Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

Desarrollo de una herramienta para la anotación asistida de imágenes

---

Alumno: Lucía Cuesta Vicente

Tutor: Aníbal Bregón Bregón

Desarrollo de una herramienta para la anotación asistida de imágenes

Lucía Cuesta Vicente

13 de julio de 2025

A mis padres y a mi hermana,
por ser siempre mi refugio al que volver.

Ignoramos nuestra verdadera estatura hasta que nos ponemos en pie.
— Emily Dickinson

El éxito no se mide en términos de fama o riqueza,
sino en el impacto positivo que tenemos en la vida de los demás.
— Lauren Jauregui

No me despedí
y lo siento.
No me dio tiempo a decir
lo mucho que te quiero.
— Sargento de Hierro, Morgan

Agradecimientos

A mis padres.
A mi madre, por recordarme cada día lo orgullosa que está de mí, y por enseñarme, con su ejemplo, lo que es el amor incondicional y el cariño más sincero.
A mi padre, por enseñarme a no conformarme, a superarme, a crecer como persona... pero a hacerlo siempre a su lado.

A mi hermana.
Por ser el pilar de mi vida. Por estar siempre a mi lado, acompañarme en las ideas más locas, en los momentos más importantes, y por regalarme esa felicidad cada vez que estamos juntas. Porque solo tú y yo nos entendemos. Y porque espero compartir toda mi vida contigo.

A mis abuelos.
Por los que ya no están y que sigo recordando día a día. Y por mi abuela.
Por enseñarme lo bonito de la vida y a valorar el tiempo que compartimos con quienes queremos. Los abuelos venís a este mundo a enseñarnos, y a cuidar de nuestra alma cuando os vais.

A mis amigos.
Por haber estado a mi lado en todo momento, por descubrir la vida juntos, por hacerme feliz cada día y por enseñarme que, a pesar del tiempo, la distancia y nuestras diferencias, me habéis mostrado lo que es tener una segunda familia.

A mis animalillos.
Por llenar mi vida de alegría y mostrarme que la felicidad está en las pequeñas cosas.

Os quiero a cada uno de vosotros.
Espero poder devolveros toda la felicidad que me habéis dado.

Y por último, a mi tutor de TFG, Aníbal. Y a Jorge y Miguel Ángel, por estar presentes en todo momento junto a él. Por vuestra paciencia y dedicación, por enseñarme lo que significa amar vuestra profesión. Y por contagiarnos cada día vuestra pasión y entusiasmo en todo lo que hacéis. Gracias por acompañarme en este camino.

Resumen

El etiquetado y anotación de imágenes es una etapa fundamental para el entrenamiento de modelos de visión por computador. La calidad y cantidad de los datos anotados influyen directamente en el rendimiento de los sistemas basados en inteligencia artificial. Sin embargo, muchas herramientas de anotación actuales presentan barreras de usabilidad o están demasiado especializadas.

Este Trabajo de Fin de Grado presenta el desarrollo de una herramienta genérica de anotación asistida de imágenes, con un enfoque inicial en la localización de objetos. La aplicación permite importar conjuntos de imágenes, definir categorías personalizadas y anotar las imágenes mediante una interfaz de usuario intuitiva.

Además, se ha incorporado un módulo que permite realizar inferencias a partir de las anotaciones, facilitando el ciclo completo de creación y validación de datasets para tareas de detección de objetos. El proyecto incluye también un estudio de herramientas existentes, cuyos elementos más útiles se han integrado en la aplicación desarrollada.

Si bien el enfoque inicial es la localización de objetos, se plantea la posibilidad de ampliar la herramienta hacia tareas de clasificación de imágenes en futuras versiones.

Palabras claves: herramienta de anotación, detección de objetos, visión por computador, inteligencia artificial, interfaz de usuario.

Abstract

Image labeling and annotation is a key stage for training computer vision models. The quality and quantity of the annotated data directly impact the performance of artificial intelligence systems. However, many current annotation tools present usability barriers or are too specialized.

This Final Degree Project presents the development of a generic image annotation tool, initially focused on object localization. The application allows users to import image sets, define custom categories, and annotate images through an intuitive user interface.

In addition, a module has been implemented to perform inference based on the annotations, facilitating the complete cycle of dataset creation and validation for object detection tasks. The project also includes a study of existing tools, incorporating their most useful features into the developed application.

Although the initial focus is object localization, the tool is designed to be extended to image classification tasks in future versions.

Keywords: annotation tool, object detection, computer vision, artificial intelligence, user interface.

Índice general

Índice de figuras

Índice de cuadros

Parte I

Memoria del Proyecto

Capítulo 1

Introducción

El desarrollo de aplicaciones basadas en visión por computador ha crecido exponencialmente en los últimos años, impulsado tanto por los avances en los algoritmos de aprendizaje profundo como por la disponibilidad de potentes infraestructuras de cómputo [20]. Estos avances han permitido que hoy en día los sistemas sean capaces de detectar, clasificar y segmentar objetos en imágenes con un nivel de precisión que, hasta hace poco, parecía inalcanzable.

Sin embargo, para alcanzar estos niveles de rendimiento, es imprescindible contar con conjuntos de datos anotados de forma rigurosa. La calidad y cantidad de los datos utilizados en el entrenamiento de los modelos determinan en gran medida su comportamiento posterior. La anotación de imágenes —ya sea mediante la asignación de etiquetas o mediante la delimitación de regiones de interés— sigue siendo una tarea en gran medida manual y costosa en términos de tiempo.

Existen diversas herramientas que buscan facilitar este proceso, como LabelImg o VoTT [40, 25]. No obstante, muchas de estas soluciones presentan limitaciones en cuanto a usabilidad, flexibilidad o capacidad de integración en flujos de trabajo más amplios. Además, en muchos casos no ofrecen mecanismos que permitan cerrar el ciclo completo de preparación de datasets: desde la anotación inicial hasta la validación mediante modelos de inferencia.

Este Trabajo de Fin de Grado surge precisamente con el objetivo de dar respuesta a estas limitaciones. Se ha desarrollado una herramienta web de anotación asistida de imágenes, centrada en la localización de objetos. La aplicación permite importar conjuntos de imágenes, definir categorías personalizadas y realizar anotaciones mediante la creación de rectángulos delimitadores sobre las imágenes.

Uno de los aspectos diferenciadores de la herramienta desarrollada es la incorporación de funcionalidades que permiten visualizar, de forma integrada, las predicciones generadas por modelos de detección de objetos previamente entrenados. De este modo, se facilita la validación del dataset y el control de la calidad de las anotaciones.

La figura que se muestra a continuación presenta un ejemplo de la interfaz de la herramienta, en la que se puede observar tanto el proceso de anotación como la visualización de las inferencias.

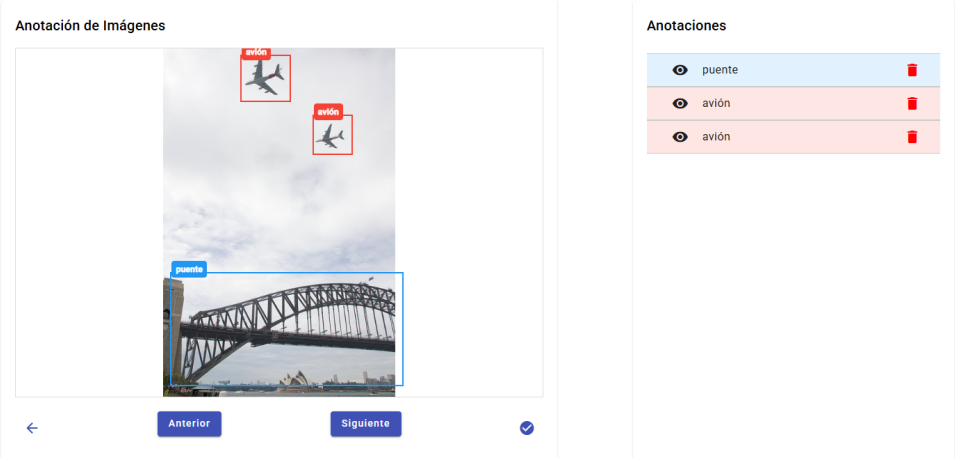


Figura 1.1: Ejemplo de la interfaz de la herramienta de anotación desarrollada.

A lo largo del desarrollo del proyecto, se ha realizado también un análisis de las principales herramientas de anotación actualmente disponibles. Este análisis ha servido como base para la definición de los requisitos funcionales y de diseño de la herramienta propuesta, buscando ofrecer una solución que sea a la vez simple de utilizar y suficientemente potente para su integración en proyectos reales de visión por computadora.

1.1. Planteamiento del problema (*Problem Statement*)

IDEAL:
El desarrollo de modelos de visión por computador requiere datasets de imágenes cuidadosamente anotados, que sirvan como base para el aprendizaje supervisado. En un escenario ideal, los equipos de desarrollo dispondrían de herramientas de anotación intuitivas y flexibles, que facilitarían la creación de estos datasets de manera rápida y eficiente, integrándose fácilmente en el flujo de trabajo completo: anotación, entrenamiento y validación de modelos. Además, estas herramientas permitirían visualizar las inferencias generadas por los modelos sobre las imágenes, facilitando así la validación de la calidad de las anotaciones.

REALIDAD:
En la actualidad, el proceso de anotación de imágenes sigue siendo mayoritariamente manual y consume una gran cantidad de tiempo. Muchas de las herramientas existentes presentan barreras de entrada para usuarios no expertos o están diseñadas para flujos

de trabajo muy concretos que no se adaptan bien a las necesidades de cada proyecto. Además, en muchos casos la integración entre el proceso de anotación y la validación de las predicciones del modelo es limitada o inexistente, lo que dificulta el cierre del ciclo completo de creación de datasets de calidad.

CONSECUENCIAS:
Esta situación provoca que los equipos de desarrollo dediquen más tiempo del necesario a la tarea de anotación, lo que retrasa el proceso global de desarrollo de modelos. Además, la falta de integración entre anotación e inferencia puede dar lugar a inconsistencias en los datasets y a una menor capacidad para detectar y corregir errores en las anotaciones, afectando negativamente al rendimiento de los modelos entrenados.

PROPUESTA:
Este Trabajo de Fin de Grado propone el desarrollo de una herramienta web de anotación asistida de imágenes, que permita realizar de forma sencilla la anotación de imágenes para tareas de localización de objetos, así como visualizar las predicciones de modelos entrenados sobre las imágenes anotadas. De este modo, se busca cerrar el ciclo completo de preparación de datasets y mejorar tanto la eficiencia como la calidad del proceso de anotación.

1.2. Objetivos del trabajo

Este Trabajo de Fin de Grado tiene como objetivo principal el desarrollo de una herramienta web de anotación asistida de imágenes, destinada a facilitar el proceso de generación de datasets para tareas de localización de objetos. A través de este proyecto, se busca mejorar tanto la eficiencia como la calidad del proceso de anotación, integrando además funcionalidades que permitan visualizar las inferencias generadas por modelos previamente entrenados.

A continuación, se detallan los objetivos planteados:

- **OBJ-01:** Analizar las principales herramientas de anotación de imágenes existentes y extraer los requisitos funcionales clave para la herramienta desarrollada.
- **OBJ-02:** Diseñar e implementar una interfaz web intuitiva que permita:
  - **OBJ-02.1:** Importar imágenes y gestionar conjuntos de datos.
  - **OBJ-02.2:** Definir categorías personalizadas y gestionar etiquetas.
  - **OBJ-02.3:** Anotar imágenes mediante la creación de rectángulos delimitadores.
- **OBJ-03:** Implementar funcionalidades de integración con modelos de inferencia que permitan visualizar las predicciones sobre las imágenes anotadas.

- **OBJ-04:** Diseñar un sistema de exportación de anotaciones en formatos estándar (por ejemplo, YOLO, COCO), que facilite su uso posterior en flujos de entrenamiento de modelos de visión por computador.
- **OBJ-05:** Validar la herramienta desarrollada mediante pruebas prácticas con conjuntos de imágenes reales.
- **OBJ-06 (a nivel personal):** Profundizar en el aprendizaje y la aplicación de tecnologías web modernas y de técnicas de visión por computador.

1.2.1. Restricciones

Las restricciones son limitaciones impuestas sobre el diseño y desarrollo del proyecto, que condicionan tanto su planificación como su alcance final. A continuación, se enumeran las principales restricciones identificadas:

- **R-01:** El alcance del proyecto se encuentra limitado por la carga de trabajo establecida para el Trabajo Fin de Grado (12 ECTS).
- **R-02:** El proyecto se ha desarrollado con recursos computacionales limitados. No se ha dispuesto de servidores con GPU para entrenamiento de modelos de detección de objetos, por lo que la herramienta se centra en la fase de anotación y visualización de inferencias, no en el entrenamiento completo de modelos.
- **R-03:** La herramienta está concebida como una solución genérica para tareas de localización de objetos, sin estar adaptada ni optimizada para un dominio industrial concreto.
- **R-04:** El alcance funcional del proyecto incluye la anotación manual asistida y la visualización de inferencias. No se abordan funcionalidades de etiquetado semántico denso (segmentación pixel a pixel), ni anotación de vídeo, ni clasificación de imágenes completa.

1.3. Estructura de la memoria

Esta memoria se estructura en un total de siete capítulos y varios apéndices, organizados de la siguiente manera:

Capítulo 1. Introducción: proporciona el contexto general del proyecto, exponiendo el planteamiento del problema, los objetivos del trabajo y las principales restricciones que condicionan su desarrollo.

Capítulo 2. Planificación: describe la metodología seguida para la gestión del proyecto, así como la planificación temporal, los recursos empleados y la estimación del esfuerzo.

Capítulo 3. Estado del arte y herramientas existentes: presenta un análisis del contexto científico-técnico y una revisión de las principales herramientas de anotación de imágenes actualmente disponibles, sirviendo como base para la definición de los requisitos de la herramienta desarrollada.

Capítulo 4. Análisis y diseño de la herramienta: detalla el proceso de análisis y diseño de la aplicación, incluyendo la definición de los requisitos funcionales y no funcionales, y la arquitectura software planteada.

Capítulo 5. Desarrollo de la herramienta: describe el proceso de implementación de la aplicación, las tecnologías utilizadas y las principales funcionalidades desarrolladas.

Capítulo 6. Validación y resultados: expone las pruebas realizadas para validar el funcionamiento de la herramienta, así como los resultados obtenidos durante su uso con conjuntos de imágenes reales.

Capítulo 7. Conclusiones y trabajos futuros: presenta las conclusiones del proyecto, un balance personal del trabajo realizado y las posibles líneas de mejora y extensión de la herramienta.

Apéndices: recogen documentación complementaria, como manuales de uso y resultados adicionales, que permiten profundizar en determinados aspectos del proyecto.

Capítulo 2

Planificación

En este capítulo se describe la metodología que se ha utilizado para la realización de este Trabajo Fin de Grado (TFG), basada en prácticas ágiles adaptadas al ámbito académico. Asimismo, se presenta la planificación previa del proyecto y el seguimiento que se ha realizado durante su desarrollo.

2.1. Metodología de trabajo

Para la organización y gestión del TFG se ha empleado la metodología ASAP (Agile Student Academic Projects), diseñada específicamente para el ámbito académico y que adapta prácticas ágiles comunes en el sector profesional [23].

ASAP propone una dinámica de trabajo incremental, estructurada en ciclos de corta duración (sprints), basada en la interacción regular entre los participantes (estudiante, tutor, comunidad y tribunal) y en la generación continua de feedback. De este modo, permite mantener un ritmo de trabajo constante y facilita la adaptación progresiva del alcance del proyecto, garantizando que no se supere la carga de trabajo establecida para el TFG (12 ECTS, equivalentes a entre 300 y 360 horas) [29].

Además, la metodología fomenta el desarrollo de competencias clave como la gestión de proyectos, la planificación eficaz, la comunicación (oral y escrita) y el trabajo colaborativo.

El uso de herramientas visuales como tableros Kanban y cuadernos de trabajo facilita el seguimiento del progreso y aporta transparencia al proceso, proporcionando a todos los participantes una visión clara y actualizada del estado del proyecto.

En las siguientes secciones se describen los elementos clave de la metodología aplicada, así como el entorno de trabajo adoptado y la planificación seguida durante el desarrollo del TFG.

2.1.1. Objetivos de aprendizaje

ASAP estructura el TFG en torno a cinco grandes objetivos de aprendizaje:

- **Proyecto:** definir un proyecto que dé respuesta a un problema real, estableciendo un planteamiento del problema claro, unos objetivos concretos y una planificación adecuada.
- **Antecedentes:** adquirir el conocimiento necesario del contexto (de negocio, científico y técnico) en el que se enmarca el TFG, comprendiendo tanto la necesidad que aborda como las soluciones existentes.
- **Desarrollo:** construir el producto de forma iterativa e incremental. En este caso, la herramienta de anotación asistida de imágenes se ha desarrollado mediante incrementos sucesivos que han permitido validar de manera temprana cada componente.
- **Aceptación:** validar el producto desarrollado, evaluando si cumple con los objetivos del proyecto y satisface las necesidades planteadas. En esta etapa se han incorporado pruebas funcionales y evaluaciones de usabilidad.
- **Comunicación:** elaborar la memoria técnica del TFG y preparar su defensa oral ante el tribunal evaluador, trabajando estas competencias de manera progresiva a lo largo de los diferentes sprints.

2.1.2. Roles

Los principales roles definidos en la metodología ASAP son:

Estudiante: responsable del desarrollo del proyecto, de la planificación y ejecución de las tareas y de la comunicación con el tutor y la comunidad.

Tutor: guía y mentor del estudiante, proporciona retroalimentación continua, facilita recursos y asegura un correcto seguimiento del proyecto.

Comunidad: formada por profesores y estudiantes que participan en los eventos abiertos (comunicación de progresos, retrospectivas), aportando feedback complementario y enriqueciendo el aprendizaje del estudiante.

Tribunal: encargado de evaluar el trabajo del estudiante en el acto de defensa, valorando tanto el producto final como la calidad de la exposición y la defensa realizada.

2.1.3. Eventos

El desarrollo del TFG se estructura en sprints de corta duración (inferiores a un mes), que incluyen los siguientes eventos:

- **Reunión de inicio**: se define el objetivo del sprint y se planifican las tareas necesarias para alcanzarlo.
- **Reunión de sincronización**: reunión semanal donde el estudiante informa al tutor sobre los avances realizados, los bloqueos encontrados y la planificación para la siguiente semana.
- **Comunicación de progresos**: presentación del trabajo consolidado al final de cada sprint, en un formato similar a la defensa final, que permite obtener feedback de la comunidad y preparar progresivamente el acto de defensa.
- **Retrospectiva**: reflexión sobre el proceso de trabajo seguido durante el sprint, identificando aspectos positivos y áreas de mejora de cara a los sprints siguientes.

2.1.4. Artefactos

Los principales artefactos generados en el marco de la metodología ASAP son:

- **Incremento**: conjunto de entregables que representan el trabajo realizado en cada sprint. Cada incremento aporta valor tangible y permite validar de manera continua el progreso del proyecto.
- **Retroalimentación**: feedback recibido del tutor y de la comunidad, que permite mejorar tanto el producto como el proceso de desarrollo.

2.1.5. Entorno de trabajo

El entorno de trabajo definido para el desarrollo del proyecto incluye:

- **Espacio de trabajo compartido**: entorno colaborativo basado en Microsoft Teams, que ha facilitado la comunicación continua entre el estudiante, el tutor y la comunidad, así como el almacenamiento y compartición de recursos.
- **Tablero de proyecto (Kanban)**: herramienta visual utilizada para planificar y hacer seguimiento del progreso del proyecto. En este TFG se ha empleado la herramienta Trello (véase Figura 2.1).
- **Cuaderno de trabajo**: documento en el que se registran las tareas realizadas y el tiempo dedicado a cada una, promoviendo la autorregulación y una mejor gestión del tiempo (véase Figura 2.2).

Figura 2.1: Ejemplo del tablero de proyecto utilizado (Trello).

Alumno	Lucía Cuesta Vicente			TFG	Desarrollo de una herramienta para la anotación asistida de imágenes
Fecha	Sprint	Actividad		Tiempo	Descripción
		Tipo	Tarea		
dd/mm/aa	Sprint #1	Dinámica de trabajo	Preparar Incremento	hh:mm:ss	Descripción de la actividad realizada
11/03/2024	Sprint #1	Dinámica de trabajo	Preparar Incremento	1:00:00	Reunión inicial con el tutor para definir el planteamiento del problema y primeros objetivos del TFG.
14/03/2024	Sprint #1	Lectura de documentación	Investigación sobre herramientas de anotación de imágenes	3:30:00	Revisión de LabelImg, VoTT y Roboflow Annotate.
18/03/2024	Sprint #1	Lectura de documentación	Investigación sobre modelos de detección de objetos	4:15:00	Estudio de IceVision, YOLO y Detectron2.
21/03/2024	Sprint #1	Programación	Configuración del entorno de desarrollo	2:00:00	Instalación y configuración de Angular y herramientas de desarrollo.
25/03/2024	Sprint #1	Lectura de documentación	Definir planteamiento del problema	1:30:00	Redacción de la sección 1.1. de la memoria.
28/03/2024	Sprint #1	Programación	Desarrollo sistema de importación de imágenes	4:30:00	Implementación básica del sistema de carga de imágenes.
03/04/2024	Sprint #2	Dinámica de trabajo	Revisión de avances con el tutor	0:45:00	Revisión del Sprint #1 y planificación del Sprint #2.
06/04/2024	Sprint #2	Redacción de memoria	Definir objetivos del TFG	1:30:00	Redacción de la sección 1.2. de la memoria.
09/04/2024	Sprint #2	Programación	Desarrollo sistema de anotación con rectángulos	5:00:00	Desarrollo inicial de la funcionalidad de anotación con rectángulos.
13/04/2024	Sprint #2	Programación	Desarrollo sistema de gestión de etiquetas	4:00:00	Implementación de la gestión y asignación de etiquetas.
17/04/2024	Sprint #2	Programación	Desarrollo de la gestión de anotaciones en miniaturas	4:00:00	Implementación de la visualización de anotaciones en miniaturas.
20/04/2024	Sprint #2	Redacción de memoria	Definir restricciones del proyecto	1:30:00	Redacción de la sección 1.2.1. de la memoria.
24/04/2024	Sprint #2	Lectura de documentación	Investigación sobre formatos de exportación de datasets	2:45:00	Revisión de los formatos YOLO y COCO.
28/04/2024	Sprint #2	Programación	Implementación del guardado automático de anotaciones	3:30:00	Implementación de la funcionalidad de guardado automático.
05/05/2024	Sprint #3	Dinámica de trabajo	Revisión de avances con el tutor	0:45:00	Revisión del Sprint #2 y planificación del Sprint #3.
09/05/2024	Sprint #3	Programación	Implementación de la validación automática de imágenes	4:00:00	Desarrollo del sistema para validar automáticamente las imágenes.
13/05/2024	Sprint #3	Programación	Implementación de la visualización de inferencias	5:00:00	Desarrollo de la funcionalidad para mostrar inferencias sobre las imágenes.
17/05/2024	Sprint #3	Programación	Implementación del sistema de navegación por carpetas	4:30:00	Desarrollo del sistema de navegación por carpetas en la aplicación.
21/05/2024	Sprint #3	Programación	Implementación del sistema de confirmación al salir sin guardar	3:00:00	Desarrollo del sistema de aviso al usuario al salir sin guardar.
05/06/2025	Sprint #3	Redacción de memoria	Redactar Capítulo 1 de la memoria	3:00:00	Redacción completa del Capítulo 1 y revisión final.

Figura 2.2: Ejemplo del cuaderno de trabajo utilizado.

2.2. Planificación temporal

Como se indica en el apartado anterior, este TFG se ha desarrollado siguiendo la metodología ASAP (Agile Student Academic Projects), que organiza el trabajo en sprints. Cada sprint se planifica en la reunión de inicio, donde se establecen los objetivos a alcanzar y las tareas a realizar para cada uno de ellos.

Los sprints tienen una duración aproximada de 4 semanas y una carga de trabajo de entre 75 y 90 horas, adaptándose a la disponibilidad del estudiante en cada fase del curso. La planificación inicial del proyecto abarca el periodo comprendido entre marzo de 2024 y junio de 2025, y se ha estructurado en cuatro sprints principales:

- **Sprint 1:** marzo 2024 – mayo 2024
- **Sprint 2:** junio 2024 – septiembre 2024
- **Sprint 3:** octubre 2024 – febrero 2025
- **Sprint 4:** marzo 2025 – junio 2025

Cada sprint se ha planificado de manera iterativa. Al comienzo de cada uno se celebraba una reunión de inicio donde se definían los objetivos del sprint y se concretaban las tareas necesarias para alcanzarlos. Al finalizar cada sprint se realizaba una comunicación de progresos y una retrospectiva para analizar los resultados y planificar las siguientes iteraciones.

En las siguientes subsecciones se describen los objetivos y tareas de cada sprint, así como el enfoque general seguido en cada fase del proyecto.

2.2.1. Sprint #1

El primer sprint se ha llevado a cabo entre el 1 de marzo de 2024 y el 24 de mayo de 2024. Su principal objetivo fue sentar las bases del proyecto, definir su alcance y contexto, y realizar las primeras investigaciones técnicas. Este sprint ha estado centrado en la adquisición de conocimiento sobre las tecnologías y herramientas necesarias para el desarrollo de una herramienta de anotación asistida de imágenes, así como en la correcta planificación del proyecto y en la elaboración de los primeros entregables.

Además, durante este sprint se definieron las dinámicas de trabajo, se estableció el tablero Kanban y se realizaron las primeras reuniones siguiendo la metodología ASAP, garantizando una adecuada organización del trabajo.

Objetivos abordados: Proyecto, Antecedentes y Comunicación.

Historias y tareas:

- **H1.1 Planificación del sprint** *(01/03/2024 - 05/03/2024)*
  - T1.1.1 Definición de los objetivos del sprint.
  - T1.1.2 Planificación de tareas a realizar durante el sprint.
  - T1.1.3 Establecimiento del tablero Kanban.
  - T1.1.4 Definición del calendario de reuniones del sprint.
- **H1.2 Desarrollo del sprint** *(05/03/2024 - 20/05/2024)*
  - T1.2.1 Reuniones de sincronización semanales.
  - T1.2.2 Actualización continua del tablero Kanban.
- **H1.3 Caracterización del proyecto** *(06/03/2024 - 12/03/2024)*
  - T1.3.1 Definición del planteamiento del problema.
  - T1.3.2 Identificación y especificación de los objetivos del proyecto.
  - T1.3.3 Identificación de restricciones y requisitos iniciales.
  - T1.3.4 Definición de la estructura de la memoria.
- **H1.4 Estado del arte** *(13/03/2024 - 29/03/2024)*
  - T1.4.1 Revisión de técnicas y herramientas de anotación de imágenes.
  - T1.4.2 Estudio de arquitecturas de detección de objetos (YOLO, IceVision).
  - T1.4.3 Revisión de formatos de anotación (YOLO, COCO).
- **H1.5 Contexto científico-técnico** *(01/04/2024 - 15/04/2024)*
  - T1.5.1 Estudio de frameworks de deep learning para detección de objetos.
  - T1.5.2 Análisis de librerías disponibles en Python para anotación de imágenes.
  - T1.5.3 Estudio de buenas prácticas en la anotación de datasets.
- **H1.6 Memoria** *(16/04/2024 - 10/05/2024)*
  - T1.6.1 Redacción parcial del capítulo de Introducción.
  - T1.6.2 Redacción parcial del capítulo de Planificación.
  - T1.6.3 Redacción parcial del capítulo de Antecedentes.
- **H1.7 Presentación** *(11/05/2024 - 20/05/2024)*
  - T1.7.1 Elaboración de la presentación del primer sprint.

- **H1.8 Finalización del sprint** *(21/05/2024 - 24/05/2024)*
  - T1.8.1 Preparación y realización de la comunicación de progresos.
  - T1.8.2 Reflexión y retrospectiva sobre el sprint.

2.2.2. Sprint #2

El segundo sprint se ha llevado a cabo entre junio de 2024 y septiembre de 2024. En este sprint se ha abordado el desarrollo de la primera versión de la herramienta de anotación, incluyendo la implementación del prototipo inicial, así como la integración básica de funcionalidades clave. Además, se ha continuado con la documentación del proyecto y con la presentación de avances a la comunidad.

Este sprint ha tenido un enfoque claramente técnico, centrado en la construcción de la herramienta, sin perder de vista la mejora continua del proceso gracias a la retroalimentación obtenida durante el primer sprint.

Objetivos abordados: Proyecto, Desarrollo y Comunicación.

Historias y tareas:

- **H2.1 Planificación del sprint** *(01/06/2024 - 05/06/2024)*
  - T2.1.1 Definición de los objetivos del sprint.
  - T2.1.2 Planificación de tareas a realizar durante el sprint.
  - T2.1.3 Revisión y actualización del tablero Kanban.
  - T2.1.4 Definición del calendario de reuniones del sprint.
- **H2.2 Desarrollo del sprint** *(06/06/2024 - 20/09/2024)*
  - T2.2.1 Reuniones de sincronización semanales.
  - T2.2.2 Actualización continua del tablero Kanban.
- **H2.3 Implementación del prototipo de la herramienta** *(06/06/2024 - 25/06/2024)*
  - T2.3.1 Implementación del contenedor principal de la interfaz de usuario (Angular).
  - T2.3.2 Implementación del componente de visualización de imágenes.
  - T2.3.3 Implementación de la herramienta de dibujo de anotaciones (rectángulos).
  - T2.3.4 Implementación del almacenamiento de anotaciones en local.

- **H2.4 Integración de funcionalidades básicas** *(26/06/2024 - 15/07/2024)*
  - T2.4.1 Implementación de la importación de imágenes.
  - T2.4.2 Implementación de la exportación de anotaciones en formato YOLO.
  - T2.4.3 Implementación del control de visibilidad de anotaciones.
  - T2.4.4 Implementación de la navegación por miniaturas.
- **H2.5 Integración con el backend** *(16/07/2024 - 31/07/2024)*
  - T2.5.1 Implementación de la subida de imágenes al servidor (Flask).
  - T2.5.2 Implementación de la persistencia de anotaciones en el backend.
  - T2.5.3 Pruebas de integración entre frontend y backend.
- **H2.6 Memoria** *(01/08/2024 - 10/09/2024)*
  - T2.6.1 Redacción del capítulo de Análisis.
  - T2.6.2 Redacción del capítulo de Diseño.
  - T2.6.3 Actualización del capítulo de Introducción.
  - T2.6.4 Actualización del capítulo de Planificación.
- **H2.7 Presentación** *(11/09/2024 - 15/09/2024)*
  - T2.7.1 Elaboración de la presentación del segundo sprint.
- **H2.8 Finalización del sprint** *(16/09/2024 - 20/09/2024)*
  - T2.8.1 Preparación y realización de la comunicación de progresos.
  - T2.8.2 Reflexión y retrospectiva sobre el sprint.

2.2.3. Sprint #3

El tercer sprint se ha llevado a cabo entre octubre de 2024 y febrero de 2025. En este sprint se ha avanzado de forma significativa en el desarrollo de la herramienta de anotación asistida de imágenes, completando gran parte de las funcionalidades previstas. Además, se ha comenzado a trabajar en la integración de un sistema de inferencia con modelos de detección de objetos, así como en la mejora de la experiencia de usuario.

Por otro lado, se ha continuado con la redacción de la memoria y se ha preparado la correspondiente presentación de avances, consolidando los resultados obtenidos durante el sprint. Éste ha supuesto un gran salto cualitativo en el prototipo y ha permitido sentar las bases para el trabajo de integración y validación que se realizará en el sprint final.

Objetivos abordados: Proyecto, Desarrollo y Comunicación.

Historias y tareas:

- **H3.1 Planificación del sprint** *(01/10/2024 - 05/10/2024)*
  - T3.1.1 Definición de los objetivos del sprint.
  - T3.1.2 Planificación de tareas a realizar durante el sprint.
  - T3.1.3 Revisión y actualización del tablero Kanban.
  - T3.1.4 Definición del calendario de reuniones del sprint.
- **H3.2 Desarrollo del sprint** *(06/10/2024 - 20/02/2025)*
  - T3.2.1 Reuniones de sincronización semanales.
  - T3.2.2 Actualización continua del tablero Kanban.
- **H3.3 Implementación avanzada de la herramienta** *(06/10/2024 - 15/12/2024)*
  - T3.3.1 Implementación del sistema de validación de imágenes.
  - T3.3.2 Implementación de la funcionalidad de edición y borrado de anotaciones.
  - T3.3.3 Implementación de la confirmación al salir sin guardar.
  - T3.3.4 Mejora de la gestión de miniaturas de imágenes.
- **H3.4 Integración del sistema de inferencia** *(16/12/2024 - 20/01/2025)*
  - T3.4.1 Preparación del entorno de inferencia.
  - T3.4.2 Implementación del endpoint de inferencia en Flask.
  - T3.4.3 Integración del endpoint en la herramienta de anotación.
  - T3.4.4 Pruebas iniciales del sistema de inferencia.
- **H3.5 Experiencia de usuario y mejoras de interfaz** *(10/01/2025 - 31/01/2025)*
  - T3.5.1 Revisión de la navegación y flujo de uso.
  - T3.5.2 Mejora de la usabilidad en el proceso de anotación.
  - T3.5.3 Adaptación de la interfaz para dispositivos de diferentes tamaños.
- **H3.6 Memoria** *(01/02/2025 - 15/02/2025)*
  - T3.6.1 Redacción del capítulo de Implementación.
  - T3.6.2 Revisión y actualización del capítulo de Diseño.
  - T3.6.3 Actualización del capítulo de Introducción.

- **H3.7 Presentación** *(16/02/2025 - 20/02/2025)*
  - T3.7.1 Elaboración de la presentación del tercer sprint.
- **H3.8 Finalización del sprint** *(21/02/2025 - 25/02/2025)*
  - T3.8.1 Preparación y realización de la comunicación de progresos.
  - T3.8.2 Reflexión y retrospectiva sobre el sprint.

2.2.4. Sprint #4

El cuarto sprint se ha llevado a cabo entre marzo de 2025 y junio de 2025. Durante este sprint se han realizado los trabajos finales del proyecto, incluyendo la integración completa de funcionalidades avanzadas en la herramienta, la validación mediante pruebas, la finalización de la documentación y la preparación para la defensa del TFG.

Este sprint ha sido clave para consolidar todos los avances realizados en las fases anteriores, asegurando que la herramienta estuviera completamente funcional, bien documentada y preparada para ser presentada. Se han realizado pruebas exhaustivas para garantizar la calidad del producto final, y se ha trabajado intensamente en la elaboración de la memoria y la preparación de la presentación final.

Objetivos abordados: Desarrollo, Calidad, Comunicación y Proyecto.

Historias y tareas:

- **H4.1 Planificación del sprint** *(01/03/2025 - 05/03/2025)*
  - T4.1.1 Definición de los objetivos del sprint.
  - T4.1.2 Planificación de tareas a realizar durante el sprint.
  - T4.1.3 Revisión y actualización del tablero Kanban.
  - T4.1.4 Definición del calendario de reuniones del sprint.
- **H4.2 Desarrollo del sprint** *(06/03/2025 - 05/06/2025)*
  - T4.2.1 Reuniones de sincronización semanales.
  - T4.2.2 Actualización continua del tablero Kanban.
- **H4.3 Finalización de funcionalidades avanzadas** *(06/03/2025 - 10/04/2025)*
  - T4.3.1 Implementación de la exportación en formato COCO.
  - T4.3.2 Implementación de edición de anotaciones existentes.

  - T4.3.3 Implementación de borrado individual de anotaciones.
  - T4.3.4 Implementación de persistencia de anotaciones validadas y su visualización en miniaturas.
  - T4.3.5 Implementación de confirmación al salir sin guardar.
  - T4.3.6 Implementación del rediseño final de la interfaz.

- **H4.4 Pruebas de validación de la herramienta** *(11/04/2025 - 30/04/2025)*
  - T4.4.1 Definición del plan de pruebas.
  - T4.4.2 Ejecución de pruebas funcionales.
  - T4.4.3 Ejecución de pruebas de usabilidad.
  - T4.4.4 Registro de incidencias y corrección de errores detectados.
  - T4.4.5 Elaboración del informe de pruebas.

- **H4.5 Memoria** *(01/05/2025 - 05/06/2025)*
  - T4.5.1 Redacción final del capítulo de Análisis.
  - T4.5.2 Redacción final del capítulo de Diseño.
  - T4.5.3 Redacción final del capítulo de Implementación.
  - T4.5.4 Redacción final del capítulo de Validación.
  - T4.5.5 Redacción del capítulo de Balance temporal y económico.
  - T4.5.6 Revisión completa y corrección de estilo de la memoria.

- **H4.6 Presentación** *(06/06/2025 - 10/06/2025)*
  - T4.6.1 Elaboración de la presentación final del TFG.
  - T4.6.2 Preparación de la defensa.
  - T4.6.3 Ensayo de la defensa con feedback de tutores.

- **H4.7 Finalización del sprint** *(11/06/2025 - 15/06/2025)*
  - T4.7.1 Preparación y realización de la comunicación final de progresos.
  - T4.7.2 Reflexión y retrospectiva sobre el sprint y el proyecto completo.

2.3. Presupuestos

En esta sección se presenta un análisis detallado de los presupuestos necesarios para llevar a cabo el Trabajo de Fin de Grado. El presupuesto se divide en tres categorías: hardware, software y recursos humanos. Su objetivo es proporcionar una estimación realista del coste que supondría realizar este proyecto en un contexto profesional.

La estructura de este apartado sigue la línea de trabajos anteriores realizados con la metodología ASAP [23], incluyendo los costes estimados de hardware, software y esfuerzo humano.

2.3.1. Hardware

En cuanto al **hardware**, el trabajo se ha desarrollado principalmente utilizando un equipo personal de gama media-alta, con especificaciones suficientes para las tareas de desarrollo y entrenamiento del modelo. Durante el entrenamiento de modelos de detección de objetos, que requiere un alto rendimiento computacional, se ha recurrido puntualmente al uso de un servidor con GPU dedicada, alquilado bajo demanda, lo que ha permitido optimizar el coste en función de las necesidades concretas de procesamiento.

Además, se ha utilizado una conexión a internet doméstica para acceder a herramientas colaborativas, consultar documentación y realizar las tareas habituales de desarrollo, así como para el uso de plataformas en la nube durante parte del entrenamiento.

A continuación, se presenta el desglose detallado del presupuesto estimado en la categoría de hardware:

Componente	**Coste total**	**Vida útil**	**% uso**	**Uso**	**Coste real**
Ordenador personal (i7, 16 GB RAM, SSD)	1200€	60 meses	30 %	12 meses	72,00€
Conexión a internet doméstica	30 €/mes	–	–	12 meses	360,00€
Servidor con GPU (uso puntual)	5 €/hora	–	–	15 horas	75,00€
Total hardware	**507,00€**				

Cuadro 2.1: Presupuesto hardware.

2.3.2. Software

En lo que respecta al **software**, se han utilizado exclusivamente herramientas de uso gratuito o con licencia académica, lo cual ha permitido cubrir todas las necesidades del proyecto sin generar costes adicionales.

Estas herramientas han sido empleadas tanto para el desarrollo de la aplicación, como para la colaboración, la gestión del proyecto y la documentación. Su elección ha permitido optimizar el flujo de trabajo y garantizar la trazabilidad y calidad de los entregables.

A continuación, se muestra el desglose de las herramientas empleadas:

Herramienta	**Coste mensual (€)**	**% uso**	**Uso (meses)**	**Coste real (€)**
Overleaf	0	100 %	12	0
Trello	0	100 %	12	0
Microsoft Teams	0	100 %	12	0
Google Colab	0	100 %	6	0
Visual Studio Code	0	100 %	12	0
Draw.io	0	100 %	12	0
Total software	**0**			

Cuadro 2.2: Presupuesto software.

2.3.3. Recursos humanos

El proyecto ha sido desarrollado íntegramente por una única persona, quien ha desempeñado diferentes roles a lo largo del ciclo de vida del mismo. Esta polivalencia ha permitido afrontar todas las fases del proyecto —desde la planificación inicial hasta la documentación final— de forma autónoma.

Para estimar el coste asociado a los **recursos humanos**, se han considerado los salarios medios en España para perfiles junior en cada uno de los roles desempeñados, según las referencias consultadas [12], [28].

Se ha estimado una dedicación total de **360 horas**, que se ha distribuido entre las siguientes funciones:

Rol	Salario hora (€)	Horas estim.	Coste total (€)
Gestor de proyecto	12,00	20	240,00
Analista	15,00	80	1200,00
Científico de datos	18,00	120	2160,00
Programador	14,00	100	1400,00
Tester de software	13,00	20	260,00
Documentalista	11,00	20	220,00
Subtotal recursos humanos	**5480,00**		
+ Seguridad Social (30,9 %)	**1692,32**		
Total recursos humanos	**7172,32**		

Cuadro 2.3: Presupuesto recursos humanos.

2.3.4. Presupuesto total del proyecto

Finalmente, en la siguiente tabla se presenta un resumen del **presupuesto total estimado** para el desarrollo completo del proyecto, consolidando los costes correspondientes a hardware, software y recursos humanos.

Este presupuesto proporciona una visión completa de los recursos necesarios para llevar a cabo el Trabajo de Fin de Grado, reflejando tanto los costes directos como el esfuerzo humano requerido en cada fase del proyecto.

Concepto	Coste (€)
Hardware	507,00
Software	0
Recursos Humanos	7172,32
TOTAL	**7679,32**

Cuadro 2.4: Presupuesto total del proyecto.

2.4. Gestión de riesgos

Una adecuada **gestión de riesgos** permite identificar de manera anticipada aquellos factores que podrían comprometer el éxito del proyecto, minimizando así su impacto y facilitando la adopción de medidas preventivas y correctivas.

Este proceso se basa en identificar los riesgos potenciales, estimar su probabilidad e impacto, y definir planes de contingencia que permitan mitigar sus efectos negativos.

La estructura de este apartado sigue la línea de trabajos anteriores realizados con la metodología ASAP [23], estableciendo un enfoque sistemático para la gestión de riesgos a lo largo del proyecto.

2.4.1. Identificación de riesgos

El primer paso en la gestión de riesgos ha consistido en identificar aquellos factores que podrían afectar negativamente al desarrollo del proyecto. A continuación, se presenta un listado de los riesgos potenciales detectados, con una breve descripción de cada uno.

Código	Descripción del riesgo
R-01	Retrasos en el desarrollo respecto a la planificación inicial.
R-02	Complejidad técnica en la integración completa del modelo de detección con la herramienta de anotación.
R-03	Limitaciones de rendimiento del equipo personal para tareas de entrenamiento intensivo.
R-04	Posible falta de experiencia previa con frameworks avanzados de deep learning.
R-05	Incompatibilidades entre versiones de librerías utilizadas en el proyecto.
R-06	Pérdida de datos debido a un fallo en el sistema de almacenamiento.

Cuadro 2.5: Listado de riesgos identificados.

2.4.2. Estimación de probabilidad de ocurrencia

Una vez identificados los riesgos, se ha procedido a estimar su **probabilidad de ocurrencia**. Esta probabilidad representa la posibilidad de que el riesgo llegue a materializarse durante el desarrollo del proyecto, considerando las características concretas del contexto y la naturaleza de las tareas a realizar. La siguiente tabla muestra la probabilidad estimada y la justificación correspondiente.

Riesgo	Probabilidad (%)	Justificación
R-01	30 %	Es posible que surjan imprevistos académicos u otras prioridades que afecten al ritmo de trabajo.
R-02	40 %	La integración de modelos de IA en aplicaciones web puede presentar retos técnicos significativos.
R-03	70 %	El equipo personal no dispone de GPU de alto rendimiento, por lo que puede haber limitaciones en tareas pesadas.
R-04	60 %	No se cuenta con experiencia previa en todas las herramientas avanzadas utilizadas.
R-05	20 %	Las versiones de las librerías pueden evolucionar rápidamente, generando incompatibilidades.
R-06	10 %	Aunque se hace uso de almacenamiento en la nube, siempre existe un riesgo residual de pérdida de datos.

Cuadro 2.6: Probabilidad de ocurrencia de los riesgos.

2.4.3. Impacto de los riesgos

Tras analizar la probabilidad, se ha evaluado el **impacto potencial** que tendría cada riesgo en caso de que llegase a materializarse. Este impacto se valora en una escala de 1 a 5, donde 1 corresponde a un impacto muy bajo y 5 a un impacto crítico que podría comprometer seriamente el éxito del proyecto. La tabla siguiente resume esta evaluación.

Riesgo	Impacto (1-5)	Justificación
R-01	4	Un retraso importante podría comprometer las fechas de entrega previstas.
R-02	5	La integración fallida del modelo afectaría directamente a la funcionalidad clave de la herramienta.
R-03	3	La limitación de rendimiento puede ralentizar el desarrollo, pero existen alternativas (uso de servidores en la nube).
R-04	3	La falta de experiencia puede traducirse en una curva de aprendizaje más prolongada.
R-05	2	Las incompatibilidades pueden retrasar ciertas tareas, pero suelen tener solución técnica.
R-06	5	La pérdida de datos sin copia de seguridad podría suponer rehacer parte del trabajo.

Cuadro 2.7: Impacto de los riesgos.

2.4.4. Matriz Probabilidad × Impacto

Para priorizar los riesgos de forma efectiva, se ha calculado un **índice de exposición** para cada riesgo, obtenido como el producto de su probabilidad y su impacto. Este valor permite identificar de manera sencilla los riesgos que requieren una atención prioritaria. La siguiente tabla muestra los resultados obtenidos.

Riesgo	Probabilidad (0-5)	Impacto (1-5)	Exposición (P × I)
R-01	2	4	8
R-02	3	5	15
R-03	4	3	12
R-04	3	3	9
R-05	1	2	2
R-06	1	5	5

Cuadro 2.8: Matriz de Prioridad de riesgos.

2.4.5. Plan de contingencia

Finalmente, se han definido una serie de **planes de contingencia** para los riesgos identificados, con el fin de minimizar su impacto en caso de que llegasen a producirse. La siguiente tabla resume las medidas preventivas o correctivas previstas para cada riesgo.

Riesgo	**Plan de contingencia**
R-02	Utilizar modelos previamente entrenados o simplificados en caso de no lograr la integración completa.
R-03	Externalizar el entrenamiento a servidores en la nube (por ejemplo, Google Colab Pro o servicios de AWS).
R-01	Revisar y actualizar periódicamente la planificación para garantizar márgenes de seguridad.
R-04	Dedicación adicional a la formación en las herramientas necesarias mediante cursos y documentación oficial.
R-06	Realizar copias de seguridad periódicas en la nube y en almacenamiento físico externo.
R-05	Mantener actualizadas las dependencias y utilizar entornos virtuales para evitar conflictos entre versiones.

Cuadro 2.9: Plan de contingencia para los riesgos identificados.

En resumen, este análisis de riesgos ha permitido establecer un marco preventivo que contribuirá a garantizar el correcto desarrollo del proyecto, anticipando posibles problemas y estableciendo las medidas necesarias para su mitigación.

2.5. Balance temporal y económico

Esta sección presenta un análisis comparativo entre la planificación inicial del proyecto y su desarrollo real, tanto desde el punto de vista temporal como económico.

El objetivo es reflejar de manera transparente cómo ha evolucionado el proyecto a lo largo de los diferentes sprints, qué desviaciones se han producido y cuál ha sido el coste final en comparación con el presupuesto estimado.

Además, se incluyen los diagramas de Gantt que ilustran el transcurso real de las tareas, facilitando la identificación de posibles desviaciones respecto a la planificación inicial.

2.5.1. Balance temporal

En esta sección se presenta el balance temporal real del proyecto, analizando los posibles desfases o desviaciones que se han producido respecto a la planificación inicial.

El proyecto se ha desarrollado en un total de **4 sprints**, siguiendo la metodología ASAP. Aunque la planificación establecía una carga de trabajo aproximada de entre **75 y 90 horas por sprint**, la dedicación real ha variado ligeramente en función de la dificultad de las tareas, la carga académica en otras asignaturas y situaciones personales a lo largo del curso.

El total de horas dedicadas al proyecto ha sido de aproximadamente **286 horas**, una cifra coherente con la carga prevista para un Trabajo de Fin de Grado de **12 ECTS**.

A continuación se describen los aspectos más relevantes de la ejecución temporal de cada sprint.

Sprint #1

Este primer sprint se desarrolló de acuerdo a lo planificado, si bien hubo que dedicar más tiempo del previsto a la búsqueda y estudio de herramientas y técnicas de anotación de imágenes. Este trabajo adicional provocó un pequeño desfase en el inicio de la redacción de los primeros capítulos de la memoria, que se recuperó en las últimas semanas del sprint.

Por otro lado, el calendario de reuniones se pudo mantener de forma estable durante todo el periodo.

Dedicación real aproximada: 69 horas.

Sprint #2

Durante este sprint surgieron ciertos desafíos en la integración inicial entre el frontend y el backend, especialmente relacionados con la gestión de la persistencia de anotaciones. Esto provocó que las tareas de integración se alargaran más de lo previsto.

Además, la carga de trabajo de otras asignaturas en el mes de septiembre hizo que se ralentizara ligeramente la redacción del capítulo de Análisis, que se completó finalmente a principios del siguiente sprint.

Dedicación real aproximada: 80 horas.

Sprint #3

El sprint 3 estuvo condicionado por la complejidad de las tareas de integración del sistema de inferencia, que requirieron más tiempo del inicialmente previsto debido a problemas con la compatibilidad de algunas versiones de librerías.

Asimismo, se dedicaron esfuerzos adicionales a la mejora de la experiencia de usuario y la usabilidad de la interfaz, tareas que no estaban previstas con tanto detalle en la planificación inicial, pero que resultaron esenciales para garantizar la calidad del producto final.

A pesar de estos ajustes, se pudieron cumplir los objetivos globales del sprint dentro de unos márgenes razonables.

Dedicación real aproximada: 71 horas.

Sprint #4

El último sprint fue el más intenso en cuanto a dedicación y concentración de tareas. El proceso de validación de la herramienta, así como la finalización de la memoria, requirieron un esfuerzo considerable.

En concreto, la integración final de las funcionalidades avanzadas y el rediseño de la interfaz presentaron algunos bloqueos técnicos que provocaron retrasos en la planificación prevista. Sin embargo, estos fueron solventados con un esfuerzo adicional en las últimas semanas del sprint.

Por otro lado, la preparación de la defensa y la revisión completa de la memoria coincidieron con el cierre de otras asignaturas, lo que supuso un reto adicional en la gestión del tiempo disponible.

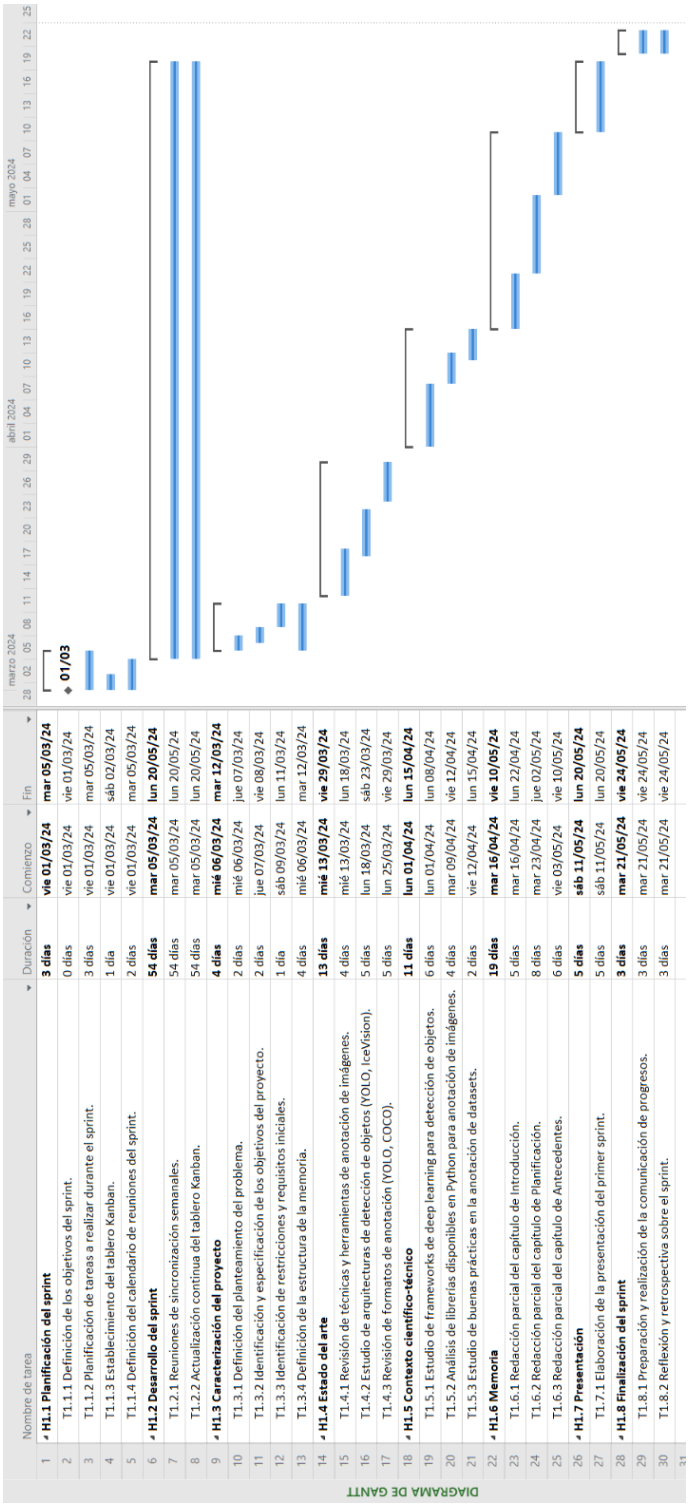
Dedicación real aproximada: 66 horas.

Conclusión general

En conjunto, el proyecto se ha desarrollado de manera satisfactoria en cuanto al cumplimiento de la planificación temporal. Si bien se han producido algunos pequeños retrasos en tareas concretas, principalmente en las fases de integración y validación, estos han sido compensados en la medida de lo posible y no han afectado significativamente al resultado global del trabajo.

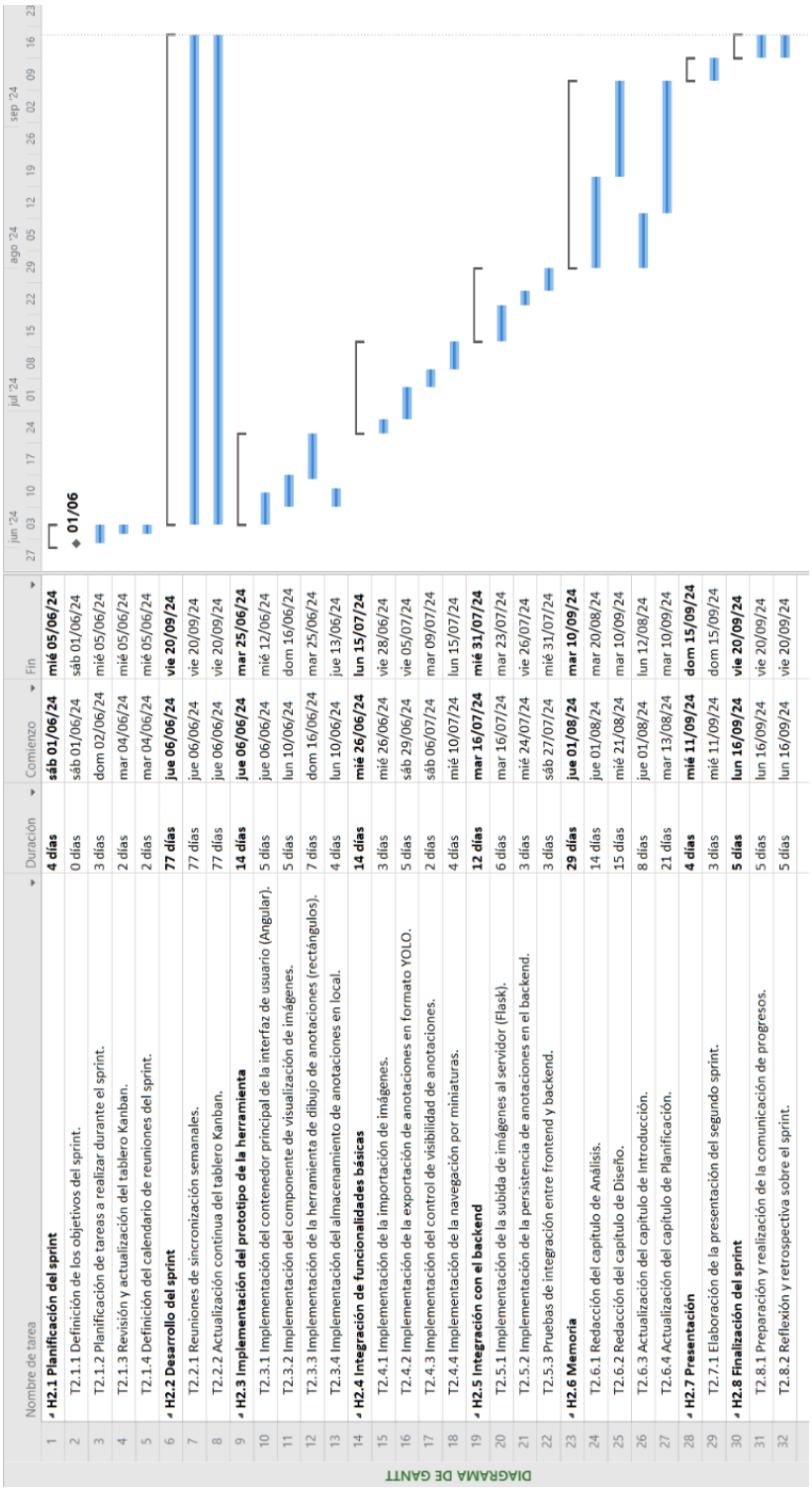
El total de **286 horas dedicadas** al proyecto se considera coherente con el alcance y los objetivos planteados, así como con la carga de trabajo esperada para un Trabajo de Fin de Grado de **12 ECTS**.

A continuación se muestran los diagramas de Gantt correspondientes a cada sprint, donde se refleja el tiempo dedicado a la realización de las tareas planificadas durante el desarrollo del proyecto.



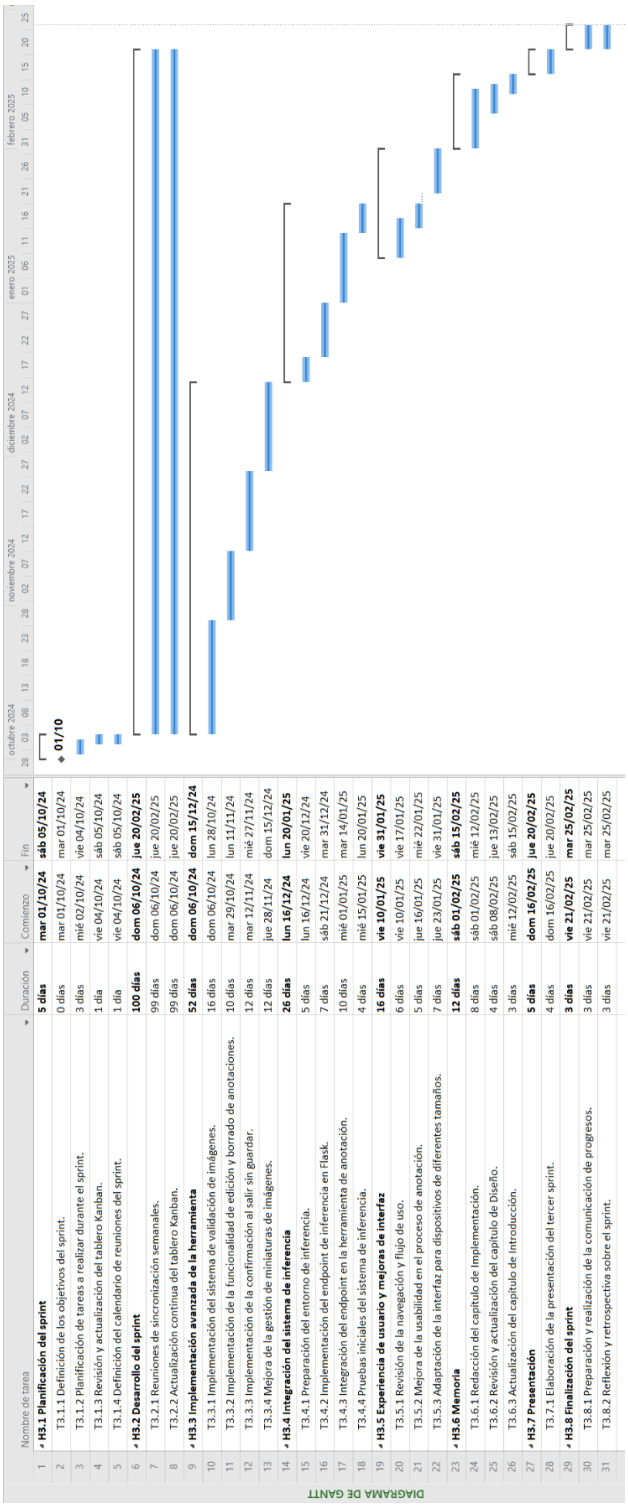
	Nombre de tarea	Duración	Comienzo	Fin
1	**H1.1 Planificación del sprint**	**3 días**	**vie 01/03/24**	**mar 05/03/24**
2	T1.1.1 Definición de los objetivos del sprint.	0 días	vie 01/03/24	vie 01/03/24
3	T1.1.2 Planificación de tareas a realizar durante el sprint.	3 días	vie 01/03/24	mar 05/03/24
4	T1.1.3 Establecimiento del tablero Kanban.	1 día	vie 01/03/24	sáb 02/03/24
5	T1.1.4 Definición del calendario de reuniones del sprint.	2 días	vie 01/03/24	mar 05/03/24
6	**H1.2 Desarrollo del sprint**	**54 días**	**mar 05/03/24**	**lun 20/05/24**
7	T1.2.1 Reuniones de sincronización semanales.	54 días	mar 05/03/24	lun 20/05/24
8	T1.2.2 Actualización continua del tablero Kanban.	54 días	mar 05/03/24	lun 20/05/24
9	**H1.3 Caracterización del proyecto**	**4 días**	**mié 06/03/24**	**mar 12/03/24**
10	T1.3.1 Definición del planteamiento del problema.	2 días	mié 06/03/24	jue 07/03/24
11	T1.3.2 Identificación y especificación de los objetivos del proyecto.	2 días	jue 07/03/24	vie 08/03/24
12	T1.3.3 Identificación de restricciones y requisitos iniciales.	1 día	sáb 09/03/24	lun 11/03/24
13	T1.3.4 Definición de la estructura de la memoria.	4 días	mié 06/03/24	mar 12/03/24
14	**H1.4 Estado del arte**	**13 días**	**mié 13/03/24**	**vie 29/03/24**
15	T1.4.1 Revisión de técnicas y herramientas de anotación de imágenes.	4 días	mié 13/03/24	lun 18/03/24
16	T1.4.2 Estudio de arquitecturas de detección de objetos (YOLO, IceVision).	5 días	lun 18/03/24	sáb 23/03/24
17	T1.4.3 Revisión de formatos de anotación (YOLO, COCO).	5 días	lun 25/03/24	vie 29/03/24
18	**H1.5 Contexto científico-técnico**	**11 días**	**lun 01/04/24**	**lun 15/04/24**
19	T1.5.1 Estudio de frameworks de deep learning para detección de objetos.	6 días	lun 01/04/24	lun 08/04/24
20	T1.5.2 Análisis de librerías disponibles en Python para anotación de imágenes.	4 días	mar 09/04/24	vie 12/04/24
21	T1.5.3 Estudio de buenas prácticas en la anotación de datasets.	2 días	vie 12/04/24	lun 15/04/24
22	**H1.6 Memoria**	**19 días**	**mar 16/04/24**	**vie 10/05/24**
23	T1.6.1 Redacción parcial del capítulo de Introducción.	5 días	mar 16/04/24	lun 22/04/24
24	T1.6.2 Redacción parcial del capítulo de Planificación.	8 días	mar 23/04/24	jue 02/05/24
25	T1.6.3 Redacción parcial del capítulo de Antecedentes.	6 días	vie 03/05/24	vie 10/05/24
26	**H1.7 Presentación**	**5 días**	**sáb 11/05/24**	**lun 20/05/24**
27	T1.7.1 Elaboración de la presentación del primer sprint.	5 días	sáb 11/05/24	lun 20/05/24
28	**H1.8 Finalización del sprint**	**3 días**	**mar 21/05/24**	**vie 24/05/24**
29	T1.8.1 Preparación y realización de la comunicación de progresos.	3 días	mar 21/05/24	vie 24/05/24
30	T1.8.2 Reflexión y retrospectiva sobre el sprint.	3 días	mar 21/05/24	vie 24/05/24

Figura 2.3: Diagrama de Gantt Sprint #1.



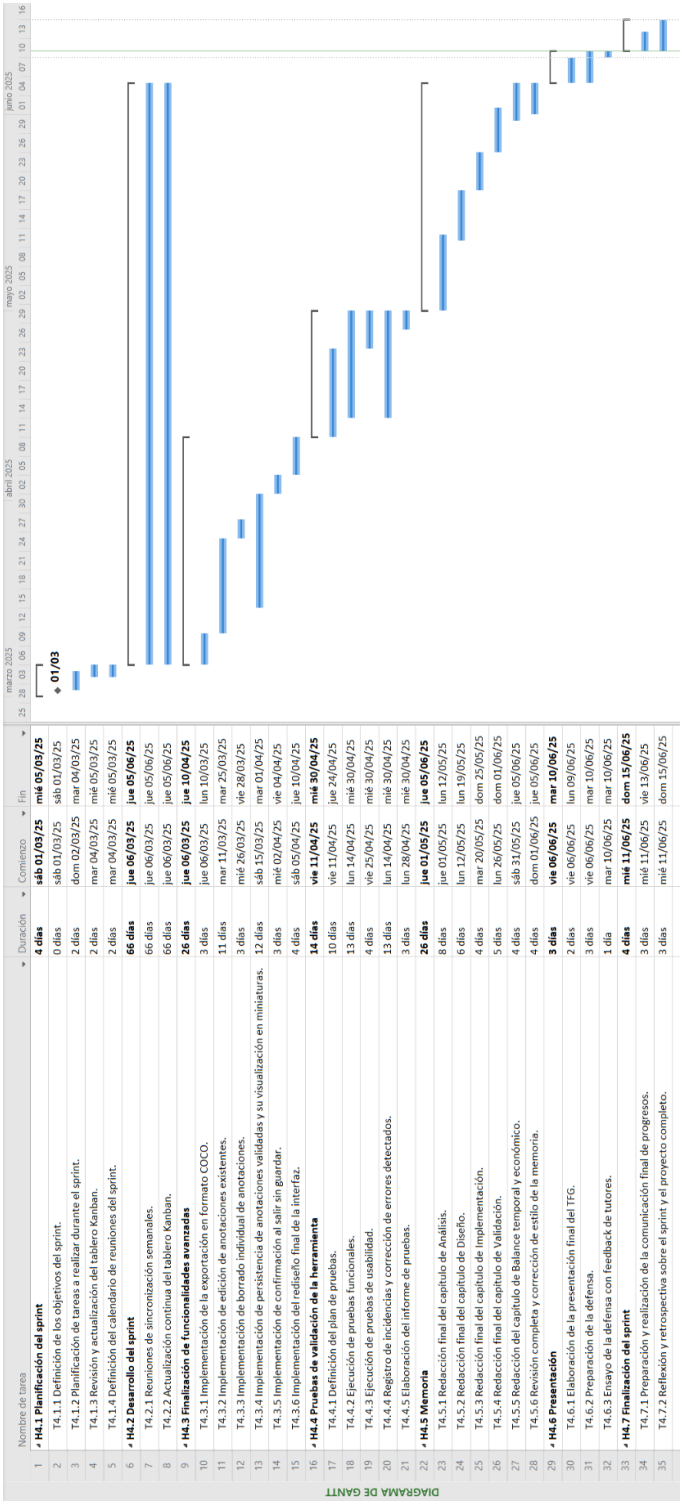
	Nombre de tarea	Duración	Comienzo	Fin
1	**H2.1 Planificación del sprint**	**4 días**	**sáb 01/06/24**	**mié 05/06/24**
2	T2.1.1 Definición de los objetivos del sprint.	0 días	sáb 01/06/24	sáb 01/06/24
3	T2.1.2 Planificación de tareas a realizar durante el sprint.	3 días	dom 02/06/24	mié 05/06/24
4	T2.1.3 Revisión y actualización del tablero Kanban.	2 días	mar 04/06/24	mié 05/06/24
5	T2.1.4 Definición del calendario de reuniones del sprint.	2 días	mar 04/06/24	mié 05/06/24
6	**H2.2 Desarrollo del sprint**	**77 días**	**jue 06/06/24**	**vie 20/09/24**
7	T2.2.1 Reuniones de sincronización semanales.	77 días	jue 06/06/24	vie 20/09/24
8	T2.2.2 Actualización continua del tablero Kanban.	77 días	jue 06/06/24	vie 20/09/24
9	**H2.3 Implementación del prototipo de la herramienta**	**14 días**	**jue 06/06/24**	**mar 25/06/24**
10	T2.3.1 Implementación del contenedor principal de la interfaz de usuario (Angular).	5 días	jue 06/06/24	mié 12/06/24
11	T2.3.2 Implementación del componente de visualización de imágenes.	5 días	lun 10/06/24	dom 16/06/24
12	T2.3.3 Implementación de la herramienta de dibujo de anotaciones (rectángulos).	7 días	dom 16/06/24	mar 25/06/24
13	T2.3.4 Implementación del almacenamiento de anotaciones en local.	4 días	lun 10/06/24	jue 13/06/24
14	**H2.4 Integración de funcionalidades básicas**	**14 días**	**mié 26/06/24**	**lun 15/07/24**
15	T2.4.1 Implementación de la importación de imágenes.	3 días	mié 26/06/24	vie 28/06/24
16	T2.4.2 Implementación de la exportación de anotaciones en formato YOLO.	5 días	sáb 29/06/24	vie 05/07/24
17	T2.4.3 Implementación del control de visibilidad de anotaciones.	2 días	sáb 06/07/24	mar 09/07/24
18	T2.4.4 Implementación de la navegación por miniaturas.	4 días	mié 10/07/24	lun 15/07/24
19	**H2.5 Integración con el backend**	**12 días**	**mar 16/07/24**	**mié 31/07/24**
20	T2.5.1 Implementación de la subida de imágenes al servidor (Flask).	6 días	mar 16/07/24	mar 23/07/24
21	T2.5.2 Implementación de la persistencia de anotaciones en el backend.	3 días	mié 24/07/24	vie 26/07/24
22	T2.5.3 Pruebas de integración entre frontend y backend.	3 días	sáb 27/07/24	mié 31/07/24
23	**H2.6 Memoria**	**29 días**	**jue 01/08/24**	**mar 10/09/24**
24	T2.6.1 Redacción del capítulo de Análisis.	14 días	jue 01/08/24	mar 20/08/24
25	T2.6.2 Redacción del capítulo de Diseño.	15 días	mié 21/08/24	mar 10/09/24
26	T2.6.3 Actualización del capítulo de Introducción.	8 días	jue 01/08/24	lun 12/08/24
27	T2.6.4 Actualización del capítulo de Planificación.	21 días	mar 13/08/24	mar 10/09/24
28	**H2.7 Presentación**	**4 días**	**mié 11/09/24**	**dom 15/09/24**
29	T2.7.1 Elaboración de la presentación del segundo sprint.	3 días	mié 11/09/24	dom 15/09/24
30	**H2.8 Finalización del sprint**	**5 días**	**lun 16/09/24**	**vie 20/09/24**
31	T2.8.1 Preparación y realización de la comunicación de progresos.	5 días	lun 16/09/24	vie 20/09/24
32	T2.8.2 Reflexión y retrospectiva sobre el sprint.	5 días	lun 16/09/24	vie 20/09/24

Figura 2.4: Diagrama de Gantt Sprint #2.



	Nombre de tarea	Duración	Comienzo	Fin
1	**H3.1 Planificación del sprint**	**5 días**	**mar 01/10/24**	**sáb 05/10/24**
2	T3.1.1 Definición de los objetivos del sprint.	0 días	mar 01/10/24	mar 01/10/24
3	T3.1.2 Planificación de tareas a realizar durante el sprint.	3 días	mié 02/10/24	vie 04/10/24
4	T3.1.3 Revisión y actualización del tablero Kanban.	1 día	vie 04/10/24	sáb 05/10/24
5	T3.1.4 Definición del calendario de reuniones del sprint.	1 día	vie 04/10/24	sáb 05/10/24
6	**H3.2 Desarrollo del sprint**	**100 días**	**dom 06/10/24**	**jue 20/02/25**
7	T3.2.1 Reuniones de sincronización semanales.	99 días	dom 06/10/24	jue 20/02/25
8	T3.2.2 Actualización continua del tablero Kanban.	99 días	dom 06/10/24	jue 20/02/25
9	**H3.3 Implementación avanzada de la herramienta**	**52 días**	**dom 06/10/24**	**dom 15/12/24**
10	T3.3.1 Implementación del sistema de validación de imágenes.	16 días	dom 06/10/24	lun 28/10/24
11	T3.3.2 Implementación de la funcionalidad de edición y borrado de anotaciones.	10 días	mar 29/10/24	lun 11/11/24
12	T3.3.3 Implementación de la confirmación al salir sin guardar.	12 días	mar 12/11/24	mié 27/11/24
13	T3.3.4 Mejora de la gestión de miniaturas de imágenes.	12 días	jue 28/11/24	dom 15/12/24
14	**H3.4 Integración del sistema de inferencia**	**26 días**	**lun 16/12/24**	**lun 20/01/25**
15	T3.4.1 Preparación del entorno de inferencia.	5 días	lun 16/12/24	vie 20/12/24
16	T3.4.2 Implementación del endpoint de inferencia en Flask.	7 días	sáb 21/12/24	mar 31/12/24
17	T3.4.3 Integración del endpoint en la herramienta de anotación.	10 días	mié 01/01/25	mar 14/01/25
18	T3.4.4 Pruebas iniciales del sistema de inferencia.	4 días	mié 15/01/25	lun 20/01/25
19	**H3.5 Experiencia de usuario y mejoras de interfaz**	**16 días**	**vie 10/01/25**	**vie 31/01/25**
20	T3.5.1 Revisión de la navegación y flujo de uso.	6 días	vie 10/01/25	vie 17/01/25
21	T3.5.2 Mejora de la usabilidad en el proceso de anotación.	5 días	jue 16/01/25	mié 22/01/25
22	T3.5.3 Adaptación de la interfaz para dispositivos de diferentes tamaños.	7 días	jue 23/01/25	vie 31/01/25
23	**H3.6 Memoria**	**12 días**	**sáb 01/02/25**	**sáb 15/02/25**
24	T3.6.1 Redacción del capítulo de Implementación.	8 días	sáb 01/02/25	mié 12/02/25
25	T3.6.2 Revisión y actualización del capítulo de Diseño.	4 días	sáb 08/02/25	jue 13/02/25
26	T3.6.3 Actualización del capítulo de Introducción.	3 días	mié 12/02/25	sáb 15/02/25
27	**H3.7 Presentación**	**5 días**	**dom 16/02/25**	**jue 20/02/25**
28	T3.7.1 Elaboración de la presentación del tercer sprint.	4 días	dom 16/02/25	jue 20/02/25
29	**H3.8 Finalización del sprint**	**3 días**	**vie 21/02/25**	**mar 25/02/25**
30	T3.8.1 Preparación y realización de la comunicación de progresos.	3 días	vie 21/02/25	mar 25/02/25
31	T3.8.2 Reflexión y retrospectiva sobre el sprint.	3 días	vie 21/02/25	mar 25/02/25

Figura 2.5: Diagrama de Gantt Sprint #3.



	Nombre de tarea	Duración	Comienzo	Fin
1	**H4.1 Planificación del sprint**	**4 días**	**sáb 01/03/25**	**mié 05/03/25**
2	T4.1.1 Definición de los objetivos del sprint.	0 días	sáb 01/03/25	sáb 01/03/25
3	T4.1.2 Planificación de tareas a realizar durante el sprint.	2 días	dom 02/03/25	mar 04/03/25
4	T4.1.3 Revisión y actualización del tablero Kanban.	2 días	mar 04/03/25	mié 05/03/25
5	T4.1.4 Definición del calendario de reuniones del sprint.	2 días	mar 04/03/25	mié 05/03/25
6	**H4.2 Desarrollo del sprint**	**66 días**	**jue 06/03/25**	**jue 05/06/25**
7	T4.2.1 Reuniones de sincronización semanales.	66 días	jue 06/03/25	jue 05/06/25
8	T4.2.2 Actualización continua del tablero Kanban.	66 días	jue 06/03/25	jue 05/06/25
9	**H4.3 Finalización de funcionalidades avanzadas**	**26 días**	**jue 06/03/25**	**jue 10/04/25**
10	T4.3.1 Implementación de la exportación en formato COCO.	3 días	jue 06/03/25	lun 10/03/25
11	T4.3.2 Implementación de edición de anotaciones existentes.	11 días	mar 11/03/25	mar 25/03/25
12	T4.3.3 Implementación de borrado individual de anotaciones.	3 días	mié 26/03/25	vie 28/03/25
13	T4.3.4 Implementación de persistencia de anotaciones validadas y su visualización en miniaturas.	12 días	sáb 15/03/25	mar 01/04/25
14	T4.3.5 Implementación de confirmación al salir sin guardar.	3 días	mié 02/04/25	vie 04/04/25
15	T4.3.6 Implementación del rediseño final de la interfaz.	4 días	sáb 05/04/25	jue 10/04/25
16	**H4.4 Pruebas de validación de la herramienta**	**14 días**	**vie 11/04/25**	**mié 30/04/25**
17	T4.4.1 Definición del plan de pruebas.	10 días	vie 11/04/25	jue 24/04/25
18	T4.4.2 Ejecución de pruebas funcionales.	13 días	lun 14/04/25	mié 30/04/25
19	T4.4.3 Ejecución de pruebas de usabilidad.	4 días	vie 25/04/25	mié 30/04/25
20	T4.4.4 Registro de incidencias y corrección de errores detectados.	13 días	lun 14/04/25	mié 30/04/25
21	T4.4.5 Elaboración del informe de pruebas.	3 días	lun 28/04/25	mié 30/04/25
22	**H4.5 Memoria**	**26 días**	**jue 01/05/25**	**jue 05/06/25**
23	T4.5.1 Redacción final del capítulo de Análisis.	8 días	jue 01/05/25	lun 12/05/25
24	T4.5.2 Redacción final del capítulo de Diseño.	6 días	lun 12/05/25	lun 19/05/25
25	T4.5.3 Redacción final del capítulo de Implementación.	4 días	mar 20/05/25	dom 25/05/25
26	T4.5.4 Redacción final del capítulo de Validación.	5 días	lun 26/05/25	dom 01/06/25
27	T4.5.5 Redacción del capítulo de Balance temporal y económico.	4 días	sáb 31/05/25	jue 05/06/25
28	T4.5.6 Revisión completa y corrección de estilo de la memoria.	4 días	dom 01/06/25	jue 05/06/25
29	**H4.6 Presentación**	**3 días**	**vie 06/06/25**	**mar 10/06/25**
30	T4.6.1 Elaboración de la presentación final del TFG.	2 días	vie 06/06/25	lun 09/06/25
31	T4.6.2 Preparación de la defensa.	3 días	vie 06/06/25	mar 10/06/25
32	T4.6.3 Ensayo de la defensa con feedback de tutores.	1 día	mar 10/06/25	mar 10/06/25
33	**H4.7 Finalización del sprint**	**4 días**	**mié 11/06/25**	**dom 15/06/25**
34	T4.7.1 Preparación y realización de la comunicación final de progresos.	3 días	mié 11/06/25	vie 13/06/25
35	T4.7.2 Reflexión y retrospectiva sobre el sprint y el proyecto completo.	3 días	mié 11/06/25	dom 15/06/25

Figura 2.6: Diagrama de Gantt Sprint #4.

2.5.2. Balance económico

El balance económico recoge los costes reales del proyecto, que resultan de la suma de los costes de hardware, software y recursos humanos.

Si bien se han producido pequeñas desviaciones en el tiempo dedicado al proyecto, los costes se han mantenido dentro de márgenes razonables respecto al presupuesto inicial.

Costes reales de hardware

El coste real de hardware no ha sufrido modificaciones respecto a lo planificado, ya que el proyecto se ha desarrollado dentro del período estimado y con los recursos inicialmente previstos.

Componente	Coste total	Vida útil	% uso	Uso	Coste real
Ordenador personal (i7, 16 GB RAM, SSD)	1200€	60 meses	30 %	12 meses	72,00€
Conexión a internet doméstica	30 €/mes	–	–	12 meses	360,00€
Servidor con GPU (uso puntual)	5 €/hora	–	–	15 horas	75,00€
Total hardware	**507,00€**				

Cuadro 2.10: Coste real de hardware.

Costes reales de software

Al igual que en la planificación inicial, el coste real de software es 0€, dado que todas las herramientas empleadas disponen de licencias gratuitas que han sido suficientes para el correcto desarrollo del proyecto.

Herramienta	Coste mensual (€)	% uso	Uso (meses)	Coste real (€)
Overleaf	0	100 %	12	0
Trello	0	100 %	12	0
Microsoft Teams	0	100 %	12	0
Google Colab	0	100 %	6	0
Visual Studio Code	0	100 %	12	0
Draw.io	0	100 %	12	0
Total software	**0**			

Cuadro 2.11: Coste real de software.

Costes reales de recursos humanos

El coste real de recursos humanos se ha ajustado ligeramente en función de la dedicación efectiva en cada sprint, que ha sido superior a la estimada inicialmente.

A continuación, se muestra la tabla actualizada con las horas reales dedicadas y el coste correspondiente.

Rol	Salario hora (€)	Horas reales	Coste total (€)
Gestor de proyecto	12,00	20	240,00
Analista	15,00	29	435,00
Científico de datos	18,00	63	1134,00
Programador	14,00	114	1596,00
Tester de software	13,00	17	221,00
Documentalista	11,00	43	473,00
Subtotal recursos humanos	**4099,00**		
+ Seguridad Social (30,9 %)	**1266,71**		
Total recursos humanos	**5365,71**		

Cuadro 2.12: Coste real de recursos humanos.

Coste total del proyecto

Finalmente, el coste total real del proyecto es el siguiente:

Concepto	Coste (€)
Hardware	507,00
Software	0
Recursos Humanos	5365,71
TOTAL	**5872,71**

Cuadro 2.13: Coste total real del proyecto.

El balance temporal y económico del proyecto pone de manifiesto que, a pesar de las pequeñas desviaciones y reajustes realizados durante el desarrollo, el proyecto se ha completado de manera satisfactoria, ajustándose razonablemente a la planificación inicial y con un control adecuado de los recursos disponibles.

Capítulo 3

Antecedentes

Este capítulo describe el contexto general en el que se desarrolla el proyecto, las herramientas utilizadas y los fundamentos científicos y tecnológicos que lo sustentan.

En primer lugar, se presenta el entorno de negocio (sección 3.1), incluyendo los interesados que participan o se ven beneficiados por el sistema. A continuación, se describen las herramientas y librerías empleadas para el desarrollo (sección 3.1.2). Finalmente, se expone el contexto científico-teórico (sección 3.2) y el estado del arte (sección 3.3).

3.1. Entorno de negocio

El avance de la Inteligencia Artificial y, en particular, de la Visión Artificial, ha propiciado en los últimos años la aparición de numerosas aplicaciones basadas en modelos de detección de objetos. Estas aplicaciones tienen un impacto creciente en sectores como la industria, la automoción, la seguridad, la sanidad, el medio ambiente, el retail, o incluso la investigación científica.

El entrenamiento de estos modelos requiere disponer de datasets de imágenes anotadas manualmente, es decir, con información sobre la localización y la clase de los objetos presentes en cada imagen. Este proceso de anotación es una de las fases más críticas y costosas en la creación de modelos robustos de detección de objetos.

El proceso completo de desarrollo de modelos de detección de objetos suele implicar varias fases bien diferenciadas, como se muestra en la Figura 3.1. La anotación de imágenes es una de las etapas más críticas y laboriosas, con un impacto directo en la calidad de los modelos resultantes.

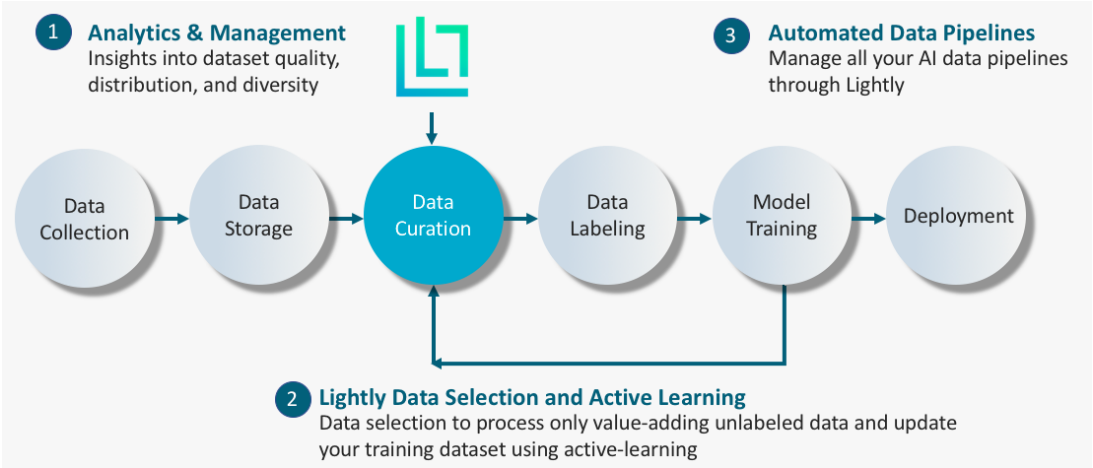


Figura 3.1: Flujo típico de un proyecto de Visión Artificial basado en detección de objetos.

Actualmente, existen herramientas genéricas de anotación, pero muchas de ellas presentan ciertas limitaciones:

- Interfaz poco intuitiva o poco adaptada a flujos de trabajo reales.
- Dificultad para gestionar proyectos completos de anotación en equipos de trabajo.
- Limitaciones en los formatos de exportación.
- Falta de integración directa con procesos de entrenamiento de modelos.

En este contexto, el presente proyecto desarrolla una aplicación web de anotación de imágenes centrada en la usabilidad, la eficiencia y la integración completa con el ciclo de vida de un proyecto de Visión Artificial. Además, permite cubrir tanto la fase de anotación manual como la de validación de resultados, integrando la inferencia de modelos de detección ya entrenados.

Con ello, se pretende facilitar el trabajo de anotadores, data scientists e investigadores en el ámbito de la detección de objetos.

3.1.1. Interesados (Stakeholders)

Un interesado o stakeholder es un individuo, grupo u organización que participa activamente en el proyecto, se ve afectado por su proceso o por sus resultados, o tiene la capacidad de influir en su desarrollo o en su impacto.
Los interesados pueden ser miembros del equipo de proyecto, de la organización que impulsa el proyecto, o ser externos a ambas.

Cada stakeholder se caracteriza por su nombre, el rol que desempeña en el proyecto, los requisitos que plantea, su grado de influencia y de interés en el mismo, su pertenencia (interno/externo), y su posición respecto al proyecto.

Interesado: Lucía Cuesta (autora del TFG)	
Rol	Desarrolladora principal
Requisitos	Diseñar e implementar la herramienta, aplicar buenas prácticas de UX, validar su funcionamiento completo.
Influencia	Alta
Interés	Éxito en el TFG. Obtener un sistema funcional y completo que demuestre sus capacidades técnicas, aplicando buenas prácticas de UX e integración de modelos de Visión Artificial.
Interno/ Externo	Interno
Posición	A favor

Cuadro 3.1: Interesado Lucía Cuesta (autora del TFG).

Interesado: Tutores del TFG	
Rol	Supervisores académicos
Requisitos	Guiar técnicamente el desarrollo, garantizar la calidad del TFG y la adecuación a los requisitos de la titulación.
Influencia	Alta
Interés	Éxito en el TFG. Que la alumna complete un proyecto de alta calidad técnica y académica, cumpliendo los plazos y objetivos establecidos y siguiendo una metodología rigurosa.
Interno/Exter	Interno
Posición	A favor

Cuadro 3.2: Interesado Tutores del TFG.

Interesado: Universidad de Valladolid (UVA)	
Rol	Entidad académica
Requisitos	Cumplimiento de la normativa de los TFG, aportación al conocimiento en el ámbito de la Visión Artificial.
Influencia	Media
Interés	Promover la investigación en el campo de la Visión Artificial. Fomentar proyectos innovadores en el ámbito del TFG y contribuir a la reputación y prestigio de la universidad en estas tecnologías emergentes.
Interno/Exter	Interno
Posición	A favor

Cuadro 3.3: Interesado Universidad de Valladolid (UVA).

Interesado: Estudiantes e investigadores	
Rol	Usuarios finales potenciales
Requisitos	Disponer de una herramienta sencilla para crear datasets anotados y acelerar el entrenamiento de modelos.
Influencia	Baja
Interés	Disponer de una herramienta accesible y versátil que facilite la creación de datasets de imágenes anotadas, acelerando sus propios proyectos de investigación o formación en detección de objetos.
Interno/Exter	Externo
Posición	A favor

Cuadro 3.4: Interesado Estudiantes e investigadores.

Interesado: Empresas de IA y Visión Artificial	
Rol	Usuarios potenciales en la industria
Requisitos	Herramienta adaptable a distintos proyectos, exportación en formatos estándar, integración con procesos de Machine Learning.
Influencia	Media
Interés	Contar con una herramienta eficiente que permita generar datasets personalizados para proyectos comerciales, con integración directa en pipelines de Machine Learning y exportación en formatos estándar.
Interno/Exter	Externo
Posición	A favor

Cuadro 3.5: Interesado Empresas de IA y Visión Artificial.

3.1.2. Herramientas y librerías

Las herramientas y librerías utilizadas en este proyecto se dividen en dos grupos: aquellas destinadas a la implementación técnica de la aplicación web y la integración con modelos de detección de objetos, y aquellas orientadas a la gestión del proyecto, comunicación y redacción de la memoria.

El uso de herramientas modernas y adaptadas a cada fase del desarrollo ha resultado clave para alcanzar un producto final robusto y usable.

Herramientas y librerías para la implementación

El sistema se ha desarrollado como una aplicación web completa compuesta por dos componentes principales:

- un frontend desarrollado en Angular
- un backend basado en Python con el framework Flask.

El uso de Angular [14] ha permitido crear una interfaz de usuario interactiva, moderna y altamente responsive, adaptada a las necesidades del proceso de anotación de imágenes. Angular proporciona además una estructura escalable que facilita la organización del código y la incorporación progresiva de nuevas funcionalidades. La flexibilidad que ofrece su sistema de componentes ha sido muy valiosa para el diseño de la interfaz de anotación personalizada.

Para el backend, se ha utilizado el framework Flask [2], que facilita el desarrollo de servicios REST y proporciona una integración sencilla con librerías de Machine Learning y Visión Artificial escritas en Python. El backend se encarga de gestionar el almacenamiento de las anotaciones, las operaciones sobre los proyectos y la integración con el proceso de inferencia automática.

El motor de inferencia utilizado se ha basado en la librería IceVision [1], una potente librería open-source para la detección de objetos que soporta múltiples frameworks de entrenamiento como PyTorch y FastAI. IceVision ha permitido integrar modelos preentrenados y realizar inferencia sobre las imágenes directamente desde la aplicación web, automatizando así parte del proceso de validación.
La combinación de IceVision con PyTorch [24] ha resultado especialmente eficaz por su rendimiento y flexibilidad.

Para el almacenamiento y manipulación de las anotaciones se ha seguido el formato YOLO [5], ampliamente utilizado en proyectos de detección de objetos, lo que facilita la exportación de datasets anotados en un formato compatible con procesos estándar de entrenamiento.

Además, se han empleado diversas librerías de soporte:

- OpenCV [27] para la manipulación básica de imágenes.
- Pandas [39] para el procesamiento estructurado de datos.
- NumPy [6] para operaciones numéricas.
- Matplotlib [4] para la generación de gráficos auxiliares.

El editor de código principal empleado ha sido Visual Studio Code [9], por su integración excelente con entornos tanto de desarrollo web (HTML/CSS/TypeScript) como de Python, y por sus extensiones para control de versiones con Git.

Herramientas para la gestión

El proyecto se ha gestionado siguiendo una metodología ágil, basada en la planificación iterativa de tareas y entregas parciales.

Para ello, se han utilizado diversas herramientas de apoyo:

- La plataforma Microsoft Teams [8] ha servido como espacio de trabajo virtual, facilitando la comunicación constante entre la autora y los tutores del TFG. A través de Teams se han compartido avances, se han resuelto dudas técnicas y se han realizado reuniones de seguimiento.
- Para la planificación de tareas y la gestión de la carga de trabajo, se ha utilizado Trello [7], una herramienta que permite gestionar las tareas en formato Kanban. Se han definido columnas para tareas pendientes, en progreso y completadas, facilitando la priorización y el seguimiento del avance del proyecto.
- El control de versiones del código se ha llevado a cabo mediante Git y la plataforma GitHub [11], lo que ha permitido mantener un historial claro de los cambios realizados, trabajar con ramas para nuevas funcionalidades y garantizar la trazabilidad completa del desarrollo.
- Finalmente, la redacción de la memoria del TFG se ha realizado utilizando Overleaf [22], un editor online de LaTeX, que ha permitido generar una documentación de alta calidad tipográfica y facilita el trabajo colaborativo en la escritura.

3.2. Contexto científico-teórico

En esta sección se presentan los fundamentos teóricos y técnicos que sustentan el desarrollo del sistema de anotación e inferencia para Visión Artificial.

Se revisan los conceptos clave en torno al aprendizaje profundo, la detección de objetos, la anotación de datasets y los flujos de trabajo típicos en proyectos de Computer Vision. Comprender este contexto es esencial para valorar las decisiones de diseño adoptadas en la aplicación web y el alcance de sus funcionalidades.

3.2.1. Visión Artificial (Computer Vision)

La **Visión Artificial**, o *Computer Vision*, es un campo de estudio de la Inteligencia Artificial que tiene como objetivo dotar a los sistemas informáticos de la capacidad de interpretar y comprender el contenido de imágenes y vídeos digitales [38]. A través de algoritmos y modelos matemáticos, la Visión Artificial busca emular la capacidad humana de percibir el entorno visual, permitiendo a los ordenadores extraer información semántica relevante a partir de datos visuales.

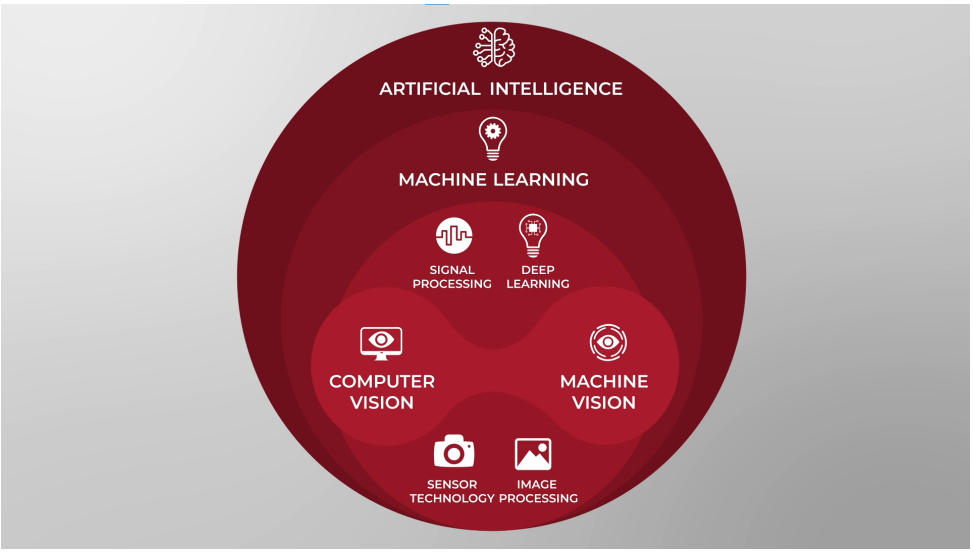


Figura 3.2: Relación entre Inteligencia Artificial, Aprendizaje Automático y Visión Artificial.

En los últimos años, el avance de las técnicas de **aprendizaje profundo** ha impulsado de manera significativa las capacidades de la Visión Artificial, haciendo posible el desarrollo de aplicaciones cada vez más sofisticadas y precisas en entornos reales [17].

Principales tareas en Computer Vision

Dentro del campo de la Visión Artificial, se pueden identificar varias tareas fundamentales [38, 13]:

- **Clasificación de imágenes**: consiste en asignar una etiqueta o clase a una imagen completa. Ejemplo: identificar si una imagen contiene un perro o un gato.
- **Detección de objetos**: localiza y clasifica múltiples objetos dentro de una imagen, devolviendo para cada uno su clase y su localización en forma de *bounding box*. Ejemplo: detectar coches, peatones y señales de tráfico en una imagen de una calle.
- **Segmentación semántica**: asigna a cada píxel de una imagen una clase, permitiendo obtener un mapa de segmentación. Ejemplo: diferenciar en una imagen los píxeles que pertenecen a carretera, acera, cielo, edificios, etc.
- **Segmentación de instancias**: similar a la segmentación semántica, pero diferenciando entre instancias individuales de un mismo objeto. Ejemplo: identificar de manera separada cada coche en una imagen con múltiples coches.
- **Reconocimiento facial**: identificar o verificar la identidad de personas a partir de imágenes faciales.
- **Reconocimiento de texto (OCR)**: detectar y extraer texto presente en imágenes.
- **Estimación de pose**: determinar la posición y orientación de un objeto o de partes del cuerpo humano.

Estas tareas pueden combinarse en sistemas más complejos, dependiendo de los requisitos de la aplicación concreta.

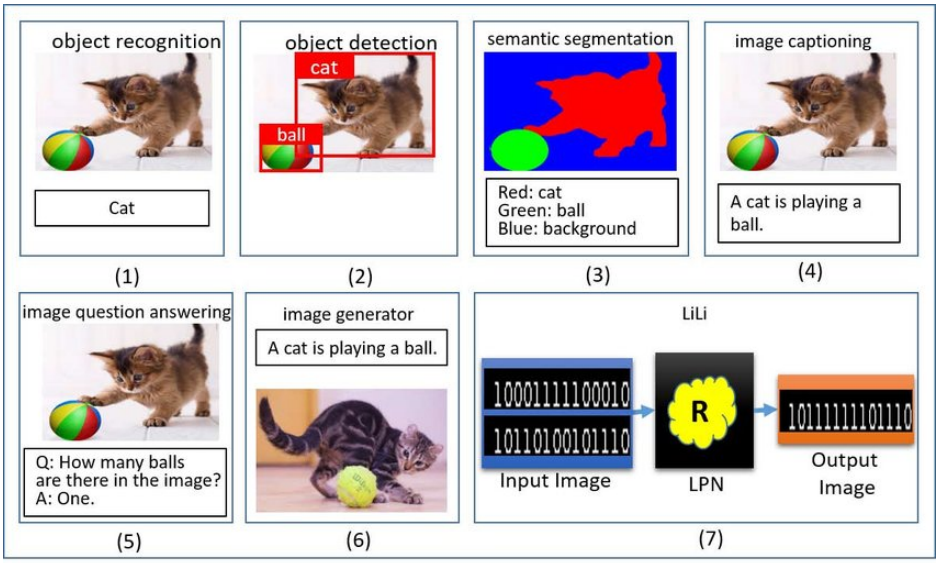


Figura 3.3: Ejemplo de tareas en Visión Artificial.

Aplicaciones de la Visión Artificial en el mundo real

La Visión Artificial es una de las áreas de la Inteligencia Artificial con un impacto más transversal en la industria y la sociedad. Algunas de las aplicaciones más destacadas incluyen [38, 17]:

- **Vehículos autónomos**: detección de peatones, señales, obstáculos y otros vehículos.
- **Medicina**: diagnóstico asistido por imágenes médicas (radiografías, resonancias, tomografías).
- **Seguridad**: sistemas de videovigilancia inteligentes, control de accesos, reconocimiento facial.
- **Retail**: análisis del comportamiento de los clientes en tienda, gestión de inventarios mediante visión.
- **Agricultura**: monitorización de cultivos, detección de plagas y enfermedades.
- **Producción**: control de calidad automatizado en líneas de producción.
- **Robótica**: visión para la navegación y manipulación de robots en entornos dinámicos.

En todos estos casos, el proceso de anotación de imágenes constituye una etapa clave para la creación de modelos de Visión Artificial de alta precisión.

3.2.2. Detección de objetos

La **detección de objetos** (*Object Detection*) es una de las tareas fundamentales dentro del campo de la Visión Artificial. Su objetivo es localizar e identificar uno o varios objetos de interés dentro de una imagen o vídeo, proporcionando como salida la clase de cada objeto detectado y su localización en forma de *bounding box* (rectángulo delimitador) [38, 42].

A diferencia de otras tareas de Computer Vision:

- La **clasificación de imágenes** asigna una única etiqueta a una imagen completa, sin proporcionar información sobre la localización de los objetos.
- La **segmentación semántica** proporciona una clasificación por píxeles, pero sin separar instancias individuales de objetos.
- La **detección de objetos** combina clasificación y localización, permitiendo identificar y ubicar múltiples objetos en una misma imagen.

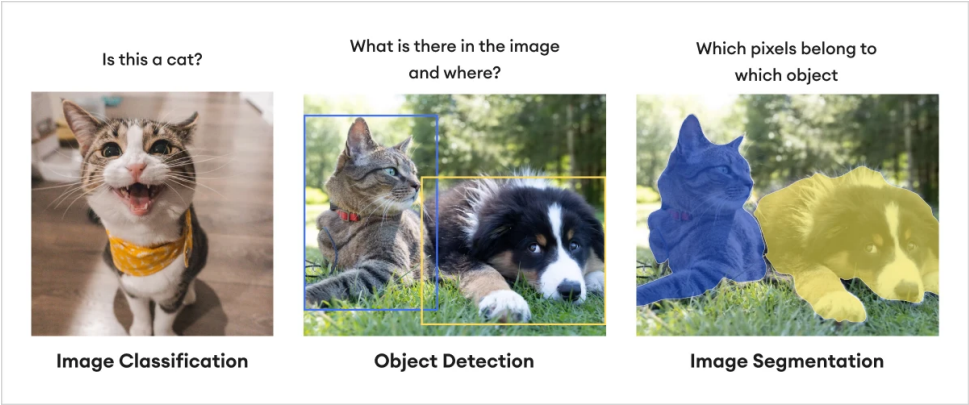


Figura 3.4: Diferencias entre: clasificación, detección de objetos y segmentación.

Componentes típicos de un detector de objetos

Los sistemas modernos de detección de objetos suelen estar compuestos por los siguientes elementos [13, 38]:

- **Modelo base o backbone**: red neuronal convolucional (CNN) encargada de extraer representaciones de alto nivel (features) de la imagen.
- **Cabeza de detección**: componente encargado de predecir:
  - Las coordenadas de las *bounding boxes* (posición y tamaño de cada objeto).
  - La clase o categoría de cada objeto detectado.
- **Procesamiento post-detección**: técnicas como *Non-Maximum Suppression* (NMS) para eliminar detecciones redundantes y quedarse con la mejor predicción para cada objeto.

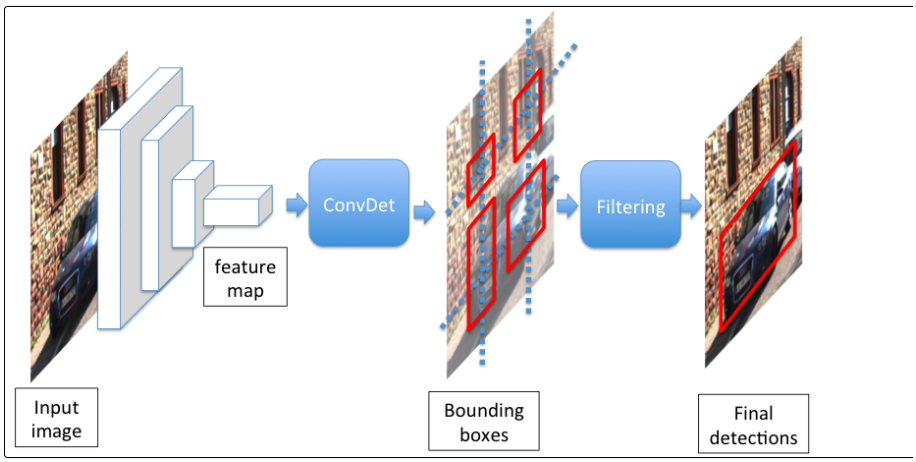


Figura 3.5: Pipeline típico de un sistema de detección de objetos.

Principales arquitecturas de detección de objetos

A lo largo de los últimos años, se han desarrollado numerosas arquitecturas de detección de objetos, que se pueden clasificar en dos grandes familias [42, 38]:

- **Two-stage detectors** (detectores en dos etapas):
  - Primero generan un conjunto de regiones candidatas (*region proposals*).
  - Después clasifican estas regiones y refinan las *bounding boxes*.
  - Ejemplo más representativo: **Faster R-CNN** [32].
- **One-stage detectors** (detectores en una etapa):
  - Realizan la detección y clasificación en una única pasada por la red.
  - Son más rápidos y adecuados para aplicaciones en tiempo real.
  - Ejemplos destacados: **YOLO** [31, 3], **SSD** [21].

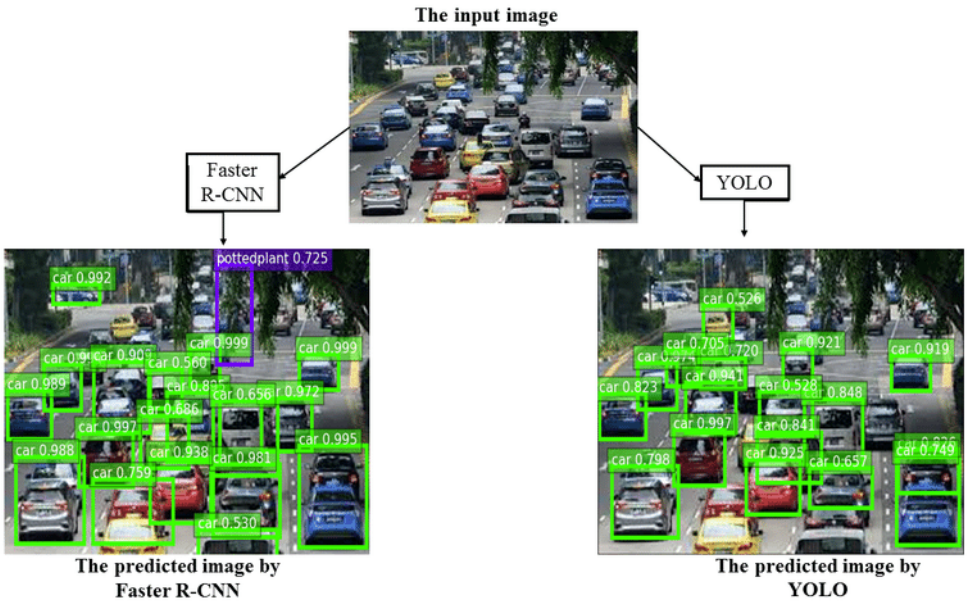


Figura 3.6: Comparativa entre detectores en una etapa (YOLO) y en dos etapas (Faster R-CNN).

Métricas habituales en detección de objetos

La evaluación de los sistemas de detección de objetos se basa en varias métricas estándar [42, 38]:

- **Intersection over Union (IoU)**:
  - Mide la superposición entre la *bounding box* predicha y la de la verdad de terreno (*ground truth*).
  - Se calcula como el área de la intersección dividida por el área de la unión de ambas cajas.
- **Precision** y **Recall**:
  - **Precision**: porcentaje de detecciones correctas entre todas las detecciones realizadas.
  - **Recall**: porcentaje de objetos reales correctamente detectados.
- **Mean Average Precision (mAP)**:
  - Métrica de referencia que resume la precisión global del detector.
  - Se calcula promediando la *Average Precision* (AP) para cada clase del dataset.
  - Se suele reportar **mAP@IoU=0.5** o **mAP@[0.5:0.95]** (cálculo en múltiples umbrales de IoU).

Estas métricas permiten comparar de manera objetiva el rendimiento de distintos modelos y configuraciones de detección de objetos.

3.2.3. Anotación de imágenes para la detección de objetos

El desarrollo de modelos precisos de **detección de objetos** requiere la disponibilidad de grandes conjuntos de datos (*datasets*) de imágenes cuidadosamente anotadas. La anotación de imágenes consiste en proporcionar, para cada imagen, la información necesaria para que el modelo pueda aprender a localizar y clasificar los objetos presentes en ellas [42, 10].

Este proceso constituye una fase esencial en el ciclo de vida de cualquier sistema de Visión Artificial, ya que la calidad de las anotaciones impacta de manera directa en el rendimiento final del modelo.

Qué es un dataset anotado

Un **dataset anotado** para la detección de objetos está formado por:

- Un conjunto de imágenes.
- Un conjunto de *anotaciones* asociadas a cada imagen, que describen los objetos presentes en ella.

Cada anotación típicamente incluye:

- Las coordenadas de la **bounding box** que delimita la región donde se encuentra el objeto.
- La **clase** o categoría a la que pertenece el objeto.

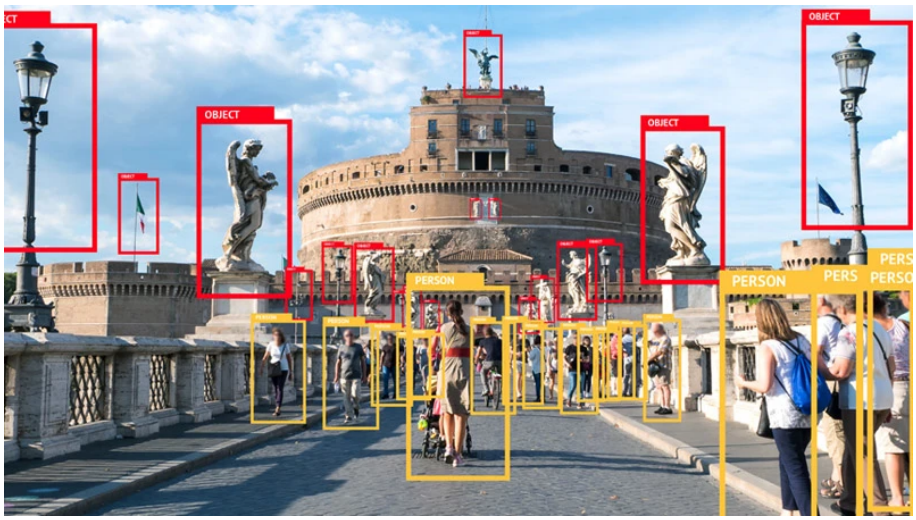


Figura 3.7: Ejemplo de anotación de imagen para la detección de objetos.

Estructura básica de las anotaciones en detección de objetos

La estructura básica de las anotaciones en un dataset de detección de objetos se puede representar, para cada imagen, como un conjunto de registros con la siguiente información [42]:

- **Clase del objeto** (por ejemplo: persona, coche, perro, botella, etc.).
- **Bounding box**, definida por sus coordenadas:
  - En formato absoluto: $(x_{\min}, y_{\min}, x_{\max}, y_{\max})$.
  - O en formato normalizado: coordenadas relativas al tamaño de la imagen.

Formatos de anotación más comunes

Existen varios formatos estándar de anotación en la comunidad de Computer Vision. Los más utilizados son:

- **YOLO** [31]:
  - Cada anotación se representa en un fichero de texto plano asociado a cada imagen.
  - Cada línea contiene: *class_id, center_x, center_y, width, height* (todos los valores normalizados entre 0 y 1).
- **Pascal VOC** [10]:
  - Anotaciones en formato XML.
  - Incluye: clase del objeto, coordenadas de la bounding box en valores absolutos, información adicional (dificultad, truncamiento...).
- **COCO** [19]:
  - Anotaciones en formato JSON.
  - Soporta bounding boxes, segmentaciones y keypoints.
  - Amplia información sobre cada imagen y cada anotación.

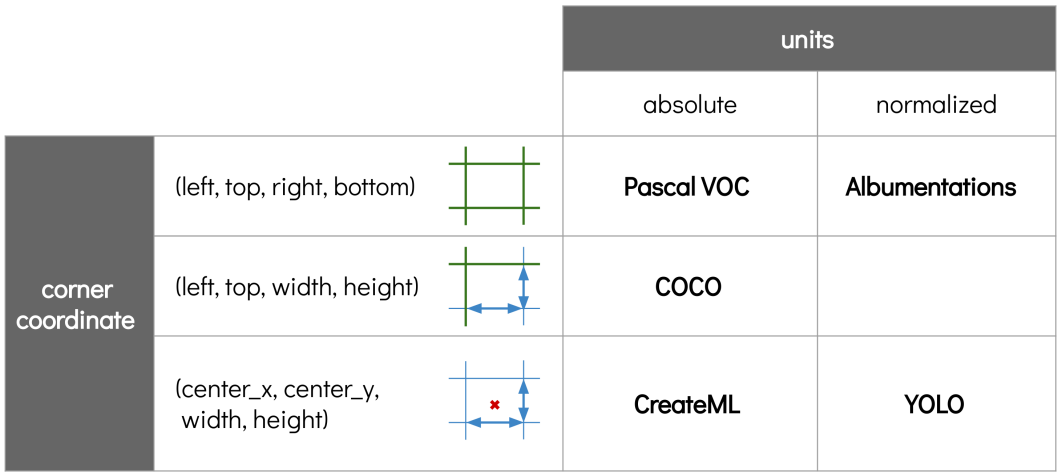


Figura 3.8: Comparativa de coordenadas en anotación de objetos (Pascal VOC, COCO y YOLO.

Problemas habituales en la anotación manual

El proceso de anotación manual de imágenes presenta varios retos y problemas habituales [34]:

- **Coste elevado**: la anotación manual es un proceso lento y costoso en términos de tiempo y recursos humanos.
- **Variabilidad entre anotadores**: distintos anotadores pueden interpretar de manera diferente los límites de los objetos o sus clases.
- **Errores e inconsistencias**: es frecuente encontrar errores de etiquetado, bounding boxes mal posicionadas o clases incorrectas.
- **Fatiga**: la tarea repetitiva de anotar grandes volúmenes de imágenes puede llevar a una disminución de la calidad de las anotaciones a lo largo del tiempo.

Importancia de la validación de las anotaciones

Dada la importancia crítica de la calidad de los datos de entrenamiento, es fundamental implementar procesos de **validación de las anotaciones** [34]:

- Revisión manual de las anotaciones por parte de expertos.
- Uso de herramientas que permitan visualizar y auditar el dataset anotado.
- Aplicación de validaciones automáticas para detectar inconsistencias o errores frecuentes.
- Integración de modelos de inferencia para preanotar y acelerar el proceso, permitiendo que el anotador humano solo valide y corrija.

En este contexto, la herramienta desarrollada en este proyecto facilita no solo la anotación manual eficiente, sino también la validación de las anotaciones mediante la integración de modelos de inferencia y un flujo de trabajo adecuado para minimizar errores y maximizar la calidad del dataset final.

3.2.4. Aprendizaje profundo aplicado a la detección de objetos

El **Aprendizaje Profundo** (*Deep Learning*) ha supuesto una revolución en el campo de la Visión Artificial en la última década, convirtiéndose en la técnica dominante para tareas como la detección de objetos [17]. Los avances en algoritmos, disponibilidad de grandes datasets y el aumento de la capacidad de cómputo (particularmente gracias a las GPU) han permitido entrenar modelos que superan ampliamente a las técnicas clásicas basadas en ingeniería manual de características.

Redes neuronales y Deep Learning

Una **red neuronal artificial** es un modelo inspirado en el funcionamiento del cerebro humano, compuesto por un conjunto de nodos o *neuronas artificiales* organizadas en capas [13]. Cada neurona realiza una operación matemática sencilla, y su combinación permite modelar funciones complejas.

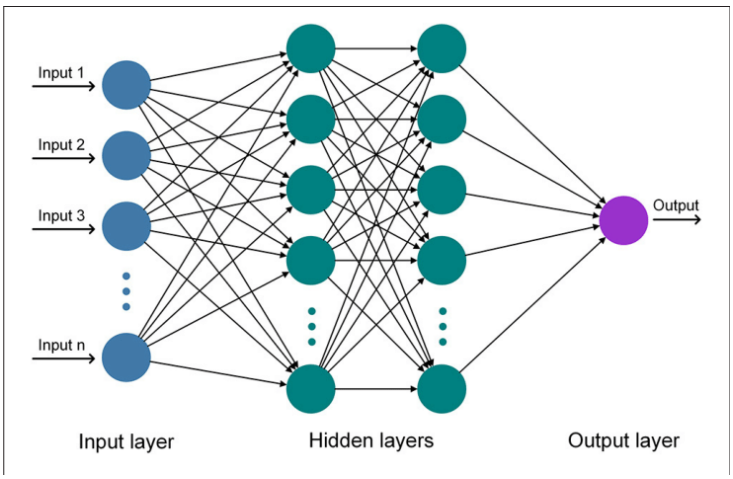


Figura 3.9: Esquema básico de una red neuronal artificial.

El **Deep Learning** se basa en redes neuronales profundas, es decir, redes con múltiples capas ocultas que permiten aprender representaciones jerárquicas de los datos [17]. En el contexto de la Visión Artificial, las arquitecturas más habituales son las **redes convolucionales** (*Convolutional Neural Networks, CNN*), que han demostrado ser especialmente eficaces para el procesamiento de datos visuales.

Ventajas del Deep Learning frente a técnicas clásicas

Antes de la adopción masiva del Deep Learning, las tareas de Computer Vision se abordaban mediante técnicas basadas en la extracción manual de características (*feature engineering*), como SIFT, HOG o SURF [38]. Estas técnicas presentaban limitaciones importantes:

- Requerían un diseño experto y específico para cada tipo de problema.
- Tenían dificultades para generalizar a condiciones de imagen muy variadas (iluminación, escala, oclusiones).
- No permitían un aprendizaje automático de la representación de los datos.

El Deep Learning aporta varias ventajas clave:

- Capacidad para aprender automáticamente las representaciones de alto nivel a partir de datos crudos (imágenes).
- Robustez frente a variaciones en los datos.
- Rendimiento superior en tareas complejas como la detección de objetos o la segmentación.
- Flexibilidad: las mismas arquitecturas pueden adaptarse a múltiples tareas de visión.

Entrenamiento de modelos de detección

El entrenamiento de un modelo de detección de objetos implica varios componentes clave [13]:

- **Dataset anotado**: conjunto de imágenes con anotaciones de bounding boxes y clases. La calidad y diversidad del dataset son cruciales para el rendimiento del modelo.
- **Función de pérdida**: mide la discrepancia entre las predicciones del modelo y las anotaciones reales. En detección de objetos se utilizan funciones de pérdida combinadas que consideran tanto la precisión de las bounding boxes como la clasificación de las instancias.
- **Optimización**: se emplean algoritmos como Adam o SGD para minimizar la función de pérdida mediante retropropagación.
- **Inferencia**: una vez entrenado, el modelo se utiliza para predecir bounding boxes y clases en nuevas imágenes. Se suelen aplicar técnicas como *Non-Maximum Suppression* (NMS) para eliminar detecciones redundantes.

El uso de **redes neuronales profundas** ha llevado la detección de objetos a niveles de precisión y robustez que no eran posibles con enfoques clásicos, posibilitando aplicaciones en sectores tan diversos como la automoción, la medicina, la industria o la robótica.

3.2.5. Procesos de inferencia y validación

En un sistema de detección de objetos, el proceso de **inferencia automática** hace referencia a la fase en la que un modelo previamente entrenado se utiliza para realizar predicciones sobre nuevas imágenes [13]. Durante esta fase, el modelo procesa cada imagen de entrada y devuelve un conjunto de detecciones: para cada objeto identificado, se predice su clase y su localización en forma de *bounding box*, acompañadas de un nivel de confianza (*confidence score*).

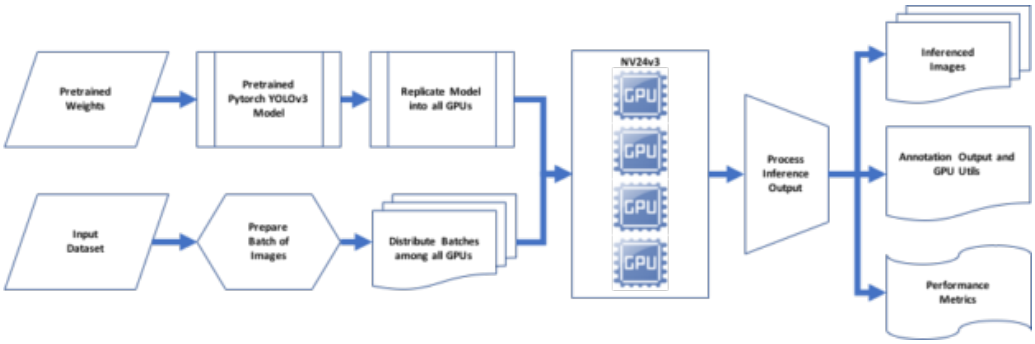


Figura 3.10: Ejemplo simplificado de proceso de inferencia en detección de objetos.

Integración de la inferencia en el flujo de trabajo de anotación

En flujos de trabajo tradicionales, la anotación de datasets para detección de objetos es un proceso manual y muy costoso [18]. Sin embargo, cuando se dispone de modelos de detección ya entrenados (aunque no sean perfectos), es posible integrarlos en el flujo de anotación para acelerar significativamente esta fase [42].

El proceso integrado se basa en los siguientes pasos:

1. Se cargan imágenes nuevas en la herramienta de anotación.
2. El motor de inferencia automática procesa estas imágenes y genera predicciones iniciales (bounding boxes + clases).
3. Las predicciones se presentan al anotador humano en la interfaz de anotación.
4. El anotador revisa las predicciones:
   - acepta aquellas que sean correctas,
   - modifica aquellas que sean imprecisas,
   - elimina aquellas que sean erróneas,
   - añade manualmente anotaciones que el modelo no haya detectado.
5. El resultado final se guarda como anotación validada.

Este flujo de trabajo híbrido, conocido como **human-in-the-loop** [15], permite combinar la velocidad del modelo con la precisión del juicio humano, obteniendo datasets anotados de alta calidad de manera mucho más eficiente.

Utilidad de la inferencia asistida

El uso de inferencia asistida aporta múltiples beneficios en el proceso de creación de datasets para detección de objetos:

- **Aceleración del proceso de anotación**: al partir de predicciones automáticas, se reduce drásticamente el número de anotaciones que deben realizarse manualmente desde cero [35].
- **Mejora de la consistencia**: las predicciones del modelo introducen un cierto grado de uniformidad en las anotaciones, que los anotadores humanos refinan.
- **Reducción del coste económico**: permite anotar más imágenes en menos tiempo, reduciendo el coste asociado a la generación de datasets grandes.
- **Entrenamiento iterativo**: posibilita ciclos de mejora incremental: tras una primera ronda de anotación asistida, el modelo puede reentrenarse con los nuevos datos validados, mejorando progresivamente su rendimiento en futuras inferencias.

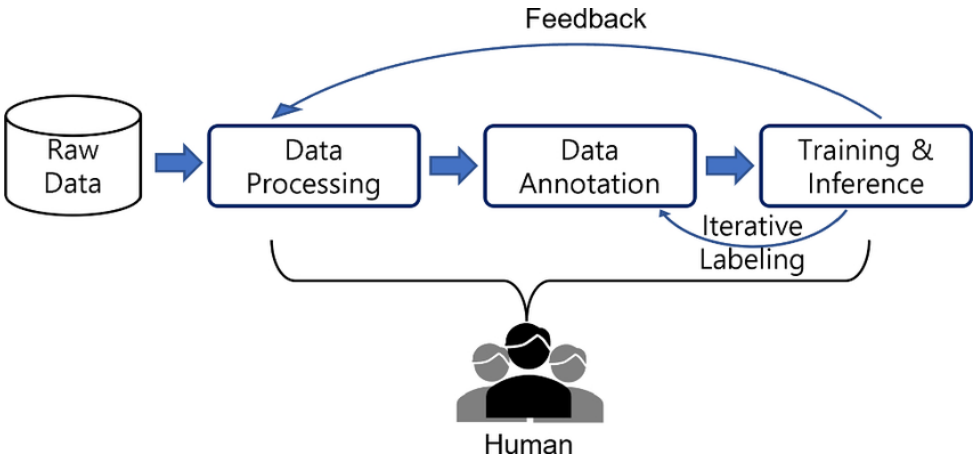


Figura 3.11: Ejemplo de flujo de trabajo Human-in-the-loop para anotación asistida.

Este enfoque es precisamente el que se ha integrado en la herramienta desarrollada en este proyecto, permitiendo al usuario validar de manera eficiente las inferencias automáticas, mejorar la calidad de los datasets y optimizar el tiempo dedicado a la anotación manual.

3.2.6. Ciclo de vida de un proyecto de Visión Artificial

El desarrollo de un sistema basado en Visión Artificial sigue habitualmente un ciclo de vida estructurado en varias fases clave [38, 35]. Cada fase plantea distintos retos técnicos y organizativos, y la calidad del resultado final depende en gran medida del éxito en todas ellas.

Fases típicas del ciclo de vida

A continuación, se describen las principales fases que componen este ciclo:

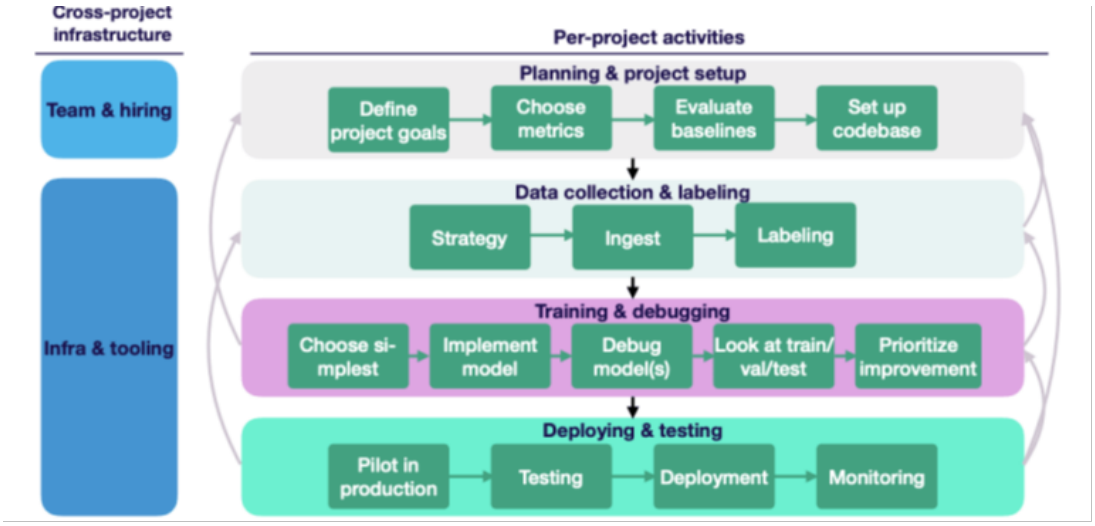


Figura 3.12: Ciclo de vida típico en un proyecto de Visión Artificial [38].

1. **Definición del problema**: consiste en establecer claramente los objetivos del sistema de Visión Artificial. Incluye definir qué objetos o patrones se quieren detectar, con qué precisión, en qué tipo de imágenes, y bajo qué restricciones de coste y tiempo de procesamiento.

2. **Creación del dataset**: es una de las fases más costosas y críticas. Consiste en recopilar imágenes representativas del problema y anotarlas adecuadamente (bounding boxes, clases, segmentación, etc.). Una anotación de calidad es clave para obtener modelos robustos [35].

3. **Entrenamiento de modelos**: a partir del dataset anotado, se entrenan redes neuronales profundas para aprender a resolver la tarea de detección. Esta fase incluye el ajuste de hiperparámetros, técnicas de regularización, y validación cruzada para evitar sobreajuste.

4. **Validación y evaluación**: se evalúa el rendimiento del modelo en un conjunto de datos independiente, mediante métricas como mAP, IOU o precisión/recall. Esta fase permite iterar sobre el dataset y el modelo para mejorar los resultados.

5. **Despliegue**: finalmente, el modelo se integra en un sistema operativo (por ejemplo en un robot, una app móvil, o un sistema industrial) y se monitoriza su rendimiento en condiciones reales.

Encaje de la herramienta desarrollada en el ciclo de vida

La herramienta desarrollada en este TFG se sitúa de manera estratégica en la segunda fase del ciclo: la **creación del dataset**.

Su propósito es facilitar el proceso de anotación y validación de imágenes para tareas de detección de objetos, proporcionando:

- Un entorno visual interactivo para dibujar y etiquetar bounding boxes.
- La posibilidad de revisar y validar las anotaciones mediante inferencia asistida.
- Funciones de exportación en formatos estándar, para una integración sencilla con procesos de entrenamiento posteriores.

Además, gracias a su flexibilidad y soporte de flujos iterativos, la herramienta permite acompañar también la fase de **validación**, permitiendo revisar anotaciones en función del rendimiento del modelo, y refinarlas para mejorar los resultados.

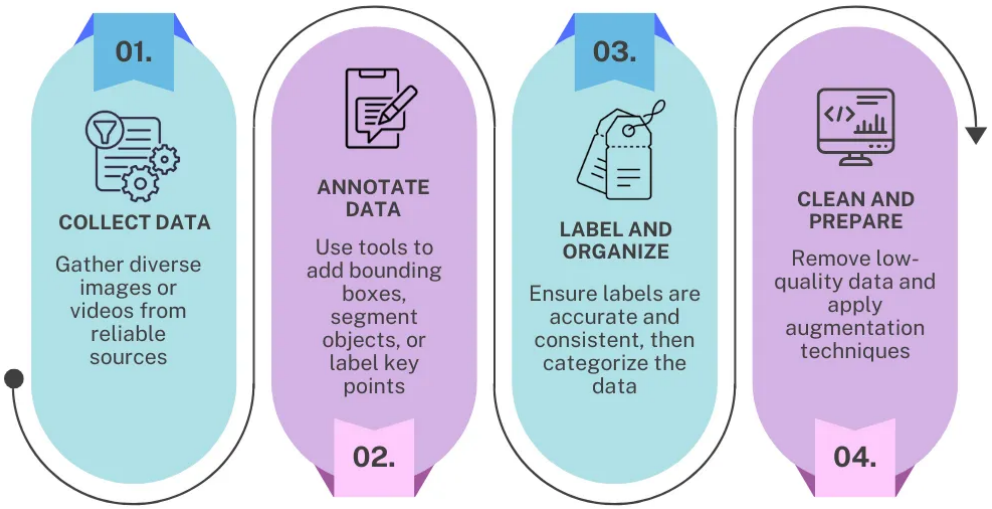


Figura 3.13: Proceso de anotación y preparación de datos en proyectos de Visión Artificial.

Importancia de la integración con formatos estándar

El uso de formatos estándar de anotación como YOLO, COCO o Pascal VOC es esencial para garantizar la interoperabilidad entre herramientas y facilitar la transición entre fases del ciclo [18, 35].

Por ello, la herramienta desarrollada permite exportar datasets anotados en el formato YOLO, ampliamente soportado por frameworks de entrenamiento como PyTorch, TensorFlow o Darknet.

Esta integración evita la necesidad de procesos de conversión manuales, minimiza la posibilidad de errores y acelera el ciclo iterativo entre anotación, entrenamiento y validación.

Conclusión: disponer de una herramienta de anotación eficiente y flexible contribuye de manera decisiva a la calidad y agilidad de todo el ciclo de vida de un proyecto de Visión Artificial, y es un factor diferenciador en la competitividad de estos sistemas en entornos reales.

3.3. Estado del arte

El proceso de agregar información a una imagen para describir su contenido se conoce como **anotación de imágenes** [15]. Esta información puede incluir etiquetas de clase, cuadros delimitadores (*bounding boxes*), puntos clave, máscaras y más. La anotación de imágenes es un paso crucial en el desarrollo de muchos sistemas de Visión Artificial, como el reconocimiento de objetos, la segmentación de imágenes y la detección de rostros.

Para realizar la anotación de imágenes, se requiere una herramienta que proporcione las funcionalidades necesarias, como la creación de etiquetas, la definición de categorías y la posibilidad de dibujar sobre las imágenes. Existen diversas herramientas disponibles, tanto online como offline, cada una con sus propias características y ventajas [35].

En este estudio, se han seleccionado cinco herramientas que se consideran especialmente útiles para el desarrollo de una herramienta de anotación de imágenes:

- **Label Studio**: plataforma de código abierto para el etiquetado de datos flexible y escalable. Admite una variedad de tipos de anotaciones, como cuadros delimitadores, puntos clave, segmentación de imágenes y más.
- **CVAT (Computer Vision Annotation Tool)**: herramienta gratuita y de código abierto para la anotación de imágenes y vídeos. Ofrece una interfaz intuitiva y admite diferentes tipos de anotaciones.

- **VGG Image Annotator**: herramienta web gratuita desarrollada por la Universidad de Oxford. Permite la anotación de imágenes con cuadros delimitadores y etiquetas de clase.
- **RectLabel**: aplicación para macOS que permite la anotación de imágenes con cuadros delimitadores y etiquetas de clase. Es una opción popular para usuarios de macOS que necesitan realizar anotaciones de forma rápida y sencilla.
- **SuperAnnotate**: plataforma de etiquetado de datos de última generación que ofrece anotación de imágenes y vídeos, herramientas de colaboración y análisis de datos.

Existen otras muchas herramientas conocidas como Roboflow [33], Labelbox [16], etc., pero con las cinco analizadas se cubren ampliamente las funcionalidades más relevantes para el contexto del proyecto.

3.3.1. Label Studio [36]

Label Studio es una herramienta de etiquetado de datos de código abierto que admite múltiples proyectos, usuarios y tipos de datos en una sola plataforma. Esta herramienta permite realizar diferentes tipos de etiquetado con muchos formatos de datos.

Label Studio también está disponible como un servicio en la nube Enterprise con seguridad mejorada (SSO, RBAC, SOC2), características de gestión de equipos, descubrimiento de datos, análisis e informes, y SLA de soporte.

Características principales

1. **Interfaz de usuario personalizable:** Label Studio permite a los usuarios crear interfaces de etiquetado personalizadas utilizando una interfaz de arrastrar y soltar.
2. **Anotación multimodal:** Label Studio admite una amplia gama de tipos de anotación, incluyendo texto, imagen, video y audio.
3. **Control de calidad:** Label Studio incluye una serie de herramientas para gestionar proyectos de etiquetado, incluyendo control de versiones, colaboración y control de calidad.
4. **Integración con modelos de aprendizaje automático:** Label Studio puede integrarse con modelos de aprendizaje automático para proporcionar predicciones para las etiquetas (pre-etiquetas), o realizar un aprendizaje activo continuo.
5. **Gestión de datos y tareas para el etiquetado:** Label Studio proporciona una aplicación web JavaScript para gestionar datos y tareas para el etiquetado.

Interfaz

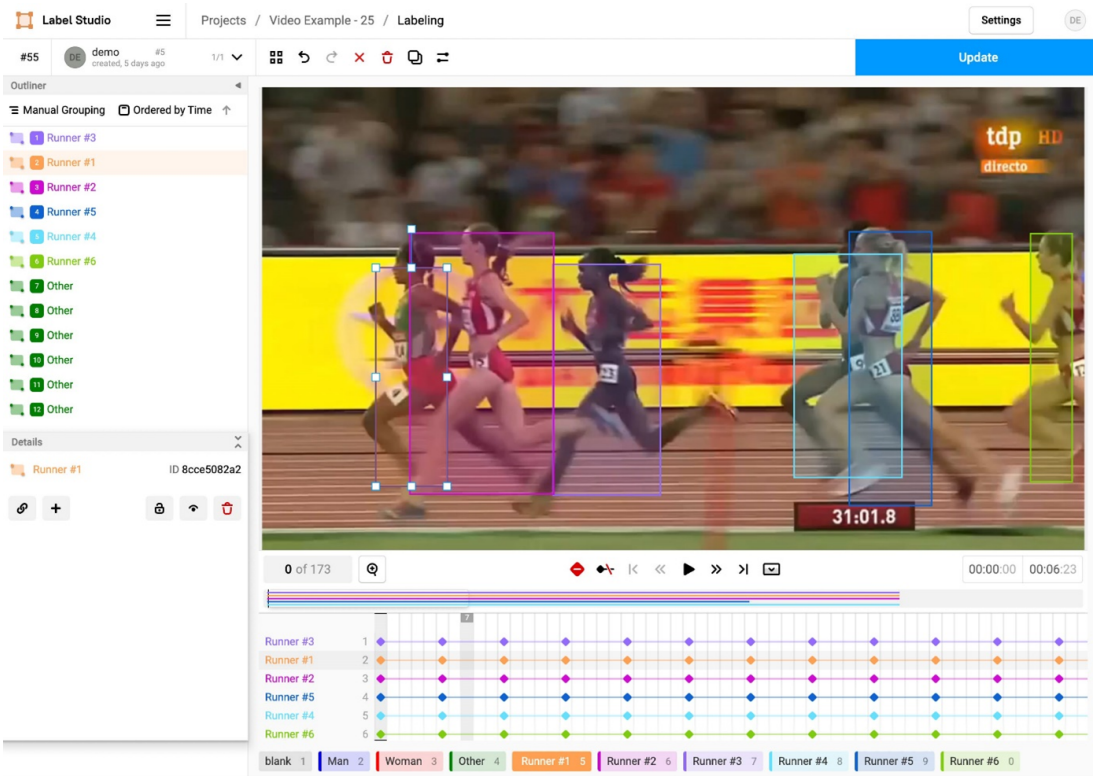


Figura 3.14: Ejemplo de interfaz de la herramienta Label Studio.

Ventajas y desventajas

Ventajas	Desventajas
Interfaz intuitiva y fácil de usar.	Curva de aprendizaje inicial para usuarios nuevos.
Amplia variedad de tareas de etiquetado.	Algunas características avanzadas requieren conocimientos técnicos.
Integración con frameworks de ML.	Requiere recursos de hardware significativos en proyectos grandes.
Automatización de tareas repetitivas.	Precio elevado para usuarios individuales.
Colaboración en tiempo real.	Limitaciones en la versión gratuita.
Comunidad activa.	Documentación extensa.

Cuadro 3.6: Ventajas y desventajas de Label Studio.

3.3.2. CVAT (Computer Vision Annotation Tool) [26]

CVAT es una herramienta gratuita y de código abierto para la anotación de imágenes y vídeos interactiva para Visión Artificial. Está escrita en Python y JavaScript, y es ampliamente utilizada en tareas de aprendizaje supervisado como detección de objetos, clasificación de imágenes, segmentación y anotación de datos 3D.

Características principales

1. **Interfaz de usuario amigable:** CVAT enfatiza la facilidad de uso, la adaptabilidad y la compatibilidad con una variedad de formatos y herramientas.
2. **Soporte de múltiples formatos:** CVAT soporta una variedad de formatos para 3D, imagen y video.
3. **Herramientas de anotación:** CVAT ofrece una amplia gama de herramientas de anotación, cada una atendiendo a diferentes aspectos de la etiquetación.
4. **Interpolación en vídeo:** Esta característica es útil para anotar videos donde los objetos se mueven de un fotograma a otro.
5. **Anotación semi-automática:** CVAT permite entrenar un modelo de aprendizaje automático para reconocer ciertos objetos, y luego usar ese modelo para ayudarte a anotar imágenes o videos en CVAT.
6. **Gestión de proyectos:** Puedes ver fácilmente todas tus tareas de anotación en un solo lugar, y puedes organizar tus tareas en proyectos.

Interfaz

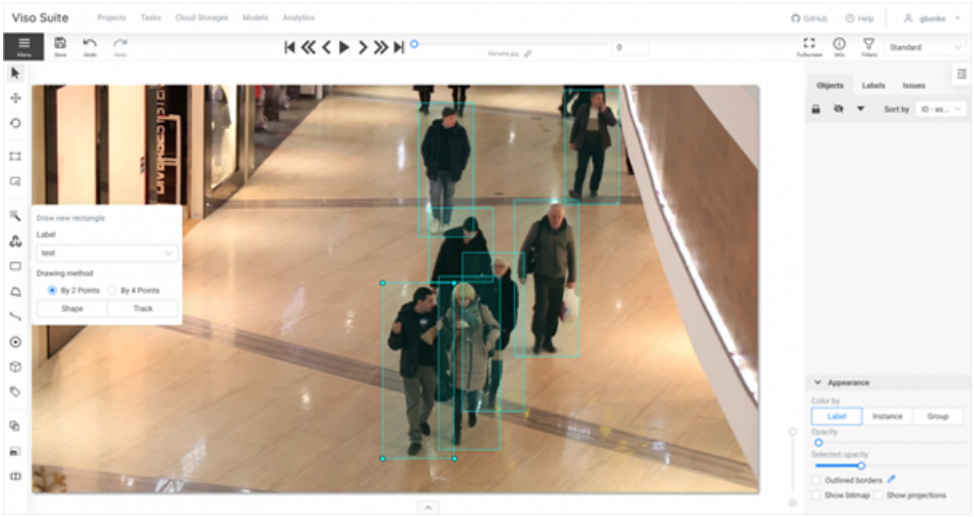


Figura 3.15: Ejemplo de interfaz de la herramienta CVAT.

Ventajas y desventajas

Ventajas	Desventajas
Amplio conjunto de herramientas de anotación.	Curva de aprendizaje inicial para usuarios nuevos.
Integración con gestión de proyectos y almacenamiento en la nube.	Requiere infraestructura de servidor para su despliegue.
Automatización de tareas repetitivas.	Requiere conocimientos técnicos para su instalación y configuración.
Soporte para múltiples formatos.	Problemas de rendimiento con grandes conjuntos de datos.
Colaboración en tiempo real.	Interfaz algo compleja para principiantes.

Cuadro 3.7: Ventajas y desventajas de CVAT.

3.3.3. VGG Image Annotator (VIA) [41]

VGG Image Annotator (VIA) es una herramienta de anotación manual simple y autónoma para imágenes, audio y video. VIA se ejecuta en un navegador web y no requiere ninguna instalación o configuración. El software VIA completo cabe en una sola página HTML autocontenida de menos de 400 kilobytes que funciona como una aplicación sin conexión en la mayoría de los navegadores web modernos.

VIA es un proyecto de código abierto basado únicamente en HTML, Javascript y CSS (sin dependencia de bibliotecas externas)

Características principales

1. **Interfaz ligera y sin instalación:** VIA es pequeño y ligero de usar y puede funcionar completamente en tu navegador web.
2. **Anotación de objetos (cuadros delimitadores y polígonos):** Puedes usar VIA para dibujar cuadros delimitadores o polígonos alrededor de objetos en tus imágenes y videos para formar un conjunto de datos para la supervisión de tus modelos de visión por computadora.
3. **Anotación de múltiples tipos de datos:** VIA permite la anotación de imágenes, audio y video.
4. **Editor de atributos:** Puedes usar el editor de atributos para definir o actualizar atributos (por ejemplo, nombre, color, etc.) de las regiones definidas por el usuario.

Interfaz

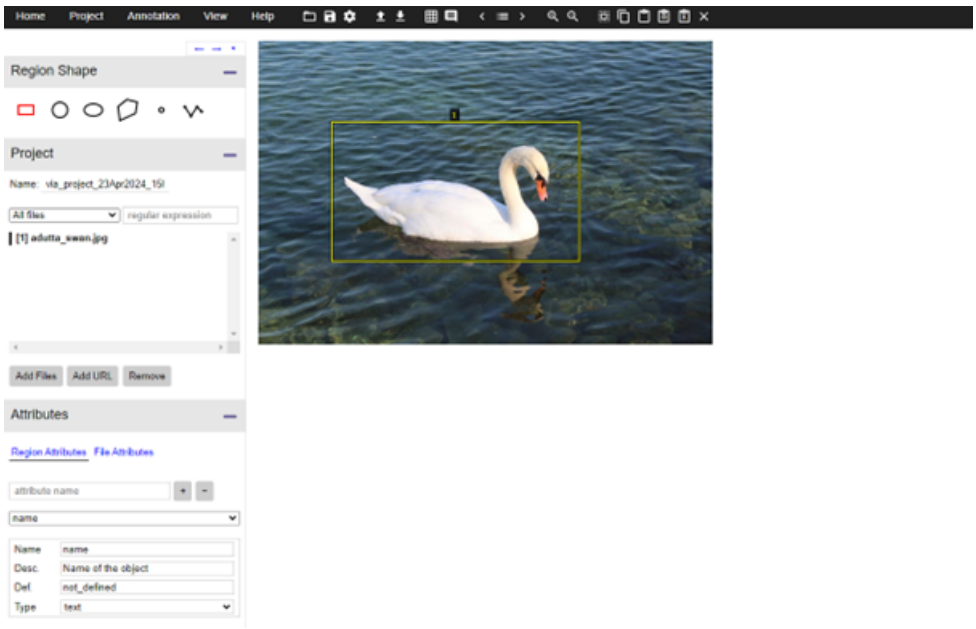


Figura 3.16: Ejemplo de interfaz de la herramienta VGG Image Annotator (VIA).

Ventajas y desventajas

Ventajas	Desventajas
Interfaz sencilla y rápida.	Limitado conjunto de herramientas de anotación.
No requiere instalación.	No ofrece segmentación semántica.
Funciona sin conexión.	Sin colaboración en tiempo real.
Personalizable mediante código fuente.	Sin integraciones externas.
Compatible con navegadores modernos.	Sin automatización de tareas.

Cuadro 3.8: Ventajas y desventajas de VGG Image Annotator (VIA).

3.3.4. RectLabel [30]

RectLabel es una herramienta de anotación de imágenes para la detección y segmentación de objetos. Es una herramienta autónoma que funciona completamente sin conexión en la mayoría de los navegadores web modernos. RectLabel es un proyecto de código abierto basado únicamente en HTML, Javascript y CSS.

Características principales

1. **Etiquetado de polígonos y píxeles:** RectLabel permite etiquetar polígonos y píxeles utilizando modelos de Segment Anything.
2. **Etiquetado automático:** RectLabel puede realizar etiquetado automático utilizando modelos de Core ML.
3. **Reconocimiento automático de texto:** RectLabel tiene la capacidad de reconocer texto automáticamente para líneas y palabras.
4. **Anotación con curvas de Bézier:** Las curvas de Bézier son útiles para anotar objetos que tienen bordes curvos.
5. **Soporte para anotación en imágenes aéreas:** RectLabel es capaz de anotar objetos en imágenes aéreas donde los objetos pueden estar orientados en cualquier dirección.
6. **Anotación de puntos clave con un esqueleto:** Esta característica es útil para anotar la estructura interna de un objeto, como la pose de una persona.

Interfaz

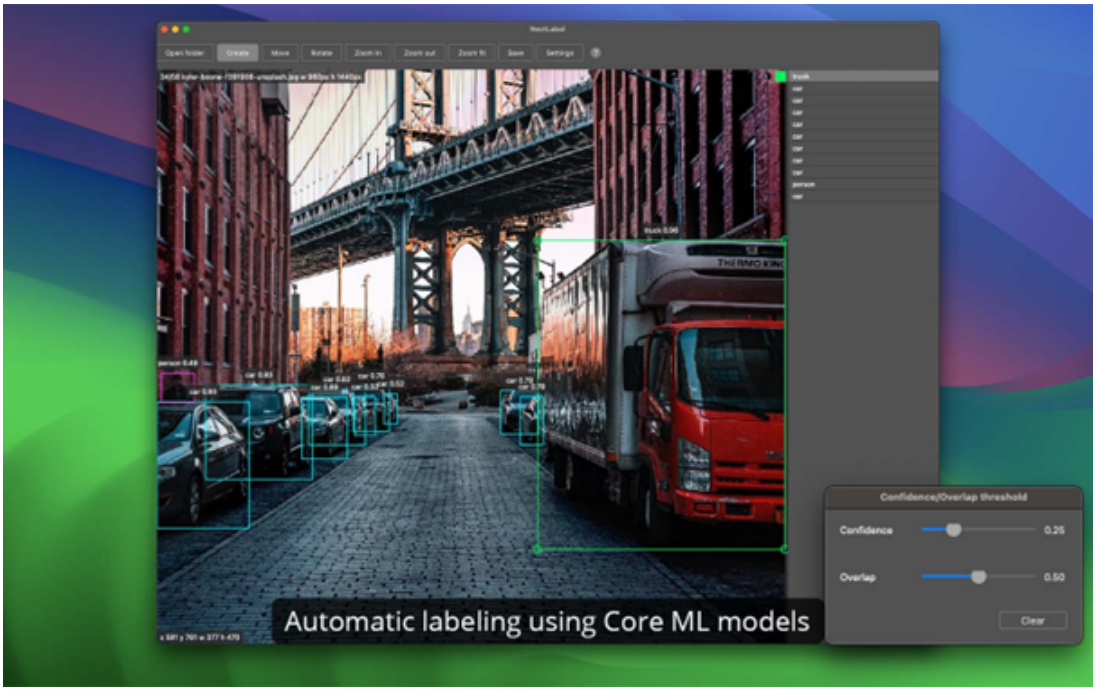


Figura 3.17: Ejemplo de interfaz de la herramienta RectLabel.

Ventajas y desventajas

Ventajas	Desventajas
Interfaz intuitiva y fácil de usar.	Solo disponible para usuarios de macOS.
Ideal para anotación de objetos.	No ofrece segmentación semántica.
Versión gratuita con funciones básicas.	Limitaciones en la versión gratuita.
Integración con TensorFlow y PyTorch.	Limitado conjunto de herramientas de anotación.
Exportación en diversos formatos.	Sin automatización de tareas.
Anotación rápida con atajos de teclado.	Sin colaboración en tiempo real.

Cuadro 3.9: Ventajas y desventajas de RectLabel.

3.3.5. SuperAnnotate [37]

SuperAnnotate es una plataforma líder para la construcción, ajuste, iteración y gestión de modelos de IA de manera rápida con los datos de entrenamiento de la más alta calidad.

Con capacidades avanzadas de anotación, herramientas de curación de datos y control de calidad, funciones de automatización, integraciones nativas y gobernanza de datos, SuperAnnotate proporciona a las empresas los recursos para construir conjuntos de datos y pipelines de ML exitosas.

Características principales

1. **Herramientas de anotación de alta precisión:** SuperAnnotate ofrece herramientas de anotación diseñadas con precisión para reflejar la intención del anotador, abordando tareas específicas con precisión excepcional.

2. **IA asistida en el etiquetado:** Incorpora algoritmos de IA que agilizan el proceso de etiquetado al sugerir etiquetas que capturan la esencia de los datos visuales, haciendo que la anotación sea más intuitiva y enriquecedora.

3. **Colaboración armoniosa:** Fomenta un entorno de colaboración donde las contribuciones individuales se entrelazan de manera fluida, generando un resultado colectivo unificado y coherente que demuestra el poder del trabajo en equipo.

4. **Integración con modelos de ML:** Se integra sin problemas con modelos de aprendizaje automático, refinando los datos anotados para satisfacer las necesidades de los sistemas de IA de manera precisa y eficiente.

Interfaz

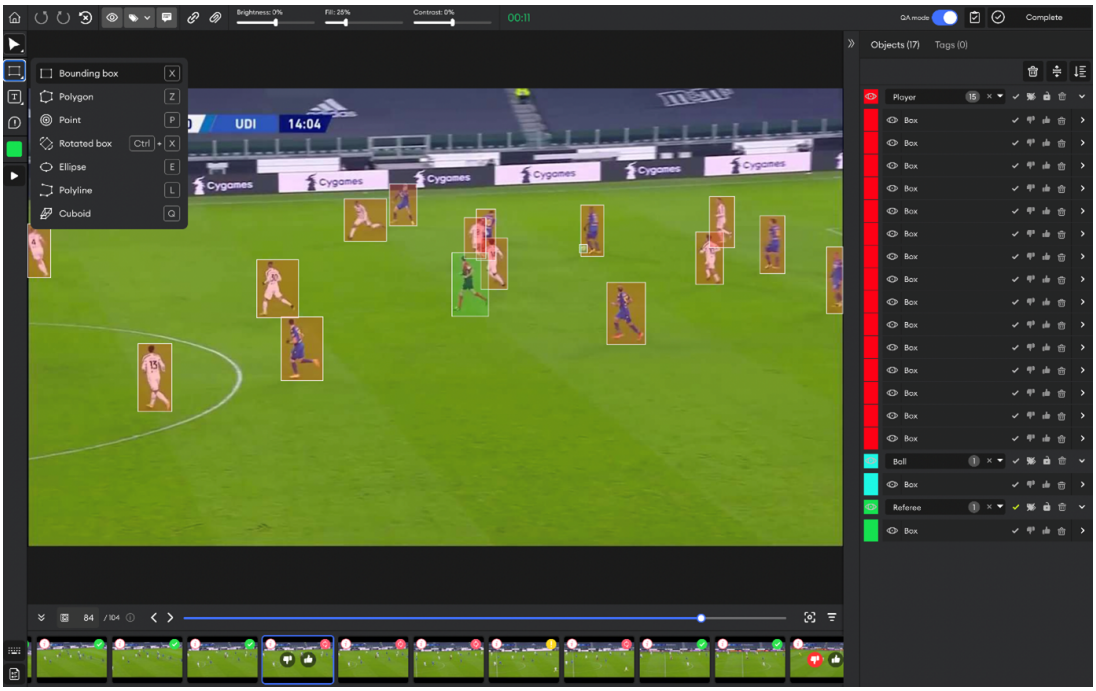


Figura 3.18: Ejemplo de interfaz de la herramienta SuperAnnotate.

Ventajas y desventajas

Ventajas	Desventajas
Interfaz intuitiva y fácil de usar.	Requiere conexión a Internet para funcionar.
Amplia gama de herramientas de anotación.	Algunas funciones avanzadas requieren conocimientos técnicos adicionales.
Capacidad de colaboración en tiempo real.	Precio elevado para usuarios individuales.
Integración con plataformas externas.	Curva de aprendizaje inicial para usuarios nuevos.
Automatización del proceso de anotación.	Limitaciones en la personalización de las tareas automáticas.
Soporte técnico robusto y actualizaciones frecuentes.	Posibles interrupciones del servicio por mantenimiento.

Cuadro 3.10: Ventajas y desventajas de SuperAnnotate.

Herramienta	Logo	Interfaz intuitiva	Entrenamiento de modelos	Coste	Escalabilidad de uso	Exportación
Label Studio		Sí	Sí	Gratuito (con limitaciones)	Pequeña a mediana escala	COCO, YOLO, Pascal VOC, CSV, TSV
CVAT		No	Sí	Gratuito (con opciones de pago Pro)	Mediana a gran escala	COCO, YOLO, Pascal VOC, CSV
VGG Image Annotator		Sí	No	Gratuito	Pequeña escala	JSON, CSV
RectLabel		Sí	No	Gratuito (con limitaciones)	Pequeña a mediana escala	COCO, YOLO, CSV, DOTA, Labelme
SuperAnnotate		Sí	Sí	Gratuito / Pro / Enterprise	Mediana a gran escala	COCO, YOLO, Pascal VOC
TFG		Sí	Sí	Gratuito	Pequeña escala	COCO, YOLO

Cuadro 3.11: Tabla comparativa de herramientas de anotación de imágenes.

3.3.6. Consideraciones para el TFG

Al desarrollar la herramienta de anotación asistida de imágenes de este TFG, se han considerado varias características clave que definen la calidad y utilidad de una solución de este tipo:

- **Facilidad de uso**: la interfaz debe ser intuitiva y permitir que los usuarios puedan comenzar a anotar imágenes rápidamente.
- **Flexibilidad**: capacidad para anotar diferentes tipos de imágenes y tareas.
- **Precisión**: herramientas precisas para una anotación de alta calidad.
- **Eficiencia**: procesos de anotación rápidos que optimicen el flujo de trabajo.
- **Escalabilidad**: posibilidad de gestionar grandes volúmenes de imágenes.

3.3.7. Conclusión

Las herramientas de anotación de imágenes representan una parte esencial del ciclo de aprendizaje automático en proyectos de Visión Artificial. Este estudio ha permitido analizar y comparar cinco herramientas representativas, que ofrecen funcionalidades variadas y adaptadas a distintos contextos de uso.

Aunque todas las herramientas revisadas permiten realizar tareas básicas de anotación, existen diferencias relevantes en cuanto a la experiencia de usuario, la integración con modelos de aprendizaje automático, la compatibilidad con distintos formatos y el enfoque en tareas específicas.

Label Studio, SuperAnnotate y Labelbox (aunque no analizada en detalle) se destacan como soluciones completas y flexibles. Roboflow constituye una buena opción para casos donde el entrenamiento automático juega un papel central. VGG Image Annotator, por su parte, es una herramienta sencilla y gratuita, ideal para proyectos individuales o de pequeña escala.

De cara al presente TFG, se ha optado por diseñar una herramienta que integre las características más valiosas observadas en las soluciones analizadas, priorizando la usabilidad, la flexibilidad en el proceso de anotación y la integración con procesos de inferencia automática, con el objetivo de facilitar la creación de datasets de alta calidad en proyectos de Visión Artificial.

Parte II

Documentación técnica

Capítulo 4

Descripción y desarrollo de la propuesta

Este capítulo presenta en detalle la propuesta desarrollada, abordando tanto la fase de análisis como las decisiones tomadas durante el diseño del sistema. Además, se incluye una descripción de los actores implicados y sus interacciones con la herramienta, así como la representación de los casos de uso principales.

El proyecto se ha desarrollado siguiendo la metodología ágil **ASAP (Agile Student Academic Projects)**, basada en prácticas de desarrollo incremental e iterativo, lo cual ha permitido una evolución continua del producto, integrando mejoras y validaciones en cada ciclo de trabajo. Esta metodología ha facilitado una mejor adaptación a los requisitos y la integración progresiva de funcionalidades, asegurando así la entrega de una herramienta funcional al final de cada iteración.

4.1. Análisis

El análisis es una fase fundamental dentro del ciclo de vida del software, ya que permite identificar, entender y documentar los requisitos funcionales y no funcionales del sistema. En este proyecto, el análisis ha estado centrado en comprender las necesidades de los usuarios (anotadores) y en definir de forma precisa las funcionalidades que debe incorporar la herramienta de anotación asistida de imágenes.

A partir de este análisis se han extraído los elementos clave que dan lugar a los casos de uso, los cuales se han representado gráficamente para facilitar su interpretación y estructurar de forma clara la interacción entre el usuario y el sistema. Dado que el objetivo principal del proyecto es facilitar el proceso de anotación de imágenes para tareas de detección de objetos, se ha prestado especial atención a la usabilidad, flexibilidad y capacidad de integración con modelos de aprendizaje automático.

4.1.1. Actores

En esta sección se describen los actores que interactúan con la aplicación. Cada actor representa un rol diferente dentro del sistema y participa en una serie de acciones específicas reflejadas en el diagrama de casos de uso. A continuación, se enumeran los actores identificados:

ID Actor	Descripción
A-01 Usuario	Actor principal del sistema. Es el responsable de importar imágenes, definir etiquetas, realizar anotaciones sobre las imágenes (dibujando bounding boxes y asignando clases), entrenar modelos, realizar inferencias automáticas y exportar los resultados. Representa al perfil técnico o analista que desea construir un conjunto de datos anotado con soporte de inferencia asistida.
A-02 Sistema	Representa el comportamiento interno automatizado del sistema. Se encarga de tareas como el entrenamiento del modelo, el guardado del mismo y la ejecución de inferencias automáticas. No requiere intervención directa del usuario, pero interactúa con él para ofrecer resultados o sugerencias en base a los datos procesados.

Cuadro 4.1: Actores del sistema.

4.1.2. Requisitos de información

Los requisitos de información definen todos los datos que el sistema necesita almacenar, gestionar o generar para su correcto funcionamiento. Incluyen tanto la información proporcionada por el usuario como la que es generada o manipulada por la aplicación durante el ciclo de vida del proyecto. A continuación, se presentan los principales requisitos de información de esta herramienta de anotación asistida, junto con un diccionario de datos que detalla los atributos asociados a cada uno de ellos.

RI-01. Información del proyecto
El sistema debe gestionar los metadatos relacionados con cada proyecto de anotación creado por el usuario.

Nombre	Descripción	Tipo
nombreProyecto	Nombre del proyecto	Cadena de texto
descripcionProyecto	Breve descripción del proyecto	Cadena de texto
rutaProyecto	Ruta en el sistema de archivos donde se guarda el proyecto	Ruta local
etiquetasDefinidas	Lista de categorías creadas por el usuario	Lista de cadenas
rutaImagenes	Carpeta con las imágenes cargadas para anotación	Ruta local
rutaAnotaciones	Carpeta donde se almacenan los ficheros de anotación	Ruta local

Cuadro 4.2: Requisito de información RI-01: Información del proyecto.

RI-02. Navegación entre imágenes
Durante el proceso de anotación, la aplicación debe gestionar la posición actual del usuario dentro del conjunto de imágenes.

Nombre	Descripción	Tipo
imagenActual	Identificador o nombre del archivo de la imagen seleccionada	Cadena de texto
listaImagenes	Lista completa de imágenes cargadas en el proyecto	Lista de cadenas

Cuadro 4.3: Requisito de información RI-02: Navegación entre imágenes.

RI-03. Parámetros de entrenamiento
El sistema permite configurar diferentes parámetros que afectan al entrenamiento del modelo de detección de objetos.

Nombre	Descripción	Tipo
numeroEpocas	Número total de épocas del entrenamiento	Entero
batchSize	Tamaño del lote (batch size)	Entero
umbralDeteccion	Confianza mínima para considerar una detección válida	Decimal (0–1)
freezeEpochs	Número de épocas con congelación parcial del modelo	Entero
porceValidacion	Porcentaje de imágenes reservadas para validación	Decimal (0–1)
constAprendizaje	Tasa de aprendizaje (learning rate)	Decimal
modeloBase	Arquitectura utilizada como modelo base (ej. YOLOv5, Faster R-CNN)	Cadena

Cuadro 4.4: Requisito de información RI-03: Parámetros de entrenamiento.

RI-04. Modelo entrenado
Una vez finalizado el proceso de entrenamiento, el sistema debe guardar el modelo generado para futuras inferencias.

Nombre	Descripción	Tipo
archivoModelo	Ruta del archivo del modelo entrenado (.pth, .pt...)	Ruta local
fechaEntrenamiento	Fecha de finalización del entrenamiento	Fecha
arquitectura	Modelo utilizado (YOLOv5, RetinaNet, etc.)	Cadena

Cuadro 4.5: Requisito de información RI-04: Modelo entrenado.

RI-05. Exportación de anotaciones
El sistema debe permitir exportar las anotaciones generadas en diferentes formatos estándar.

Nombre	Descripción	Tipo
archivoCOCO	Ruta del archivo JSON exportado en formato COCO	Ruta local
archivoYOLO	Ruta del archivo TXT exportado en formato YOLO	Ruta local
fechaExportacion	Fecha en la que se realizó la exportación	Fecha

Cuadro 4.6: Requisito de información RI-05: Exportación de anotaciones.

RI-06. Información de inferencias
Durante la validación automática, el sistema genera predicciones sobre nuevas imágenes que deben almacenarse de forma estructurada.

Nombre	Descripción	Tipo
imagenInferida	Nombre del archivo de la imagen inferida	Cadena de texto
boundingBoxes	Coordenadas de las detecciones realizadas	Lista de coordenadas
clasesDetectadas	Categorías asociadas a cada detección	Lista de cadenas
confianzaDeteccion	Score de confianza por cada predicción	Lista de decimales

Cuadro 4.7: Requisito de información RI-06: Información de inferencias.

En conjunto, estos requisitos garantizan que la herramienta sea capaz de gestionar correctamente toda la información necesaria durante el ciclo completo de anotación, entrenamiento e inferencia. Además, permiten mantener la trazabilidad del proyecto, asegurar la compatibilidad con otros entornos y facilitar futuras ampliaciones o migraciones del sistema.

4.1.3. Casos de uso

Los casos de uso son una herramienta clave en el proceso de análisis de requisitos, ya que permiten describir cómo los distintos actores interactúan con el sistema para lograr un objetivo concreto. Cada caso de uso representa un escenario de interacción entre el usuario o el sistema externo y la aplicación, especificando el flujo de eventos que se produce bajo condiciones normales y, en su caso, las posibles situaciones alternativas o de error.

En esta sección se describe el conjunto de casos de uso identificados para la aplicación web desarrollada en este Trabajo de Fin de Grado. La herramienta permite importar conjuntos de imágenes, definir etiquetas, realizar anotaciones manuales y automáticas (por inferencia), entrenar modelos y exportar las anotaciones generadas. Además, se contempla la gestión de proyectos, la edición de etiquetas y anotaciones, así como el guardado y carga de modelos entrenados previamente.

El actor principal es el usuario, que interactúa directamente con la interfaz web de la aplicación. Por otro lado, el sistema también actúa como actor secundario en los procesos automáticos de inferencia y entrenamiento.

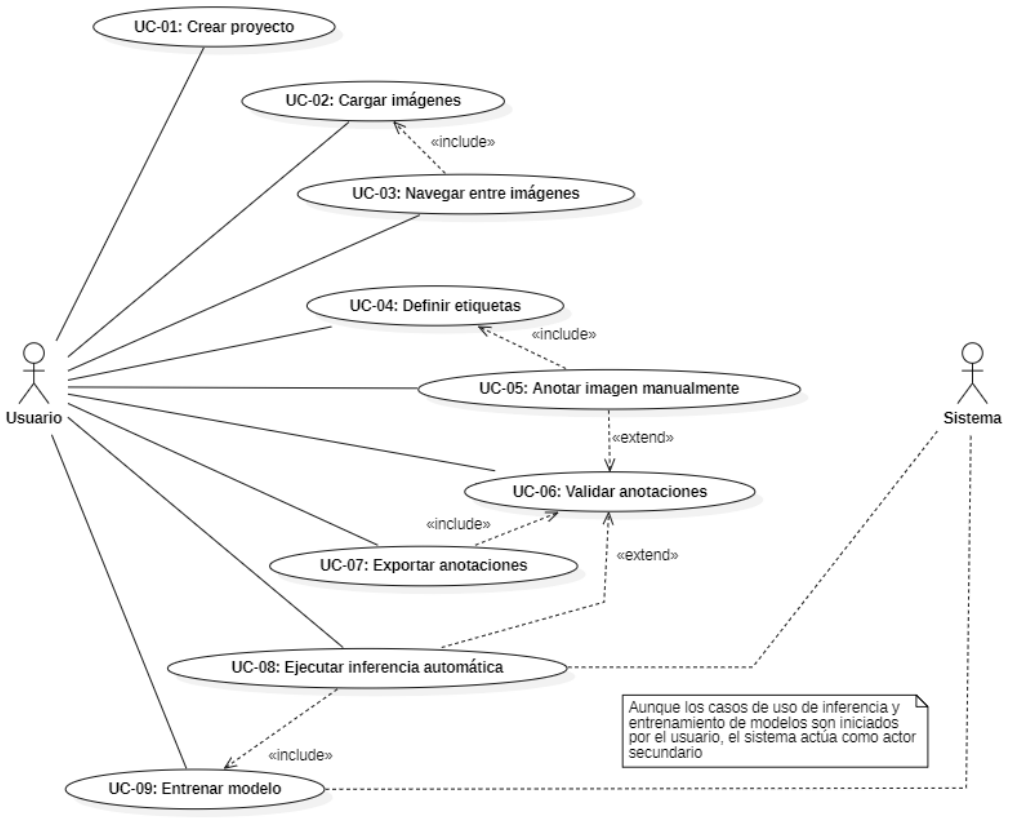


Figura 4.1: Diagrama de casos de uso del sistema.

A continuación, se presenta la especificación detallada de los casos de uso definidos en el sistema.

Especificación de casos de uso

Caso de uso	**UC-01: Crear proyecto**
Actor	Usuario
Descripción	El usuario define el nombre del proyecto, la ubicación en disco y las etiquetas que se utilizarán. Se crean las carpetas necesarias.
Precondición	El sistema debe estar en ejecución y el usuario debe haber accedido a la interfaz.
Secuencia normal	1. El usuario pulsa el botón "Nuevo proyecto". 2. Introduce nombre, ubicación y etiquetas iniciales. 3. El sistema valida los datos y crea la estructura de carpetas. 4. Se muestra el nuevo proyecto vacío en pantalla.
Postcondición	Se ha creado un nuevo proyecto disponible para añadir imágenes y anotaciones.
Excepciones	Si el nombre está vacío o la ruta es inválida, se muestra un mensaje de error y no se crea el proyecto.

Cuadro 4.8: Especificación del caso de uso UC-01: Crear proyecto.

Caso de uso	**UC-02: Cargar imágenes**
Actor	Usuario
Descripción	El usuario selecciona una carpeta con imágenes y el sistema carga todas las compatibles en el proyecto.
Precondición	Debe existir un proyecto activo y estar definido el directorio de imágenes.
Secuencia normal	1. El usuario pulsa "Importar imágenes". 2. Selecciona una carpeta local. 3. El sistema valida y carga los archivos con extensión admitida (ej. .jpg, .png). 4. Se muestran las miniaturas de las imágenes importadas.
Postcondición	Las imágenes están disponibles en la vista principal para ser navegadas y anotadas.
Excepciones	Si no se seleccionan imágenes válidas, se muestra un aviso y no se realiza la importación.

Cuadro 4.9: Especificación del caso de uso UC-02: Cargar imágenes.

Caso de uso	**UC-03: Navegar entre imágenes**
Actor	Usuario
Descripción	El usuario puede avanzar o retroceder entre las imágenes del proyecto mediante los controles de navegación.
Precondición	Debe existir al menos una imagen importada en el proyecto actual.
Secuencia normal	1. El usuario pulsa el botón “Siguiente“ o “Anterior“. 2. El sistema cambia la imagen mostrada en el visor principal.
Postcondición	La nueva imagen queda visible en el visor, lista para su anotación o revisión.
Excepciones	Si se intenta avanzar o retroceder fuera de los límites (primera o última imagen), no se realiza ninguna acción.

Cuadro 4.10: Especificación del caso de uso UC-03: Navegar entre imágenes.

Caso de uso	**UC-04: Definir etiquetas**
Actor	Usuario
Descripción	El usuario define el conjunto de etiquetas que podrán asignarse a las anotaciones de objetos en las imágenes.
Precondición	Debe haber un proyecto activo y cargado.
Secuencia normal	1. El usuario accede al menú de etiquetas. 2. Introduce nuevas etiquetas o modifica las existentes. 3. El sistema guarda los cambios.
Postcondición	El sistema actualiza la lista de etiquetas disponibles para usar en anotaciones.
Excepciones	Si se introduce una etiqueta vacía o duplicada, se muestra un mensaje de advertencia.

Cuadro 4.11: Especificación del caso de uso UC-04: Definir etiquetas.

Caso de uso	**UC-05: Anotar imagen manualmente**
Actor	Usuario
Descripción	El usuario dibuja un rectángulo sobre la imagen, selecciona una etiqueta y guarda la anotación correspondiente.
Precondición	Debe haber una imagen cargada y al menos una etiqueta definida.
Secuencia normal	1. El usuario pulsa sobre la imagen y arrastra para dibujar una caja delimitadora. 2. Se abre un cuadro emergente con las etiquetas disponibles. 3. El usuario selecciona la etiqueta y confirma. 4. El sistema guarda la anotación asociada a esa imagen.
Postcondición	La imagen queda anotada con la etiqueta seleccionada.
Excepciones	Si el usuario cancela el proceso o no selecciona ninguna etiqueta, no se guarda la anotación.

Cuadro 4.12: Especificación del caso de uso UC-05: Anotar imagen manualmente.

Caso de uso	**UC-06: Validar anotaciones**
Actor	Usuario
Descripción	El usuario revisa las anotaciones automáticas o manuales de una imagen y las valida si son correctas.
Precondición	Debe existir al menos una anotación en la imagen.
Secuencia normal	1. El usuario selecciona una imagen anotada. 2. Revisa las anotaciones mostradas. 3. Pulsa el botón de "Validar imagen". 4. El sistema marca la imagen como validada.
Postcondición	La imagen pasa al conjunto de imágenes validadas y se desactiva su edición.
Excepciones	Si no hay anotaciones, se muestra un aviso y no se permite validar.

Cuadro 4.13: Especificación del caso de uso UC-06: Validar anotaciones.

Caso de uso	**UC-07: Exportar anotaciones**
Actor	Usuario
Descripción	El usuario selecciona el formato de exportación (COCO o YOLO) y guarda las anotaciones del proyecto en dicho formato.
Precondición	Debe existir al menos una imagen validada con anotaciones.
Secuencia normal	1. El usuario pulsa el botón "Exportar". 2. Selecciona el formato de exportación deseado. 3. El sistema genera los archivos correspondientes y los guarda en la carpeta del proyecto.
Postcondición	Se ha creado un archivo de exportación con las anotaciones validadas en el formato seleccionado.
Excepciones	Si no hay imágenes validadas o el proceso falla, se muestra un mensaje de error.

Cuadro 4.14: Especificación del caso de uso UC-07: Exportar anotaciones.

Caso de uso	**UC-08: Ejecutar inferencia automática**
Actor	Usuario
Descripción	El usuario selecciona una imagen y ejecuta un modelo de detección previamente entrenado para generar anotaciones automáticas.
Precondición	Debe existir un modelo entrenado cargado en el sistema.
Secuencia normal	1. El usuario pulsa el botón "Inferencia". 2. El sistema ejecuta el modelo sobre la imagen seleccionada. 3. Se generan automáticamente las cajas delimitadoras con sus etiquetas. 4. Las anotaciones se muestran sobre la imagen.
Postcondición	La imagen contiene anotaciones sugeridas por el modelo.
Excepciones	Si no hay modelo cargado o hay un error en la inferencia, se muestra un mensaje de error.

Cuadro 4.15: Especificación del caso de uso UC-08: Ejecutar inferencia automática.

Caso de uso	**UC-09: Entrenar modelo**
Actor	Usuario
Descripción	El usuario define los parámetros de entrenamiento y lanza el proceso para generar un modelo de detección de objetos.
Precondición	Deben existir anotaciones exportadas en el formato requerido por el entrenamiento.
Secuencia normal	1. El usuario accede a la pestaña de entrenamiento. 2. Introduce los parámetros: modelo, épocas, batch size, umbral, etc. 3. Pulsa el botón "Entrenar modelo". 4. El sistema lanza el proceso de entrenamiento con los datos exportados.
Postcondición	Se genera un archivo con el modelo entrenado, listo para usarse en inferencia.
Excepciones	Si falta algún parámetro o los datos de entrenamiento no son válidos, se interrumpe el proceso y se informa al usuario.

Cuadro 4.16: Especificación del caso de uso UC-09: Entrenar modelo.

4.1.4. Requisitos funcionales

Los requisitos funcionales describen las capacidades y comportamientos que debe ofrecer la aplicación web desarrollada. Se han definido a partir de los casos de uso especificados anteriormente, agrupando las funcionalidades por bloques temáticos: gestión de proyectos, etiquetas, imágenes, anotaciones, modelos e inferencia. A continuación, se detallan dichos requisitos:

Gestión de proyectos

ID	Requisito funcional
RF-01	El usuario podrá crear un nuevo proyecto indicando nombre, ubicación y etiquetas iniciales.
RF-02	El usuario podrá abrir un proyecto previamente guardado.
RF-03	El usuario podrá eliminar un proyecto existente del sistema.
RF-04	El usuario podrá guardar un proyecto con las imágenes y anotaciones realizadas.

Cuadro 4.17: Requisitos funcionales de gestión de proyectos.

Gestión de etiquetas

ID	Requisito funcional
RF-05	El usuario podrá definir etiquetas para anotar objetos en las imágenes.
RF-06	El usuario podrá editar las etiquetas existentes.
RF-07	El usuario podrá eliminar etiquetas previamente definidas.

Cuadro 4.18: Requisitos funcionales de gestión de etiquetas.

Gestión de imágenes

ID	Requisito funcional
RF-08	El usuario podrá importar conjuntos de imágenes compatibles (.jpg, .png).
RF-09	El usuario podrá navegar entre las imágenes importadas en el proyecto.

Cuadro 4.19: Requisitos funcionales de gestión de imágenes.

Anotación de imágenes

ID	Requisito funcional
RF-10	El usuario podrá anotar imágenes de forma manual dibujando cajas delimitadoras y asignando etiquetas.
RF-11	El usuario podrá editar las anotaciones realizadas en las imágenes.
RF-12	El usuario podrá eliminar anotaciones previamente creadas.
RF-13	El usuario podrá exportar las anotaciones validadas en formato COCO o YOLO.

Cuadro 4.20: Requisitos funcionales de anotación de imágenes.

Entrenamiento de modelos

ID	Requisito funcional
RF-14	El sistema permitirá entrenar modelos de detección de objetos a partir de las anotaciones realizadas.
RF-15	El usuario podrá configurar los parámetros del entrenamiento (modelo base, épocas, batch size, etc.).
RF-16	El usuario podrá guardar los modelos entrenados para su uso posterior.
RF-17	El usuario podrá cargar modelos previamente entrenados en el sistema.

Cuadro 4.21: Requisitos funcionales de entrenamiento de modelos.

Inferencia automática

ID	Requisito funcional
RF-18	El sistema permitirá ejecutar inferencias sobre imágenes utilizando los modelos cargados.
RF-19	El sistema representará visualmente los resultados de las inferencias mediante cuadros delimitadores en las imágenes.

Cuadro 4.22: Requisitos funcionales de inferencia automática.

4.1.5. Requisitos no funcionales

Los requisitos no funcionales representan características generales o restricciones del sistema software. A diferencia de los requisitos funcionales, no definen qué funcionalidades debe tener el sistema, sino que describen cómo debe comportarse en términos de calidad, rendimiento, usabilidad, seguridad, eficiencia o mantenimiento.

Estos requisitos garantizan que el sistema sea práctico, fiable y escalable en contextos reales de uso. A continuación, se especifican los requisitos no funcionales identificados para la aplicación desarrollada:

Usabilidad

ID	Requisito no funcional
RNF-01	La interfaz de usuario deberá ser intuitiva, clara y fácil de usar, minimizando la curva de aprendizaje para usuarios sin conocimientos técnicos.
RNF-02	La herramienta de entrenamiento deberá mostrar métricas básicas (como precisión o pérdida) que ayuden a interpretar el rendimiento del modelo entrenado.

Cuadro 4.23: Requisitos no funcionales de usabilidad.

Compatibilidad

ID	Requisito no funcional
RNF-03	La aplicación deberá ser completamente funcional en los principales navegadores web modernos (Chrome, Firefox, Edge).
RNF-04	El sistema deberá permitir importar y exportar anotaciones en los formatos más comunes: COCO y YOLO.

Cuadro 4.24: Requisitos no funcionales de compatibilidad.

Eficiencia y rendimiento

ID	Requisito no funcional
RNF-05	El tiempo de respuesta de la aplicación al realizar acciones como cargar imágenes, ejecutar inferencias o entrenar modelos deberá ser razonablemente rápido.
RNF-06	El sistema deberá ser eficiente en el uso de recursos computacionales, minimizando el consumo de CPU, memoria y almacenamiento.

Cuadro 4.25: Requisitos no funcionales de eficiencia y rendimiento.

Escalabilidad

ID	Requisito no funcional
RNF-07	La aplicación deberá ser capaz de manejar conjuntos de imágenes de tamaño variable, desde unas pocas unidades hasta miles de elementos sin degradar notablemente el rendimiento.
RNF-08	El sistema deberá permitir incorporar modelos de distinto tamaño y complejidad, sin necesidad de modificaciones manuales.

Cuadro 4.26: Requisitos no funcionales de escalabilidad.

Seguridad y privacidad

ID	Requisito no funcional
RNF-09	La aplicación deberá garantizar la protección de los datos del usuario, evitando pérdidas, accesos no autorizados o filtraciones.
RNF-10	El sistema deberá cumplir con las regulaciones de protección de datos aplicables (por ejemplo, GDPR), en la medida en que se gestionen datos sensibles.

Cuadro 4.27: Requisitos no funcionales de seguridad y privacidad.

Fiabilidad y robustez

ID	Requisito no funcional
RNF-11	El sistema deberá ser tolerante a errores, gestionando fallos inesperados sin comprometer el estado del proyecto o la integridad de los datos.
RNF-12	La aplicación deberá validar los datos de entrada del usuario (etiquetas, rutas, formatos) para evitar errores en la ejecución.

Cuadro 4.28: Requisitos no funcionales de fiabilidad y robustez.

Extensibilidad y documentación

ID	Requisito no funcional
RNF-13	El sistema deberá estar diseñado de forma modular, facilitando la adición de nuevas funcionalidades sin necesidad de modificar partes principales del código.
RNF-14	El código deberá estar estructurado de forma clara para permitir su mantenimiento, actualización o mejora futura.
RNF-15	La aplicación deberá contar con documentación accesible que describa su instalación, configuración y uso básico, incluyendo ejemplos de entrenamiento y exportación.

Cuadro 4.29: Requisitos no funcionales de extensibilidad y documentación.

4.2. Diseño

La fase de diseño consiste en transformar los requisitos obtenidos durante el análisis en una representación estructurada que permita implementar el sistema de forma organizada y eficiente. En esta etapa se definen los componentes del sistema, su organización interna y las relaciones entre ellos.

Este diseño abarca tanto la arquitectura física como la arquitectura lógica. La primera se centra en la disposición de los elementos físicos y su interacción, mientras que la segunda organiza los componentes lógicos del software en distintas capas funcionales. Para facilitar su comprensión, ambos diseños se representan mediante diagramas explicativos.

4.2.1. Arquitectura física

La arquitectura física representa los componentes físicos del sistema y cómo se relacionan entre sí. En este caso, el sistema desarrollado es una aplicación web que se ejecuta completamente en local, por lo que tanto el cliente (navegador del usuario) como el servidor (aplicación Flask) y los modelos de detección se alojan en el mismo equipo.

En la figura 4.2 se muestra cómo el usuario accede a la aplicación a través de un navegador, que se comunica con el servidor mediante HTTP. El servidor gestiona la carga de imágenes, las anotaciones, el entrenamiento y la ejecución de inferencias, utilizando archivos y modelos almacenados localmente.

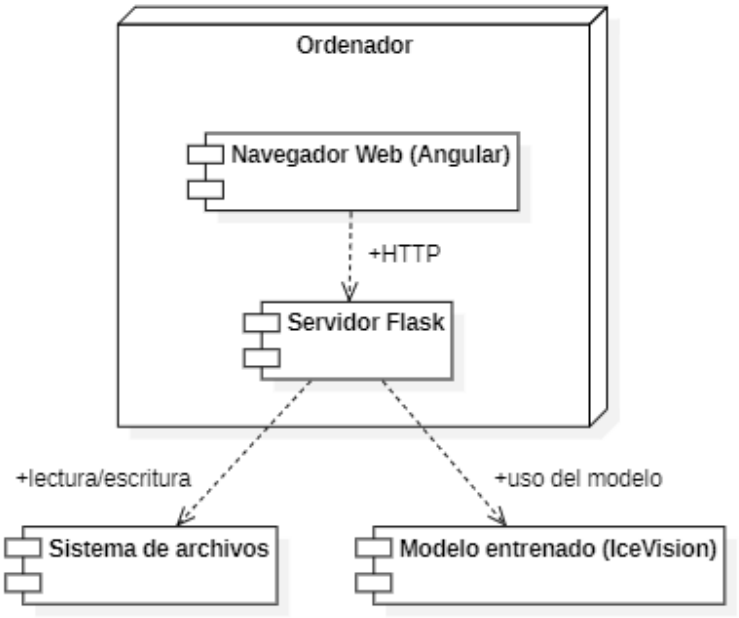


Figura 4.2: Arquitectura física del sistema.

4.2.2. Arquitectura lógica

La arquitectura lógica representa la organización interna del sistema desde un punto de vista funcional. A diferencia de la arquitectura física, que muestra cómo se despliega el sistema sobre componentes físicos, la arquitectura lógica se centra en los módulos del software, su agrupación en capas y las relaciones entre ellos.

El sistema desarrollado se organiza en tres capas principales:

- **Capa de presentación**: desarrollada con Angular, contiene la interfaz web que permite al usuario interactuar con la aplicación para gestionar proyectos, etiquetar imágenes, lanzar entrenamientos y realizar inferencias.
- **Capa de lógica de negocio**: implementada en Flask, contiene los componentes responsables de procesar las acciones del usuario, incluyendo el gestor de proyectos, anotaciones y modelos, así como los módulos específicos de entrenamiento e inferencia.
- **Capa de acceso a datos**: se encarga de almacenar y recuperar los elementos persistentes del sistema, como las imágenes, anotaciones y modelos entrenados. Se realiza mediante el sistema de archivos local.

La figura 4.3 muestra esta estructura lógica y las relaciones entre los diferentes componentes.

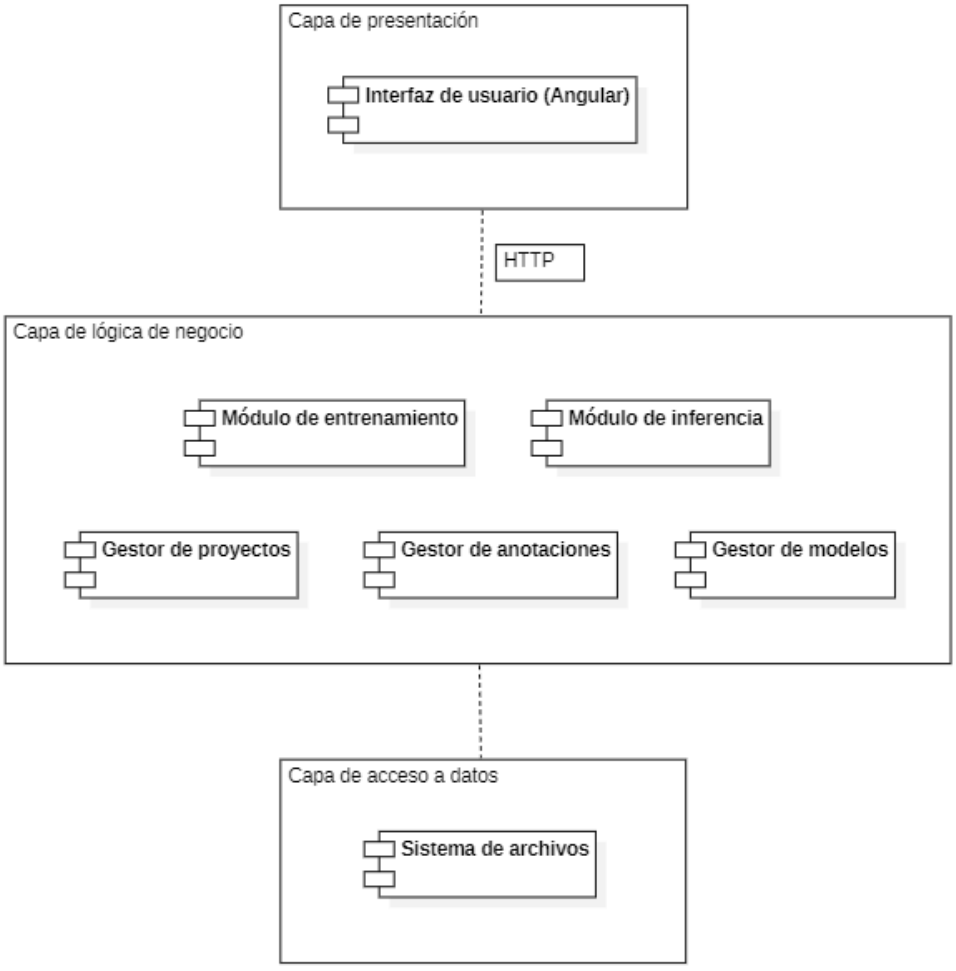


Figura 4.3: Diagrama de arquitectura lógica del sistema.

4.2.3. Diagrama de estados

El diagrama de estados representa el ciclo de vida de una imagen dentro del flujo de trabajo de la aplicación. A través de una serie de estados y transiciones, se describe cómo evoluciona una imagen desde el momento en que es importada hasta que es exportada tras ser anotada y validada.

En la figura 4.4 se muestra este ciclo, que comienza con la importación de la imagen, seguida por posibles fases de anotación (manual o automática), validación y, finalmente, exportación. Este diagrama es útil para entender el comportamiento dinámico del sistema ante las acciones del usuario.

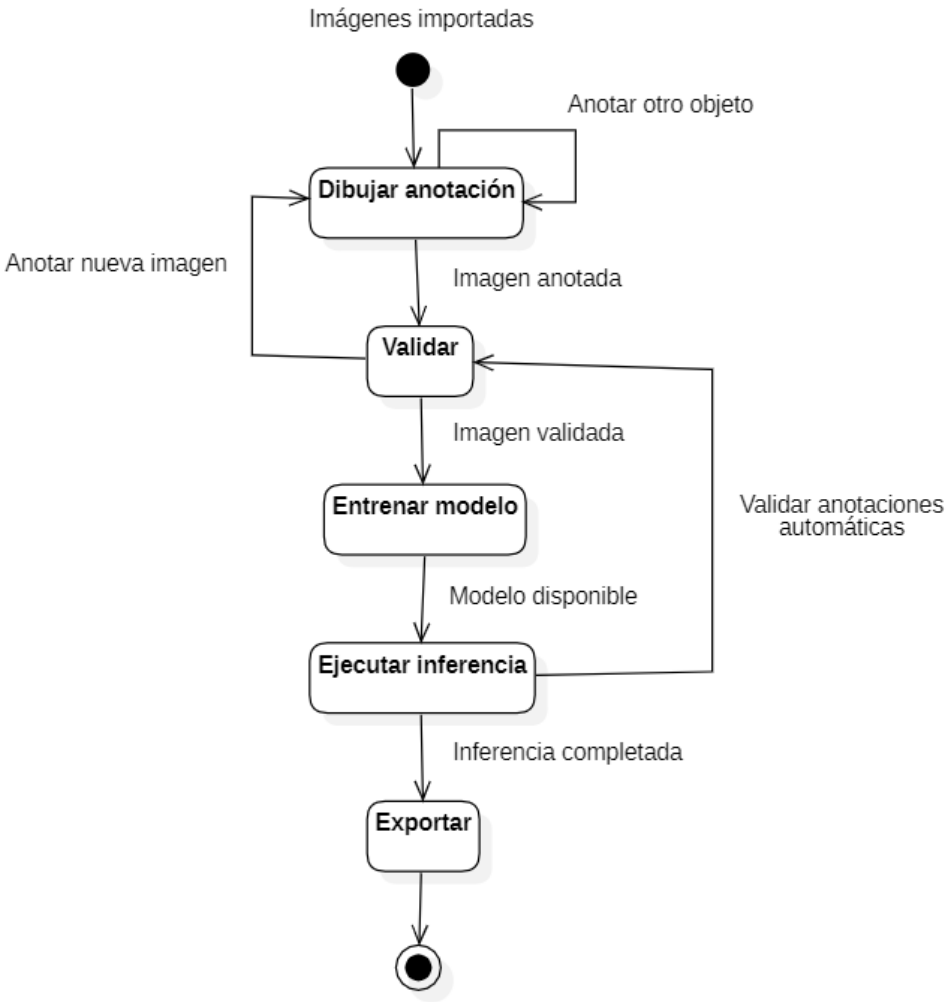


Figura 4.4: Diagrama de estados del ciclo de vida de una imagen.

4.2.4. Diagrama de comunicación

El diagrama de comunicación ilustra cómo interactúan los distintos componentes del sistema durante la ejecución del caso de uso CU-08: Ejecutar inferencia automática. Este tipo de diagrama muestra la secuencia de mensajes intercambiados entre los participantes para completar una funcionalidad.

En este caso, el usuario lanza la inferencia desde la interfaz Angular, que envía una petición HTTP al servidor Flask. Este, a su vez, accede al modelo entrenado y a la imagen correspondiente para ejecutar la inferencia y devolver las anotaciones generadas.

La figura 4.5 muestra esta secuencia de mensajes paso a paso.

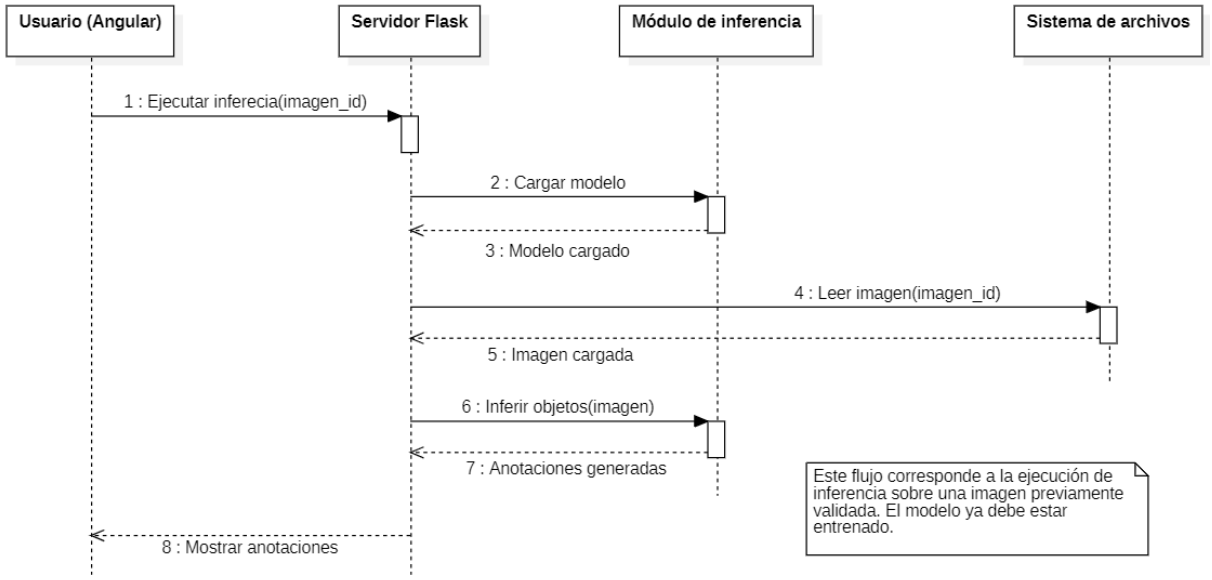
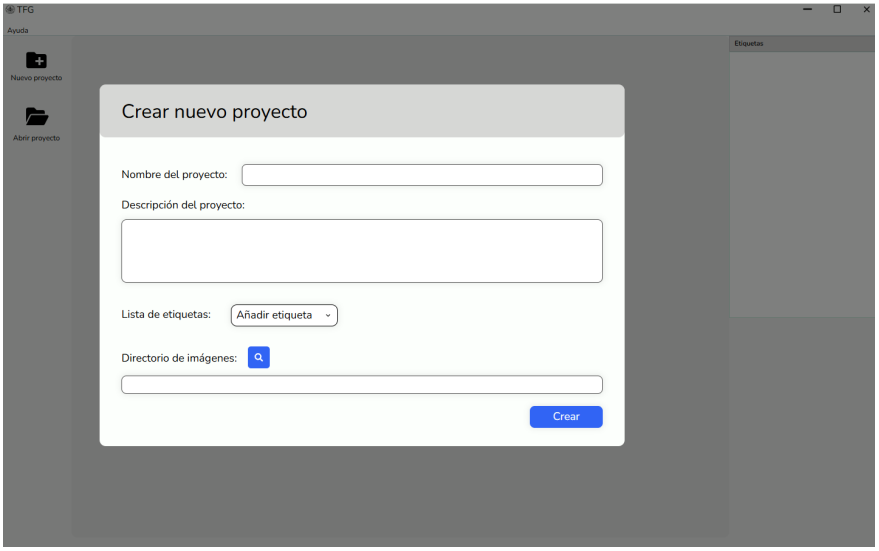


Figura 4.5: Diagrama de comunicación del caso de uso CU-08: Ejecutar inferencia automática.

4.2.5. Diseño de interfaz de usuario

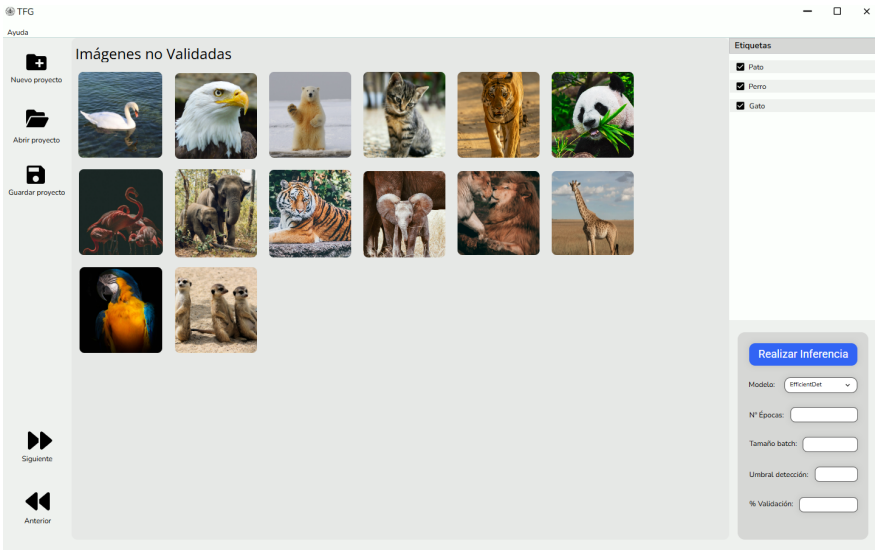
El diseño de interfaz de usuario permite analizar cómo los usuarios interactúan con la aplicación, centrándose en facilitar una experiencia eficiente, intuitiva y agradable. Cada pantalla está pensada para guiar al usuario en el flujo completo: desde la creación de un nuevo proyecto, la importación de imágenes, la navegación, la anotación, validación e inferencia. A continuación, se describen las distintas interfaces implementadas en la aplicación.

ID	IU-01
Descripción de pantalla	Cuadro de diálogo para crear un nuevo proyecto. El usuario introduce el nombre, descripción, etiquetas iniciales y el directorio donde se encuentran las imágenes.
Eventos	Al hacer clic en "Nuevo proyecto" desde el menú lateral.
Elementos clave	Input de texto para nombre del proyecto, textarea para descripción, selector desplegable de etiquetas, explorador de directorio de imágenes y botón de confirmación "Crear".
Propósito	Permitir al usuario iniciar un nuevo flujo de trabajo definiendo todos los parámetros iniciales del proyecto.
Boceto	

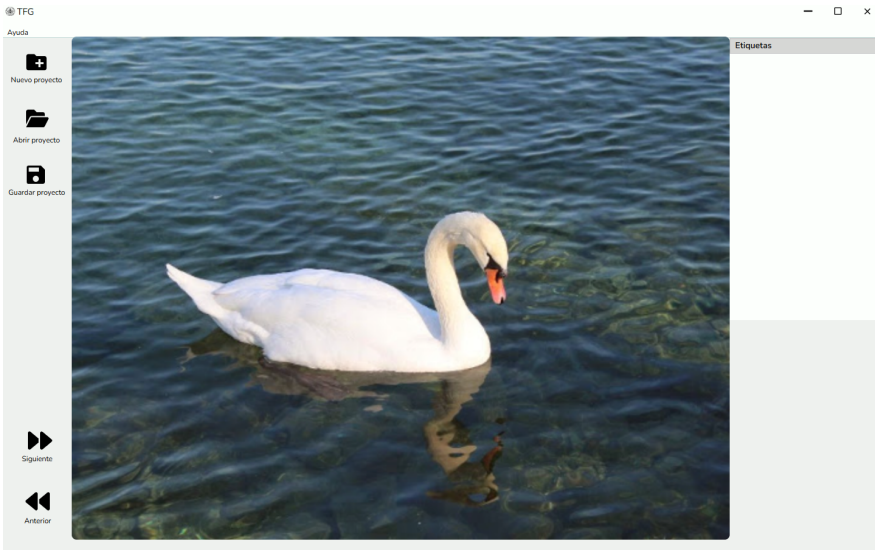
Cuadro 4.30: Interfaz IU-01: Creación de nuevo proyecto.

ID	IU-02
Descripción de pantalla	Pantalla principal tras la creación del proyecto, cuando aún no se han importado imágenes. Se informa al usuario y se le invita a importar.
Eventos	Se muestra automáticamente tras crear un nuevo proyecto si no se han cargado imágenes.
Elementos clave	Botón azul "Importar" centrado y mensaje indicando que no hay imágenes disponibles.
Propósito	Guiar al usuario al siguiente paso imprescindible: cargar imágenes para poder trabajar.
Boceto	

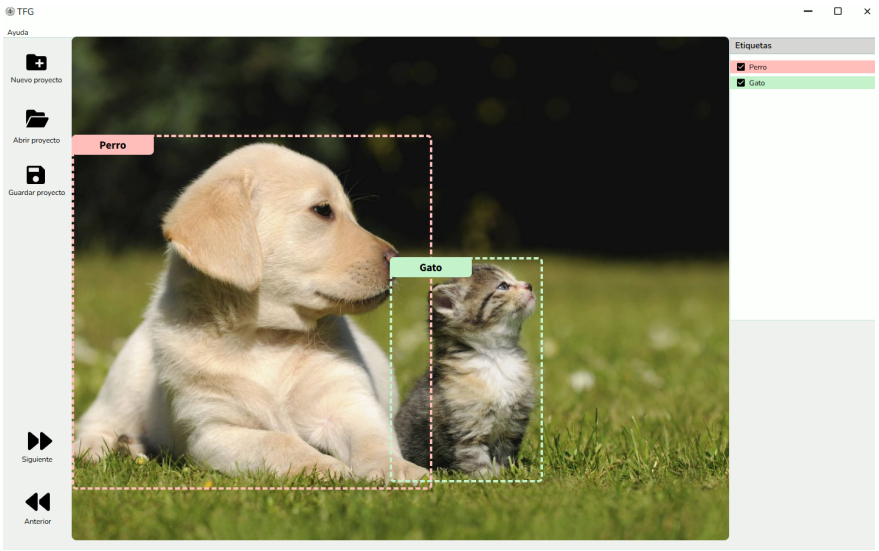
Cuadro 4.31: Interfaz IU-02: Importación inicial de imágenes.

ID	IU-03
Descripción de pantalla	Visualización en cuadrícula de todas las imágenes no validadas. Se permite navegar por ellas, seleccionar etiquetas y configurar los parámetros para ejecutar inferencia.
Eventos	Tras importar imágenes, se muestra esta vista de gestión previa a la anotación.
Elementos clave	Miniaturas de imágenes, selector de etiquetas (checkbox), parámetros de inferencia (modelo, número de épocas, batch size, umbral...), botón "Realizar inferencia".
Propósito	Facilitar el acceso visual a todas las imágenes, configurar la inferencia y organizar el flujo de anotaciones.
Boceto	

Cuadro 4.32: Interfaz IU-03: Gestión de imágenes no validadas.

ID	IU-04
Descripción de pantalla	Pantalla de anotación manual: una imagen a pantalla completa, con posibilidad de dibujar cajas y seleccionar etiquetas.
Eventos	Al hacer clic sobre una miniatura de imagen en IU-03.
Elementos clave	Imagen centrada, barra lateral con etiquetas, botones de navegación (Siguiente / Anterior), posibilidad de dibujar y asignar etiquetas a cada objeto detectado.
Propósito	Permitir la anotación precisa de los objetos dentro de cada imagen con una interfaz simple e interactiva.
Boceto	

Cuadro 4.33: Interfaz IU-04: Vista de anotación de una imagen.

ID	IU-05
Descripción de pantalla	Imagen anotada con cajas delimitadoras y etiquetas visibles. Se permite modificar, eliminar o validar las anotaciones existentes.
Eventos	Tras haber completado la anotación de una imagen (manual o por inferencia).
Elementos clave	Visualización de cajas en colores distintos según etiqueta, nombres de las etiquetas, panel lateral con listado de etiquetas activas.
Propósito	Revisar, modificar y validar las anotaciones antes de pasar al siguiente paso (entrenamiento o exportación).
Boceto	

Cuadro 4.34: Interfaz IU-05: Imagen anotada con etiquetas visibles.

4.3. Implementación

La implementación del proyecto se estructura en dos componentes principales: el **backend**, desarrollado en `Python` utilizando `Flask`, y el **frontend**, construido con `Angular`. En esta sección se detallan los aspectos más relevantes del código fuente, la arquitectura funcional y las decisiones clave de desarrollo.

4.3.1. Estructura del proyecto

La organización general del código se muestra en la Figura 4.6.

- `backend/`: Contiene la lógica del servidor Flask, con un archivo `app.py` que gestiona las rutas de creación de proyectos, carga de imágenes, validación, anotaciones, etiquetas, entrenamiento e inferencia.
- `frontend/`: Aplicación Angular con los componentes organizados en carpetas como `home`, `create-project`, `image-manager`, `image-annotator` y `services` para manejar las llamadas al backend.
- `projects/`: Carpeta donde se almacenan los proyectos creados, imágenes, anotaciones, configuraciones, modelos entrenados y resultados de inferencia.

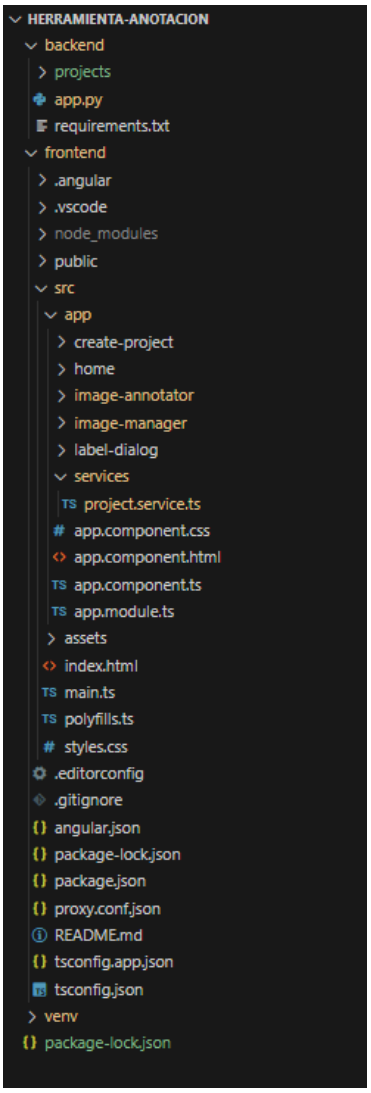


Figura 4.6: Estructura de carpetas del proyecto.

4.3.2. Backend: lógica en Flask

El archivo `app.py` implementa las rutas de la API REST que permiten la gestión de proyectos. Entre las funcionalidades más destacadas se encuentran:

- **Creación de proyectos**: Ruta POST a `/api/create_project`, que genera la estructura de carpetas y el fichero `project.json`.
- **Carga de imágenes**: Ruta POST a `/api/upload_images/`, que guarda las imágenes subidas y actualiza el fichero del proyecto.
- **Gestión de etiquetas**: Rutas para añadir, eliminar y recuperar etiquetas de un proyecto.
- **Anotaciones**: Ruta POST a `/api/annotate_image` que guarda en formato `JSON` las anotaciones manuales realizadas.
- **Validación de imágenes**: Se permite mover imágenes de la carpeta de no validadas a validadas, así como revertirlo.
- **Entrenamiento del modelo**: Ruta POST a `/api/train_model` que permite entrenar un modelo de detección con los parámetros seleccionados.
- **Inferencia sobre imágenes**: Ruta POST a `/api/infer` que aplica un modelo entrenado para predecir anotaciones sobre nuevas imágenes.
- **Consulta y eliminación de proyectos**: Se incluye la ruta `/api/list_projects` y `/api/delete_project` para consultar y eliminar proyectos.

4.3.3. Frontend: estructura y navegación en Angular

El frontend implementa una SPA en Angular con Angular Material. Se divide en varios componentes:

- **HomeComponent**: Muestra el listado de proyectos existentes y permite crear uno nuevo.
- **CreateProjectComponent**: Formulario para definir nombre, descripción y directorio de imágenes del nuevo proyecto.
- **ImageManagerComponent**: Gestión centralizada de las imágenes. Muestra las miniaturas, permite validarlas, eliminarlas, etiquetar, ejecutar inferencias y configurar el entrenamiento del modelo.

- **ImageAnnotatorComponent**: Herramienta de anotación con visualización de etiquetas, dibujo de rectángulos, selección de etiquetas mediante un diálogo y guardado de anotaciones.
- **LabelDialogComponent**: Componente auxiliar para seleccionar etiquetas de una lista desplegable en forma de diálogo.

La navegación entre componentes se realiza mediante rutas como:

- `/home`: Vista inicial.
- `/create-project`: Formulario de nuevo proyecto.
- `/image-manager/ projectName`: Gestión de imágenes de un proyecto.
- `/image-annotator/ projectName/ imageName/ folder`: Pantalla de anotación.

4.3.4. Componentes destacables y lógica

Gestión de etiquetas El usuario puede añadir etiquetas a un proyecto desde la vista `image-manager`. Estas se guardan en `project.json` y se utilizan para las anotaciones.

Anotación de imágenes La lógica en `image-annotator.component.ts` permite dibujar cajas delimitadoras relativas (valores en porcentaje) sobre la imagen seleccionada. Al soltar el ratón, se muestra un cuadro de diálogo para elegir la etiqueta. Las anotaciones se pueden mostrar/ocultar individualmente, eliminar o validar.

Guardado y recuperación Las anotaciones se guardan en ficheros `JSON` por imagen, almacenados dentro de la carpeta `annotations`. Al validar una imagen, se mueve su fichero a formato validado.

Exportación Desde la barra superior, el usuario puede exportar los datos anotados en formato `YOLO`, `COCO` o `PascalVOC`. Esta funcionalidad está disponible y lista para utilizar.

Entrenamiento de modelo En la vista de imágenes validadas, el usuario puede configurar parámetros como modelo, `batch size`, `epochs`, tasa de aprendizaje, `freeze epochs` y umbral de detección. Esta información se envía al backend, donde se entrena el modelo y se guarda el archivo resultante.

Inferencia Una vez entrenado el modelo, se puede aplicar sobre las imágenes no validadas. El resultado de la inferencia son anotaciones automáticas que se muestran sobre las imágenes, y pueden ser editadas o validadas por el usuario.

4.3.5. Servicios Angular y comunicación con backend

Todas las operaciones del cliente con el servidor se centralizan en `project.service.ts`. Este servicio contiene funciones como:

- `createProject(name, description)`
- `uploadImages(projectName, files)`
- `getTags(name)` y `addTag(name, tag)`
- `annotateImage(...)` y `getAnnotations(...)`
- `trainModel(...)` y `infer(...)`, para entrenamiento e inferencia

El uso de `Observable` permite suscripciones y manejo de respuestas asíncronas.

4.4. Pruebas

Las pruebas representan una revisión final de las especificaciones, el diseño y la implementación del sistema. Aunque durante el desarrollo se han llevado a cabo pruebas unitarias y locales de forma continua, ha sido necesario definir una estrategia de validación más sistemática para verificar que todas las funcionalidades clave del sistema funcionan correctamente.

Para ello, se ha optado por el uso de **pruebas de caja negra**, centradas en evaluar el comportamiento observable del sistema sin tener en cuenta su estructura interna. Cada prueba incluye una descripción, precondiciones, datos de entrada, salida esperada y salida obtenida.

ID de prueba	PR-01: Creación de proyecto
Descripción	Verifica que se puede crear un nuevo proyecto introduciendo nombre y descripción.
Precondición	El servidor debe estar ejecutándose.
Datos de entrada	Nombre = “proyecto1”, Descripción = “Proyecto de prueba”.
Resultado esperado	Se crea la carpeta correspondiente y un fichero `project.json`.
Resultado obtenido	Se crea correctamente la estructura del proyecto y se devuelve mensaje de éxito.

Cuadro 4.35: Prueba PR-01: Creación de proyecto

ID de prueba	PR-02: Carga de imágenes
Descripción	Comprueba que se pueden importar correctamente varias imágenes.
Precondición	Debe existir un proyecto creado.
Datos de entrada	3 imágenes en formato JPG.
Resultado esperado	Las imágenes se guardan en la carpeta "No validadas" y se actualiza el `project.json`.
Resultado obtenido	Las imágenes se almacenan correctamente y el fichero se actualiza.

Cuadro 4.36: Prueba PR-02: Carga de imágenes

ID de prueba	PR-03: Anotación manual
Descripción	Verifica que se puede dibujar una anotación sobre una imagen y guardarla con una etiqueta.
Precondición	Debe haber al menos una imagen importada y al menos una etiqueta creada.
Datos de entrada	Dibujar un rectángulo y seleccionar la etiqueta "persona".
Resultado esperado	Se guarda un fichero JSON con las coordenadas relativas y la etiqueta asignada.
Resultado obtenido	El fichero se guarda correctamente y es recuperable desde el sistema.

Cuadro 4.37: Prueba PR-03: Anotación manual

ID de prueba	PR-04: Validación de imagen
Descripción	Comprueba que al validar una imagen, se mueve correctamente a la carpeta "Validadas" y sus anotaciones se renombran.
Precondición	Debe haber una imagen con anotaciones en "No validadas".
Datos de entrada	Imagen con nombre "imagen1.jpg".
Resultado esperado	La imagen se mueve a "Validadas" y el fichero de anotaciones pasa a llamarse "imagen1.jpg.validated.json".
Resultado obtenido	Se realiza la validación correctamente.

Cuadro 4.38: Prueba PR-04: Validación de imagen

ID de prueba	PR-05: Entrenamiento de modelo
Descripción	Comprueba que se puede iniciar el entrenamiento de un modelo a partir de imágenes validadas.
Precondición	Deben existir imágenes validadas y sus anotaciones correspondientes.
Datos de entrada	Modelo = YOLOv5, Epochs = 50, Batch size = 16, Learning rate = 0.001.
Resultado esperado	Se lanza el proceso de entrenamiento y se guarda un fichero del modelo
Resultado obtenido	El entrenamiento finaliza correctamente y se guarda el archivo `model.pth`.

Cuadro 4.39: Prueba PR-05: Entrenamiento de modelo

ID de prueba	PR-06: Inferencia de modelo entrenado
Descripción	Verifica que se puede lanzar una inferencia sobre una imagen usando el modelo entrenado.
Precondición	Debe existir un modelo entrenado y una imagen a inferir.
Datos de entrada	Imagen: “imagen2.jpg”, Umbral = 0.5
Resultado esperado	Se obtienen predicciones con etiquetas, coordenadas y puntuaciones.
Resultado obtenido	Se recibe una lista con las detecciones realizadas correctamente.

Cuadro 4.40: Prueba PR-06: Inferencia de modelo entrenado

Capítulo 5

Conclusiones y trabajo futuro

En este capítulo se presenta una reflexión general sobre el trabajo realizado, evaluando el grado de cumplimiento de los objetivos establecidos, analizando el impacto de la metodología empleada y compartiendo una perspectiva personal sobre la experiencia. Asimismo, se detallan posibles mejoras y líneas de trabajo futuras que permitirían ampliar las funcionalidades del sistema.

5.1. Conclusiones

5.1.1. Perspectiva del proyecto

El objetivo principal de este Trabajo de Fin de Grado ha sido desarrollar una herramienta web de anotación de imágenes que permita crear proyectos, importar y gestionar imágenes, definir etiquetas, realizar anotaciones manuales mediante rectángulos, validar dichas anotaciones, entrenar modelos de detección de objetos e inferir resultados automáticos.

A lo largo del proyecto se han alcanzado los siguientes objetivos:

- Se ha diseñado e implementado una interfaz intuitiva y funcional con Angular, basada en Material Design, que facilita la experiencia de usuario en todas las fases del proceso de anotación.
- Se ha construido un backend en Flask que expone una API REST para gestionar proyectos, imágenes, etiquetas, anotaciones y entrenamiento del modelo.
- Se ha incorporado un sistema de almacenamiento estructurado para proyectos, con directorios separados para imágenes validadas y no validadas, anotaciones y modelo entrenado.

- Se han implementado funciones para entrenar modelos de detección de objetos a partir de anotaciones manuales y exportar los resultados en diferentes formatos (`YOLO`, `COCO`, `PascalVOC`).
- Se ha creado una funcionalidad de inferencia automática sobre nuevas imágenes, permitiendo evaluar la calidad del modelo entrenado.

La herramienta cumple satisfactoriamente los objetivos funcionales y técnicos definidos en la fase inicial del proyecto, destacando su modularidad, escalabilidad y facilidad de uso.

Además, el uso de la metodología **ASAP** ha sido clave para mantener una organización constante, permitiendo detectar problemas a tiempo, establecer objetivos alcanzables por sprint y realizar reuniones de seguimiento que han favorecido el avance continuo del proyecto. La flexibilidad de la metodología también ha permitido adaptar el alcance del desarrollo según las prioridades y el tiempo disponible.

5.1.2. Perspectiva personal

Desde mi punto de vista personal, este proyecto ha sido un reto muy grande pero también una experiencia súper enriquecedora. Aunque ya tenía algo de experiencia en desarrollo web, nunca me había enfrentado a crear una aplicación tan completa, con backend, frontend, gestión de anotaciones y, además, funciones relacionadas con entrenamiento de modelos.

Durante todo el desarrollo he aprendido muchísimo. He podido reforzar mis conocimientos en Angular, Flask y cómo se comunican entre sí a través de peticiones HTTP. También he trabajado con estructuras de datos, guardado de archivos, formularios, rutas con parámetros y el uso de observables en Angular, que al principio me costaban bastante. Además, me he enfrentado a problemas reales como la sincronización de datos, validaciones, mejoras en la interfaz o cómo mantener el código bien organizado. Todo esto me ha ayudado a desarrollar mejores prácticas y ser más ordenada programando.

Una de las partes que más he disfrutado ha sido poder diseñar toda la herramienta desde cero: decidir cómo iba a funcionar, cómo organizar las pantallas, elegir los colores, cuidar que fuera fácil de usar... Me ha encantado ir viendo cómo todo cobraba forma poco a poco y sentir que estaba construyendo algo mío, a mi manera.

Otra parte que me ha parecido muy interesante (aunque también complicada) ha sido trabajar con imágenes y anotaciones. Conseguir que se dibujaran los rectángulos en posiciones relativas, que se guardaran bien, se pudieran ocultar, borrar, exportar... y que todo esto funcionara también en las miniaturas ha sido todo un reto, pero al final ha merecido muchísimo la pena.

5.2. Trabajo futuro

Existen diversas líneas de mejora que podrían explorarse en versiones futuras de la herramienta:

- **Historial de cambios en anotaciones**: incluir la posibilidad de deshacer y rehacer acciones realizadas sobre las anotaciones.
- **Soporte multiusuario**: permitir que diferentes usuarios puedan trabajar sobre los mismos proyectos con sesiones autenticadas.
- **Mejoras en la inferencia**: mostrar la precisión, confianza o índice de acierto del modelo sobre las predicciones automáticas.
- **Entrenamiento en segundo plano**: delegar el proceso de entrenamiento a un microservicio o contenedor independiente que permita no bloquear la aplicación mientras se entrena.
- **Notificaciones y sistema de logs**: proporcionar avisos al usuario sobre operaciones completadas o errores detectados, y mantener un registro del uso de la herramienta.
- **Anotaciones con otras formas**: extender el sistema de anotación para permitir el uso de polígonos, puntos o líneas además de rectángulos.
- **Integración con datasets externos**: permitir importar anotaciones desde conjuntos de datos en formatos estándares para su revisión o modificación.
- **Evaluación automática del modelo**: implementar métricas como mAP o IoU sobre un conjunto de validación para valorar el rendimiento real del modelo entrenado.

Estas mejoras contribuirían a dotar de mayor robustez, funcionalidad y aplicabilidad real a la herramienta, ampliando su uso tanto en contextos educativos como en entornos profesionales de anotación de datos visuales.

Parte III

Apéndices

Apéndice A

Manual de Instalación

Este apéndice describe los pasos necesarios para instalar y ejecutar la aplicación web desarrollada, compuesta por un **frontend en Angular** y un **backend en Flask (Python)**. También se indican los prerrequisitos, versiones de librerías utilizadas y recomendaciones para su correcto despliegue en un entorno local. El sistema operativo utilizado durante el desarrollo ha sido **Windows 11**.

A.1. Requisitos previos

- Python 3.10 o superior
- Node.js (v18.x o superior)
- Angular CLI (v16.x o superior)
- pip y virtualenv para gestión de entornos en Python

A.2. Instalación del backend (Flask)

1. Instalar Python desde su página oficial: `https://www.python.org/`
2. Crear un entorno virtual:

```
python -m venv venv
venv\Scripts\activate
```

3. Instalar las librerías necesarias ejecutando:

```
pip install -r requirements.txt
```

Donde `requirements.txt` contiene:

- Flask==2.3.2
- flask-cors==3.0.10
- numpy==1.26.4
- opencv-python==4.9.0.80
- matplotlib==3.7.1
- torch==2.2.1
- torchvision==0.17.1
- icevision==0.12.0
- fastai==2.7.14

4. Lanzar el servidor con:

```
python app.py
```

5. Si todo está correcto, Flask ejecutará el backend en `http://localhost:5000/`

A continuación se muestra una captura de la interfaz de ejecución del backend (Flask), como comprobación de que la aplicación se ha desplegado correctamente en local.

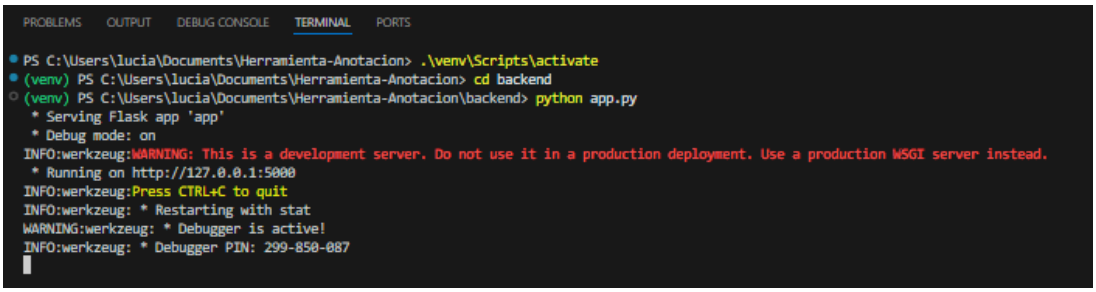


Figura A.1: Ejecución correcta del servidor Flask.

A.3. Instalación del frontend (Angular)

1. Instalar Node.js desde `https://nodejs.org`
2. Instalar Angular CLI globalmente:

```
npm install -g @angular/cli
```

3. Navegar a la carpeta `frontend/`:

```
cd frontend
npm install
```

4. Iniciar la aplicación con:

```
ng serve
```

5. Angular ejecutará el frontend en `http://localhost:4200/`

A continuación se muestra una captura de la interfaz de ejecución del frontend (Angular), como comprobación de que la aplicación se ha desplegado correctamente en local.

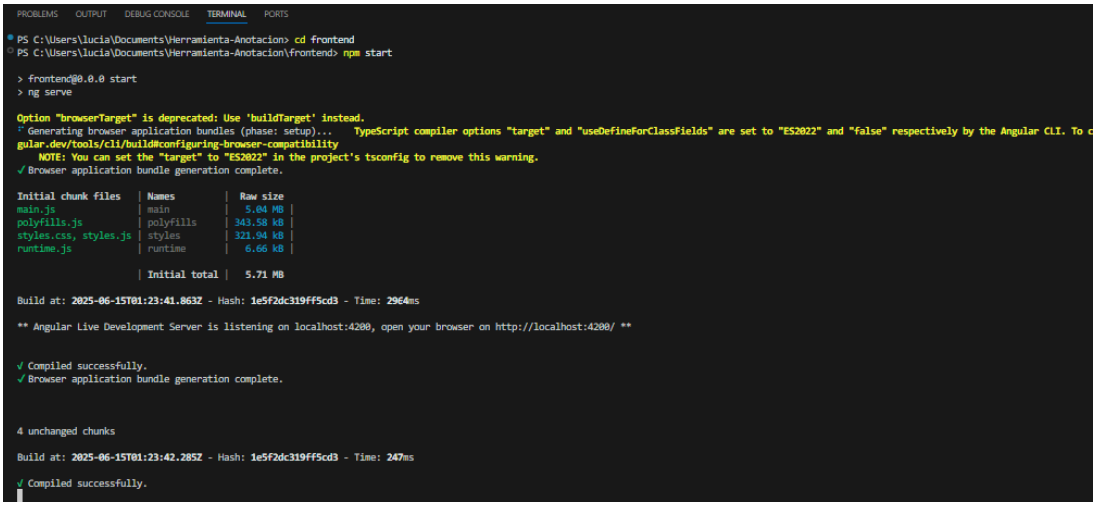


Figura A.2: Compilación exitosa del frontend en Angular.

A.4. Configuración del entorno de desarrollo

Para trabajar cómodamente en el desarrollo de esta aplicación, se recomienda configurar un entorno de desarrollo adecuado con las siguientes herramientas:

- **Visual Studio Code (VSCode)** como editor principal. Algunas extensiones útiles son:
  - Python (de Microsoft), para resaltar sintaxis y ejecutar scripts fácilmente.
  - Angular Language Service, para facilitar el autocompletado y la navegación en Angular.
  - ESLint y Prettier, para mantener un estilo de código limpio y consistente.
  - GitLens, para una gestión avanzada de control de versiones.
- **Postman** para probar las rutas del backend, especialmente útil durante el desarrollo del API REST.
- **Git** como sistema de control de versiones, recomendable para trabajar de forma ordenada y segura.

A.5. Notas finales y posibles errores

Durante el despliegue o uso de la aplicación pueden surgir ciertos problemas comunes. A continuación se recopilan algunas recomendaciones finales y soluciones habituales:

- Asegúrate de que el **backend esté corriendo antes de lanzar el frontend**, ya que este realiza llamadas a la API.
- Si Angular no puede acceder al backend, revisar las **políticas CORS** en Flask. Se recomienda usar `flask-cors`.
- Evita usar **nombres de carpetas con espacios o caracteres especiales** (como acentos o eñes), ya que pueden causar errores en rutas.
- Si el `ng serve` falla, asegúrate de que los puertos no estén ocupados. Angular suele usar el puerto 4200 y Flask el 5000.
- Si el servidor de Flask no reconoce una librería, revisa que esté instalada en el entorno virtual correspondiente.
- Se recomienda mantener el navegador actualizado y usar Google Chrome o Firefox para asegurar compatibilidad total.
- Si tras un cambio no se reflejan los resultados, intenta limpiar la caché del navegador o reiniciar el servidor.

Apéndice B

Manual de Usuario

Este manual está dirigido a los usuarios que deseen utilizar la herramienta web desarrollada para la gestión de anotaciones en imágenes y el posterior entrenamiento de modelos de detección de objetos.

La interfaz ha sido diseñada para ser intuitiva y fácil de usar, incluso para usuarios sin experiencia técnica. A continuación, se detallan todas las funcionalidades de la aplicación, acompañadas de capturas explicativas.

B.1. Pantalla de inicio

Nada más acceder a la aplicación, se muestra la pantalla de inicio, que permite al usuario gestionar sus proyectos de anotación.

- En la parte izquierda se encuentra el botón **Crear Nuevo Proyecto**, que abre un formulario donde se pueden definir el nombre, etiquetas y parámetros del modelo.
- En la parte derecha aparece el listado de **Proyectos Existentes**, mostrados como tarjetas con su nombre. También se incluye un buscador para filtrar por nombre.

Figura B.1: Pantalla principal con la opción de crear y seleccionar proyectos.

B.2. Crear un nuevo proyecto

Al pulsar sobre el botón **Crear Nuevo Proyecto** en la pantalla principal, el usuario accede a un formulario que permite definir un nuevo proyecto de anotación.

- **Nombre del proyecto**: Campo obligatorio. Se utilizará como identificador único del proyecto, tanto a nivel visual como en la estructura de carpetas.
- **Descripción del proyecto**: Campo opcional donde se puede incluir información adicional sobre el propósito del proyecto o el tipo de imágenes que contiene.
- Una vez rellenados los campos, se pulsa el botón **Crear** para generar automáticamente la estructura de carpetas asociada al proyecto.

Figura B.2: Formulario para la creación de un nuevo proyecto.

B.3. Gestión de imágenes sin contenido

Cuando se accede a un proyecto recién creado, la interfaz muestra un aviso indicando que no se ha cargado ninguna imagen.
En esta pantalla se encuentran las siguientes funcionalidades:

- **Importar imágenes**: Botón que permite seleccionar y cargar múltiples imágenes desde el equipo. Al cargarlas, se almacenan en la carpeta del proyecto correspondiente.
- **Gestión de etiquetas**: A la derecha se encuentra un campo para añadir etiquetas que luego se usarán durante el proceso de anotación. Las etiquetas añadidas se guardan automáticamente en el fichero `tags.json`.
- **Cabecera de navegación**: Incluye iconos para volver a la pantalla de inicio, acceder al menú de exportación, editar el nombre del proyecto o eliminarlo por completo.

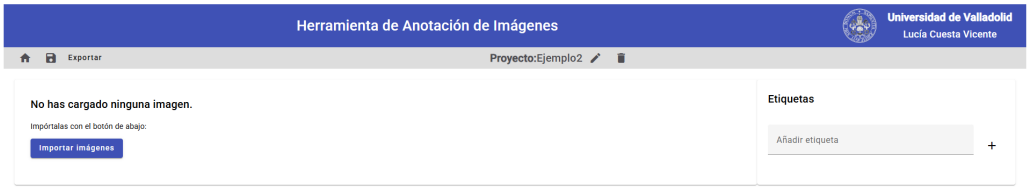


Figura B.3: Pantalla de gestión de imágenes sin imágenes cargadas.

B.4. Gestión de imágenes con contenido

Una vez importadas las imágenes en el proyecto, el sistema muestra automáticamente una cuadrícula con las miniaturas de todas las imágenes cargadas. Esta vista es clave para la gestión y selección previa al proceso de anotación.

Las funcionalidades disponibles en esta pantalla son:

- **Visualización en miniatura**: Permite una visión rápida de todas las imágenes cargadas en el proyecto, tanto validadas como no validadas. La pestaña superior permite alternar entre ambas vistas.
- **Botón "Importar imágenes"**: Permite añadir nuevas imágenes al proyecto en cualquier momento.
- **Campo de etiquetas**: Situado a la derecha, permite añadir nuevas etiquetas personalizadas que luego serán seleccionables durante la fase de anotación.
- **Acceso a anotación**: Al hacer clic sobre una miniatura, se redirige a la pantalla de anotación de dicha imagen.

Figura B.4: Pantalla del gestor de imágenes tras importar contenido.

B.5. Anotación de imágenes

Al hacer clic sobre una imagen en el gestor, se accede a la pantalla de anotación. Esta interfaz se divide en tres secciones claramente diferenciadas:

- **Columna izquierda - Otras Imágenes**: Muestra una lista en miniatura de las imágenes del proyecto para facilitar la navegación rápida entre ellas.
- **Centro - Imagen seleccionada**: Área principal donde se visualiza la imagen seleccionada a tamaño completo. Aquí es donde el usuario puede dibujar los rectángulos de anotación directamente con el ratón. Una vez definido el área, se abrirá un cuadro de diálogo para seleccionar una etiqueta.
- **Columna derecha - Anotaciones**: Lista con todas las anotaciones existentes sobre la imagen actual. Se pueden eliminar, mostrar/ocultar individualmente o editar.

En la parte inferior se encuentran los botones de navegación:

- **Anterior / Siguiente**: Permiten cambiar de imagen rápidamente sin volver a la vista general.
- **Botón de Validación (✓)**: Marca la imagen como validada, moviéndola a la carpeta correspondiente dentro del proyecto.

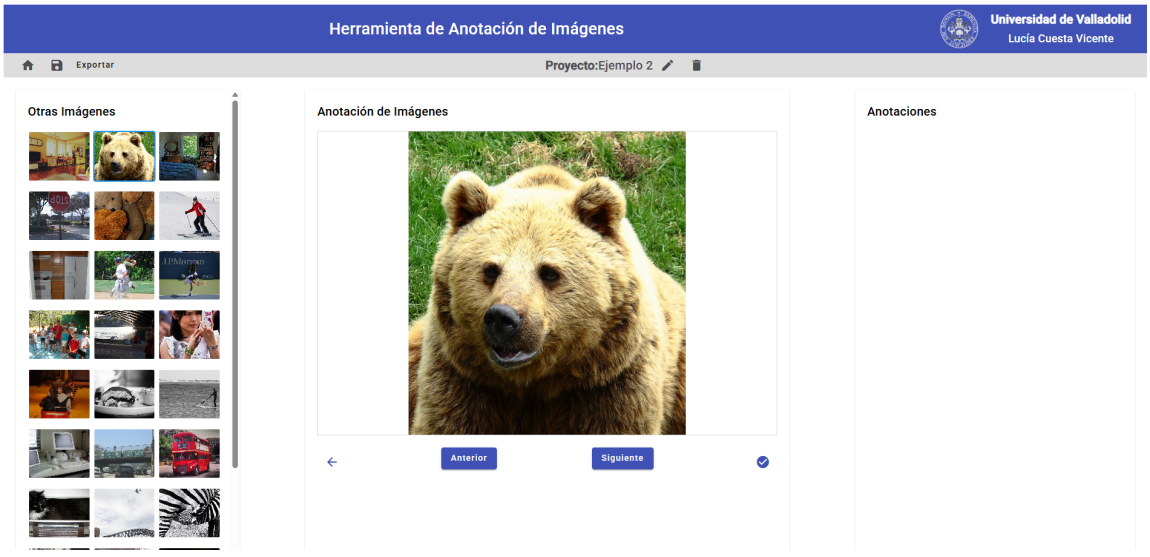


Figura B.5: Pantalla de anotación de imágenes con navegación y panel lateral.

B.6. Anotación con etiqueta aplicada

Una vez el usuario ha dibujado un rectángulo sobre la imagen, se abre automáticamente un cuadro de diálogo que permite seleccionar la etiqueta que se desea asignar.

Tras confirmar la etiqueta, esta se muestra directamente sobre la imagen (parte superior del rectángulo) y se añade a la lista de anotaciones en el panel lateral derecho. Desde ahí se puede:

- Alternar la visibilidad de cada anotación con el icono del ojo.
- Eliminarla mediante el icono de la papelera.
- Visualizar el nombre de la etiqueta de forma clara y ordenada.

Esta funcionalidad permite tener un control visual e inmediato del proceso de etiquetado, facilitando una experiencia de usuario clara e intuitiva.

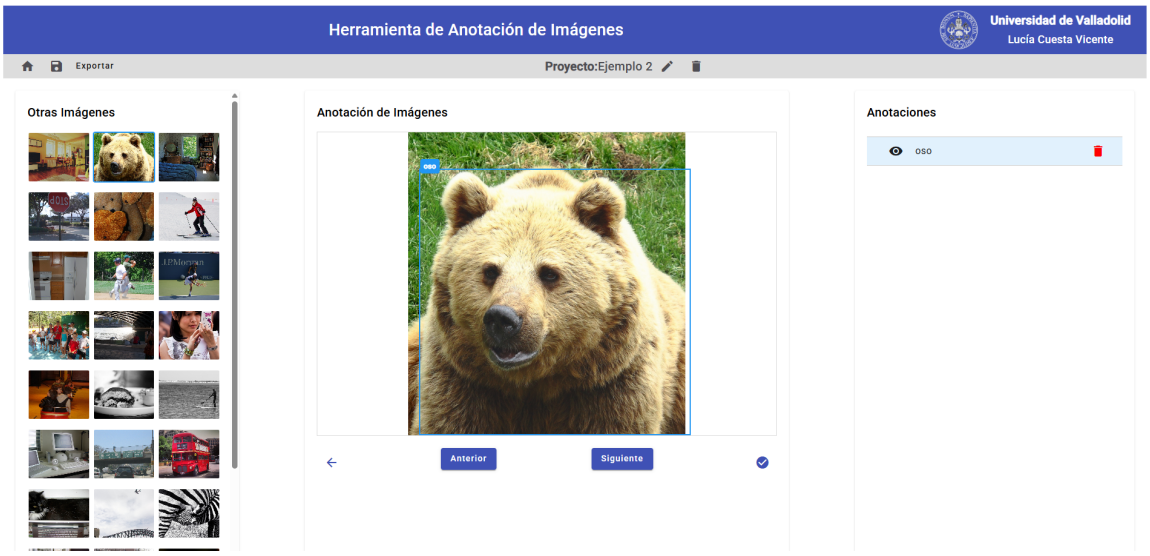


Figura B.6: Anotación con etiqueta aplicada y visible en el panel lateral.

Bibliografía

[1] Airctic. *IceVision: A Framework for Object Detection and Image Segmentation*. Consultado en junio de 2025. https://airctic.com/. 2021.

[2] Armin Ronacher et al. *Flask*. Consultado en junio de 2025. https://flask.palletsprojects.com/. 2024.

[3] Glenn Jocher et al. *Ultralytics YOLOv5*. https://github.com/ultralytics/yolov5. 2023.

[4] John D. Hunter et al. *Matplotlib: Python Plotting*. Consultado en junio de 2025. https://matplotlib.org/. 2020.

[5] Joseph Redmon et al. *YOLO: You Only Look Once*. Consultado en junio de 2025. https://pjreddie.com/darknet/yolo/. 2020.

[6] Travis E. Oliphant et al. *NumPy*. Consultado en junio de 2025. https://numpy.org/. 2020.

[7] Atlassian. *Trello*. Consultado en junio de 2025. https://trello.com/. 2024.

[8] Microsoft Corporation. *Microsoft Teams*. Consultado en junio de 2025. https://www.microsoft.com/es-es/microsoft-teams/group-chat-software. 2024.

[9] Microsoft Corporation. *Visual Studio Code*. Consultado en junio de 2025. https://code.visualstudio.com/. 2024.

[10] Mark Everingham et al. «The Pascal Visual Object Classes (VOC) challenge». En: *International journal of computer vision* 88.2 (2010), págs. 303-338.

[11] Inc. GitHub. *GitHub*. Consultado en junio de 2025. https://github.com/. 2024.

[12] Glassdoor. *Salary Data for Spain*. https://www.glassdoor.es. Consultado en 2025.

[13] Ian Goodfellow, Yoshua Bengio y Aaron Courville. *Deep Learning*. MIT Press, 2016. ISBN: 978-0262035613. URL: https://www.deeplearningbook.org/.

[14] Google LLC. *Angular*. Consultado en junio de 2025. https://angular.io/. 2024.

[15] Andreas Holzinger. «Interactive machine learning for health informatics: when do we need the human-in-the-loop?» En: *Brain Informatics* 3.2 (2016), págs. 119-131. DOI: 10.1007/s40708-016-0042-6.

[16] Labelbox. *Labelbox*. https://labelbox.com/. 2024.

[17] Yann LeCun, Yoshua Bengio y Geoffrey Hinton. «Deep learning». En: *Nature* 521.7553 (2015), págs. 436-444. DOI: 10.1038/nature14539.

[18] Tsung-Yi Lin et al. «Microsoft COCO: Common Objects in Context». En: *European Conference on Computer Vision (ECCV)*. Springer. 2014, págs. 740-755. DOI: 10.1007/978-3-319-10602-1_48.

[19] Tsung-Yi Lin et al. «Microsoft COCO: Common objects in context». En: *European conference on computer vision (ECCV)*. Springer. 2014, págs. 740-755.

[20] Wei Liu et al. «Deep Learning for Generic Object Detection: A Survey». En: *International Journal of Computer Vision* 128 (2020), págs. 261-318. DOI: 10.1007/s11263-019-01247-4.

[21] Wei Liu et al. «SSD: Single shot multibox detector». En: *European conference on computer vision (ECCV)*. Springer. 2016, págs. 21-37.

[22] Overleaf Ltd. *Overleaf*. Consultado en junio de 2025. https://www.overleaf.com/. 2024.

[23] Miguel A. Martínez-Prieto et al. «Una metodología basada en prácticas ágiles para la realización de Trabajos Fin de Grado». En: *Actas de las JENUI* (2023).

[24] Meta AI Research. *PyTorch*. Consultado en junio de 2025. https://pytorch.org/. 2024.

[25] Microsoft. *VoTT: Visual Object Tagging Tool*. https://github.com/microsoft/VoTT. GitHub repository. 2019.

[26] CVAT - OpenCV. *CVAT Documentation*. https://opencv.github.io/cvat/. 2024.

[27] OpenCV.org. *OpenCV: Open Source Computer Vision Library*. Consultado en junio de 2025. https://opencv.org/. 2020.

[28] PayScale. *Average Salaries in Spain*. https://www.payscale.com. Consultado en 2025.

[29] *Real Decreto 43/2015, de 2 de febrero, por el que se modifica el Real Decreto 1393/2007, de 29 de octubre, por el que se establece la ordenación de las enseñanzas universitarias oficiales*. BOE-A-2015-1176. 2015.

[30] RectLabel. *RectLabel*. https://rectlabel.com/. 2024.

[31] Joseph Redmon et al. «You only look once: Unified, real-time object detection». En: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*. 2016, págs. 779-788.

[32] Shaoqing Ren et al. «Faster R-CNN: Towards real-time object detection with region proposal networks». En: *Advances in neural information processing systems (NeurIPS)*. 2015, págs. 91-99.

[33] Roboflow. *Roboflow*. https://roboflow.com/. 2024.

[34] Patricia Rodriguez, Luis Herranz y Miroslaw Bober. «Dataset Annotation: Challenges, Recommendations and Future Directions». En: *Computer Vision and Image Understanding* 213 (2021), pág. 103295.

[35] Yerin Roh, Geon Heo y Sun Whang. «A survey on data collection for machine learning: a big data—AI integration perspective». En: *IEEE Transactions on Knowledge and Data Engineering* 33.4 (2021), págs. 1328-1347. DOI: 10.1109/TKDE.2019.2946162.

[36] Label Studio. *Label Studio Documentation*. https://labelstud.io/. 2024.

[37] SuperAnnotate. *SuperAnnotate*. https://www.superannotate.com/. 2024.

[38] Richard Szeliski. *Computer Vision: Algorithms and Applications*. 2nd. Springer, 2022. ISBN: 978-3030343699. URL: https://szeliski.org/Book/.

[39] The Pandas Development Team. *Pandas: Python Data Analysis Library*. Consultado en junio de 2025. https://pandas.pydata.org/. 2020.

[40] Tzutalin. *LabelImg: Label Object Bounding Boxes in Images*. https://github.com/tzutalin/labelImg. GitHub repository. 2015.

[41] University of Oxford Visual Geometry Group. *VGG Image Annotator (VIA)*. https://www.robots.ox.ac.uk/~vgg/software/via/. 2024.

[42] Zeming Zhao et al. «Object detection with deep learning: A review». En: *IEEE transactions on neural networks and learning systems* 30.11 (2019), págs. 3212-3232.