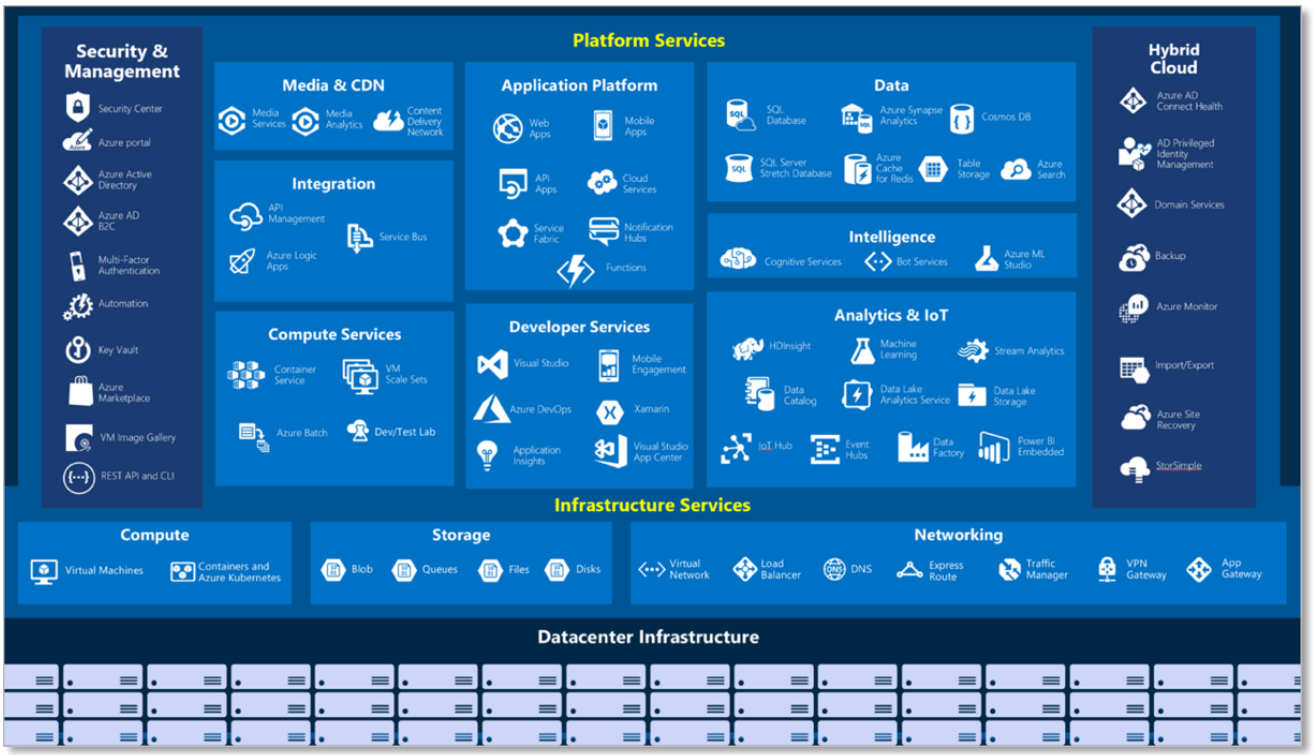


Ilustración 7: Servicios de Microsoft Azure

- El Portal de Azure es una consola, basada en web, que proporciona una alternativa amigable y sencilla a las herramientas de línea de comandos y permite administrar la suscripción de Azure mediante una interfaz gráfica de usuario.
- Azure MarketPlace[11] (Ilustración 8) es la herramienta que conecta a los usuarios de Azure con los partners de Microsoft, proveedores de software independientes y empresas que ofrecen sus soluciones y servicios, optimizados para ejecutarse en Azure. Los clientes de Azure MarketPlace pueden buscar, probar, comprar y aprovisionar aplicaciones y servicios de cientos de los principales proveedores de servicios. Todas las soluciones y los servicios que se ofrecen en este sitio están certificados para ejecutarse en Azure.

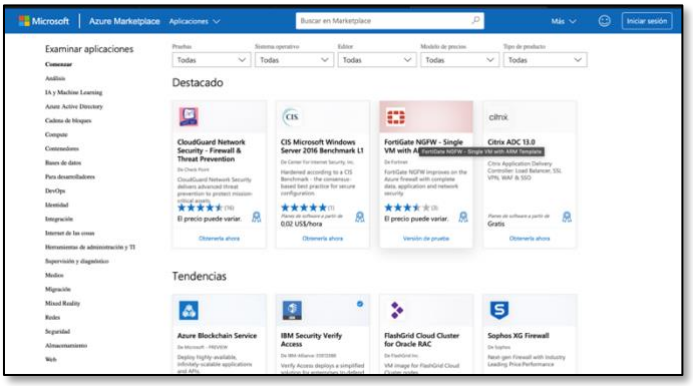


Ilustración 8: Azure MarketPlace

[11] https://azuremarketplace.microsoft.com/es-ES

Azure, como queda dicho anteriormente, es una nube pública, de pago, y abarca los tres tipos de servicios esenciales a los que nos referíamos en el punto 4.2 (Iaas, PaaS y SaaS), para entender mejor lo cual es relevante la el modelo de nube preconizado por Microsoft que se muestra en la Ilustración 9.

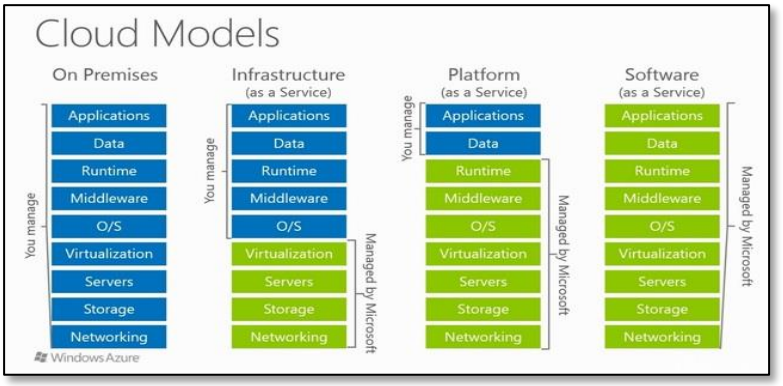


Ilustración 9: Gestión de IaaS, PaaS y SaaS (Fuente: Microsoft)

Microsoft agrupa los servicios de Azure en las categorías que se relacionan a continuación, junto con la descripción que el propio Microsoft añade para cada una de ellas [15]:

- **Administración y Gobernanza.** "*Obtenga herramientas de administración y gobernanza de Azure integradas que ayudarán a los administradores del sistema y a los desarrolladores a mantener los recursos protegidos y conformes con la normativa, tanto en el entorno local como en la nube. Supervise la infraestructura y las aplicaciones, aprovisione y configure recursos, actualice aplicaciones, analice posibles amenazas, cree copias de seguridad de los recursos, implemente recuperación ante desastres, aplique directivas, automatice los procesos e, incluso, administre los costos, a lo largo de todo el ciclo de vida de TI*:
  - *Herramientas de administración para supervisión*
  - *Herramientas de administración para configuración*
  - *Herramientas de administración para gobernanza*
  - *Herramientas de administración para seguridad y protección*"
- **Almacenamiento**. "*Obtenga un almacenamiento seguro en la nube que proteja la infraestructura de datos mientras crea aplicaciones y mejores servicios para sus clientes. Integre los datos locales con sus datos en la nube para obtener información crucial. Escale globalmente y ejecute las cargas de trabajo más exigentes al mismo tiempo que satisface los requisitos de conformidad y privacidad*".
- **Análisis**. "*Recopile, almacene, procese, analice y visualice datos de cualquier variedad, volumen o velocidad*".
- **Bases de datos**. "*Azure ofrece una selección de bases de datos relacionales, NoSQL y en memoria totalmente administradas, que abarcan motores patentados y de código abierto, para adaptarse a las necesidades de los desarrolladores de aplicaciones actuales. La administración de la infraestructura, incluidas la escalabilidad, la disponibilidad y la seguridad, está automatizada, lo que ahorra tiempo y dinero*".
- **Blockchain**. "*Cree y administre aplicaciones basadas en la cadena de bloques con un conjunto de herramientas integradas*".

- **Contenedores**. "*Ahorre costos al migrar mediante lift-and-shift sus aplicaciones actuales a contenedores y cree aplicaciones de microservicios para ofrecer valor a los usuarios con más rapidez. Sea cual sea la fase del proceso de modernización de sus aplicaciones en la que se encuentra, puede agilizar el desarrollo de aplicaciones de contenedor y satisfacer sus requisitos de seguridad*".
- **DevOps**. "*Ya sea que esté comenzando la implementación de DevOps o si busca integrarla con su cadena de herramientas y procesos actuales, crear una canalización de entrega continua completa es más rápido y seguro con las tecnologías de Azure DevOps. Pase menos tiempo administrando su conjunto de herramientas y más tiempo centrándose en el valor del cliente. Crear, lanzar, probar y supervisar aplicaciones móviles y en la nube es fácil y confiable con tecnologías de DevOps que permiten ofrecer innovación de forma continua mediante cualquier cadena de herramientas*".
- **Herramientas para desarrolladores**. "*Azure ofrece un conjunto integral de herramientas de desarrollo que permiten a cualquier desarrollador ofrecer aplicaciones en la nube usando la plataforma o el lenguaje que prefiera. Codifique con el lenguaje que elija con una serie de SDK y benefíciese de entornos de desarrollo integrados (IDE) completos y editores con funcionalidades de depuración avanzadas y soporte técnico de Azure integrado*".
- **Hybrid + Multicloud**. "*Obtenga la coherencia y la flexibilidad necesarias para poder innovar en cualquier lugar, ya sea en el entorno local, en un entorno multinube o en el perímetro*".
- **Inteligencia Artificial y Machine Learning.** "*Obtenga características de Inteligencia Artificial probadas, seguras y responsables que puede usar como desee con Azure AI. Cree soluciones críticas que puedan analizar imágenes, reconocer la voz, hacer predicciones con datos e imitar otros comportamientos humanos inteligentes, todo ello con Azure AI*".
- **Identidad**. "*Proteja sus aplicaciones y datos en la puerta de entrada con la identidad de Azure y las soluciones de administración de acceso. Defiéndase de los intentos de inicio de sesión malintencionados y proteja las credenciales con controles de acceso basados en el riesgo, herramientas de protección de identidad y sólidas opciones de autenticación, sin interrumpir la productividad*".
- **Integración**. "*Cree soluciones integradas que conecten aplicaciones y servicios en el entorno local y en la nube. Unifique sus flujos empresariales para que sean coherentes y escalables. Exponga sus API para los desarrolladores y cree oportunidades de nuevos modelos empresariales*".
- **Internet de las cosas**. "*Obtenga soluciones de IoT productivas y escalables. Tanto si ya es un desarrollador para la nube con experiencia como si acaba de empezar a trabajar con IoT, nuestros eficaces servicios y herramientas le ayudarán a desarrollar soluciones de IoT de nueva generación*".
- **Migración**. "*Simplifique y acelere la migración a la nube con guías, herramientas y recursos*".
- **Movilidad**. "*Cree aplicaciones móviles de alta calidad basadas en la nube para llegar a sus clientes en cualquier dispositivo*".
- **Multimedia**. "*Entregue contenido de vídeo de alta calidad donde quiera, cuando quiera y en el dispositivo que quiera*".
- **Proceso**. "*Acceda a funcionalidad de proceso, virtualización y escalado a petición en la nube, y pague solo por los recursos que use". "Independientemente de que*

vaya a crear aplicaciones o implementar las existentes, Azure Compute ofrece la infraestructura necesaria para ejecutar las aplicaciones".

- **Realidad mixta**. "*Combine el mundo físico y el mundo digital para crear experiencias de colaboración inmersivas*".
- **Redes**. "*Conecte las infraestructuras y los servicios locales con los de la nube para ofrecer a los clientes y usuarios la mejor experiencia posible*".
- **Seguridad**. "*Proteja datos, aplicaciones e infraestructura rápidamente con servicios de seguridad integrados en Azure, que incluyen inteligencia de seguridad inigualable para ayudar a identificar anticipadamente amenazas que evolucionan con rapidez, de forma que pueda responder a ellas rápidamente. Implemente una estrategia en capas para una defensa en profundidad de su identidad, datos, hosts y redes. Unifique la administración de seguridad y habilite la protección contra amenazas avanzada para entornos en la nube híbrida*".
- **Web**. "Compile e implemente rápidamente sus aplicaciones web con una plataforma totalmente administrada, sin la carga de administrar la infraestructura".
- **Windows Virtual Desktop.** "*La mejor experiencia de escritorio virtual, entregada en Azure*".

5.1 Servicios "*cloud computing*" de Azure

5.1.1 Iaas en Azure

La oferta de IaaS de *Azure* proporciona las capacidades que vimos al tratar las generalidades de IaaS (ver 4.2.1), subcontratando a *Microsoft* las necesidades de red y computación, lo que permite a los clientes prescindir de sus servidores. *Microsoft* administra la infraestructura en nombre del cliente, permitiendo comprar, instalar, configurar y operar el software que se ejecute en ella tales como sistemas operativos, aplicaciones y middleware.

Azure ofrece una amplia gama de prestaciones que incluye desde el procesamiento y la conexión en red hasta la seguridad y el almacenamiento, incluido el servicio de contenedores y las máquinas virtuales.

5.1.2 PaaS en Azure

Azure ofrece un amplio conjunto de servicios que aporta a los desarrolladores una gran libertad para elaborar sus sistemas y aplicaciones con la seguridad de contar con las versiones y parches adecuados, sin tener que preocuparse por la configuración del servidos o por otras cuestiones como la disponibilidad de recursos hardware.

5.1.3 SaaS en Azure

Además de usar *Azure* para implementar las aplicaciones propias, es posible acceder a otros software disponible en forma de *SaaS* tales como *Office 365* (Ilustración 10) y *Dynamics 365* (Ilustración 11).

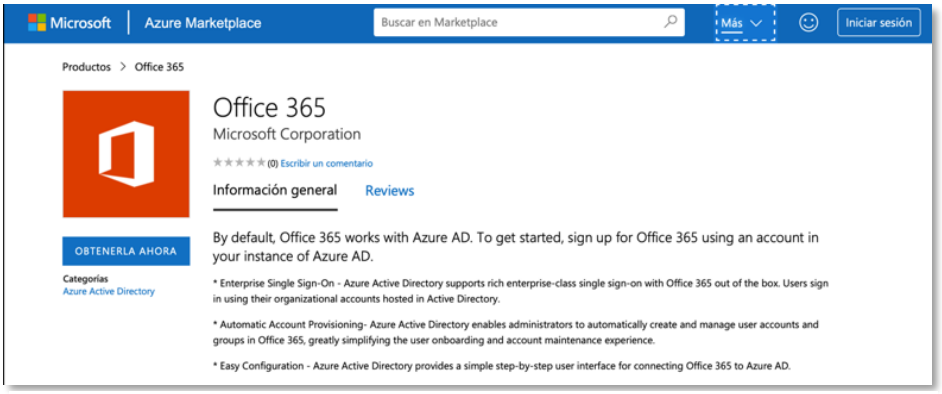


Ilustración 10:MS Office 365 disponible en Azure Marketplace

Ilustración 11: MS Dynamics disponible en Azure Marketplace

A modo de resumen, Microsoft Azure es un servicio de computación en la nube creado por Microsoft para construir, probar, desplegar y administrar aplicaciones y servicios mediante el uso de sus centros de datos.

Proporciona Software como Servicio (SaaS), Plataforma como Servicio (PaaS) e Infraestructura como Servicio (IaaS).

Es compatible con muchos lenguajes, herramientas y marcos de programación diferentes, incluidos software y sistemas específicos de Microsoft y de terceros.

5.2 Azure Cognitive Services

En el contexto de la informática en la nube, la Inteligencia Artificial se basa en una amplia gama de servicios, entre los que destaca especialmente el aprendizaje automático ("*Machine Learning*"), una técnica de ciencia de datos que permite a los equipos prever tendencias en base a los datos existentes, lo que permite a los equipos aprender sin necesidad de programarlos explícitamente.

Al principio de este punto se han relacionado los diversos grupos en los que Microsoft clasifica los servicios de *Azure*, entre los que hemos visto que se encuentra el de "*Inteligencia Artificial y Machine Learning*"[12], que incluye una serie de servicios que permiten desarrollar soluciones de Inteligencia Artificial.

A su vez, esos servicios se subdividen en:

- Servicios de aprendizaje automático (machine learning)[13]
- Servicios de minería de conocimientos (cognitive search)[14]
- Aplicaciones de Inteligencia Artificial.

Entre estas últimas se encuentran una serie de servicios estrechamente relacionados: los servicios de conocimiento ("*cognitive services"*)[15].

Estos servicios de conocimiento o *cognitive services* de *Microsoft Azure* pueden considerarse como servicios de *Inteligencia Artificial, como Servicio* (AIaaS), con API REST[16] y SDK de biblioteca cliente disponibles, que permiten integrar inteligencia cognitiva en las aplicaciones sin necesidad de tener conocimientos previos sobre Inteligencia Artificial, a saber:

- Servicios de decisión
  - **Anomaly Detector**. Funcionalidades de detección de anomalías.
  - **Content Moderator**. Moderación automatizada de imágenes, texto y vídeo
  - **Metrics Advisor**. Servicio que supervisa las métricas y diagnostica problemas.
  - **Personalize**r. Servicio de Inteligencia Artificial que ofrece una experiencia del usuario personalizada
- Servicios de lenguaje
  - **Immersive Reader**. Servicio que permite leer y comprender textos.
  - **Language Understanding**. Servicio que permite enseñar a las aplicaciones a entender los comandos de sus usuarios
  - **QnA Maker.** Servicio que convierte la información en respuestas de conversación de fácil navegación
  - **Text Analytics**. Servicio para evaluar las opiniones y temas, para comprender lo que los usuarios requieren.
  - **Translator Text.** Traducción automática con una llamada a una API.
- Servicios de voz
  - **Speech to Text**. Servicio de voz que convierte audio hablado en texto.
  - **Text to Speech**. Servicio de voz que convierte texto en voz realista.
  - **Speaker Recognition**. Servicio de voz que verifica e identifica a los hablantes
  - **Traducción de voz**. Servicio de traducción de voz en tiempo real.
- Servicios de visión
  - **Computer Vision**. Es un servicio de reconocimiento de imágenes.

[12] https://azure.microsoft.com/es-es/overview/ai-platform/
[13] https://azure.microsoft.com/es-es/services/machine-learning/
[14] https://azure.microsoft.com/es-es/services/search/
[15] https://azure.microsoft.com/es-es/services/cognitive-services/
[16] https://westcentralus.dev.cognitive.microsoft.com/docs/services

- **Custom Vision**. Es un servicio de reconocimiento de imágenes, pero, a diferencia del anterior, con este se pueden crear, implementar y mejorar identificadores de imágenes propios.
- **Face**. Permite detectar, identificar, analizar, organizar y etiquetar caras en las fotos.
- **Form Recognizer**. Servicio de extracción de documentos basado en Inteligencia Artificial.
- **Video Indexer.** Proporciona la capacidad de extraer información profunda (sin necesidad de análisis de datos o conocimientos de codificación) mediante el uso de modelos de Machine Learning basados en varios canales (voz, voces y objeto visual), facilitando una mejora en la interacción con los usuarios.

Al igual que los otros servicios en la nube, AIaaS proporciona una serie de ventajas que, básicamente, son las mismas o se derivan de las ventajas de aquellos:

— Las aplicaciones de Inteligencia Artificial necesitan mucha potencia de procesamiento, lo que hace que se requiera de una infraestructura costosa y a veces inabordable si no fuese por la oportunidad que ofrece la computación en la nube.
— La escalabilidad: Se puede empezar a usar los servicios de Inteligencia Artificial para proyectos pequeños para ir creciendo a medida que surgen nuevas necesidades.
— La usabilidad: Los servicios de IA en la nube, tales como los que ofrece Microsoft Azure, son sencillos de utilizar y no requieren que los desarrolladores sean expertos en esa tecnología.

Con respecto a los inconvenientes, el más importante es la seguridad, pero es un riesgo que se compensa con las ventajas y que puede reducirse haciendo un buen uso de las herramientas que ofrecen los proveedores de servicios.

En lo relativo a las tarifas de Microsoft Azure, existen múltiples opciones en las que se incluye una serie de productos que son siempre gratuitos.

Con el objeto de dar a conocer el producto existe la posibilidad de acceder a una cuenta gratuita, que incluye acceso a un gran número de productos de Azure durante 12 meses y un crédito de €170 para gastar durante los primeros 30 días después de registrarse, así como la de Azure para estudiantes, que también proporciona acceso gratuito durante 12 meses, renovables si se sigue siendo estudiante.

5.3 Computer Vision [16]

Como veíamos en el punto anterior, el servicio *Computer Vision* de Azure es uno de los *servicios de visión* incluido entre los "*Azure Cognitive Services*", que proporciona acceso a algoritmos avanzados que permiten procesar imágenes para obtener información basada en sus contenidos, tanto imágenes accesibles desde el equipo local como imágenes de Internet, accesibles a través de su URL (sin necesidad de descargarlas).

Al cargar una imagen (o al especificar la URL de una imagen), los algoritmos del servicio *Computer Vision* pueden analizar el contenido de la imagen de diferentes maneras basándose en las opciones que elija el usuario, creando aplicaciones que hacen uso de

Inteligencia Artificial sin necesidad de conocimientos específicos en esta rama del conocimiento gracias a:

- El Software Development Kit (SDK) de una biblioteca cliente.
- Llamadas directas a la API REST.

Para el presente trabajo nos vamos a centrar en las capacidades que proporciona la versión 3.1 de la API de este servicio[17] y el correspondiente SDK para PYTHON[18], que nos proporcionan las funcionalidades que se describen a continuación, señalando para cada una de ellas el formato de las correspondientes llamadas, tanto vía SDK como API, así como los parámetros que admiten esas llamadas.

5.3.1 Análisis de una imagen

Computer Vision identifica y etiqueta las características de una imagen, tanto del objeto principal como de su entorno, en interiores o en exteriores, catalogándolos entre una gran variedad de objetos, seres vivos y acciones reconocibles.

Tabla 2: Formas de invocar a la funcionalidad de análisis de imágenes

SDK	`analyze_image_in_stream(image, visual_features=None, details=None, language='en', description_exclude=None, custom_headers=None, raw=False, callback=None, **operation_config)`
API	`https://{endpoint}/vision/v3.1/analyze[?visualFeatures][&details][&language]`

- Método HTTP: POST
- Parámetros:
  - **"Visual features".** Una cadena de texto que indica qué tipos de características visuales se solicita que se analicen (admite varios valores separados por comas).
    - *"Adult" – contenido para adultos, devolviendo una puntuación de confianza para las diferentes clasificaciones (ver anexo 3, punto 1.1).*
    - *"Brands" – detecta marcas comerciales (solo en inglés).*
    - *"Categories" – categoriza el contenido de la imagen de acuerdo con una taxonomía establecida[19] (ver Ilustración 12) (ver anexo 3, punto 1.2).*
    - *"Color" – determina características del color, como el color predominante en la imagen o si es una imagen en blanco y negro (ver anexo 3, punto 1.2).*
    - *"Description" – describe la imagen con una frase (ver anexo 3, punto 1.3).*
    - *Faces – detecta si hay caras en la imagen y, de haberlas, devuelve coordenadas, además de un interpretación de su género y edad (ver anexo 3, punto 1.4).*
    - *"ImageType" – detecta las características de una imagen, por ejemplo si es una imagen prediseñada (clipart) o un dibujo (ver anexo 3, punto 1.4).*
    - *"Objects" – detecta diferentes objetos en la imagen y su localización (solo en inglés) (ver anexo 3, punto 1.5).*
    - *"Tags" – etiqueta la imagen con una lista de palabras relacionadas con su contenido (ver anexo 3, punto 1.2).*
  - **"Details"**. Detecta si hay contenido específico:

[17] https://westcentralus.dev.cognitive.microsoft.com/docs/services/computer-vision-v3-1-ga/operations/56f91f2e778daf14a499f21b
[18] https://docs.microsoft.com/es-es/python/api/azure-cognitiveservices-vision-computervision/azure.cognitiveservices.vision.computervision.operations.computervisionclientoperationsmixin?view=azure-python
[19] https://docs.microsoft.com/es-es/azure/cognitive-services/computer-vision/concept-categorizing-images

- *"Celebrities" – identifica personas conocidas, si las hay en la imagen (ver anexo 3, punto 1.1)*
- *"Landmarks" - identifica edificios o lugares relevantes, si los hay en la imagen.*

- **"Language"**. Una cadena de texto que indica el idioma en el que se devuelve la respuesta
  - *"en" – Inglés (por defecto)*
  - *"es" - Español.*
  - *"ja" - Japonés.*
  - *"pt" - Portugués.*
  - *"zh" - Chino*

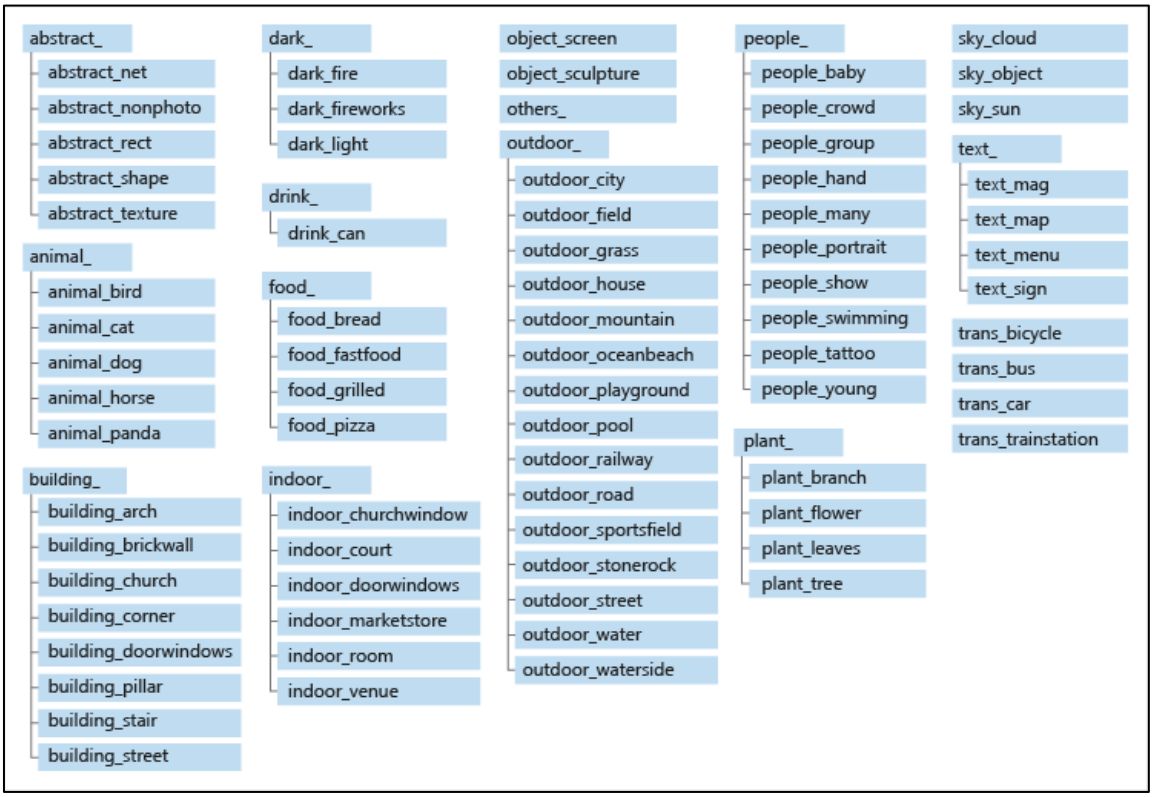


Ilustración 12: Las 86 categorías de la taxonomía de Computer Vision

5.3.2 Descripción de una imagen

Computer Vision genera varias descripciones[20] de la imagen en función de los objetos que identifica en ella, evalúa cada una de esas descripciones y les asigna una puntuación de confianza, devolviendo una lista de puntuaciones de confianza ordenadas de la más alta a la más baja (ver anexo 3, punto 2).

Tabla 3: Formas de invocar a la funcionalidad de descripción de imágenes

SDK	`describe_image_in_stream(image, max_candidates=1, language='en', description_exclude=None, custom_headers=None, raw=False, callback=None, **operation_config)`

[20] Con la opción "*description*" de la funcionalidad de analizar (el punto anterior) se consigue la misma salida que si en esta funcionalidad se solicita solamente una descripción, diferenciándose esta funcionalidad de aquella en que con esta se puede seleccionar el número de descripciones que se desea obtener.

API	https://{endpoint}/vision/v3.1/describe[?maxCandidates][&language]

- Método HTTP: POST
- Parámetros
  - **"maxCandidates".** Máximo número de posibles descripciones a devolver (1 por defecto)
  - **"Language"**. Una cadena de texto que indica el idioma en el que se devuelve la descripción de la imagen
    - *"en" – Inglés (por defecto)*
    - *"es" - Español.*
    - *"ja" - Japonés.*
    - *"pt" - Portugués.*
    - *"zh" - Chino*

5.3.3 Detección de objetos en una imagen

La detección de objetos es *similar* al etiquetado, pero en este caso la API devuelve las coordenadas del rectángulo delimitador para cada etiqueta aplicada (ver anexo 3, punto 3), además de devolver un número menor de elementos.

Esta funcionalidad devuelve el mismo resultado que si se emplea el parámetro "*objects*" de la de análisis de imagen (el punto anterior).

Tabla 4: Formas de invocar a la funcionalidad de detección de objetos en las imágenes

SDK	detect_objects_in_stream(image, custom_headers=None, raw=False, callback=None, **operation_config)
API	https://{endpoint}/vision/v3.1/detect

- Método HTTP: POST

5.3.4 Obtención del área de interés de una imagen

Analiza el contenido de una imagen para devolver las coordenadas del área de la imagen que considera que es de mayor interés (ver anexo 3, punto 4).

Tabla 5: Formas de invocar a la funcionalidad de obtención del área de interés de una imagen

SDK	get_area_of_interest(url, custom_headers=None, raw=False, **operation_config)
API	https://{endpoint}/vision/v3.1/areaOfInterest

- Método HTTP: POST

5.3.5 Obtención de la miniatura de una imagen

Computer Vision permite generar miniaturas de una imagen con altura y ancho especificado. Por defecto identifica un área de interés (región of interest, ROI) y genera una miniatura relativa a esa área (la llamada a Computer Vision devuelve el archivo binario correspondiente a la miniatura generada) (ver anexo 3, punto 5).

Tabla 6: Formas de invocar a la funcionalidad de generación de miniaturas de una imagen

SDK	get_area_of_interest_in_stream(image, custom_headers=None, raw=False, callback=None, **operation_config)
API	https://{endpoint}/vision/v3.1/generateThumbnail[?width][&height][&smartCropping]

- Método HTTP: POST
- Parámetros

- **"Width"**. Número entre 1 y 1024 que corresponde a la anchura (se recomienda un mínimo de 50).
- **"Height"**. Número entre 1 y 1024 que corresponde a la altura (se recomienda un mínimo de 50).
- **"SmartCropping"**. Valor booleano ("*True" / "False"*) para permitir el recorte inteligente.

5.3.6 Reconocimiento óptico de caracteres (OCR) en una imagen

Computer Vision incluye funcionalidades de reconocimiento óptico de caracteres (OCR), permitiendo extraer texto impreso y texto manuscrito de imágenes y de documentos (ver anexo 3, punto 6).

Tabla 7: Formas de invocar a la funcionalidad de OCR en la imagen

SDK	`recognize_printed_text_in_stream(image, detect_orientation=True, language='unk', custom_headers=None, raw=False, callback=None, **operation_config)`
API	`https://{endpoint}/vision/v3.1/ocr[?language][&detectOrientation]`

- Método HTTP: POST
- Parámetros:
  - **"Language"**. El código BCP-47 del texto a ser detectado en la imagen. El valor por defecto es "*unk*", lo que hace que el servicio detecte automáticamente el idioma.
    - *"unk" -autodetectar*
    - *"zh-Hans" - Chino simplificado*
    - *"zh-Hant" - Chino tradicional*
    - *"cs" - Checo*
    - *"da" - Danés*
    - *"nl" - Holandés*
    - *"en" - Inglés*
    - *"fi" - Finlandés*
    - *"fr" - Francés*
    - *"de" - Alemán*
    - *"el" - Griego*
    - *"hu" - Húngaro*
    - *"it" - Italiano*
    - *"ja" - Japonés*
    - *"ko" - Coreano*
    - *"nb" - Noruego*
    - *"pl" - Polaco*
    - *"pt" - Portugués*
    - *"ru" - Ruso*
    - *"es" - Español*
    - *"sv" - Sueco*
    - *"tr" - Turco*
    - *"ar" - Árabe*
    - *"ro" - Rumano*
    - *"sr – Cyrl" - Serbio - cirílico*
    - *"sr – Latn" - Serbio - latín*

- *"sk" - Eslovaco*

- **"detectOrientation"**. Valor booleano ("*True*" / "*False*") para permitir detectar automáticamente la orientación del texto.

5.3.7 Lectura de texto en una imagen (Read)

Con esta funcionalidad se ejecuta una operación "*read*", que permite reconocer texto en imágenes con mucho texto, y varios idiomas mezclados (ver anexo 3, punto 9).

Se ejecuta de forma asíncrona, por lo que se complementa con la funcionalidad "*Get read result*" (ver punto 5.3.10), para lo que esta operación "*read*" develve un valor "*Operation-Location*", que incluye la URL con un identificador[21] que debe usarse con el segundo paso (la citada funcionalidad "*Get read result*").

El tiempo para que este "*read*" devuelva el resultado de la lectura de la imagen depende del volumen de texto.

La API reconoce textos en idioma ingles, francés, español, holandés, alemán, italiano y portugués (para texto escrito a mano solamente reconoce el inglés).

Los formatos susceptibles de ser analizados son diferentes de las del resto de funcionalidades: JPEG, PNG, BMP, PDF and TIFF.

Tabla 8:Formas de invocar a la funcionalidad lectura (Read) de texto en una imagen

SDK	`read_in_stream(image, language='en', custom_headers=None, raw=False, callback=None, **operation_config)`
API	`https://{endpoint}/vision/v3.1/read/analyze[?language]`

- Método HTTP: POST
- Parámetros:
  - **"Language"**. Una cadena de texto que indica el idioma en el que se devuelve la descripción de la imagen. Para texto escrito se reconocen siete idiomas, para texto escrito a mano solo inglés. Como parámetro solo puede pasarse un único idioma para el caso de que se desee procesar solamente en él, aunque si no se indica ningún idioma se realiza un análisis que incluye todos los idiomas soportados.
    - *"en" – inglés*
    - *"nl" – holandés*
    - *"fr" – francés*
    - *"de" – alemán*
    - *"it" - italiano*
    - *"pt" – portugués*
    - *"es" - español*

5.3.8 Reconocimiento del contenido de un modelo específico en una imagen

Esta operación reconoce el contenido de la imagen aplicando un modelo de dominio específico (ver anexo 3, punto 7).

[21] El valor es del tipo: https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/vision/v3.1/read/analyzeResults/921f6c2b-6a5a-4d32-ab91-b70b44b91f07

La lista actualizada de modelos específicos se puede conseguir con gracias a la lista de modelos específicos (ver 5.3.11). Actualmente existen dos: "*celebrities*" y "*landmarks*", que son los posibles valores del parámetro "*model*" de la llamada a la API).

El resultado es el mismo que se obtiene con la correspondiente petición empleando el parámetro "details" con la funcionalidad de análisis de una imagen (ver punto 5.3.1)

Tabla 9: Formas de invocar a la funcionalidad de reconocimiento de modelos específicos

SDK	`analyze_image_by_domain_in_stream(model, image, language='en', custom_headers=None, raw=False, callback=None, **operation_config)`
API	`https://{endpoint}/vision/v3.1/models/{model}/analyze[?language]`

- Método HTTP: POST
- Parámetros:
  - **"Language"**. Idioma en el que se devuelven las etiquetas
    - *"en" – Inglés (por defecto)*
    - *"es" - Español.*
    - *"ja" - Japonés.*
    - *"pt" - Portugués.*
    - *"zh" - Chino*

5.3.9 Etiquetado de elementos de una imagen

Computer Vision genera una lista de palabras (etiquetas) que caracterizan de alguna forma la imagen en base a objetos, seres vivos, decorados o acciones (ver anexo 3, punto 8).

No obstante, debe evitarse el error de confundir las etiquetas ("*tags*") que se encuentran en la salida JSON de la funcionalidad de descripción de una imagen, que sirven de base para la elaboración de las descripciones, de las etiquetas ("*tags*") a las que se refiere este punto, por la confusión que pueden crear que esos dos elementos se llamen del mismo modo ("*tags*"), incluso con los objetos, que, como concepto, cuesta trabajo diferenciar de las etiquetas.

Tabla 10: Formas de invocar a la funcionalidad de etiquetar elementos de una imagen

SDK	`tag_image_in_stream(image, language='en', custom_headers=None, raw=False, callback=None, **operation_config)`
API	`https://{endpoint}/vision/v3.1/tag[?language]`

- Parámetros
  - **"Language"**. Idioma en el que se devuelven las etiquetas
    - *"en" – Inglés (por defecto)*
    - *"es" - Español.*
    - *"ja" - Japonés.*
    - *"pt" - Portugués.*
    - *"zh" – Chino.*

5.3.10 Consecución de los datos obtenidos con lectura de texto "read"

Funcionalidad que recupera el estado y el resultado OCR de la operación "read" expuesta en el punto 5.3.7 (ver anexo 3, punto 9).

Tabla 11: Formas de invocar a la funcionalidad que recuepra datos obtenidos con funcionalidad "Read"

SDK	`get_read_result(operation_id, custom_headers=None, raw=False, **operation_config)`
API	`https://{endpoint}/vision/v3.1/read/analyzeResults/{operationId}`

- Método HTTP: GET
- Parámetros:
  - operationId. URL que incluye el identificador de la operación Read de la que se solicita el resultado[21].

Además del texto reconocido, la salida *JSON de respuesta* incluye campos relevantes para el código de los programas que invocan a esta funcionalidad pues permiten chequear la situación en la que se encuentra la petición POST a la que se refiere este GET:

- "*status*". Situación de la operación read. Valores posibles:
  - *"notStarted"*: La operación no se ha empezado
  - *"running"*: La operación se está ejecutando
  - *"failed"*: La operación ha fallado
  - *"succeeded"*: La operación ha finalizado correctamente
- "*analyzeResult*". Resultado de la operación "read"

5.3.11 Listado de modelos de dominio específicos

Esta operación devuelve los valores disponibles para los "modelos de dominio específicos" que pueden usarse con Computer Vision que pueden usarse en las funcionalidades expuesta en 5.3.1 y 5.3.8 (ver anexo 3, punto 10).

Actualmente la API admite dos valores:

- "celebrities" (para reconocimiento de personas famosas)
- "landmark" (para reconocimiento de obras o lugares famosos)

Tabla 12:Formas de invocar a la funcionalidad que aporta la lista de dominios específicos disponibles

SDK	`list_models(custom_headers=None, raw=False, **operation_config)`
API	`https://{endpoint}/vision/v3.1/models`

- Método HTTP: GET

Las imágenes que se deseen analizar con Computer Vision deben cumplir los siguientes requisitos:

- Debe estar en formato JPEG, PNG, GIF o BMP.
- El tamaño de archivo debe ser inferior a 4 MB.
- Las dimensiones deben ser mayores que 50 x 50 píxeles
  - Para la API Read, las dimensiones de la imagen deben estar entre 50 x 50 y 10 000 x 10 000 píxeles.
  - Para las funciones OCR las dimensiones deben estar entre 50 x 50 y 4200 x 4200 pixels y la imagen no puede ser tener más de 10 megapixels.

5.3.12 Llamadas directas a la API de Computer Vision

Las llamadas directas a la API se pueden realizar mediante métodos GET o POST:

- Llamadas POST[22]:
  - Analizar una imagen (ver 5.3.1)
  - Describir una imagen (ver 5.3.2)
  - Detectar objetos en una imagen (ver 5.3.3)
  - Obtener área de interés de una imagen (ver 5.3.4)
  - Obtener una miniatura de una imagen (ver 5.3.5)
  - Reconocimiento óptico de caracteres (OCR) en una imagen (ver 5.3.6)
  - Lectura de texto en una imagen (Read) (ver 5.3.7)
  - Reconocer contenido de un modelo específico en una imagen (ver 5.3.8)
  - Etiquetar elementos de una imagen Imagen de etiqueta (ver 5.3.9)
- Llamadas GET[23]
  - Conseguir datos obtenidos con lectura de texto “read” (ver 5.3.10)
  - Listar modelos de dominio específicos (ver 5.3.11)

Las solicitudes POST y GET, además de la URL que corresponda en función de la capacidad solicitada y de los parámetros (“*params*”) que se seleccionen, deben incluir:

- Encabezados (“*headers*”). Las opciones posibles son:
  - “Content-Type”. Tipo de medio del cuerpo enviado a la API:
    - *“application/json” para una URL de imagen*
    - *“application/octet-stream” o “multipart/form-data” para datos de una imagen binaria*
  - “Ocp-Apim-Subscription-Key”. La clave de suscripción que proporciona acceso a la API.

[22] El método POST no escribe los parámetros en la dirección URL, sino que los adjunta al encabezado HTTP, lo que aporta discreción. Es el empleado normalmente cuando se tienen que enviar al servidor web paquetes grandes de datos, como imágenes o datos de formulario de carácter privado [18]

[23] EL método GET existía ya en los inicios de la *world wide web* para solicitar recursos del servidor web, como un archivo HTML. Con este método, los datos que se envían al servidor se escriben en la misma dirección URL. [18]

- Cuerpo de la solicitud. Los métodos de entrada admitidos son:
  - Binario de imagen sin procesar ((“*data*”)
  - URL de imagen (“*json*”).

A continuación, se muestran ejemplos de código usados para analizar la Ilustración 13,

Ilustración 13: Imagen usada para las descripciones mediante llamada directa a API

En la Ilustración 14 puede verse la forma de realizar llamadas directas a la API usando PYTHON para conseguir la descripción de una versión de la imagen accesible en una URL de Internet.

```
import requests

AZURE_mi_clave = "x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x"
AZURE_mi _URL = "https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/"

URL_descripcion = AZURE_mi_URL + "vision/v3.1/describe"
cabeceras = {'Ocp-Apim-Subscription-Key': AZURE_mi_clave, 'Content-Type': 'application/json'}
parametros = {'maxCandidates': '5', 'language': 'es'}

imagen = {'url': 'https://historia.nationalgeographic.com.es/medio/2018/03/01/coliseo-romano_16022ed4_1280x720.jpg'}
respuesta = requests.post(URL_descripcion, headers=cabeceras, params=parametros, json=imagen)

for descripcion_generada in respuesta.json()["description"]["captions"]:
    print("La imagen corresponde a {} (con una certeza del {:.2f}%)"
          .format(descripcion_generada["text"], descripcion_generada["confidence"] * 100))
```

Ilustración 14: Ejemplo de llamada directa a API para describir imagen disponible en una URL de Internet

La Ilustración 15 muestra el código para hacer la misma tarea anterior, pero en este caso sobre una copia de la imagen almacenada en el equipo local.

```
import requests

AZURE_mi_clave = "x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x"
AZURE_mi_URL = "https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/"

URL_descripcion = AZURE_mi_URL + "vision/v3.1/describe"
cabeceras = {'Ocp-Apim-Subscription-Key': AZURE_mi_clave, 'Content-Type': 'application/octet-stream'}
parametros = {'maxCandidates': '5', 'language': 'es'}

imagen = open("coliseo-romano.jpg", "rb").read()
respuesta = requests.post(URL_descripcion, headers=cabeceras, params=parametros, data=imagen)

for descripcion_generada in respuesta.json()["description"]["captions"]:
    print("La imagen corresponde a {} (con una certeza del {:.2f}%)"
          .format(descripcion_generada["text"], descripcion_generada["confidence"] * 100))
```

Ilustración 15: Ejemplo de llamada directa a API para describir imagen disponible en el equipo local

Como puede verse, la diferencia principal en el código de los dos programas es el contenido de la variable "*imagen*", que en un caso es un fichero y en el otro una URL (ambos se refieren a la misma imagen). Como consecuencia de esa diferencia es necesario ajustar el valor de "Content-type" (*'application/octet-stream'* para la imagen local y *'application/json'* para la imagen remota) y el parámetro con el que se remite la imagen mediante el POST ("*data*" para la imagen local y "*json*" para la imagen remota).

Aparte de eso, es especialmente relevante el contenido de las variables "AZURE_mi_clave" y "AZURE_mi _URL", donde se almacenan las credenciales de acceso a Computer Vision (ver final del punto 6.2.2).

Para ejecutar el método POST se ha optado por emplear un objeto *request* de la librería *requests.*

La Ilustración 16 muestra la salida en consola de ambos programas, que, lógicamente, es la misma para los dos, pues los resultados son independientes del lugar donde se almacene el fichero.

```
La imagen corresponde a Coliseo de un edificio (con una certeza del 66.52%)
La imagen corresponde a edificio de piedra con Coliseo de fondo (con una certeza del 66.42%)
La imagen corresponde a Coliseo de un edificio de ladrillo (con una certeza del 63.25%)
La imagen corresponde a torre de Coliseo (con una certeza del 63.15%)
La imagen corresponde a un torre de Coliseo (con una certeza del 63.05%)

Process finished with exit code 0
```

Ilustración 16: Salida por consola de programas que llaman directamente a la API

La respuesta devuelta por Computer Vision siempre se da en formato JSON, excepto para la generación de miniaturas, lógicamente. En la Ilustración 17 se muestra el contenido completo de la salida JSON devuelta por la API para las descripciones solicitadas con los programas anteriores (se han eliminado algunas "tags" para conseguir mayor claridad, evitando una lista de etiquetas demasiado larga que no aportaría ningún valor añadido a los efectos de este trabajo).

```
{
        "description": {
                "tags": [
                        "pasto",
                        "edificio",
                        "exterior",
                        "frente",
                        "grande",
                        "viejo",
                        "tabla",
                        ****

                ],
                "captions": [
                        {
                                "text": "Coliseo de un edificio",
                                "confidence": 0.6651820709352974
                        },
                        {
                                "text": "edificio de piedra con Coliseo de fondo",
                                "confidence": 0.6641820709352974
                        },
                        {
                                "text": "Coliseo de un edificio de ladrillo",
                                "confidence": 0.6324514370969317
                        },
                        {
                                "text": "torre de Coliseo",
                                "confidence": 0.6314514370969317
                        },
                        {
                                "text": "un torre de Coliseo",
                                "confidence": 0.6304514370969317
                        }
                ]
        },
        "requestId": "1005a0f2-bee9-42af-80be-7b48c4db43a1",
        "metadata": {
                "height": 720,
                "width": 1280,
                "format": "Jpeg"
        }
}
```


Ilustración 17: Salida JSON devuelta a las llamadas directas a la API

5.3.13 Llamadas a la API de Computer Vision a través del SDK para PYTHON

A continuación, se presenta el código de dos programas (Ilustración 18 e Ilustración 19) que emplean el SDK de PYTHON para realizar las mismas funciones que los programas mostrados en el punto anterior, para facilitar la comparación de los procedimientos que usan llamadas a la API a través del SDK de PYTHON y los que lo hacen con llamadas directas a esa API, siendo reseñable que, como cabía esperar, los resultados obtenidos son exactamente los mismos con ambos procedimientos (Ilustración 20).

Con respecto a estos programas, la primer diferencia que se puede apreciar es que aquí no se emplea la librería "*requests*", porque los métodos del SDK (*describe_image_in_stream* y *describe_image* en estos ejemplos) resuelven por sí mismos la ejecución del POST.

Por otra parte, se necesitan dos librerías que se usan para crear el objeto que contiene las credenciales de uso de Computer Vision y que no eran necesarias cuando se llamaba

directamente a la API. En este caso, almacenamos en variables las credenciales que obtuvimos del portal de Azure (las mencionadas al final del punto 6.2.2), para después crear un objeto *ComputerVisionClient*, usando para ello:

- El punto de conexión (el dato presentado como "extremo" en el portal de Azure)
- El objeto *CognitiveServicesCredentials* creado con una de las dos claves que nos ofrece el portal de Azure (preferiblemente la clave 1 por las razones expuestas anteriormente). Por razones de seguridad, en la imagen siguiente no se muestra la clave real correspondiente al recurso empleado para este desarrollo.

Esta instancia de *ComputerVisionClient* se almacena en Usuario_*Computer*_Vision para su uso posterior.

Por último, nótese que es diferente el método que se emplea para tratar la imagen local y el que se usa para la imagen remota, diferenciándose en el sufijo "_in_stream" del final de su nombre. Todos los métodos mencionados en el punto 5.3 siguen el mismo patrón, de forma que cada método cuyo nombre finaliza en "_in_stream" proporciona la funcionalidad para ficheros locales mientras los que no tienen ese sufijo la proporcionan para ficheros remotos.

```
from azure.cognitiveservices.vision.computervision import ComputerVisionClient
from msrest.authentication import CognitiveServicesCredentials

AZURE_mi_clave = "x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x"
AZURE_mi_URL = "https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/"

Usuario_Computer_Vision = ComputerVisionClient(AZURE_mi_URL, CognitiveServicesCredentials(AZURE_mi_clave))

URL_imagen = "https://historia.nationalgeographic.com.es/medio/2018/03/01/coliseo-romano_16022ed4_1280x720.jpg"

respuesta = Usuario_Computer_Vision.describe_image(URL_imagen, max_candidates=5, language="es")

for escripcion_generada in respuesta.captions:
    print("La imagen corresponde a {} (con una certeza del {:.2f}%)".
          format(escripcion_generada.text, escripcion_generada.confidence * 100))
```

Ilustración 19: Ejemplo de llamada a la API mediante SDK para describir imagen disponible en una URL de Internet

```
from azure.cognitiveservices.vision.computervision import ComputerVisionClient
from msrest.authentication import CognitiveServicesCredentials

AZURE_mi_clave = "x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x"
AZURE_mi_URL = "https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/"

Usuario_Computer_Vision = ComputerVisionClient(AZURE_mi_URL, CognitiveServicesCredentials(AZURE_mi_clave))

Imagen = open("coliseo-romano.jpg", "rb")
respuesta = Usuario_Computer_Vision.describe_image_in_stream(Imagen, max_candidates=5, language="es")

for escripcion_generada in respuesta.captions:
    print("La imagen corresponde a {} (con una certeza del {:.2f}%)".
          format(escripcion_generada.text, escripcion_generada.confidence * 100))
```

Ilustración 18: Ejemplo de llamada a la API mediante SDK para describir imagen disponible en el equipo local

```
La imagen corresponde a Coliseo de un edificio (con una certeza del 66.52%)
La imagen corresponde a edificio de piedra con Coliseo de fondo (con una certeza del 66.42%)
La imagen corresponde a Coliseo de un edificio de ladrillo (con una certeza del 63.25%)
La imagen corresponde a torre de Coliseo (con una certeza del 63.15%)
La imagen corresponde a un torre de Coliseo (con una certeza del 63.05%)

Process finished with exit code 0
```

Ilustración 20: Salida por consola de programas que llaman a la API a través del SDK

5.4 Comparación de procedimientos usando SDK y llamadas directas a la API

Como puede apreciarse en los ejemplos anteriores, a la hora de programar, las diferencias en el código son mínimas si se opta por emplear el SDK o por llamadas directas a la API, pero ante la necesidad de optar por una de las dos, **se ha decidido trabajar con las llamadas directas a la API** por la claridad que aportan las salidas JSON que devuelve *Computer Vision* con esa opción además de que, lógicamente, debe ser la opción más eficiente, al evitar el paso intermedio que supone el uso del SDK.

6 Entorno de desarrollo para el proyecto

6.1 Requisitos HW/SW

Para el desarrollo de la aplicación de identificación de coches empleando Azure (Computer Vision) se ha empleado:

- Ordenador: MacBook Air 9,1
  - Sistema Operativo: MacOS Big Sur 11.2.2
- Python 3.9.2.
- Entorno de Desarrollo Integrado (IDE) de Python: PyCharm 2020.3.3.

Por otra parte, como requisitos previos al desarrollo de una aplicación que emplee las capacidades de Azure (Computer Vision), es necesario:

- Crear una cuenta de Microsoft Azure
- En Azure, crear recurso de Cumputer Vision
- Proceder a la instalación que corresponda para el lenguaje en el que se vaya a hacer el desarrollo, en nuestro caso PYTHON.

6.2 Entorno de Azure

Para poder emplear las capacidades de Computer Vision mediante llamadas a su API es necesario disponer de una cuenta de Azure, para lo que deben seguirse los pasos que se relacionan a continuación.

6.2.1 Creación de una cuenta de Microsoft Azure

El primer paso, obviamente, es acceder al portal de Microsoft Azure (Ilustración 21).

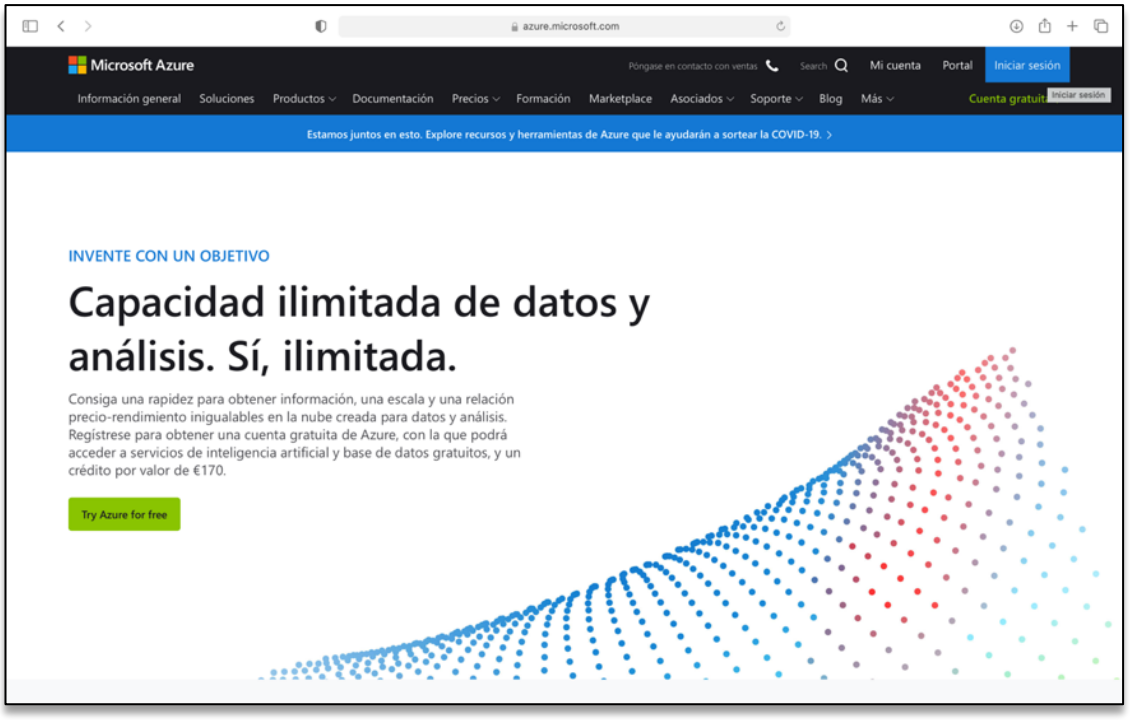


Ilustración 21: Página de acceso a Azure

Una vez en la página de acceso a Microsoft Azure debemos iniciar sesión con una cuenta de Microsoft creada previamente. En nuestro caso, para crear una cuenta de Azure para estudiantes, accederemos mediante la dirección de correo de alumno de la UAH (Ilustración 22). Una vez introducida la dirección de correo (aún no se pide la contraseña), el Sistema nos redirige (Ilustración 23) a una página de la UAH (Ilustración 24) donde se validan las credenciales como alumno.

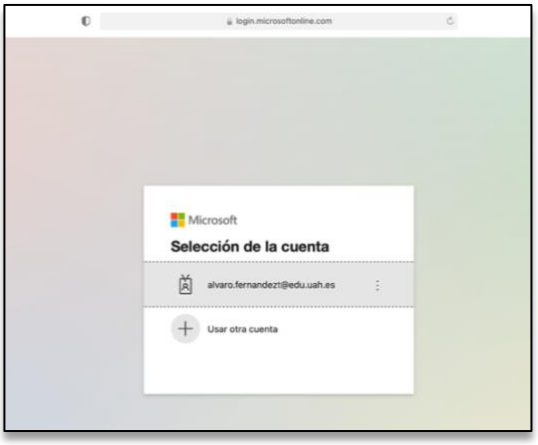


Ilustración 22: Selección de cuenta para inicio de sesión

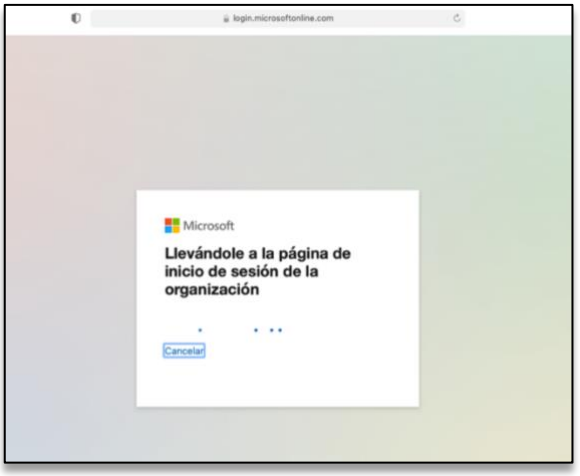


Ilustración 23: Redireccionamiento a página de la UAH

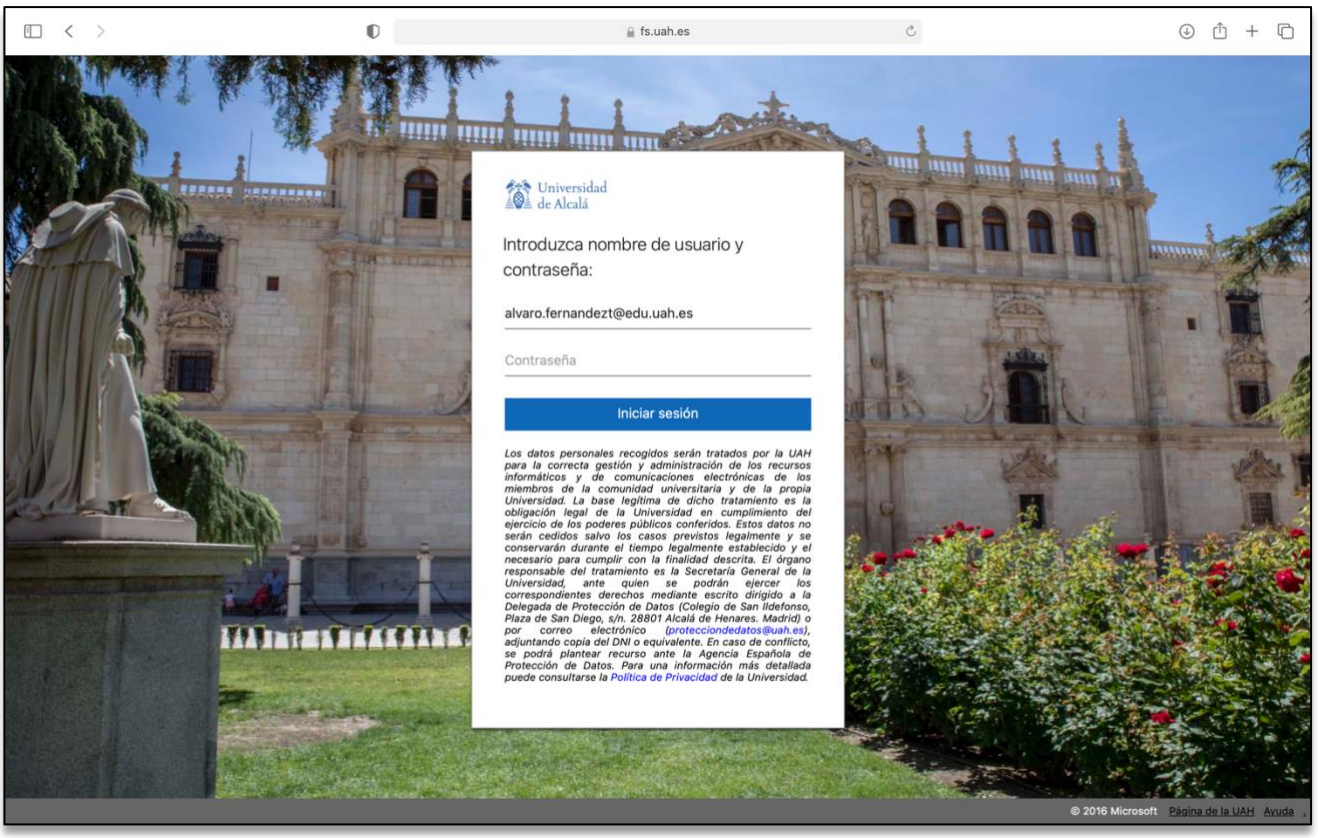


Ilustración 24: Página de la UAH para comprobar credenciales

Una vez introducidas las credenciales y validadas se nos redirige a Azure, donde procedemos a crear la cuenta gratuita (opción “Empiece gratis” en la Ilustración 25).

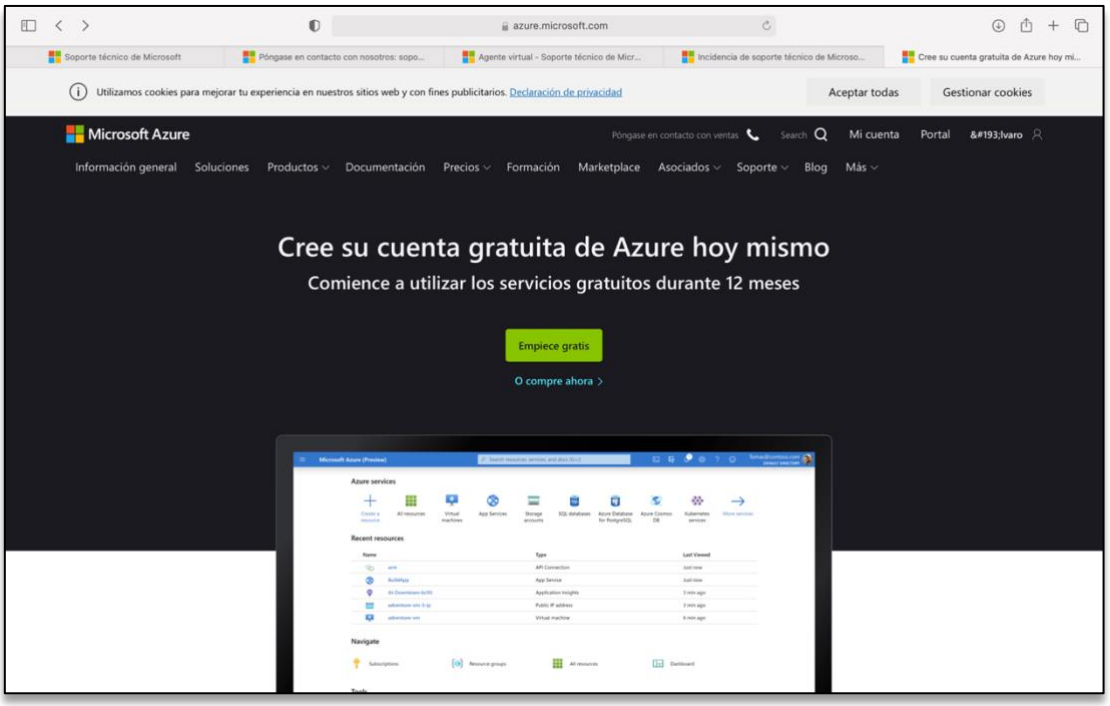


Ilustración 25: Azure - creación de cuenta gratuita

6.2.2 Creación de un recurso “Computer Vision”

Una vez creada la cuenta de Azure, el Sistema nos presentará la página de la Ilustración 26:

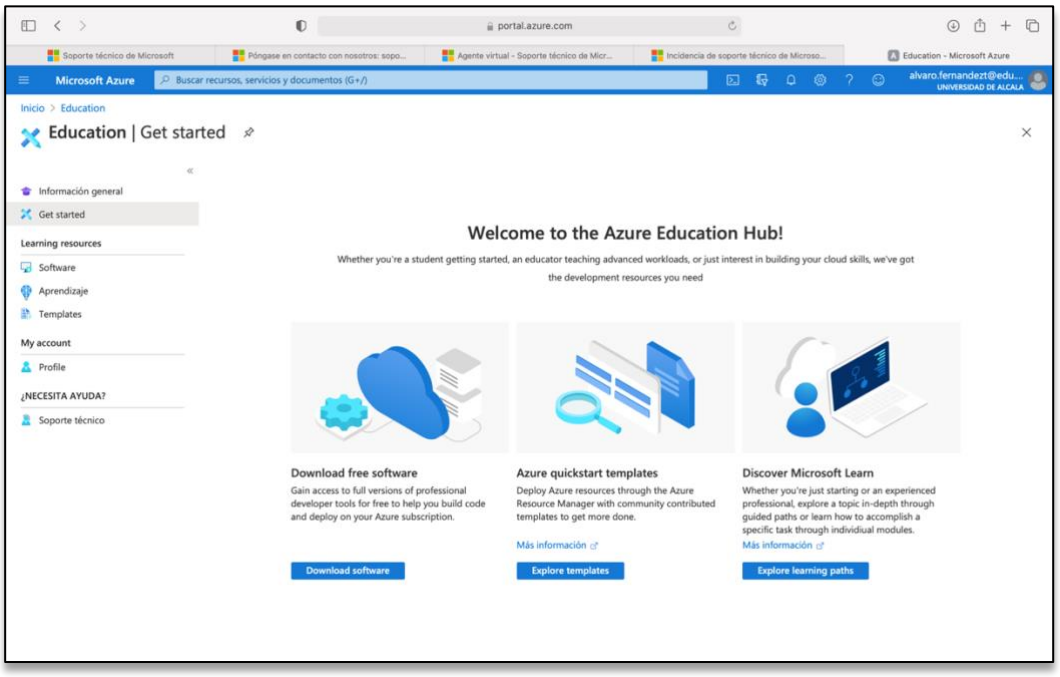


Ilustración 26: Página de bienvenida a estudiantes

En el menú de la parte superior del lado izquierdo, abandonamos el sitio “*Education*” y nos situamos en “*Inicio*” (Ilustración 27), donde nos encontramos con la posibilidad de acceder a las diferentes opciones de Azure.

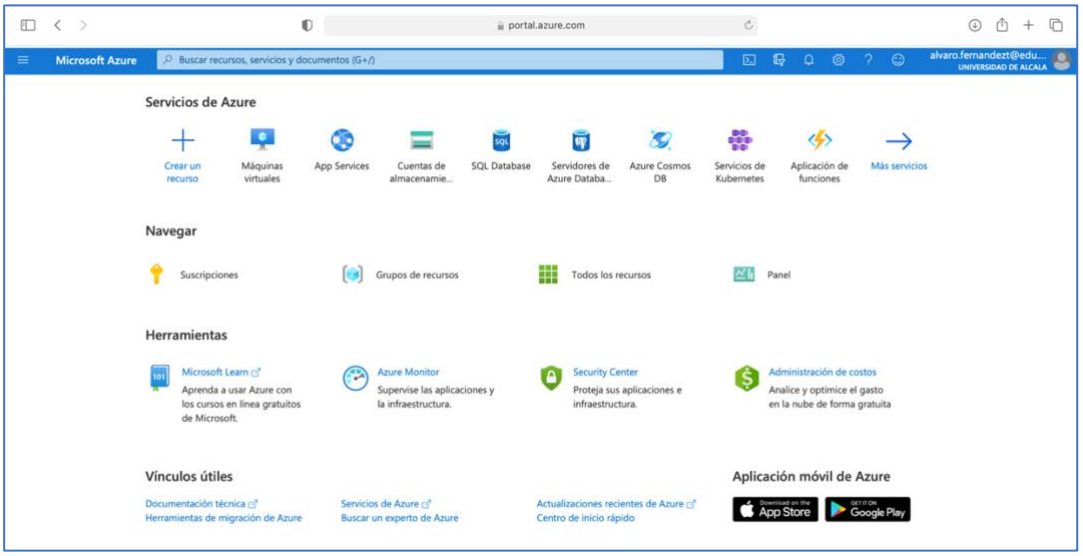


Ilustración 27: Portal de Azure (https://portal.azure.com/#home)

De entre las opciones disponibles, nos interesa la de "*Crear un recurso*", mediante la cual accedemos a la pantalla de la Ilustración 28, en la que escribiremos el nombre del recurso que queremos añadir, en nuestro caso "*Computer Vision*".

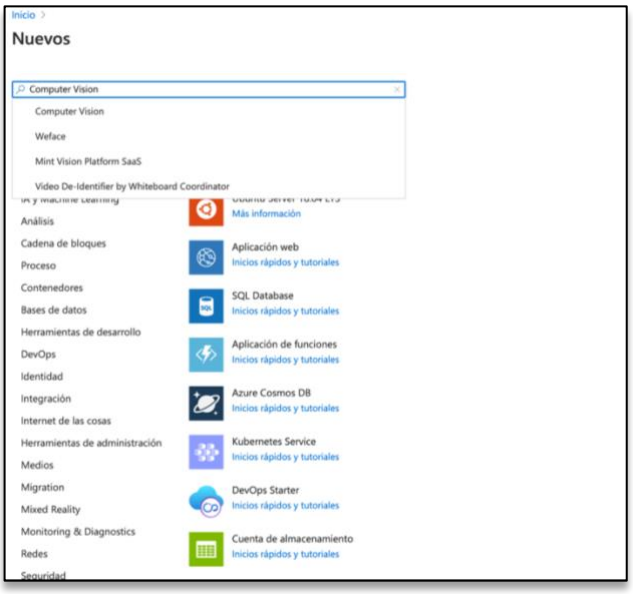


Ilustración 28: Ventana de creación de nuevo recurso

Al seleccionar "*Computer Vision*" nos aparecerá la pantalla de la Ilustración 29, en la que se da información general de ese servicio, así como la opción de crear el recurso, haciendo clic en el correspondiente botón.

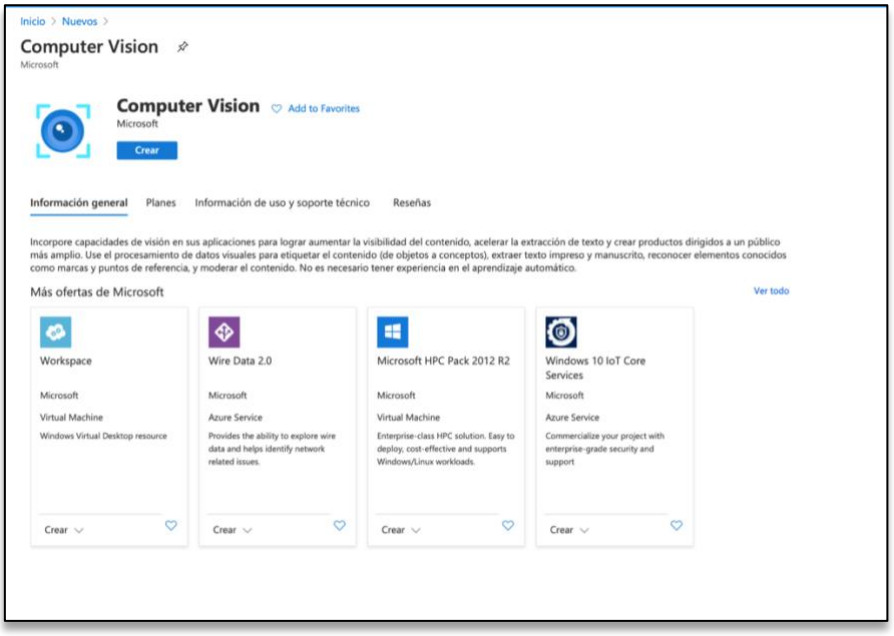


Ilustración 29: Página de inicio de creación del recurso "Computer Vision"

Después de hacer clic en "*Crear*" se nos presenta la pantalla de la Ilustración 30, en la que debemos rellenar una serie de datos necesarios para configurar el nuevo recurso.

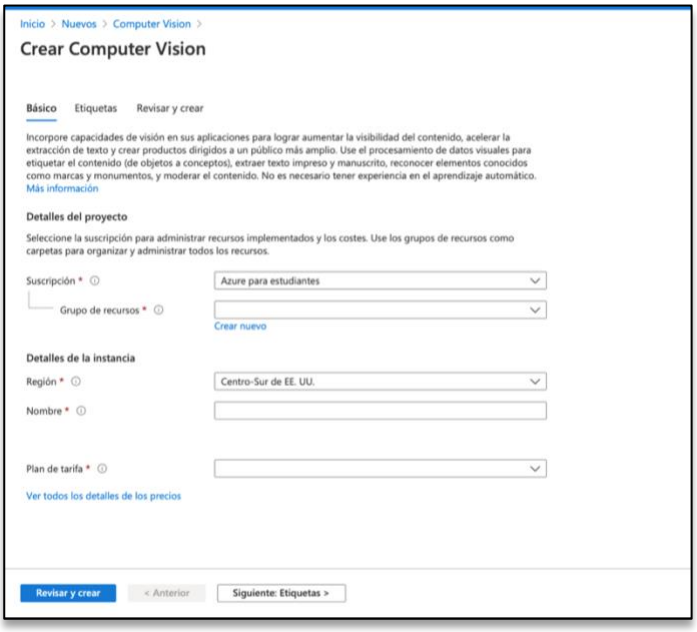


Ilustración 30: Datos iniciales para creación de recurso de "Computer Vision"

La arquitectura de *Azure* obliga a que cualquier recurso se incluya en un *grupo de recursos*, por lo que deberemos crear un *grupo de recursos* (un contenedor que almacena los recursos relacionados con una solución de Azure, incluyendo metadatos acerca de esos recursos) en el que ubicar nuestro nuevo *recurso* "*Computer Vision*".

En nuestro caso vamos a crear un grupo de recursos usando como nombre uno que haga mención del proyecto y al nombre del desarrollador*, como se ve en la* Ilustración 31*.*

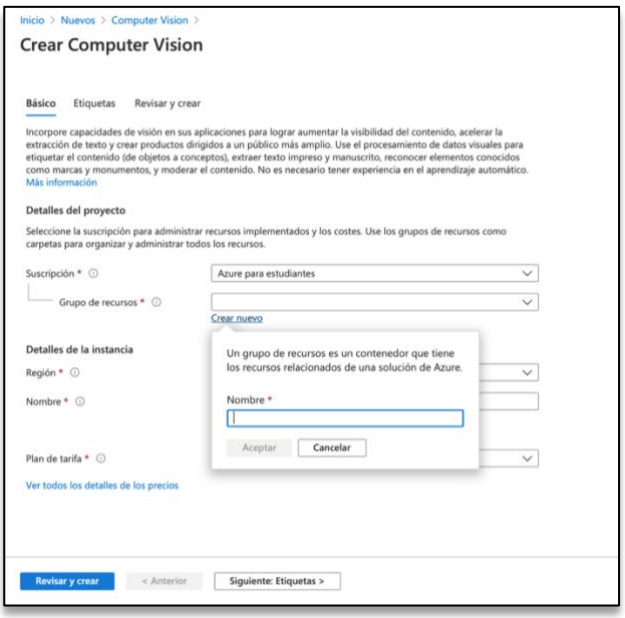


Ilustración 31: Creación de grupo de recursos

Con respecto a los detalles de la instancia:

- Mantenemos la región de "*Centro-Sur de EE. UU.*" para asegurar que no se presenten problemas derivados del tipo de cuenta de estudiante.
- Asignamos el nombre al recurso "Computer Vision" ("*Computer-Vision-Alvaro-FTG*")

Acabamos seleccionando la tarifa F0 (gratuita), como se ve en la Ilustración 32 y procedemos a "revisar y crear" para que el sistema proceda a la validación de los datos que hemos proporcionado.

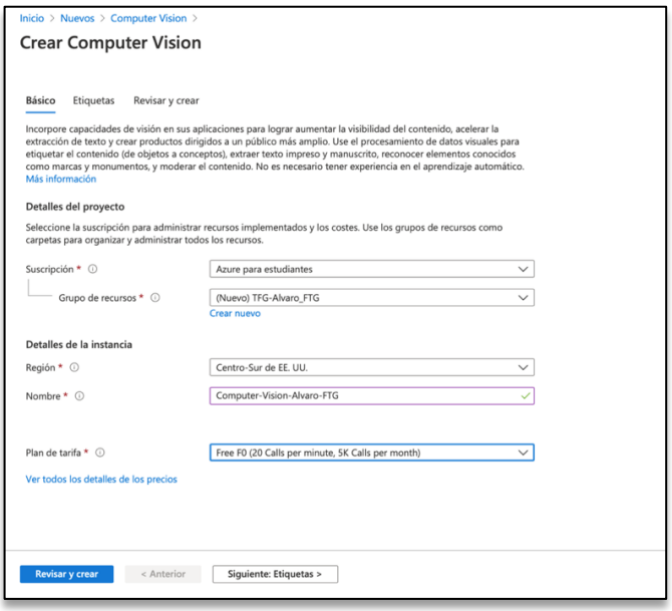


Ilustración 32: Datos finales para creación del recurso de "Computer Vision"

Una vez validados los datos, pulsamos "crear" (Ilustración 33) para que se cree nuestro recurso de "Computer Vision".

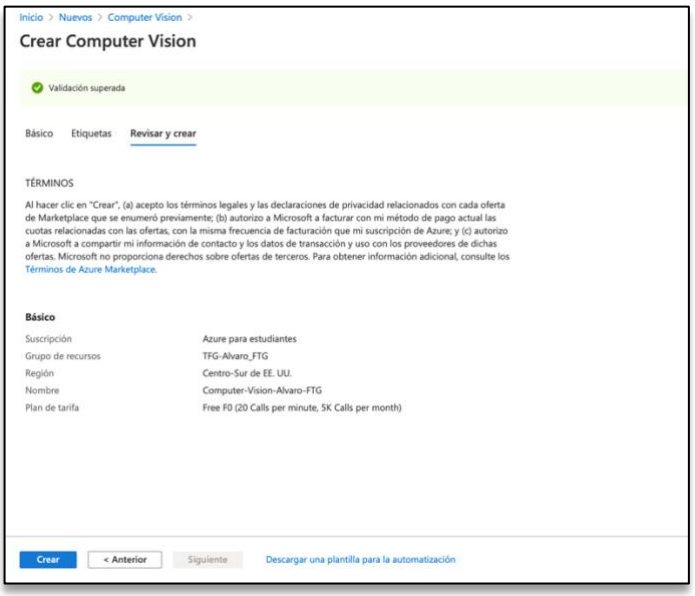


Ilustración 33: Creación del recurso de "Computer Vision"

En este momento, una vez creado el nuevo recurso, podemos comprobar cómo ha cambiado la página de inicio que teníamos al principio de este punto, que ahora incluye una nueva sección "*recursos recientes*" (Ilustración 34) en la que, además de la suscripción a *Azure para estudiantes*, vemos el *grupo de recursos* que hemos creado y el *recurso de "Computer Vision"*.

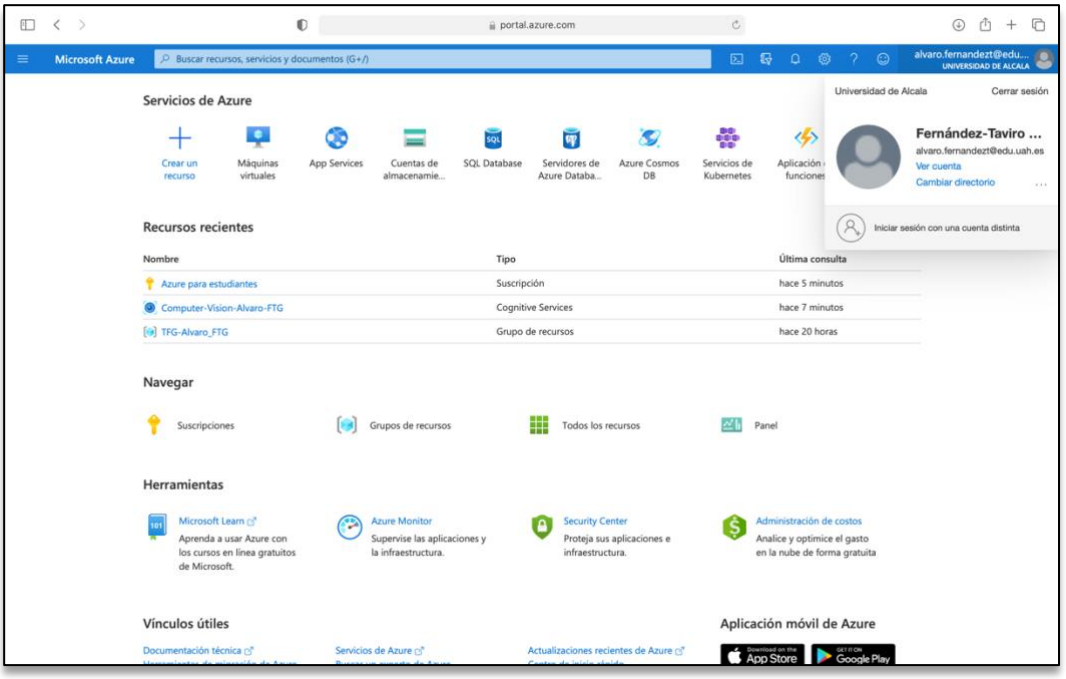


Ilustración 34: Portal de Azure después de crear el recurso de "Computer Vision"

A continuación, accedemos a la información general del nuevo recurso (Ilustración 35), para comprobar sus características, siendo especialmente relevante la URL indicada en el campo "*Extremo*", porque será uno de los dos valores que nos permitirá acceder a las funciones de *Computer Vision* desde el programa que desarrollaremos en *PYTHON*:

https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/

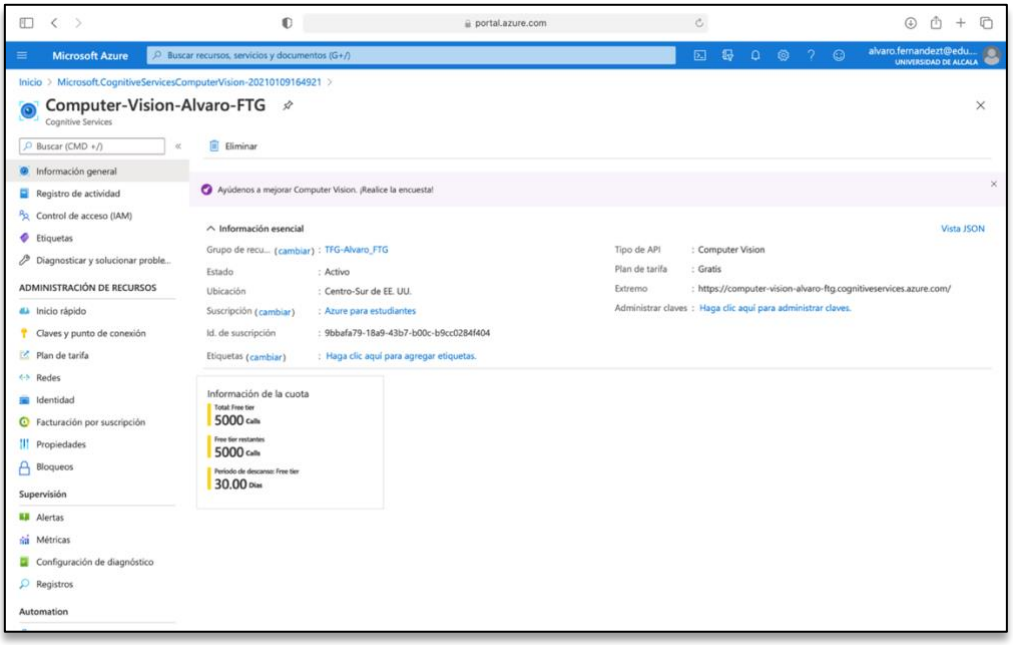


Ilustración 35: Información general del recurso "Computer Vision" creado

Por último, es necesario acceder a la página de "*claves y punto de conexión*" del grupo de "*administración de recursos*" (Ilustración 36) donde, además de la URL mencionada en el punto anterior, podemos encontrar *dos claves* [17] que son necesarias para acceder a las funciones de *Computer Vision*. Se puede utilizar cualquiera de esas dos claves para emplear *Computer Vision* y acceder a *Azure Storage*, aunque *Microsoft* recomienda usar la primera clave y reservar el uso de la segunda clave para cuando se roten las claves, además de regenerarlas periódicamente y no compartirlas.

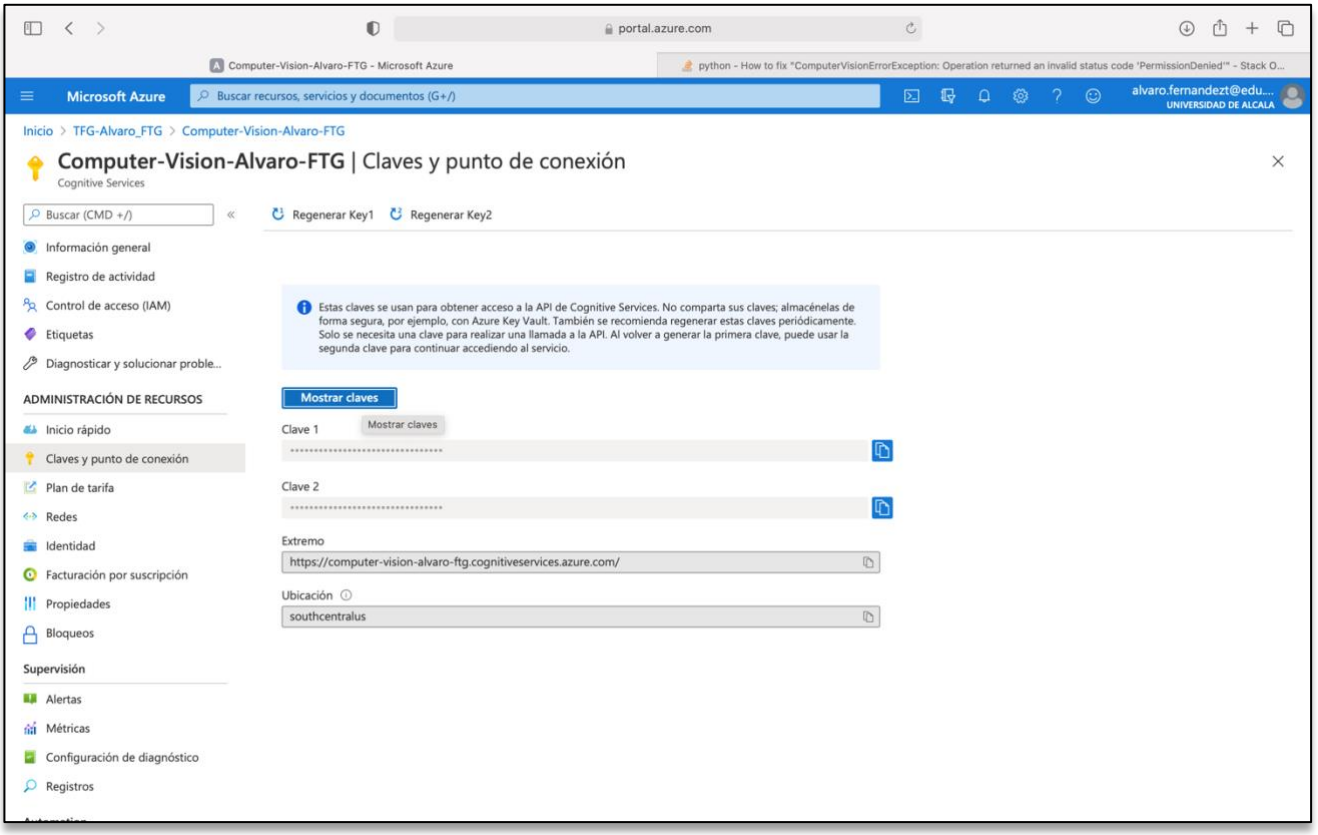


Ilustración 36: Claves y punto de conexión del recurso "Computer Vision"

6.3 Entorno de PYTHON

Aunque el Sistema Operativo MacOS incluye PYTHON 2 preinstalado, para nuestro desarrollo hemos optado por instalar la versión más reciente de PYTHON 3 disponible en la página de descargas de PYTHON[24] (Ilustración 37).

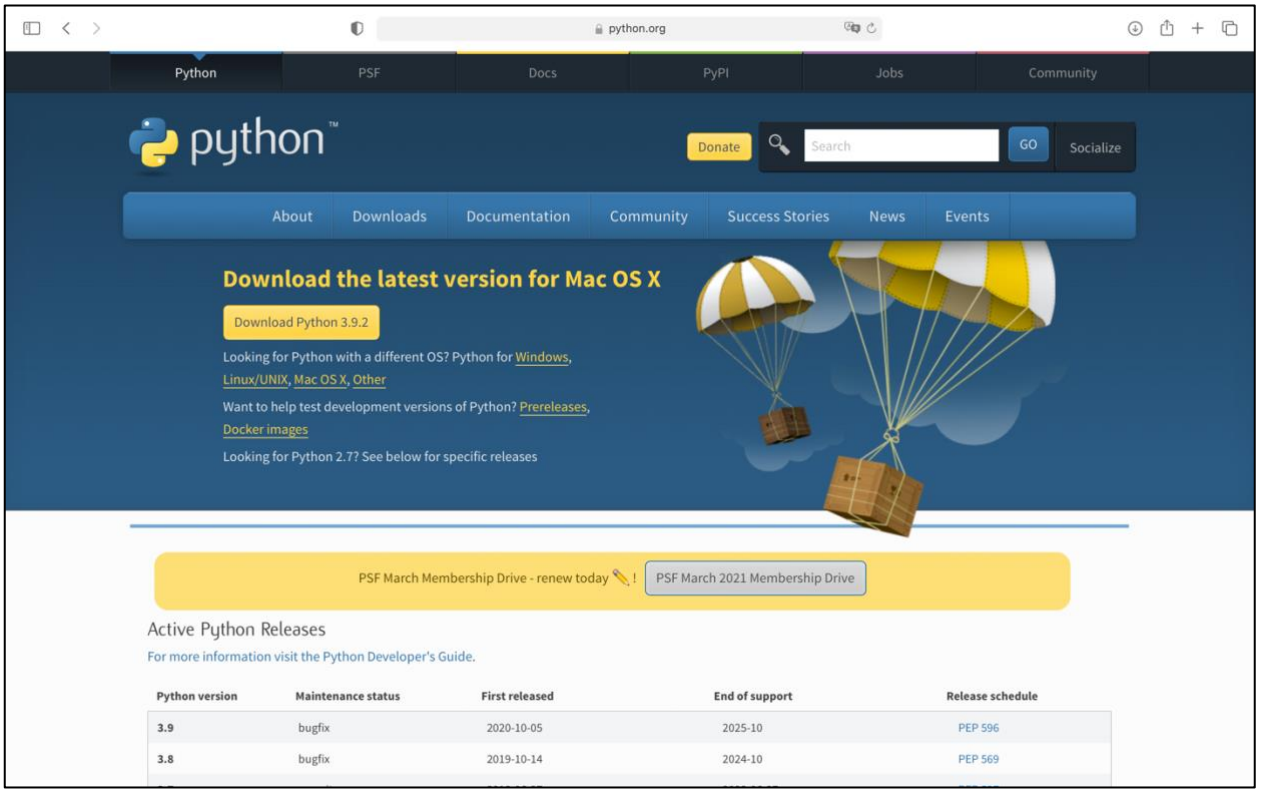


Ilustración 37: Página para descarga de PYTHON 3.9.2.

6.3.1 Bibliotecas

Para instalar las bibliotecas que necesitaremos para nuestro desarrollo, empezaremos por descargar e instalar PIP, un sistema de gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en Python. Para instalarlo, se `puede optar por:

- Hacerlo directamente desde su sitio WEB[25].
- Hacerlo mediante comando de terminal del MacBook:
  - Descargar un programa PYTHON para instalar PIL: ***curl -O https://bootstrap.pypa.io/get-pip.py***
  - Ejecutar la aplicación descargada: ***python get-pip.py***

Una vez instalado PIP procedemos a instalar las librerías que se van a usar para el desarrollo con la finalidad de poder presentar de forma visual y amigable una imagen en la que junto a la fotografía analizada se muestre la descripción generada por Computer Vision y el número de coches detectados en ella:

[24] https://www.python.org/downloads/
[25] https://pypi.org/project/pip/

— Librería MATPLOTLIB[26] (se usará el módulo PYPLOT

- Comando en el terminal: **python3 -m pip install matplotlib**

— Librería PIL (PILLOW)[27]

- Comando en el terminal: **python3 -m pip install pillow**

Por último, es necesario instalar el módulo del servicio *Computer Vision* de *Azure* para *PYTHON*.

- Comando en el terminal: **python3 -m pip install azure-cognitiveservices-vision-computervision**

[26] https://matplotlib.org
[27] https://pypi.org/project/Pillow/

7 Ejemplo práctico del uso de Microsoft Azure (*Computer Vision*)[28]

Llegados a este punto, se estima conveniente recordar lo que se exponía al principio de este documento sobre el objetivo del trabajo:

> *"El objetivo principal de este trabajo es realizar un "tutorial" y desarrollar una aplicación que muestre de forma práctica el funcionamiento de las herramientas de Inteligencia Artificial que ofrece Microsoft Azure para el análisis de imágenes, aplicándolas en la detección de un tipo de objeto determinado…".*

En lo concerniente a la *aplicación que muestre de forma práctica el funcionamiento de las herramientas de Inteligencia Artificial que ofrece Microsoft Azure para el análisis de imágenes, aplicándolas en la detección de un tipo de objeto determinado*, la aplicación que se presenta a continuación se focaliza en detectar los objetos de la imagen, para contar el número de coches[29] que hay en ella. No obstante, además del número de coches detectado, se muestra por consola una lista de varias descripciones de la imagen (cinco) y la lista completa de objetos detectados.

Para hacer el programa más robusto, además de gestionar algunos posibles errores, se ha incluido:

- Una ventana de bienvenida
- Una ventana para seleccionar del fichero a analizar
- Una ventana para presentar el resumen los resultados de salida de forma amigable, incluyendo:
  - El nombre del fichero analizado.
  - Una copia de la imagen analizada.
  - Un texto con la descripción puntuada con mayor porcentaje de certeza.
  - Un texto con el número de coches detectados por Computer Vision en la imagen.

Cabe mencionar que se contactó con un "Agente de Ventas Virtual" (danirey@microsoft.com) que proporcionó las siguientes alternativas para resolver las dudas que pudieran surgir durante el desarrollo:

- Aprendizaje de Azure[30].
- Expertos de la comunidad[31].
- Página para Desarrolladores de Microsoft (MSDN)[32].

La lógica general del programa se presenta en el flujograma de la Ilustración 38.

[28] En el anexo 2 se incluye el código completo del programa.

[29] Aunque se expondrá con más detalle más adelante, se quiere adelantar ya el problema que supone que Azure detecte como objetos con nombres diferentes a distintos tipos de coches, puesto que, al no haber una funcionalidad que devuelva directamente el valor buscado lo deseable habría sido que todos los coches tuvieran una misma etiqueta ("car" o la que procediese) o un conjunto de etiquetas definido y conocido para poder programar el contador. Al no haberse localizado esa taxonomía, el programa contabiliza como coches los tipos de objeto que se han podido conocer ("car", "*taxi*", "*van*", "*hatchback*" y "*race car)*", pero no es descartable que pudieran existir otros tipos en cuyo caso el programa no los contaría.

[30] https://azure.microsoft.com/es-es/developer/students/

[31] https://azure.microsoft.com/es-es/support/community/

[32] https://social.msdn.microsoft.com/Forums/es-ES/home?forum=windowsazureplatform%2Cazuremarketplace%2Cwindowsazureplatformctp

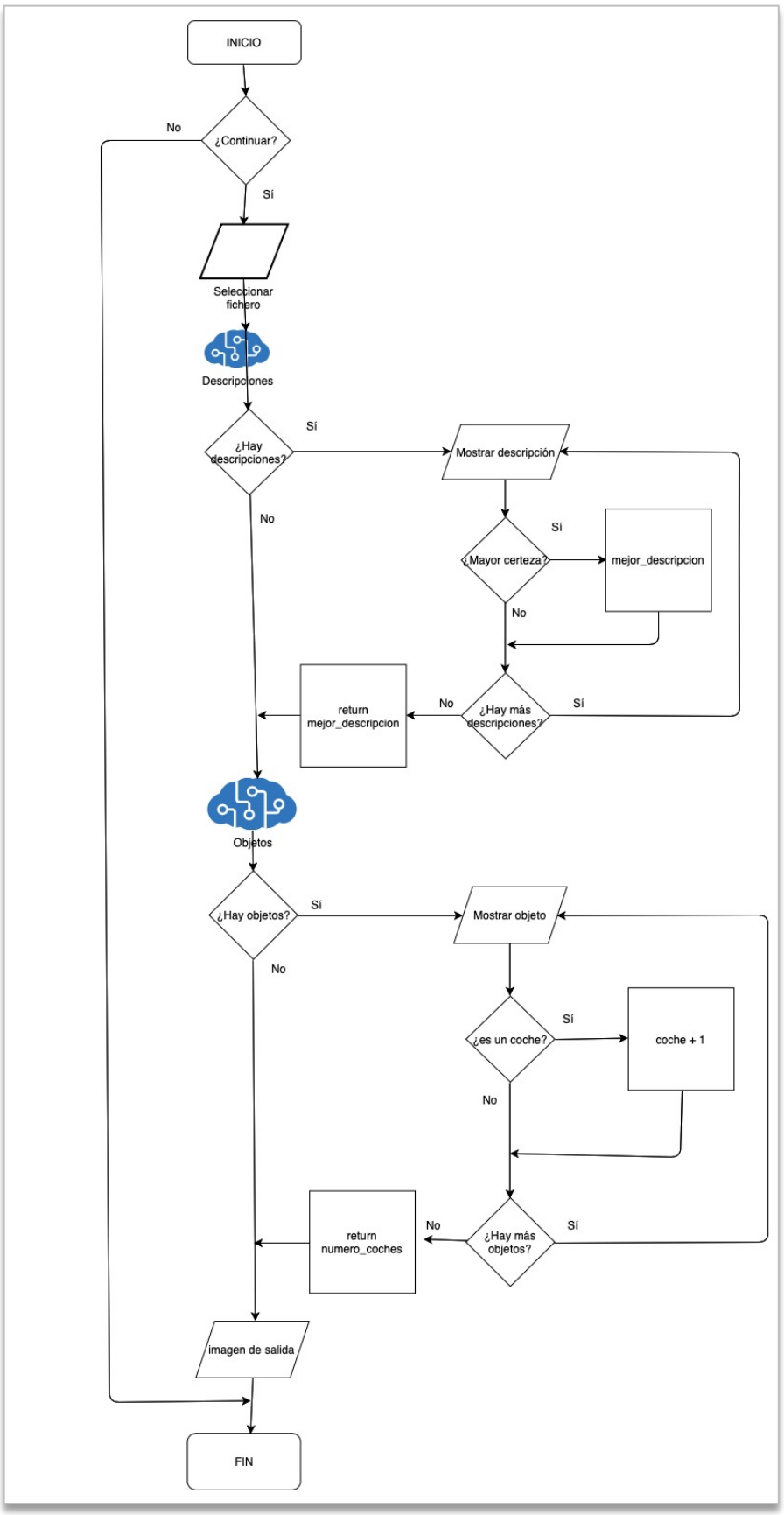


Ilustración 38: Flujograma

Como puede verse en la imagen anterior, el flujo es el siguiente:

1. Se presenta una ventana de bienvenida en la que se puede optar por continuar con la ejecución del programa o detenerlo.
2. Se presenta una ventana que permite seleccionar el fichero (la imagen) que se desea analizar.
3. Se obtienen las descripciones del contenido de la imagen
   a. Determinación de la descripción con mayor certeza[33]
4. Se obtiene la relación de objetos presentes en la imagen
   b. Cálculo del número de coches
5. Se presenta la ventana de salida con las características expuestas más arriba.

7.1 Explicación del código del programa

El código comienza con un primer bloque (Ilustración 39) en el que se incluye una serie de importaciones:

— Os.

- El módulo "os" nos provee de varios métodos para trabajar con las funcionalidades del sistema operativo, en nuestro caso con la finalidad de obtener el nombre del fichero seleccionado mediante *os.path.split()*.

— Sys.

- Este módulo provee acceso a algunas variables usadas o mantenidas por el intérprete y a funciones que interactúan fuertemente con el intérprete. En nuestro caso se usa para finalizar la ejecución del programa con *sys.exit().*

— Tkinter.

- Es un módulo que facilita la presentación de la interfaz gráfica, en realidad se trata de un puente para usar otra biblioteca gráfica llamada Tcl/Tk en Python. En nuestro programa se usa en la función que presenta la ventana de bienvenida y en la que permite seleccionar el fichero que se va a analizar.

— PIL.

- Es una librería que permite la edición de imágenes, lo usaremos para abrir el fichero que contiene la imagen a analizar, como paso necesario para incrustarla en la imagen de salida.

— Io.

- Este módulo implementa las clases detrás de la función *open()*, incorporada del intérprete para operaciones de entrada y salida basadas en archivos.

— Requests.

- *Requests* es una librería para HTTP, que usaremos en nuestro programa para realizar las peticiones POST a Computer Vision en las dos funciones que invocan a este servicio mediante las llamadas *requests.post().*

— Matplotlib.

[33] En todos los ejemplos se ha correspondido con la primera descripción, pero la lógica del programa aconseja hacer esta validación)

- *Matplotlib* es una biblioteca para la generación de gráfico, que usaremos en nuestro programa para generar la ventana final de salida de datos.

Consideramos incluido en este primer bloque la declaración de las variables en las que guardamos los valores de las credenciales proporcionadas por *Computer Vision* para acceder a sus funcionalidades (la clave se muestra oculta por razones de seguridad).

```
# ======================== IMPORTACIONES DE BIBLIOTECAS O MODULOS ================================================
# ····························· Bibliotecas sistema operativo ····························································
import os
import sys
# ····························· Modulos tkinter ····························································
from tkinter import *
from tkinter import messagebox as ventana_mensaje
from tkinter import filedialog, Tk  # lo usamos para presentar la ventana de selección del fichero a analizar
# ····························· Modulos PIL ····························································
from PIL import Image as Imagen_foto_PIL  # lo usamos para abrir el fichero de la foto y volcarlo a la imagen final
from io import BytesIO  # BytesIO permite trabajar con Bytes en memoria, necesario para la salida
# ····························· Modulos Salida ····························································
import requests
import matplotlib.pyplot as imagen_generada  # Genera imagen de salida con foto y los textos con información

# ========================  VARIABLES ================================================================================
# ····················· Credenciales de COMPUTER VISION (AZURE)
AZURE_mi_clave = X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-"
AZURE_mi_URL = "https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/"
```

Ilustración 39: Código (1) - importaciones y variables

A continuación, el código del programa incluye las funciones que lo componen.

La primera función (*pantalla_inicio*), Ilustración 40, define las características de la ventana de inicio (Ilustración 46), incluyendo dos botones que permiten entrar en la ejecución del programa propiamente dicho o finalizarla directamente, mediante la llamada a la función *salir* (también se finaliza pulsando el botón ☒)

```
# ··············· VENTANA DE INICIO DEL PROGRAMA ·····························································
def pantalla_inicio():
    ventana = Tk()
    ventana.title("Práctica del TFG de Alvaro Fernández-Taviro Gracia")
    ventana.geometry('700x500+200+100')
    ventana.config(background="#617858")
    ventana.resizable(False, False)
    ventana.protocol("WM_DELETE_WINDOW", salir)

    imagen = PhotoImage(file='Logo-UAH.png')

    fondo = Label(ventana, image=imagen).place(x=200, y=100)

    texto1 = Label(ventana, text="GRADO DE SISTEMAS DE INFORMACIÓN",
                   font=("Agency FB", 20, "bold"),
                   background="#617858",
                   fg="yellow").place(x=170, y=5)

    texto2 = Label(ventana, text="Trabajo de Fin de Grado",
                   font=("Agency FB", 15, "italic"),
                   background="#617858",
                   fg="yellow").place(x=260, y=35)

    texto3 = Label(ventana, text="Álvaro Fernández-Taviro Gracia",
                   font=("Agency FB", 15, "italic"),
                   background="#617858",
                   fg="yellow").place(x=235, y=65)

    boton_salir = Button(ventana, text="Salir",
                         font=("Agency FB", 20),
                         command=quit,
                         foreground="black",                # LETRAS
                         activeforeground="gray",           # LETRAS PULSADAS
                         borderwidth=4,
                         width=15).place(x=150, y=440)

    boton_continuar = Button(ventana, text="Comenzar",
                             font=("Agency FB", 20),
                             command=ventana.destroy,
                             foreground="black",            # LETRAS
                             activeforeground="gray",       # LETRAS PULSADAS
                             borderwidth=4,
                             width=15).place(x=400, y=440)

    ventana.mainloop()


# ·················· Salida del programa si se elige 'CANCELAR' en la ventana de inicio·····························
def salir():

    ventana = Tk()
    ventana_mensaje.showinfo('Saliendo del programa', 'Pulsar para salir')
    ventana.destroy()
    sys.exit()
```

Ilustración 40: Código (2) - Ventana de inicio

La siguiente función (seleccionar_fichero), Ilustración 41, se usa para que el usuario seleccione el *fichero* que desea analizar, para lo cual se le presenta una ventana empleando las funcionalidades del módulo tkinter (ver Ilustración 47). La función devuelve la ruta completa del fichero seleccionado.

```
# ····· VENTANA PARA SELECCIÓN DE LA IMAGEN QUE SE VA A ANALIZAR ·····
def seleccionar_fichero():

    ventana = Tk()
    ventana.withdraw()
    ventana.filename = filedialog.askopenfilename(initialdir="/xcvfg", title="Select file")

    foto_seleccionada = ventana.filename
    ventana.destroy()

    return foto_seleccionada
```

Ilustración 41: Código (3) - Ventana de selección de fichero

A continuación, el programa incluye las dos funciones que podríamos considerar más importantes para el objeto del presente trabajo, pues son las que realmente interactúan con Microsoft Azure.

Ambas funciones siguen un mismo esquema general, que coincide con lo expuesto en los diversos apartados del punto 5.3 y lo mostrado en los ejemplos del anexo 3, haciendo llamadas directas a la API mediante peticiones POST a las URL correspondientes, incluyendo:

- La URL, compuesta de una primera parte que es "*el punto de conexión"* obtenido de los datos de recurso "*Computer Vision*" en el portal de Azure y una segunda que identifica la funcionalidad que queremos emplear
- La clave de uso de Azure y el tipo de contenido que se va a pasar.
- Los parámetros de la propia funcionalidad
- El fichero (la imagen) a analizar.

Ambas funciones reciben dos parámetros: el nombre del fichero (se usa como encabezamientos de los datos que se muestran en la salida por consola) y la dirección del fichero local que se va a analizar (que se pasa a Computer Vision con la petición POST)

En el caso de la función *obtener_descripciones* (Ilustración 42), se hace uso de la funcionalidad de descripción de una imagen expuesta en el punto 5.3.2. El programa solicita un máximo de cinco (5) descripciones, presentando todas ellas por consola junto con el porcentaje de certeza de cada descripción (se resalta en negrita y color amarillo la descripción, ver Ilustración 49. A la vez, se almacena en una variable la que tenga mayor certeza, que es el valor que devuelve la función (para ser presentado en la imagen de salida).

En el código *PYTHON* incluimos una sentencia *try / except* para controlar la posibilidad de que se produzca un error al cargar la imagen (consecuencia, por ejemplo, de que su tamaño sea superior a los 4 MB que permite Azure como máximo)

```
# ····· OBTENCIÓN DE DESCRIPCIONES ····· ····· ·····
def obtener_descripciones(nombre_fichero, foto_a_analizar):
    descripcion_imagen = ""
    mejor_descripcion = ""

    print("===== DESCRIPCIÓN DE LA IMAGEN '" + nombre_fichero + "' =====")

    # Se invoca a la API para conseguir [5] descripciones (usamos 5 para mostrar la funcionalidad)

    try:
        URL_descripcion = AZURE_mi_URL + "vision/v3.1/describe"
        cabeceras = {'Ocp-Apim-Subscription-Key': AZURE_mi_clave, 'Content-Type': 'application/octet-stream'}
        parametros = {'maxCandidates': '5', 'language': 'es'}
        datos_imagen = open(foto_a_analizar, "rb").read()

        descripcion_imagen = requests.post(URL_descripcion, headers=cabeceras, params=parametros, data=datos_imagen)

    except:  # se gestiona la salida ordenada si se produce error en tamaño fichero, fallo autentificacion u otro
motivo
        print("¡Se ha producido un error al intenta el fichero seleccionado!")
        quit()

    # Obtener el texto de la(s) descripción(es),  con el nivel de confianza otorgado a cada descripción
    if len(descripcion_imagen.json()["description"]["captions"]) == 0:
        print("No hay descripción disponible.")
    else:
        confianza = 0
        contador = 1
        descripciones_encontradas = descripcion_imagen.json()["description"]["captions"]

        print()
        for descripcion_generada in descripciones_encontradas:
            print("{}) La imagen corresponde a \033[1;33m '{}' \033[0;m (con una certeza del {:.2f}%)".
                  format(contador, descripcion_generada["text"], descripcion_generada["confidence"] * 100))

            # la descricpción más fiable se almacena para presentarla en la imagen de la salida final
            if descripcion_generada["confidence"] > confianza:
                confianza = descripcion_generada["confidence"]
                mejor_descripcion = descripcion_generada["text"] + \
                    " (" + str(round((descripcion_generada["confidence"] * 100), 2)) + "%)"

            contador += 1

        print()
        print("La descripcion con mas fiabilidad es :"+mejor_descripcion)
        print()
    return mejor_descripcion
```

Ilustración 42: Código (4) - Obtención de descripciones

.

En cuanto a la función *obtener_objetos* (Ilustración 43), se hace uso de la funcionalidad expuesta en el punto 5.3.3.

Como resultado se presenta (en la consola) la relación de objetos, con su grado de certeza y su ubicación dentro de la imagen y devuelve el número de coches encontrados entre esos objetos. Dado que Azure no ofrece la posibilidad de obtener directamente el

número de objetos de una determinada clase, para obtener el número de coches detectados en la imagen, debemos programar un contador:

- Se cuenta el número de coches y se almacena el valor en una variable que será presentado en la imagen de salida del programa.
- Con relación a este contador, debe considerarse que no todos los coches son objetos del tipo "*car*" ya que en las imágenes que hemos usado algunos coches son clasificados como "*taxi*", "*van*", "*hatchback*" o "*race car*", por lo que el contador acumula esos cinco tipos de objetos, no siendo descartable que existan más tipos de objetos que correspondan a coches y que deberían añadirse a ese contador en el caso de detectarse, en cuyo caso habría que añadir el correspondiente valor al array *tipo_de_coches*.

```
# ............ OBJETOS DETECTADOS EN LA IMAGEN ....................................................................
def obtener_objetos(nombre_fichero, foto_a_analizar):
    numero_coches_encontrado = 0
    tipos_de_coches = ["CAR", "VAN", "TAXI", "HATCHBACK", "RACE CAR"]

    print("===== OBJETOS ENCONTRADOS EN LA IMAGEN '" + nombre_fichero + "' =====")

    # Invocar  API

    URL_descripcion = AZURE_mi_URL + "vision/v3.1/detect"
    cabeceras = {'Ocp-Apim-Subscription-Key': AZURE_mi_clave, 'Content-Type': 'application/octet-stream'}
    datos_imagen = open(foto_a_analizar, "rb").read()

    objetos_encontrados = requests.post(URL_descripcion, headers=cabeceras, data=datos_imagen)
    print()

    # Resultados obtenidos y coordenadas de donde se encuentran (sirve de contar nº de coches)
    if len(objetos_encontrados.json()) == 0:
        print("No existe ningún objeto.")
    else:
        numero_coches_encontrado = 0
        contador = 1

        for datos_objeto in objetos_encontrados.json()["objects"]:
            print("  {}.-- \033[1;33m {} \033[0;m (con una certeza del {:.2f}%).".format(
                contador, datos_objeto["object"], datos_objeto["confidence"] * 100, ))

            print("           * Situado en la imagen en x1: {}; x2: {}; y1: {}; y2: {}".format(
                datos_objeto["rectangle"]["x"],
                datos_objeto["rectangle"]["x"] + datos_objeto["rectangle"]["w"],
                datos_objeto["rectangle"]["y"],
                datos_objeto["rectangle"]["y"] + datos_objeto["rectangle"]["h"]))

            # CONTAMOS OBJETOS QUE SON CAR, TAXI, RACE CAR, VAN o HATCHBACK ----- ¿HAY MÁS TIPOS DE COCHES? -----
            if datos_objeto["object"].upper() in tipos_de_coches:
                numero_coches_encontrado = numero_coches_encontrado + 1

            contador += 1

        print("\033[1;33m Número de coches:  {}\033[0;m ".format(numero_coches_encontrado))
    print()
    return numero_coches_encontrado
```

Ilustración 43: Código (5) - Obtención de objetos

Por último, mediante la función *mostrar_salida* (Ilustración 44) se genera la imagen de salida (ver Ilustración 48) con las capacidades de la librería MATPLOTLIB, incluyendo en ella:

- En el título de la imagen el nombre del fichero analizado
- Encima de la imagen, la descripción y el porcentaje de certeza asignado a ella
- Debajo de la imagen el número de coches detectados

```
# ····· VENTANA DE PRESENTACIÓN DE RESULTADOS ·····
def mostrar_salida(foto, nombre, descripcion, numero):

    imagen_con_read = open(foto, "rb").read()                    # abrir fichero de la foto
    imagen_buffer = BytesIO(imagen_con_read)                     # imagen en buffer en formato binario
    imagen_incrustada = Imagen_foto_PIL.open(imagen_buffer)      # abre la imagen

    imagen_generada.figure(num=nombre)                           # muestro nombre de fichero en título de la ventana
    imagen_generada.imshow(imagen_incrustada)                    # foto analizada (formato binario)
    imagen_generada.axis("on")                                   # para mostrar texto nº de coches
    imagen_generada.xticks([])                                   # ocultar marcas eje x
    imagen_generada.yticks([])                                   # ocultar marcas eje y
    imagen_generada.title(descripcion.upper(), size="x-large")   # texto descripcion ofrecida por AZURE
    imagen_generada.xlabel('Número de coches: ' + str(numero))   # texto con nº de coches
    imagen_generada.show()
```

Ilustración 44: Código (6) - Imagen de salida de datos obtenidos

El código acaba con la función *MAIN* (Ilustración 45) que llama sucesivamente a cada una de las funciones que componen el programa:

- *pantalla_inicio()*
- *seleccionar_fichero* (recibe como resultado la ruta del fichero a analizar y a partir de ella obtiene el nombre del fichero y su extensión)
- *obtener_descripciones()* (recibe como resultado la mejor descripción)
- *obtener_objetos()* (recibe como resultado el número de coches encontrados)
- *mostrar_salida()* (muestra la imagen, la mejor descripción y el número de coches)

```
# ============================================================================================================
# ==================== M A I N ===============================================================================
# ============================================================================================================
def main():
    direccion_partida = ""

    pantalla_inicio()
    foto_a_analizar = seleccionar_fichero()

    # La variable "foto_a_analizar" contiene la ruta completa del fichero.
    # Necesitamos obtente el nombre del fichero a partir de la ruta completa
    if not foto_a_analizar:
        print("¡No se ha seleccionado ninguna foto")
        quit()
    else:
        direccion_partida = os.path.split(foto_a_analizar)

    # Descartamos el elemento [0] porque no necesitamos la ruta
    # Usamos solo el elemento [1] porque es el que contiene el nomebre del fichero
    nombre_fichero = direccion_partida[1]

    # llamadas a las funciones que devuelven los valores que proporciona Computer Vision
    mejor_descripcion = obtener_descripciones(nombre_fichero, foto_a_analizar)
    numero_coches = obtener_objetos(nombre_fichero, foto_a_analizar)

    # presentación de imagen de salida con los datos requeridos
    mostrar_salida(foto_a_analizar, nombre_fichero, mejor_descripcion, numero_coches)

# ............................................................................................
if __name__ == "__main__":
    main()
```

Ilustración 45: Código (7) - Función MAIN

8 Resultados de ejecución

La aplicación de prueba desarrollada presenta inicialmente una pantalla de bienvenida con los datos básicos del TFG y las opciones de salir de la aplicación o bien comenzar con la prueba (Ilustración 46).

Pulsando en Comenzar se arranca el programa.

Ilustración 46: Ventana de inicio

Al inicio del programa, el usuario debe seleccionar la imagen que desea analizar (la carpeta que se presenta por defecto es aquella en la que se encontraba la imagen analizada la última vez que se ejecutó).

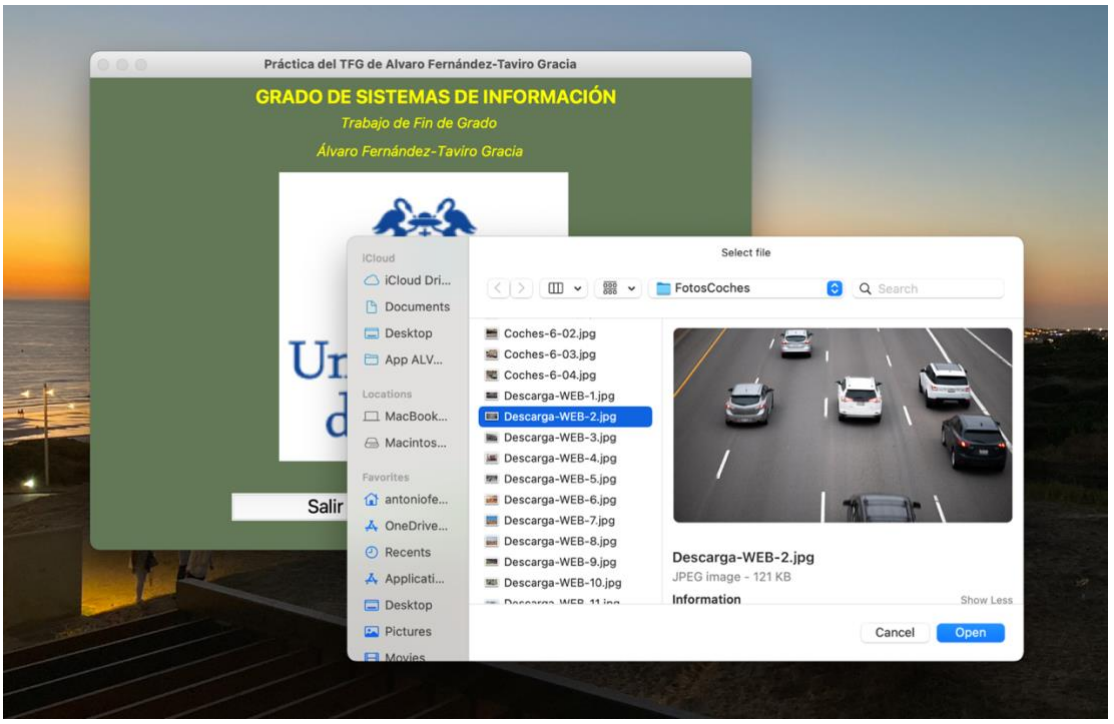


Ilustración 47: Ventana de selección de imagen

El programa genera una imagen de salida (Ilustración 48) en la que se presenta:

- En el título de la ventana, el nombre del fichero analizado.
- Encima de la fotografía analizada, la descripción de esa fotografía.
- Debajo de la fotografía analizada, el número de coches detectado.

Ilustración 48: Imagen mostrando datos de salida del programa

Como ampliación de lo solicitado, se presentan (en la consola, en formato texto), los datos a partir de los cuales se han obtenido tanto la descripción como el número de coches, tal y como se explicaba en el punto anterior (ver Ilustración 49).

```
===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-2.jpg' =====

1) La imagen corresponde a  'una carretera con autos'  (con una certeza del 64.77%)
2) La imagen corresponde a  'una carretera con coches'  (con una certeza del 64.67%)
3) La imagen corresponde a  'un coche deportivo en una carretera'  (con una certeza del 64.57%)
4) La imagen corresponde a  'una carretera con autos estacionados'  (con una certeza del 64.47%)
5) La imagen corresponde a  'un coche en una carretera'  (con una certeza del 64.37%)

La descripcion con mas fiabilidad es :una carretera con autos (64.77%)

===== OBJETOS ENCONTRADOS EN LA IMAGEN 'Descarga-WEB-2.jpg' =====

  1.--  car  (con una certeza del 76.70%).
          * Situado en la imagen en x1: 360; x2: 466; y1: 9; y2: 98
  2.--  Van  (con una certeza del 61.20%).
          * Situado en la imagen en x1: 826; x2: 1004; y1: 115; y2: 265
  3.--  car  (con una certeza del 87.60%).
          * Situado en la imagen en x1: 175; x2: 336; y1: 175; y2: 325
  4.--  car  (con una certeza del 78.10%).
          * Situado en la imagen en x1: 541; x2: 705; y1: 162; y2: 312
  5.--  car  (con una certeza del 83.20%).
          * Situado en la imagen en x1: 909; x2: 1126; y1: 287; y2: 484
  6.--  car  (con una certeza del 61.30%).
          * Situado en la imagen en x1: 585; x2: 829; y1: 554; y2: 644
 Número de coches:  6
```

Ilustración 49: Datos de salida por consola

9 Conclusiones

Dejando aparte cuestiones que, sin ser el objetivo fundamental de este trabajo, ya se han puesto en valor a lo largo del trabajo, tales como puede ser la flexibilidad y economía que proporciona el "*cloud computing*", a continuación, se presentan las conclusiones obtenidas tras el uso de *Computer Vision*.

Lo primero a destacar es la rapidez y facilidad con la que se pueden emplear capacidades de análisis de imágenes que ofrece Azure (*Computer Vision)* sin tener formación previa ni experiencia en el ámbito de la Inteligencia Artificial ni de la visión artificial .

Sin perder de vista nunca esa gran virtud, se considera que la herramienta tiene margen de mejora, no en la forma de emplearla, pero sí en los parámetros con los que se lanzan las consultas y en las respuestas que devuelve a esas consultas:

- No existe una taxonomía ni una funcionalidad que faciliten consultas del tipo "¿Cuántos objetos del tipo x hay en la imagen?
- Las descripciones no siempre son aproximadas a la realidad
- Los textos en español son incorrectos, se usa el idioma español de Estados Unidos
- Las coordenadas devueltas son difíciles de entender para detectar el objeto al que se refiere
- Los valores devueltos por el OCR son bastante complicados de manejar, siendo mejor y más sencilla la herramienta "read", aunque más lenta.
- Se echa en falta una funcionalidad que permita comparar caras encontradas con otras conocidas que no sean "celebrities", aunque podría ser que Azure ofrezca esa funcionalidad gracias al servicio *Custom Vision*.
- Las etiquetas devueltas son diferentes según la funcionalidad con las que se obtengan, lo cual causa confusión.
- Teniendo en cuenta que entre las etiquetas que devuelve se encuentra la de "licence plate" y que existe la capacidad OCR, parece una necesidad bastante obvia la de disponer de una funcionalidad que devuelva el número de matrícula de la placa.

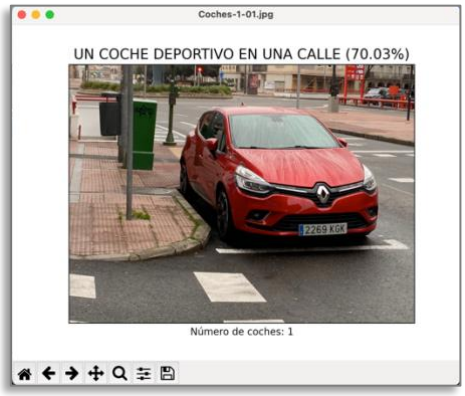
La experiencia de usuario es definitivamente positiva, teniendo en cuenta que la presenta un usuario no experimentado en este campo, si bien es probable que un usuario experimentado la pueda encontrar pobre.

Por último, como propuesta de futuro, se cree conveniente realizar trabajos similares al presente con herramientas de otros proveedores para poder comparar los resultados obtenidos.

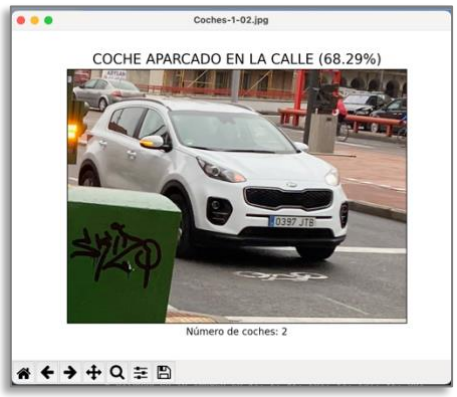
ANEXO 1: Análisis de imágenes de coches (descripción y objetos)

1 Imágenes analizadas

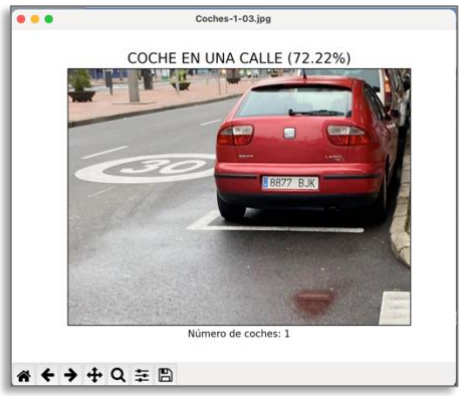
A continuación, se incluye una serie de fotografías junto con el resultado de la ejecución del programa para cada una de ellas, según lo explicado en el punto 8.



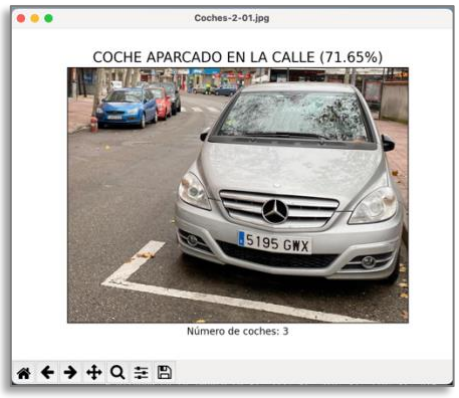

```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-1-01.jpg' =====
1) La imagen corresponde a 'un coche deportivo en una calle' (con una certeza del 70.03%)
2) La imagen corresponde a 'un coche deportivo en una esquina de una calle' (con una
certeza del 68.18%)
3) La imagen corresponde a 'coche en una calle' (con una certeza del 65.08%)
4) La imagen corresponde a 'un coche en una calle' (con una certeza del 64.98%)
5) La imagen corresponde a 'un coche en la calle' (con una certeza del 64.88%)

===== Objetos detectados en la imagen =====
  1.-- car (con una certeza del 92.00%).
          * Situado en la imagen en x1: 520; x2: 1469; y1: 199; y2: 851
Número de coches:  1
```




```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-1-02.jpg' =====
1) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 68.29%)
2) La imagen corresponde a 'un coche deportivo en la calle' (con una certeza del 68.19%)
3) La imagen corresponde a 'un coche deportivo en una calle' (con una certeza del 68.09%)
4) La imagen corresponde a 'coche en una calle' (con una certeza del 63.72%)
5) La imagen corresponde a 'un coche en la calle' (con una certeza del 63.62%)

===== Objetos detectados en la imagen =====
  1.-- Van (con una certeza del 50.90%).
          * Situado en la imagen en x1: 18; x2: 174; y1: 15; y2: 112
  2.-- Container (con una certeza del 51.20%).
          * Situado en la imagen en x1: 2; x2: 281; y1: 257; y2: 601
  3.-- car (con una certeza del 51.00%).
          * Situado en la imagen en x1: 26; x2: 618; y1: 74; y2: 405
Número de coches:  2
```




```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-1-03.jpg' =====
1) La imagen corresponde a 'coche en una calle' (con una certeza del 72.22%)
2) La imagen corresponde a 'un coche en una calle' (con una certeza del 72.12%)
3) La imagen corresponde a 'un coche deportivo en una calle' (con una certeza del 72.02%)
4) La imagen corresponde a 'un coche en la calle' (con una certeza del 71.92%)
5) La imagen corresponde a 'un coche en un calle' (con una certeza del 71.82%)

===== Objetos detectados en la imagen =====
  1.-- Vehicle registration plate (con una certeza del 58.10%).
          * Situado en la imagen en x1: 796; x2: 1012; y1: 426; y2: 500
  2.-- car (con una certeza del 90.00%).
          * Situado en la imagen en x1: 571; x2: 1370; y1: 10; y2: 643
Número de coches:  1
```

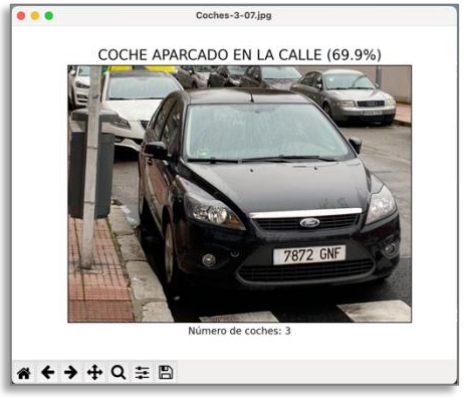
	===== DESCRIPCIÓN DE LA IMAGEN 'Coches-2-01.jpg' ===== 1) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 71.65%) 2) La imagen corresponde a 'un coche deportivo en la calle' (con una certeza del 71.55%) 3) La imagen corresponde a 'un coche estacionado al lado de una calle' (con una certeza del 66.67%) 4) La imagen corresponde a 'coche en una calle' (con una certeza del 66.57%) 5) La imagen corresponde a 'un coche en la calle' (con una certeza del 66.47%) ===== Objetos detectados en la imagen ===== 1.-- car (con una certeza del 50.20%). * Situado en la imagen en x1: 925; x2: 1082; y1: 89; y2: 173 2.-- Van (con una certeza del 51.80%). * Situado en la imagen en x1: 177; x2: 573; y1: 110; y2: 374 3.-- car (con una certeza del 83.90%). * Situado en la imagen en x1: 613; x2: 2083; y1: 68; y2: 1340 Número de coches: 3
	===== DESCRIPCIÓN DE LA IMAGEN 'Coches-2-02.jpg' ===== 1) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 70.49%) 2) La imagen corresponde a 'coche en una calle' (con una certeza del 67.08%) 3) La imagen corresponde a 'un coche deportivo en una calle' (con una certeza del 66.98%) 4) La imagen corresponde a 'un coche deportivo en la calle' (con una certeza del 66.88%) 5) La imagen corresponde a 'un coche en una calle' (con una certeza del 66.78%) ===== Objetos detectados en la imagen ===== 1.-- Land vehicle (con una certeza del 55.40%). * Situado en la imagen en x1: 367; x2: 781; y1: 440; y2: 657 2.-- Land vehicle (con una certeza del 58.60%). * Situado en la imagen en x1: 105; x2: 510; y1: 570; y2: 849 3.-- car (con una certeza del 82.20%). * Situado en la imagen en x1: 581; x2: 1603; y1: 601; y2: 1090 4.-- car (con una certeza del 85.50%). * Situado en la imagen en x1: 1513; x2: 3325; y1: 632; y2: 1623 Número de coches: 2
	===== DESCRIPCIÓN DE LA IMAGEN 'Coches-2-03.jpg' ===== 1) La imagen corresponde a 'un camión de doble piso en la calle' (con una certeza del 60.13%) 2) La imagen corresponde a 'un camión de doble piso parado en la calle' (con una certeza del 58.76%) 3) La imagen corresponde a 'un camión de doble piso en la acera de una calle' (con una certeza del 56.66%) 4) La imagen corresponde a 'un camión de doble piso en medio de la calle' (con una certeza del 56.56%) 5) La imagen corresponde a 'un camión de doble piso en la esquina de una calle' (con una certeza del 56.24%) ===== Objetos detectados en la imagen ===== 1.-- bus (con una certeza del 51.50%). * Situado en la imagen en x1: 1945; x2: 2509; y1: 156; y2: 612 2.-- car (con una certeza del 85.80%). * Situado en la imagen en x1: 555; x2: 1723; y1: 405; y2: 779 3.-- car (con una certeza del 76.50%). * Situado en la imagen en x1: 4; x2: 289; y1: 740; y2: 1365 4.-- shuttle bus (con una certeza del 51.60%). * Situado en la imagen en x1: 845; x2: 2387; y1: 78; y2: 625 Número de coches: 2
	===== DESCRIPCIÓN DE LA IMAGEN 'Coches-2-04.jpg' ===== 1) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 65.16%) 2) La imagen corresponde a 'un coche en la calle' (con una certeza del 65.06%) 3) La imagen corresponde a 'un coche en una calle' (con una certeza del 63.25%) 4) La imagen corresponde a 'coche en una calle' (con una certeza del 61.13%) 5) La imagen corresponde a 'un coche estacion en el suelo' (con una certeza del 58.97%) ===== Objetos detectados en la imagen ===== 1.-- car (con una certeza del 73.20%). * Situado en la imagen en x1: 1588; x2: 1834; y1: 42; y2: 186 2.-- Vehicle registration plate (con una certeza del 50.80%). * Situado en la imagen en x1: 869; x2: 1291; y1: 603; y2: 727 3.-- car (con una certeza del 84.00%). * Situado en la imagen en x1: 397; x2: 1796; y1: 0; y2: 1329 Número de coches: 2


```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-3-01.jpg' =====
1) La imagen corresponde a 'un coche en una calle' (con una certeza del 64.08%)
2) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 63.98%)
3) La imagen corresponde a 'coche en una calle' (con una certeza del 61.93%)
4) La imagen corresponde a 'un coche en la calle' (con una certeza del 61.83%)
5) La imagen corresponde a 'un coche deportivo en una carretera' (con una certeza del
61.17%)

===== Objetos detectados en la imagen =====
  1.-- car (con una certeza del 89.70%).
          * Situado en la imagen en x1: 253; x2: 1295; y1: 884; y2: 1680
  2.-- car (con una certeza del 88.40%).
          * Situado en la imagen en x1: 1527; x2: 2290; y1: 917; y2: 1605
  3.-- car (con una certeza del 88.70%).
          * Situado en la imagen en x1: 2571; x2: 3657; y1: 908; y2: 1633
Número de coches:  3
```


```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-3-02.jpg' =====
1) La imagen corresponde a 'un coche en un estacionamiento' (con una certeza del 70.68%)
2) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 67.79%)
3) La imagen corresponde a 'un coche estacion en el suelo' (con una certeza del 63.93%)
4) La imagen corresponde a 'un coche deportivo de color blanco' (con una certeza del
63.83%)
5) La imagen corresponde a 'un coche estacionado en la calle' (con una certeza del 63.73%)

===== Objetos detectados en la imagen =====
  1.-- car (con una certeza del 52.50%).
          * Situado en la imagen en x1: 2834; x2: 3370; y1: 362; y2: 994
  2.-- car (con una certeza del 76.00%).
          * Situado en la imagen en x1: 2237; x2: 3092; y1: 397; y2: 1381
  3.-- car (con una certeza del 87.30%).
          * Situado en la imagen en x1: 24; x2: 2696; y1: 405; y2: 2554
Número de coches:  3
```


```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-3-03.jpg' =====
1) La imagen corresponde a 'un coche en una calle' (con una certeza del 68.65%)
2) La imagen corresponde a 'un coche en la calle' (con una certeza del 68.55%)
3) La imagen corresponde a 'coche en una calle' (con una certeza del 66.65%)
4) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 66.55%)
5) La imagen corresponde a 'un coche deportivo en la calle' (con una certeza del 66.45%)

===== Objetos detectados en la imagen =====
  1.-- car (con una certeza del 82.30%).
          * Situado en la imagen en x1: 1129; x2: 1494; y1: 233; y2: 475
  2.-- car (con una certeza del 79.40%).
          * Situado en la imagen en x1: 1443; x2: 1713; y1: 242; y2: 464
  3.-- car (con una certeza del 82.10%).
          * Situado en la imagen en x1: 311; x2: 1615; y1: 212; y2: 1258
Número de coches:  3
```

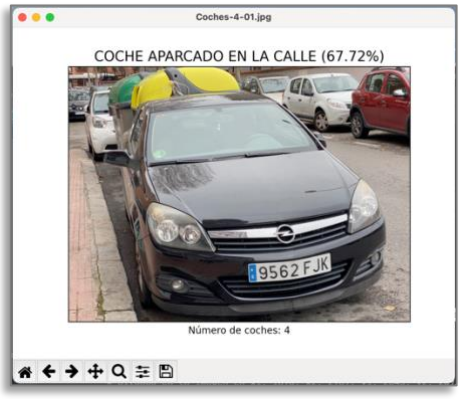

```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-3-04.jpg' =====
1) La imagen corresponde a 'camioneta en la calle' (con una certeza del 63.56%)
2) La imagen corresponde a 'camioneta estacionada en la calle' (con una certeza del 63.46%)
3) La imagen corresponde a 'una camioneta estacionada en la calle' (con una certeza del
63.36%)
4) La imagen corresponde a 'una camioneta en una calle' (con una certeza del 61.32%)
5) La imagen corresponde a 'una camioneta en la calle' (con una certeza del 61.22%)

===== Objetos detectados en la imagen =====
  1.-- car (con una certeza del 53.20%).
          * Situado en la imagen en x1: 0; x2: 921; y1: 680; y2: 1498
  2.-- taxi (con una certeza del 50.80%).
          * Situado en la imagen en x1: 948; x2: 1798; y1: 932; y2: 1583
  3.-- car (con una certeza del 90.10%).
          * Situado en la imagen en x1: 1879; x2: 3088; y1: 914; y2: 1773
Número de coches:  3
```

	===== DESCRIPCIÓN DE LA IMAGEN 'Coches-3-05.jpg' ===== 1) La imagen corresponde a 'un coche estacion en el suelo' (con una certeza del 63.37%) 2) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 63.27%) 3) La imagen corresponde a 'un coche estacionado en la calle' (con una certeza del 63.17%) 4) La imagen corresponde a 'un coche deportivo en la calle' (con una certeza del 63.07%) 5) La imagen corresponde a 'un coche deportivo en una calle' (con una certeza del 62.97%) ===== Objetos detectados en la imagen ===== 1.-- Land vehicle (con una certeza del 51.70%). * Situado en la imagen en x1: 138; x2: 270; y1: 11; y2: 123 2.-- car (con una certeza del 78.90%). * Situado en la imagen en x1: 214; x2: 591; y1: 5; y2: 429 3.-- car (con una certeza del 85.40%). * Situado en la imagen en x1: 1679; x2: 2460; y1: 33; y2: 475 4.-- car (con una certeza del 91.90%). * Situado en la imagen en x1: 588; x2: 2444; y1: 98; y2: 1264 Número de coches: 3
	===== DESCRIPCIÓN DE LA IMAGEN 'Coches-3-06.jpg' ===== 1) La imagen corresponde a 'coche en una calle' (con una certeza del 68.77%) 2) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 68.67%) 3) La imagen corresponde a 'un coche en una calle' (con una certeza del 68.57%) 4) La imagen corresponde a 'un coche deportivo en una calle' (con una certeza del 68.47%) 5) La imagen corresponde a 'un coche en un calle' (con una certeza del 68.37%) ===== Objetos detectados en la imagen ===== 1.-- car (con una certeza del 84.90%). * Situado en la imagen en x1: 24; x2: 732; y1: 563; y2: 937 2.-- taxi (con una certeza del 52.20%). * Situado en la imagen en x1: 1600; x2: 2387; y1: 594; y2: 1075 3.-- taxi (con una certeza del 79.10%). * Situado en la imagen en x1: 527; x2: 1814; y1: 598; y2: 1304 Número de coches: 3
	===== DESCRIPCIÓN DE LA IMAGEN 'Coches-3-07.jpg' ===== 1) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 69.90%) 2) La imagen corresponde a 'un coche estacionado en la calle' (con una certeza del 68.68%) 3) La imagen corresponde a 'un coche estacionado' (con una certeza del 68.58%) 4) La imagen corresponde a 'un coche estacionado al lado de la calle' (con una certeza del 65.81%) 5) La imagen corresponde a 'un coche estacionado al lado de una calle' (con una certeza del 65.17%) ===== Objetos detectados en la imagen ===== 1.-- car (con una certeza del 83.70%). * Situado en la imagen en x1: 851; x2: 1240; y1: 12; y2: 236 2.-- car (con una certeza del 70.30%). * Situado en la imagen en x1: 33; x2: 456; y1: 14; y2: 332 3.-- car (con una certeza del 86.10%). * Situado en la imagen en x1: 241; x2: 1250; y1: 66; y2: 874 Número de coches: 3
	===== DESCRIPCIÓN DE LA IMAGEN 'Coches-3-08.jpg' ===== 1) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 73.02%) 2) La imagen corresponde a 'coche en una calle con nieve' (con una certeza del 61.00%) 3) La imagen corresponde a 'un coche en una calle con nieve' (con una certeza del 60.90%) 4) La imagen corresponde a 'un coche deportivo en una calle' (con una certeza del 60.80%) 5) La imagen corresponde a 'coche en una calle' (con una certeza del 60.70%) ===== Objetos detectados en la imagen ===== 1.-- car (con una certeza del 74.00%). * Situado en la imagen en x1: 2024; x2: 2886; y1: 781; y2: 1223 2.-- car (con una certeza del 88.50%). * Situado en la imagen en x1: 1194; x2: 2710; y1: 741; y2: 1484 3.-- car (con una certeza del 85.30%). * Situado en la imagen en x1: 234; x2: 2148; y1: 740; y2: 2145 Número de coches: 3


```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-3-09.jpg' =====
1) La imagen corresponde a 'un coche estacionado al lado de un edificio' (con una certeza
del 70.43%)
2) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 69.79%)
3) La imagen corresponde a 'un coche estacionado al lado de una calle' (con una certeza del
64.94%)
4) La imagen corresponde a 'un coche en una calle' (con una certeza del 64.84%)
5) La imagen corresponde a 'coche en una calle' (con una certeza del 64.74%)

===== Objetos detectados en la imagen =====
  1.-- car (con una certeza del 90.90%).
           * Situado en la imagen en x1: 29; x2: 1229; y1: 1036; y2: 1505
  2.-- car (con una certeza del 82.70%).
           * Situado en la imagen en x1: 1124; x2: 1669; y1: 1028; y2: 1376
  3.-- car (con una certeza del 91.80%).
           * Situado en la imagen en x1: 1385; x2: 2927; y1: 1026; y2: 2125
Número de coches:  3
```


```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-3-10.jpg' =====
1) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 70.79%)
2) La imagen corresponde a 'un coche deportivo en una calle' (con una certeza del 70.69%)
3) La imagen corresponde a 'un coche estacionado al lado de una calle' (con una certeza del
66.34%)
4) La imagen corresponde a 'un coche estacionado al lado de un calle' (con una certeza del
65.32%)
5) La imagen corresponde a 'coche en una calle' (con una certeza del 65.22%)

===== Objetos detectados en la imagen =====
  1.-- car (con una certeza del 50.90%).
           * Situado en la imagen en x1: 1319; x2: 1389; y1: 249; y2: 532
  2.-- car (con una certeza del 78.00%).
           * Situado en la imagen en x1: 952; x2: 1358; y1: 173; y2: 477
  3.-- Van (con una certeza del 53.60%).
           * Situado en la imagen en x1: 0; x2: 316; y1: 198; y2: 616
  4.-- car (con una certeza del 84.20%).
           * Situado en la imagen en x1: 124; x2: 1266; y1: 179; y2: 938
Número de coches:  4
```


```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-3-11.jpg' =====
1) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 64.28%)
2) La imagen corresponde a 'coche en una calle' (con una certeza del 60.61%)
3) La imagen corresponde a 'un coche en una calle' (con una certeza del 60.51%)
4) La imagen corresponde a 'un coche en la calle' (con una certeza del 60.41%)
5) La imagen corresponde a 'un coche en un calle' (con una certeza del 60.31%)

===== Objetos detectados en la imagen =====
  1.-- person (con una certeza del 58.60%).
           * Situado en la imagen en x1: 288; x2: 463; y1: 75; y2: 255
  2.-- car (con una certeza del 64.50%).
           * Situado en la imagen en x1: 15; x2: 335; y1: 179; y2: 602
  3.-- car (con una certeza del 88.10%).
           * Situado en la imagen en x1: 1240; x2: 1938; y1: 199; y2: 761
  4.-- car (con una certeza del 89.30%).
           * Situado en la imagen en x1: 65; x2: 1005; y1: 198; y2: 811
Número de coches:  3
```




```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-4-01.jpg' =====
1) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 67.72%)
2) La imagen corresponde a 'un coche estacionado en la calle' (con una certeza del 66.17%)
3) La imagen corresponde a 'un coche estacionado al lado de la calle' (con una certeza del
63.34%)
4) La imagen corresponde a 'un coche estacionado al lado de una calle' (con una certeza del
62.82%)
5) La imagen corresponde a 'un coche estacion en el suelo' (con una certeza del 61.39%)

===== Objetos detectados en la imagen =====
  1.-- Land vehicle (con una certeza del 55.20%).
           * Situado en la imagen en x1: 5; x2: 279; y1: 196; y2: 525
  2.-- car (con una certeza del 79.10%).
           * Situado en la imagen en x1: 2301; x2: 3164; y1: 72; y2: 725
  3.-- car (con una certeza del 81.50%).
           * Situado en la imagen en x1: 3048; x2: 3720; y1: 19; y2: 833
  4.-- car (con una certeza del 53.90%).
           * Situado en la imagen en x1: 176; x2: 621; y1: 155; y2: 1008
  5.-- Vehicle registration plate (con una certeza del 53.30%).
           * Situado en la imagen en x1: 1896; x2: 2969; y1: 2045; y2: 2395
  6.-- car (con una certeza del 64.80%).
           * Situado en la imagen en x1: 622; x2: 3376; y1: 197; y2: 2747
Número de coches:  4
```


```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-4-02.jpg' =====
1) La imagen corresponde a 'un coche estacion en el suelo' (con una certeza del 64.84%)
2) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 64.74%)
3) La imagen corresponde a 'un coche estacionado en la calle' (con una certeza del 64.64%)
4) La imagen corresponde a 'un coche antiguo estacionado en la calle' (con una certeza del
64.54%)
5) La imagen corresponde a 'un coche estacionado' (con una certeza del 64.44%)

===== Objetos detectados en la imagen =====
  1.-- car (con una certeza del 60.30%).
          * Situado en la imagen en x1: 653; x2: 1023; y1: 43; y2: 284
  2.-- Van (con una certeza del 55.00%).
          * Situado en la imagen en x1: 2493; x2: 2685; y1: 12; y2: 531
  3.-- car (con una certeza del 82.70%).
          * Situado en la imagen en x1: 1434; x2: 2251; y1: 0; y2: 445
  4.-- car (con una certeza del 76.70%).
          * Situado en la imagen en x1: 8; x2: 664; y1: 67; y2: 805
  5.-- car (con una certeza del 69.50%).
          * Situado en la imagen en x1: 446; x2: 2624; y1: 197; y2: 1928
Número de coches:  5
```


```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-4-03.jpg' =====
1) La imagen corresponde a 'una calle con coches' (con una certeza del 63.05%)
2) La imagen corresponde a 'coche en una calle' (con una certeza del 61.55%)
3) La imagen corresponde a 'un coche en una calle' (con una certeza del 61.45%)
4) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 61.35%)
5) La imagen corresponde a 'un coche en un calle' (con una certeza del 61.25%)

===== Objetos detectados en la imagen =====
  1.-- car (con una certeza del 55.40%).
          * Situado en la imagen en x1: 3205; x2: 3337; y1: 1117; y2: 1569
  2.-- Land vehicle (con una certeza del 55.40%).
          * Situado en la imagen en x1: 2619; x2: 3316; y1: 2194; y2: 2494
  3.-- car (con una certeza del 85.80%).
          * Situado en la imagen en x1: 44; x2: 729; y1: 933; y2: 1456
  4.-- car (con una certeza del 86.70%).
          * Situado en la imagen en x1: 833; x2: 1884; y1: 918; y2: 1545
  5.-- car (con una certeza del 84.20%).
          * Situado en la imagen en x1: 1795; x2: 3329; y1: 942; y2: 1733
Número de coches:  4
```


```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-4-04.jpg' =====
1) La imagen corresponde a 'una camioneta estacionada en la calle' (con una certeza del
65.38%)
2) La imagen corresponde a 'camioneta en la calle' (con una certeza del 62.91%)
3) La imagen corresponde a 'un camión de bomberos en la calle' (con una certeza del 55.37%)
4) La imagen corresponde a 'un camión de bomberos estacionado en la calle' (con una certeza
del 53.79%)
5) La imagen corresponde a 'un camión de bomberos en una calle' (con una certeza del
52.57%)

===== Objetos detectados en la imagen =====
  1.-- car (con una certeza del 58.10%).
          * Situado en la imagen en x1: 2050; x2: 2428; y1: 1440; y2: 1606
  2.-- car (con una certeza del 75.30%).
          * Situado en la imagen en x1: 2290; x2: 2878; y1: 1440; y2: 1647
  3.-- car (con una certeza del 72.80%).
          * Situado en la imagen en x1: 3007; x2: 3781; y1: 1435; y2: 1739
  4.-- car (con una certeza del 80.80%).
          * Situado en la imagen en x1: 843; x2: 2087; y1: 1398; y2: 2325
Número de coches:  4
```

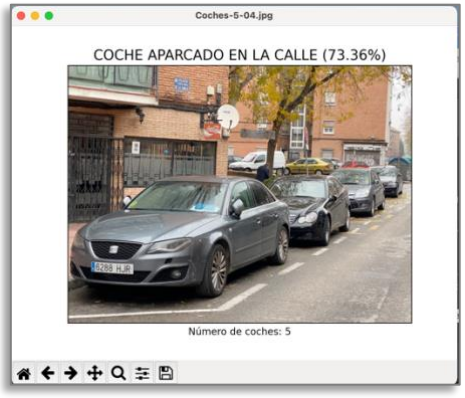

```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-4-05.jpg' =====
1) La imagen corresponde a 'un coche en una calle' (con una certeza del 71.77%)
2) La imagen corresponde a 'coche en una calle' (con una certeza del 69.74%)
3) La imagen corresponde a 'un coche en un calle' (con una certeza del 69.64%)
4) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 69.54%)
5) La imagen corresponde a 'una calle con autos estacionados' (con una certeza del 69.44%)

===== Objetos detectados en la imagen =====
  1.-- car (con una certeza del 67.40%).
          * Situado en la imagen en x1: 2599; x2: 3079; y1: 1154; y2: 1471
  2.-- car (con una certeza del 69.50%).
          * Situado en la imagen en x1: 2980; x2: 3721; y1: 1194; y2: 1664
  3.-- car (con una certeza del 73.50%).
          * Situado en la imagen en x1: 3446; x2: 4023; y1: 1235; y2: 1930
  4.-- car (con una certeza del 73.90%).
          * Situado en la imagen en x1: 601; x2: 2243; y1: 960; y2: 2344
Número de coches:  4
```


```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-4-06.jpg' =====
1) La imagen corresponde a 'una calle con carros' (con una certeza del 62.09%)
2) La imagen corresponde a 'una calle con coches' (con una certeza del 61.99%)
3) La imagen corresponde a 'calle con carros' (con una certeza del 60.76%)
4) La imagen corresponde a 'un coche en una calle' (con una certeza del 60.66%)
5) La imagen corresponde a 'coche en una calle' (con una certeza del 59.26%)

===== Objetos detectados en la imagen =====
  1.-- dormer window (con una certeza del 51.80%).
          * Situado en la imagen en x1: 1390; x2: 1553; y1: 131; y2: 309
  2.-- stop sign (con una certeza del 53.80%).
          * Situado en la imagen en x1: 337; x2: 673; y1: 330; y2: 450
  3.-- stop sign (con una certeza del 54.10%).
          * Situado en la imagen en x1: 732; x2: 1104; y1: 340; y2: 459
  4.-- stop sign (con una certeza del 53.10%).
          * Situado en la imagen en x1: 1265; x2: 1888; y1: 438; y2: 559
  5.-- car (con una certeza del 71.40%).
          * Situado en la imagen en x1: 4; x2: 232; y1: 580; y2: 794
  6.-- car (con una certeza del 85.10%).
          * Situado en la imagen en x1: 92; x2: 549; y1: 615; y2: 956
  7.-- car (con una certeza del 88.70%).
          * Situado en la imagen en x1: 580; x2: 1193; y1: 622; y2: 855
  8.-- Van (con una certeza del 54.00%).
          * Situado en la imagen en x1: 1210; x2: 1775; y1: 638; y2: 1012
  9.-- car (con una certeza del 88.00%).
          * Situado en la imagen en x1: 1818; x2: 2298; y1: 708; y2: 1071
Número de coches:  5
```


```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-4-07.jpg' =====
1) La imagen corresponde a 'edificio en frente de un coche' (con una certeza del 65.33%)
2) La imagen corresponde a 'camioneta estacionada en la calle' (con una certeza del 63.39%)
3) La imagen corresponde a 'una camioneta estacionada en la calle' (con una certeza del
63.29%)
4) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 63.19%)
5) La imagen corresponde a 'camioneta en la calle' (con una certeza del 60.63%)

===== Objetos detectados en la imagen =====
  1.-- car (con una certeza del 76.70%).
          * Situado en la imagen en x1: 11; x2: 1032; y1: 1007; y2: 1559
  2.-- car (con una certeza del 80.20%).
          * Situado en la imagen en x1: 892; x2: 2033; y1: 1043; y2: 1592
  3.-- car (con una certeza del 85.70%).
          * Situado en la imagen en x1: 1960; x2: 2937; y1: 1117; y2: 1645
  4.-- car (con una certeza del 88.60%).
          * Situado en la imagen en x1: 2964; x2: 3832; y1: 1198; y2: 1743
Número de coches:  4
```


```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-4-08.jpg' =====
1) La imagen corresponde a 'un coche estacionado al lado de una casa' (con una certeza del
61.03%)
2) La imagen corresponde a 'un coche estacionado al lado de un edificio' (con una certeza
del 60.93%)
3) La imagen corresponde a 'casa en frente de un coche' (con una certeza del 54.30%)
4) La imagen corresponde a 'un coche estacionado al lado de una calle' (con una certeza del
54.20%)
5) La imagen corresponde a 'una camioneta estacionada en la calle' (con una certeza del
54.10%)

===== Objetos detectados en la imagen =====
  1.-- car (con una certeza del 82.80%).
          * Situado en la imagen en x1: 78; x2: 1100; y1: 1094; y2: 1906
  2.-- car (con una certeza del 84.20%).
          * Situado en la imagen en x1: 1238; x2: 2165; y1: 1201; y2: 1787
  3.-- car (con una certeza del 87.40%).
          * Situado en la imagen en x1: 1903; x2: 3062; y1: 1190; y2: 1696
  4.-- car (con una certeza del 77.20%).
          * Situado en la imagen en x1: 2773; x2: 3463; y1: 1187; y2: 1644
Número de coches:  4
```


```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-5-01.jpg' =====
1) La imagen corresponde a 'una calle con coches' (con una certeza del 67.16%)
2) La imagen corresponde a 'una calle con carros' (con una certeza del 66.84%)
3) La imagen corresponde a 'un coche en una calle de noche' (con una certeza del 62.01%)
4) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 61.91%)
5) La imagen corresponde a 'un coche en una calle' (con una certeza del 61.81%)

===== Objetos detectados en la imagen =====
  1.-- Land vehicle (con una certeza del 66.90%).
          * Situado en la imagen en x1: 289; x2: 674; y1: 1327; y2: 1589
  2.-- car (con una certeza del 56.30%).
          * Situado en la imagen en x1: 736; x2: 1191; y1: 1392; y2: 1661
  3.-- car (con una certeza del 62.20%).
          * Situado en la imagen en x1: 1234; x2: 1674; y1: 1393; y2: 1727
  4.-- car (con una certeza del 70.50%).
          * Situado en la imagen en x1: 0; x2: 236; y1: 2025; y2: 2719
  5.-- car (con una certeza del 85.60%).
          * Situado en la imagen en x1: 1577; x2: 2661; y1: 1327; y2: 1847
  6.-- car (con una certeza del 84.00%).
          * Situado en la imagen en x1: 2163; x2: 3450; y1: 1421; y2: 1998
Número de coches:  5
```

	===== DESCRIPCIÓN DE LA IMAGEN 'Coches-5-02.jpg' ===== 1) La imagen corresponde a 'un coche deportivo de color blanco' (con una certeza del 66.00%) 2) La imagen corresponde a 'un coche deportivo' (con una certeza del 65.90%) 3) La imagen corresponde a 'un coche en una estación' (con una certeza del 63.07%) 4) La imagen corresponde a 'un coche de carreras en una pista de aeropuerto' (con una certeza del 61.41%) 5) La imagen corresponde a 'un coche de carreras en una pista' (con una certeza del 60.10%) ===== Objetos detectados en la imagen ===== 1.-- car (con una certeza del 65.20%). * Situado en la imagen en x1: 35; x2: 838; y1: 433; y2: 731 2.-- car (con una certeza del 74.60%). * Situado en la imagen en x1: 771; x2: 1659; y1: 412; y2: 848 3.-- taxi (con una certeza del 50.40%). * Situado en la imagen en x1: 1154; x2: 2119; y1: 445; y2: 895 4.-- car (con una certeza del 86.00%). * Situado en la imagen en x1: 1847; x2: 2741; y1: 455; y2: 993 5.-- car (con una certeza del 87.00%). * Situado en la imagen en x1: 2600; x2: 3649; y1: 455; y2: 1179 Número de coches: 5
	===== DESCRIPCIÓN DE LA IMAGEN 'Coches-5-03.jpg' ===== 1) La imagen corresponde a 'una calle con coches y edificios' (con una certeza del 73.98%) 2) La imagen corresponde a 'una calle con carros' (con una certeza del 71.04%) 3) La imagen corresponde a 'una calle con autos estacionados' (con una certeza del 70.94%) 4) La imagen corresponde a 'una calle con coches' (con una certeza del 70.76%) 5) La imagen corresponde a 'calle con carros' (con una certeza del 70.23%) ===== Objetos detectados en la imagen ===== 1.-- car (con una certeza del 52.00%). * Situado en la imagen en x1: 1172; x2: 1464; y1: 1215; y2: 1349 2.-- Land vehicle (con una certeza del 51.60%). * Situado en la imagen en x1: 2358; x2: 2578; y1: 1273; y2: 1436 3.-- car (con una certeza del 51.90%). * Situado en la imagen en x1: 2634; x2: 2976; y1: 1263; y2: 1428 4.-- car (con una certeza del 56.00%). * Situado en la imagen en x1: 31; x2: 223; y1: 1269; y2: 1833 5.-- car (con una certeza del 51.20%). * Situado en la imagen en x1: 123; x2: 586; y1: 1300; y2: 2028 6.-- car (con una certeza del 84.50%). * Situado en la imagen en x1: 183; x2: 1604; y1: 1297; y2: 2374 Número de coches: 5
	===== DESCRIPCIÓN DE LA IMAGEN 'Coches-5-04.jpg' ===== 1) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 73.36%) 2) La imagen corresponde a 'coche en una calle' (con una certeza del 69.81%) 3) La imagen corresponde a 'un coche deportivo en una calle' (con una certeza del 69.71%) 4) La imagen corresponde a 'un coche estacionado al lado de un edificio' (con una certeza del 69.61%) 5) La imagen corresponde a 'un coche en una calle' (con una certeza del 69.51%) ===== Objetos detectados en la imagen ===== 1.-- car (con una certeza del 60.80%). * Situado en la imagen en x1: 1709; x2: 2100; y1: 925; y2: 1130 2.-- car (con una certeza del 68.00%). * Situado en la imagen en x1: 2323; x2: 2802; y1: 955; y2: 1164 3.-- Land vehicle (con una certeza del 52.70%). * Situado en la imagen en x1: 3157; x2: 3490; y1: 1061; y2: 1386 4.-- car (con una certeza del 64.60%). * Situado en la imagen en x1: 2805; x2: 3372; y1: 1047; y2: 1598 5.-- car (con una certeza del 77.10%). * Situado en la imagen en x1: 2075; x2: 2921; y1: 1164; y2: 1926 6.-- car (con una certeza del 86.90%). * Situado en la imagen en x1: 32; x2: 2334; y1: 1162; y2: 2443 Número de coches: 5
	===== DESCRIPCIÓN DE LA IMAGEN 'Coches-5-05.jpg' ===== 1) La imagen corresponde a 'un coche en una calle' (con una certeza del 55.97%) 2) La imagen corresponde a 'coche en una calle' (con una certeza del 53.64%) 3) La imagen corresponde a 'un coche en un calle' (con una certeza del 53.54%) 4) La imagen corresponde a 'un coche en la calle' (con una certeza del 53.44%) 5) La imagen corresponde a 'una calle con coches' (con una certeza del 53.34%) ===== Objetos detectados en la imagen ===== 1.-- car (con una certeza del 70.30%). * Situado en la imagen en x1: 31; x2: 584; y1: 324; y2: 615 2.-- car (con una certeza del 79.80%). * Situado en la imagen en x1: 713; x2: 1648; y1: 273; y2: 709 3.-- car (con una certeza del 53.00%). * Situado en la imagen en x1: 1359; x2: 2284; y1: 296; y2: 712 4.-- car (con una certeza del 85.30%). * Situado en la imagen en x1: 7; x2: 625; y1: 394; y2: 1151 5.-- car (con una certeza del 86.90%). * Situado en la imagen en x1: 1412; x2: 3313; y1: 323; y2: 862 Número de coches: 5


```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-6-01.jpg' =====
1) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 64.33%)
2) La imagen corresponde a 'coche en una calle' (con una certeza del 60.81%)
3) La imagen corresponde a 'un coche en una calle' (con una certeza del 60.71%)
4) La imagen corresponde a 'una camioneta estacionada en la calle' (con una certeza del
59.52%)
5) La imagen corresponde a 'camioneta en la calle' (con una certeza del 56.75%)

===== Objetos detectados en la imagen =====
  1.-- car (con una certeza del 51.10%).
           * Situado en la imagen en x1: 1238; x2: 1664; y1: 1004; y2: 1312
  2.-- car (con una certeza del 64.00%).
           * Situado en la imagen en x1: 1984; x2: 2341; y1: 975; y2: 1286
  3.-- car (con una certeza del 71.90%).
           * Situado en la imagen en x1: 1412; x2: 2169; y1: 978; y2: 1556
  4.-- car (con una certeza del 82.60%).
           * Situado en la imagen en x1: 2278; x2: 2925; y1: 978; y2: 1517
  5.-- car (con una certeza del 83.30%).
           * Situado en la imagen en x1: 2865; x2: 3496; y1: 968; y2: 1472
  6.-- car (con una certeza del 79.40%).
           * Situado en la imagen en x1: 8; x2: 1451; y1: 915; y2: 1773
Número de coches:  6
```


```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-6-02.jpg' =====
1) La imagen corresponde a 'una calle con edificios de fondo' (con una certeza del 69.50%)
2) La imagen corresponde a 'una calle con coches y edificios' (con una certeza del 66.67%)
3) La imagen corresponde a 'una calle con edificios' (con una certeza del 66.57%)
4) La imagen corresponde a 'una calle con un edificio de ladrillo' (con una certeza del
66.47%)
5) La imagen corresponde a 'una calle con coches' (con una certeza del 63.80%)

===== Objetos detectados en la imagen =====
  1.-- car (con una certeza del 74.20%).
           * Situado en la imagen en x1: 0; x2: 302; y1: 861; y2: 1046
  2.-- car (con una certeza del 80.20%).
           * Situado en la imagen en x1: 371; x2: 940; y1: 854; y2: 1039
  3.-- car (con una certeza del 74.10%).
           * Situado en la imagen en x1: 947; x2: 1403; y1: 862; y2: 1032
  4.-- car (con una certeza del 80.80%).
           * Situado en la imagen en x1: 1421; x2: 1802; y1: 853; y2: 1009
  5.-- car (con una certeza del 69.60%).
           * Situado en la imagen en x1: 1813; x2: 2058; y1: 862; y2: 990
  6.-- car (con una certeza del 56.10%).
           * Situado en la imagen en x1: 2215; x2: 2324; y1: 868; y2: 983
Número de coches:  6
```


```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-6-03.jpg' =====
1) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 69.11%)
2) La imagen corresponde a 'edificio en frente de un coche' (con una certeza del 65.81%)
3) La imagen corresponde a 'coche en una calle' (con una certeza del 64.35%)
4) La imagen corresponde a 'un coche en una calle' (con una certeza del 64.25%)
5) La imagen corresponde a 'un coche en la calle' (con una certeza del 64.15%)

===== Objetos detectados en la imagen =====
  1.-- car (con una certeza del 68.90%).
           * Situado en la imagen en x1: 798; x2: 1212; y1: 1336; y2: 1564
  2.-- car (con una certeza del 60.20%).
           * Situado en la imagen en x1: 1507; x2: 1854; y1: 1285; y2: 1643
  3.-- car (con una certeza del 55.60%).
           * Situado en la imagen en x1: 0; x2: 209; y1: 1290; y2: 1802
  4.-- car (con una certeza del 85.40%).
           * Situado en la imagen en x1: 17; x2: 921; y1: 1329; y2: 1952
  5.-- car (con una certeza del 88.90%).
           * Situado en la imagen en x1: 862; x2: 1794; y1: 1406; y2: 2126
  6.-- car (con una certeza del 82.30%).
           * Situado en la imagen en x1: 1770; x2: 3385; y1: 1207; y2: 2334
Número de coches:  6
```


```
===== DESCRIPCIÓN DE LA IMAGEN 'Coches-6-04.jpg' =====
1) La imagen corresponde a 'un coche en una calle con nieve' (con una certeza del 51.27%)
2) La imagen corresponde a 'coche cubierto de nieve' (con una certeza del 51.17%)
3) La imagen corresponde a 'coche cubierto con nieve' (con una certeza del 51.07%)
4) La imagen corresponde a 'un coche estacion en el suelo' (con una certeza del 50.97%)
5) La imagen corresponde a 'un coche con nieve' (con una certeza del 50.87%)

===== Objetos detectados en la imagen =====
  1.-- car (con una certeza del 67.50%).
           * Situado en la imagen en x1: 30; x2: 262; y1: 837; y2: 1049
  2.-- car (con una certeza del 54.90%).
           * Situado en la imagen en x1: 1165; x2: 1393; y1: 889; y2: 1001
  3.-- car (con una certeza del 60.30%).
           * Situado en la imagen en x1: 1883; x2: 2140; y1: 873; y2: 1014
  4.-- car (con una certeza del 71.30%).
           * Situado en la imagen en x1: 362; x2: 864; y1: 839; y2: 1203
  5.-- Land vehicle (con una certeza del 85.30%).
           * Situado en la imagen en x1: 2015; x2: 2784; y1: 786; y2: 1140
  6.-- car (con una certeza del 74.70%).
           * Situado en la imagen en x1: 772; x2: 1346; y1: 952; y2: 1462
  7.-- car (con una certeza del 71.90%).
           * Situado en la imagen en x1: 1043; x2: 2677; y1: 1010; y2: 1819
Número de coches:  6
```

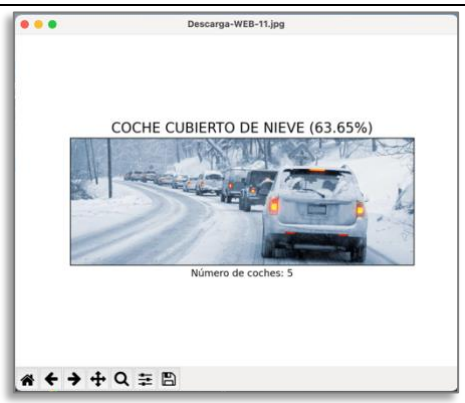
	===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-1.jpg' ===== 1) La imagen corresponde a 'una persona en frente de un coche' (con una certeza del 59.54%) 2) La imagen corresponde a 'una persona caminando en la calle junto a un coche' (con una certeza del 58.98%) 3) La imagen corresponde a 'una persona en frente de un coche' (con una certeza del 58.88%) 4) La imagen corresponde a 'mujer sentada en un coche' (con una certeza del 57.15%) 5) La imagen corresponde a 'una persona sentada en un coche' (con una certeza del 57.05%) ===== Objetos detectados en la imagen ===== 1.-- Wheel (con una certeza del 64.10%). * Situado en la imagen en x1: 71; x2: 210; y1: 219; y2: 360 2.-- person (con una certeza del 69.50%). * Situado en la imagen en x1: 424; x2: 587; y1: 9; y2: 371 3.-- car (con una certeza del 69.60%). * Situado en la imagen en x1: 243; x2: 660; y1: 13; y2: 359 Número de coches: 1
	===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-2.jpg' ===== 1) La imagen corresponde a 'una carretera con autos' (con una certeza del 64.77%) 2) La imagen corresponde a 'una carretera con coches' (con una certeza del 64.67%) 3) La imagen corresponde a 'un coche deportivo en una carretera' (con una certeza del 64.57%) 4) La imagen corresponde a 'una carretera con autos estacionados' (con una certeza del 64.47%) 5) La imagen corresponde a 'un coche en una carretera' (con una certeza del 64.37%) ===== Objetos detectados en la imagen ===== 1.-- car (con una certeza del 76.70%). * Situado en la imagen en x1: 360; x2: 466; y1: 9; y2: 98 2.-- Van (con una certeza del 61.20%). * Situado en la imagen en x1: 826; x2: 1004; y1: 115; y2: 265 3.-- car (con una certeza del 87.60%). * Situado en la imagen en x1: 175; x2: 336; y1: 175; y2: 325 4.-- car (con una certeza del 78.10%). * Situado en la imagen en x1: 541; x2: 705; y1: 162; y2: 312 5.-- car (con una certeza del 83.20%). * Situado en la imagen en x1: 909; x2: 1126; y1: 287; y2: 484 6.-- car (con una certeza del 61.30%). * Situado en la imagen en x1: 585; x2: 829; y1: 554; y2: 644 Número de coches: 6
	===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-3.jpg' ===== 1) La imagen corresponde a 'un coche deportivo en una carretera' (con una certeza del 77.29%) 2) La imagen corresponde a 'coche en una carretera' (con una certeza del 73.17%) 3) La imagen corresponde a 'un coche en una carretera' (con una certeza del 73.07%) 4) La imagen corresponde a 'un coche en la carretera' (con una certeza del 72.97%) 5) La imagen corresponde a 'coche circulando en la carretera' (con una certeza del 69.73%) ===== Objetos detectados en la imagen ===== 1.-- car (con una certeza del 90.30%). * Situado en la imagen en x1: 221; x2: 944; y1: 330; y2: 683 Número de coches: 1
	===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-4.jpg' ===== 1) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 65.58%) 2) La imagen corresponde a 'coche en una calle' (con una certeza del 62.06%) 3) La imagen corresponde a 'un coche deportivo de color rojo' (con una certeza del 61.78%) 4) La imagen corresponde a 'un coche en la calle' (con una certeza del 61.68%) 5) La imagen corresponde a 'un coche en una calle' (con una certeza del 61.58%) ===== Objetos detectados en la imagen ===== 1.-- person (con una certeza del 82.30%). * Situado en la imagen en x1: 499; x2: 663; y1: 274; y2: 817 2.-- car (con una certeza del 68.40%). * Situado en la imagen en x1: 205; x2: 1228; y1: 388; y2: 792 Número de coches: 1

	===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-5.jpg' ===== 1) La imagen corresponde a 'una carretera con coches' (con una certeza del 66.27%) 2) La imagen corresponde a 'una carretera con coches circulando' (con una certeza del 66.01%) 3) La imagen corresponde a 'una carretera con autos' (con una certeza del 65.52%) 4) La imagen corresponde a 'vista de una carretera con coches' (con una certeza del 63.86%) 5) La imagen corresponde a 'carretera en medio de la calle' (con una certeza del 63.76%) ===== Objetos detectados en la imagen ===== 1.-- stop sign (con una certeza del 61.60%). * Situado en la imagen en x1: 540; x2: 609; y1: 17; y2: 65 2.-- stop sign (con una certeza del 55.70%). * Situado en la imagen en x1: 423; x2: 565; y1: 52; y2: 89 3.-- car (con una certeza del 65.40%). * Situado en la imagen en x1: 554; x2: 591; y1: 207; y2: 238 4.-- car (con una certeza del 57.30%). * Situado en la imagen en x1: 691; x2: 744; y1: 200; y2: 232 5.-- stop sign (con una certeza del 50.80%). * Situado en la imagen en x1: 413; x2: 552; y1: 16; y2: 90 Número de coches: 2
	===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-6.jpg' ===== 1) La imagen corresponde a 'imagen de la pantalla de un video juego de un carro' (con una certeza del 50.02%) 2) La imagen corresponde a 'un dibujo de un coche' (con una certeza del 49.92%) 3) La imagen corresponde a 'dibujo digital de un coche' (con una certeza del 49.82%) 4) La imagen corresponde a 'dibujo de un coche' (con una certeza del 49.72%) 5) La imagen corresponde a 'un coche de carreras en una pista de aterrizaje' (con una certeza del 49.62%) ===== Objetos detectados en la imagen ===== 1.-- car (con una certeza del 66.00%). * Situado en la imagen en x1: 201; x2: 505; y1: 376; y2: 588 2.-- car (con una certeza del 88.40%). * Situado en la imagen en x1: 568; x2: 986; y1: 316; y2: 571 Número de coches: 2
	===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-7.jpg' ===== 1) La imagen corresponde a 'una camioneta estacionada en la arena' (con una certeza del 57.93%) 2) La imagen corresponde a 'un grupo de personas en el desierto' (con una certeza del 57.83%) 3) La imagen corresponde a 'un grupo de personas en un desierto' (con una certeza del 57.73%) 4) La imagen corresponde a 'grupo de personas en el desierto' (con una certeza del 57.63%) 5) La imagen corresponde a 'grupo de personas en un desierto' (con una certeza del 57.53%) ===== Objetos detectados en la imagen ===== 1.-- Van (con una certeza del 51.30%). * Situado en la imagen en x1: 344; x2: 448; y1: 250; y2: 307 2.-- Van (con una certeza del 55.90%). * Situado en la imagen en x1: 483; x2: 567; y1: 242; y2: 309 3.-- person (con una certeza del 65.80%). * Situado en la imagen en x1: 56; x2: 85; y1: 282; y2: 342 Número de coches: 2
	===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-8.jpg' ===== 1) La imagen corresponde a 'una camioneta en un desierto' (con una certeza del 65.75%) 2) La imagen corresponde a 'camioneta en un desierto' (con una certeza del 65.64%) 3) La imagen corresponde a 'una camioneta estacionada al lado de una montaña' (con una certeza del 53.49%) 4) La imagen corresponde a 'vehículo parado en una montaña' (con una certeza del 53.39%) 5) La imagen corresponde a 'una camioneta estacionada en una montaña' (con una certeza del 53.29%) ===== Objetos detectados en la imagen ===== 1.-- Van (con una certeza del 58.20%). * Situado en la imagen en x1: 331; x2: 474; y1: 264; y2: 339 Número de coches: 1


```
===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-9.jpg' =====
1) La imagen corresponde a 'un coche deportivo de color verde' (con una certeza del 71.34%)
2) La imagen corresponde a 'coche deportivo de color verde' (con una certeza del 70.38%)
3) La imagen corresponde a 'un coche deportivo de color verde estacionado' (con una certeza
del 68.63%)
4) La imagen corresponde a 'un coche deportivo de color rojo' (con una certeza del 68.53%)
5) La imagen corresponde a 'un coche deportivo de color verde sobre una superficie' (con
una certeza del 63.74%)

===== Objetos detectados en la imagen =====
  1.-- race car (con una certeza del 60.10%).
          * Situado en la imagen en x1: 35; x2: 335; y1: 135; y2: 244
  2.-- car (con una certeza del 85.60%).
          * Situado en la imagen en x1: 226; x2: 576; y1: 124; y2: 314
  3.-- car (con una certeza del 90.10%).
          * Situado en la imagen en x1: 495; x2: 864; y1: 162; y2: 382
Número de coches:  3
```


```
===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-9.jpg' =====
1) La imagen corresponde a 'un coche deportivo de color verde' (con una certeza del 71.34%)
2) La imagen corresponde a 'coche deportivo de color verde' (con una certeza del 70.38%)
3) La imagen corresponde a 'un coche deportivo de color verde estacionado' (con una certeza
del 68.63%)
4) La imagen corresponde a 'un coche deportivo de color rojo' (con una certeza del 68.53%)
5) La imagen corresponde a 'un coche deportivo de color verde sobre una superficie' (con
una certeza del 63.74%)

===== Objetos detectados en la imagen =====
  1.-- race car (con una certeza del 60.10%).
          * Situado en la imagen en x1: 35; x2: 335; y1: 135; y2: 244
  2.-- car (con una certeza del 85.60%).
          * Situado en la imagen en x1: 226; x2: 576; y1: 124; y2: 314
  3.-- car (con una certeza del 90.10%).
          * Situado en la imagen en x1: 495; x2: 864; y1: 162; y2: 382
Número de coches:  3
```


```
===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-10.jpg' =====
1) La imagen corresponde a 'un coche en una carretera' (con una certeza del 66.67%)
2) La imagen corresponde a 'coche en una carretera' (con una certeza del 64.84%)
3) La imagen corresponde a 'una carretera con autos' (con una certeza del 63.77%)
4) La imagen corresponde a 'un coche en la carretera' (con una certeza del 63.67%)
5) La imagen corresponde a 'un coche deportivo en una carretera' (con una certeza del
63.57%)

===== Objetos detectados en la imagen =====
  1.-- car (con una certeza del 82.20%).
          * Situado en la imagen en x1: 391; x2: 489; y1: 9; y2: 93
  2.-- car (con una certeza del 83.50%).
          * Situado en la imagen en x1: 217; x2: 309; y1: 28; y2: 104
  3.-- car (con una certeza del 87.20%).
          * Situado en la imagen en x1: 150; x2: 291; y1: 170; y2: 301
  4.-- car (con una certeza del 84.30%).
          * Situado en la imagen en x1: 400; x2: 571; y1: 206; y2: 376
Número de coches:  4
```



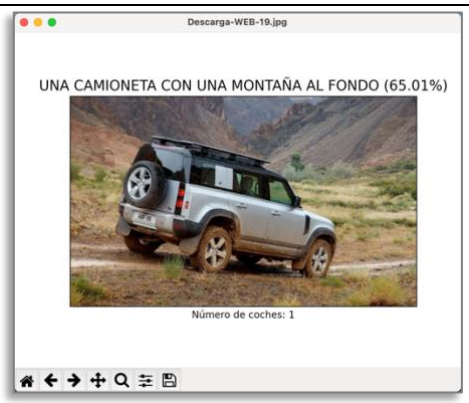

```
===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-11.jpg' =====
1) La imagen corresponde a 'coche cubierto de nieve' (con una certeza del 63.65%)
2) La imagen corresponde a 'camioneta blanca en la nieve' (con una certeza del 59.17%)
3) La imagen corresponde a 'un coche en la nieve' (con una certeza del 59.07%)
4) La imagen corresponde a 'coche cubierto con nieve' (con una certeza del 58.97%)
5) La imagen corresponde a 'una camioneta estacionada en la nieve' (con una certeza del
58.87%)

===== Objetos detectados en la imagen =====
  1.-- car (con una certeza del 59.00%).
          * Situado en la imagen en x1: 349; x2: 399; y1: 120; y2: 171
  2.-- car (con una certeza del 73.10%).
          * Situado en la imagen en x1: 396; x2: 479; y1: 112; y2: 187
  3.-- car (con una certeza del 55.40%).
          * Situado en la imagen en x1: 487; x2: 566; y1: 102; y2: 207
  4.-- car (con una certeza del 67.80%).
          * Situado en la imagen en x1: 508; x2: 664; y1: 93; y2: 240
  5.-- car (con una certeza del 89.10%).
          * Situado en la imagen en x1: 599; x2: 973; y1: 81; y2: 396
Número de coches:  5
```

	===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-12.jpg' ===== 1) La imagen corresponde a 'un coche en un bosque' (con una certeza del 57.10%) 2) La imagen corresponde a 'un coche en el bosque' (con una certeza del 57.00%) 3) La imagen corresponde a 'un coche estacion en el bosque' (con una certeza del 56.90%) 4) La imagen corresponde a 'imagen en blanco y negro de un coche en un bosque' (con una certeza del 46.32%) 5) La imagen corresponde a 'imagen de la pantalla de un video juego de un carro' (con una certeza del 43.44%) ===== Objetos detectados en la imagen ===== 1.-- car (con una certeza del 89.80%). * Situado en la imagen en x1: 104; x2: 474; y1: 258; y2: 434 Número de coches: 1
	===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-13.jpg' ===== 1) La imagen corresponde a 'una camioneta estacionada al lado de un río' (con una certeza del 62.92%) 2) La imagen corresponde a 'un río al lado de una camioneta' (con una certeza del 59.80%) 3) La imagen corresponde a 'una camioneta estacionada al lado de un lago' (con una certeza del 59.70%) 4) La imagen corresponde a 'un río al lado de un coche' (con una certeza del 59.60%) 5) La imagen corresponde a 'un par de personas junto a un cuerpo de agua junto a una camioneta' (con una certeza del 54.35%) ===== Objetos detectados en la imagen ===== 1.-- car (con una certeza del 77.90%). * Situado en la imagen en x1: 131; x2: 424; y1: 147; y2: 294 Número de coches: 1
	===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-14.jpg' ===== 1) La imagen corresponde a 'una persona con un caballo' (con una certeza del 68.01%) 2) La imagen corresponde a 'una persona en un caballo' (con una certeza del 65.77%) 3) La imagen corresponde a 'un coche de caballo jalando a una persona' (con una certeza del 65.67%) 4) La imagen corresponde a 'una persona con un caballo' (con una certeza del 65.57%) 5) La imagen corresponde a 'una persona con un caballo en un área abierta' (con una certeza del 63.79%) ===== Objetos detectados en la imagen ===== 1.-- person (con una certeza del 79.60%). * Situado en la imagen en x1: 420; x2: 476; y1: 88; y2: 235 2.-- person (con una certeza del 81.30%). * Situado en la imagen en x1: 521; x2: 573; y1: 74; y2: 243 3.-- horse (con una certeza del 81.90%). * Situado en la imagen en x1: 365; x2: 510; y1: 113; y2: 336 4.-- car (con una certeza del 80.40%). * Situado en la imagen en x1: 20; x2: 305; y1: 176; y2: 325 5.-- horse (con una certeza del 88.10%). * Situado en la imagen en x1: 439; x2: 662; y1: 118; y2: 349 Número de coches: 1
	===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-15.jpg' ===== 1) La imagen corresponde a 'un carro cubierto de nieve' (con una certeza del 66.93%) 2) La imagen corresponde a 'coche cubierto de nieve' (con una certeza del 66.83%) 3) La imagen corresponde a 'coche cubierto con nieve' (con una certeza del 66.73%) 4) La imagen corresponde a 'un coche en la nieve' (con una certeza del 65.79%) 5) La imagen corresponde a 'un coche con nieve' (con una certeza del 65.69%) ===== Objetos detectados en la imagen ===== 1.-- snowplow (con una certeza del 55.50%). * Situado en la imagen en x1: 444; x2: 592; y1: 323; y2: 458 2.-- car (con una certeza del 80.10%). * Situado en la imagen en x1: 721; x2: 990; y1: 319; y2: 521 Número de coches: 1


```
===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-16.jpg' =====
1) La imagen corresponde a 'coche aparcado en la calle' (con una certeza del 62.72%)
2) La imagen corresponde a 'coche en una calle' (con una certeza del 59.01%)
3) La imagen corresponde a 'un coche deportivo en una calle' (con una certeza del 58.91%)
4) La imagen corresponde a 'un coche en una calle' (con una certeza del 58.81%)
5) La imagen corresponde a 'un coche en la calle' (con una certeza del 58.71%)

===== Objetos detectados en la imagen =====
  1.-- person (con una certeza del 67.50%).
          * Situado en la imagen en x1: 506; x2: 594; y1: 166; y2: 462
  2.-- Tire (con una certeza del 72.60%).
          * Situado en la imagen en x1: 203; x2: 321; y1: 358; y2: 475
  3.-- car (con una certeza del 79.80%).
          * Situado en la imagen en x1: 29; x2: 588; y1: 198; y2: 479
Número de coches:  1
```


```
===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-17.jpg' =====
1) La imagen corresponde a 'una carretera con coches circulando en la calle' (con una
certeza del 67.54%)
2) La imagen corresponde a 'una carretera con coches' (con una certeza del 66.26%)
3) La imagen corresponde a 'una carretera con autos' (con una certeza del 66.01%)
4) La imagen corresponde a 'una carretera con coches circulando' (con una certeza del
65.81%)
5) La imagen corresponde a 'una carretera en el tráfico' (con una certeza del 63.25%)

===== Objetos detectados en la imagen =====
  1.-- car (con una certeza del 52.40%).
          * Situado en la imagen en x1: 1199; x2: 1253; y1: 124; y2: 174
  2.-- car (con una certeza del 67.20%).
          * Situado en la imagen en x1: 583; x2: 658; y1: 212; y2: 273
  3.-- car (con una certeza del 74.10%).
          * Situado en la imagen en x1: 723; x2: 808; y1: 267; y2: 343
  4.-- car (con una certeza del 81.50%).
          * Situado en la imagen en x1: 440; x2: 593; y1: 391; y2: 530
  5.-- car (con una certeza del 89.50%).
          * Situado en la imagen en x1: 727; x2: 897; y1: 528; y2: 697
Número de coches:  5
```


```
===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-18.jpg' =====
1) La imagen corresponde a 'una carretera cerca de un puente' (con una certeza del 63.62%)
2) La imagen corresponde a 'una carretera con coches y edificios' (con una certeza del
63.52%)
3) La imagen corresponde a 'una carretera con luces' (con una certeza del 55.99%)
4) La imagen corresponde a 'un barco en el agua junto a una carretera' (con una certeza del
55.89%)
5) La imagen corresponde a 'un puente sobre el agua' (con una certeza del 55.79%)

===== Objetos detectados en la imagen =====
  1.-- car (con una certeza del 62.00%).
          * Situado en la imagen en x1: 606; x2: 679; y1: 312; y2: 363
  2.-- Land vehicle (con una certeza del 51.30%).
          * Situado en la imagen en x1: 815; x2: 871; y1: 301; y2: 348
  3.-- car (con una certeza del 80.80%).
          * Situado en la imagen en x1: 222; x2: 385; y1: 287; y2: 433
Número de coches:  2
```




```
===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-19.jpg' =====
1) La imagen corresponde a 'una camioneta con una montaña al fondo' (con una certeza del
65.01%)
2) La imagen corresponde a 'una camioneta en una montaña' (con una certeza del 61.74%)
3) La imagen corresponde a 'una camioneta estacionada al lado de una montaña' (con una
certeza del 61.64%)
4) La imagen corresponde a 'una camioneta estacionada en una montaña' (con una certeza del
61.54%)
5) La imagen corresponde a 'camioneta en una montaña' (con una certeza del 61.44%)

===== Objetos detectados en la imagen =====
  1.-- Tire (con una certeza del 62.20%).
          * Situado en la imagen en x1: 99; x2: 171; y1: 254; y2: 296
  2.-- Tire (con una certeza del 67.10%).
          * Situado en la imagen en x1: 436; x2: 495; y1: 265; y2: 345
  3.-- Tire (con una certeza del 62.70%).
          * Situado en la imagen en x1: 97; x2: 181; y1: 109; y2: 209
  4.-- Tire (con una certeza del 55.00%).
          * Situado en la imagen en x1: 230; x2: 314; y1: 242; y2: 332
  5.-- Van (con una certeza del 67.20%).
          * Situado en la imagen en x1: 85; x2: 509; y1: 71; y2: 347
Número de coches:  1
```


```
===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-20.jpg' =====
1) La imagen corresponde a 'una persona sentado en un coche' (con una certeza del 52.81%)
2) La imagen corresponde a 'una persona sentada en un coche' (con una certeza del 52.71%)
3) La imagen corresponde a 'mujer sentada en un coche' (con una certeza del 52.47%)
4) La imagen corresponde a 'una persona encima de un coche' (con una certeza del 52.37%)
5) La imagen corresponde a 'un par de personas en un coche' (con una certeza del 50.92%)

===== Objetos detectados en la imagen =====
  1.-- Luggage and bags (con una certeza del 67.90%).
          * Situado en la imagen en x1: 542; x2: 660; y1: 770; y2: 938
  2.-- person (con una certeza del 72.50%).
          * Situado en la imagen en x1: 306; x2: 569; y1: 530; y2: 1142
  3.-- car (con una certeza del 63.90%).
          * Situado en la imagen en x1: 0; x2: 868; y1: 280; y2: 1139
Número de coches:  1
```


```
===== DESCRIPCIÓN DE LA IMAGEN 'Descarga-WEB-21.jpg' =====
1) La imagen corresponde a 'un coche deportivo en una montaña' (con una certeza del 64.22%)
2) La imagen corresponde a 'un coche en una montaña' (con una certeza del 58.02%)
3) La imagen corresponde a 'coche en una montaña' (con una certeza del 57.62%)
4) La imagen corresponde a 'una persona en frente de un coche' (con una certeza del 57.52%)
5) La imagen corresponde a 'personas en frente de un coche' (con una certeza del 57.42%)

===== Objetos detectados en la imagen =====
  1.-- person (con una certeza del 84.80%).
          * Situado en la imagen en x1: 128; x2: 189; y1: 210; y2: 369
  2.-- person (con una certeza del 88.60%).
          * Situado en la imagen en x1: 202; x2: 309; y1: 174; y2: 381
  3.-- hatchback (con una certeza del 65.60%).
          * Situado en la imagen en x1: 290; x2: 771; y1: 174; y2: 393
Número de coches:  1
```


```
===== DESCRIPCIÓN DE LA IMAGEN 'Gente calle Libreros.jpg' =====
1) La imagen corresponde a 'un grupo de personas caminando en una plaza' (con una certeza
del 73.87%)
2) La imagen corresponde a 'un grupo de personas caminando en la calle de una ciudad' (con
una certeza del 73.08%)
3) La imagen corresponde a 'un grupo de personas caminando en la calle cerca de un
edificio' (con una certeza del 72.78%)
4) La imagen corresponde a 'un grupo de personas en una plaza' (con una certeza del 72.68%)
5) La imagen corresponde a 'un grupo de personas caminando en la calle cerca de unos
edificios' (con una certeza del 71.74%)

===== Objetos detectados en la imagen =====
  1.-- person (con una certeza del 72.30%).
          * Situado en la imagen en x1: 574; x2: 831; y1: 1463; y2: 2274
  2.-- person (con una certeza del 76.40%).
          * Situado en la imagen en x1: 1488; x2: 1867; y1: 1448; y2: 2552
  3.-- person (con una certeza del 81.30%).
          * Situado en la imagen en x1: 1806; x2: 2187; y1: 1453; y2: 2542
  4.-- person (con una certeza del 86.30%).
          * Situado en la imagen en x1: 2512; x2: 2904; y1: 1407; y2: 2926
  5.-- person (con una certeza del 85.70%).
          * Situado en la imagen en x1: 3191; x2: 3636; y1: 1484; y2: 2791
  6.-- person (con una certeza del 83.60%).
          * Situado en la imagen en x1: 2862; x2: 3228; y1: 1864; y2: 2834
Número de coches:  0
```

ANEXO 2: Código fuente de la aplicación de prueba en PYTHON

```
# ======================== IMPORTACIONES DE BIBLIOTECAS O MODULOS ================================================
# ····························· Bibliotecas sistema operativo ·····························································
import os
import sys
# ····························· Modulos tkinter ·····························································
from tkinter import *
from tkinter import messagebox as ventana_mensaje
from tkinter import filedialog, Tk  # lo usamos para presentar la ventana de selección del fichero a analizar
# ····························· ····· ·····  ·····  ·····  ·····  ·····  ·····  ·····  ·····  ·····  ·····  ·····  ·····
from PIL import Image as Imagen_foto_PIL  # lo usamos para abrir el fichero de la foto y volcarlo a la imagen final
from io import BytesIO  # BytesIO permite trabajar con Bytes en memoria, necesario para la salida
# ····························· ····· ·····  ·····  ·····  ·····  ·····  ·····  ·····  ·····  ·····  ·····  ·····  ·····
import requests
import matplotlib.pyplot as imagen_generada  # Genera imagen de salida con foto y los textos con información

# ========================  VARIABLES ===========================================================================
# ····················· Credenciales de COMPUTER VISION (AZURE)
AZURE_mi_clave = "X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X"
AZURE_mi_URL = "https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/"


# ======================== FUNCIONES ===========================================================================
# ··············· VENTANA DE INICIO DEL PROGRAMA ·····························································
def pantalla_inicio():
    ventana = Tk()
    ventana.title("Práctica del TFG de Alvaro Fernández-Taviro Gracia")
    ventana.geometry('700x500+200+100')
    ventana.config(background="#617858")
    ventana.resizable(False, False)
    ventana.protocol("WM_DELETE_WINDOW", salir)

    imagen = PhotoImage(file='Logo-UAH.png')

    fondo = Label(ventana, image=imagen).place(x=200, y=100)

    texto1 = Label(ventana, text="GRADO DE SISTEMAS DE INFORMACIÓN",
                   font=("Agency FB", 20, "bold"),
                   background="#617858",
                   fg="yellow").place(x=170, y=5)

    texto2 = Label(ventana, text="Trabajo de Fin de Grado",
                   font=("Agency FB", 15, "italic"),
                   background="#617858",
                   fg="yellow").place(x=260, y=35)

    texto3 = Label(ventana, text="Álvaro Fernández-Taviro Gracia",
                   font=("Agency FB", 15, "italic"),
                   background="#617858",
                   fg="yellow").place(x=235, y=65)

    boton_salir = Button(ventana, text="Salir",
                         font=("Agency FB", 20),
                         command=quit,
                         foreground="black",                # LETRAS
                         activeforeground="gray",           # LETRAS PULSADAS
                         borderwidth=4,
                         width=15).place(x=150, y=440)

    boton_continuar = Button(ventana, text="Comenzar",
                             font=("Agency FB", 20),
                             command=ventana.destroy,
                             foreground="black",            # LETRAS
                             activeforeground="gray",       # LETRAS PULSADAS
                             borderwidth=4,
                             width=15).place(x=400, y=440)

    ventana.mainloop()


# ··············· Salida del programa si se elige 'CANCELAR' en la ventana de inicio····························
def salir():

    ventana = Tk()
    ventana_mensaje.showinfo('Saliendo del programa', 'Pulsar para salir')
    ventana.destroy()
    sys.exit()


# ··········· VENTANA PARA SELECCIÓN DE LA IMAGEN QUE SE VA A ANALIZAR ····································
```

```
def seleccionar_fichero():

    ventana = Tk()
    ventana.withdraw()
    ventana.filename = filedialog.askopenfilename(initialdir="/xcvfg", title="Select file")

    foto_seleccionada = ventana.filename
    ventana.destroy()

    return foto_seleccionada


# ····· OBTENCIÓN DE DESCRIPCIONES ····· ·····
def obtener_descripciones(nombre_fichero, foto_a_analizar):
    descripcion_imagen = ""
    mejor_descripcion = ""

    print("===== DESCRIPCIÓN DE LA IMAGEN '" + nombre_fichero + "' =====")

    # Se invoca a la API para conseguir [5] descripciones (usamos 5 para mostrar la funcionalidad)

    try:
        URL_descripcion = AZURE_mi_URL + "vision/v3.1/describe"
        cabeceras = {'Ocp-Apim-Subscription-Key': AZURE_mi_clave, 'Content-Type': 'application/octet-stream'}
        parametros = {'maxCandidates': '5', 'language': 'es'}
        datos_imagen = open(foto_a_analizar, "rb").read()

        descripcion_imagen = requests.post(URL_descripcion, headers=cabeceras, params=parametros, data=datos_imagen)

    except:  # se gestiona la salida ordenada si se produce error en tamaño fichero, fallo autentificacion u otro motivo
        print("¡Se ha producido un error al intenta el fichero seleccionado!")
        quit()

    # Obtener el texto de la(s) descripción(es),  con el nivel de confianza otorgado a cada descripción
    if len(descripcion_imagen.json()["description"]["captions"]) == 0:
        print("No hay descripción disponible.")
    else:
        confianza = 0
        contador = 1
        descripciones_encontradas = descripcion_imagen.json()["description"]["captions"]

        print()
        for descripcion_generada in descripciones_encontradas:
            print("{}) La imagen corresponde a \033[1;33m '{}' \033[0;m (con una certeza del {:.2f}%)".
                  format(contador, descripcion_generada["text"], descripcion_generada["confidence"] * 100))

            # la descricpción más fiable se almacena para presentarla en la imagen de la salida final
            if descripcion_generada["confidence"] > confianza:
                confianza = descripcion_generada["confidence"]
                mejor_descripcion = descripcion_generada["text"] + \
                    " (" + str(round((descripcion_generada["confidence"] * 100), 2)) + "%)"

            contador += 1

        print()
        print("La descripcion con mas fiabilidad es :"+mejor_descripcion)
        print()
    return mejor_descripcion


# ····· OBJETOS DETECTADOS EN LA IMAGEN ····· ·····
def obtener_objetos(nombre_fichero, foto_a_analizar):
    numero_coches_encontrado = 0
    tipos_de_coches = ["CAR", "VAN", "TAXI", "HATCHBACK", "RACE CAR"]

    print("===== OBJETOS ENCONTRADOS EN LA IMAGEN '" + nombre_fichero + "' =====")

    # Invocar  API

    URL_descripcion = AZURE_mi_URL + "vision/v3.1/detect"
    cabeceras = {'Ocp-Apim-Subscription-Key': AZURE_mi_clave, 'Content-Type': 'application/octet-stream'}
    datos_imagen = open(foto_a_analizar, "rb").read()

    objetos_encontrados = requests.post(URL_descripcion, headers=cabeceras, data=datos_imagen)
    print()

    # Resultados obtenidos y coordenadas de donde se encuentran (sirve de contar nº de coches)
    if len(objetos_encontrados.json()) == 0:
        print("No existe ningún objeto.")
    else:
        numero_coches_encontrado = 0
        contador = 1

        for datos_objeto in objetos_encontrados.json()["objects"]:
```

```
            print("  {}.-- \033[1;33m {} \033[0;m (con una certeza del {:.2f}%).".format(
                contador, datos_objeto["object"], datos_objeto["confidence"] * 100, ))

            print("           * Situado en la imagen en x1: {}; x2: {}; y1: {}; y2: {}".format(
                datos_objeto["rectangle"]["x"],
                datos_objeto["rectangle"]["x"] + datos_objeto["rectangle"]["w"],
                datos_objeto["rectangle"]["y"],
                datos_objeto["rectangle"]["y"] + datos_objeto["rectangle"]["h"]))

            # CONTAMOS OBJETOS QUE SON CAR, TAXI, RACE CAR, VAN o HATCHBACK ----- ¿HAY MÁS TIPOS DE COCHES? -----
            if datos_objeto["object"].upper() in tipos_de_coches:
                numero_coches_encontrado = numero_coches_encontrado + 1

            contador += 1

        print("\033[1;33m Número de coches:  {}\033[0;m ".format(numero_coches_encontrado))
    print()
    return numero_coches_encontrado


# ·····------ VENTANA DE PRESENTACIÓN DE RESULTADOS ·····------
def mostrar_salida(foto, nombre, descripcion, numero):

    imagen_con_read = open(foto, "rb").read()                   # abrir fichero de la foto
    imagen_buffer = BytesIO(imagen_con_read)                    # imagen en buffer en formato binario
    imagen_incrustada = Imagen_foto_PIL.open(imagen_buffer)     # abre la imagen

    imagen_generada.figure(num=nombre)                          # muestro nombre de fichero en título de la ventana
    imagen_generada.imshow(imagen_incrustada)                   # foto analizada (formato binario)
    imagen_generada.axis("on")                                  # para mostrar texto nº de coches
    imagen_generada.xticks([])                                  # ocultar marcas eje x
    imagen_generada.yticks([])                                  # ocultar marcas eje y
    imagen_generada.title(descripcion.upper(), size="x-large")  # texto descripcion ofrecida por AZURE
    imagen_generada.xlabel('Número de coches: ' + str(numero))  # texto con nº de coches
    imagen_generada.show()


# ==========================================================================================================
# ================== M A I N ===============================================================================
# ==========================================================================================================
def main():
    direccion_partida = ""

    pantalla_inicio()
    foto_a_analizar = seleccionar_fichero()

    # La variable "foto_a_analizar" contiene la ruta completa del fichero.
    # Necesitamos obtente el nombre del fichero a partir de la ruta completa
    if not foto_a_analizar:
        print("¡No se ha seleccionado ninguna foto")
        quit()
    else:
        direccion_partida = os.path.split(foto_a_analizar)

    # Descartamos el elemento [0] porque no necesitamos la ruta
    # Usamos solo el elemento [1] porque es el que contiene el nomebre del fichero
    nombre_fichero = direccion_partida[1]

    # llamadas a las funciones que devuelven los valores que proporciona Computer Vision
    mejor_descripcion = obtener_descripciones(nombre_fichero, foto_a_analizar)
    numero_coches = obtener_objetos(nombre_fichero, foto_a_analizar)

    # presentación de imagen de salida con los datos requeridos
    mostrar_salida(foto_a_analizar, nombre_fichero, mejor_descripcion, numero_coches)

# ·····
if __name__ == "__main__":
    main()
```

ANEXO 3: EJEMPLOS DE USO DE LAS FUNCIONALIDADES DE COMPUTER VISION

En este anexo se muestra una serie de ejemplos que permiten conocer las capacidades de las funcionalidades de Computer Vision, así como mostrar el formato general de los programas con los que se llama a la API para ejecutarlos y los resultados obtenidos en formato de datos JSON.

1 Analizar imagen

A continuación, se va a mostrar el uso de esta funcionalidad con varios ejemplos, para poder observar los resultados obtenidos en función de los parámetros que se seleccionen.

1.1 Contenido para adultos y celebridades

Estas dos funcionalidades, que se van a probar con la Ilustración 50, no guardan ninguna relación con el desarrollo elaborado para la parte práctica del presente trabajo, aunque es evidente su posible utilidad para otras finalidades.

Ilustración 50: Fotografía de Frank Sinatra

Con la demostración de la funcionalidad de "*celebrities*" se da por demostrada también la de "*landmarks*", que se usa exactamente de la misma forma.

En este caso los resultados obtenidos son absolutamente precisos, tanto en la identificación del personaje famoso (Frank Sinatra) como en la ausencia de contenido para adultos en la imagen.

1.1.1 Código del programa

```
import requests                                 # Lo usamos para hacer petición POST
import json                                     # Lo usamos para gestionar la salida JSON

# Credenciales de COMPUTER VISION (AZURE)
AZURE_mi_clave = "X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X"
AZURE_mi_URL = "https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/"

Foto_a_analizar = "foto-famoso.jpg"

URL_descripcion = AZURE_mi_URL + "vision/v3.1/analyze"
cabeceras = {'Ocp-Apim-Subscription-Key': AZURE_mi_clave, 'Content-Type': 'application/octet-stream'}
parametros = {'Visualfeatures': 'Adult', 'details': 'celebrities', 'language': 'es'}
datos_imagen = open(Foto_a_analizar, "rb").read()

respuesta = requests.post(URL_descripcion, headers=cabeceras, params=parametros, data=datos_imagen)

JSON_formateado = json.dumps(respuesta.json(), indent=8)
print(JSON_formateado)
```

1.1.2 Salida JSON

```
{
        "categories": [
                {
                        "name": "gente_",
                        "score": 0.609375,
                        "detail": {
                                "celebrities": [
                                        {
```

```
                                        "name": "Frank Sinatra",
                                        "confidence": 0.999982476234436,
                                        "faceRectangle": {
                                                "left": 297,
                                                "top": 76,
                                                "width": 111,
                                                "height": 111
                                        }
                                }
                        ]
                }
        }
    ],
    "adult": {
            "isAdultContent": false,
            "isRacyContent": false,
            "isGoryContent": false,
            "adultScore": 0.016210006549954414,
            "racyScore": 0.01834241859614849,
            "goreScore": 0.004559162072837353
    },
    "requestId": "fa66cd03-5018-44c8-a9af-fa8ddb6e6357",
    "metadata": {
            "height": 405,
            "width": 720,
            "format": "Jpeg"
    }
}
```


1.2 Categorías, etiquetas y color

Con el análisis de la Ilustración 51, además de ver los valores devueltos por las tres funcionalidades mencionadas en el título se va a mostrar la forma de pasar simultáneamente varios parámetros de "*visualfeatures*", así como el JSON resultante de analizar varias características en una misma llamada a la API.

Ilustración 51: Fotografía de un partido de polo

El resultado del análisis se considera aceptable, aunque llama la atención que haya una etiqueta "caballo" (se sobreentiende que no se repiten etiquetas para objetos similares), otra "animal" y otra "cabestro", pues probablemente con las tres se hace referencia al mismo tipo de "objeto" (cada uno de los cinco caballos).

1.2.1 Código del programa

```
import requests                                 # Lo usamos para hacer petición POST
import json                                     # Lo usamos para gestionar la salida JSON

# Credenciales de COMPUTER VISION (AZURE)
AZURE_mi_clave = "X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X"
AZURE_mi_URL = "https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/"

Foto_a_analizar = "partido-polo.jpg"

URL_descripcion = AZURE_mi_URL + "vision/v3.1/analyze"
cabeceras = {'Ocp-Apim-Subscription-Key': AZURE_mi_clave, 'Content-Type': 'application/octet-stream'}
parametros = {'visualfeatures': 'Categories,Tags,Color', 'language': 'es'}
datos_imagen = open(Foto_a_analizar, "rb").read()

respuesta = requests.post(URL_descripcion, headers=cabeceras, params=parametros, data=datos_imagen)

JSON_formateado = json.dumps(respuesta.json(), indent=8)
print(JSON_formateado)
```

1.2.2 Salida JSON


```
{
	"categories": [
		{
			"name": "gente_muchos",
			"score": 0.83984375
		}
	],
	"color": {
		"dominantColorForeground": "Brown",
		"dominantColorBackground": "Black",
		"dominantColors": [
			"Black"
		],
		"accentColor": "AF1F1C",
		"isBwImg": false,
		"isBWImg": false
	},
	"tags": [
		{
			"name": "pasto",
			"confidence": 0.9978961944580078
		},
		{
			"name": "\u00e1rbol",
			"confidence": 0.9978867769241333
		},
		{
			"name": "caballo",
			"confidence": 0.9907234907150269
		},
		{
			"name": "exterior",
			"confidence": 0.9659996032714844
		},
		{
			"name": "animal",
			"confidence": 0.8820403814315796
		},
		{
			"name": "carreras de caballos",
			"confidence": 0.7901645898881897
		},
		{
			"name": "ecuestre",
			"confidence": 0.6839743852615356
		},
		{
			"name": "rienda",
			"confidence": 0.6255576610565186
		},
		{
			"name": "jinete",
			"confidence": 0.6153470277786255
		},
		{
			"name": "brida",
			"confidence": 0.5975544452667236
		},
		{
			"name": "cabestro",
			"confidence": 0.5838940143585205
		},
		{
			"name": "grupo",
			"confidence": 0.5764282941818237
		},
		{
			"name": "silla de montar",
			"confidence": 0.5446920394897461
		},
		{
			"name": "pista de caballos",
			"confidence": 0.5300869345664978
		}
	],
	"requestId": "08400b4f-3004-4093-ad64-280a9b53dcd1",
	"metadata": {
		"height": 467,
		"width": 700,
		"format": "Jpeg"
	}
```

```
}
```

1.3 Descripción

Para mostrar el funcionamiento de esta funcionalidad vamos a usar la misma imagen del Coliseo de Roma (Ilustración 52) que se usó en el punto 5.3.12, donde se obtuvieron las descripciones usando la opción "describir una imagen" (punto 5.3.2), para comprobar si las descripciones obtenidas con las dos alternativas son las mismas, como se piensa que, a priori, debería ocurrir.

Ilustración 52: Fotografía del Coliseo (Roma)

La conclusión, como puede comprobarse comparando las dos salidas JSON, es que se obtiene el mismo resultado, si bien a través de la funcionalidad de "analizar" no es posible conseguir nada más que una descripción, mientras que con este se puede pasar un parámetro para indicar el número máximo de descripciones que se desea obtener. Por otra parte, en relación con la descripción conseguida es llamativo que a la vez que se identifica perfectamente el tipo de edificio (un coliseo), se haga una descripción tan extraña como "coliseo de un edificio", como si un coliseo pudiera ser una parte de un edificio y no un edificio en sí mismo.

1.3.1 Código del programa

```
import requests                                 # Lo usamos para hacer petición POST
import json                                     # Lo usamos para gestionar la salida JSON

# Credenciales de COMPUTER VISION (AZURE)
AZURE_mi_clave = "X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X"
AZURE_mi_URL = "https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/"

Foto_a_analizar = "coliseo-romano.jpg"

URL_descripcion = AZURE_mi_URL + "vision/v3.1/analyze"
cabeceras = {'Ocp-Apim-Subscription-Key': AZURE_mi_clave, 'Content-Type': 'application/octet-stream'}
parametros = {'visualfeatures': 'Description', 'language': 'es'}
datos_imagen = open(Foto_a_analizar, "rb").read()

respuesta = requests.post(URL_descripcion, headers=cabeceras, params=parametros, data=datos_imagen)

JSON_formateado = json.dumps(respuesta.json(), indent=8)
print(JSON_formateado)
```

1.3.2 Salida JSON

```
{
        "description": {
                "tags": [
                        "pasto",
                        "edificio",
                        "exterior",
                        "frente",
                        "grande",
                        "viejo",
                        "tabla",
                        "pastel",
                        "piedra",
                        "flor",
                        "iglesia",
                        "alto",
                        "tren",
                        "montar a caballo",
                        "mujer",
```

```
                        "parado",
                        "reloj",
                        "campo",
                        "hombre",
                        "caballo",
                        "caminando",
                        "gente",
                        "lugar",
                        "boda",
                        "pantalla",
                        "manejar",
                        "torre",
                        "calle",
                        "ciudad",
                        "estacionado",
                        "edificio de gobierno",
                        "cami\u00f3n",
                        "paraguas"
                ],
                "captions": [
                        {
                                "text": "Coliseo de un edificio",
                                "confidence": 0.6651820709352974
                        }
                ]
        },
        "requestId": "3bc4a224-819e-43a3-8db2-a0c381718ad0",
        "metadata": {
                "height": 720,
                "width": 1280,
                "format": "Jpeg"
        }
}
```


1.4 Caras y tipo de imagen

Con la Ilustración 53 vamos a comprobar el resultado del análisis de una imagen para detectar las caras que hay en ella y el tipo de imagen.

Con respecto al resultado del análisis, podemos comprobar que detecta cuatro caras, aunque hay cinco, y que a una de ellas le asigna una edad de 41 años, aunque todas las caras parecen ser de personas más jóvenes de esa edad. La cara que no detecta es la de la chica que está más al fondo.

Ilustración 53: Fotografía de varias personas

1.4.1 Código del programa

```
import requests                                 # Lo usamos para hacer petición POST
import json                                     # Lo usamos para gestionar la salida JSON

# Credenciales de COMPUTER VISION (AZURE)
AZURE_mi_clave = "X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X"
AZURE_mi_URL = "https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/"

Foto_a_analizar = "caras.jpg"

URL_descripcion = AZURE_mi_URL + "vision/v3.1/analyze"
cabeceras = {'Ocp-Apim-Subscription-Key': AZURE_mi_clave, 'Content-Type': 'application/octet-stream'}
parametros = {'visualfeatures': 'Faces,ImageType', 'language': 'es'}
datos_imagen = open(Foto_a_analizar, "rb").read()

respuesta = requests.post(URL_descripcion, headers=cabeceras, params=parametros, data=datos_imagen)

JSON_formateado = json.dumps(respuesta.json(), indent=8)
print(JSON_formateado)
```

1.4.2 Salida JSON


```
{
        "imageType": {
```

```
            "clipArtType": 0,
            "lineDrawingType": 0
        },
        "faces": [
            {
                "age": 32,
                "gender": "Male",
                "faceRectangle": {
                    "left": 66,
                    "top": 134,
                    "width": 113,
                    "height": 113
                }
            },
            {
                "age": 29,
                "gender": "Male",
                "faceRectangle": {
                    "left": 325,
                    "top": 127,
                    "width": 111,
                    "height": 111
                }
            },
            {
                "age": 41,
                "gender": "Male",
                "faceRectangle": {
                    "left": 216,
                    "top": 140,
                    "width": 90,
                    "height": 90
                }
            },
            {
                "age": 24,
                "gender": "Female",
                "faceRectangle": {
                    "left": 462,
                    "top": 88,
                    "width": 78,
                    "height": 78
                }
            }
        ],
        "requestId": "549189a6-5a35-4990-83ab-a832e4442ffe",
        "metadata": {
            "height": 417,
            "width": 626,
            "format": "Jpeg"
        }
}
```


1.5 Objetos

Se ha elegido la imagen de la Ilustración 54 por la variedad de objetos que incorpora, si bien, como puede verse, el resultado del análisis ha resultado bastante sorprendente ya que solamente ha detectado tres de ellos (dos plantas y un televisor).

Ilustración 54: Fotografía de un dormitorio

En la salida JSON es reseñable la estructura de los "*parent*" de cada objeto, lo que genera una "estructura jerárquica" de esos objetos (no confundir con las categorías de la Ilustración 12, con las que no guardan relación).

1.5.1 Código del programa

```
import requests                                 # Lo usamos para hacer petición POST
import json                                     # Lo usamos para gestionar la salida JSON

# Credenciales de COMPUTER VISION (AZURE)
AZURE_mi_clave = "X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X"
AZURE_mi_URL = "https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/"

Foto_a_analizar = "habitacion.jpg"

URL_descripcion = AZURE_mi_URL + "vision/v3.1/analyze"
cabeceras = {'Ocp-Apim-Subscription-Key': AZURE_mi_clave, 'Content-Type': 'application/octet-stream'}
parametros = {'visualfeatures': 'Objects', 'language': 'es'}
datos_imagen = open(Foto_a_analizar, "rb").read()

respuesta = requests.post(URL_descripcion, headers=cabeceras, params=parametros, data=datos_imagen)

JSON_formateado = json.dumps(respuesta.json(), indent=8)
print(JSON_formateado)
```

1.5.2 Salida JSON


```
{
        "objects": [
                {
                        "rectangle": {
                                "x": 348,
                                "y": 37,
                                "w": 229,
                                "h": 392
                        },
                        "object": "potted plant",
                        "confidence": 0.554,
                        "parent": {
                                "object": "plant",
                                "confidence": 0.697
                        }
                },
                {
                        "rectangle": {
                                "x": 0,
                                "y": 249,
                                "w": 112,
                                "h": 209
                        },
                        "object": "potted plant",
                        "confidence": 0.507,
                        "parent": {
                                "object": "plant",
                                "confidence": 0.598
                        }
                },
                {
                        "rectangle": {
                                "x": 1076,
                                "y": 289,
                                "w": 202,
                                "h": 222
                        },
                        "object": "television",
                        "confidence": 0.533,
                        "parent": {
                                "object": "display",
                                "confidence": 0.554
                        }
                }
        ],
        "requestId": "e38f34a8-d738-4b62-9151-8ba962f2ba8d",
        "metadata": {
                "height": 638,
                "width": 1280,
                "format": "Jpeg"
        }
}
```

2 Describir imagen

Esta funcionalidad es la que ya se empleó en el punto 5.3.12.

En este caso optamos por una imagen de animales al aire libre (Ilustración 55) y pedimos que se nos devuelva un máximo de dos descripciones.

Como puede verse, las descripciones son acertadas, aunque solo se refieren a una de las dos aves.

Ilustración 55: Foto de aves en la playa

2.1 Código del programa

```
import requests                                 # Lo usamos para hacer petición POST
import json                                     # Lo usamos para gestionar la salida JSON

# Credenciales de COMPUTER VISION (AZURE)
AZURE_mi_clave = "X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X"
AZURE_mi_URL = "https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/"

Foto_a_analizar = "aves.jpg"

URL_descripcion = AZURE_mi_URL + "vision/v3.1/describe"
cabeceras = {'Ocp-Apim-Subscription-Key': AZURE_mi_clave, 'Content-Type': 'application/octet-stream'}
parametros = {'maxCandidates': '2', 'language': 'es'}
datos_imagen = open(Foto_a_analizar, "rb").read()

respuesta = requests.post(URL_descripcion, headers=cabeceras, params=parametros, data=datos_imagen)

JSON_formateado = json.dumps(respuesta.json(), indent=8)
print(JSON_formateado)
```

2.2 Salida JSON


```
{
        "description": {
                "tags": [
                        "exterior",
                        "agua",
                        "p\u00e1jaro",
                        "animal",
                        "playa",
                        "ave",
                        "parado",
                        "vuelo",
                        "reba\u00f1o",
                        "costa",
                        "cuerpo",
                        "oc\u00e9ano",
                        "comida",
                        "arena",
                        "grupo",
                        "arenoso",
                        "lago",
                        "nadar",
                        "aterrizar"
                ],
                "captions": [
                        {
                                "text": "ave a la orilla de la playa",
                                "confidence": 0.74699027806718
                        },
                        {
                                "text": "ave parada en la arena de la playa",
```

```
                        "confidence": 0.74599027806718
                }
        ]
    },
    "requestId": "aed028e8-5786-43c2-beb5-261057a3bf7a",
    "metadata": {
        "height": 639,
        "width": 960,
        "format": "Jpeg"
    }
}
```


3 Detectar objetos

Este programa se ha ejecutado usando la misma imagen que se analizó en el punto 1.5 y, como puede comprobarse, con ambas opciones se han conseguido exactamente los mismos resultados.

3.1 Código del programa

```
import requests                                 # Lo usamos para hacer petición POST
import json                                     # Lo usamos para gestionar la salida JSON

# Credenciales de COMPUTER VISION (AZURE)
AZURE_mi_clave = "X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X"
AZURE_mi_URL = "https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/"

Foto_a_analizar = "habitacion.jpg"

URL_descripcion = AZURE_mi_URL + "vision/v3.1/detect"
cabeceras = {'Ocp-Apim-Subscription-Key': AZURE_mi_clave, 'Content-Type': 'application/octet-stream'}
datos_imagen = open(Foto_a_analizar, "rb").read()

respuesta = requests.post(URL_descripcion, headers=cabeceras, data=datos_imagen)

JSON_formateado = json.dumps(respuesta.json(), indent=8)
print(JSON_formateado)
```

3.2 Salida JSON


```
{
    "objects": [
        {
            "rectangle": {
                "x": 348,
                "y": 37,
                "w": 229,
                "h": 392
            },
            "object": "potted plant",
            "confidence": 0.554,
            "parent": {
                "object": "plant",
                "confidence": 0.697
            }
        },
        {
            "rectangle": {
                "x": 0,
                "y": 249,
                "w": 112,
                "h": 209
            },
            "object": "potted plant",
            "confidence": 0.507,
            "parent": {
                "object": "plant",
                "confidence": 0.598
            }
        },
        {
            "rectangle": {
                "x": 1076,
                "y": 289,
                "w": 202,
                "h": 222
            },
            "object": "television",
            "confidence": 0.533,
            "parent": {
```

```
                        "object": "display",
                        "confidence": 0.554
                }
        }
    ],
    "requestId": "e7205073-a70b-4df0-b14a-5232ce877511",
    "metadata": {
        "height": 638,
        "width": 1280,
        "format": "Jpeg"
    }
}
```


4 Detectar área de interés

Para esta funcionalidad se ha empleado la Ilustración 56 porque resulta evidente cual es la parte de ella que incluye el "área de interés".

El "*área de interés*" se identifica por sus coordenadas, pero para mayor claridad se va a usar esta misma imagen en la funcionalidad de generación de miniaturas, por la similitud con aquella, que identifica lo que denomina "*región de interés*".

Ilustración 56: Fotografía de un elefante en la sabana

4.1 Código del programa

```
import requests                                 # Lo usamos para hacer petición POST
import json                                     # Lo usamos para gestionar la salida JSON

# Credenciales de COMPUTER VISION (AZURE)
AZURE_mi_clave = "X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X"
AZURE_mi_URL = "https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/"

Foto_a_analizar = "elefante.jpg"

URL_descripcion = AZURE_mi_URL + "vision/v3.1/areaOfInterest"
cabeceras = {'Ocp-Apim-Subscription-Key': AZURE_mi_clave, 'Content-Type': 'application/octet-stream'}
datos_imagen = open(Foto_a_analizar, "rb").read()

respuesta = requests.post(URL_descripcion, headers=cabeceras, data=datos_imagen)

JSON_formateado = json.dumps(respuesta.json(), indent=8)
print(JSON_formateado)
```

4.2 Salida JSON


```
{
        "areaOfInterest": {
                "x": 253,
                "y": 0,
                "w": 483,
                "h": 485
        },
        "requestId": "290ab375-3eeb-45cc-a9f4-fecaf39ebbe5",
        "metadata": {
                "height": 485,
                "width": 863,
                "format": "Jpeg"
        }
}
```


5 Generar miniatura

Como se comentaba en el punto anterior, para esta funcionalidad también se va a usar la Ilustración 56 empleando la opción "*smartcropping*", que identifica la "región de interés"

(diferente del área de interés del punto anterior) de la imagen y crea la miniatura de esa región.

5.1 Código del programa

```
import requests
from PIL import Image
from io import BytesIO

# Credenciales de COMPUTER VISION (AZURE)
AZURE_mi_clave = "X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X"
AZURE_mi_URL = "https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/"

Foto_a_analizar = "elefante.jpg"

URL_descripcion = AZURE_mi_URL + "vision/v3.1/generateThumbnail"
cabeceras = {'Ocp-Apim-Subscription-Key': AZURE_mi_clave, 'Content-Type': 'application/octet-stream'}
parametros = {'width': '200', 'height': '200', 'smartCropping': 'true'}
datos_imagen = open(Foto_a_analizar, "rb").read()

respuesta = requests.post(URL_descripcion, headers=cabeceras, params=parametros, data=datos_imagen)

fichero_thumbnail = Image.open(BytesIO(respuesta.content))

fichero_thumbnail.save('thumbnail-elefante.png')
```

5.2 Miniatura generada

6 OCR

Esta funcionalidad es especialmente relevante por el interés que tiene el reconocer texto dentro de la imagen para luego poder tratarlo de la forma que proceda.

Para esta funcionalidad se vuelve a usar una fotografía con coches, pues se quiere comprobar la posibilidad de detectar matrículas gracias a *Computer Vision*.

En cuanto al código para explotar esta funcionalidad, tiene una particularidad con respecto a las otras, ya que es necesario usar tres bucles anidados para conseguir el texto y la explicación se ve claramente en el formato de la salida JSON, ya que la salida se divide en regiones (“*regions*”) que a su vez se dividen en líneas (“*lines*”) y éstas en palabras (“*words*”), cada una de las cuales consta de un texto (“*text*”) y sus correspondientes coordenadas.

Por otra parte, se ha incluido un desarrollo muy simple que permite detectar si la línea de texto sigue el patrón de las matrículas españolas actuales (cuatro letras, un espacio y tres letras) para poder identificar ese texto como una posible matrícula, ya que, en la actualidad Computer Vision no tiene capacidad de reconocer directamente el texto de las matrículas (aunque ha podido comprobarse que sí reconoce y etiqueta algunos artículos como “*Vehicle registration plate*”).

Como puede verse en los resultados obtenido, la fiabilidad de la detección de matrículas y de texto en general es bastante baja, y, además, solo sería aplicable al patrón concreto de matriculas españolas actuales “estándar”.

6.1 Código del programa

```
import requests                          # Lo usamos para hacer petición POST
```

```
# Credenciales de COMPUTER VISION (AZURE)
AZURE_mi_clave = "x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x"
AZURE_mi_URL = "https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/"

Foto_a_analizar = "Coches-reco-OCR.jpg"

URL_OCR = AZURE_mi_URL + "vision/v3.1/ocr"
cabeceras = {'Ocp-Apim-Subscription-Key': AZURE_mi_clave, 'Content-Type': 'application/octet-stream'}
parametros = {'language': 'es'}
datos_imagen = open(Foto_a_analizar, "rb").read()

respuesta = requests.post(URL_OCR, headers=cabeceras, params=parametros, data=datos_imagen)

for zona_de_texto in respuesta.json()["regions"]:

    for linea_de_texto in zona_de_texto["lines"]:
        Texto = ""

        for palabra_detectada in linea_de_texto["words"]:
            Texto += palabra_detectada["text"] + " "

        # Se busca texto con el patrón de las matrículas españolas actuales
        Texto = Texto.strip()
        if len(Texto) == 8 and (Texto[0:4]).isnumeric() and (Texto[5:8]).isalpha():
            Texto = "'" + Texto + "'" + " (probablemente corresponde a una matrícula española.)"
        elif len(Texto) == 9 and (Texto[1:5]).isnumeric() and (Texto[6:9]).isalpha():
            Texto = "'" + Texto + "'" + " (probablemente corresponde a una matrícula española " \
                                        "a la que se ha añadido un carácter inicial.)"
        elif len(Texto) == 9 and (Texto[0:4]).isnumeric() and (Texto[5:8]).isalpha():
            Texto = "'" + Texto + "'" + " (probablemente corresponde a una matrícula española " \
                                        "a la que se ha añadido un carácter final.)"
        else:
            Texto = "'" + Texto + "'"

        print("    - TEXTO: " + Texto)
        print("         * Ubicación: {}".format(linea_de_texto["boundingBox"]))
print()
```

6.2 Ejemplo 1

Con la Ilustración 57 se trata de valorar la precisión de Computer Vision para identificar texto en varios soportes, incluyendo una matrícula de coche.

Ilustración 57: Imagen con texto y coches

Como puede verse, los resultados son claramente mejorables, incluyendo la lectura de la matrícula, en la que interpreta la letra “N” como si fuera la combinación “IV”.

6.2.1 Salida JSON


```
{
        "language": "es",
        "textAngle": 0.1466076571675243,
        "orientation": "Up",
        "regions": [
                {
                        "boundingBox": "2382,321,350,417",
                        "lines": [
                                {
                                        "boundingBox": "2382,321,350,82",
                                        "words": [
                                                {
                                                        "boundingBox": "2382,321,177,57",
                                                        "text": "CEPSA"
                                                },
                                                {
                                                        "boundingBox": "2587,338,145,65",
                                                        "text": "///////"
                                                }
                                        ]
                                },
                                {
                                        "boundingBox": "2437,644,77,39",
                                        "words": [
                                                {
                                                        "boundingBox": "2437,644,77,39",
```

```
                                        "text": "1.369"
                                }
                        ]
                },
                {
                        "boundingBox": "2429,701,76,37",
                        "words": [
                                {
                                        "boundingBox": "2429,701,76,37",
                                        "text": "1.5?9"
                                }
                        ]
                }
        ]
},
{
        "boundingBox": "3258,387,208,365",
        "lines": [
                {
                        "boundingBox": "3381,387,85,30",
                        "words": [
                                {
                                        "boundingBox": "3381,387,85,30",
                                        "text": "park"
                                }
                        ]
                },
                {
                        "boundingBox": "3314,410,152,44",
                        "words": [
                                {
                                        "boundingBox": "3314,410,75,35",
                                        "text": "San"
                                },
                                {
                                        "boundingBox": "3408,419,58,35",
                                        "text": "Luc"
                                }
                        ]
                },
                {
                        "boundingBox": "3358,531,108,33",
                        "words": [
                                {
                                        "boundingBox": "3358,531,108,33",
                                        "text": "parkit"
                                }
                        ]
                },
                {
                        "boundingBox": "3273,555,193,50",
                        "words": [
                                {
                                        "boundingBox": "3273,555,36,32",
                                        "text": "El"
                                },
                                {
                                        "boundingBox": "3330,561,136,44",
                                        "text": "Mercac"
                                }
                        ]
                },
                {
                        "boundingBox": "3335,678,131,36",
                        "words": [
                                {
                                        "boundingBox": "3335,678,131,36",
                                        "text": "parkinc"
                                }
                        ]
                },
                {
                        "boundingBox": "3258,703,208,49",
                        "words": [
                                {
                                        "boundingBox": "3258,703,36,27",
                                        "text": "Lo"
                                },
                                {
                                        "boundingBox": "3304,711,162,41",
                                        "text": "'Juzgado;"
                                }
                        ]
                }
```

```
                ]
            },
            {
                "boundingBox": "732,1373,626,278",
                "lines": [
                    {
                        "boundingBox": "732,1373,77,51",
                        "words": [
                            {
                                "boundingBox": "732,1373,77,51",
                                "text": "HIJI"
                            }
                        ]
                    },
                    {
                        "boundingBox": "969,1525,389,126",
                        "words": [
                            {
                                "boundingBox": "969,1525,202,94",
                                "text": "2587"
                            },
                            {
                                "boundingBox": "1206,1555,152,96",
                                "text": "FIVL"
                            }
                        ]
                    }
                ]
            }
        ]
}
```


6.2.2 Salida del programa

```
    - TEXTO: 'CEPSA ///////'
        * Ubicación: 2382,321,350,82
    - TEXTO: '1.369'
        * Ubicación: 2437,644,77,39
    - TEXTO: '1.5?9'
        * Ubicación: 2429,701,76,37
    - TEXTO: 'park'
        * Ubicación: 3381,387,85,30
    - TEXTO: 'San Luc'
        * Ubicación: 3314,410,152,44
    - TEXTO: 'parkit'
        * Ubicación: 3358,531,108,33
    - TEXTO: 'El Mercac'
        * Ubicación: 3273,555,193,50
    - TEXTO: 'parkinc'
        * Ubicación: 3335,678,131,36
    - TEXTO: 'Lo 'Juzgado;'
        * Ubicación: 3258,703,208,49
    - TEXTO: 'HIJI'
        * Ubicación: 732,1373,77,51
    - TEXTO: '2587 FIVL' (probablemente corresponde a una matrícula española a la que se ha añadido un carácter final.)
        * Ubicación: 969,1525,389,126
```

6.3 Ejemplo 2

Con el segundo ejemplo del empleo de la funcionalidad del OCR, en la Ilustración 58, no se muestra la salida JSON porque ya se ha visto su estructura en el punto anterior.

En la salida puede verse que, a pesar de haber dos matrículas claramente visibles, solo identifica una de ellas, porque de la otra no identifica dos caracteres numéricos que son claramente “44” (probablemente por la mancha que hay sobre ellos).

Ilustración 58: Tres vehículos con matrículas visibles

6.3.1 Salida del programa

```
    - TEXTO: '382 KHC'
        * Ubicación: 0,1236,158,35
```

```
    - TEXTO: '83 GKB'
        * Ubicación: 1359,1384,155,40
    - TEXTO: '1495 JVM' (probablemente corresponde a una matrícula española.)
        * Ubicación: 2518,1358,250,54
```

7 Detectar contenido específico

Para comprobar esta funcionalidad usamos la misma imagen (Ilustración 50) que en el punto 1.1 y podemos confirmar que se consigue los mismos resultados, por lo que resulta indiferente elegir una u otra cuando se necesite este tipo de análisis.

7.1 Código del programa

```
import requests                                 # Lo usamos para hacer petición POST
import json                                     # Lo usamos para gestionar la salida JSON

# Credenciales de COMPUTER VISION (AZURE)
AZURE_mi_clave = "X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X"
AZURE_mi_URL = "https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/"

Foto_a_analizar = "foto-famoso.jpg"

URL_descripcion = AZURE_mi_URL + "vision/v3.1/models/celebrities/analyze"
cabeceras = {'Ocp-Apim-Subscription-Key': AZURE_mi_clave, 'Content-Type': 'application/octet-stream'}
parametros = {'language': 'es'}
datos_imagen = open(Foto_a_analizar, "rb").read()

respuesta = requests.post(URL_descripcion, headers=cabeceras, params=parametros, data=datos_imagen)

JSON_formateado = json.dumps(respuesta.json(), indent=8)
print(JSON_formateado)
```

7.2 Salida JSON

```
{
        "result": {
                "celebrities": [
                        {
                                "name": "Frank Sinatra",
                                "confidence": 0.999982476234436,
                                "faceRectangle": {
                                        "left": 297,
                                        "top": 76,
                                        "width": 111,
                                        "height": 111
                                }
                        }
                ]
        },
        "requestId": "d38436cd-ab19-4601-aef7-190a02aed030",
        "metadata": {
                "height": 405,
                "width": 720,
                "format": "Jpeg"
        }
}
```

8 Detección de etiquetas

Para comprobar esta funcionalidad se va a usar la misma imagen que la usada para conseguir las etiquetas mediante el análisis de una imagen (punto 1.2), para comparar los resultados de ambas.

Estudiando los resultados se confirma que las etiquetas obtenidas por los dos procedimientos son exactamente iguales, tanto el texto como el porcentaje de certeza.

8.1 Código del programa

```
import requests                                 # Lo usamos para hacer petición POST

# Credenciales de COMPUTER VISION (AZURE)
AZURE_mi_clave = "X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X"
AZURE_mi_URL = "https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/"

Foto_a_analizar = "Partido-polo.jpg"
```

```
URL_descripcion = AZURE_mi_URL + "vision/v3.1/tag"
cabeceras = {'Ocp-Apim-Subscription-Key': AZURE_mi_clave, 'Content-Type': 'application/octet-stream'}
parametros = {'language': 'es'}
datos_imagen = open(Foto_a_analizar, "rb").read()

respuesta = requests.post(URL_descripcion, headers=cabeceras, params=parametros, data=datos_imagen)

JSON_formateado = json.dumps(respuesta.json(), indent=8)
print(JSON_formateado)
```

8.2 Salida JSON


```
{
        "tags": [
                {
                        "name": "pasto",
                        "confidence": 0.9978961944580078
                },
                {
                        "name": "\u00e1rbol",
                        "confidence": 0.9978867769241333
                },
                {
                        "name": "caballo",
                        "confidence": 0.9907234907150269
                },
                {
                        "name": "exterior",
                        "confidence": 0.9659996032714844
                },
                {
                        "name": "animal",
                        "confidence": 0.8820403814315796
                },
                {
                        "name": "carreras de caballos",
                        "confidence": 0.790164589881897
                },
                {
                        "name": "ecuestre",
                        "confidence": 0.6839743852615356
                },
                {
                        "name": "rienda",
                        "confidence": 0.6255576610565186
                },
                {
                        "name": "jinete",
                        "confidence": 0.6153470277786255
                },
                {
                        "name": "brida",
                        "confidence": 0.5975544452667236
                },
                {
                        "name": "cabestro",
                        "confidence": 0.5838940143585205
                },
                {
                        "name": "grupo",
                        "confidence": 0.5764282941818237
                },
                {
                        "name": "silla de montar",
                        "confidence": 0.5446920394897461
                },
                {
                        "name": "pista de caballos",
                        "confidence": 0.5300869345664978
                }
        ],
        "requestId": "9163fff6-0294-480c-bc06-3c7e84e3803b",
        "metadata": {
                "height": 467,
                "width": 700,
                "format": "Jpeg"
        }
}
```

9 Read / Get read

Estas dos funcionalidades están ligadas, debido a que la petición "*read*" no devuelve instantáneamente su resultados, por estar diseñada para tratamiento de ficheros con un texto significativamente grande, de modo que inicialmente debe enviarse la petición read (POST) y con posterioridad la petición de devolución de los resultados (GET). No obstante, en nuestro programa de ejemplo se ha incluido la funcionalidad GET a continuación de la "*read*", dado que el análisis del texto de la imagen que se usa como ejemplo lleva poco tiempo.

A diferencia de la escasa precisión que se vio en el análisis con la funcionalidad OCR, la que se ve en este punto parece ser mucho más alta (aunque más lenta), a pesar de que el tamaño del texto de la Ilustración 59 es mucho más pequeño.

Debido al gran tamaño de la salida JSON, en este caso se ha optado por mostrar la salida formateada. No obstante, el código del programa incluye la salida JSON por consola (se han marcado como comentario las dos líneas de código necesarias para generar esa salida, por si se considerase relevante ejecutar el programa obteniendo la salida JSON).

Ilustración 59: Anuncio en la ventana de un banco

9.1 Código de programa

```
import requests                                 # Lo usamos para hacer petición POST
import json                                     # Lo usamos para gestionar la salida JSON
import time

# Credenciales de COMPUTER VISION (AZURE)
AZURE_mi_clave = "X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X"
AZURE_mi_URL = "https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/"

Foto_a_analizar = "Escaparate_banco.jpg"

URL_descripcion = AZURE_mi_URL + "vision/v3.1/read/analyze"
cabeceras = {'Ocp-Apim-Subscription-Key': AZURE_mi_clave, 'Content-Type': 'application/octet-stream'}
parametros = {'language': 'es'}
datos_imagen = open(Foto_a_analizar, "rb").read()
```

```
respuesta = requests.post(URL_descripcion, headers=cabeceras, params=parametros, data=datos_imagen)

URL_resultado = respuesta.headers["Operation-Location"]


# EL TEXTO RECONOCIDO NO ESTÁ DISPONIBLE AUTOMÁTICAMENTE, HAY QUE ESPERAR, PARA LO QUE SE CREA UN BUCLE DE ESPERA
JSON_salida = {}
esperar_resultado = True

while esperar_resultado:
    respuesta_final = requests.get(respuesta.headers["Operation-Location"], headers=cabeceras)
    JSON_salida = respuesta_final.json()

    time.sleep(1)
    if "analyzeResult" in JSON_salida:
        esperar_resultado = False
    if "status" in JSON_salida and JSON_salida['status'] == 'failed':
        esperar_resultado = False

# JSON_formateado = json.dumps(JSON_salida, indent=8)
# print(JSON_formateado)

for linea_de_texto in JSON_salida["analyzeResult"]["readResults"]:
    print(linea_de_texto)
    Texto = ""

    for palabra_detectada in linea_de_texto["lines"]:
        print(palabra_detectada["text"])
```

9.2 Salida

```
coonectar
Disponibilidad. Queremos estar cerca.
En esta oficina y por todos los medios.
Porque nos tienes a mano con CONECTA-MI GESTOR.
En tu banca a distancia, Intercambiando Información,
documentos y gestionando tus productos.
Este nuestro modelo. Esta es tu banca cooperativa.
Si lo que buscas son respuestas ...
Q
#larespuestacooperativa
in
RESPONDEMOS CON SERVICIO Y SOLUCIONES, DANDO LO MEJOR DE NOSCTROS. DESDE ESTA OFICINA. COMO SIEMPRE. COMO EL PRIMER DÍA
TENEMOS UN COMPROMISO CON NUESTRA GENTE ESTAR A TU LADO. AQUÍ Y POR TODOS LOS MEDIOS.
DESDE NUESTRA WEB, NUESTRA BANCA ELECTRÓNICA O NUESTRA APP MÓVIL
PARA TENERNOS SIEMPRE A MANO, ESCANEA ESTE CÓDIGO QR.
```

10 Listar modelos de dominio específicos

Esta funcionalidad es de utilidad exclusiva de los desarrolladores, ya que permite conocer los tipos de modelos específicos que reconoce Computer Vision y que pueden ser empleados como parámetros en la correspondientes llamadas a la API. En nuestro caso, los tipos encontrados son los dos citados reiteradas veces a lo largo del trabajo ("*celebrities*" y "*landmarks*").

10.1 Código del programa

```
import requests                                 # Lo usamos para hacer petición GET
import json                                     # Lo usamos para gestionar la salida JSON

# Credenciales de COMPUTER VISION (AZURE)
AZURE_mi_clave = "X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X"
AZURE_mi_URL = "https://computer-vision-alvaro-ftg.cognitiveservices.azure.com/"

URL_descripcion = AZURE_mi_URL + "vision/v3.1/models"
cabeceras = {'Ocp-Apim-Subscription-Key': AZURE_mi_clave, 'Content-Type': 'application/octet-stream'}

respuesta = requests.get(URL_descripcion, headers=cabeceras)

JSON_formateado = json.dumps(respuesta.json(), indent=8)
print(JSON_formateado)
```

10.2 Salida JSON

```
{
        "models": [
                {
                        "name": "celebrities",
                        "categories": [
```

```
                        "people_",
                        "\u4eba_",
                        "pessoas_",
                        "gente_"
                ]
        },
        {
                "name": "landmarks",
                "categories": [
                        "outdoor_",
                        "\u6237\u5916_",
                        "\u5c4b\u5916_",
                        "aoarlivre_",
                        "alairelibre_",
                        "building_",
                        "\u5efa\u7b51_",
                        "\u5efa\u7269_",
                        "edif\u00edcio_"
                ]
        }
    ]
}
```

Bibliografía

[1] Computerworld, «Microsoft factura más en 2020 que AWS y Google Cloud juntos,» 23 Febrero 2021. [En línea]. Available: https://www.computerworld.es/negocio/microsoft-azure-factura-mas-en-2020-que-aws-y-google-cloud-juntos. [Último acceso: 24 Febrero 2021].

[2] Association for the Advancement of Artificial Intelligence (AAAI) , «A Brief History of AI,» 2021. [En línea]. Available: https://aitopics.org/misc/brief-history. [Último acceso: 25 Febrero 2021].

[3] SAS, «Historia de la inteligencia artificial,» [En línea]. Available: https://www.sas.com/es_es/insights/analytics/what-is-artificial-intelligence.html. [Último acceso: 27 Enero 2021].

[4] [R]evolución artificial. Blog con soluciones de Visión Artificial y nuevas tecnologías, «Historia de la visión artificial,» 29 Enero 2020. [En línea]. Available: https://blog.infaimon.com/historia-vision-artificial/. [Último acceso: 5 Febrero 2021].

[5] eloobook, «Captura fotográfica,» 29 Febrero 2020. [En línea]. Available: https://eloobook.com/tips-de-lectura-y-creatividad/captura-fotografica-todo-sobre-los-tipos-de-camaras-fotograficas-y-la-imagen-digital/. [Último acceso: 2 Febrero 2021].

[6] COGNEX, «Qué es la visión artificial,» 2021. [En línea]. Available: https://www.cognex.com/es-es/what-is/machine-vision/what-is-machine-vision. [Último acceso: Febrero 2021].

[7] Solución ingenieril, «Conceptos básicos de Visión Artificial,» 2021. [En línea]. Available: http://solucioningenieril.com/vision_artificial/conceptos_basicos_de_vision_artificial. [Último acceso: 5 Febrero 2021].

[8] Digital guide IONOS, «La nube pública: más recursos para todos,» [En línea]. Available: https://www.ionos.es/digitalguide/servidores/know-how/public-cloud/. [Último acceso: Febrero 2021].

[9] Red hat, «Find a Red hat business partner,» [En línea]. Available: https://redhat.secure.force.com/finder/FindPartner.

[10] CITRIX, «¿Qué es la nube híbrida?,» [En línea]. Available: https://www.citrix.com/es-es/glossary/what-is-hybrid-cloud.html. [Último acceso: Febrero 2021].

[11] MS Azure, «Tipos de computación en red,» 2021. [En línea]. Available: https://azure.microsoft.com/es-es/overview/types-of-cloud-computing/. [Último acceso: Febrero 2021].

[12] Mediacloud, «XaaS: ¿cuáles son todos los servicios que puedes usar en la nube?,» 2018. [En línea]. Available: https://blog.mdcloud.es/xaas-todos-servicios-nube/. [Último acceso: Febrero 2021].

[13] Monkeylearn, «AI as a Service (AIaaS) - Definition, Types & Top Providers,» 11 Noviembre 2020. [En línea]. Available: https://monkeylearn.com/blog/aiaas/. [Último acceso: 12 Febrero 2021].

[14] MS Azure, «Zonas geográficas de Azure,» 2021. [En línea]. Available: https://azure.microsoft.com/es-es/global-infrastructure/geographies/. [Último acceso: Febrero 2021].

[15] MS Azure, «Paseo por los servicios de Azure,» 2021. [En línea]. Available: https://docs.microsoft.com/es-es/learn/modules/intro-to-azure-fundamentals/tour-of-azure-services. [Último acceso: Enero 2021].

[16] MS Azure, «Computer Vision,» [En línea]. Available: https://azure.microsoft.com/es-es/services/cognitive-services/computer-vision/. [Último acceso: Febrero 2021].

[17] MS Azure, «Administración de las claves de acceso de la cuenta de almacenamiento,» 24 04 2020. [En línea]. Available: https://docs.microsoft.com/es-es/azure/storage/common/storage-account-keys-manage?tabs=azure-portal. [Último acceso: Enero 2021].

[18] Digital guide IONOS, «HTTP Request: los métodos de petición que debes conocer,» 15 Julio 2020. [En línea]. Available: https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/http-request/. [Último acceso: 27 Febrero 2021].

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR