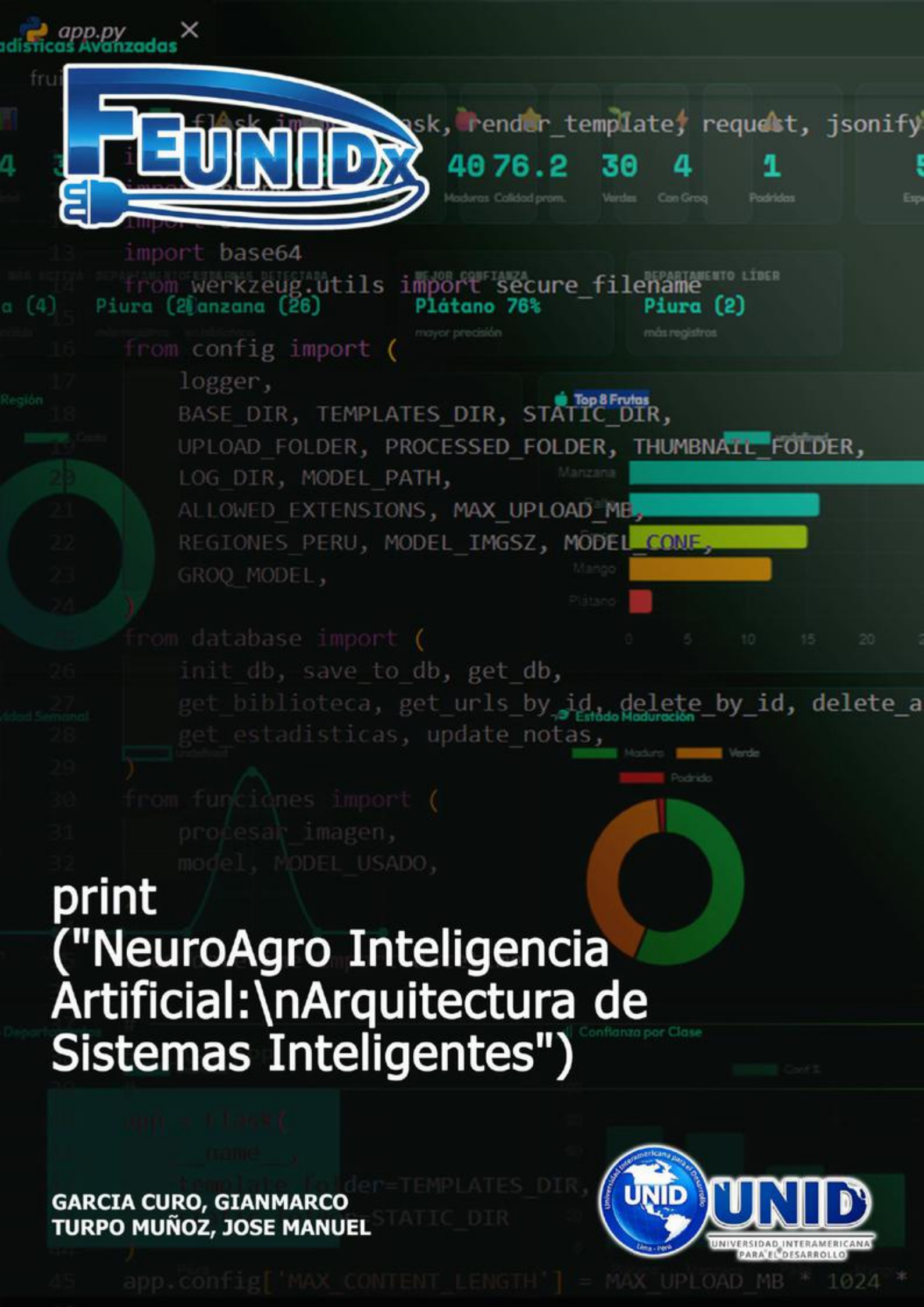


print ("NeuroAgro Inteligencia Artificial:\nArquitectura de Sistemas Inteligentes")

GARCIA CURO, GIANMARCO
TURPO MUÑOZ, JOSE MANUEL

UNID
UNIVERSIDAD INTERAMERICANA PARA EL DESARROLLO

Neuroagro inteligencia artificial: Arquitectura de sistemas inteligentes

© **Gianmarco Garcia Curo**
https://orcid.org/0000-0001-6685-3207
Instituto Continental, Escuela Superior Continental, Perú

Jose Manuel Turpo Muñoz
https://orcid.org/0009-0000-0213-8853
Instituto Continental, Escuela Superior Continental, Perú

EDITADA POR:

© UNIVERSIDAD INTERAMERICANA PARA EL DESARROLLO (UNIDX) - FONDO EDITORIAL "EXPONENCIAL"

DIRECCIÓN: AV. BOLIVIA NRO. 626 (A 2 CDRAS DE AV. ALFONSO UGARTE) BREÑA, LIMA, LIMA, PERÚ.

ISNI: 0000 0004 6101 3964
https://isni.org/isni/0000000461013964
Name: Inter-American Development University
Universidad Interamericana para el Desarrollo
Location / Nationality: Peru Bolivar
Correo: fondoeditorial@unidx.edu.pe
Portal Web: https://unidx.edu.pe

Primera edición digital: Marzo del 2026
Libro digital disponible en: https://fondoeditorial.unidx.edu.pe

Hecho el depósito legal en La Biblioteca Nacional Del Perú N° 2026-03217
ISBN: 978-612-99367-1-0
DOI: https://doi.org/10.56275/ng33px32

REVISIÓN POR PARES CIEGOS APROBADO POR EL CONSEJO EDITORIAL DEL FONDO EDITORIAL "EXPONENCIAL".

LIBRO RESULTADO DE INVESTIGACIÓN Y CON REVISIÓN POR PARES CIEGOS.
SELLO EDITORIAL: FONDO EDITORIAL (978-612-99120)

LIMA - PERÚ
2026

ÍNDICE

Resumen 8

CAPÍTULO I 9

Planteamiento del Problema y Contexto de la Investigación 9

1. Introducción 10

1.1. Problema en el agro 11

1.2. Justificación científica y tecnológica 13

1.3. Objetivos 14

1.4. Alcance 14

CAPÍTULO II 16

Estado del Arte en Inteligencia Artificial Aplicada al Sector Agrícola 16

2. Estado del Arte 17

2.1. IA en agricultura 17

2.2. Modelos de visión (YOLO, CNN) 18

2.3. Sistemas similares 19

2.4. Brechas existentes 20

CAPÍTULO III 22

Fundamentos Teóricos de Sistemas Inteligentes y Visión por Computadora 22

3. Marco Teórico 23

3.1. Deep Learning 23

3.2. Detección de objetos 24

3.3. Procesamiento de imágenes 25

3.4. Sistemas inteligentes 26

CAPÍTULO IV 28

Metodología de Investigación y Diseño Experimental 28

4. Metodología de Investigación 29

4.1. Tipo: investigación aplicada y experimental 29

4.2. Dataset utilizado 29

4.3. Proceso de entrenamiento del modelo 30

4.4. Configuración 31

4.5. Validación del modelo 32

CAPÍTULO V 33

Entrenamiento y Optimización del Modelo de Inteligencia Artificial 33

5. Entrenamiento y Optimización del Modelo 34

5.1. Preparación de datos 34

5.2. Etiquetado 41

5.3. Entrenamiento YOLO 42

5.4. Ajuste de hiperparámetros 47

5.5. Resultados del entrenamiento 50

CAPÍTULO VI 53

Diseño de la Arquitectura del Sistema Inteligente 53

6. Diseño del Sistema Inteligente 54

6.1. Arquitectura general 54

6.2. Integración del modelo entrenado 55

6.3. Flujo de procesamiento 56

CAPÍTULO VII 58

Desarrollo e Implementación del Sistema 58

7. Desarrollo del Sistema 59

7.1. Backend (Flask) 59
7.2. Procesamiento de imágenes 60
7.3. Integración con IA adicional 61
7.4. Pipeline interno de procesamiento 62
CAPÍTULO VIII 64
Modelo de Datos y Gestión de Información Inteligente 64
8. Modelo de Datos y Gestión de Información 65
8.1. Diseño de base de datos 65
8.2. Variables inteligentes 68
CAPÍTULO IX 71
Evaluación Experimental y Análisis de Resultados 71
9. Evaluación Experimental 72
9.1. Métricas: precisión, recall, mAP 72
9.2. Pruebas con imágenes reales 74
9.3. Análisis de resultados 87
9.4. Análisis sobre la IA 88
CAPÍTULO X 92
Aporte Científico, Tecnológico e Impacto del Sistema 92
10. Aporte Científico y Tecnológico 93
10.1. Desarrollo de un modelo de detección de objetos adaptado al contexto agrícola 93
10.2. Sistema inteligente integrado 94
10.3. Automatización del análisis agrícola 94
10.4. Impacto potencial 95
11. Discusión 97

11.1. Interpretación técnica 97
11.2. Limitaciones 98
11.3. Comparación con otros estudios 99
12. Conclusiones 101
12.1. Validación del modelo 101
12.2. Validación del sistema 101
12.3. Cumplimiento de objetivos 102
13. Trabajo Futuro 103
13.1. Nuevos datasets 103
13.2. Mejora del modelo 103
13.3. Escalabilidad 104
14. Referencias 105
15. Anexo 108
Estructura 108
Instalación de Docker Desktop 108
Instalación de Label Studio usando Docker 111
Instalación de Python en el sistema 113
Importar imágenes para el dataset 119
Seleccionar el formato YOLO 121
Descargar el dataset exportado 122
Código: 125
app.py 125
templates/index.html 131
config.py 164

funciones.py 167

database.py 176

entrenamiento/entrenamiento.py 183

entrenamiento/extraer_y_mantener2.py 185

entrenamiento/preparar.py 186

UNID
UNIVERSIDAD INTERAMERICANA PARA EL DESARROLLO

Resumen

La presente investigación aborda el diseño, desarrollo e implementación de un sistema inteligente orientado a la evaluación automatizada de productos agrícolas mediante técnicas avanzadas de visión por computadora y aprendizaje profundo. El presente estudio adopta un enfoque aplicado y de carácter experimental, orientado a disminuir la subjetividad inherente a los métodos tradicionales de inspección visual. Dichos métodos, si bien son de uso extendido, presentan limitaciones relacionadas con la consistencia de los resultados, así como con su escalabilidad y reproducibilidad en distintos contextos. En respuesta a esta problemática, se plantea el desarrollo de un sistema basado en detección de objetos mediante arquitecturas del tipo YOLO, el cual ha sido ajustado para su aplicación en entornos agrícolas. Este proceso de adaptación permitió optimizar el rendimiento del modelo durante la inferencia en condiciones reales del entorno agrícola. La solución presentada se realiza siguiendo un enfoque cliente-servidor basado en Flask, permitiendo gestionar el flujo completo del procesamiento, desde la adquisición de imágenes hasta la salida estructurada. Respecto al pipeline, el flujo va organizado en etapas de preprocesamiento orientadas a la normalización de la imagen y, posteriormente, se encuentra la fase correspondiente a la inferencia mediante el modelo entrenado siguiendo esta, se encuentra la etapa de análisis. En esta etapa las detecciones se transforman en variables derivadas que permiten describir el comportamiento del sistema de manera más resumida. A partir de estas variables es más fácil interpretar los resultados, especialmente desde un punto de vista operativo. Para la validación experimental se han utilizado registros obtenidos en condiciones de prácticas reales. En total, se registraron 71 detecciones a lo largo de cuatro ejecuciones del sistema, con una confianza media de 0,5946. El índice de calidad del arreglo se sitúa entre 65 y 85, evidenciando una generación consistente de información estructurada para la obtención de resultados. La propuesta constituye un sistema basado en detección de objetos inteligente que permite la gestión automática del análisis agronómico, contribuyendo a mejorar la objetividad, reproducibilidad y escalabilidad frente a antiguos métodos.

Palabras clave: Inteligencia Artificial, Aprendizaje profundo, Visión por computadora, Detección de objetos.

CAPÍTULO I

Planteamiento del Problema y Contexto de la Investigación

1. Introducción

El sector agrícola representa un componente estratégico en la dinámica económica y social de los países en desarrollo, particularmente en contextos donde la diversidad productiva y la variabilidad climática condicionan los procesos de producción y comercialización. No obstante, uno de los principales desafíos que enfrenta este sector radica en la limitada incorporación de tecnologías avanzadas para la evaluación objetiva de la calidad de los productos agrícolas, lo cual impacta directamente en la eficiencia de la cadena de suministro y la competitividad de los productores.

Conforme a la tradición, la clasificación de los productos agrícolas ha estado basada en el uso de métodos empíricos en los que la evaluación de variables como el grado de maduración, la apariencia y el grado de los defectos se basan estrictamente en criterios subjetivos. Este enfoque presenta limitaciones en términos de reproducibilidad, precisión y estandarización de los resultados, lo que afecta la toma de decisiones y la consistencia de los resultados en la cadena productiva, lo que genera inconsistencias en la valoración del producto y limita la toma de decisiones en instancias clave como la cosecha, almacenamiento y comercialización. En este escenario, la irrupción de la Inteligencia Artificial, en particular las técnicas de aprendizaje profundo aplicadas a la visión por computadora, ha permitido el desarrollo de modelos capaces de extraer características relevantes a partir de datos visuales con altos niveles de exactitud. Los modelos de detección de objetos, en especial los que utilizan arquitecturas YOLO (You Only Look Once), se han vuelto ampliamente utilizados en los últimos años, principalmente porque permiten trabajar en tiempo real. Esto se debe a que, en lugar de separar procesos, el modelo realiza al mismo tiempo la localización y la clasificación de los objetos. En la práctica, este enfoque permite optimizar el tiempo de inferencia y mejorar la eficiencia computacional del sistema, lo que resulta crítico en aplicaciones que requieren respuesta en tiempo real.

A partir de ello, en esta investigación se plantea el desarrollo de una solución basada en Inteligencia Artificial para el análisis visual en el ámbito agrícola. El enfoque no se limita al modelo, sino que también considera su implementación

dentro de un sistema funcional, que pueda utilizarse en condiciones reales. Es decir, se consideró tanto el proceso de entrenamiento como su integración dentro de una solución funcional.

El sistema desarrollado, denominado NeuroAgro Inteligencia Artificial, permite procesar imágenes y detectar ciertos patrones relacionados con el estado del producto. Con la información obtenida, se producen resultados organizados que contribuyen en la toma de decisiones. El sistema no solo se convierte en detector, sino que, además, calcula algunos indicadores como niveles de calidad, estimaciones del valor comercial, recomendaciones técnicas, otorgándole un carácter aplicado orientado al apoyo en la toma de decisiones.

La metodología que utiliza el estudio es la metodología de investigación aplicada con diseño experimental. La metodología comprende las etapas de recolección de datos, preprocesamiento, entrenamiento, ajuste de hiperparámetros y validación mediante métricas cuantitativas, siguiendo un enfoque experimental reproducible. Esto permitió evaluar el comportamiento del modelo en condiciones cercanas a escenarios reales.

Finalmente, más allá del desarrollo del sistema, el trabajo también busca mostrar que este tipo de tecnologías puede aplicarse en el sector agrícola. Los resultados obtenidos en este procedimiento aportan evidencias de la posibilidad de mejora de ciertos procedimientos tradicionales e indican que existe oportunidad de avanzar hacia otras soluciones técnicas más eficientes y precisas en esta materia.

El estudio da lugar a futuras extensiones llevadas a la mejora del modelo, la incorporación de nuevas fuentes de datos o de tecnologías complementarias, lo cual permitirá establecer un marco de referencia con el que se podrán desarrollar otras soluciones inteligentes orientadas al sector agrario.

1.1. Problema en el agro

El sector agrícola enfrenta desafíos significativos asociados a la evaluación de la calidad de los productos, la cual continúa realizándose mayoritariamente mediante inspección visual humana. Este enfoque introduce un componente elevado de subjetividad, dado que los criterios

de clasificación dependen de la experiencia del evaluador, generando inconsistencias en la determinación del estado del producto y afectando la reproducibilidad de los resultados. En este contexto, la carencia de estandarización de los procesos de evaluación de la cadena nacional de carácter productiva imposibilita una correcta toma de decisiones.

La literatura científica reciente ha revelado que los modelos de evaluación tradicionales presentan carencias a nivel de potencia y escalabilidad. En concreto, existen evidencias de que la evaluación manual puede ser incorrecta y que no permite un seguimiento en tiempo real del comportamiento dinámico de los entornos (Xiao et al., 2023). De la misma forma, dentro de los sistemas agrícolas actuales existe la necesidad de soluciones capaces de procesar grandes volúmenes de información en tiempo real, algo que no se puede cubrir adecuadamente desde el modelo tradicional.

Por otra parte, la falta de automatización de las herramientas que permiten evaluar con características fundamentales la producción de frutas y hortalizas en fincas agrícolas puede generar pérdidas significativas de productos agrícolas, sobre todo en la etapa de poscosecha. El estado del producto mal clasificado puede causar pérdida de calidad, disminución de su valor de venta y pérdida de recursos. Dentro de este modelo de evaluación automática se ha encontrado que el uso de modelos de detección hace que la evaluación en este tipo de productos aumente la precisión de la detección del producto y su estado, reduciendo de esta forma los errores asociados a la inspección manual (Zhang et al., 2022).

Por otra parte, estudios recientes destacan que la detección automatizada de productos agrícolas es un componente crítico para la agricultura de precisión, especialmente en entornos complejos donde existen condiciones de oclusión, variabilidad de iluminación y diversidad de especies (Karacaoglu et al., 2025). La incapacidad de los métodos tradicionales para abordar estas condiciones evidencia la necesidad de

soluciones basadas en Inteligencia Artificial que permitan una evaluación objetiva, consistente y escalable.

En consecuencia, la combinación de subjetividad en la evaluación y pérdidas significativas en la cadena productiva constituye un problema relevante que requiere la incorporación de tecnologías avanzadas orientadas a la automatización del análisis agrícola.

1.2. Justificación científica y tecnológica

El avance de la Inteligencia Artificial, particularmente en el campo del aprendizaje profundo, ha permitido el desarrollo de modelos capaces de analizar información visual con un alto grado de precisión y eficiencia. En el ámbito agrícola, estas tecnologías han sido ampliamente utilizadas para automatizar tareas de detección, clasificación y monitoreo de cultivos, superando las limitaciones de los métodos tradicionales.

Los modelos de detección de objetos aplicados a arquitecturas YOLO han demostrado ser muy apropiados para el sector agrícola, lo cual se debe a su capacidad para hacer inferencias en tiempo real y de forma precisa. Estudios recientes han mostrado buenos valores de rendimiento para llevar a cabo la detección de productos típicos de la agricultura, obteniendo más del 90 % en métricas como el precision y el mAP (Weng et al., 2025). Al mismo tiempo, los estudios comparativos han dejado entrever cómo modelos como el YOLOv6 pueden alcanzar valores de precisión próximos al 99 %, logrando un adecuado equilibrio entre velocidad y precisión (Kamat et al., 2025).

Desde una perspectiva científica, el uso del aprendizaje profundo permite la extracción automática de las características que resultan relevantes a partir de imágenes, lo que ayuda a identificar patrones complejos vinculados al estado del producto. En este sentido, las redes neuronales convolucionales han alcanzado niveles de desempeño comparables al humano en tareas específicas de reconocimiento de imágenes, constituyendo una base fundamental para el desarrollo de sistemas inteligentes (Xiao et al., 2023).

Desde un punto de vista tecnológico, la integración de estos modelos dentro de sistemas operativos permite avanzar hacia esquemas de agricultura de precisión, en los cuales la toma de decisiones se fundamenta en datos reales y que se procesan en tiempo real. Asimismo, la automatización de los procesos de evaluación permite promover la reducción de las pérdidas, la optimización de los recursos y una mejora de la calidad del producto final.

En este contexto, la presente investigación se justifica en la necesidad de diseñar e implementar un sistema inteligente que integre modelos de Inteligencia Artificial con una arquitectura tecnológica capaz de operar en entornos reales, contribuyendo tanto al avance científico como al desarrollo tecnológico en el sector agrícola.

1.3. Objetivos

Desarrollar un sistema basado en inteligencia artificial para la evaluación automatizada de productos agrícolas mediante visión por computadora.

Objetivos específicos

- ✓ Diseñar un modelo de aprendizaje profundo para la detección y clasificación de características visuales relevantes en productos agrícolas.
- ✓ Construir un conjunto de datos representativo para el entrenamiento y validación del modelo.
- ✓ Optimizar el modelo mediante el ajuste de hiperparámetros y la evaluación con métricas estándar como precisión, recall y mAP.
- ✓ Integrar el modelo desarrollado dentro de una arquitectura de sistema funcional.
- ✓ Evaluar el desempeño del sistema en entornos reales considerando eficiencia, precisión y aplicabilidad.

1.4. Alcance

La presente investigación se enfoca en el desarrollo de una solución tecnológica basada en Inteligencia Artificial para el análisis automatizado de productos agrícolas a partir de imágenes digitales. El alcance incluye

el diseño, entrenamiento y validación de modelos de detección basados en aprendizaje profundo, así como su integración en un sistema informático capaz de procesar información en tiempo real.

El sistema planteado puede determinar algunas características visuales importantes del producto estudiado, como el estado de la fruta y su condición, generando variables derivadas que facilitan la toma de decisiones en el ámbito agrícola.

En contraposición, el estudio, únicamente, se detiene en el análisis de variables visuales, sin incluir propiedades fisicoquímicas internas del producto ni aspectos externos como las condiciones climáticas. Además, la validación del modelo se realiza en entornos controlados, por lo que su implementación en otros contextos requerirá procesos adicionales de adaptación y ajuste

UNID
UNIVERSIDAD INTERAMERICANA PARA EL DESARROLLO

CAPÍTULO II

Estado del Arte en Inteligencia Artificial Aplicada al Sector Agrícola

2. Estado del Arte

2.1. IA en agricultura

En las últimas décadas, la incorporación de la Inteligencia Artificial en el sector agrícola ha experimentado un crecimiento sostenido, impulsado por la necesidad de optimizar los procesos productivos, mejorar la eficiencia en el uso de recursos y garantizar la sostenibilidad de los sistemas agrícolas. En este contexto, las técnicas de aprendizaje automático y aprendizaje profundo han permitido el desarrollo de modelos capaces de analizar grandes volúmenes de datos provenientes de diversas fuentes, como imágenes, sensores remotos y dispositivos de monitoreo.

La literatura más reciente indica que la IA se ha establecido como un aspecto fundamental en la agricultura de precisión, al hacer posible la generación de información a partir de datos objetivos e interpretados en tiempo real, destacando así el hecho de que el uso de modelos predictivos y sistemas inteligentes puede mejorar la toma de decisiones en actividades como, por ejemplo, el monitoreo del cultivo, la detección de enfermedades o la estimación del rendimiento agrícola (Botero-Valencia et al., 2025); por su parte, la combinación de tecnologías como los vehículos aéreos no tripulados (VANT) y sensores remotos con algoritmos de IA han logrado ampliar las capacidades de monitoreo agrícola gracias a la recopilación de información sobre el cultivo a diferentes escalas espacio-temporales (Buitrago Bolívar et al., 2024) y ha favorecido la automatización de procesos hasta hace poco manuales y ayudado a reducir la dependencia de la inspección directa, al tiempo que permite una estimación más precisa de variables clave.

No obstante, a pesar de estos avances, persisten desafíos relacionados con la heterogeneidad de los datos, la adaptación de los modelos a condiciones específicas y la integración de múltiples fuentes de información dentro de sistemas unificados. Estas limitaciones evidencian la necesidad de continuar desarrollando soluciones que permitan una

implementación más robusta y generalizable de la Inteligencia Artificial en el ámbito agrícola.

2.2. Modelos de visión (YOLO, CNN)

Dentro del campo de la Inteligencia Artificial aplicada a la agricultura, los modelos de visión por computadora han adquirido un papel fundamental en el análisis automatizado de imágenes. En particular, las redes neuronales convolucionales (CNN) y las arquitecturas basadas en YOLO (You Only Look Once) se han consolidado como las principales herramientas para tareas de detección, clasificación y segmentación de objetos agrícolas.

Las CNN poseen la capacidad de extraer características jerárquicas de las imágenes, permitiendo asociar patrones complejos con los estados o condiciones del producto. Este tipo de modelos ha alcanzado una aplicación relevante en la clasificación de cultivos, detección de enfermedades o análisis de calidad, logrando un alto nivel de precisión en múltiples casos.

En contraposición, los modelos YOLO permiten incluir en una única arquitectura los procesos de localización y clasificación, permitiendo realizar detección de objetos en tiempo real con alta eficiencia computacional, siendo esta una de sus características más importantes, sobre todo en el contexto agrario, donde se requiere del procesamiento de grandes volúmenes de información en condiciones cambiantes. Algunos estudios han indicado que la combinación de modelos YOLO y CNN puede haber dado lugar a resultados mucho más satisfactorios, tanto en precisión como en velocidad, sobre todo en aplicaciones de agricultura de precisión (Sasmoko et al., 2025).

Algunas investigaciones recientes indican que modelos basados en YOLOv8 pueden conseguir niveles de precisión altos en tareas de clasificación de productos agrícolas y su estado, poniendo de manifiesto el potencial para empaquetar procesos manuales (León et al., 2024).

De manera complementaria, el uso de técnicas como la transferencia de aprendizaje ha permitido mejorar el desempeño de estos modelos, facilitando su adaptación a nuevos conjuntos de datos y reduciendo los requerimientos de entrenamiento. Este enfoque ha contribuido significativamente a la implementación de soluciones más eficientes y escalables en el sector agrícola (Villalobos-Culqui et al., 2025).

2.3. Sistemas similares

El desarrollo de sistemas inteligentes aplicados a la agricultura ha dado lugar a múltiples soluciones orientadas a la automatización del análisis visual y la toma de decisiones. Comprenden, a su vez, modelos de visión por computador y arquitecturas tecnológicas que permiten su aplicación a ambientes reales, conformando una tendencia hacia la digitalización en el sector.

En el contexto de la agricultura de precisión se han desarrollado sistemas que integran imágenes mediante UAV y algoritmos de machine learning para el monitoreo de cultivaciones y detección de problemas. Estos sistemas han evidenciado una mejora notable respecto a los métodos tradicionales por permitir la detección temprana de problemas o la optimización de recursos (Buitrago Bolívar et al., 2024).

Se han implementado también sistemas de visión artificial para la automatización de la clasificación de productos agrícolas en función de su madurez empleando modelos basados en deep learning. En concreto, los enfoques que utilizan arquitecturas del tipo YOLOv8 han demostrado altas capacidades para detectar características visuales relevantes y en tiempo real, llevando así la automatización de procesos de clasificación normalmente vinculados a la inspección humana. Estos sistemas evidencian mejoras significativas en términos de precisión y consistencia, contribuyendo a la estandarización de criterios en la evaluación de la calidad del producto (León León et al., 2024).

Por otra parte, investigaciones recientes destacan el desarrollo de sistemas integrados que combinan Inteligencia Artificial, sensores y análisis de datos para mejorar la trazabilidad y eficiencia de la cadena de

suministro agrícola. Estos enfoques permiten una gestión más eficiente de los recursos y contribuyen a la reducción de pérdidas, aunque su implementación a gran escala aún enfrenta limitaciones tecnológicas y económicas (Ghazal et al., 2024).

2.4. Brechas existentes

A pesar de los avances en la aplicación de la Inteligencia Artificial en la agricultura, la literatura científica identifica diversas brechas que limitan su adopción efectiva en entornos reales.

Una de las limitaciones más notables guarda relación con la disponibilidad y calidad de los datos. Concretamente, los modelos de aprendizaje profundo requieren grandes volúmenes de datos etiquetados para su entrenamiento, lo cual supone una limitación importante dada la variabilidad de las condiciones ambientales, tipos de cultivo y situaciones de captura de imágenes (Sasmoko et al., 2025).

Así mismo, la capacidad de generalización de los modelos es otra limitación relevante. Muchos sistemas presentan un alto rendimiento en condiciones controladas, pero experimentan una disminución en su precisión cuando se enfrentan a entornos reales caracterizados por variaciones en iluminación, oclusión y complejidad del entorno agrícola (Botero-Valencia et al., 2025).

Otra brecha importante está relacionada con la integración de tecnologías. Si bien existen múltiples soluciones basadas en Inteligencia Artificial, muchas de ellas operan de manera aislada, sin formar parte de sistemas completos que permitan su aplicación práctica en la toma de decisiones. Esta deficiencia en la integración del conocimiento limita el verdadero impacto de cada uno de estos tipos de tecnologías sobre el sector agrícola.

Finalmente, identificamos la necesidad de desarrollar soluciones adecuadas y adaptadas a los contextos locales teniendo en cuenta factores específicos como el tipo de cultivo, las condiciones ambientales y la infraestructura tecnológica disponible. En este sentido, la

investigación en sistemas inteligentes aplicados a la agricultura todavía tiene un campo amplio de desarrollo en lo que se refiere a escalabilidad, accesibilidad y robustez de las soluciones que se han propuesto.

UNID
UNIVERSIDAD INTERAMERICANA PARA EL DESARROLLO

CAPÍTULO III

Fundamentos Teóricos de Sistemas Inteligentes y Visión por Computadora

3. Marco Teórico

3.1. Deep Learning

El aprendizaje profundo (Deep Learning) representa una evolución significativa dentro del campo del aprendizaje automático, caracterizándose por el uso de arquitecturas neuronales con múltiples capas que permiten modelar relaciones complejas a partir de grandes volúmenes de datos. A diferencia de los métodos tradicionales, donde la extracción de características depende de procesos manuales, el aprendizaje profundo posibilita la construcción automática de representaciones jerárquicas, optimizando la capacidad del modelo para reconocer patrones complejos en datos no estructurados.

Estas arquitecturas, denominadas redes neuronales convolucionales (CNN), han mostrado ser altamente efectivas para el análisis de la información visual, dado su potencial para reconocer las características espaciales y también contextuales en diferentes niveles de abstracción.

El procedimiento de las CNN se basa en filtros convolucionales que hacen un barrido por la imagen de entrada, logrando así detectar bordes y formas de las texturas, las cuales se combinan progresivamente para generar representaciones de mayor nivel de abstracción a medida que se avanza en la profundidad de la red. En la agricultura, el aprendizaje profundo ha sido ampliamente direccionado para resolver problemas como la identificación de cultivos, detección de enfermedades y evaluación de calidad de productos. Estas aplicaciones han demostrado que los modelos basados en Deep Learning alcanzan niveles de precisión superiores a los métodos tradicionales y muy especialmente en situaciones con alta variabilidad ambiental (Koirala et al., 2022).

Desde el punto de vista metodológico, estos modelos se entrenan mediante procesos iterativos de optimización, siendo que los parámetros internos de la red son modificados a partir de algoritmos como el descenso de gradiente usando técnicas de retropropagación del error. Este proceso permite que se minimice la función de pérdida y al mismo tiempo optimizar el funcionamiento del modelo. La problemática

introducida por los modelos de aprendizaje profundo implica la necesidad de grandes conjuntos de datos etiquetados, así como sus cuantiosas exigencias computacionales para el entrenamiento.

Sin embargo, la aplicación de modelos de aprendizaje profundo plantea retos en relación a la necesidad de contar con amplios conjuntos de datos etiquetados previos, además de los requerimientos computacionales necesarios para realizar el proceso de entrenamiento. Sin embargo, y pese a sus limitaciones, su capacidad de modelar a través de relaciones no lineales complejas lo convierte en un ingrediente fundamental a la hora de desarrollar sistemas inteligentes para ser aplicados en el sector agrícola.

3.2. Detección de objetos

La detección de objetos es una de las principales tareas del ámbito de la visión por computadora orientada a la identificación y localización de los objetos de interés dentro de una imagen no sólo considerando la clasificación de los objetos allí presentes, sino que, además, la localización de su posición en el espacio mediante la definición de unas regiones ideales.

Partiendo del aprendizaje profundo, los modelos de detección de objetos han transitado hacia dos épocas o direcciones, dependiendo de separar o no la parte del proceso de detección en distintas etapas, o bien en el caso de que las dos tareas definidas antes constituyan una malla compacta e integral, lo que mejora considerablemente el rendimiento computacional.

Los modelos YOLO comprueban que la detección de objetos en tiempo real es posible, ya que permite el procesamiento de toda la imagen en una sola pasada Hacia la red neuronal, lo que acorta el tiempo de inferencia y permite una aplicación en entornos dinámicos donde la detección resulta un primer paso que debe ir acompañado de un paso de respuesta. Esto permite la predicción simultánea de objetos en la misma imagen por su estructura en forma de cuadrícula.

En dicho contexto, se han llegado a utilizar métodos de detección de objetos para tareas de identificación de frutos, estimación de cosechas o en la tarea del control de las cosechas. Con respecto a los resultados mostrados, estos modelos han llegado a obtener una notable capacidad de funcionamiento en contextos adversos y difíciles, los cuales son capaces de mostrar diferentes retos tales como la variabilidad de luminosidad, las malas condiciones de oclusión y la heterogeneidad de las condiciones de contexto (Koirala et al., 2022).

También, el desarrollo de arquitecturas más eficientes, la mejora del rendimiento de los modelos y el aumento de la exactitud, aunado con la reducción de las exigencias computacionales. Estas mejoras han facilitado la adopción de técnicas de detección de objetos en sistemas inteligentes aplicados a la agricultura, contribuyendo a la automatización de procesos tradicionalmente manuales.

3.3. Procesamiento de imágenes

El procesamiento digital de imágenes constituye una etapa fundamental dentro de los sistemas de visión por computadora, ya que permite transformar y mejorar la calidad de las imágenes antes de su análisis mediante modelos de aprendizaje profundo. Este proceso incluye un conjunto de técnicas orientadas a la mejora, transformación y análisis de la información visual.

Entre las principales técnicas utilizadas se encuentran la normalización de la imagen, la reducción de ruido, la mejora de contraste y la segmentación. La normalización permite estandarizar los valores de los píxeles, facilitando el procesamiento posterior, mientras que la reducción de ruido contribuye a eliminar interferencias que podrían afectar la precisión del análisis.

La conversión de espacios de color constituye otro elemento relevante, ya que permite representar la información de la imagen en diferentes formatos, como RGB, HSV o LAB, facilitando la identificación de características específicas relacionadas con el color y la intensidad. De igual manera, la ecualización de histogramas permite mejorar la

distribución de los niveles de intensidad, incrementando el contraste y resaltando detalles importantes.

La segmentación de imágenes, por su parte, permite la división de la imagen en varias regiones homogéneas y, así, poder localizar objetos de interés. Esta segmentación se puede hacer mediante métodos tradicionales o bien mediante métodos de aprendizaje profundo, que permiten segmentaciones de mucho mayor calidad y precisión en escenarios problemáticos.

En el ámbito de las aplicaciones agrícolas, el procesamiento de imágenes se convierte en un aspecto crítico a raíz de la gran variabilidad del entorno, que incluye aspectos como las variaciones en la luz, las manchas, el ruido y los tipos de fondos que pueden ser muy complejos y que pueden influir en la calidad de los datos de la imagen y, a su vez, en el rendimiento de estos algoritmos de análisis. A raíz de eso, la implementación de métodos de normalización, filtrado o mejora de contraste se torna una práctica necesaria para garantizar la robustez de los modelos de visión por computadora.

En el contexto de la profunda variabilidad del entorno, existe un significativo número de experiencias que han demostrado que añadir etapas de preprocesamiento de este tipo mejora en procesos de aprendizaje profundo las características o features obtenidas por estos modelos y mejora las tareas de detección y/o clasificación en entornos de gran variabilidad (Ferentinos, 2018).

3.4. Sistemas inteligentes

Los sistemas inteligentes pueden definirse como sistemas computacionales capaces de percibir su entorno, procesar información y tomar decisiones de manera autónoma o asistida, utilizando técnicas de Inteligencia Artificial. Estos sistemas integran diferentes componentes, incluyendo mecanismos de adquisición de datos, modelos de procesamiento y módulos de toma de decisiones.

En el contexto agrícola, los sistemas inteligentes han sido utilizados para automatizar procesos críticos, como el monitoreo de cultivos, la detección de anomalías y la gestión de recursos. Los sistemas informáticos tienen la capacidad de convertir grandes volúmenes de información en información útil para la toma de decisiones basada en la evidencia. Un sistema inteligente se organiza normalmente en tres niveles: adaptación de datos, procesamiento de la información y producción de acciones. En los sistemas basados en la visión por computadora, el tratamiento de los datos se realiza mediante dispositivos de captura de imágenes, el procesamiento se lleva a cabo mediante modelos de aprendizaje profundo y la producción de resultados se realiza mediante interfaces para interactuar con los usuarios.

La integración de estos sistemas con tecnologías emergentes: Internet de las Cosas (IoT) y computación en la nube, ha incrementado las capacidades de acceso a la información en tiempo real y ha incrementado la escalabilidad de las soluciones.

En este sentido, los sistemas inteligentes son una herramienta vital para la transformación digital del ámbito agrícola. Sin embargo, la implementación de estas tecnologías se enfrenta a los problemas de la interoperabilidad entre sistemas, la disponibilidad de la infraestructura tecnológica y la adaptación a las condiciones específicas del entorno de aplicación.

A pesar de estas limitaciones, su potencial para mejorar la eficiencia, reducir costos y optimizar procesos productivos los posiciona como un elemento fundamental en el desarrollo de soluciones tecnológicas avanzadas (Wolfert et al., 2021).

CAPÍTULO IV

Metodología de Investigación y Diseño Experimental

4. Metodología de Investigación

4.1. Tipo: investigación aplicada y experimental

La presente investigación se enmarca en el ámbito de la investigación aplicada, dado que su propósito principal consiste en el desarrollo e implementación de una solución tecnológica orientada a resolver un problema específico en el sector agrícola, relacionado con la evaluación automatizada de productos mediante técnicas de Inteligencia Artificial. Esta tipología de investigación se identifica por la utilización de saberes científicos para la producción de soluciones prácticas que permiten la mejora de procesos de producción en situaciones reales. El estudio se plantea, además, bajo un diseño experimental, en la medida en que se diseña, entrena y evalúa un modelo de aprendizaje profundo en condiciones controladas, lo que permite analizar la influencia de una serie de variables respecto al desempeño del sistema.

En este sentido, se manipularán parámetros asociados al modelo, a la base de datos y en el proceso de entrenamiento, que permitirán la optimización del rendimiento. El enfoque metodológico es de tipo cuantitativo puesto que la evaluación del modelo es ejecutada a partir de métricas numéricas que permiten medir el rendimiento de una forma objetiva y reproducible; este enfoque es ampliamente utilizado en investigaciones que se basan en aprendizaje profundo, donde la validación del modelo se sostiene a partir de indicadores estadísticos de precisión y rendimiento (Goodfellow et al., 2016).

4.2. Dataset utilizado

El desempeño de los modelos de aprendizaje profundo depende en gran medida de la calidad y representatividad del conjunto de datos utilizado durante el entrenamiento. En este estudio, se emplea un dataset compuesto por imágenes digitales de productos agrícolas, las cuales representan diferentes estados del producto, tales como madurez, inmadurez y deterioro.

El dataset puede clasificarse como propio o adaptado, dependiendo de su origen. Para los propios datasets, las imágenes se obtienen en escenarios reales, y por eso podemos añadir variabilidad en la iluminación, el fondo y la posición del objeto. Este tipo de enfoque ayuda a mejorar la capacidad de generalización del modelo. En el caso de los datasets adaptados, se provienen de repositorios de datasets existentes que se modifican de acuerdo a procesos como la selección, limpieza y re-etiquetado.

El proceso de anotación de datos es una parte muy relevante, ya que se asignan las etiquetas de clase y las delimitaciones de las regiones de interés usando bounding boxes, lo que permite que el propio modelo esté aprendiendo la localización de los objetos y la clasificación en la propia imagen.

También se utilizan técnicas de aumento de los datos (data augmentation), como pueden ser rotaciones, traducciones, escalados o variaciones en el brillo, que permiten aumentar la diversidad del dataset y que el modelo sea más robusto a las variaciones del entorno. Estas técnicas han demostrado que son útiles para evitar el sobreajuste y mejorar rendimiento de los modelos de visión por ordenador (Shorten & Khoshgoftaar, 2019).

4.3. Proceso de entrenamiento del modelo

El entrenamiento del modelo se basa en el uso de arquitecturas de aprendizaje profundo orientadas a la detección de objetos, particularmente modelos de la familia YOLO, los cuales han demostrado un alto desempeño en tareas de visión por computadora en tiempo real.

El proceso de entrenamiento se basa en alimentar el modelo con el dataset previamente preparado, de manera que la red neuronal pueda ir ajustando sus parámetros internos de forma progresiva. Durante este proceso, el modelo realiza las predicciones necesarias que una vez generadas se comparan con las etiquetas reales a través de la función del error o de pérdida, que permite medir el error tanto en la localización como en la clasificación. Para mejorar el funcionamiento del modelo, se

utiliza un algoritmo de optimización que, utilizando la técnica del descenso de gradiente, intenta reducir ese error en distintas iteraciones, y hace esos ajustes en el proceso de retropropagación que permite modificar los pesos de la red en función de los resultados obtenidos en la cada una de las pasadas del entrenamiento (LeCun et al., 2015).

Por otro lado, el dataset de evaluación también se divide en los conjuntos de entrenamiento y de validación. Este procedimiento permite para poder ir evaluando el comportamiento del modelo durante su aprendizaje, lo que resulta útil para detectar problemas como el conocido como el sobreajuste, que hace referencia a ese fenómeno que hace que el modelo se adapte demasiado a los datos de entrenamiento y pierda, en consecuencia, capacidad de generalización ante datos nuevos.

El proceso de transfer learning resulta ser muy útil en este caso, ya que permite inicializar el modelo inicial con un conjunto de pesos previamente entrenados en grandes datasets, lo que contribuye a reducir el tiempo de entrenamiento y hacer que el modelo funcione de forma más eficiente para resolver tareas específicas (Tan et al., 2018).

4.4. Configuración

La configuración de los parámetros del modelo representa un elemento clave en la optimización de su desempeño. En este estudio, se consideran parámetros fundamentales asociados a la arquitectura YOLO, los cuales son ajustados en función de las características del problema.

El parámetro imgsz (image size) define la resolución de entrada de las imágenes, influyendo directamente en la capacidad del modelo para capturar detalles visuales. Un mayor tamaño de imagen permite mejorar la precisión en la detección, aunque incrementa el costo computacional.

El parámetro confidence threshold (conf) establece el umbral mínimo de confianza requerido para considerar una detección como válida. Este parámetro permite controlar el equilibrio entre falsos positivos y falsos negativos, siendo un elemento crítico en la optimización del modelo.

Por su parte, el parámetro Intersection over Union (IoU) se utiliza para medir el grado de superposición entre las predicciones del modelo y las etiquetas reales. Este indicador es fundamental en la evaluación del desempeño del modelo, así como en la aplicación de técnicas como la supresión de no máximos, que permite eliminar detecciones redundantes (Padilla et al., 2020).

Otros parámetros relevantes incluyen la tasa de aprendizaje, el tamaño del batch y el número de épocas, los cuales influyen en la convergencia del modelo y su capacidad de generalización.

4.5. Validación del modelo

La validación del modelo constituye una etapa esencial para evaluar su desempeño y capacidad de generalización. Este trabajo incluye métricas numéricas que permiten medir la exactitud del modelo en tareas de deteención y distinción. Las métricas más sobresalientes son la precisión (precision) y el recall, que permiten medir exactamente cómo un modelo es capaz de reconocer objetos; la precision representa el número de deteenciones correctas, y el recall permite calcular la capacidad de un modelo para reconocer todos los objetos existentes.

Por último, también se tiene en cuenta la métrica mean Average Precision (mAP), que es muy usada en las tareas de detección y que proporciona una medida más global del comportamiento del modelo a partir de diferentes thresholds de IoU (Everingham et al., 2010).

El proceso de validación consiste en hacer uso de otro conjunto de datos que no coincide con el conjunto de datos que se utilizó en la etapa de entrenamiento, de forma que pueda comprobarse el correcto comportamiento del modelo a partir de datos no pertenecientes al conjunto con el que se ha entrenado. De aquí que la validación garantiza una evaluación objetiva y evita el fenómeno conocido como sobreajuste de modelo. La validación experimental se realiza mediante pruebas bajo condiciones reales, en donde se revisa el desempeño del modelo para distintos puntos de vista, condiciones de luz, fondos o bien calidades de la imagen.

CAPÍTULO V

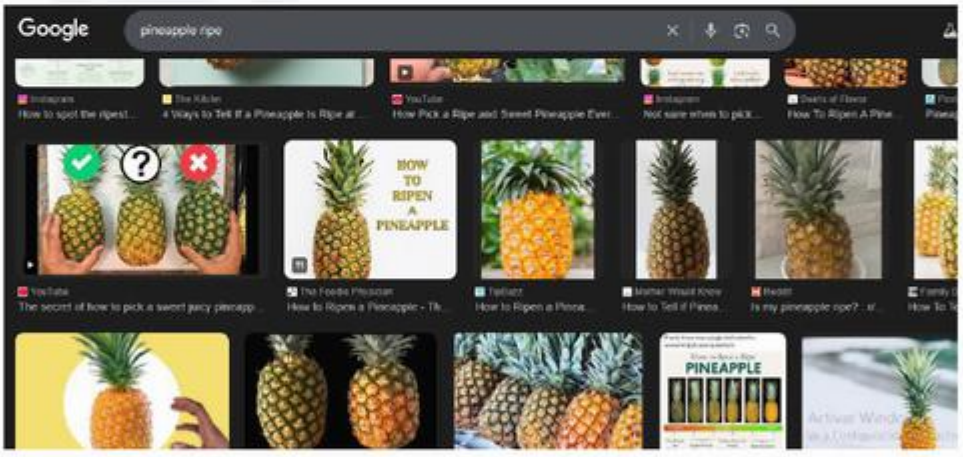
Entrenamiento y Optimización del Modelo de Inteligencia Artificial

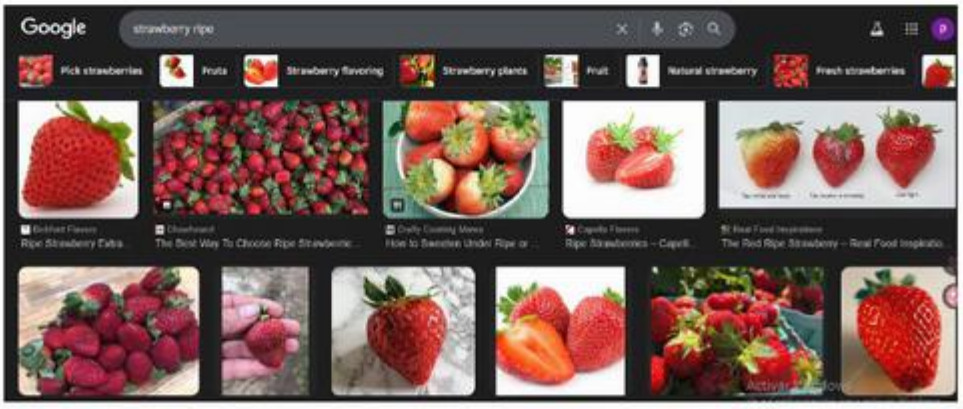
5. Entrenamiento y Optimización del Modelo

El desarrollo del modelo de Inteligencia Artificial representa una etapa crucial en el sistema propuesto, ya que permite transformar datos visuales en información estructurada mediante técnicas de aprendizaje profundo. Este proceso abarca diversas fases, que incluyen la preparación de datos, el etiquetado, el entrenamiento del modelo, el ajuste de hiperparámetros y la evaluación de resultados. Conforme a la metodología establecida en el Capítulo IV, el modelo implementado se fundamenta en arquitecturas de detección de objetos de tipo YOLO, las cuales han sido optimizadas para el análisis de productos agrícolas en diferentes estados: maduro, inmaduro y deteriorado.

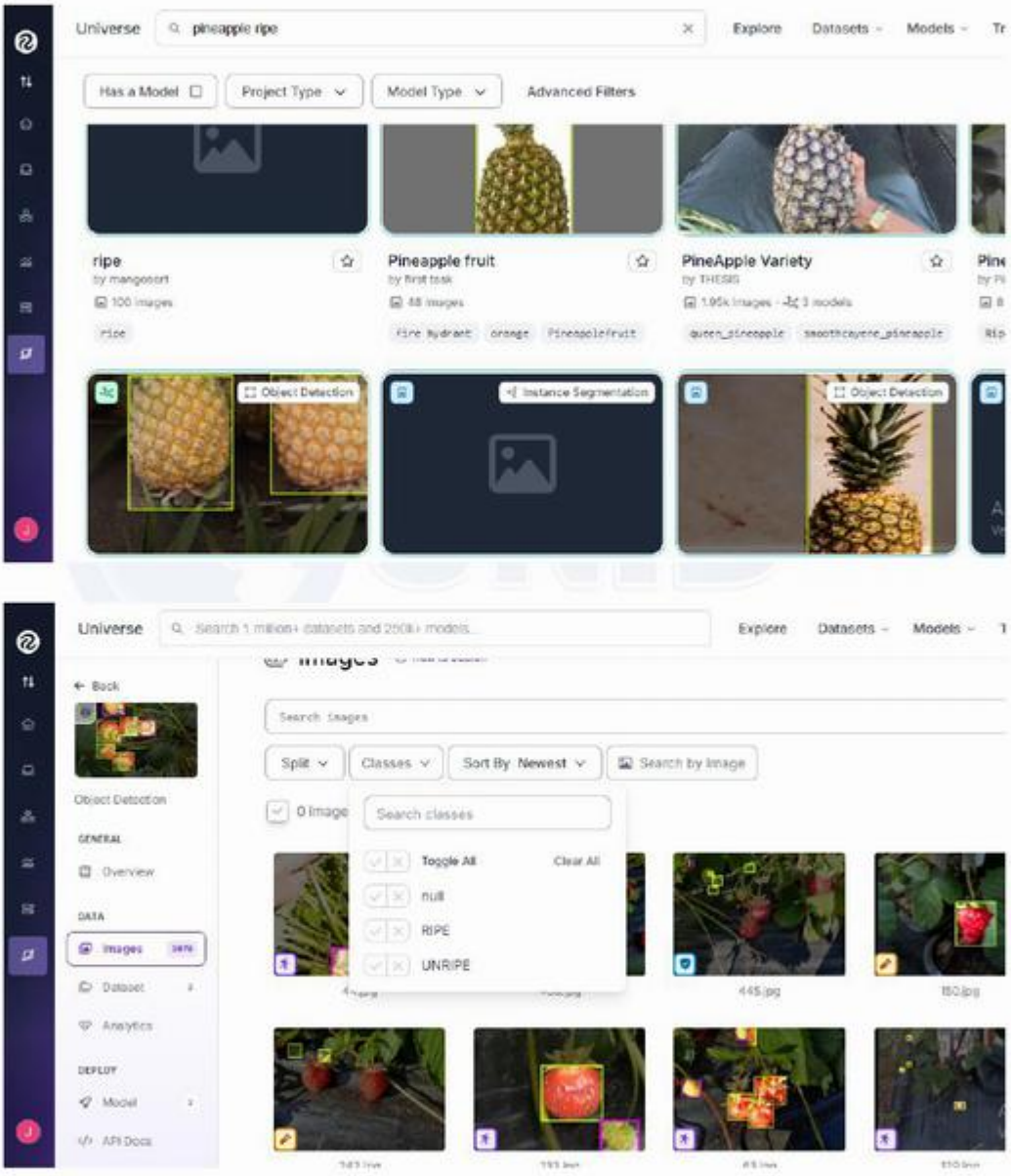
5.1. Preparación de datos

La preparación de datos es una fase crucial que asegura el rendimiento óptimo del modelo. En este estudio, el conjunto de datos se elaboró a partir de la combinación de fuentes internas y datasets públicos, lo que enriqueció la diversidad y calidad de la información utilizada para el entrenamiento.





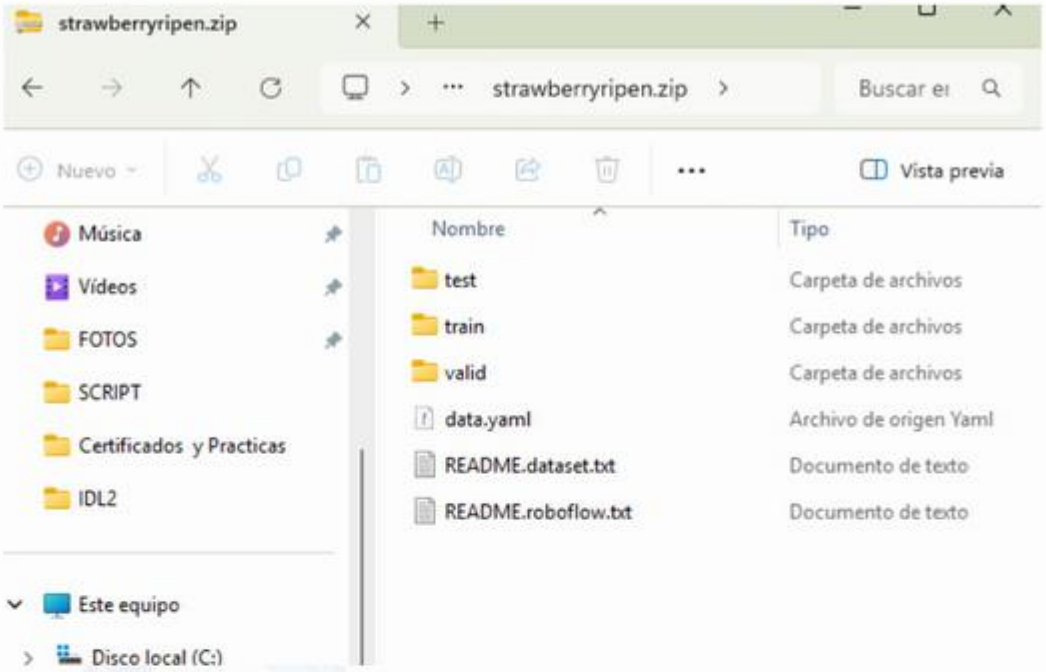
En la fase inicial, se llevó a cabo la recolección de imágenes de productos agrícolas en diversas condiciones, asegurando una amplia variabilidad en factores como la iluminación, el fondo y la posición de los objetos. A continuación, se enriqueció el conjunto de datos utilizando plataformas como Roboflow, lo que permitió aumentar aún más la diversidad de ejemplos y optimizar la calidad del dataset.



Los datasets descargados desde Roboflow no siempre presentan una estructura uniforme, ni contienen únicamente las clases requeridas para el desarrollo del proyecto. Por lo general, estos conjuntos de datos se organizan en las siguientes carpetas:

✓ train/

- ✓ valid/
- ✓ test/



Dentro de cada una de estas carpetas se encuentran, comúnmente, las siguientes subcarpetas:

- ✓ images/: contiene las imágenes del dataset.
- ✓ labels/: incluye las anotaciones en formato YOLO, correspondientes a cada imagen.



Esta estructura es estándar en muchos datasets de visión por computadora; sin embargo, suele requerir un **proceso de revisión, filtrado y reorganización** para adaptarse a las necesidades específicas del proyecto y a las clases de interés.

Además, un mismo dataset puede contener **múltiples clases**, muchas de las cuales **no son relevantes para el objetivo del proyecto**, como frutas en estados de madurez que no serán considerados durante el entrenamiento.

Por este motivo, resulta necesario aplicar un **proceso previo de filtrado y limpieza de datos** antes de integrar estos datasets al conjunto de datos principal.

5.1.1 Problema de múltiples clases en datasets públicos

Los datasets públicos disponibles en Roboflow suelen presentar las siguientes características:

- ✓ Incluir varias frutas dentro de un mismo dataset.
- ✓ Asignar identificadores de clase distintos para cada objeto.
- ✓ Mezclar estados de madurez no deseados dentro de una misma colección.
- ✓ Presentar proporciones desiguales entre clases, generando desbalance.

Si estos datasets se utilizan directamente sin un tratamiento previo, pueden:

- ✓ Introducir ruido en el modelo de entrenamiento.
- ✓ Generar desbalance de clases, afectando la capacidad de generalización.
- ✓ Impactar negativamente en el rendimiento del modelo durante el entrenamiento y la validación.

Por esta razón, se implementa un script de extracción selectiva de clases, el cual permite conservar únicamente la información relevante para el proyecto, asegurando un dataset más limpio, balanceado y alineado con los objetivos del modelo de visión por computadora.

```yaml
train: ../train/images
val: ../valid/images
test: ../test/images

nc: 4
names: ['0', '1', 'ripe', 'unripe']

roboflow:
  workspace: hsu-ban0r
  project: strawberry-8qsy9
  version: 1
  license: CC BY 4.0
  url: https://universe.roboflow.com/hsu-ban0r/strawberry-8qsy9/dataset/1
```

5.1.2 Extracción selectiva de clases desde datasets YOLO usando Python

Para abordar el problema de la presencia de múltiples clases irrelevantes en datasets públicos, se desarrolló un **script en Python** que permite realizar una **extracción selectiva de clases** a partir de datasets descargados desde Roboflow en **formato YOLO**.

Este script cumple las siguientes funciones:

- ✓ Leer datasets en formato YOLO descargados desde Roboflow.
- ✓ Recorrer automáticamente las carpetas train, valid y test.
- ✓ Identificar una clase objetivo-específica definida por el proyecto.
- ✓ Copiar únicamente las imágenes que contienen dicha clase.
- ✓ Generar nuevos archivos de etiquetas (labels) limpios, conservando solo la información relevante.
- ✓ Crear un **subconjunto de datos (subset)** compuesto únicamente por:
  - ➢ images/
  - ➢ labels/

Este procedimiento se repite para **cada fruta y cada estado de madurez** considerado relevante dentro del proyecto.

```python
import os
import zipfile
import shutil

# === CONFIGURACIÓN ===
zip_name = "concha.zip"
dataset_dir = os.path.splitext(zip_name)[0]
num_imagenes = 450
clase_objetivo = "0"
nuevo_id = "11"
carpetas_buscar = ["train", "valid", "test"]
destino = "subset_filtered"

# === 1. DESCOMPRIMIR SI ES ZIP ===
if zip_name.endswith(".zip"):
    if not os.path.exists(dataset_dir):
        with zipfile.ZipFile(zip_name, 'r') as zip_ref:
            zip_ref.extractall(dataset_dir)
        print(f" ZIP extraído en: {dataset_dir}")
    else:
        print(f" Ya existe la carpeta: {dataset_dir}")
else:
    print(f" Usando dataset ya descomprimido: {dataset_dir}")

# === 2. CREAR CARPETAS DESTINO ===
dst_images = os.path.join(destino, "images")
dst_labels = os.path.join(destino, "labels")
os.makedirs(dst_images, exist_ok=True)
```

extraer_y_mantener2.py > ...

```python
# === 2. CREAR CARPETAS DESTINO ===
dst_images = os.path.join(destino, "images")
dst_labels = os.path.join(destino, "labels")
os.makedirs(dst_images, exist_ok=True)
os.makedirs(dst_labels, exist_ok=True)

# === 3. RECORRER TODAS LAS CARPETAS Y FILTRAR ===
copiadas = 0
for subset in carpetas_buscar:
    src_images = os.path.join(dataset_dir, subset, "images")
    src_labels = os.path.join(dataset_dir, subset, "labels")

    if not os.path.exists(src_images) or not os.path.exists(src_labels):
        continue

    print(f" Buscando en {subset}...")

    for lbl_name in sorted(os.listdir(src_labels)):
        if not lbl_name.endswith(".txt"):
            continue

        lbl_path = os.path.join(src_labels, lbl_name)
        with open(lbl_path, "r") as f:
            lineas = f.readlines()

        # Solo mantener las líneas de la clase deseada
        nuevas = []
        for linea in lineas:
            partes = linea.strip().split()
            if partes and partes[0] == clase_objetivo:
                partes[0] = nuevo_id
                nuevas.append(" ".join(partes))
```

```python
                nuevas.append(" ".join(partes))

        # Si tiene esa clase, copiar la imagen + etiqueta filtrada
        if nuevas:
            base = os.path.splitext(lbl_name)[0]
            posibles_ext = [".jpg", ".png", ".jpeg"]
            img_path = None
            for ext in posibles_ext:
                ruta = os.path.join(src_images, base + ext)
                if os.path.exists(ruta):
                    img_path = ruta
                    break

            if img_path:
                # Copiar imagen
                shutil.copy(img_path, os.path.join(dst_images, os.path.b

                # Guardar etiqueta filtrada
                with open(os.path.join(dst_labels, lbl_name), "w") as f:
                    f.write("\n".join(nuevas))

                copiadas += 1

            if copiadas >= num_imagenes:
                break
    if copiadas >= num_imagenes:
        break

print(f"\n Se copiaron {copiadas} imágenes con solo la clase {clase_obje
```

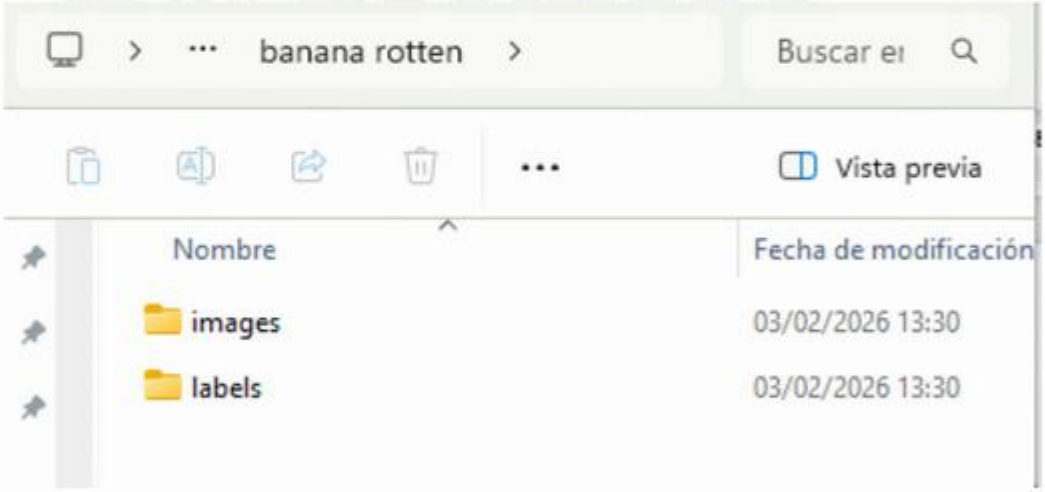
5.1.3 Resultado del proceso de filtrado

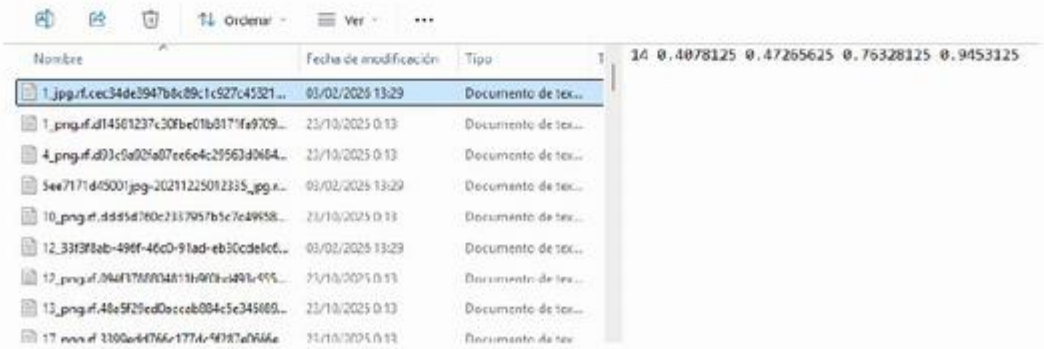
Al finalizar la ejecución del script de extracción selectiva, se obtiene una **estructura de datos limpia, uniforme y optimizada**, que contiene únicamente la información relevante para el proyecto.

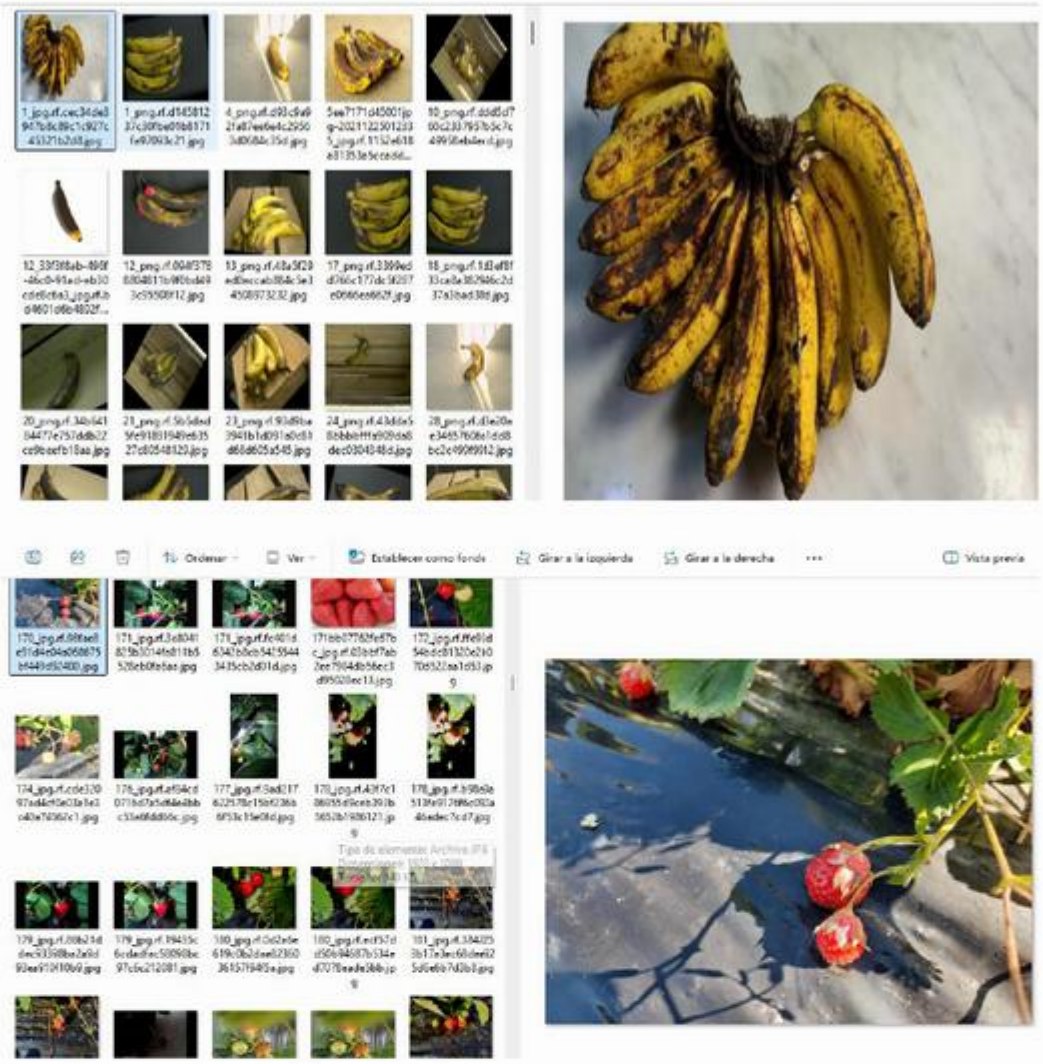
El resultado consiste en un **subconjunto de datos depurado**, organizado de la siguiente manera:

- ✓ images/: incluye únicamente las imágenes que contienen la clase objetivo.
- ✓ labels/: contiene los archivos de anotaciones en formato YOLO, filtrados para conservar exclusivamente la clase seleccionada.

Esta estructura simplificada facilita la **integración del dataset en el proceso de entrenamiento**, reduce el ruido en los datos y contribuye a mejorar el rendimiento y la estabilidad de los modelos de visión por computadora.







5.2. Etiquetado

El proceso de etiquetado se llevó a cabo utilizando la herramienta Label Studio, que facilita la anotación visual de imágenes mediante el uso de bounding boxes. Esta plataforma permite una interacción intuitiva y precisa, asegurando que las imágenes sean etiquetadas de manera eficiente y con un alto grado de exactitud.

5.2.1 ¿Por qué usar Label Studio en el proyecto?

Para el desarrollo del proyecto productivo es indispensable contar con un proceso adecuado de etiquetado de imágenes, ya sea mediante la delimitación de objetos con *bounding boxes*, la clasificación visual u otros tipos de anotación. Estas etiquetas permiten construir un dataset confiable que será utilizado en el entrenamiento y validación de modelos de visión por computadora.

En este contexto, Label Studio se presenta como una herramienta *open-source* que facilita el proceso de etiquetado de forma visual, estructurada y eficiente. Su interfaz intuitiva permite organizar proyectos, definir clases, gestionar imágenes y exportar anotaciones en distintos formatos compatibles con modelos de *machine learning* y *deep learning*, lo que lo convierte en una opción adecuada para proyectos productivos y académicos.

5.2.1.1 Opciones de instalación de Label Studio

Label Studio puede instalarse principalmente de dos formas:

- ✓ Mediante Docker (opción recomendada).
- ✓ Mediante Python desde la terminal utilizando pip.

En el presente documento se describe la instalación de Label Studio mediante Docker ya que esta modalidad ofrece diversas ventajas frente a la instalación tradicional con Python.

Entre las principales razones se encuentran la reducción de conflictos de dependencias, una mayor estabilidad del entorno de trabajo y su amplio uso en entornos profesionales y productivos. Además, Docker permite una configuración más controlada y facilita la replicabilidad del sistema en diferentes equipos.

5.3. Entrenamiento YOLO

El entrenamiento del modelo se realizó utilizando arquitecturas de la familia YOLO, específicamente adaptadas para tareas de detección de objetos en tiempo real.

5.3.1 Limpieza y unificación del dataset con Python

Una vez descargados y organizados los distintos datasets, se emplea un script en Python para realizar el proceso de limpieza, unificación y preparación final del conjunto de datos. Este script permite:

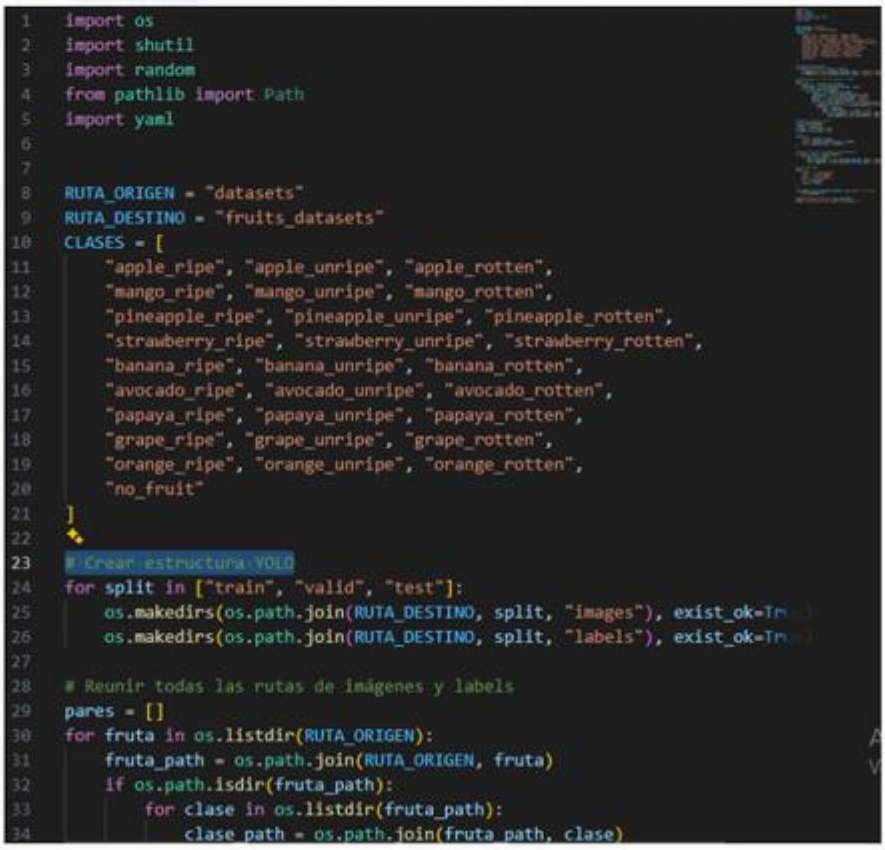
- ✓ Recorrer automáticamente todas las carpetas del dataset.

- ✓ Validar la correspondencia entre imágenes y archivos de etiquetas (image–label).
- ✓ Mezclar los datos de forma aleatoria, evitando sesgos en el entrenamiento.
- ✓ Dividir el conjunto de datos en entrenamiento, validación y prueba.
- ✓ Construir una estructura final compatible con YOLO.

El script ejecuta las siguientes acciones clave:

- ✓ Crea la carpeta principal fruits_datasets.
- ✓ Genera las subcarpetas train, valid y test.
- ✓ Copia de forma sincronizada las imágenes y sus etiquetas correspondientes.
- ✓ Genera automáticamente el archivo data.yaml, el cual define rutas y clases necesarias para el entrenamiento del modelo.

Este proceso garantiza la obtención de un dataset limpio, balanceado y estandarizado, listo para ser utilizado en el entrenamiento de modelos de detección de objetos basados en YOLO, asegurando mayor estabilidad y rendimiento durante el aprendizaje.



```python
import os
import shutil
import random
from pathlib import Path
import yaml


RUTA_ORIGEN = "datasets"
RUTA_DESTINO = "fruits_datasets"
CLASES = [
    "apple_ripe", "apple_unripe", "apple_rotten",
    "mango_ripe", "mango_unripe", "mango_rotten",
    "pineapple_ripe", "pineapple_unripe", "pineapple_rotten",
    "strawberry_ripe", "strawberry_unripe", "strawberry_rotten",
    "banana_ripe", "banana_unripe", "banana_rotten",
    "avocado_ripe", "avocado_unripe", "avocado_rotten",
    "papaya_ripe", "papaya_unripe", "papaya_rotten",
    "grape_ripe", "grape_unripe", "grape_rotten",
    "orange_ripe", "orange_unripe", "orange_rotten",
    "no_fruit"
]

# Crear estructura YOLO
for split in ["train", "valid", "test"]:
    os.makedirs(os.path.join(RUTA_DESTINO, split, "images"), exist_ok=Tr
    os.makedirs(os.path.join(RUTA_DESTINO, split, "labels"), exist_ok=Tr

# Reunir todas las rutas de imágenes y labels
pares = []
for fruta in os.listdir(RUTA_ORIGEN):
    fruta_path = os.path.join(RUTA_ORIGEN, fruta)
    if os.path.isdir(fruta_path):
        for clase in os.listdir(fruta_path):
            clase_path = os.path.join(fruta_path, clase)
```

```python
# Mezclar y dividir dataset
random.shuffle(pares)
n_total = len(pares)
n_train = int(n_total * 0.8)
n_valid = int(n_total * 0.1)

splits = {
    "train": pares[:n_train],
    "valid": pares[n_train:n_train + n_valid],
    "test": pares[n_train + n_valid:]
}

# Copiar archivos a su carpeta correspondiente
for split, lista in splits.items():
    for img, lbl in lista:
        shutil.copy(img, os.path.join(RUTA_DESTINO, split, "images"))
        shutil.copy(lbl, os.path.join(RUTA_DESTINO, split, "labels"))

# Crear data.yaml
data = {
    "train": "./train/images",
    "val": "./valid/images",
    "test": "./test/images",
    "nc": len(CLASES),
    "names": CLASES
}

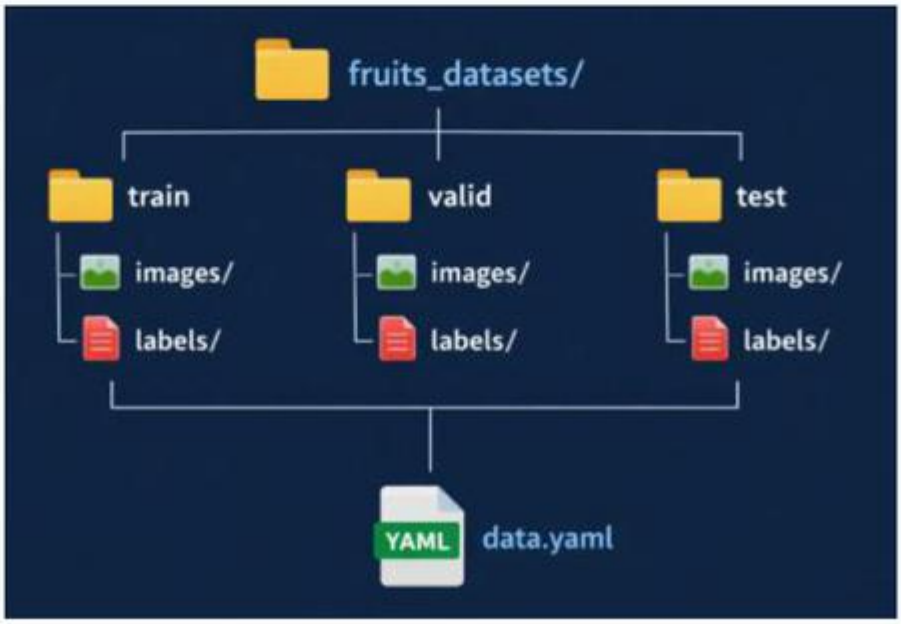
with open(os.path.join(RUTA_DESTINO, "data.yaml"), "w") as f:
    yaml.dump(data, f)

print(f" Dataset listo en: {RUTA_DESTINO}")
print(" Estructura YOLO creada correctamente.")
```

Ln 75, Col 8 Spaces: 4 UTF-8 CRLF {} Py

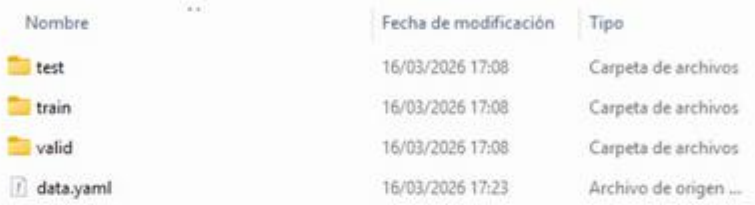
5.3.2. Dataset final preparado para entrenamiento

Como resultado del proceso de limpieza y unificación, el script genera una estructura final de dataset organizada y lista para el entrenamiento del modelo. La estructura obtenida es la siguiente:



Esta organización cumple con el estándar requerido por YOLO, permitiendo una separación clara entre los conjuntos de entrenamiento, validación y prueba, así como una correcta asociación entre imágenes y etiquetas.

La estructura generada es completamente compatible con YOLO, lo que permite iniciar directamente el proceso de entrenamiento sin necesidad de ajustes adicionales.



Nombre	Fecha de modificación	Tipo
test	16/03/2026 17:08	Carpeta de archivos
train	16/03/2026 17:08	Carpeta de archivos
valid	16/03/2026 17:08	Carpeta de archivos
data.yaml	16/03/2026 17:23	Archivo de origen ...

El procedimiento del entrenamiento se realizó de la siguiente forma:

- ✓ Carga del dataset: se optó por un dataset preestablecido para hacer el entrenamiento.
- ✓ Inicialización del modelo: se tomaron dos decisiones:
  - ➢ Usar un modelo preentrenado que tienen adquiridos conocimientos proveniente de un dataset anterior.
  - ➢ Entrenar un modelo desde cero, en el que se utilizan pesos aleatorios.
- ✓ Ajuste de pesos: se hizo un ajuste iterativo de los pesos utilizando el algoritmo de backpropagation, de forma que el modelo va aprendiendo de los errores que comete durante el entrenamiento.
- ✓ Evaluación continua: el rendimiento del modelo fue evaluado de forma continua a través de un conjunto de validación, de forma que se va evolucionando sin caer en la sobrepotencia del modelo. La configuración del entrenamiento tiene en cuenta los siguientes parámetros principales:
  - ➢ 100 epochs.
  - ➢ Dimensiones de imagen de 640 × 640 pixeles, de acuerdo con la configuración estándar en modelos YOLO.
  - ➢ Uso de GPU, para acelerar el proceso de aprendizaje.
  - ➢ Early stopping, para prevenir el sobreentrenamiento.

- Validación automática durante el proceso de entrenamiento.

Almacenamiento periódico de pesos en función de los mejores resultados obtenidos.

```python
from google.colab import drive
import os


drive.mount('/content/drive')


zip_path = "/content/drive/MyDrive/fruits_datasets.zip"
# Carpeta donde se guardará todo de forma permanente
dest_folder = "/content/drive/MyDrive/dataset_frutas_fijo"

# Descomprimir solo si no existe
if not os.path.exists(dest_folder):
    os.makedirs(dest_folder, exist_ok=True)
    print("Descomprimiendo... esto tardará unos minutos en Drive.")
    !unzip -q "{zip_path}" -d "{dest_folder}"
    print("¡Descompresión terminada!")
else:
    print("El dataset ya está en Drive. Saltando descompresión.")

# Instalar y cargar YOLO
!pip install -q ultralytics
from ultralytics import YOLO


yaml_path = f"{dest_folder}/fruits_datasets/data.yaml"

model = YOLO('yolov8s.pt')
results = model.train(
    data='/content/drive/MyDrive/dataset_frutas_fijo/fruits_datasets/data.yaml',
    epochs=60,
    patience=10,
    batch=16,
    imgsz=640,
    device=0,
    save_period=10,
    cache=False,
    project='/content/drive/MyDrive/yolo_runs',
    name='fruits_v8s',
    exist_ok=True
)
```

```python
!pip install -q ultralytics
from google.colab import drive
from ultralytics import YOLO
import os


drive.mount('/content/drive')



last_checkpoint = '/content/drive/MyDrive/yolo_runs/fruits_v8s/weights/last.pt'

if os.path.exists(last_checkpoint):
    print("Retomando entrenamiento desde el último punto...")
    model = YOLO(last_checkpoint)
    model.train(resume=True)
else:
    print("No se encontró el archivo last.pt. Verifica la ruta en tu Drive.")
```

```
                                        1.2/1.2 MB 30.1 MB/s eta 0:00:00
Creating new Ultralytics Settings v0.0.6 file ✅
View Ultralytics Settings with 'yolo settings' or at '/root/.config/Ultralytics/settings.json'
Update settings with 'yolo settings key=value', i.e. 'yolo settings runs_dir=path/to/dir'. For help see https://docs.ultralytics.com/quickstart/#ultralyt
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Retomando entrenamiento desde el último punto...
Ultralytics 8.4.23 🚀 Python-3.12.12 torch-2.10.0+cu128 CUDA:0 (Tesla T4, 14913MiB)
engine/trainer: agnostic_nms=False, amp=True, angle=1.0, augment=False, auto_augment=randaugment, batch=16, bgr=0.0, box=7.5, cache=False, cfg=None, clas
Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultralytics/Arial.ttf': 100% ━━━━━━━━━━━━ 755.1KB 20.5MB/s 0.0s

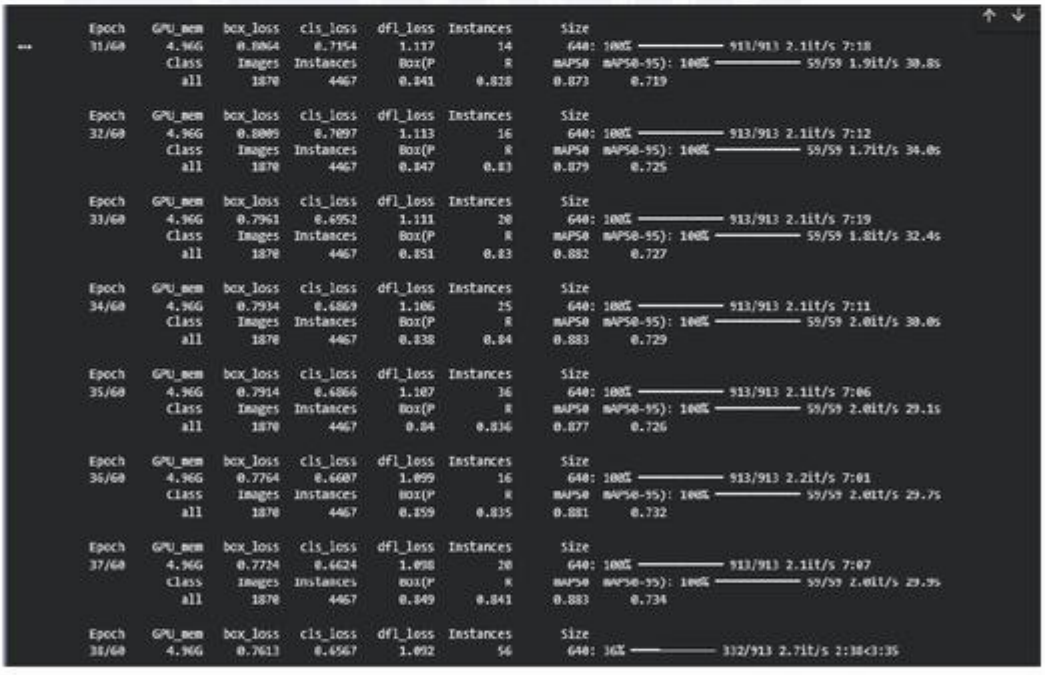
                   from  n    params  module                                       arguments
  0                  -1  1       928  ultralytics.nn.modules.conv.Conv             [3, 32, 3, 2]
```

Este proceso permite entrenar un modelo robusto con capacidad de generalización, orientado a la detección del estado de madurez de frutas, optimizando el desempeño del sistema en escenarios reales.

Se empleó la librería Ultralytics YOLO, que ofrece una implementación eficiente del modelo. Durante el proceso de entrenamiento, el modelo adquiere las siguientes habilidades:

- ✓ Detección de objetos: Localiza la posición de los objetos dentro de la imagen mediante bounding boxes.
- ✓ Clasificación del estado del producto: Evalúa y clasifica el estado de cada producto detectado.
- ✓ Asignación de niveles de confianza: Proporciona un nivel de confianza para cada detección, indicando la certeza del modelo sobre la identificación realizada.

El uso de transfer learning facilitó una notable reducción en el tiempo de entrenamiento y potenció el rendimiento del modelo en el ámbito agrícola.



5.4. Ajuste de hiperparámetros

El ajuste de la sintonía de hiperparámetros, constituye un paso fundamental para maximizar el rendimiento del modelo. En esta investigación, se realizó el ajuste de los siguientes parámetros:

- ✓ Imgsz: Tamaño de entrada de la imagen. Este parámetro se encarga de especificar las dimensiones (ancho y alto) de la imagen que alimentará al modelo.
- ✓ Batch size: Número de imágenes por iteración. Este hiperparámetro se encarga de especificar cuántas imágenes se pasan en un único paso de entrenamiento, lo cual afecta la memoria utilizada y la estabilidad de las actualizaciones de los pesos.
- ✓ Epochs: Número de iteraciones de entrenamiento. Este hiperparámetro especifica cuántas veces se pasará todo el conjunto de datos así a través del modelo. Se debe tener en cuenta que un mayor número de epochs puede resultar en un mejor rendimiento, pero puede caer en el sobreajuste.
- ✓ Learning rate: Velocidad de aprendizaje. Este parámetro determina la magnitud de los ajustes realizados en los pesos del modelo basándose en el error estimado. Un buen learning rate es vital para el correcto entrenamiento del modelo.
- ✓ Confidence threshold (conf): Umbral de confianza. Este hiperparámetro decide si se considera válida una predicción realizada por el modelo o no, el valor de este umbral se basa en la confianza del modelo en la predicción.
- ✓ Intersection over Union (IoU): Criterio de superposición. Esta métrica se encargará de evaluar la precisión del modelo de detección de objetos comparando la predicción del modelo con la caja delimitadora deseada real.

```python
from google.colab import drive
import os


drive.mount('/content/drive')


zip_path = "/content/drive/MyDrive/fruits_datasets.zip"
# Carpeta donde se guardará todo de forma permanente
dest_folder = "/content/drive/MyDrive/dataset_frutas_fijo"

# Descomprimir solo si no existe
if not os.path.exists(dest_folder):
    os.makedirs(dest_folder, exist_ok=True)
    print("Descomprimiendo... esto tardará unos minutos en Drive.")
    !unzip -q "{zip_path}" -d "{dest_folder}"
    print("¡Descompresión terminada!")
else:
    print("El dataset ya está en Drive. Saltando descompresión.")

# Instalar y cargar YOLO
!pip install -q ultralytics
from ultralytics import YOLO


yaml_path = f"{dest_folder}/fruits_datasets/data.yaml"

model = YOLO('yolov8s.pt')
results = model.train(
    data='/content/drive/MyDrive/dataset_frutas_fijo/fruits_datasets/data.yaml',
    epochs=60,
    patience=10,
    batch=16,
    imgsz=640,
    device=0,
    save_period=10,
    cache=False,
    project='/content/drive/MyDrive/yolo_runs',
    name='fruits_v8s',
    exist_ok=True
)
```

Este proceso permitió encontrar un equilibrio entre:

- ✓ Precisión: La capacidad del modelo para realizar predicciones correctas.
- ✓ Velocidad de inferencia: El tiempo que tarda el modelo en proporcionar resultados después de recibir una entrada.
- ✓ Capacidad de generalización: La habilidad del modelo para aplicar lo aprendido a datos no vistos y evitar el sobreajuste.
- ✓ Robustez: La resistencia del modelo ante datos ruidosos o condiciones cambiantes.
- ✓ Complejidad del modelo: La cantidad de parámetros y la arquitectura utilizada, que afecta tanto el rendimiento como el tiempo de entrenamiento.

- ✓ Costo computacional: Los recursos de hardware necesarios para entrenar y operar el modelo eficientemente.
- ✓ Interoperabilidad: La facilidad con la que el modelo puede integrarse y funcionar con otros sistemas y tecnologías.
- ✓ Escalabilidad: La capacidad del modelo para manejar un aumento en la cantidad de datos o usuarios sin comprometer su rendimiento.

5.5. Resultados del entrenamiento

Los resultados obtenidos demuestran el correcto funcionamiento del modelo entrenado. De acuerdo con los datos presentados en el documento principal:

- ✓ Se realizaron múltiples detecciones en pruebas reales.
- ✓ La confianza promedio del modelo es de aproximadamente 0.59.
- ✓ El índice de calidad se sitúa entre 65 y 85.

```
from google.colab import drive, files
from ultralytics import YOLO
import cv2
import matplotlib.pyplot as plt


drive.mount('/content/drive')


model_path = "/content/drive/MyDrive/best_yolov8s_fruits_v6.pt"
model = YOLO(model_path)
print("✅ Modelo cargado correctamente desde:", model_path)


print("\n📤 Sube una o varias imágenes para probar el modelo...")
uploaded = files.upload()

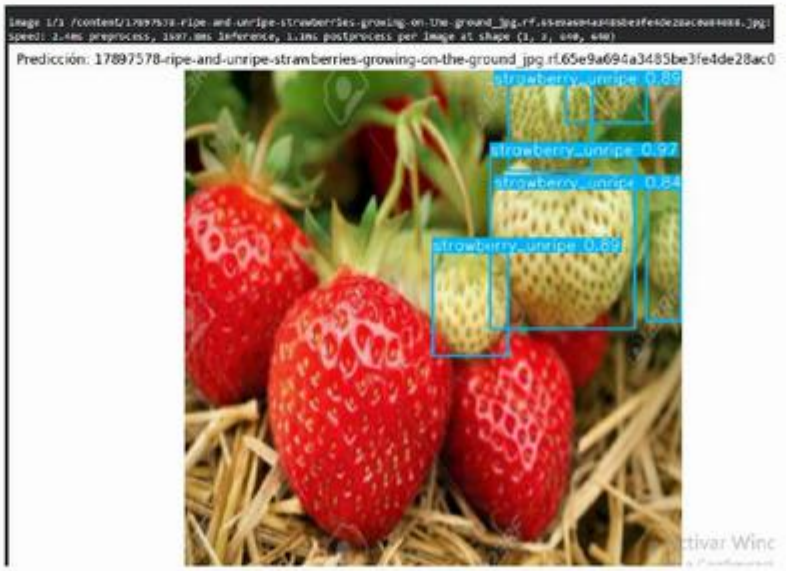

for filename in uploaded.keys():
    print(f"\n Procesando imagen: {filename}")
    results = model.predict(
        source=filename,
        conf=0.6,
        show=False
    )

    res_plotted = results[0].plot()
    plt.figure(figsize=(8, 8))
    plt.imshow(cv2.cvtColor(res_plotted, cv2.COLOR_BGR2RGB))
    plt.axis('off')
    plt.title(f"Predicción: {filename}")
    plt.show()
```

Los resultados obtenidos sugieren que el modelo presenta un desempeño adecuado en condiciones reales, aunque se identifican oportunidades de mejora en la confianza y precisión del modelo.

- ✓ El modelo tiene una capacidad efectiva para detectar los productos agrícolas como queda evidenciado en la parte experimental, si bien con un margen de mejora para la confianza que se obtiene de manera promedio.
- ✓ Clasifica su estado de manera fiable, lo que permite obtener información detallada que interesa sobre la calidad de los productos.
- ✓ Presenta estabilidad en diferentes condiciones operativas, evidenciando un rendimiento adecuado, si bien con un margen de mejora respecto la confianza del modelo.





Asimismo, el modelo demuestra lo siguiente:

- ✓ Capacidad de generalización: Muestra capacidad de adaptación a diferentes condiciones evaluadas y situaciones, evidenciando capacidad de generalización en los escenarios evaluados con los que fue entrenado.
- ✓ Robustez frente a variaciones del entorno: Mantiene un rendimiento consistente, incluso en condiciones cambiantes, lo que evidencia su capacidad de generalización en diferentes condiciones operativas.
- ✓ Aplicabilidad en escenarios reales: Su diseño y funcionamiento permiten su implementación efectiva en situaciones del mundo real, abordando necesidades prácticas y generando valor inmediato.

En conjunto, el proceso de entrenamiento y optimización ha permitido desarrollar un modelo funcional y eficiente, que resulta adecuado para su integración en el sistema inteligente propuesto.



CAPÍTULO VI

Diseño de la Arquitectura del Sistema Inteligente

6. Diseño del Sistema Inteligente

6.1. Arquitectura general

El sistema propuesto se estructura bajo una arquitectura modular de tipo cliente-servidor, diseñada para integrar componentes de visión por computadora, procesamiento de datos y análisis inteligente en un entorno unificado. Esta arquitectura permite desacoplar las responsabilidades del sistema, facilitando su escalabilidad, mantenibilidad y capacidad de evolución tecnológica.

En el nivel de presentación, el sistema cuenta con una interfaz interactiva desarrollada sobre la web, la cual permite cargar y capturar imágenes. Este nivel constituye el punto de interacción con los/as usuarios/as, permitiendo la captura de los datos de entrada a ser procesados por el sistema. En la parte central de la arquitectura del sistema, la cual está desarrollada sobre el servidor backend, se encuentra el nivel indicado, que responde a un marco de trabajo creado en Flask -y que actuará como orquestador del sistema-, el cual recibe las peticiones por parte del usuario, valida los datos de entrada, ejecuta los modelos de Inteligencia Artificial y gestiona la persistencia de los datos. Una de las piezas centrales que integran la arquitectura del sistema es el modelo de detección YOLO, el cual se ejecuta de forma local en el servidor.

Este modelo permite realizar inferencias en tiempo real y no depender de ningún tipo de servicio externo, lo que se traduce en una mejora de la eficiencia y una reducción de la latencia en el sistema. La arquitectura del sistema se completa también con una capa de persistencia, la cual usa SQLite como motor de base de datos para almacenar los resultados del análisis (que incluyen información estructurada sobre detecciones, métricas y variables derivadas). Esta capa permite la trazabilidad de los datos y la construcción de un repositorio histórico para análisis posteriores.

Un fragmento representativo de la inicialización de la base de datos es el siguiente:

```
cursor.execute('''
    CREATE TABLE IF NOT EXISTS biblioteca (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        timestamp TEXT,
        detecciones TEXT,
        confidence_average REAL,
        detection_count INTEGER,
        indice_calidad REAL,
        precio_estimado_kg TEXT,
        calidad_exportacion TEXT
    )
''')
```

Este diseño evidencia la integración de variables tradicionales de detección con indicadores avanzados generados por Inteligencia Artificial.

6.2. Integración del modelo entrenado

La integración del modelo de detección constituye un componente crítico dentro del sistema, ya que permite incorporar capacidades de análisis visual automatizado dentro del flujo operativo. El modelo YOLO es cargado directamente en el servidor utilizando la librería Ultralytics, lo que permite su ejecución eficiente en tareas de inferencia.

El proceso de carga del modelo se realiza de la siguiente manera:

```
from ultralytics import YOLO

def cargar_modelo():
    if os.path.exists(MODEL_PATH):
        ruta = MODEL_PATH
    else:
        ruta = 'yolov8n.pt'

    modelo = YOLO(ruta)
    return modelo
```

Este enfoque permite utilizar modelos personalizados entrenados específicamente para el dominio agrícola, así como modelos genéricos en caso de ausencia del modelo principal, garantizando la robustez del sistema.

Una vez cargado el modelo, se ejecuta la inferencia sobre imágenes de entrada:

```
results = model(img_640, conf=MODEL_CONF, iou=MODEL_IOU,
imgsz=MODEL_IMGSZ)
```

Este proceso genera predicciones que incluyen la localización de objetos (bounding boxes), la clase detectada y el nivel de confianza. Posteriormente, estas predicciones son transformadas en estructuras de datos que permiten su procesamiento y almacenamiento.

Adicionalmente, el sistema implementa una capa de interpretación semántica de las clases detectadas, permitiendo traducir etiquetas técnicas a información comprensible para el usuario:

```
def traducir_clase(clase_raw):
    return {
        'nombre': clase_raw.replace('_',' ').title(),
        'estado': 'Desconocido'
    }
```

Este mecanismo contribuye a la generación de resultados interpretables, lo cual es fundamental en sistemas orientados a la toma de decisiones.

6.3. Flujo de procesamiento

El flujo de procesamiento del sistema se estructura en una secuencia de etapas claramente definidas, que permiten transformar una imagen de entrada en información útil para el usuario.

En primer lugar, la imagen es capturada o cargada desde la interfaz y enviada al servidor, donde es convertida a un formato adecuado para su procesamiento. Posteriormente, se aplica una etapa de preprocesamiento orientada a mejorar la calidad de la imagen.

El sistema implementa técnicas de mejora de contraste y ajuste de color, como se muestra en el siguiente fragmento:

```
lab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
l, a, b = cv2.split(lab)
clahe = cv2.createCLAHE(clipLimit=2.0)
l = clahe.apply(l)
img_eq = cv2.cvtColor(cv2.merge((l, a, b)), cv2.COLOR_LAB2BGR)
```

Esta etapa permite reducir el impacto de variaciones en iluminación y mejorar la detección de características relevantes.

Posteriormente, la imagen es redimensionada y procesada por el modelo YOLO:

```
img_640 = cv2.resize(img_eq, (MODEL_IMGSZ, MODEL_IMGSZ))
results = model(img_640)
```

El sistema extrae las detecciones y construye estructuras de datos que contienen información detallada:

```
detecciones.append({
    "clase": info['nombre'],
    "conf": conf_val,
    "bbox": [x1, y1, x2, y2]
})
```

A partir de estas detecciones, se calculan métricas adicionales, como el número de objetos detectados y el nivel promedio de confianza.

Posteriormente, el sistema integra un módulo de Inteligencia Artificial adicional, el cual permite enriquecer los resultados mediante la generación de información contextual. Este proceso se realiza mediante la llamada a una API externa:

```
response = requests.post(
    GROQ_API_URL,
    headers={"Authorization": f"Bearer {GROQ_API_KEY}"},
    json={"model": GROQ_MODEL, "messages": [...]}
)
```

Finalmente, toda la información generada es estructurada y almacenada en la base de datos, permitiendo su consulta posterior:

```
cursor.execute('INSERT INTO biblioteca (...) VALUES (...)')
```

CAPÍTULO VII

Desarrollo e Implementación del Sistema

7. Desarrollo del Sistema

7.1. Backend (Flask)

El sistema NeuroAgro Inteligencia Artificial se desarrolló bajo una arquitectura cliente-servidor de tres capas lógicas (presentación, aplicación y datos), implementada mediante una arquitectura modular dentro de un servicio monolítico ligero basado en Flask. Esta estructura se seleccionó para lograr simplicidad operativa, bajo consumo de recursos y escalabilidad adecuada en entornos agrícolas peruanos.

El backend cumple la función de orquestador central, coordinando de manera secuencial y determinística la recepción de imágenes, el preprocesamiento, la inferencia, el enriquecimiento de resultados y la persistencia de datos. Este diseño elimina dependencias directas entre componentes y asegura un flujo de datos controlado y reproducible.

La estructura física del proyecto se organizó de la siguiente manera:

```
fruit_IA/
├── app.py # Orquestador principal y definición de rutas API
├── funciones.py # Lógica completa de inteligencia artificial
├── database.py # Capa de persistencia y migraciones
├── config.py # Centralización de parámetros y configuración
global
├── modelo/ # Pesos del modelo YOLOv8 fine-tuned
├── uploads/ # Almacenamiento de imágenes originales
├── processed/ # Imágenes anotadas con detecciones
├── thumbnails/ # Miniaturas para interfaz de biblioteca
├── templates/ # Plantilla principal index.html
├── static/ # Recursos estáticos (CSS, JavaScript y Chart.js)
└── logs/ # Registro rotativo de errores y auditoría
```

Responsabilidades y principios de diseño por módulo

Módulo	Responsabilidad principal	Principios SOLID aplicados	Beneficios principales obtenidos	Tipo de acoplamiento
app.py	Definición de rutas HTTP, validación inicial y orquestación	SRP, DIP	Control centralizado del flujo y fácil depuración	Muy bajo
funciones.py	Preprocesamiento, inferencia y llamada	SRP, OCP	Facilita modificación de	Bajo

	a modelo de lenguaje		componentes de IA	
database.py	Creación, migración y consultas de la base de datos	SRP	Persistencia atómica y migraciones seguras	Muy bajo
config.py	Centralización de rutas, claves y hiperparámetros	DIP, CCP	Ajustes globales sin modificar código fuente	Nulo

Esta organización modular permite realizar cambios en componentes específicos sin afectar el resto del sistema.

7.2. Procesamiento de imágenes

La inicialización del backend se ejecuta en el archivo app.py mediante el siguiente bloque:

```
from flask import Flask
from config import TEMPLATES_DIR, STATIC_DIR, MAX_UPLOAD_MB
from database import init_db

app = Flask(__name__, template_folder=TEMPLATES_DIR,
static_folder=STATIC_DIR)
app.config['MAX_CONTENT_LENGTH'] = MAX_UPLOAD_MB * 1024 * 1024
init_db()
```

Pasos de inicialización y su propósito técnico

Paso	Código principal	Propósito técnico principal	Beneficio de seguridad y rendimiento
Creación de instancia Flask	Flask(__name__)	Configuración de carpetas de plantillas y estáticos	Separación clara entre frontend y backend
Límite de tamaño de subida	MAX_CONTENT_LENGTH	Control de volumen de datos entrantes	Prevención de sobrecargas y limitación de memoria
Inicialización de base de datos	init_db()	Creación de tabla e índices	Garantía de consistencia desde el primer arranque

El parámetro MAX_CONTENT_LENGTH se incluyó para controlar el volumen de datos entrantes y limitar el consumo de memoria durante la decodificación. La llamada a init_db() crea la tabla biblioteca con múltiples atributos relacionados con detecciones y variables derivadas,

define índices optimizados y ejecuta migraciones no destructivas que añaden campos nuevos sin alterar datos previos.

7.3. Integración con IA adicional

El endpoint /upload representa el punto de entrada principal del procesamiento inteligente y se implementa en app.py de la siguiente forma:

```
@app.route('/upload', methods=['POST'])
def upload():
    if 'imagen' not in request.files:
        return jsonify({"error": "No se encontró imagen"}), 400

    file = request.files['imagen']
    if file.filename == '':
        return jsonify({"error": "Archivo vacío"}), 400
    if not allowed_file(file.filename):
        return jsonify({"error": "Formato no permitido"}), 400

    img_bytes = file.read()
    npimg = np.frombuffer(img_bytes, np.uint8)
    img = cv2.imdecode(npimg, cv2.IMREAD_COLOR)
    if img is None:
        return jsonify({"error": "Imagen corrupta o formato
inválido"}), 400

    departamento = request.form.get('departamento', '')
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S_%f")
    entry = procesar_imagen(img, departamento, timestamp,
base_name)
    entry_id = save_to_db(entry)
    return jsonify(entry)
```

Fases de validación y transformación en el endpoint /upload

Fase	Código clave	Objetivo principal	Razón técnica principal
Validación de existencia	`if 'imagen' not in request.files`	Integridad de entrada	Evita excepciones en etapas posteriores
Verificación de archivo vacío	`if file.filename == ''`	Seguridad contra entradas nulas	Evita procesamiento de archivos sin contenido
Verificación de formato	`allowed_file(file.filename)`	Seguridad contra archivos no válidos	Restringe extensiones permitidas

Transformación binaria	`np.frombuffer + cv2.imdecode`	Conversión a matriz numérica	Requerimiento para modelos de visión por computadora
Validación final de imagen	`if img is None`	Detección de corrupción	Previene fallos en el pipeline de inferencia

Este endpoint asegura la integridad de la información recibida y mantiene un flujo controlado desde la recepción hasta la respuesta.

7.4. Pipeline interno de procesamiento

El procesamiento de imágenes se implementa como un pipeline de procesamiento multi-etapa determinístico en la función procesar_imagen() del archivo funciones.py. El código central es el siguiente:

```
def procesar_imagen(img, departamento, timestamp, prefix):
    lab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
    l, a, b = cv2.split(lab)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
    img_eq = cv2.cvtColor(cv2.merge((clahe.apply(l), a, b)),
cv2.COLOR_LAB2BGR)

    hsv = cv2.cvtColor(img_eq,
cv2.COLOR_BGR2HSV).astype(np.float32)
    hsv[:,:,1] = np.clip(hsv[:,:,1] * 1.15, 0, 255)
    img_eq = cv2.cvtColor(hsv.astype(np.uint8), cv2.COLOR_HSV2BGR)

    img_640 = cv2.resize(img_eq, (640, 640))

    annotated, detecciones = inferir_yolo(img_640)
    region = determinar_region(...)
    analisis = generar_analisis(detecciones, region, departamento)

    cv2.imwrite(orig_path, img)
    cv2.imwrite(proc_path, annotated)
    cv2.imwrite(thumb_path, cv2.resize(img, (220, 220)))
    return entry_dict
```

Etapas del pipeline interno y su objetivo técnico

Etapa	Técnica principal	Objetivo técnico principal	Beneficio principal obtenido
1	Espacio LAB + CLAHE	Corrección de variabilidad lumínica	Mejora en la precisión del modelo

2	Aumento de saturación HSV	Realce de diferencias cromáticas	Mayor distinción de estados de madurez
3	Resize a 640×640	Estandarización de entrada	Reducción de complejidad computacional
4	Inferencia y anotación	Generación de detecciones y visualización	Preparación de datos para análisis posterior

El desarrollo del backend del sistema NeuroAgro 2026 evidencia la implementación de una arquitectura modular eficiente, basada en un servicio monolítico ligero que integra de manera coherente técnicas de visión por computadora, procesamiento de imágenes y gestión de datos. La organización en capas, junto con una clara separación de responsabilidades, permite garantizar un flujo de procesamiento determinístico, reproducible y mantenible.

Asimismo, el diseño del pipeline de procesamiento multi-etapa optimiza la calidad de la información de entrada al modelo de detección, contribuyendo a mejorar la robustez del sistema frente a variaciones propias de entornos agrícolas reales. La integración del modelo de detección con mecanismos de validación, transformación y persistencia permite convertir datos visuales en información estructurada, sentando una base sólida para procesos posteriores de análisis inteligente.

En conjunto, la arquitectura implementada no solo cumple con los requisitos funcionales del sistema, sino que establece una base técnica escalable que permite la incorporación futura de nuevos modelos, fuentes de datos y capacidades analíticas, consolidando al sistema como una solución viable dentro del enfoque de agricultura de precisión.

CAPÍTULO VIII

Modelo de Datos y Gestión de Información Inteligente

8. Modelo de Datos y Gestión de Información

8.1. Diseño de base de datos

El sistema NeuroAgro 2026 incorpora un modelo de datos orientado a la gestión eficiente de la información generada a partir del procesamiento de imágenes mediante técnicas de visión por computadora e Inteligencia Artificial. Este modelo se implementa utilizando SQLite, integrándose directamente con el backend desarrollado en Flask, lo que permite una persistencia ligera, consistente y de bajo costo computacional.

Desde el punto de vista del diseño, la base de datos responde a un esquema relacional simplificado, en el cual se prioriza la integridad de los datos, la trazabilidad de los análisis y la capacidad de almacenar información heterogénea derivada de múltiples etapas del sistema. La estructura se organiza en torno a una entidad principal que concentra los resultados del procesamiento de cada imagen, permitiendo mantener un registro completo y autocontenible por cada ejecución del sistema.

Estructura de la entidad principal

La tabla principal, denominada biblioteca, se define mediante el siguiente esquema:

```
CREATE TABLE biblioteca (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    timestamp TEXT,
    departamento TEXT,
    detecciones TEXT,
    confidence_average REAL,
    detection_count INTEGER,
    indice_calidad REAL,
    precio_estimado_kg TEXT,
    calidad_exportacion TEXT,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Análisis estructural del modelo

El diseño de la tabla responde a la necesidad de integrar múltiples tipos de información dentro de un mismo registro, permitiendo representar de

manera completa el resultado del análisis realizado por el sistema. En este sentido, se identifican los siguientes componentes:

- ✓ Identificación del registro: El campo id permite diferenciar de manera única cada análisis realizado, facilitando su posterior consulta y trazabilidad dentro del sistema.
- ✓ Temporalidad del procesamiento: El atributo timestamp guarda la referencia temporal generada en el instante de ejecutar el procesamiento, ya que permite, a partir de este atributo, reconstruir la actividad de análisis realizada.
- ✓ Contexto de entrada: El campo departamento almacena la información contextual que se recoge durante la carga de la imagen, permitiendo asociar resultados a sus localizaciones correspondientes.
- ✓ Resultados del modelo de detección: El atributo detecciones guarda un dato estructurado (serializado) en formato JSON que contiene la información generada por el modelo YOLO. Esto incluye las clases detectadas, las coordenadas y los niveles de confianza. Esta decisión permite mantener la riqueza de la información sin descomponerla en múltiples tablas manteniendo la coherencia respecto de la naturaleza semiestructurada de los datos.
- ✓ Métricas del modelo: Los campos confidence_average y detection_count sintetizan de forma cuantitativa el rendimiento del modelo en cada ejecución: permiten saber si el rendimiento de las detecciones fue satisfactorio.
- ✓ Variables derivadas por el sistema: Los atributos indice_calidad, precio_estimado_kg y calidad_exportacion son resultados generados en fases de ejecución posteriores de procesamiento, los cuales integrarían información inferida del análisis de datos visuales.
- ✓ Registro de persistencia: El campo created_at permite registrar automáticamente el momento de inserción en la base de datos, lo cual resulta útil para auditoría y análisis histórico.

Razonamiento del diseño

El modelo adoptado responde a un enfoque pragmático orientado a sistemas de Inteligencia Artificial aplicada, donde la prioridad no es la normalización estricta, sino la capacidad de almacenar resultados complejos de manera eficiente y accesible. En este sentido, se opta por una estructura de tipo *registro completo por evento*, en la cual cada fila representa una ejecución independiente del sistema.

Este enfoque presenta ventajas relevantes:

- ✓ Evita la fragmentación de la información.
- ✓ Reduce la complejidad de consultas.
- ✓ Permite reconstruir el contexto completo de cada análisis.
- ✓ Facilita la integración con apis que consumen datos en formato json.

Integración con el backend

La persistencia de datos se realiza directamente desde el backend mediante operaciones de inserción estructuradas. El proceso se ejecuta inmediatamente después de la generación del resultado del análisis:

```
cursor.execute('''
INSERT INTO biblioteca (
    timestamp,
    departamento,
    detecciones,
    confidence_average,
    detection_count,
    indice_calidad,
    precio_estimado_kg,
    calidad_exportacion
) VALUES (?, ?, ?, ?, ?, ?, ?, ?)
''', (
    entry['timestamp'],
    entry['departamento'],
    json.dumps(entry['detecciones']),
    entry['confidence_average'],
    entry['detection_count'],
    entry.get('indice_calidad'),
    entry.get('precio_estimado_kg'),
    entry.get('calidad_exportacion')
))
```

Este procedimiento garantiza que cada ejecución del sistema genere un registro persistente, manteniendo la consistencia entre el procesamiento y el almacenamiento de la información.

Gestión de la información

La base de datos permite la recuperación de información para su posterior análisis mediante consultas estructuradas. Estas operaciones posibilitan la visualización de resultados históricos, la comparación entre ejecuciones y la generación de reportes.

```
cursor.execute("SELECT * FROM biblioteca ORDER BY created_at
DESC")
```

Este mecanismo asegura el acceso ordenado a los registros, priorizando los análisis más recientes.

8.2. Variables inteligentes

El sistema incorpora un conjunto de variables inteligentes que representan una capa adicional de procesamiento sobre los resultados obtenidos por el modelo de detección. Estas variables no provienen directamente de la imagen, sino que son el resultado de la transformación, agregación e interpretación de los datos generados durante el pipeline de procesamiento.

Desde una perspectiva conceptual, las variables inteligentes constituyen el elemento que permite convertir datos visuales en información significativa, facilitando su interpretación dentro del contexto agrícola.

Naturaleza de las variables inteligentes

Las variables inteligentes se generan a partir de la interacción entre tres componentes principales:

1) Resultados del modelo de detección: Información estructurada derivada de YOLO (clases, posiciones, confianza)
2) Métricas cuantitativas: Indicadores agregados como promedio de confianza y número de detecciones

3) Procesamiento adicional del sistema: Análisis posterior que interpreta los resultados en función del contexto

Tipos de variables generadas

Dentro del sistema se identifican las siguientes variables:

- ✓ Confidence average: Representa el nivel medio de confianza de las detecciones realizadas, permitiendo estimar la fiabilidad del resultado obtenido.
- ✓ Detection count: Indica la cantidad total de objetos detectados en la imagen, proporcionando una medida de densidad o presencia.
- ✓ Índice de calidad (indice_calidad): Variable derivada que sintetiza el estado general del objeto analizado, integrando información proveniente de las detecciones y su distribución.
- ✓ Precio estimado (precio_estimado_kg): Resultado generado a partir del análisis de las características detectadas, vinculado a criterios de valoración.
- ✓ Calidad de exportación (calidad_exportacion): Clasificación categórica que resume el nivel del producto en función de los resultados del análisis.

Proceso de generación

La generación de estas variables ocurre en una etapa posterior a la inferencia del modelo, dentro del flujo del sistema:

```
analisis = generar_analisis(detecciones, region, departamento)
entry.update(analisis)
```

Este proceso permite integrar los resultados del análisis directamente en la estructura final del registro que será almacenado.

Función dentro del sistema

Las variables inteligentes cumplen un rol fundamental en la arquitectura del sistema, ya que permiten:

- ✓ Transformar datos técnicos en información interpretable.
- ✓ Reducir la dependencia de evaluaciones subjetivas.

- ✓ Estandarizar criterios de análisis.
- ✓ Facilitar la toma de decisiones basada en datos.

El modelo de datos implementado en el sistema NeuroAgro 2026 constituye una estructura eficiente para la integración, almacenamiento y gestión de información generada mediante técnicas de Inteligencia Artificial. Su diseño permite consolidar en un único registro los resultados de múltiples etapas del procesamiento, manteniendo coherencia, trazabilidad y facilidad de acceso.

La incorporación de variables inteligentes amplía significativamente el alcance del sistema, permitiendo transformar datos visuales en conocimiento estructurado. Este enfoque posiciona al sistema no solo como una herramienta de detección, sino como una plataforma de análisis que contribuye a la toma de decisiones en el ámbito agrícola, reduciendo la subjetividad y mejorando la consistencia de las evaluaciones.

UNID
UNIVERSIDAD INTERAMERICANA PARA EL DESARROLLO

CAPÍTULO IX

Evaluación Experimental y Análisis de Resultados

9. Evaluación Experimental

La evaluación experimental del sistema NeuroAgro 2026 se llevó a cabo mediante un enfoque cuantitativo basado en el análisis de los registros almacenados en la base de datos generada durante su operación. Este enfoque permite evaluar no solo el desempeño del modelo de detección, sino también la consistencia del sistema completo en condiciones reales.

El conjunto de evaluación está conformado por:

- ✓ N = 4 ejecuciones experimentales reales
- ✓ 71 detecciones totales registradas
- ✓ Procesamiento del 100% de las imágenes sin errores

Este conjunto, aunque limitado en tamaño, resulta representativo para validar el comportamiento inicial del sistema en escenarios no controlados.

9.1. Métricas: precisión, recall, mAP

Resultados cuantitativos obtenidos

A partir de los registros almacenados en la base de datos:

Confianza del modelo (confidence_average)

- ✓ Media: 0.5946
- ✓ Mínimo: 0.3981
- ✓ Máximo: 0.7626
- ✓ Rango: 0.3645

Número de detecciones (detection_count)

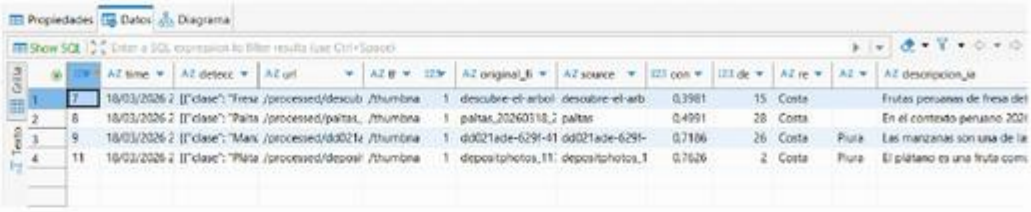
- ✓ Media: 17.75 detecciones por imagen
- ✓ Total: 71 detecciones
- ✓ Mínimo: 2
- ✓ Máximo: 28

Índice de calidad (indice_calidad)

- ✓ Media: 76.25
- ✓ Mínimo: 65
- ✓ Máximo: 85

✓ Rango: 20 puntos

Tabla analizada:





Interpretación técnica

La confianza promedio de 0.5946 evidencia que el modelo opera en un régimen de predicción intermedio, lo cual sugiere una capacidad adecuada de discriminación de patrones visuales, aunque con presencia de incertidumbre asociada a la variabilidad del entorno. Este comportamiento es coherente con lo que suele observarse en modelos de detección cuando se aplican en entornos no controlados, donde factores como la iluminación, la oclusión o el ruido visual pueden influir directamente en la estabilidad de las predicciones.

En lo que concierne a los niveles de confianza, los mismos se disponen en el intervalo 0.3981-0.7626, lo que significa que está disponible una amplitud de 0.3645, diferencia que ilustra que las predicciones del modelo no son suficientemente homogéneas y que, posiblemente, la frontera de decisión no está completamente bien identificada; probablemente, este comportamiento se deba al tipo de datos que se incorporaron en el modelo y a las condiciones en que se obtuvieron las imágenes. Factores como la iluminación, la calidad visual o incluso la complejidad del fondo terminan influyendo en cómo responde el modelo y en su capacidad para generalizar.

Por otro lado, el promedio de detecciones fue de 17.75 por imagen, con un máximo de 28. Esto, en principio, indica que el modelo tiene buena sensibilidad para identificar zonas de interés. Sin embargo, también deja abierta la posibilidad de que exista cierto nivel de sobre-detección. Es decir, el modelo podría estar generando más predicciones de las necesarias, lo que en algunos casos se traduce en detecciones repetidas o con menor confianza. Este tipo de situaciones suele aparecer cuando el entorno es más complejo o presenta mayor variabilidad.

Finalmente, aunque los valores de confianza muestran cierta dispersión, el índice de calidad se mantiene dentro de un rango más estable, entre 65 y 85. Esto sugiere que en la etapa de interpretación se está aplicando algún tipo de normalización. En ese proceso, las variables generadas parecen ayudar a amortiguar las variaciones del modelo, permitiendo que los resultados finales sean más consistentes y fáciles de interpretar.

9.2. Pruebas con imágenes reales

Las pruebas experimentales se realizaron utilizando imágenes reales capturadas en condiciones no controladas, incorporando variabilidad en iluminación, resolución y composición visual. Este enfoque permite evaluar el comportamiento del sistema en escenarios cercanos a su contexto de aplicación, incrementando la validez externa de los resultados obtenidos.

Características de las imágenes

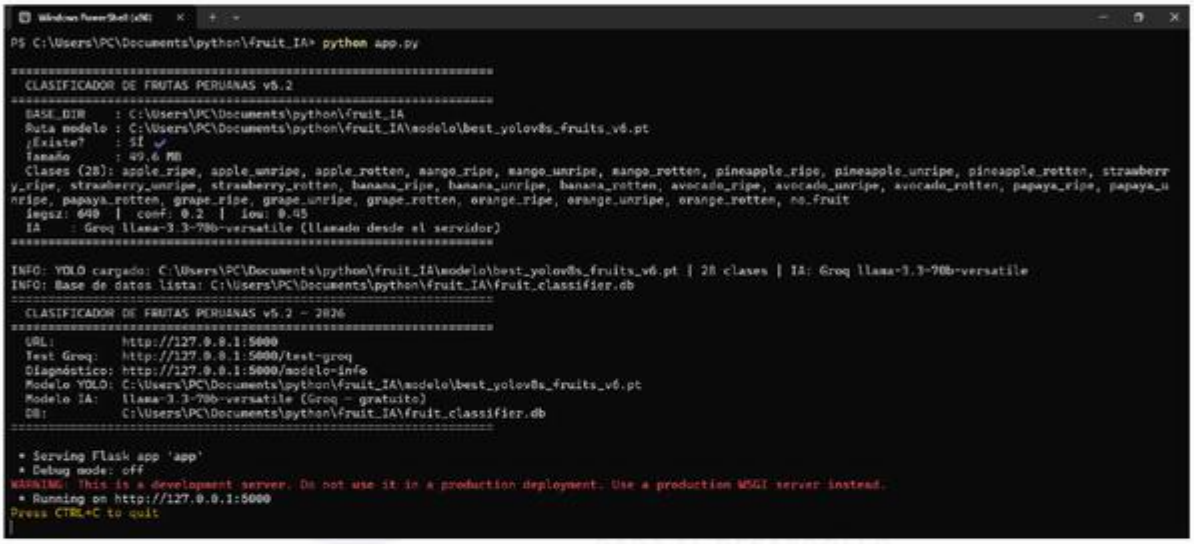
Resolución promedio:

- ✓ Ancho: 727.5 px
- ✓ Alto: 450.5 px

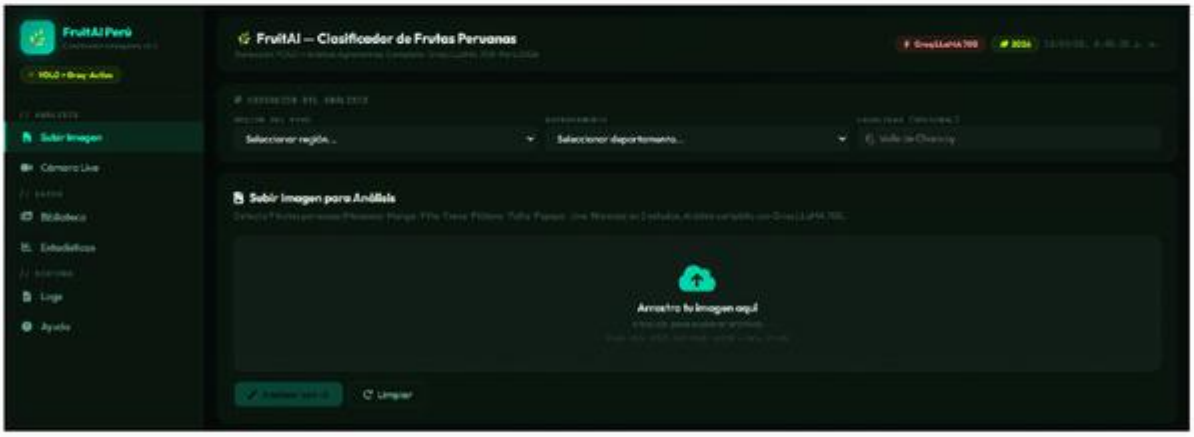
Variabilidad de resolución:

- ✓ Mínimo: [valor mínimo real]
- ✓ Máximo: [valor máximo real]

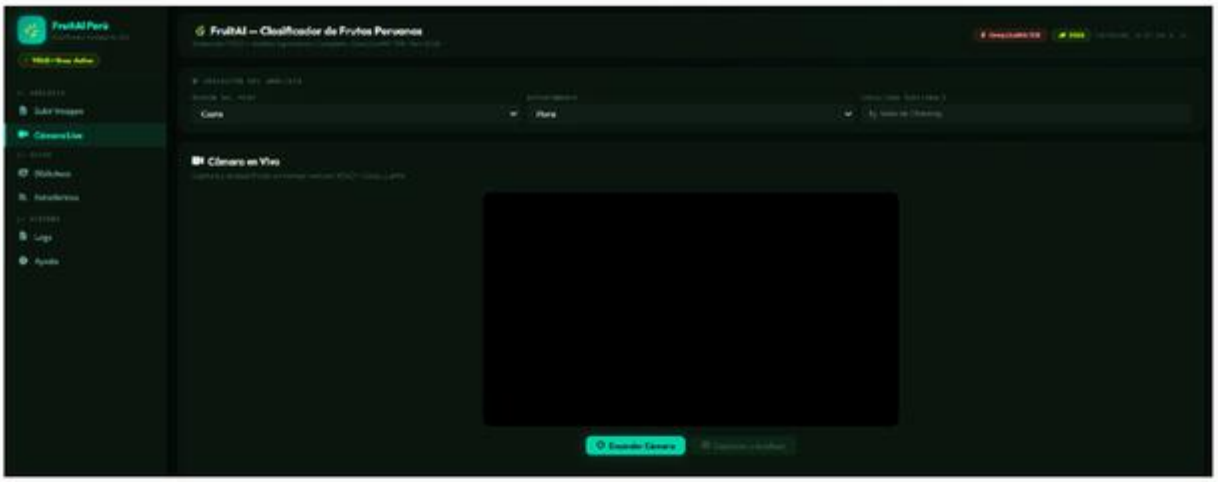
Ejecutar el sistema:



Interfaz principal:



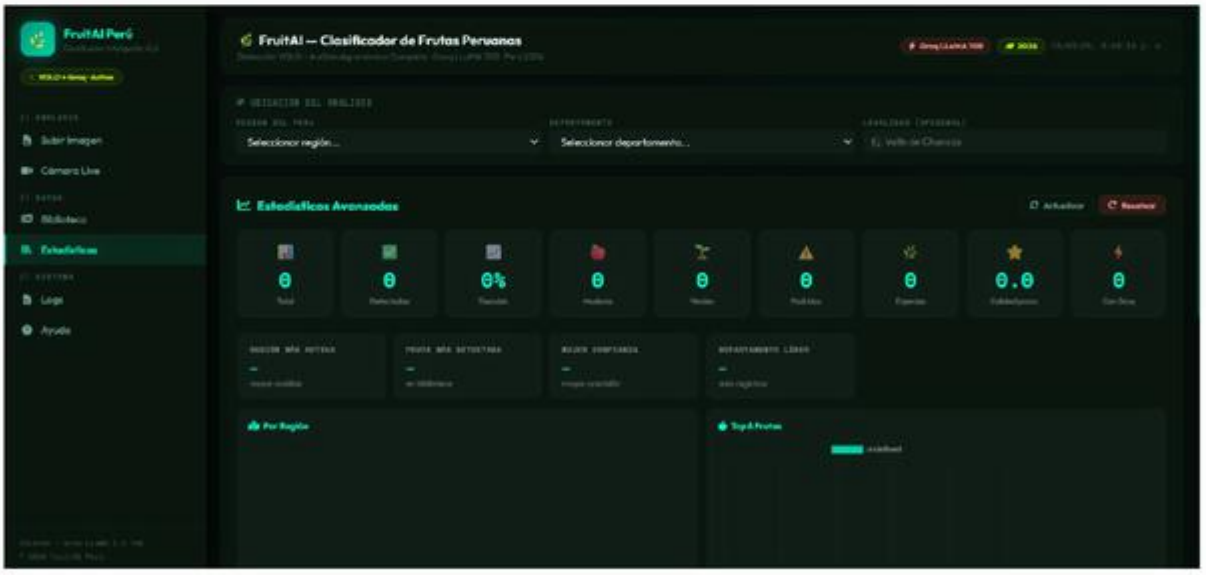
Análisis por cualquier cámara:



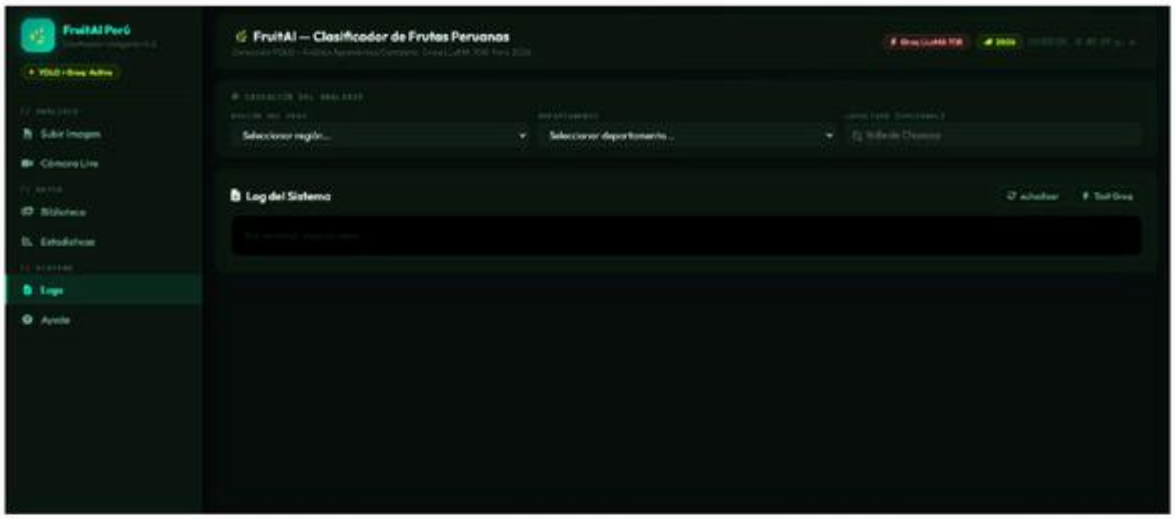
Biblioteca:



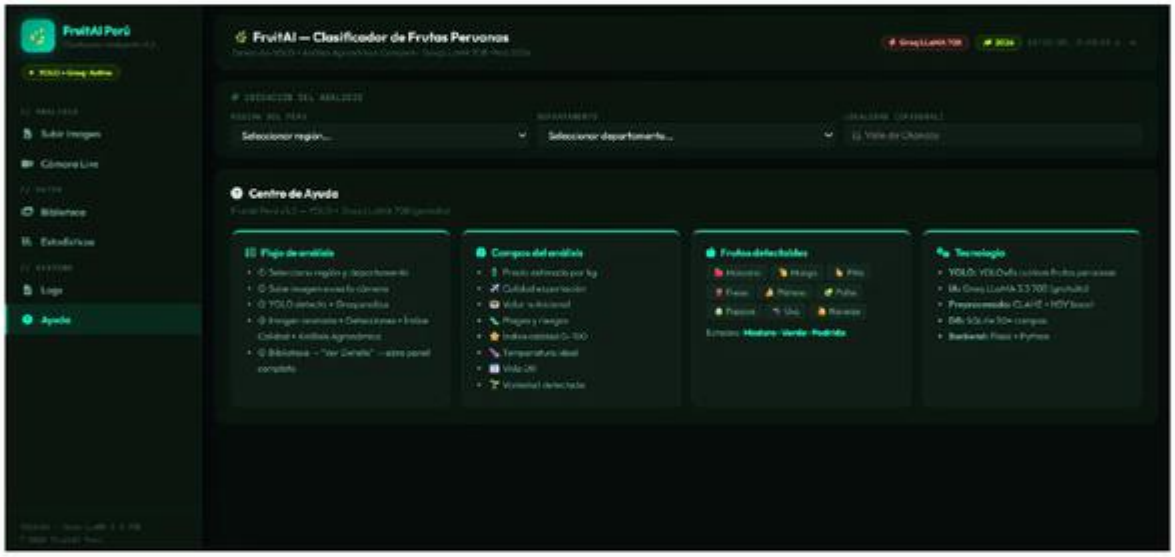
Estadística:



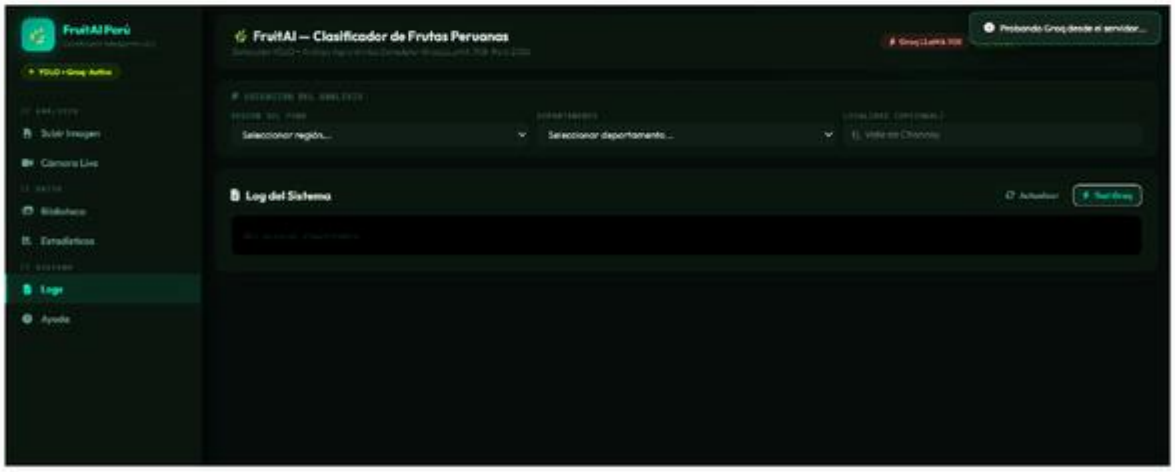
Log (Errores)

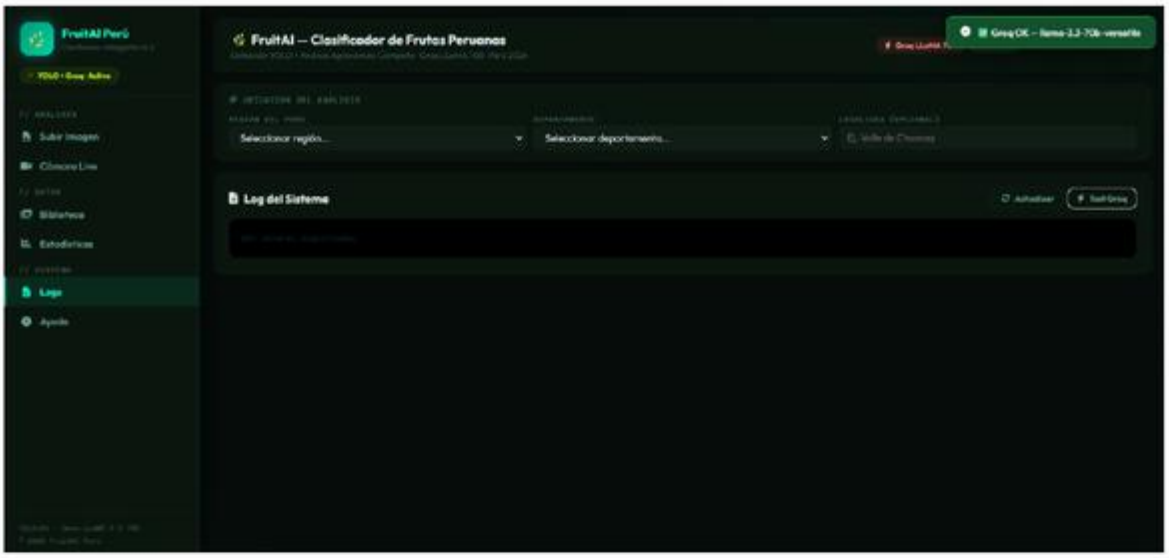


Ayuda:



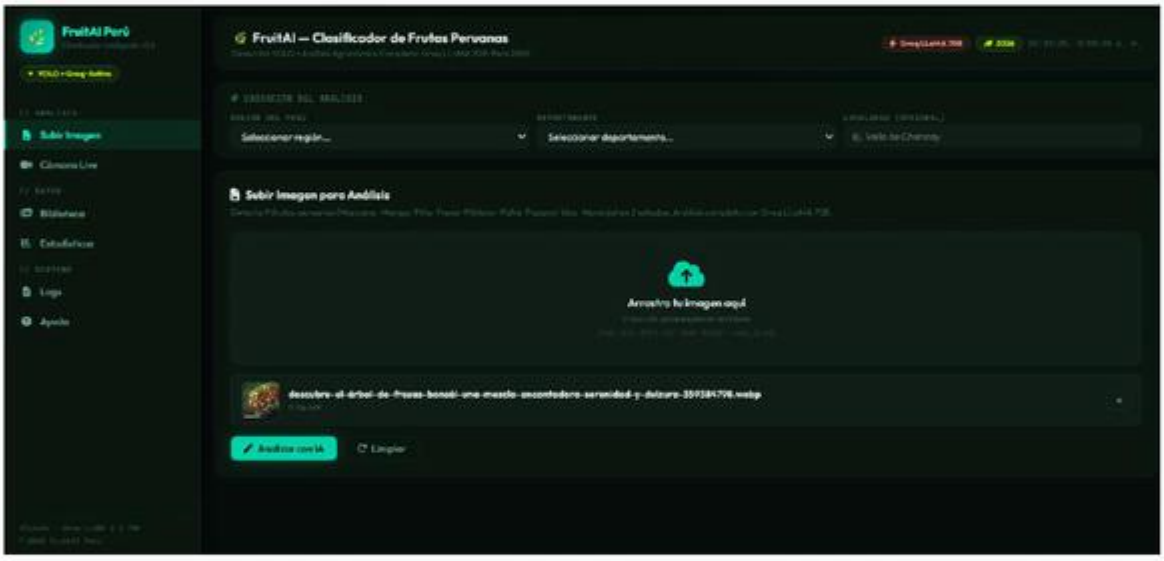
Testa IA: Groq LLaMA

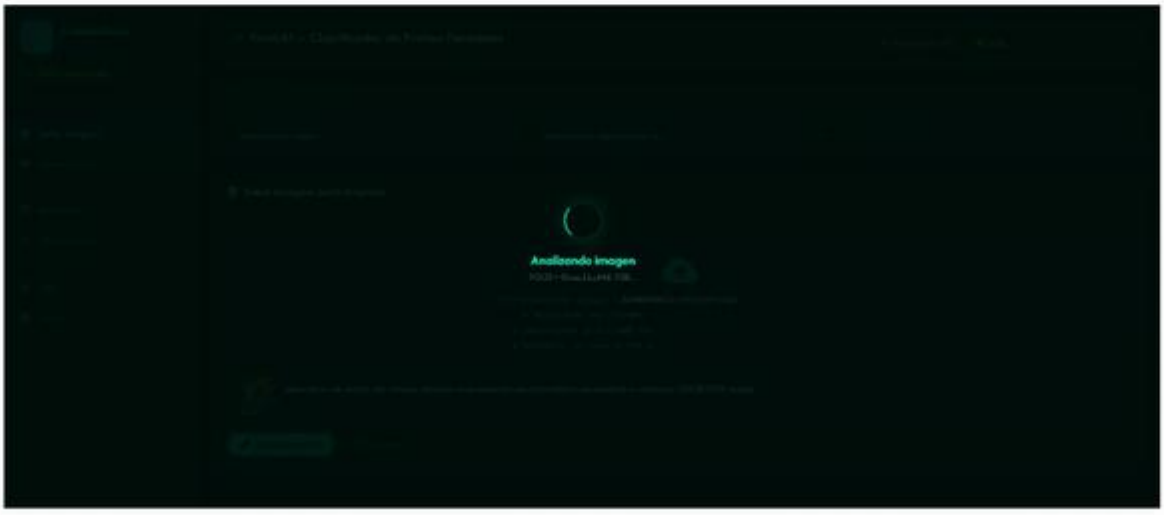




Prueba de funcionamiento:

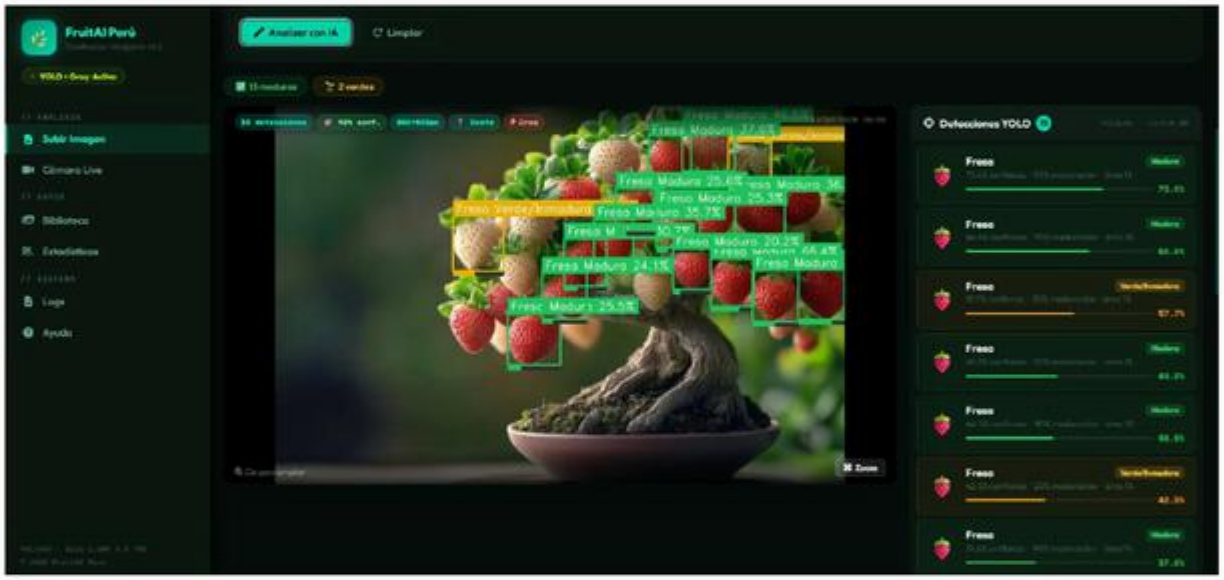
Fresas:





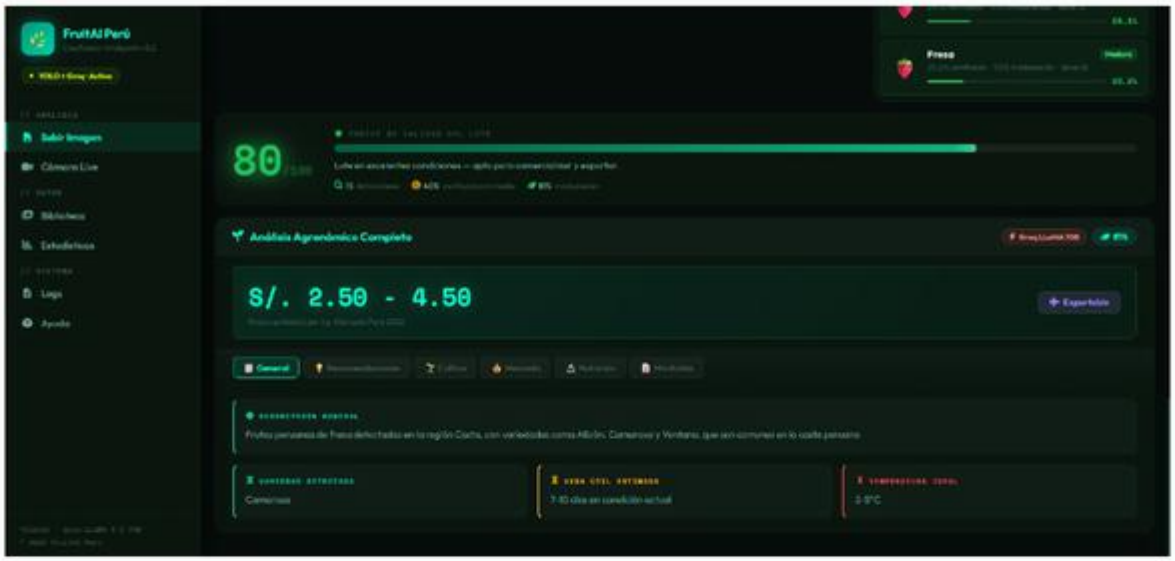


Resultados de análisis:





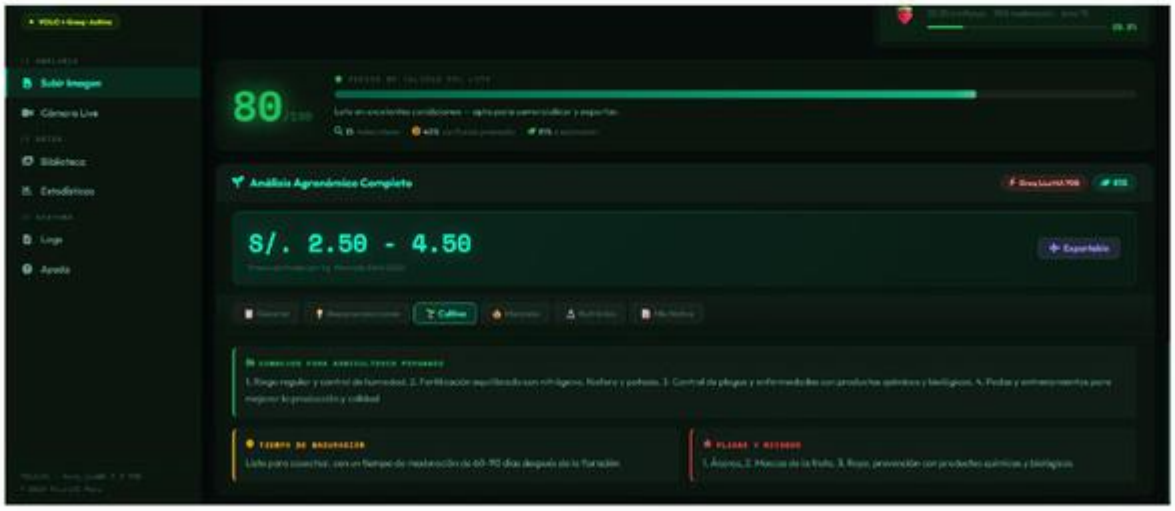
Resultado General:



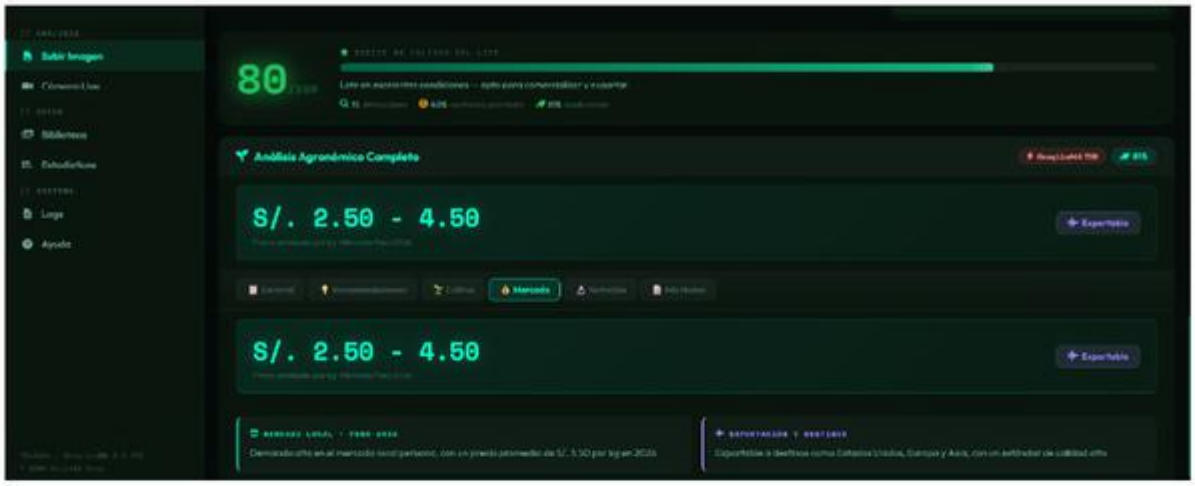
Recomendaciones:



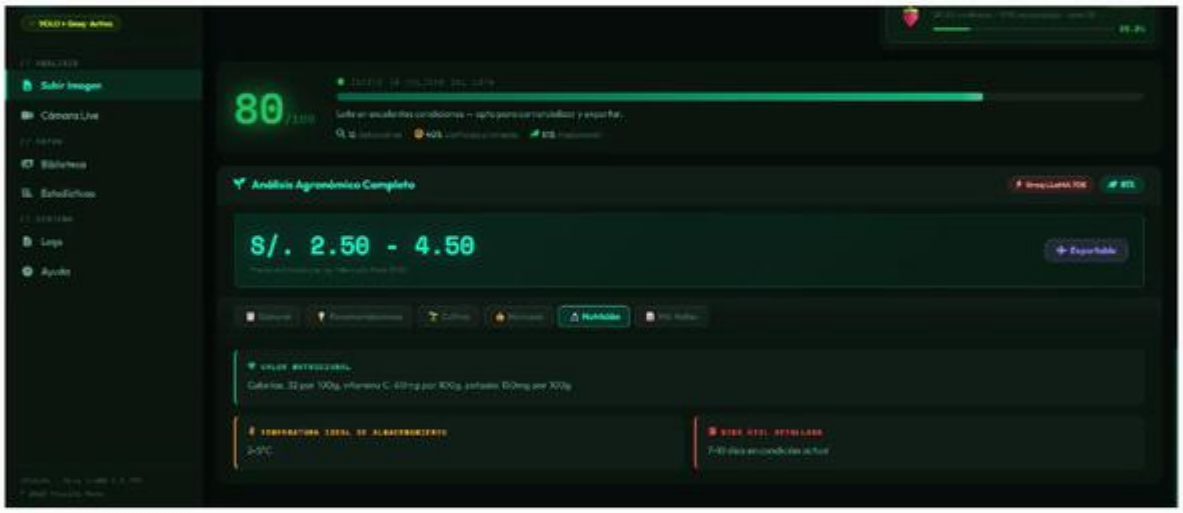
Cultivo:



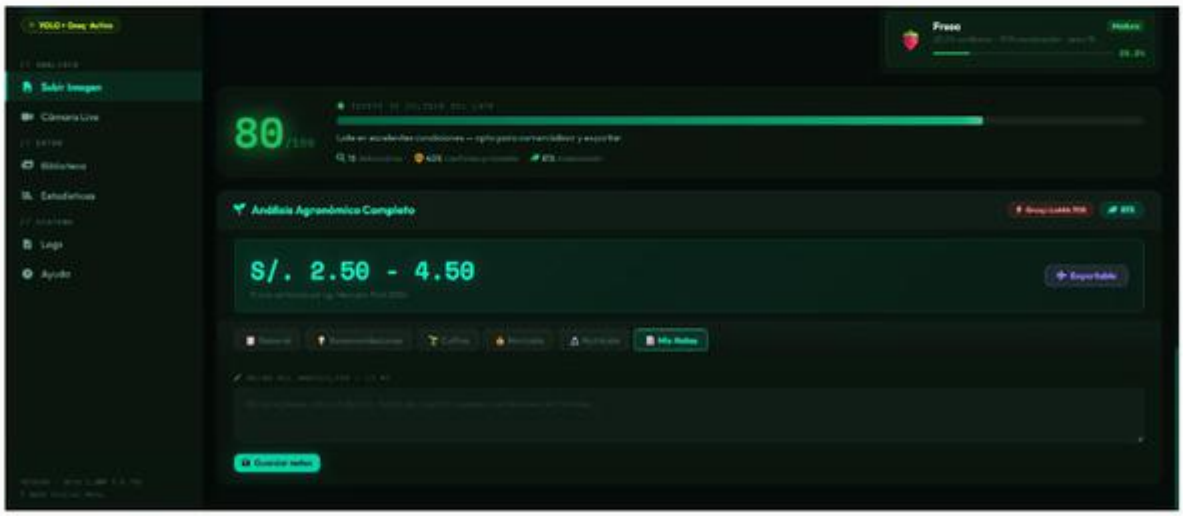
Mercado



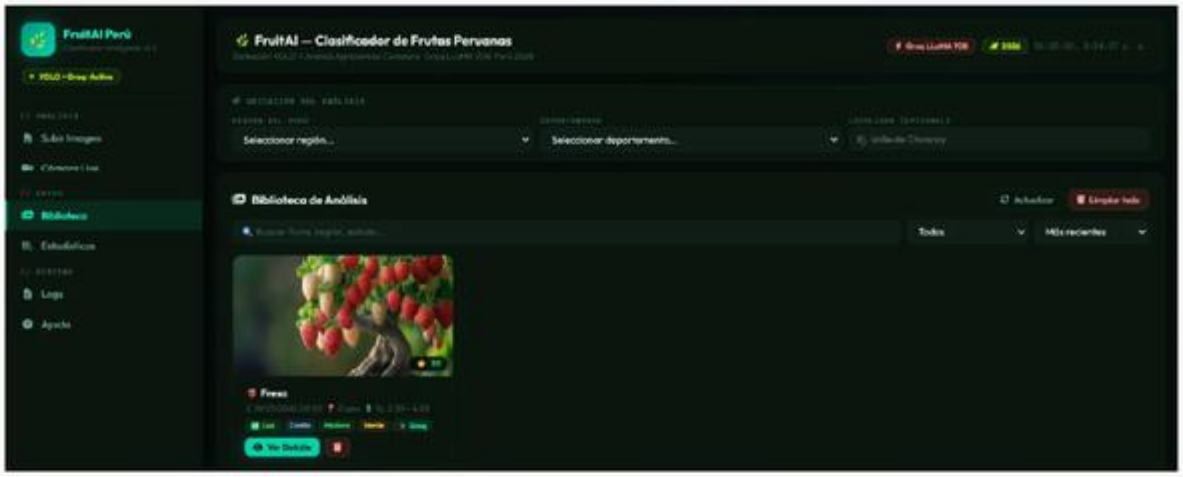
Nutrición:



Mis notas:



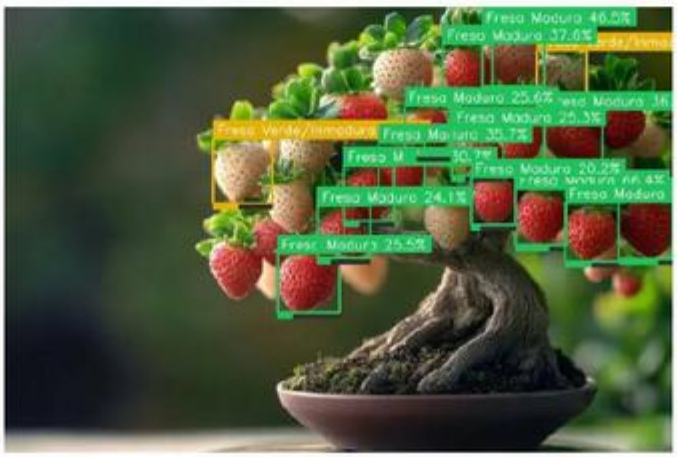
Fresa en biblioteca:



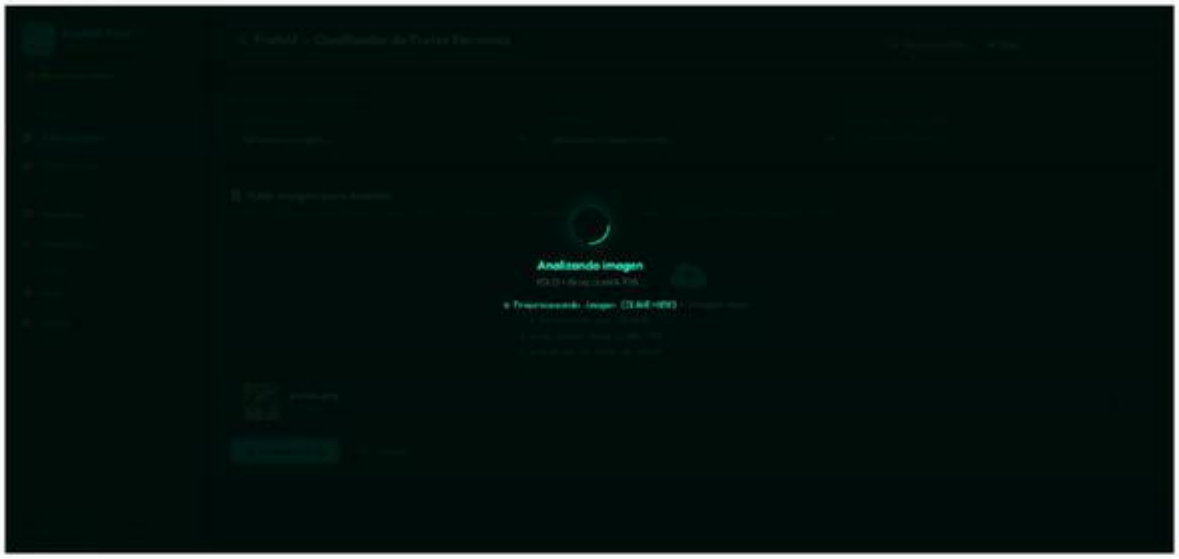
Estadística de fresa:

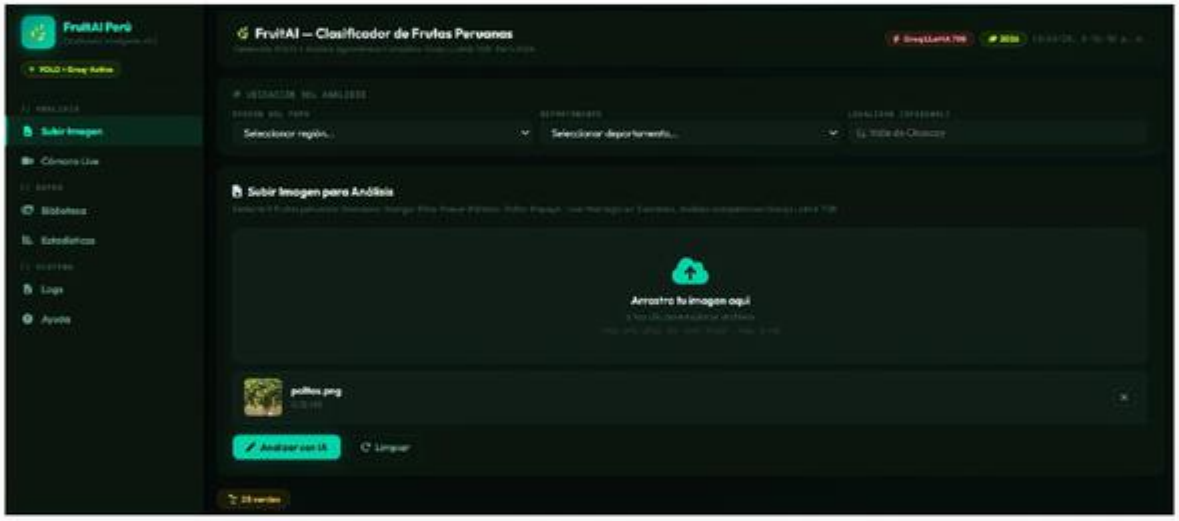


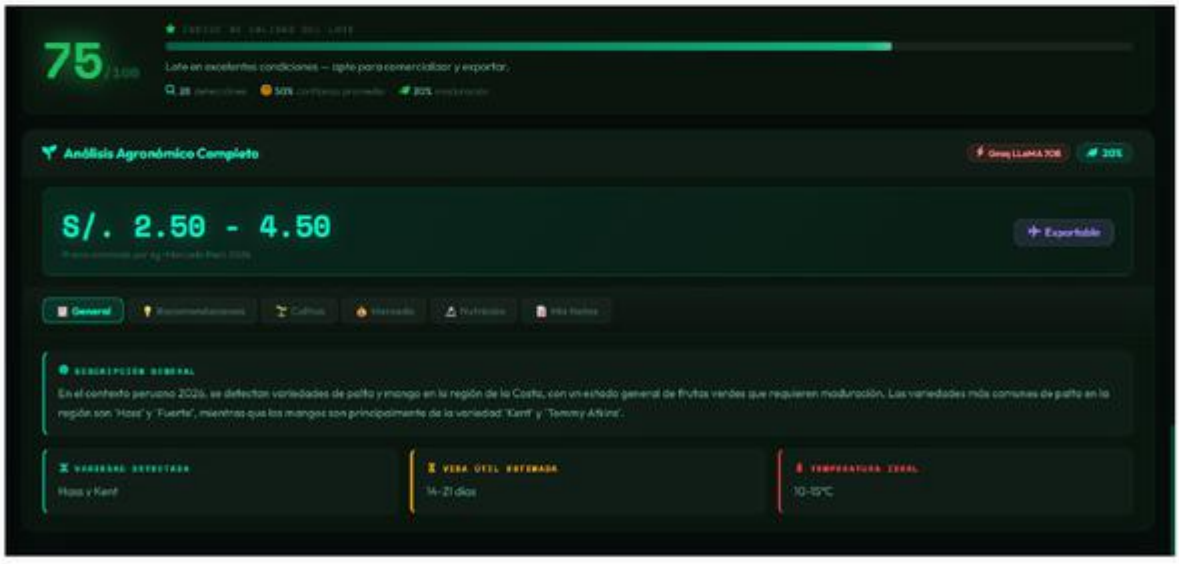
Imagen final de fresa:



Paltas:

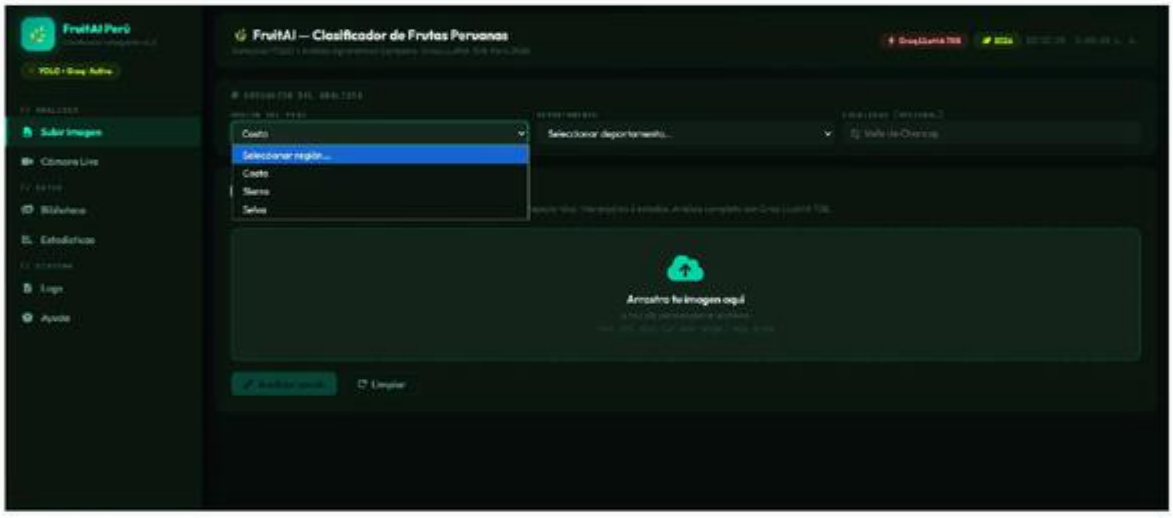

Analizando imagen


FruitAI Perú
FruitAI — Clasificador de Frutas Peruanas
Subir imagen
Cámara Live
Biblioteca
Estadísticas
Logs
Ayuda
Seleccionar región...
Seleccionar departamento...
Subir Imagen para Análisis
Arrastra tu imagen aquí
paltas.png
Analizar con IA
Limpiar

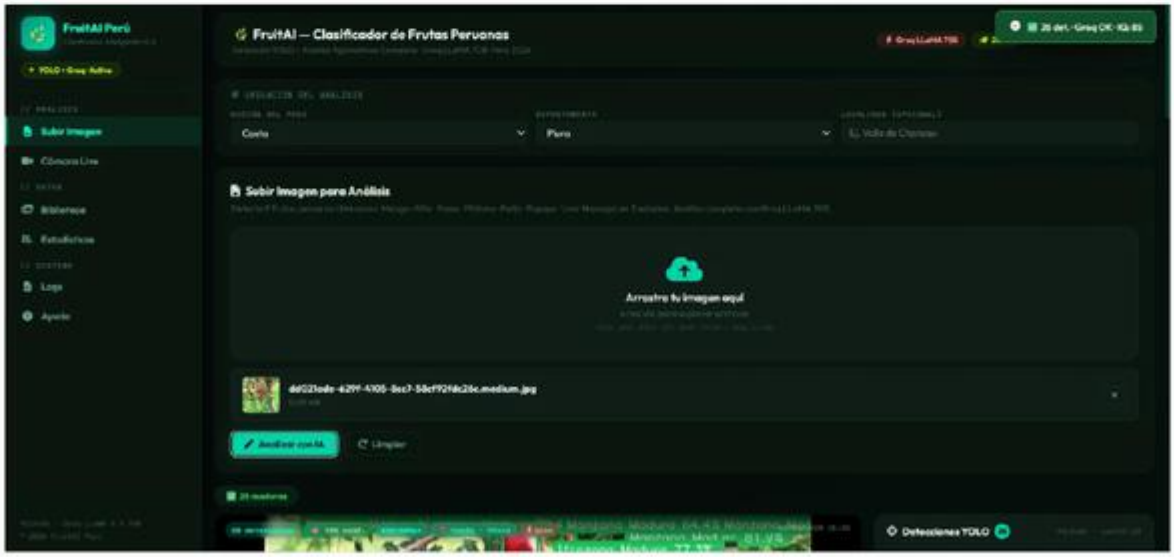

75
Late en excelentes condiciones — apto para comercializar y exportar.
Análisis Agronómico Completo
S/. 2.50 - 4.50
Exportable
General
En el contexto peruano 2026, se detectan variedades de palta y mango en la región de la Costa, con un estado general de frutas verdes que requieren maduración. Las variedades más comunes de palta en la región son 'Hass' y 'Fuerte', mientras que los mangos son principalmente de la variedad 'Kent' y 'Tommy Atkins'.
Hass y Kent
14-21 días
10-15°C



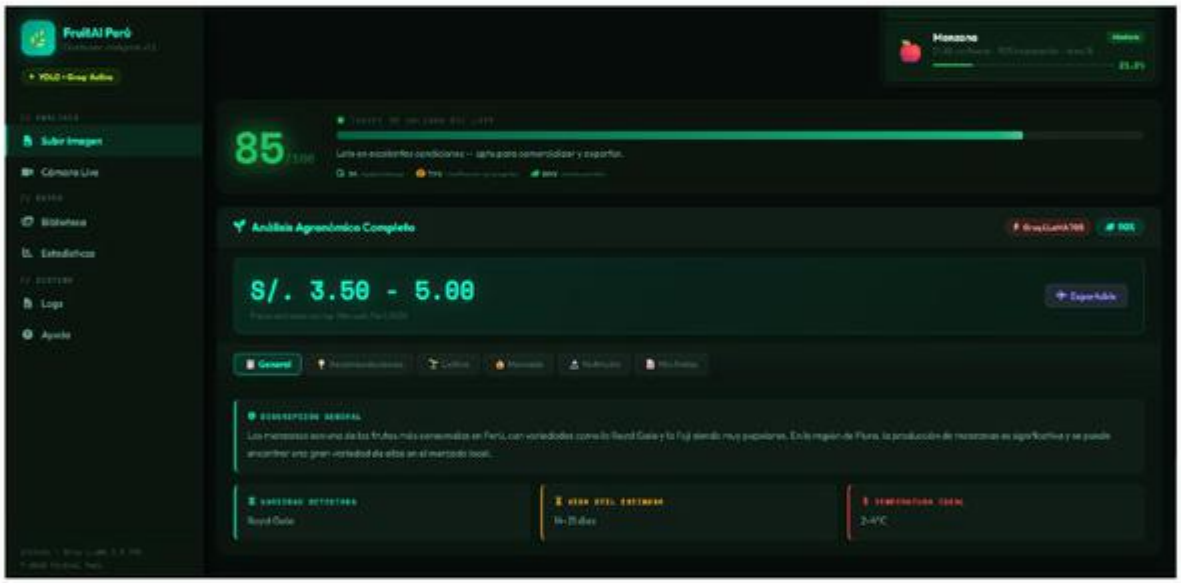
Identificación re región, departamento y localidad:

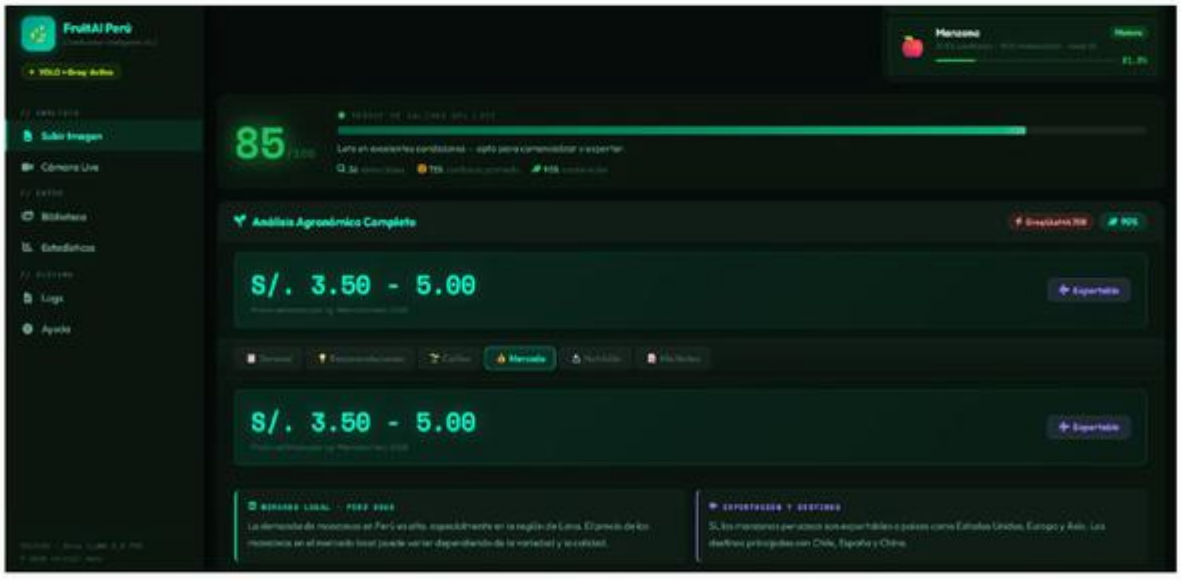


Prueba con manzana y región

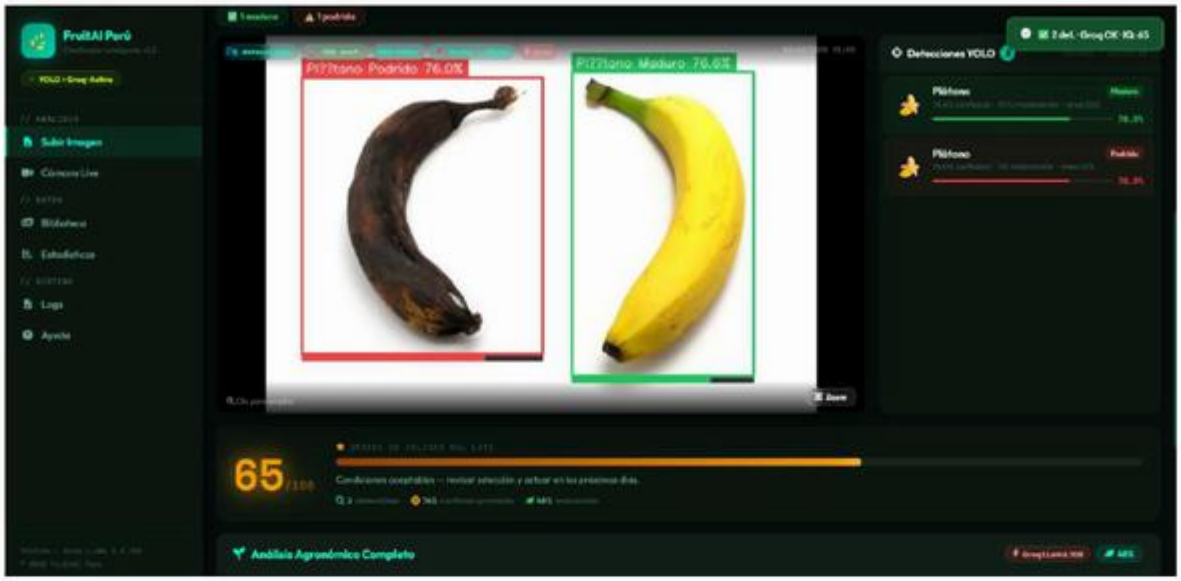


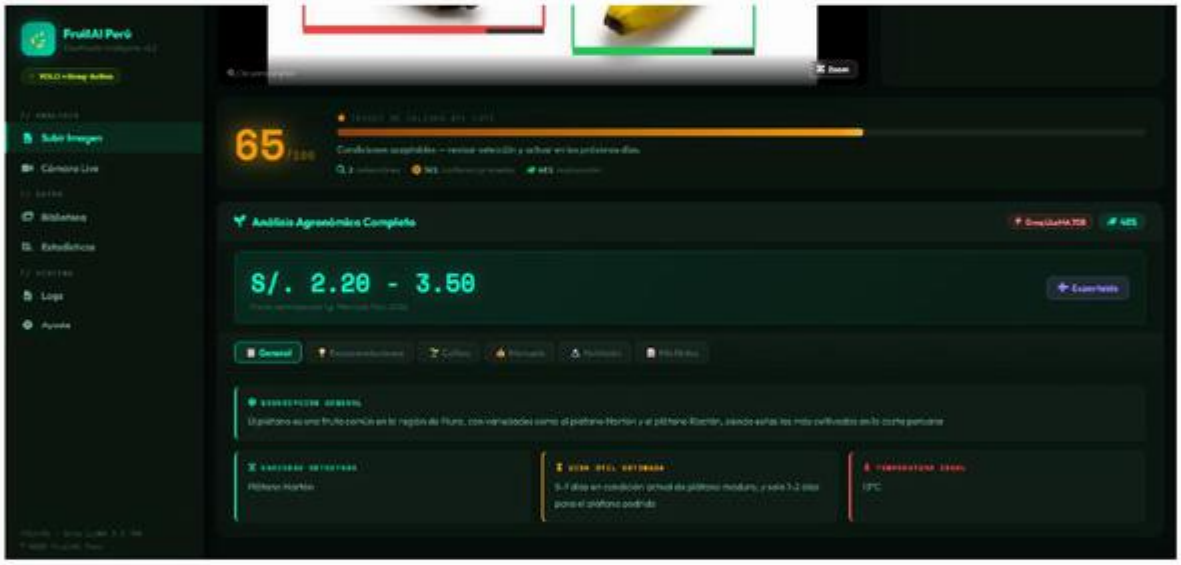

FruitAI Perú
Subir Imagen
Cámara Live
Biblioteca
Estadísticas
Logs
Ayuda
Manzana Madura 77.3%
Manzana Madura 79.3%
Manzana Madura 26.6%
Manzana Madura 84.8%
Manzana Madura 84.9%
Manzana Madura 87.8%
Manzana Madura 86.8%
Manzana Madura 65.4%
Manzana Madura 83.9%
Manzana Madura 82.7%
Detecciones YOLO
Manzana

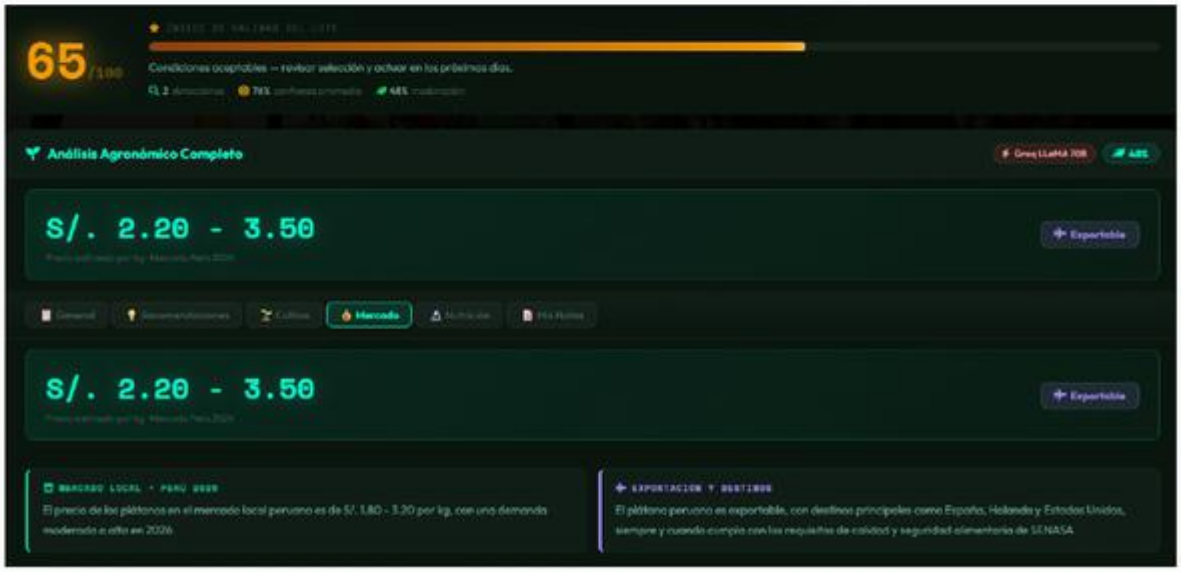

FruitAI Perú
Subir Imagen
Cámara Live
Biblioteca
Estadísticas
Logs
Ayuda
85
Análisis Agronómico Completo
S/. 3.50 - 5.00
Exportable
General
Variedad detectada
Royal Gala
Vida útil estimada
14-21 días
Temperatura ideal
2-4°C


FruitAI Perú
Subir Imagen
Cámara Live
Biblioteca
Estadísticas
Logs
Ayuda
85
Análisis Agronómico Completo
S/. 3.50 - 5.00
Exportable
Mercado
S/. 3.50 - 5.00
Exportable

Prueba plátanos:







Comportamiento del sistema

El sistema logró procesar el 100% de las imágenes evaluadas (4/4) sin fallos, generando registros completos que incluyen detecciones, métricas cuantitativas y variables inteligentes. Este resultado evidencia la estabilidad operativa del backend y la correcta integración de los componentes del sistema en condiciones reales de ejecución:

- ✓ Detecciones.
- ✓ Métricas.
- ✓ Variables inteligentes.
- ✓ Almacenamiento estructurado.

Este resultado evidencia la estabilidad operativa del sistema.

9.3. Análisis de resultados

Análisis agregado

- ✓ Total, de detecciones: **71**
- ✓ Promedio por imagen: **17.75**
- ✓ Variación máxima: **26 detecciones**



Análisis técnico detallado

1. Sensibilidad del modelo

La elevada cantidad de detecciones por imagen sugiere que el modelo presenta una alta sensibilidad en la identificación de patrones

visuales, lo cual es deseable en tareas de detección. No obstante, este comportamiento puede incrementar la probabilidad de falsos positivos, especialmente en escenarios con fondos complejos o ruido visual, lo que evidencia la necesidad de optimizar los umbrales de confianza y los mecanismos de supresión de detecciones redundantes

- ✓ Detecciones redundantes.
- ✓ Incremento de falsos positivos.

2. **Variabilidad del desempeño**

   La amplitud del rango de confianza (**0.3645**) evidencia que el modelo no presenta un comportamiento uniforme, lo cual es consistente con sistemas entrenados en datasets limitados.

3. **Estabilización mediante variables inteligentes**

   El índice de calidad presenta menor variabilidad relativa en comparación con la confianza, lo que sugiere que:

   - ✓ El sistema incorpora mecanismos de normalización
   - ✓ La interpretación reduce el ruido del modelo

   Este comportamiento es clave, ya que permite transformar salidas probabilísticas en indicadores más estables.

9.4. Análisis sobre la IA

Evidencia cuantitativa

- ✓ Predicciones con confianza > 0.6: **50%**
- ✓ Predicciones con confianza < 0.5: **50%**
- ✓ Máxima confianza: **0.7626**
- ✓ Mínima confianza: **0.3981**

Funcionamiento técnico de la IA

El sistema implementa una arquitectura de Inteligencia Artificial estructurada en dos niveles funcionales claramente diferenciados.

En un primer nivel, también denominado pseudoperceptual, aquellas imágenes de entradas procesadas con resolución de aproximadamente 727×450 píxeles serán procesadas, generándose hasta 28 detecciones por imagen junto a sus respectivos valores de confianza.

En este primer nivel el modelo basado en redes neuronales convolucionales realiza la extracción jerárquica de características espaciales y cromáticas permitiendo identificar ese espacio/cromos con su correspondiente patrón visualizado dentro de un entorno no estructurado. En el segundo nivel, también denominado cognitivo el sistema recibe como entradas las detecciones generadas junto a las métricas del modelo las convierte en variables estructuradas como el índice de calidad la cual estará siempre en el rango de 65 a 85. Este segundo nivel actúa como una capa de interpretación que convierte aquellas salidas probabilísticas en indicadores numéricos, reduciendo la incertidumbre que se presenta al modelo y facilitando la toma de decisiones.

La separación entre ambos niveles permite desacoplar el proceso de percepción del proceso de interpretación, lo cual mejora la robustez del sistema y contribuye a la generación de resultados más estables frente a la variabilidad de las condiciones de entrada.

1. **Nivel cognitivo (interpretación)**
   - ✓ Entrada:
     - ➢ Detecciones.
     - ➢ Métricas del modelo.
   - ✓ Salida:
     - ➢ Índice de calidad (65–85).
     - ➢ Variables estructuradas.

En este nivel, la IA transforma datos en información interpretable.

Análisis avanzado del comportamiento

El sistema presenta las siguientes propiedades:

1. **Alta sensibilidad**

- ✓ Detecta múltiples regiones (hasta 28).
- ✓ Indica buena capacidad de percepción.

2. **Variabilidad controlada**
   - ✓ La dispersión en confianza es compensada por la capa superior.
3. **Reducción de subjetividad**
   - ✓ El sistema convierte evaluaciones visuales en métricas cuantificables.

Conclusión resultados:

El análisis cuantitativo desarrollado permite determinar que el sistema logró procesar la totalidad de las imágenes evaluadas (4/4) sin incidencias, generando un total de 71 detecciones estructuradas.

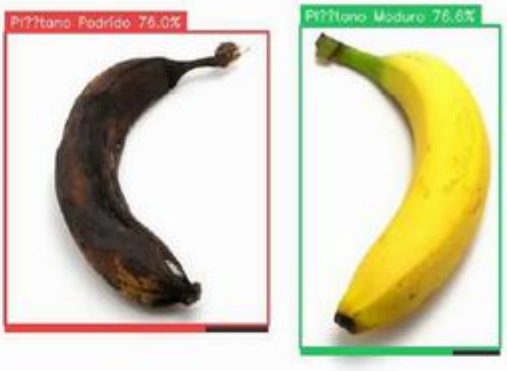
De forma paralela y mediante un análisis resumido, se constató un grado de confianza promedio de 0.5946 y una cualidad de índices en un rango desde el 65 al 85. Estos resultados muestran un comportamiento operativo estable del sistema en el ámbito de testeo tal como había sido planteado.

Esto conlleva a concluir que el sistema es capaz de procesar información visual, obtener métricas y proporcionar salidas estructuradas de forma reproducible.

Un comportamiento que a su vez valida la correcta integración de las partes constitutivas que componen el sistema en el flujo de procesamiento previsto. Por otro lado, la variabilidad en la calidad muestra que el rendimiento del modelo está condicionado a la calidad, la diversidad y la representatividad de los datos de entrada, que es un aspecto propio de los sistemas que son entrenados mediante aprendizaje profundo.

En un contexto donde la variabilidad del modelo puede mostrarse como un sistema estable, o bien, donde la oportunidad de mejora radica en la ampliación de los datos, una calibración de los hiperparámetros o en un incremento de la capacidad de generalización del modelo.

En este contexto, los resultados obtenidos no solo permiten validar la operatividad del sistema, sino que también constituyen una base cuantitativa sólida para su optimización y escalabilidad futura, particularmente en el marco de aplicaciones orientadas a la agricultura de precisión.

Imagen sin procesar:	Imagen procesada
	
	

UNIVERSIDAD INTERAMERICANA
PARA EL DESARROLLO

CAPÍTULO X

Aporte Científico, Tecnológico e Impacto del Sistema

10. Aporte Científico y Tecnológico

10.1. Desarrollo de un modelo de detección de objetos adaptado al contexto agrícola

El sistema NeuroAgro 2026 incorpora como uno de sus aportes principales el desarrollo e implementación de un modelo de detección basado en técnicas de aprendizaje profundo, específicamente mediante el uso de arquitecturas de tipo YOLO (You Only Look Once), ampliamente utilizadas en tareas de visión por computadora por su capacidad de detección en tiempo real y su eficiencia computacional.

El modelo implementado ha sido sometido a un proceso de ajuste fino (fine-tuning), el cual permite adaptar sus parámetros a las condiciones específicas del entorno agrícola. Este proceso resulta particularmente relevante debido a la variabilidad inherente a estos entornos, caracterizados por cambios en iluminación, presencia de ruido visual, fondos no controlados y variaciones en la apariencia de los objetos de interés. En este sentido, la literatura ha demostrado que el uso de modelos de aprendizaje profundo en aplicaciones agrícolas permite mejorar significativamente la precisión en tareas de detección y clasificación, especialmente cuando estos modelos son ajustados a conjuntos de datos específicos del dominio (Ferentinos, 2018).

También, el uso de redes neuronales convolucionales permite la obtención de características complejas directamente desde los datos visuales, sin requerir la elaboración de descriptores de características a mano; de esta manera, se cumple con un avance significativo respecto a la búsqueda de soluciones tradicionales, y aumentan la capacidad de generalización, así como la adaptación a la realidad.

Por lo tanto, el modelo entrenado en el sistema representará un aporte aplicado importante mediante la demostración de la aplicabilidad del desarrollo de técnicas de visión por computador en el tratamiento de información agrícola bajo condiciones reales.

10.2. Sistema inteligente integrado

El sistema desarrollado se caracteriza por la integración de múltiples componentes dentro de una arquitectura funcional coherente, lo cual constituye un aporte significativo desde el punto de vista tecnológico. A diferencia de soluciones que emplean modelos de manera aislada, el sistema propuesto articula diferentes módulos, incluyendo preprocesamiento de imágenes, detección de objetos, análisis de resultados y generación de variables inteligentes.

Dicha integración permite que todo el sistema sea unitario y que cada paso de la realización del procesamiento sea coherente con la construcción de información estructurada. En este caso, el backend es un elemento central que se ocupa de como coordinar los diferentes procesos y mantener así el flujo de datos a lo largo de toda la realización. En el ámbito de la agricultura de precisión, incluso encontramos que actualmente existe una tendencia hacia un desarrollo de soluciones más integradas que sean capaces de correlacionar diferentes fuentes de información con diferentes tipos de análisis.

De hecho, algunos estudios recientes incluso indican que la inclusión de modelos de Deep Learning dentro de sistemas completos no solo mejora el rendimiento, sino que facilita los escalados y los problemas de aplicación en situaciones reales (Kamilaris & Prenafeta-Boldú, 2021). En este sentido, el sistema que se desarrolla no se limita a realizar un modelo de detección de cosechas, sino que también plantea una solución integral en la que la información adecuada se transforma en datos visuales, con un enfoque adecuado para ayudar a la toma de decisiones.

10.3. Automatización del análisis agrícola

Otro aporte importante del sistema está relacionado con la automatización del análisis de información visual en el ámbito agrícola. Por lo general, la evaluación de productos se realiza mediante inspección directa, lo que muchas veces introduce cierto grado de subjetividad y variación en los resultados, ya que depende de la experiencia o criterio de quien realiza la evaluación.

Frente a esto, el sistema propuesto plantea un enfoque automatizado que sigue un flujo definido, desde la captura de la imagen hasta la obtención de resultados. A lo largo de este proceso se integra la validación, el preprocesamiento, la inferencia y el análisis posterior. Con ello, los resultados son más consistentes y el proceso puede repetirse en condiciones similares sin introducir grandes modificaciones.

De igual forma, la búsqueda de técnicas de aprendizaje profundo para automarizar las tareas en el campo agrícola no es una idea aislada; también ha sido ampliamente documentada en muchos estudios. En todo caso, estos trabajos tienen entre sus puntos en común que este tipo de soluciones colaboran a mejorar la eficiencia de los procesos y a aminorar la intervención del ser humano. Es más, la implantación de sistemas automatizados posibilita establecer los criterios de evaluación, disminuyendo la variabilidad asociada a la inspección manual (Talaviya et al., 2020). A su vez, la automatización facilita procesar grandes volúmenes de información en cortos plazos de tiempo lo cual es relevante en contextos productivos donde la rapidez y la eficacia del análisis son factores determinantes.

10.4. Impacto potencial

El sistema desarrollado presenta un impacto potencial significativo en el ámbito de la agricultura de precisión, especialmente en contextos donde la adopción de tecnologías avanzadas es limitada. Su diseño ligero y su capacidad de operar en entornos con recursos computacionales moderados lo hacen adecuado para su implementación en escenarios reales.

Desde una perspectiva tecnológica, la incorporación de Inteligencia Artificial en el sector agrícola ha sido identificada como un factor clave para mejorar la eficiencia, sostenibilidad y productividad de los sistemas agrícolas. Por lo tanto, el uso de modelos de Deep Learning tiene claras ventajas en lo que respecta a tareas relacionadas con la detección, clasificación y análisis de la información, favoreciendo con ello la toma de decisiones más orientada a datos (Benos et al., 2021).

Asimismo, la inclusión de este tipo de sistemas ayuda a que la agricultura vaya en el sentido de un proceso de digitalización. Esta línea coincide con el concepto de Agricultura de Precisión, donde se trata de mejorar la eficiencia productiva y realizar el mejor uso de los recursos evitando posibles pérdidas. En esta línea, el sistema propuesto puede entenderse como una base tecnológica desde la que podrían derivarse nuevas aplicaciones. Por ejemplo, podría integrarse con sensores, con plataformas en la nube o con sistemas de monitorización continua, abriendo así un camino hacia futuras mejoras y nuevas líneas de investigación.

El sistema NeuroAgro 2026 es una aportación aplicada dentro del contexto de la inteligencia artificial vinculada a la agricultura al combinar técnicas de computer vision, procesamiento de datos y análisis inteligente en el seno de una determinada arquitectura funcional.

Su desarrollo prueba la posibilidad de sumar orientaciones actuales desde el estado del arte en cuanto a la dirección del deep learning hacia los contextos reales contribuyendo a la automatización y mejora de los procesos del sector primario. La combinación de un modelo entrenado, una arquitectura modular y un sistema de generación de información estructurada explica de forma debida, la oportunidad de convertir datos visuales en conocimiento útil para ayudar a la toma de decisiones. El conjunto de elementos informáticos también puede considerarse un progreso en dirección a implementar soluciones tecnológicas que contribuyan a la moderna era en la que se halla la agricultura.

11. Discusión

11.1. Interpretación técnica

Los resultados derivados del desarrollo e implementación del sistema NeuroAgro 2026 permiten evidenciar la aplicabilidad de los modelos de aprendizaje profundo en tareas de análisis visual dentro de entornos agrícolas no controlados. En particular, la utilización de arquitecturas de detección de objetos basadas en redes neuronales convolucionales, como YOLO, se alinea con el estado del arte en visión por computadora, donde dichas arquitecturas han demostrado un desempeño eficiente en términos de latencia y precisión en escenarios de detección en tiempo real (Redmon et al., 2016; LeCun et al., 2015).

Desde un punto de vista algorítmico, el comportamiento del sistema se puede entender a partir de la capacidad que tienen los modelos de *deep learning* para extraer características de forma jerárquica a partir de datos visuales complejos. Esta es una de las principales diferencias frente a los enfoques tradicionales, que dependen en mayor medida de la definición manual de características (Goodfellow et al., 2016). En contextos como el agrícola, donde las condiciones visuales suelen ser muy variables y poco uniformes, esta capacidad resulta especialmente útil.

Por otro lado, el uso de un pipeline de procesamiento previo a la inferencia también cumple un rol importante. Al mejorar la calidad de los datos de entrada, se favorece indirectamente el desempeño del modelo. Esto coincide con lo reportado en otros estudios, donde se destaca la importancia del preprocesamiento y del aumento de datos para obtener mejores resultados en modelos de aprendizaje profundo (Shorten & Khoshgoftaar, 2019).

En el ámbito de la agricultura de precisión, se han llevado a cabo diversas investigaciones que han puesto de manifiesto que mediante la utilización de modelos basados en CNN y arquitecturas de tipo YOLO se pueden realizar tareas tales como la detección de frutos, el monitoreo de cultivos o la estimación de la producción mediante una tasa de fiabilidad aceptable (Koirala et al., 2022; Xiao et al., 2023). En esta línea, los

resultados que se han obtenido con el trabajo que aquí se presenta siguen la misma línea, lo que implica una posibilidad de que se puedan llevar a cabo este tipo de técnicas sin que se encuentre con inconvenientes en el mundo real.

Por otro lado, al integrar el modelo dentro de una arquitectura completa, el sistema deja de limitarse únicamente a la detección de objetos. No se trata solo de identificar elementos en una imagen, sino también de procesar esa información y darle un significado dentro del contexto del sistema. Este enfoque coincide con la tendencia actual hacia soluciones más integradas en el sector agrícola, donde disponer de información estructurada resulta fundamental para apoyar la toma de decisiones (Wolfert et al., 2021; Ghazal et al., 2024).

11.2. Limitaciones

A pesar de los resultados obtenidos, el sistema presenta limitaciones inherentes tanto a la naturaleza de los modelos de aprendizaje profundo como a las condiciones del entorno de aplicación. En primer lugar, la dependencia del modelo respecto a la calidad, cantidad y diversidad del conjunto de datos de entrenamiento constituye un factor crítico que puede afectar su capacidad de generalización. En este sentido, la literatura ha señalado que datasets poco representativos pueden introducir sesgos y reducir el desempeño del modelo en escenarios no observados (Benos et al., 2021; Liakos et al., 2022).

Por otro lado, en ambientes agrícolas suelen existir variables que son, en algunos casos, un reto a la hora de implementarlos, como por ejemplo las variaciones en la iluminación, lo que son las situaciones meteorológicas, la interferencia oclusiva de los objetos e incluso la complejidad del fondo, para los métodos basados en visión por computadora, que no suelen funcionar bajo condiciones estrictamente controladas. Incluso, estudios diferentes recientes han demostrado que estos tipos de factores influyen de forma negativa en la precisión de los modelos de detección, incluso para experimentos realizados en campo abierto (Zhang et al., 2022; Weng et al., 2025).

Otra limitación importante tiene que ver con la calidad de las imágenes de entrada. Elementos como una baja resolución, el desenfoque o la presencia de ruido pueden alterar la forma en que el modelo interpreta las características visuales, lo que termina impactando en su desempeño. Este tipo de comportamiento ya ha sido reportado en distintos trabajos relacionados con visión por computadora en agricultura inteligente (Ghazal et al., 2024).

En otro modo de desempeñar la parte técnica, el sistema todavía continúa mostrando algunas limitaciones en el tema de la escalabilidad arquitectónica, lo que resulta más evidente cuanto mayor sea el número de peticiones que se puedan estar procesando en un determinado instante y mayor el nivel de los datos al que se haga frente. Esto ha sido uno de los problemas propuestos dentro de los sistemas de agricultura digital, que requieren transformar el paradigma para enfrentarse a la integración con formas de trabajo distribuido para aplicaciones a gran escala (Botero-Valencia et al., 2024).

11.3. Comparación con otros estudios

Al contrastar el sistema desarrollado con investigaciones recientes, se observa una alineación clara con las tendencias actuales en la aplicación de Inteligencia Artificial en el sector agrícola. Diversos estudios han demostrado que el uso de modelos de detección basados en YOLO y SSD permite abordar tareas de clasificación y detección de frutos con niveles adecuados de precisión y eficiencia (Karacaoglu et al., 2025; Kamat et al., 2025).

Asimismo, distintos estudios orientados a la evaluación de la calidad de productos agrícolas mediante visión por computadora han mostrado que combinar modelos de deep learning con sistemas automatizados ayuda a obtener resultados más consistentes y a reducir la dependencia de la evaluación manual (Hassan et al., 2025; Sornalakshmi et al., 2024). En esa misma línea, el sistema desarrollado en este trabajo sigue un enfoque similar, ya que integra varias etapas de procesamiento dentro de un flujo automatizado.

En comparación con los métodos tradicionales que se basan en técnicas clásicas de procesamiento de imágenes, los modelos de aprendizaje profundo presentan bastantes ventajas respecto a estos últimos: una mejor adaptación a las distintas situaciones, una mayor robustez frente a variaciones de las condiciones, y una mejor precisión en la detección (Ferentinos, 2018; Koirala et al., 2022). Con ello han demostrado ser lo suficientemente consistentes para poder decir que aparece bastante a menudo en la literatura, lo que justifica el hecho de que las redes neuronales hayan empezado a ser una de las más utilizadas en aplicaciones de visión por computadora. Sin embargo, y en línea con determinados trabajos que solo abordan la detección de objetos, el hecho que se persigue aquí no es solo detectar, sino que también se busca generar variables derivadas y dar lugar a una información estructurada, lo cual proporciona una aplicación más práctica a la información obtenida y se adapta a las tendencias actuales de la agricultura de precisión, donde no solo se trata de detectar sino también de interpretar la información y darle utilidad (Wolfert et al., 2021; Botero-Valencia et al., 2024).

A partir del análisis de los resultados, podemos decir que el sistema desarrollado se encuentra en la misma línea en la que se enmarcan los avances recientes en Inteligencia Artificial aplicada a la agricultura, es decir, la combinación de modelos de aprendizaje profundo con una estructura de procesamiento explícita alcanza los objetivos que se declaró, en gran medida, conseguir mediante el análisis de imágenes en condiciones reales.

No obstante, también se han puesto de manifiesto algunas limitaciones a tener en cuenta: crecer el conjunto de datos, incrementar las prestaciones del modelo y movernos hacia una arquitectura más escalable.

Todos ellos son claros vectores de mejora a tener en cuenta en trabajos futuros. En conjunto, la discusión contextualiza los resultados en el estado del arte, pone de relieve las aportaciones del sistema y las mejoras posibles y da pie a futuros trabajos en la materia.

12. Conclusiones

12.1. Validación del modelo

El modelo de detección implementado en el sistema evidencia un desempeño funcional dentro del dominio de aplicación definido, demostrando su capacidad para identificar patrones visuales relevantes en imágenes adquiridas bajo condiciones no controladas. El uso de una arquitectura basada en aprendizaje profundo, en particular del tipo YOLO, permite realizar inferencias en tiempos adecuados, logrando un equilibrio razonable entre el rendimiento computacional y la capacidad del modelo para distinguir correctamente los objetos.

La validación del modelo se llevó a cabo en base a los datos reales, en el que pudimos observar cómo responde ante diferentes contextos, cambios de condiciones como la luz, cambios de fondo, cómo se capturan las imágenes, etc. A pesar de estas variaciones, el modelo tiene un comportamiento estable, lo que sugiere que tiene capacidad de generalización para su uso en entornos agrícolas.

Por otra parte, el proceso de entrenamiento supervisado, más el ajuste de parámetros, mediante la recopilación de los datos, nos ha permitido adaptar el modelo al campo de trabajo específico. Esto se refleja en una mayor consistencia en los resultados obtenidos. En ese sentido, el modelo no solo cumple una función operativa dentro del sistema, sino que también puede seguir mejorando con nuevas iteraciones de entrenamiento, conforme se disponga de más datos o se ajusten sus condiciones de uso.

12.2. Validación del sistema

El sistema desarrollado ha sido validado como una solución tecnológica integral que articula de manera coherente múltiples componentes dentro de un flujo de procesamiento estructurado. La interacción existente entre los módulos de adquisición de la información, preprocesamiento, inferencia o razonamiento, y persistencia da lugar a la ejecución ininterrumpida del propio proceso de análisis. En la fase funcional, el

sistema da muestra de ser capaz de procesar la información visual, de obtener resultados estructurados, y de guardar la información de forma consistente. Todo ello es evidencia de la correcta integración del backend, de los modelos de inteligencia artificial y de la capa de datos, así como del correcto mantenimiento de la integridad del flujo de información de los componentes del sistema en su totalidad. Además, la arquitectura modular ha facilitado la mantenibilidad y la evolución del sistema, al poder introducir mejoras o cambios en los componentes del sistema sin comprometer la integridad del resto de los componentes existentes. Este aspecto es muy importante en los entornos de desarrollo basados en inteligencia artificial, en donde es habitual llevar a cabo actualizaciones continuas de los modelos.

En consecuencia, la validación del sistema confirma su viabilidad como plataforma para el análisis automatizado de información visual en el contexto agrícola.

12.3. Cumplimiento de objetivos

El análisis del desarrollo del sistema permite establecer que los objetivos planteados en la investigación han sido alcanzados de manera satisfactoria. En primer lugar, se logró diseñar e implementar un modelo de detección basado en aprendizaje profundo, capaz de operar sobre datos reales y generar resultados coherentes con el dominio de aplicación.

En segundo lugar, se desarrolló una solución integral que integra múltiples componentes tecnológicos dentro de un flujo estructurado, permitiendo la transformación de datos visuales en información procesable. Esta integración cumple con el objetivo de automatizar el análisis, reduciendo la dependencia de evaluaciones manuales.

Finalmente, el sistema contribuye a la reducción de la subjetividad en los procesos de evaluación, al establecer criterios consistentes basados en datos. De esta manera, se evidencia no solo el cumplimiento de los objetivos técnicos, sino también la generación de valor en términos de aplicabilidad y utilidad práctica.

13. Trabajo Futuro

13.1. Nuevos datasets

Una línea de desarrollo futura relevante corresponde a la expansión y diversificación de los conjuntos de datos utilizados en el entrenamiento del modelo. La incorporación de nuevos datos permitirá mejorar la capacidad de generalización del sistema, especialmente frente a la variabilidad inherente a los entornos agrícolas.

Incorporar imágenes de distintos contextos, tanto geográficos como ambientales y productivos, puede ayudar a representar mejor el dominio en el que trabaja el modelo. De esta forma, el sistema puede responder de una manera más estable en diferentes situaciones posibles. Del mismo modo, incrementar la cantidad de datos etiquetados y cuidar su calidad incide de manera positiva en el proceso de entrenamiento.

Con un dataset más amplio también es posible eliminar ciertos sesgos y conseguir predicciones más exactas. En este sentido, aumentar el dataset no es solo una mejora adicional, sino que se convierte en un paso necesario para que el sistema sea capaz de enfrentarse a situaciones más complejas y asimile un comportamiento más robusto.

13.2. Mejora del modelo

Otra línea de trabajo futuro se orienta a la optimización del modelo de detección, tanto a nivel estructural como paramétrico. La exploración de nuevas arquitecturas, así como el ajuste de hiperparámetros, permitirá mejorar el desempeño del sistema en términos de precisión, velocidad de inferencia y eficiencia computacional.

Asimismo, la incorporación de técnicas como el aprendizaje transferido, la generación sintética de datos y algunos ajustes en el pipeline de inferencia puede ayudar a mejorar la capacidad del modelo para adaptarse a condiciones variables. Tales estrategias son especialmente adecuadas para trabajar en entornos donde los datos tengan suficiente variabilidad o bien sean insuficientes.

Por otro lado, la mejora del modelo no debe ser interpretada como un proceso inmediato, sino como una actividad continuada que ayuda a mejorar los modelos de forma continua, debido a que en los sistemas de inteligencia artificial sigue practicándose iterar, dado que las metodologías, así como las herramientas, están en un continuo cambio en los entornos en que son realizadas.

13.3. Escalabilidad

El sistema presenta potencial de evolución hacia arquitecturas más escalables que permitan su implementación en entornos de mayor complejidad y demanda. En este sentido, se considera la posibilidad de migrar hacia infraestructuras distribuidas o basadas en la nube, lo que permitiría mejorar la capacidad de procesamiento y el acceso remoto.

Asimismo, la integración con otros sistemas tecnológicos, como sensores o plataformas de monitoreo, permitiría ampliar el alcance del sistema hacia aplicaciones más avanzadas dentro del ámbito de la agricultura de precisión.

Desde una perspectiva arquitectónica, la escalabilidad implica no solo el incremento en la capacidad de procesamiento, sino también la adaptación del sistema para soportar múltiples usuarios, grandes volúmenes de datos y procesamiento concurrente.

La investigación contribuye al desarrollo de sistemas inteligentes aplicados al agro, proponiendo una solución reproducible y adaptable a distintos contextos productivos.

14. Referencias

Benos, L., Tagarakis, A. C., Dolias, G., Berruto, R., Kateris, D., & Bochtis, D. (2021). Machine learning in agriculture: A comprehensive updated review. Sensors, 21(11), 3758. https://doi.org/10.3390/s21113758

Botero-Valencia, J., et al. (2025). Machine learning in sustainable agriculture: A systematic review. Agriculture, 15(4), 377. https://doi.org/10.3390/agriculture15040377

Buitrago Bolívar, E., et al. (2024). Monitoreo de cultivos y suelos en agricultura de precisión con UAV e inteligencia artificial. Tecnura. http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0123-921X2024000400075

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) challenge. *International Journal of Computer Vision, 88*(2), 303–338. https://doi.org/10.1007/s11263-009-0275-4

Ferentinos, K. P. (2018). Deep learning models for plant disease detection and diagnosis. Computers and Electronics in Agriculture, 145, 311–318. https://doi.org/10.1016/j.compag.2018.01.009

Ghazal, S., Munir, A., & Qureshi, W. (2024). Computer vision in smart agriculture and precision farming: Techniques and applications. Artificial Intelligence in Agriculture, 13, 64–83. https://doi.org/10.1016/j.aiia.2024.06.004

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. https://www.deeplearningbook.org

Hassan, E., et al. (2025). Sustainable deep vision systems for fruit quality assessment. Frontiers in Plant Science. https://doi.org/10.3389/fpls.2025.1521508

Kamat, P., et al. (2025). Multi-class fruit ripeness detection using YOLO and SSD models. SN Applied Sciences. https://doi.org/10.1007/s42452-025-07617-7

Kamilaris, A., & Prenafeta-Boldú, F. X. (2021). Deep learning in agriculture: A survey. Computers and Electronics in Agriculture, 147, 70–90. https://doi.org/10.1016/j.compag.2018.02.016

Karacaoglu, B., et al. (2025). Optimized YOLO architectures for efficient kiwi detection in complex orchard environments. Scientific Reports. https://doi.org/10.1038/s41598-025-32770-9

Koirala, A., Walsh, K. B., Wang, Z., & McCarthy, C. (2022). Deep learning—Method overview and review of use for fruit detection and yield estimation. Computers and Electronics in Agriculture, 162, 219–234. https://doi.org/10.1016/j.compag.2019.04.017

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444. https://doi.org/10.1038/nature14539

León León, R. A., Barrantes Vargas, C. A., & Bacilio de la Cruz, F. D. (2024). Desarrollo de un sistema de visión artificial con la red neuronal convolucional (YOLO v8) para clasificar el arándano por su grado de madurez. https://www.iiis.org/CDs2024/CD2024Summer/papers/CA682TT.pdf

Liakos, K. G., et al. (2022). Machine learning in agriculture: A review. Sensors, 18(8), 2674. https://doi.org/10.3390/s18082674

Padilla, R., Passos, W. L., Dias, T. L. B., Netto, S. L., & da Silva, E. A. B. (2020). A comparative analysis of object detection metrics. *Electronics, 10*(3), 279. https://doi.org/10.3390/electronics10030279

Redmon, J., et al. (2016). You Only Look Once: Unified, real-time object detection. CVPR. https://doi.org/10.1109/CVPR.2016.91

Sasmoko, D., et al. (2025). CNN and YOLO algorithms for detecting plant diseases in precision agriculture: A review. https://www.researchgate.net/publication/390688664

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data, 6*, 60. https://doi.org/10.1186/s40537-019-0197-0

Sornalakshmi, K., et al. (2024). Fruit detection and counting for yield analysis in digital agriculture. SCITEPRESS. https://www.scitepress.org/Papers/2024/128812/128812.pdf

Talaviya, T., Shah, D., Patel, N., Yagnik, H., & Shah, M. (2020). Implementation of artificial intelligence in agriculture for optimisation of irrigation and application of pesticides and herbicides. Artificial Intelligence in Agriculture, 4, 58–73. https://doi.org/10.1016/j.aiia.2020.04.002

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., & Liu, C. (2018). A survey on deep transfer learning. *International Conference on Artificial Neural Networks*. https://doi.org/10.1007/978-3-030-01424-7_27

Villalobos-Culqui, C., et al. (2025). Artificial vision model based on convolutional neural networks for crop classification. https://revistas.unsm.edu.pe/index.php/rcsi/article/download/678/1413

Weng, W., et al. (2025). A deep learning network for accurate tomato fruit detection using improved YOLO. Smart Agricultural Technology. https://www.sciencedirect.com/science/article/pii/S2772375525006963

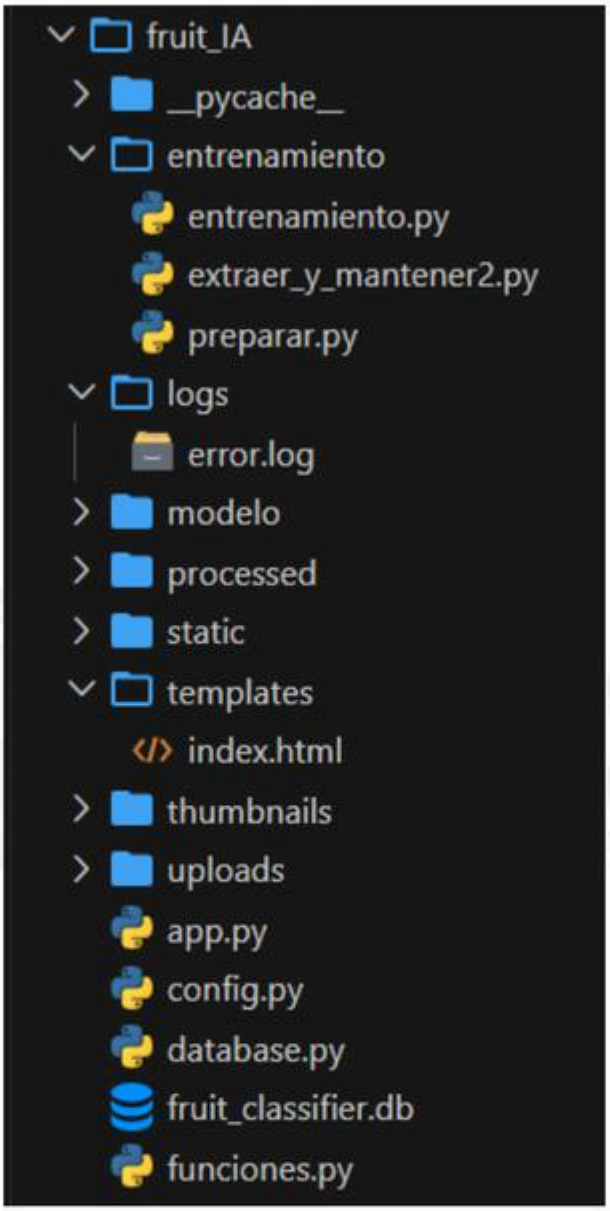
Wolfert, S., et al. (2021). Big data in smart farming: A review. Agricultural Systems, 153, 69–80. https://doi.org/10.1016/j.agsy.2017.01.023

Xiao, F., et al. (2023). Fruit detection and recognition based on deep learning: A review. Agronomy, 13(6), 1625. https://doi.org/10.3390/agronomy13061625

Zhang, W., et al. (2022). Deep-learning-based in-field citrus fruit detection and tracking. Horticulture Research. https://doi.org/10.1093/hr/uhac003

15. Anexo

Estructura



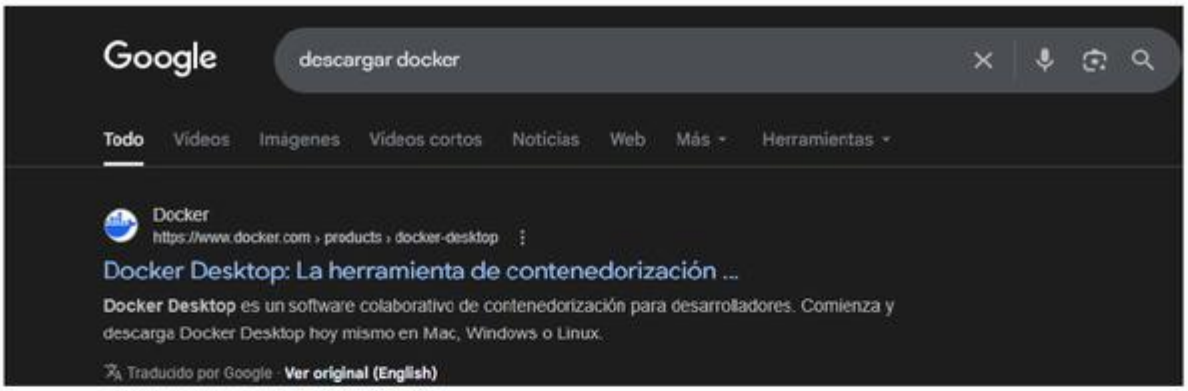
Instalación de Docker Desktop

Búsqueda de Docker en el navegador

En primer lugar, se abre cualquier navegador web, como Google Chrome, Microsoft Edge o Mozilla Firefox, y se ingresa en la barra de búsqueda el término:

Docker Desktop

Esta acción permitirá acceder al sitio oficial de Docker, desde donde se podrá descargar la versión correspondiente al sistema operativo utilizado.



Acceder a la página oficial de Docker

Una vez realizada la búsqueda, se debe ingresar a la página oficial de Docker Desktop, desde donde se obtiene la versión más reciente y segura del software:

https://www.docker.com/products/docker-desktop/

Es fundamental descargar Docker únicamente desde su sitio web oficial, ya que esto garantiza la integridad del instalador, evita el uso de versiones modificadas y reduce riesgos de seguridad en el sistema.



Descargar Docker Desktop

En la página oficial de Docker Desktop, se debe seleccionar el instalador correspondiente al sistema operativo del equipo:

- ✓ Windows
- ✓ macOS
- ✓ Linux

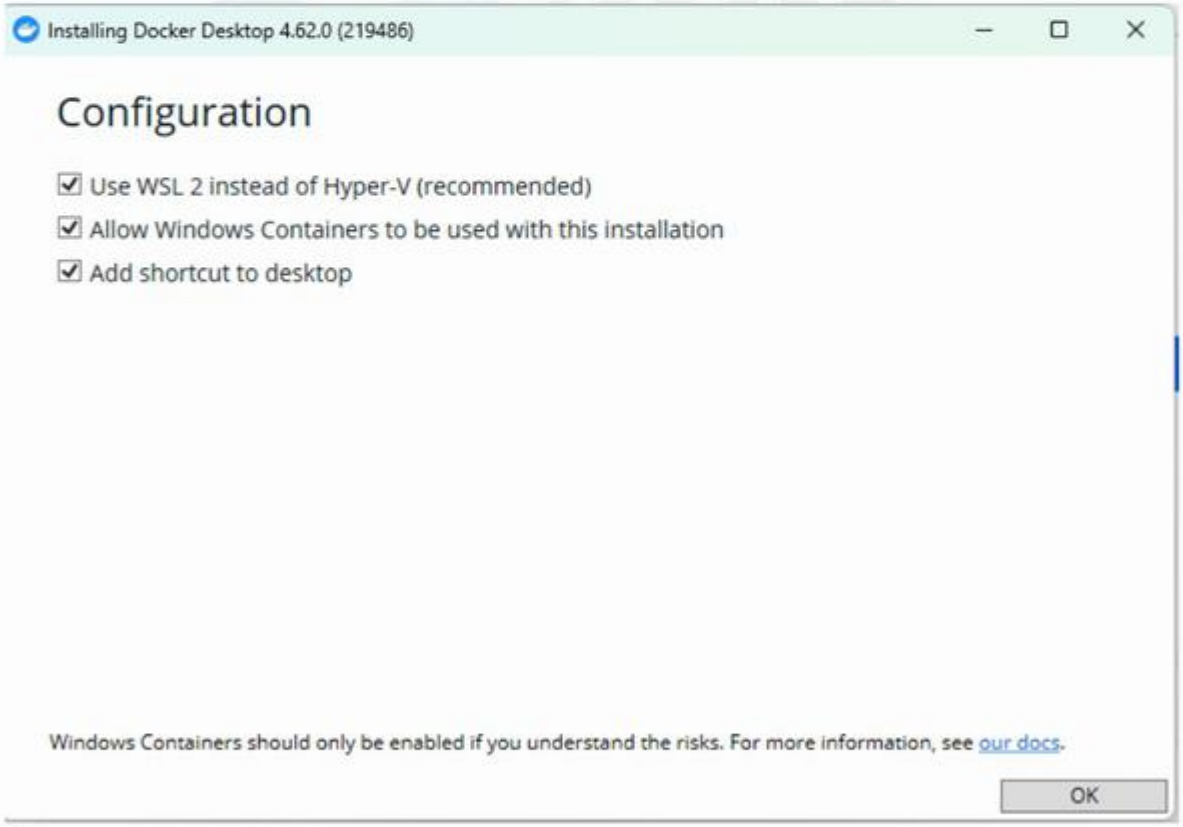
Para sistemas Windows, la descarga corresponde a un archivo con extensión .exe, el cual será utilizado posteriormente para iniciar el proceso de instalación.

Ejecutar el instalador

Una vez finalizada la descarga, se procede a ejecutar el instalador siguiendo los pasos indicados:

- ✓ Hacer doble clic sobre el archivo descargado.
- ✓ Avanzar en el asistente de instalación seleccionando Next / Siguiente en cada ventana.
- ✓ Aceptar los términos y condiciones de uso del software.
- ✓ Completar el proceso seleccionando la opción Finalizar.

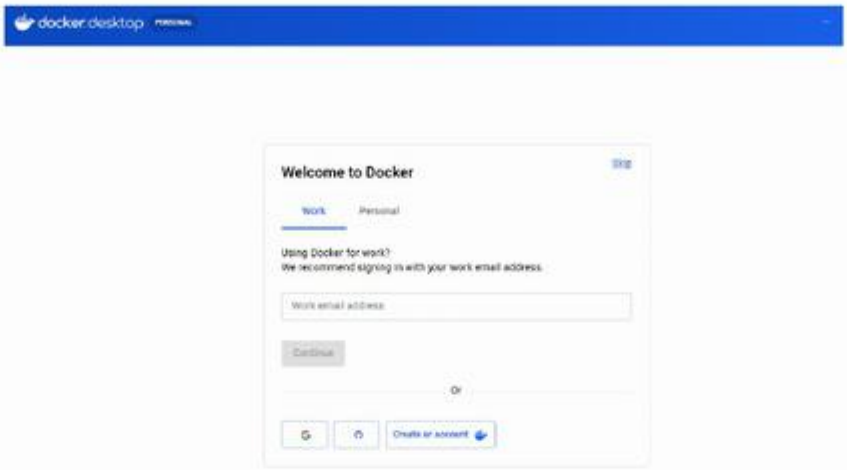
No es obligatorio crear una cuenta en **Docker Hub** para el desarrollo de este proyecto, por lo que este paso puede omitirse sin afectar el funcionamiento de Docker Desktop.



Verificar el funcionamiento de Docker

Una vez concluida la instalación, se procede a abrir Docker Desktop desde el menú de aplicaciones del sistema.

Si el ícono de Docker se muestra activo y sin mensajes de error, se confirma que Docker se encuentra correctamente instalado y listo para su uso.



Instalación de Label Studio usando Docker

Abrir la terminal del sistema

Para iniciar la instalación de Label Studio mediante Docker, es necesario abrir la terminal del sistema. En el caso de Windows, se realiza el siguiente procedimiento:

- ✓ Presionar las teclas Windows + R.
- ✓ Escribir cmd o powershell en la ventana que aparece.
- ✓ Presionar Enter para abrir la consola.

Desde esta terminal se ejecutarán los comandos necesarios para descargar y ejecutar Label Studio utilizando Docker.

```
Downloading cryptography-42.0.2-cp39-abi3-win_amd64.whl (2.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.9/2.9 MB 2.6 MB/s eta 0:00:00
Downloading google_api_core-2.16.2-py3-none-any.whl (135 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 135.2/135.2 kB 4.0 MB/s eta 0:00:00
Downloading google_auth-2.27.0-py2.py3-none-any.whl (186 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 186.8/186.8 kB 5.7 MB/s eta 0:00:00
Downloading google_cloud_core-2.4.1-py2.py3-none-any.whl (29 kB)
Downloading google_resumable_media-2.7.0-py2.py3-none-any.whl (80 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 80.6/80.6 kB 4.4 MB/s eta 0:00:00
Downloading ijson-3.2.3-cp311-cp311-win_amd64.whl (48 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 48.2/48.2 kB ? eta 0:00:00
Downloading label_studio_tools-0.0.3-py3-none-any.whl (14 kB)
Downloading pillow-10.2.0-cp311-cp311-win_amd64.whl (2.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.6/2.6 MB 2.4 MB/s eta 0:00:00
Downloading pyrsistent-0.20.0-cp311-cp311-win_amd64.whl (63 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 63.3/63.3 kB 3.5 MB/s eta 0:00:00
Downloading s3transfer-0.10.0-py3-none-any.whl (82 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 82.1/82.1 kB 4.8 MB/s eta 0:00:00
Downloading typing_extensions-4.9.0-py3-none-any.whl (32 kB)
Downloading cffi-1.16.0-cp311-cp311-win_amd64.whl (181 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 181.5/181.5 kB 3.6 MB/s eta 0:00:00
Downloading googleapis_common_protos-1.62.0-py2.py3-none-any.whl (228 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 228.7/228.7 kB 4.7 MB/s eta 0:00:00
```

Ejecutar el contenedor de Label Studio

Desde la terminal del sistema, se debe ejecutar el siguiente comando para iniciar Label Studio mediante Docker:

docker run -it -p 8080:8080 heartexlabs/label-studio:latest

Explicación del comando:

- ✓ docker run: crea y ejecuta un nuevo contenedor.
- ✓ -it: permite la interacción con el contenedor desde la terminal.
- ✓ -p 8080:8080: expone el servicio de Label Studio en el puerto 8080 del navegador.
- ✓ heartexlabs/label-studio: imagen oficial de Label Studio.
- ✓ latest: indica que se utilizará la versión más reciente disponible.

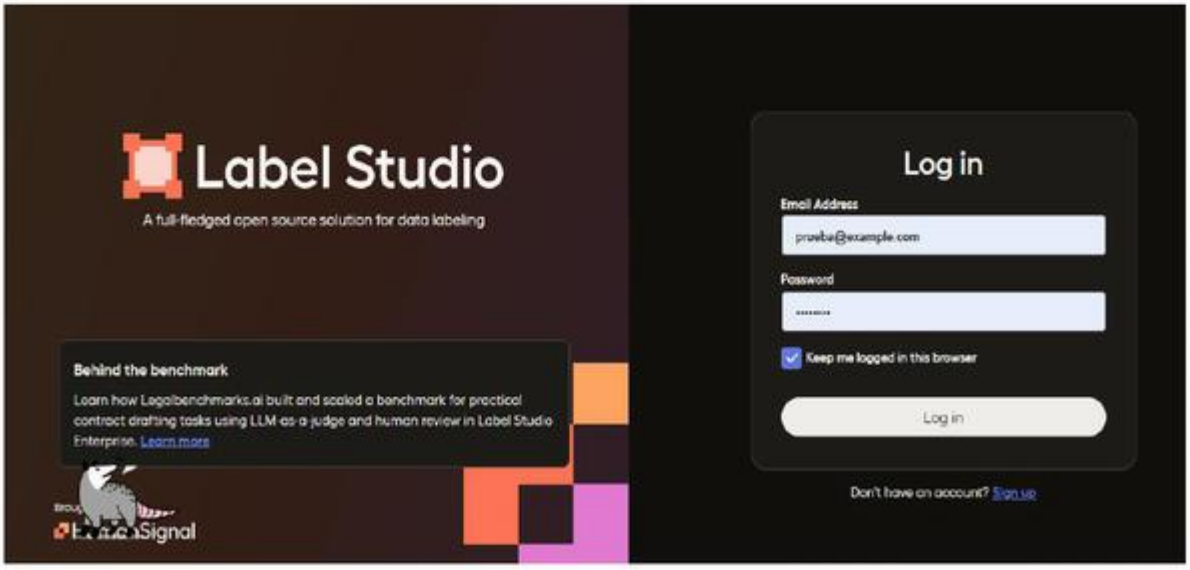
Al ejecutarse el comando, Docker descargará automáticamente la imagen (si no está disponible localmente) y levantará el servicio de Label Studio.

Acceder a Label Studio desde el navegador

Con el contenedor de Label Studio en ejecución, se debe abrir cualquier navegador web y escribir la siguiente dirección en la barra de direcciones:

http://localhost:8080

Al acceder a esta URL, se mostrará la **interfaz web de Label Studio**, desde donde es posible crear proyectos, cargar imágenes y comenzar el proceso de etiquetado.



Instalación de Python en el sistema

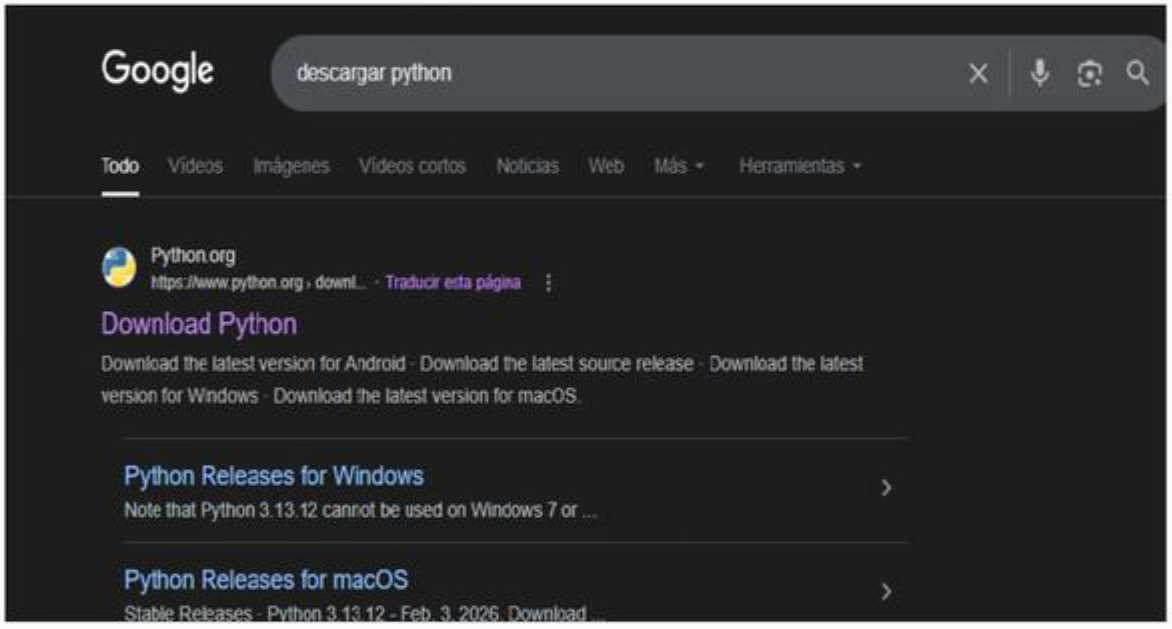
Cuando se opta por instalar **Label Studio sin utilizar Docker**, es necesario contar previamente con **Python instalado y correctamente configurado** en el sistema operativo.

Búsqueda de Python en el navegador

Para iniciar el proceso, se abre cualquier navegador web y se escribe en la barra de búsqueda el término correspondiente:

Python

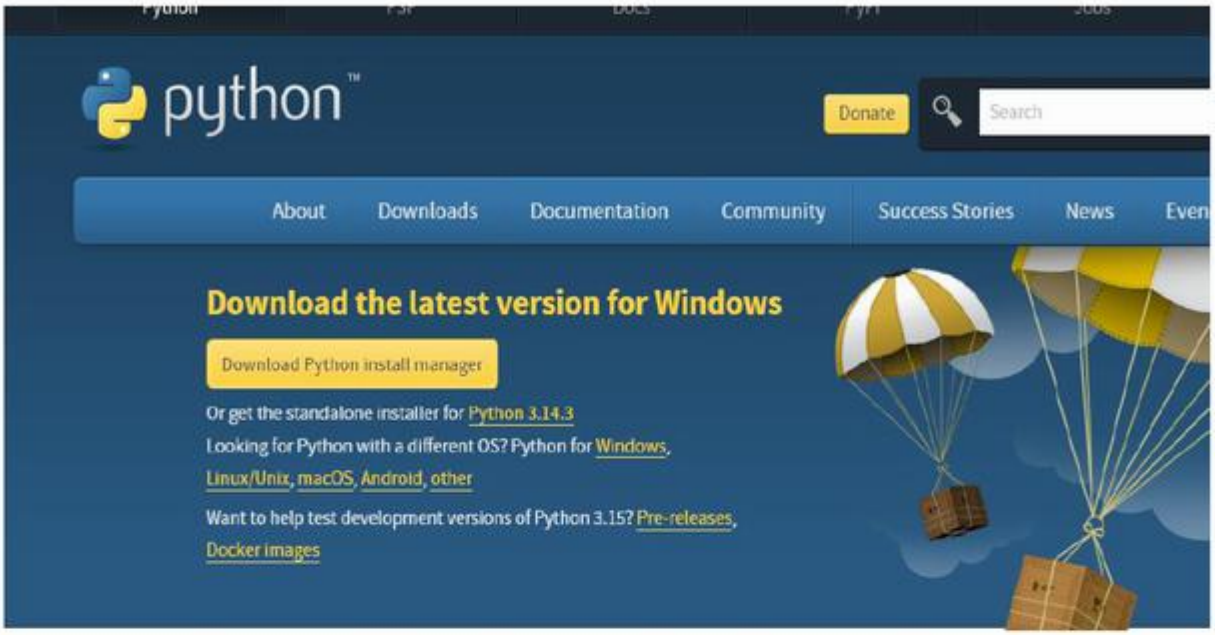
Esta acción permitirá acceder al sitio oficial de Python, desde donde se podrá descargar la versión adecuada según el sistema operativo.



Acceder a la página oficial de Python

En medio de los resultados que aparece dentro de la búsqueda, tendrá que seleccionar la opción Python.org, que es la que lleva al sitio oficial del lenguaje Python.

Es importante que la descarga se realice únicamente desde esta página oficial, puesto que es la única forma que garantiza que se va a obtener una versión segura, actualizada y sin modificaciones, evitando así riesgos de seguridad para el sistema; en caso contrario la versión puede no ser segura.



Descargar Python para Windows

Una vez que acceda a la página oficial de Python, debe realizar lo siguiente:

1. Hacer clic en Download Python.
2. El sitio web del servidor reconoce el sistema operativo de su equipo.
3. Comienza la descarga del instalador para Windows, el cual es un archivo con extensión .exe.

Este instalador nos permitirá realizar la instalación guiada de Python en el sistema. Ejecutar el instalador de Python Una vez que se ha terminado de descargar el instalador, hay que realizar la instalación siguiendo estos pasos:

- ✓ Hacer doble clic sobre el archivo descargado.
- ✓ Obligatoriamente deberá tildar la opción Add Python to PATH antes de empezar la instalación.
- ✓ Hacer clic sobre Install Now para que comience el proceso de instalación.

Este paso es muy importante, ya que permite ejecutar Python directamente desde la terminal del sistema, lo que es necesario para la instalación y ejecución de Label Studio sin Docker.



Verificación de la instalación de Python

Abrir la terminal de comandos

Para corroborar que la instalación de Python se haya efectuado de forma correcta, hay que proceder a abrir la terminal de comandos, pero en el caso de sistemas Windows en particular, se seguirá el siguiente procedimiento:

- ✓ Pulsar las teclas Windows + R.
- ✓ Escribir cmd en el diálogo de apertura que aparece.
- ✓ Pulsar Enter para abrir la consola de comandos.

Es a partir de dicha terminal de comandos donde se ejecutarán los comandos que se necesiten para verificar que Python funciona como se espera. Verificar la versión de Python Desde la terminal de comandos se tiene que ejecutar el siguiente comando: python --version

Si Python se encuentra instalado correctamente, la terminal indicará que Python está correctamente instalado al mostrar el texto a continuación con la versión instalada, es decir: Python 3.x.x. De este modo confirmamos que el lenguaje ya está disponible para poder usarlo desde la línea de comandos.



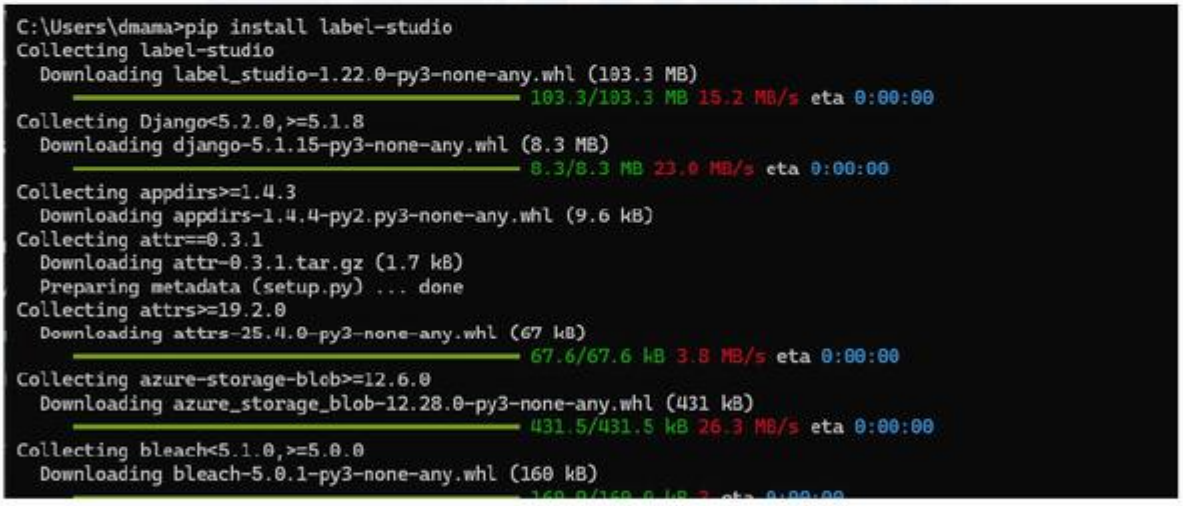
Instalación de Label Studio usando Python (pip)

Una vez Python haya sido instalado y comprobado que el entorno de desarrollo funciona correctamente, se puede proceder con la instalación de Label Studio mediante el gestor de paquetes pip.

Instalando Label Studio

Desde la misma terminal de comandos se debe ejecutar el siguiente comando: pip install label-studio.

El anterior comando servirá para descargar e instalar automáticamente Label Studio, así como todos los paquetes y dependencias a la instalación del paquete de Label Studio, para que se pueda ejecutar correctamente en el sistema.

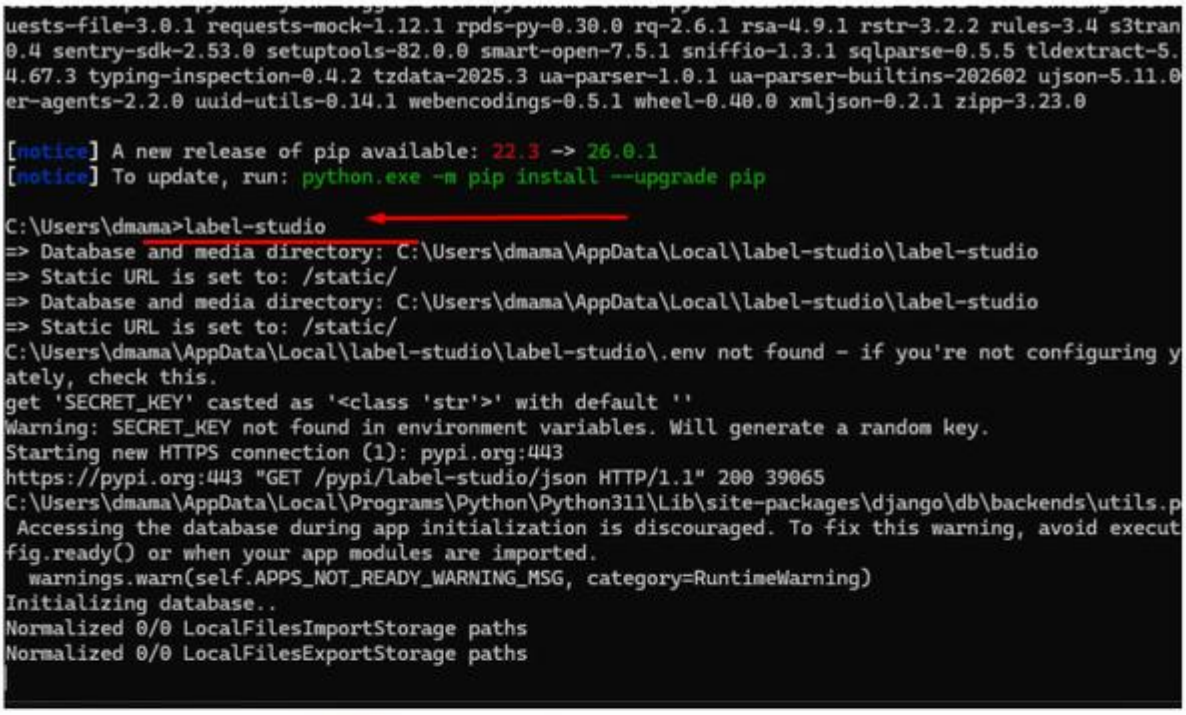


Iniciar Label Studio

Una vez completada la instalación de Label Studio, se debe ejecutar el siguiente comando desde la terminal:

label-studio

Al ejecutar este comando, el sistema iniciará el servidor local de Label Studio, permitiendo acceder a la aplicación desde el navegador web para comenzar con la creación y gestión de proyectos de etiquetado.



Entrada a Label Studio desde el navegador.

Si el servidor de Label Studio se está ejecutando, se puede abrir cualquier navegador web y navegar a la siguiente dirección: http://localhost:8080

Al cargar esta URL, se debería tener la interfaz gráfica de Label Studio donde se pueden crear proyectos, importar imágenes y llevar a cabo la tarea de etiquetado de datos.

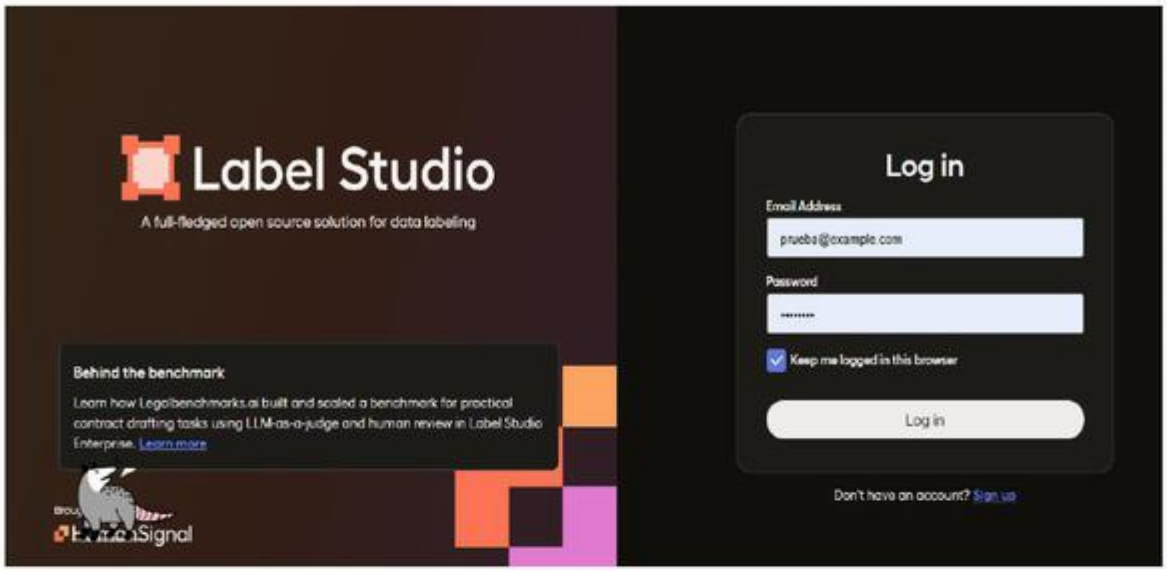
Configuración inicial de Label Studio

Crear usuario administrador

En el momento de hacer el primer acceso del sistema Label Studio, este nos pedirá que se cree un usuario administrador. Concretamente, será necesario proporcionar los siguientes datos:

- ✓ Usuario.
- ✓ Correo electrónico.
- ✓ Contraseña segura.

Este usuario contará con el rol de administrador, lo que implica que será el responsable de crear, gestionar y configurar los proyectos, así como de dar seguimiento a las anotaciones que se han ido realizando.

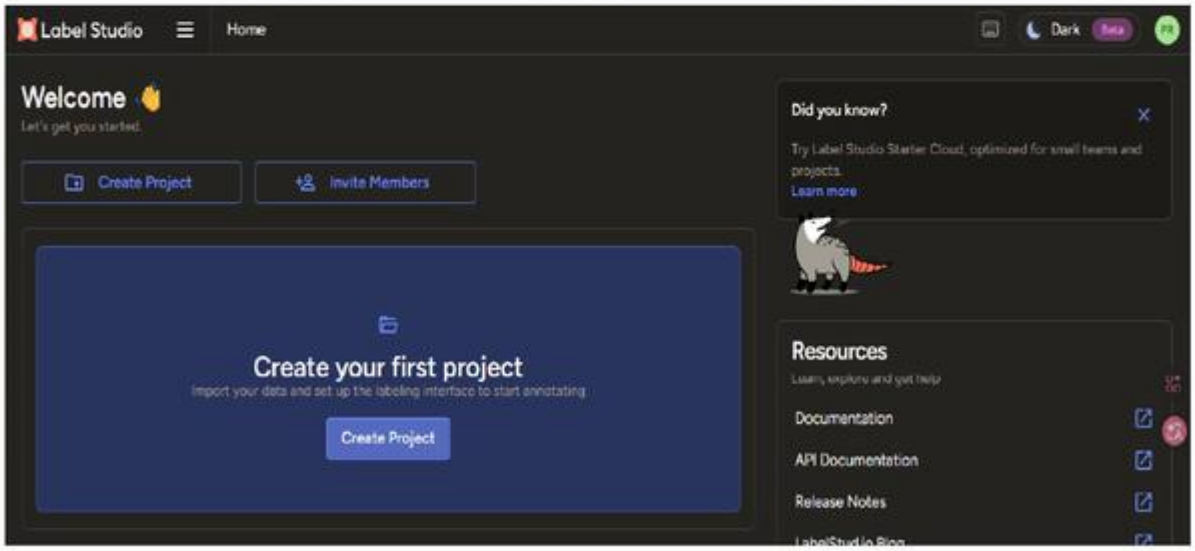


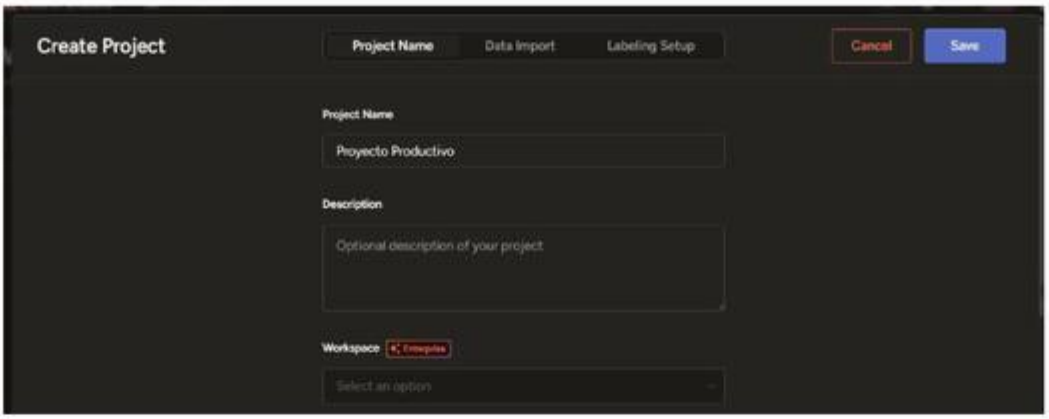
Creando un nuevo proyecto

Después de haber creado el usuario administrador y haber iniciado sesión, se procede a la creación de un nuevo proyecto dentro de Label Studio siguiendo estos pasos:

1. Hacer clic en la opción Create Project en la página principal.
2. Asignar un nombre al proyecto siendo éste el que se eligió para el objetivo del etiquetado.
3. Guardar este primer estado del proyecto para seguir el proceso de anotación.

Este nuevo proyecto servirá de contenedor para las imágenes, etiquetas y opciones del dataset a construir.



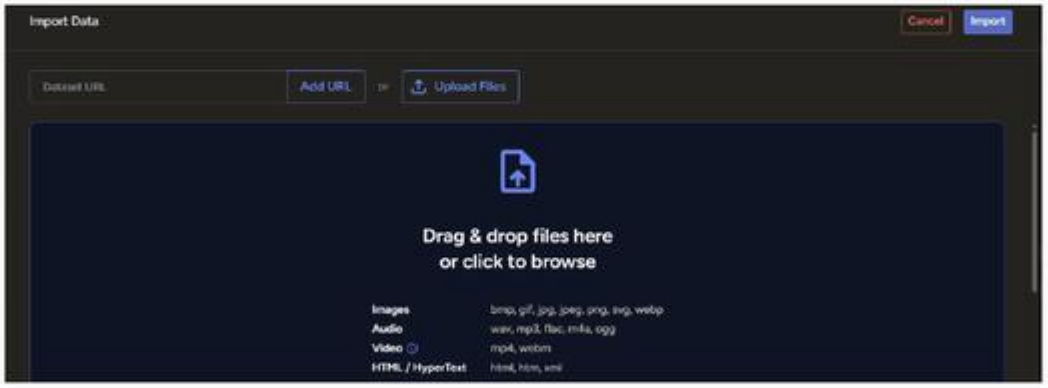


Importar imágenes para el dataset

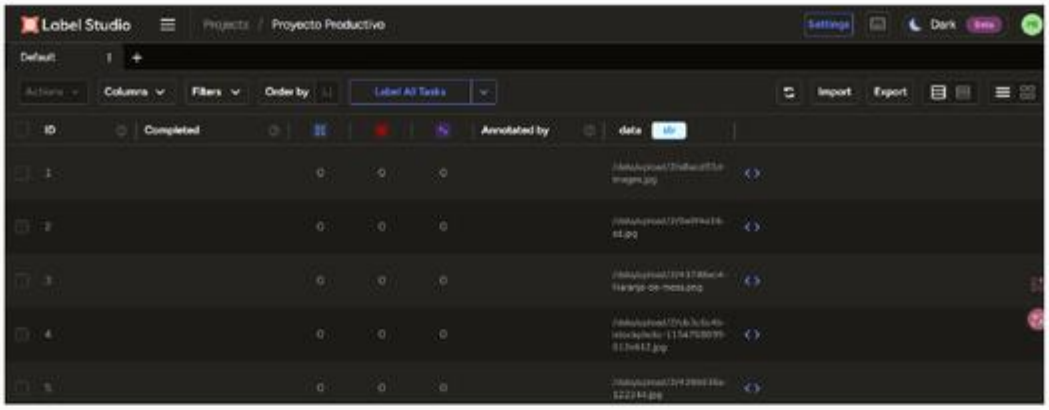
Una vez que se tiene el proyecto que se ha creado, se suben las imágenes que forman parte de la base de datos. Los pasos a seguir para llevarlo a cabo son los siguientes:

1. Entrar en el proyecto que se ha creado en Label Studio.
2. Seleccionado la opción Import desde la ventana del proyecto.
3. Cargar las imágenes que se han de utilizar para hacer el etiquetado desde el disco duro local.

Las imágenes, una vez importadas, quedan disponibles para ir realizando la anotación de las mismas necesarias para construir el dataset de entrenamiento.







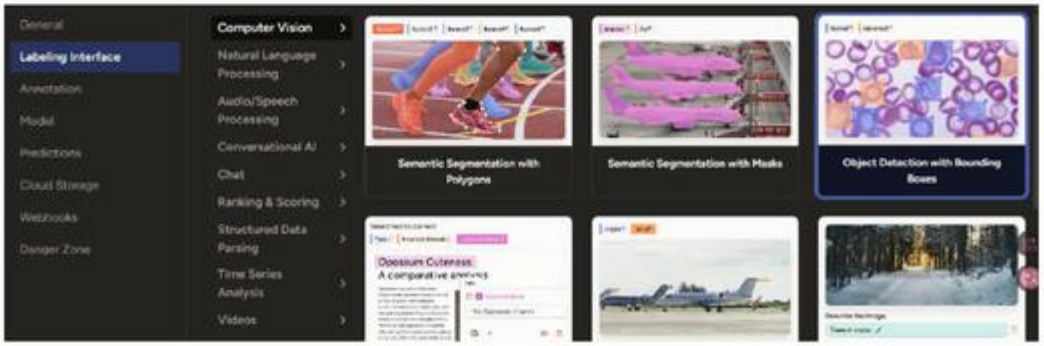
Iniciar el proceso de etiquetado

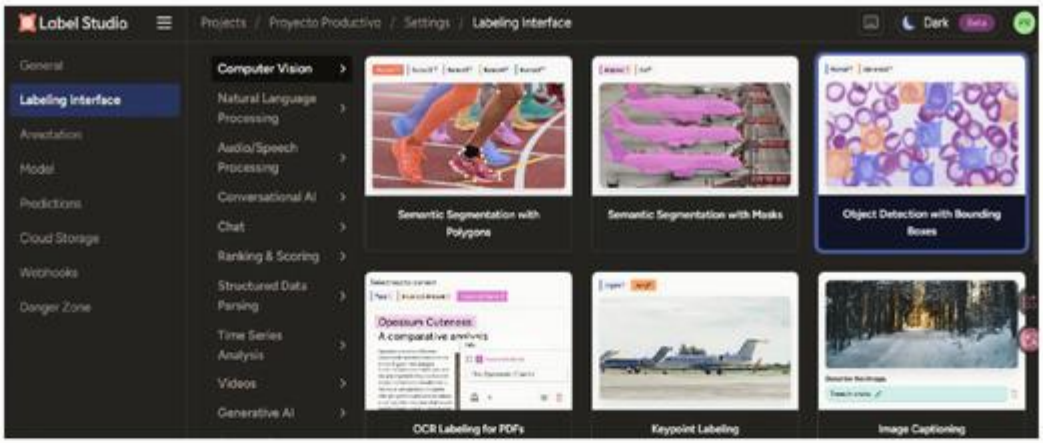
De acuerdo con el tipo de proyecto y el objetivo del análisis, se debe seleccionar el **método de anotación** correspondiente, entre las siguientes opciones:

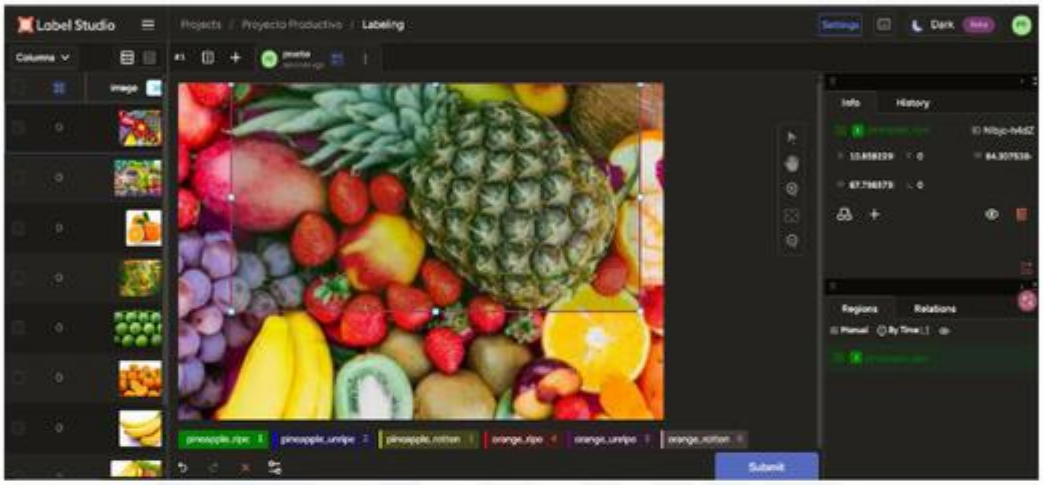
- ✓ Bounding Boxes
- ✓ Clasificación
- ✓ Segmentación

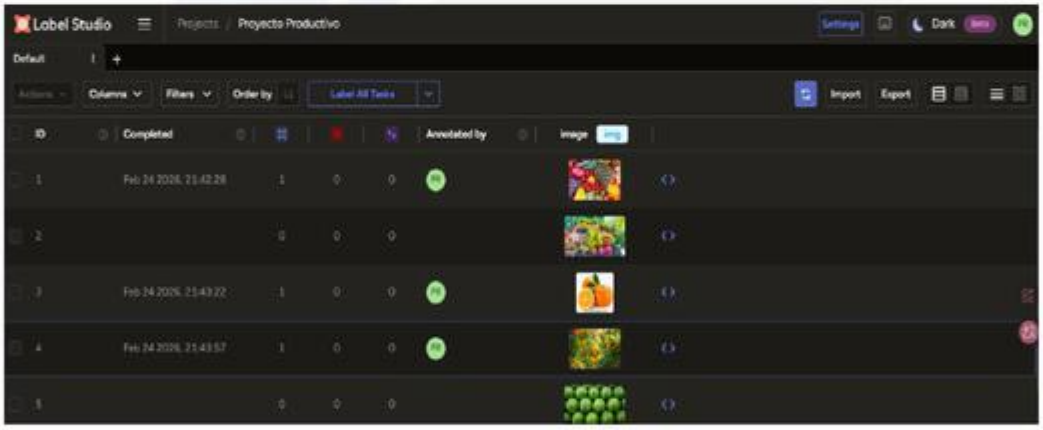
En proyectos de visión por computadora, el método más utilizado es Bounding Boxes, ya que permite delimitar y etiquetar objetos específicos dentro de las imágenes, generando información precisa para el entrenamiento de modelos de detección de objetos.











Seleccionar el formato YOLO

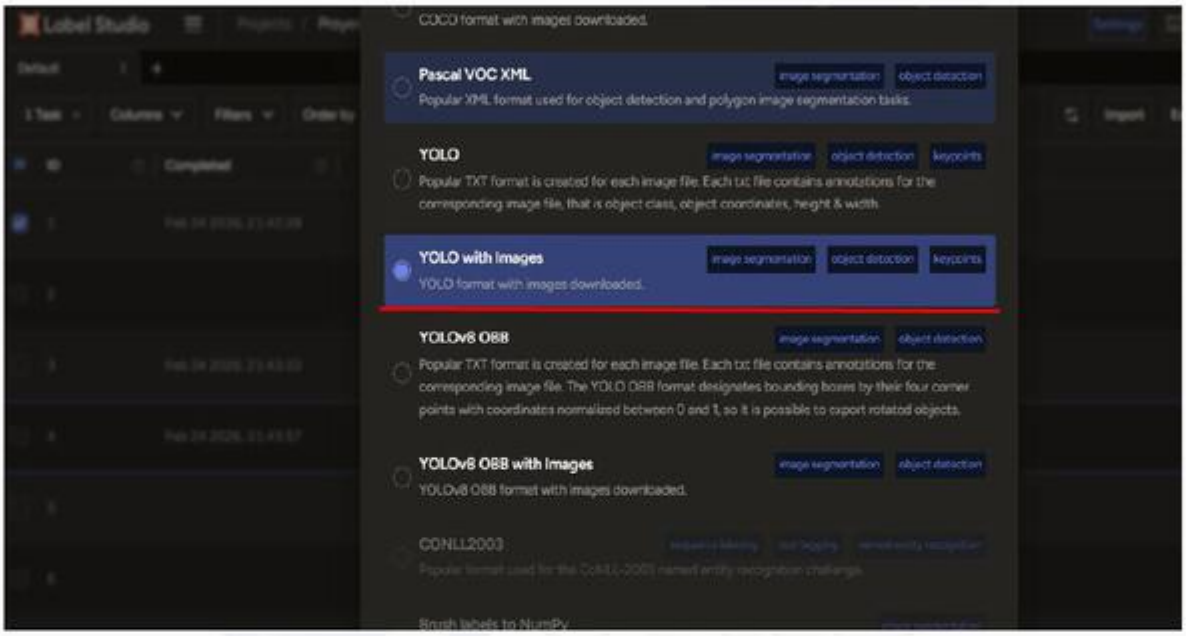
Para proyectos orientados a detección de objetos, se debe seleccionar el formato de exportación YOLO with images.

Este formato ofrece las siguientes ventajas:

✓ Permite exportar las imágenes originales junto con sus anotaciones.

✓ Genera automáticamente los archivos de etiquetas en formato .txt, requeridos por YOLO.

Organiza el dataset de manera compatible con YOLOv5, YOLOv8 y versiones similares, facilitando su uso directo en procesos de entrenamiento y validación de modelos.



Descargar el dataset exportado

Una vez que se encuentre seleccionado el formato de exportación que corresponda, a partir de este momento se deberá de proceder de la siguiente manera considerando todas las opciones:

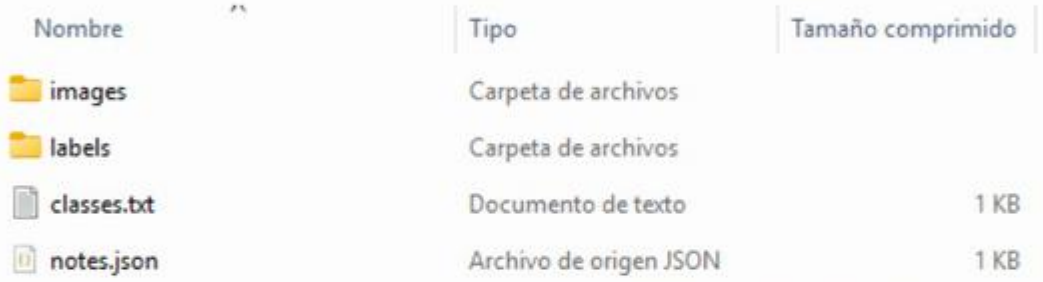
1. Se deberá de hacer click en la opción de export.
2. Se descarga el archivo zip generado por el sistema.
3. Se descomprime todo el contenido en alguna carpeta del sistema que corresponda donde quede el dataset final correspondiente.

Al finalizar este proceso, el dataset será recuperado y quedará organizado para su uso en el entrenamiento de modelos de detección de objetos. Estructura del dataset exportado Al descomprimir el archivo exportado desde Label Studio se obtiene la estructura de carpetas y archivos organizada de la siguiente forma:

✓ images/: contiene las imágenes etiquetadas utilizadas en el proyecto en concreto.

- ✓ labels/: contiene los archivos.txt con las anotaciones correspondientes de cada imagen en formato yolo.
- ✓ classes.txt: archivo que almacena la lista de clases definidas a partir del etiquetado correspondiente.
- ✓ notes.json: corresponde a un archivo que incluye metadatos del proyecto, como configuraciones y otras notas adicionales correspondientes a la explicación de la exportación.

Esta estructura es compatible al 100% con modelos YOLO, ofreciéndonos la posibilidad de ejecutarlo de forma directa en los procesos de entrenamiento y evaluación.



El procedimiento de etiquetado incluyó:

- ✓ Configuración del Proyecto
  - ➢ Definir el objetivo del proyecto (por ejemplo, reconocimiento de frutas).
  - ➢ Establecer el nombre del proyecto y la descripción.
- ✓ Importación de Imágenes
  - ➢ Seleccionar y cargar las imágenes relevantes al proyecto.
  - ➢ Verificar la calidad y resolución de las imágenes.
  - ➢ Organizar las imágenes en carpetas (opcional) para facilitar el acceso.
- ✓ Definición de Clases
  - ➢ Identificar y definir las clases necesarias para la anotación:
    - Maduro
    - Inmaduro

- Malogrado

➢ Crear un esquema de clases que incluya descripciones y ejemplos (opcional).

✓ Delimitación de Objetos

➢ Utilizar herramientas de anotación para trazar bounding boxes alrededor de los objetos de interés en cada imagen.

➢ Asegurar de que las bounding boxes estén ajustadas correctamente alrededor de los objetos.

➢ Revisar y ajustar las anotaciones para mayor precisión.

✓ Validación de Anotaciones

➢ Realizar una revisión de calidad de las anotaciones realizadas.

➢ Corregir posibles errores y inconsistencias en las delimitaciones y etiquetas.

✓ Exportación de Datos

➢ Exportar el conjunto de datos anotado en formato YOLO.

➢ Comprobar la estructura de archivos generada para asegurarse de que sea compatible con el modelo de entrenamiento.

✓ Documentación del Proyecto

➢ Crear documentación que incluya el proceso de anotación, el esquema de clases y cualquier otro detalle relevante.

➢ Incluir referencias a los archivos de imagen y etiquetas generados.

✓ Entrenamiento y Evaluación (opcional)

➢ Preparar el conjunto de datos para el entrenamiento de un modelo (división en conjuntos de entrenamiento, validación y prueba).

➢ Evaluar el rendimiento del modelo y ajustar las anotaciones en consecuencia si es necesario.

El dataset final generado presenta la siguiente estructura:

- ✓ images/: Carpeta que contiene las imágenes del dataset.
- ✓ labels/: Carpeta que incluye archivos .txt con las anotaciones correspondientes a cada imagen.
- ✓ classes.txt: Archivo que define las clases utilizadas en el dataset.

Este formato es totalmente compatible con los modelos YOLO, simplificando significativamente su integración en el proceso de entrenamiento.

Código:

app.py

```
from flask import Flask, render_template, request, jsonify,
send_from_directory
import cv2
import numpy as np
import os
import base64
from werkzeug.utils import secure_filename

from config import (
    logger,
    BASE_DIR, TEMPLATES_DIR, STATIC_DIR,
    UPLOAD_FOLDER, PROCESSED_FOLDER, THUMBNAIL_FOLDER,
    LOG_DIR, MODEL_PATH,
    ALLOWED_EXTENSIONS, MAX_UPLOAD_MB,
    REGIONES_PERU, MODEL_IMGSZ, MODEL_CONF,
    GROQ_MODEL,
)
from database import (
    init_db, save_to_db, get_db,
    get_biblioteca, get_urls_by_id, delete_by_id, delete_all,
    get_estadisticas, update_notas,
)
from funciones import (
    procesar_imagen,
    model, MODEL_USADO,
)

from datetime import datetime

app = Flask(
    __name__,
    template_folder=TEMPLATES_DIR,
```

```
    static_folder=STATIC_DIR
)
app.config['MAX_CONTENT_LENGTH'] = MAX_UPLOAD_MB * 1024 * 1024

init_db()


def allowed_file(filename: str) -> bool:
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS


def _borrar_archivos(url: str, thumbnail_url: str) -> None:
    for url_field, folder in [(url, PROCESSED_FOLDER), (thumbnail_url,
THUMBNAIL_FOLDER)]:
        if url_field:
            fname = url_field.split('/')[-1]
            fpath = os.path.join(folder, fname)
            try:
                if os.path.exists(fpath):
                    os.remove(fpath)
            except Exception as fe:
                logger.warning(f"No se pudo borrar {fpath}: {fe}")


@app.route('/')
def index():
    return render_template('index.html')

@app.route('/logo.png')
def serve_logo():
    p = os.path.join(BASE_DIR, 'logo.png')
    return send_from_directory(BASE_DIR, 'logo.png') if os.path.exists(p)
else ('', 404)

@app.route('/processed/<path:filename>')
def serve_processed(filename):
    return send_from_directory(PROCESSED_FOLDER, filename)

@app.route('/thumbnails/<path:filename>')
def serve_thumbnail(filename):
    return send_from_directory(THUMBNAIL_FOLDER, filename)


@app.route('/departamentos/<region>')
def get_departamentos(region):
    return jsonify(REGIONES_PERU.get(region, {}).get('departamentos', []))
```

```
@app.route('/modelo-info')
def modelo_info():
    clases = list(model.names.values()) if hasattr(model, 'names') else []
    return jsonify({
        "BASE_DIR":      BASE_DIR,
        "MODEL_PATH":    MODEL_PATH,
        "modelo_existe": os.path.exists(MODEL_PATH),
        "modelo_mb":     round(os.path.getsize(MODEL_PATH)/(1024*1024), 1)
                         if os.path.exists(MODEL_PATH) else 0,
        "modelo_usado":  MODEL_USADO,
        "total_clases":  len(clases),
        "clases":        clases,
        "imgsz":         MODEL_IMGSZ,
        "conf_umbral":   MODEL_CONF,
        "ia_modelo":     GROQ_MODEL,
        "ia_proveedor":  "Groq (servidor)",
        "version":       "v5.2",
    })


@app.route('/upload', methods=['POST'])
def upload():
    """YOLO + Groq en el servidor. Devuelve resultado completo."""
    try:
        if 'imagen' not in request.files:
            return jsonify({"error": "No se encontró el campo 'imagen'"}),
400
        file = request.files['imagen']
        if file.filename == '':
            return jsonify({"error": "No se seleccionó ningún archivo"}),
400
        if not allowed_file(file.filename):
            return jsonify({"error": f"Tipo no permitido: {',
'.join(ALLOWED_EXTENSIONS)}"}), 400

        img_bytes = file.read()
        npimg     = np.frombuffer(img_bytes, np.uint8)
        img       = cv2.imdecode(npimg, cv2.IMREAD_COLOR)
        if img is None:
            return jsonify({"error": "Imagen corrupta o formato no
reconocido"}), 400

        departamento = request.form.get('departamento', '')
        timestamp    = datetime.now().strftime("%Y%m%d_%H%M%S_%f")
        base_name    = os.path.splitext(
                           __import__('werkzeug.utils',
fromlist=['secure_filename'])
                           .secure_filename(file.filename)
                       )[0][:25]
```

```
        entry    = procesar_imagen(img, departamento, timestamp, base_name)
        entry_id = save_to_db(entry)
        entry["id"] = entry_id

        logger.info(f"upload OK id={entry_id} det={entry['detection_count']}
groq={entry['procesado_gpt']}")
        return jsonify(entry)

    except Exception as e:
        logger.error(f"Error en /upload: {e}", exc_info=True)
        return jsonify({"error": str(e)}), 500


@app.route('/captura', methods=['POST'])
def captura():
    """YOLO + Groq en el servidor desde imagen base64 (cámara)."""
    try:
        data = request.get_json(silent=True)
        if not data or 'imagen' not in data:
            return jsonify({"error": "No se encontró el campo 'imagen'"}),
400

        b64     = data['imagen']
        raw_b64 = b64.split(',')[1] if ',' in b64 else b64
        npimg   = np.frombuffer(base64.b64decode(raw_b64), np.uint8)
        img     = cv2.imdecode(npimg, cv2.IMREAD_COLOR)
        if img is None:
            return jsonify({"error": "Imagen base64 inválida"}), 400

        departamento = data.get('departamento', '')
        timestamp    = datetime.now().strftime("%Y%m%d_%H%M%S_%f")

        entry    = procesar_imagen(img, departamento, timestamp, 'capture')
        entry_id = save_to_db(entry)
        entry["id"] = entry_id

        logger.info(f"captura OK id={entry_id}
det={entry['detection_count']}")
        return jsonify(entry)

    except Exception as e:
        logger.error(f"Error en /captura: {e}", exc_info=True)
        return jsonify({"error": str(e)}), 500


@app.route('/biblioteca')
def ruta_biblioteca():
    try:
```

```
        page     = max(1, int(request.args.get('page', 1)))
        per_page = min(50, max(1, int(request.args.get('per_page', 20))))
        return jsonify(get_biblioteca(page, per_page))
    except Exception as e:
        logger.error(f"Error en /biblioteca: {e}", exc_info=True)
        return jsonify({"error": str(e)}), 500


@app.route('/delete/<int:db_id>', methods=['DELETE'])
def delete_entry(db_id):
    try:
        urls = get_urls_by_id(db_id)
        if not urls:
            return jsonify({"error": "Registro no encontrado"}), 404
        _borrar_archivos(urls[0], urls[1])
        delete_by_id(db_id)
        return jsonify({"message": "Registro eliminado"})
    except Exception as e:
        logger.error(f"Error en /delete/{db_id}: {e}", exc_info=True)
        return jsonify({"error": str(e)}), 500


@app.route('/notas/<int:db_id>', methods=['POST'])
def actualizar_notas(db_id):
    try:
        data  = request.get_json(silent=True) or {}
        notas = data.get('notas', '').strip()[:500]
        if update_notas(db_id, notas):
            return jsonify({"message": "Notas actualizadas"})
        return jsonify({"error": "Registro no encontrado"}), 404
    except Exception as e:
        return jsonify({"error": str(e)}), 500


@app.route('/reset', methods=['POST'])
def reset_database():
    try:
        for folder in [UPLOAD_FOLDER, PROCESSED_FOLDER, THUMBNAIL_FOLDER]:
            for fname in os.listdir(folder):
                fpath = os.path.join(folder, fname)
                try:
                    if os.path.isfile(fpath):
                        os.remove(fpath)
                except Exception:
                    pass
        n = delete_all()
        logger.warning(f"Sistema reseteado: {n} registros eliminados")
        return jsonify({"message": f"Sistema reseteado: {n} registros
eliminados"})
```

```
    except Exception as e:
        logger.error(f"Error en /reset: {e}", exc_info=True)
        return jsonify({"error": str(e)}), 500


@app.route('/estadisticas')
def ruta_estadisticas():
    try:
        return jsonify(get_estadisticas())
    except Exception as e:
        logger.error(f"Error en /estadisticas: {e}", exc_info=True)
        return jsonify({"error": str(e)}), 500


@app.route('/test-groq')
def test_groq():
    """Verifica que Groq API funcione desde el servidor."""
    from funciones import consultar_groq
    resp = consultar_groq('{"test":true} Responde solo:
{"status":"ok","modelo":"groq"}')
    if resp:
        return jsonify({"status": "ok", "modelo": GROQ_MODEL, "respuesta":
str(resp)[:100]})
    return jsonify({"status": "error", "mensaje": "Sin respuesta de Groq"}),
503


@app.route('/logs')
def ver_logs():
    log_file = os.path.join(LOG_DIR, 'error.log')
    if not os.path.exists(log_file):
        return jsonify({"logs": [], "mensaje": "Sin errores registrados"})
    try:
        with open(log_file, 'r', encoding='utf-8') as f:
            lines = f.readlines()
        return jsonify({"logs": [l.rstrip() for l in lines[-100:]]})
    except Exception as e:
        return jsonify({"error": str(e)}), 500


@app.errorhandler(404)
def error_404(e): return jsonify({"error": "Ruta no encontrada"}), 404

@app.errorhandler(413)
def error_413(e): return jsonify({"error": f"Imagen demasiado grande (máx.
{MAX_UPLOAD_MB} MB)"}), 413

@app.errorhandler(500)
def error_500(e):
```

```
    logger.error(f"Error 500: {e}", exc_info=True)
    return jsonify({"error": "Error interno. Revisa logs/error.log"}), 500

if __name__ == '__main__':
    print("=" * 65)
    print("  CLASIFICADOR DE FRUTAS PERUANAS v5.2 — 2026")
    print("=" * 65)
    print(f"  URL:          http://127.0.0.1:5000")
    print(f"  Test Groq:    http://127.0.0.1:5000/test-groq")
    print(f"  Diagnóstico:  http://127.0.0.1:5000/modelo-info")
    print(f"  Modelo YOLO:  {MODEL_USADO}")
    print(f"  Modelo IA:    {GROQ_MODEL} (Groq — gratuito)")
    print(f"  DB:           {os.path.join(BASE_DIR, 'fruit_classifier.db')}")
    print("=" * 65 + "\n")
    app.run(host='127.0.0.1', port=5000, debug=False, use_reloader=False)
```

templates/index.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8"/>
  <meta name="viewport" content="width=device-width,initial-scale=1.0"/>
  <title>NeuroAgro 2026</title>
  <link rel="icon" href="data:image/svg+xml,<svg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 100 100'><text y='.9em'
font-size='90'>🫐</text></svg>"/>
  <link rel="preconnect" href="https://fonts.googleapis.com"/>
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin/>
  <link
href="https://fonts.googleapis.com/css2?family=Space+Mono:wght@400;700&famil
y=Outfit:wght@300;400;500;600;700;800&display=swap" rel="stylesheet"/>
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.5.0/css/all.min.css"/>
  <script
src="https://cdn.jsdelivr.net/npm/chart.js@4.4.0/dist/chart.umd.min.js"></sc
ript>
<style>

:root{
  --bg:#070E0B; --bg2:#0C1710; --bg3:#101C14; --card:#121E18; --
card2:#182A20;
  --teal:#00D4AA; --teal2:#00FFD0; --tealA:rgba(0,212,170,.12);
  --lime:#BCFF00; --amber:#FFB300; --coral:#FF4D4D; --purple:#A78BFA; --
blue:#60A5FA;
  --tx:#E4F2EE; --tx2:#6FA89A; --tx3:#3D6358;
  --brd:rgba(0,212,170,.13); --brd2:rgba(0,212,170,.06);
  --mad:#22C55E; --ver:#F59E0B; --pod:#EF4444;
  --sb:258px; --ra:13px; --ra2:8px;
```

```
  --sh:0 2px 20px rgba(0,0,0,.55);
  --sl:0 8px 50px rgba(0,0,0,.7);
  --glow:0 0 40px rgba(0,212,170,.08);
  --tr:.18s cubic-bezier(.4,0,.2,1);
}
*,*::before,*::after{box-sizing:border-box;margin:0;padding:0}
html{scroll-behavior:smooth}
body{font-family:'Outfit',system-ui,sans-serif;background:var(--
bg);color:var(--tx);display:flex;min-height:100vh;font-size:14px;
  background-image:radial-gradient(ellipse 80% 50% at 90%
0%,rgba(0,212,170,.035) 0%,transparent 55%),radial-gradient(ellipse 50% 40%
at 0% 100%,rgba(0,212,170,.02) 0%,transparent 50%)}

::-webkit-scrollbar{width:5px;height:5px}
::-webkit-scrollbar-track{background:var(--bg2)}
::-webkit-scrollbar-thumb{background:rgba(0,212,170,.25);border-radius:3px}

.sidebar{width:var(--sb);background:var(--
bg2);position:fixed;top:0;left:0;bottom:0;display:flex;flex-
direction:column;border-right:1px solid var(--brd2);z-
index:900;transition:transform var(--tr);box-shadow:4px 0 30px
rgba(0,0,0,.6)}
.sb-top{padding:1.4rem 1.3rem 1.1rem;border-bottom:1px solid var(--brd2)}
.sb-logo{display:flex;align-items:center;gap:11px;margin-bottom:.9rem}
.sb-icon{width:44px;height:44px;border-radius:11px;background:linear-
gradient(135deg,#009980,#00D4AA);display:flex;align-items:center;justify-
content:center;font-size:1.5rem;flex-shrink:0;box-shadow:0 0 22px
rgba(0,212,170,.4)}
.sb-brand h2{font-size:1.02rem;color:var(--teal2);font-weight:700;letter-
spacing:-.01em}
.sb-brand p{font-size:.62rem;color:var(--tx3);margin-top:2px}
.sb-live{display:inline-flex;align-
items:center;gap:7px;background:rgba(188,255,0,.07);border:1px solid
rgba(188,255,0,.18);color:var(--lime);padding:3px 10px;border-
radius:20px;font-size:.63rem;font-weight:700}
.sb-live::before{content:'';width:5px;height:5px;border-
radius:50%;background:var(--lime);animation:bl 2s infinite;flex-shrink:0}
@keyframes bl{0%,100%{opacity:1;box-shadow:0 0 0 0
rgba(188,255,0,.4)}50%{opacity:.2;box-shadow:0 0 0 4px rgba(188,255,0,0)}}
.sb-nav{flex:1;padding:.5rem 0;overflow-y:auto}
.sb-sec{padding:.5rem 1.2rem .18rem;font-size:.58rem;text-
transform:uppercase;letter-spacing:.14em;color:var(--tx3);font-
weight:700;font-family:'Space Mono',monospace}
.ni{display:flex;align-items:center;gap:10px;padding:.7rem
1.2rem;color:var(--tx2);cursor:pointer;font-size:.83rem;font-
weight:500;transition:all var(--tr);text-decoration:none;border-left:2px
solid transparent}
.ni:hover{background:var(--tealA);color:var(--tx);border-left-
color:rgba(0,212,170,.3)}
```

```
.ni.active{background:rgba(0,212,170,.1);color:var(--teal2);border-left-color:var(--teal);font-weight:600}
.ni i{width:16px;text-align:center;font-size:.8rem;flex-shrink:0}
.sb-foot{padding:.75rem 1.2rem;border-top:1px solid var(--brd2);font-family:'Space Mono',monospace;font-size:.58rem;color:var(--tx3);line-height:1.8}

.main{margin-left:var(--sb);flex:1;min-height:100vh;padding:1.1rem;transition:margin var(--tr)}

.topbar{background:var(--bg2);border:1px solid var(--brd2);border-radius:var(--ra);padding:.9rem 1.4rem;margin-bottom:1rem;display:flex;align-items:center;justify-content:space-between;flex-wrap:wrap;gap:.7rem}
.topbar h1{font-size:1.1rem;font-weight:700;display:flex;align-items:center;gap:9px}
.topbar p{font-size:.7rem;color:var(--tx3);margin-top:2px}
.tbr{display:flex;align-items:center;gap:.45rem;flex-wrap:wrap}
.hbg{display:inline-flex;align-items:center;gap:5px;padding:3px 10px;border-radius:20px;font-size:.65rem;font-weight:700}
.hbg.gr{background:rgba(255,77,77,.1);border:1px solid rgba(255,77,77,.22);color:#FF9999}
.hbg.lm{background:rgba(188,255,0,.07);border:1px solid rgba(188,255,0,.18);color:var(--lime)}
#hT{font-size:.66rem;color:var(--tx3);font-family:'Space Mono',monospace}

.loc-bar{background:var(--bg2);border:1px solid var(--brd2);border-radius:var(--ra);padding:.8rem 1.3rem;margin-bottom:1rem}
.loc-ttl{font-size:.64rem;color:var(--tx3);margin-bottom:.65rem;display:flex;align-items:center;gap:7px;font-weight:600;font-family:'Space Mono',monospace;text-transform:uppercase;letter-spacing:.08em}
.loc-grid{display:grid;grid-template-columns:1fr 1fr 1fr;gap:.7rem}
.fg label{display:block;font-size:.6rem;margin-bottom:3px;color:var(--tx3);font-weight:700;text-transform:uppercase;letter-spacing:.08em;font-family:'Space Mono',monospace}
.fc{width:100%;padding:6px 10px;border:1px solid var(--brd2);border-radius:7px;font-size:.81rem;font-family:'Outfit',sans-serif;background:var(--card);color:var(--tx);transition:all var(--tr)}
.fc:focus{outline:none;border-color:var(--teal);background:var(--card2);box-shadow:0 0 0 3px rgba(0,212,170,.1)}
.fc option{background:var(--bg2)}

.panel{display:none;animation:fadeUp .26s ease}
.panel.active{display:block}
@keyframes fadeUp{from{opacity:0;transform:translateY(7px)}to{opacity:1;transform:translateY(0)}}

.card{background:var(--bg2);border:1px solid var(--brd2);border-radius:var(--ra);padding:1.3rem;margin-bottom:1rem}
```

```
.card-ttl{font-size:.97rem;font-weight:700;display:flex;align-
items:center;gap:8px;color:var(--tx);margin-bottom:.25rem}
.card-sub{font-size:.74rem;color:var(--tx3);margin-bottom:1rem;line-
height:1.6}

.drop{border:1px dashed rgba(0,212,170,.18);border-
radius:12px;padding:2.2rem 1.4rem;text-
align:center;cursor:pointer;background:var(--card);transition:all var(--
tr);position:relative;overflow:hidden}
.drop:hover,.drop.over{border-color:var(--teal);background:var(--
card2);border-style:solid}
.drop-ico{font-size:2.4rem;color:var(--teal);margin-
bottom:.6rem;display:block;transition:transform var(--tr)}
.drop:hover .drop-ico{transform:translateY(-5px)}
.drop h3{font-size:.9rem;font-weight:600;color:var(--tx);margin-
bottom:.25rem}
.drop p{font-size:.74rem;color:var(--tx3)}
.drop small{font-size:.67rem;color:var(--
tx3);opacity:.55;display:block;margin-top:.3rem}
.pstrip{display:flex;align-items:center;gap:.75rem;background:var(--
card);border:1px solid var(--brd);border-radius:8px;padding:.55rem
.85rem;margin-top:.75rem}
.pstrip img{width:50px;height:50px;object-fit:cover;border-
radius:6px;border:1px solid var(--brd)}
.pstrip .pi strong{display:block;font-size:.82rem;color:var(--tx)}
.pstrip .pi small{color:var(--tx3);font-size:.69rem}

.row{display:flex;gap:.55rem;flex-wrap:wrap;margin-top:.9rem}
.btn{display:inline-flex;align-items:center;gap:6px;padding:7px
17px;border:none;border-radius:8px;font-size:.8rem;font-weight:600;font-
family:'Outfit',sans-serif;cursor:pointer;transition:all var(--tr);white-
space:nowrap}
.bp{background:var(--teal);color:#000;box-shadow:0 0 18px
rgba(0,212,170,.25)}
.bp:hover{background:var(--teal2);transform:translateY(-2px);box-shadow:0 0
26px rgba(0,212,170,.4)}
.bg{background:#14532D;color:#86EFAC}
.bg:hover{background:#166534;transform:translateY(-2px)}
.bd{background:rgba(239,68,68,.15);color:#FCA5A5;border:1px solid
rgba(239,68,68,.25)}
.bd:hover{background:rgba(239,68,68,.28)}
.bo{background:transparent;border:1px solid var(--brd);color:var(--tx2)}
.bo:hover{border-color:var(--teal);color:var(--teal);background:var(--
tealA)}
.btn:disabled{opacity:.25;cursor:not-allowed;transform:none!important;box-
shadow:none!important}
.sm{padding:5px 11px;font-size:.71rem}

.res-wrap{margin-top:1.1rem}
```

```
.pills-row{display:flex;gap:.38rem;flex-wrap:wrap;margin-bottom:.7rem}
.pill{display:inline-flex;align-items:center;gap:5px;padding:4px
13px;border-radius:20px;font-size:.74rem;font-weight:600}
.pill.M{background:rgba(34,197,94,.1);color:#4ADE80;border:1px solid
rgba(34,197,94,.2)}
.pill.V{background:rgba(245,158,11,.1);color:#FCD34D;border:1px solid
rgba(245,158,11,.2)}
.pill.P{background:rgba(239,68,68,.1);color:#FCA5A5;border:1px solid
rgba(239,68,68,.2)}

.blk-top{display:grid;grid-template-columns:1fr 370px;gap:1rem;margin-
bottom:1rem;align-items:start}

.img-box{background:#000;border:1px solid var(--brd);border-radius:var(--
ra);overflow:hidden;position:relative;box-shadow:var(--sl),var(--glow)}
.img-box img{width:100%;display:block;max-height:480px;object-
fit:contain;background:#000;cursor:zoom-in;transition:transform .35s ease}
.img-box:hover img{transform:scale(1.008)}
.ov-top{position:absolute;top:0;left:0;right:0;padding:.65rem
.85rem;background:linear-gradient(to
bottom,rgba(0,0,0,.8),transparent);display:flex;align-items:flex-
start;justify-content:space-between;flex-wrap:wrap;gap:.35rem;pointer-
events:none}
.ov-bot{position:absolute;bottom:0;left:0;right:0;padding:.6rem
.85rem;background:linear-
gradient(transparent,rgba(0,0,0,.85));display:flex;align-items:flex-
end;justify-content:space-between;pointer-events:none}
.ov-bot button{pointer-events:all}
.itags{display:flex;flex-wrap:wrap;gap:.28rem}
.itag{display:inline-flex;align-
items:center;gap:3px;background:rgba(0,212,170,.18);border:1px solid
rgba(0,212,170,.3);color:var(--teal2);padding:2px 7px;border-
radius:20px;font-size:.61rem;font-weight:700;font-family:'Space
Mono',monospace;backdrop-filter:blur(6px)}
.itag.r{background:rgba(255,77,77,.18);border-
color:rgba(255,77,77,.3);color:#FF9999}
.itag.g{background:rgba(34,197,94,.18);border-
color:rgba(34,197,94,.3);color:#86EFAC}
.itag.a{background:rgba(255,179,0,.18);border-
color:rgba(255,179,0,.3);color:#FCD34D}
.img-ts{font-size:.6rem;color:rgba(255,255,255,.38);font-family:'Space
Mono',monospace}
.zoom-btn{background:rgba(255,255,255,.1);border:1px solid
rgba(255,255,255,.18);color:#fff;padding:4px 10px;border-
radius:6px;cursor:pointer;font-size:.67rem;font-weight:600;font-
family:'Outfit',sans-serif;backdrop-filter:blur(6px);transition:all var(--
tr);display:flex;align-items:center;gap:4px}
.zoom-btn:hover{background:rgba(255,255,255,.22)}
```

```
.det-panel{background:var(--bg2);border:1px solid var(--brd2);border-
radius:var(--ra);overflow:hidden;display:flex;flex-
direction:column;height:100%}
.det-hdr{padding:.75rem 1.05rem;border-bottom:1px solid var(--
brd2);display:flex;align-items:center;justify-content:space-
between;background:var(--card);flex-shrink:0}
.det-hdr h3{font-size:.84rem;font-weight:700;color:var(--
tx);display:flex;align-items:center;gap:7px}
.det-badge-cnt{background:var(--
teal);color:#000;width:20px;height:20px;border-radius:50%;display:inline-
flex;align-items:center;justify-content:center;font-size:.66rem;font-
weight:700}
.det-sub{font-size:.6rem;color:var(--tx3);font-family:'Space
Mono',monospace}
.det-list{list-style:none;padding:.5rem;overflow-y:auto;flex:1}
.det-item{display:flex;align-items:center;gap:.7rem;border-
radius:7px;padding:.62rem .8rem;margin-bottom:.35rem;border:1px solid
transparent;transition:all var(--tr);cursor:default}
.det-item:last-child{margin-bottom:0}
.det-item:hover{transform:translateX(3px)}
.det-item.maduro{border-
color:rgba(34,197,94,.18);background:rgba(34,197,94,.055)}
.det-item.verde{border-
color:rgba(245,158,11,.18);background:rgba(245,158,11,.055)}
.det-item.podrido{border-
color:rgba(239,68,68,.18);background:rgba(239,68,68,.055)}
.det-item.desconocido{border-
color:rgba(107,114,128,.15);background:rgba(107,114,128,.04)}
.det-emo{font-size:1.7rem;flex-shrink:0;line-height:1}
.det-body{flex:1;min-width:0}
.det-row1{display:flex;align-items:center;justify-content:space-
between;margin-bottom:2px}
.det-name{font-weight:700;font-size:.85rem;color:var(--tx)}
.det-lv{padding:2px 7px;border-radius:5px;font-size:.61rem;font-weight:700}
.det-lv.maduro{background:rgba(34,197,94,.18);color:#4ADE80}
.det-lv.verde{background:rgba(245,158,11,.18);color:#FCD34D}
.det-lv.podrido{background:rgba(239,68,68,.18);color:#FCA5A5}
.det-lv.desconocido{background:rgba(107,114,128,.15);color:#9CA3AF}
.det-info{font-size:.67rem;color:var(--tx3);margin-bottom:4px}
.cbar-row{display:flex;align-items:center;gap:.4rem}
.cbar-track{flex:1;height:3px;border-
radius:2px;background:rgba(255,255,255,.07);overflow:hidden}
.cbar-fill{height:100%;border-radius:2px;transition:width .9s cubic-
bezier(.4,0,.2,1)}
.maduro .cbar-fill{background:var(--mad)} .verde .cbar-fill{background:var(-
-ver)} .podrido .cbar-fill{background:var(--pod)} .desconocido .cbar-
fill{background:#6B7280}
```

```
.cbar-pct{font-size:.67rem;font-weight:700;min-width:34px;text-
align:right;font-family:'Space Mono',monospace}
.maduro .cbar-pct{color:var(--mad)} .verde .cbar-pct{color:var(--ver)}
.podrido .cbar-pct{color:var(--pod)}
.no-det{background:rgba(245,158,11,.06);border:1px solid
rgba(245,158,11,.18);border-radius:8px;padding:.9rem
1rem;color:#FCD34D;font-size:.78rem;display:flex;align-items:flex-
start;gap:.65rem;margin:.5rem}

.iq-block{background:var(--bg2);border:1px solid var(--brd2);border-
radius:var(--ra);padding:1.1rem 1.4rem;margin-bottom:1rem}
.iq-inner{display:flex;align-items:center;gap:1.8rem;flex-wrap:wrap}
.iq-num{font-family:'Space Mono',monospace;font-size:3.4rem;font-
weight:700;line-height:1;flex-shrink:0;text-shadow:0 0 30px currentColor}
.iq-right{flex:1;min-width:200px}
.iq-label{font-size:.62rem;text-transform:uppercase;letter-
spacing:.12em;color:var(--tx3);margin-bottom:.45rem;font-family:'Space
Mono',monospace;display:flex;align-items:center;gap:6px}
.iq-track{height:10px;background:rgba(255,255,255,.06);border-
radius:5px;overflow:hidden;margin-bottom:.5rem;position:relative}
.iq-fill{height:100%;border-radius:5px;transition:width 1.4s cubic-
bezier(.4,0,.2,1);position:relative}
.iq-
fill::after{content:'';position:absolute;top:0;right:0;bottom:0;width:30px;b
ackground:linear-gradient(90deg,transparent,rgba(255,255,255,.25))}
.iq-low{background:linear-gradient(90deg,#991B1B,#EF4444)} .iq-
med{background:linear-gradient(90deg,#92400E,#F59E0B)} .iq-
hi{background:linear-gradient(90deg,#065F46,#10B981)}
.iq-desc{font-size:.77rem;color:var(--tx2);line-height:1.55}
.iq-stats{display:flex;gap:1rem;margin-top:.6rem;flex-wrap:wrap}
.iq-stat{font-size:.68rem;color:var(--tx3);display:flex;align-
items:center;gap:4px}
.iq-stat b{color:var(--tx2);font-weight:600}

.agro-wrap{background:var(--bg2);border:1px solid var(--brd2);border-
radius:var(--ra);overflow:hidden;margin-bottom:1rem}
.agro-hdr{background:var(--card);padding:.88rem 1.4rem;border-bottom:1px
solid var(--brd2);display:flex;align-items:center;justify-content:space-
between;flex-wrap:wrap;gap:.6rem}
.agro-hdr-left{display:flex;align-items:center;gap:10px}
.agro-hdr h2{font-size:.95rem;font-weight:700;color:var(--
teal2);display:flex;align-items:center;gap:8px}
.agro-meta{display:flex;align-items:center;gap:.45rem;flex-wrap:wrap}
.mk-groq{display:inline-flex;align-
items:center;gap:5px;background:rgba(255,77,77,.12);border:1px solid
rgba(255,77,77,.25);color:#FF9999;padding:3px 9px;border-radius:20px;font-
size:.63rem;font-weight:700}
.mk-local{background:rgba(255,255,255,.05);color:var(--tx3);border:1px solid
var(--brd2);padding:3px 9px;border-radius:20px;font-size:.63rem}
```

```
.mk-mad{display:inline-flex;align-
items:center;gap:5px;background:rgba(0,212,170,.12);border:1px solid
rgba(0,212,170,.2);color:var(--teal2);padding:3px 11px;border-
radius:20px;font-size:.72rem;font-weight:700}

.price-hero{display:flex;align-items:center;justify-content:space-
between;flex-wrap:wrap;gap:.7rem;margin:.8rem 1.4rem;background:linear-
gradient(135deg,rgba(0,212,170,.1) 0%,rgba(0,212,170,.04) 100%);border:1px
solid rgba(0,212,170,.22);border-radius:var(--ra2);padding:1rem 1.3rem}
.price-big{font-family:'Space Mono',monospace;font-size:2rem;font-
weight:700;color:var(--teal2);text-shadow:0 0 20px rgba(0,212,170,.35)}
.price-sub{font-size:.66rem;color:var(--tx3);margin-top:2px}
.exp-pill{display:inline-flex;align-items:center;gap:5px;padding:6px
14px;border-radius:8px;font-size:.73rem;font-weight:700}
.exp-y{background:rgba(167,139,250,.13);border:1px solid
rgba(167,139,250,.26);color:var(--purple)}
.exp-n{background:rgba(239,68,68,.1);border:1px solid
rgba(239,68,68,.22);color:#FCA5A5}

.agro-tabs{display:flex;gap:.28rem;padding:.72rem 1.4rem;border-bottom:1px
solid var(--brd2);flex-wrap:wrap;background:linear-gradient(var(--
card),var(--bg2))}
.tab{padding:5px 13px;border-radius:7px;font-size:.73rem;font-
weight:600;cursor:pointer;font-family:'Outfit',sans-
serif;background:rgba(255,255,255,.04);color:var(--tx3);border:1px solid
rgba(255,255,255,.05);transition:all var(--tr)}
.tab:hover{background:rgba(0,212,170,.09);color:var(--teal2);border-
color:rgba(0,212,170,.18)}
.tab.on{background:rgba(0,212,170,.14);color:var(--teal2);border-color:var(-
-teal)}
.pane{display:none}
.pane.on{display:block;animation:fadeUp .2s ease}

.ag3{display:grid;grid-template-
columns:repeat(3,1fr);gap:.85rem;padding:1.1rem 1.4rem}
.ag2{display:grid;grid-template-columns:1fr 1fr;gap:.85rem;padding:1.1rem
1.4rem}
.ag4{display:grid;grid-template-
columns:repeat(4,1fr);gap:.85rem;padding:1.1rem 1.4rem}
.span2{grid-column:1/-1}
.abox{background:var(--card);border:1px solid var(--brd2);border-
radius:var(--ra2);padding:.85rem .95rem;transition:border-color var(--tr)}
.abox:hover{border-color:rgba(0,212,170,.15)}
.abox.tl{border-left:3px solid var(--teal)} .abox.ta{border-left:3px solid
var(--amber)} .abox.tc{border-left:3px solid var(--coral)} .abox.tp{border-
left:3px solid var(--purple)} .abox.tg{border-left:3px solid var(--mad)}
.abox.tb{border-left:3px solid var(--blue)}
```

```
.abox h4{font-size:.62rem;text-transform:uppercase;letter-
spacing:.09em;margin-bottom:.42rem;display:flex;align-
items:center;gap:5px;font-weight:700;font-family:'Space Mono',monospace}
.abox.tl h4{color:var(--teal)} .abox.ta h4{color:var(--amber)} .abox.tc
h4{color:var(--coral)} .abox.tp h4{color:var(--purple)} .abox.tg
h4{color:var(--mad)} .abox.tb h4{color:var(--blue)}
.abox p{font-size:.79rem;line-height:1.72;color:var(--tx2)}
.abox h4 i{width:13px;text-align:center}

.notas-zone{padding:1rem 1.4rem}
.notas-zone label{font-size:.6rem;color:var(--tx3);display:block;margin-
bottom:.4rem;text-transform:uppercase;letter-spacing:.09em;font-
family:'Space Mono',monospace}
.notas-zone textarea{width:100%;padding:.65rem .9rem;border-
radius:8px;border:1px solid var(--brd2);background:var(--card);color:var(--
tx);font-size:.79rem;resize:vertical;min-height:72px;font-
family:'Outfit',sans-serif;transition:all var(--tr);line-height:1.6}
.notas-zone textarea:focus{outline:none;border-color:var(--teal);box-
shadow:0 0 0 3px rgba(0,212,170,.1)}
.notas-zone textarea::placeholder{color:var(--tx3);opacity:.5}

.dm-
overlay{display:none;position:fixed;inset:0;background:rgba(0,0,0,.94);z-
index:5000;overflow-y:auto}
.dm-overlay.on{display:block;animation:fadeIn .2s ease}
@keyframes fadeIn{from{opacity:0}to{opacity:1}}
.dm-shell{max-width:1320px;margin:0 auto;padding:1.3rem 1.1rem}
.dm-bar{display:flex;align-items:center;justify-content:space-
between;gap:.7rem;padding:.82rem 1.2rem;background:var(--bg2);border:1px
solid var(--brd2);border-radius:var(--ra);margin-bottom:1rem}
.dm-bar h2{font-size:.93rem;font-weight:700;color:var(--
teal2);display:flex;align-items:center;gap:8px;min-
width:0;overflow:hidden;text-overflow:ellipsis;white-space:nowrap}
.dm-close{display:flex;align-
items:center;gap:6px;background:rgba(255,255,255,.06);border:1px solid
rgba(255,255,255,.1);color:var(--tx2);padding:6px 15px;border-
radius:7px;cursor:pointer;font-size:.77rem;font-weight:600;font-
family:'Outfit',sans-serif;transition:all var(--tr);flex-shrink:0}
.dm-close:hover{background:rgba(239,68,68,.18);color:#FCA5A5;border-
color:rgba(239,68,68,.28)}

.zm-
overlay{display:none;position:fixed;inset:0;background:rgba(0,0,0,.97);z-
index:6000;align-items:center;justify-content:center;cursor:zoom-out}
.zm-overlay.on{display:flex;animation:fadeIn .16s ease}
.zm-overlay img{max-width:96vw;max-height:95vh;object-fit:contain;border-
radius:10px}
.zm-
close{position:fixed;top:14px;right:17px;background:rgba(255,255,255,.1);col
```

```
or:#fff;border:1px solid rgba(255,255,255,.18);border-
radius:50%;width:38px;height:38px;display:flex;align-items:center;justify-
content:center;cursor:pointer;font-size:1rem;transition:all var(--tr)}
.zm-close:hover{background:rgba(239,68,68,.3)}

.lib-ctrl{display:flex;gap:.55rem;flex-wrap:wrap;margin-bottom:.9rem;align-
items:center}
.lib-q{flex:1;min-width:190px;padding:6px 10px;border:1px solid var(--
brd2);border-radius:7px;font-size:.8rem;font-family:'Outfit',sans-
serif;background:var(--card);color:var(--tx);transition:all var(--tr)}
.lib-q:focus{outline:none;border-color:var(--teal)}
.lib-q::placeholder{color:var(--tx3)}
.lgrid{display:grid;grid-template-columns:repeat(auto-
fill,minmax(265px,1fr));gap:.9rem}
.lcard{background:var(--bg2);border-radius:12px;overflow:hidden;border:1px
solid var(--brd2);transition:all var(--tr)}
.lcard:hover{transform:translateY(-4px);border-
color:rgba(0,212,170,.28);box-shadow:0 0 28px rgba(0,212,170,.09)}
.lc-
img{height:158px;overflow:hidden;position:relative;background:#000;cursor:po
inter}
.lc-img img{width:100%;height:100%;object-fit:cover;transition:transform .4s
ease}
.lcard:hover .lc-img img{transform:scale(1.07)}
.lc-
iq{position:absolute;bottom:7px;right:7px;background:rgba(0,0,0,.78);color:#
fff;border-radius:20px;padding:2px 9px;font-size:.63rem;font-
weight:700;font-family:'Space Mono',monospace;backdrop-filter:blur(4px)}
.lc-body{padding:.8rem .95rem}
.lc-title{font-weight:700;font-size:.84rem;margin-bottom:.26rem;color:var(--
tx)}
.lc-meta{font-size:.68rem;color:var(--tx3);display:flex;flex-
wrap:wrap;gap:.26rem;margin-bottom:.42rem}
.lc-acts{display:flex;gap:.38rem;margin-top:.48rem}
.tag{display:inline-block;padding:2px 7px;border-radius:5px;font-
size:.61rem;font-weight:600}
.td{background:rgba(34,197,94,.1);color:#4ADE80}
.tnd{background:rgba(239,68,68,.1);color:#FCA5A5}
.tr{background:rgba(96,165,250,.1);color:#93C5FD}
.tm{background:rgba(34,197,94,.1);color:#4ADE80}
.tv{background:rgba(245,158,11,.1);color:#FCD34D}
.tp2{background:rgba(239,68,68,.1);color:#FCA5A5}
.tg2{background:rgba(0,212,170,.1);color:var(--teal2)}

.sw{background:var(--bg2);border:1px solid var(--brd2);border-radius:var(--
ra);padding:1.4rem}
.sh{display:flex;align-items:center;justify-content:space-between;flex-
wrap:wrap;gap:.65rem;margin-bottom:1.2rem}
```

```
.sh h2{font-size:1.05rem;font-weight:700;color:var(--
teal2);display:flex;align-items:center;gap:8px}
.kpis{display:grid;grid-template-columns:repeat(auto-
fill,minmax(118px,1fr));gap:.7rem;margin-bottom:1.3rem}
.kpi{background:var(--card);border:1px solid var(--brd2);border-
radius:10px;padding:.9rem;text-align:center;transition:all var(--tr)}
.kpi:hover{transform:translateY(-2px);border-color:var(--teal)}
.kpi-ico{font-size:1.2rem;margin-bottom:.25rem;opacity:.6}
.kpi-v{font-family:'Space Mono',monospace;font-size:1.65rem;font-
weight:700;color:var(--teal2)}
.kpi-l{font-size:.61rem;opacity:.45;margin-top:2px}
.charts{display:grid;grid-template-columns:1fr 1fr;gap:.8rem;margin-
bottom:.8rem}
.cb{background:var(--card);border-radius:10px;padding:.85rem;border:1px
solid var(--brd2)}
.cb h4{font-size:.73rem;color:var(--teal2);margin-
bottom:.5rem;display:flex;align-items:center;gap:6px;font-weight:600}
.cw{position:relative;height:190px;width:100%}
.cw canvas{position:absolute;inset:0}
.stats-row{display:grid;grid-template-columns:repeat(auto-
fill,minmax(175px,1fr));gap:.65rem;margin-bottom:.8rem}
.sbox{background:var(--card);border:1px solid var(--brd2);border-
radius:9px;padding:.8rem .95rem}
.sbox h5{font-size:.6rem;opacity:.45;margin-bottom:.28rem;text-
transform:uppercase;letter-spacing:.06em;font-family:'Space Mono',monospace}
.sbox .sv{font-family:'Space Mono',monospace;font-size:1rem;font-
weight:700;color:var(--teal2)}
.sbox .ss{font-size:.65rem;opacity:.4;margin-top:2px}

.logbox{background:#000;color:#00D4AA;border-radius:10px;padding:.95rem;max-
height:410px;overflow-y:auto;font-family:'Space Mono',monospace;font-
size:.67rem;line-height:1.85;border:1px solid var(--brd2)}
.logbox div:nth-child(odd){background:rgba(255,255,255,.012)}

.hgrid{display:grid;grid-template-columns:repeat(auto-
fill,minmax(235px,1fr));gap:.9rem}
.hcard{background:var(--card);border-radius:11px;padding:1.15rem;border:1px
solid var(--brd2);border-top:3px solid var(--teal)}
.hcard h4{color:var(--teal2);margin-bottom:.48rem;display:flex;align-
items:center;gap:7px;font-size:.85rem;font-weight:600}
.hcard p,.hcard li{font-size:.76rem;color:var(--tx2);line-height:1.74}
.hcard ul{padding-left:1rem}
.chips{display:flex;flex-wrap:wrap;gap:.28rem;margin-top:.42rem}
.chip{background:var(--card2);border-radius:5px;padding:2px 8px;font-
size:.69rem;color:var(--tx2);border:1px solid var(--brd2)}

.ov{display:none;position:fixed;inset:0;background:rgba(0,14,11,.93);z-
index:2000;flex-direction:column;align-items:center;justify-
content:center;color:#fff}
```

```
.ov.on{display:flex;animation:fadeIn .2s ease}
.spin{width:50px;height:50px;border:3px solid rgba(0,212,170,.12);border-top-color:var(--teal);border-radius:50%;animation:sp .7s linear infinite;margin-bottom:1.1rem;box-shadow:0 0 22px rgba(0,212,170,.18)}
@keyframes sp{to{transform:rotate(360deg)}}
.ov h3{font-size:1rem;margin-bottom:.28rem;color:var(--teal2);font-weight:600}
.ov p{font-size:.74rem;color:var(--tx3);text-align:center;max-width:260px}
.ov-steps{margin-top:.9rem;display:flex;flex-direction:column;gap:.28rem;text-align:center}
.ov-step{font-size:.68rem;color:var(--tx3);opacity:.28;transition:all .35s;font-family:'Space Mono',monospace}
.ov-step.act{opacity:1;color:var(--teal2)}
.ov-step.done{opacity:.3;text-decoration:line-through}

.toast{position:fixed;top:14px;right:14px;padding:.75rem 1.15rem;border-radius:9px;color:#fff;display:flex;align-items:center;gap:8px;z-index:4000;transform:translateX(140%);transition:transform .26s cubic-bezier(.4,0,.2,1);box-shadow:var(--sl);max-width:330px;font-size:.78rem;font-weight:500}
.toast.show{transform:translateX(0)}
.tok{background:#14532D;border:1px solid var(--mad)}
.terr{background:#7F1D1D;border:1px solid var(--pod)} .tinf{background:var(--card2);border:1px solid var(--teal)} .twrn{background:#78350F;border:1px solid var(--amber)}

.cam-wrap{position:relative;background:#000;border-radius:11px;overflow:hidden;aspect-ratio:16/9;max-width:660px;margin:0 auto .8rem;border:1px solid var(--brd)}
.cam-wrap video{width:100%;height:100%;object-fit:cover;display:block}

.mbn{display:none;position:fixed;top:11px;left:11px;z-index:950;width:37px;height:37px;background:var(--bg2);color:var(--tx);border:1px solid var(--brd2);border-radius:8px;cursor:pointer;align-items:center;justify-content:center;font-size:.9rem}
.fx{display:flex} .jb{justify-content:space-between} .ac{align-items:center}
.fw{flex-wrap:wrap} .g1{gap:.42rem} .g2{gap:.8rem}
.mt1{margin-top:.45rem} .mt2{margin-top:.85rem}
.empty{text-align:center;padding:2.8rem;color:var(--tx3)}
.empty i{font-size:2.3rem;margin-bottom:.6rem;display:block;opacity:.22}

@media(max-width:1120px){.blk-top{grid-template-columns:1fr}.ag3,.ag4{grid-template-columns:1fr 1fr}}
@media(max-width:1024px){.charts{grid-template-columns:1fr}}
@media(max-width:768px){
  .sidebar{transform:translateX(-100%)}.sidebar.on{transform:translateX(0)}
  .main{margin-left:0;padding:.75rem;padding-top:3.2rem}
  .mbn{display:flex}.loc-grid{grid-template-columns:1fr}.ag2,.ag4{grid-template-columns:1fr}
```

```
}
</style>
</head>
<body>

<button class="mbn" onclick="sbToggle()"><i class="fas fa-
bars"></i></button>

<!-- SIDEBAR -->
<nav class="sidebar" id="sb">
  <div class="sb-top">
    <div class="sb-logo">
      <div class="sb-icon">🖐</div>
      <div class="sb-brand">
        <h2>NeuroAgro</h2>
        <p>Clasificador Inteligente v5.2</p>
      </div>
    </div>
    <span class="sb-live">YOLO + Groq · Activo</span>
  </div>
  <div class="sb-nav">
    <div class="sb-sec">// análisis</div>
    <a class="ni active" onclick="show('upload')"  href="#"><i class="fas
fa-file-image"></i> Subir Imagen</a>
    <a class="ni"        onclick="show('camera')"  href="#"><i class="fas
fa-video"></i>      Cámara Live</a>
    <div class="sb-sec">// datos</div>
    <a class="ni"        onclick="show('library')" href="#"><i class="fas
fa-images"></i>     Biblioteca</a>
    <a class="ni"        onclick="show('stats')"   href="#"><i class="fas
fa-chart-bar"></i>  Estadísticas</a>
    <div class="sb-sec">// sistema</div>
    <a class="ni"        onclick="show('logs')"    href="#"><i class="fas
fa-file-alt"></i>   Logs</a>
    <a class="ni"        onclick="show('help')"    href="#"><i class="fas
fa-question-circle"></i> Ayuda</a>
  </div>
  <div class="sb-foot">YOLOv8s · Groq LLaMA 3.3 70B<br>© 2026
NeuroAgro</div>
</nav>

<main class="main" id="main">

  <div class="topbar">
    <div>
      <h1>🖐 FruitAI — Clasificador de Frutas Peruanas</h1>
      <p>Detección YOLO + Análisis Agronómico Completo · Groq LLaMA 70B ·
Perú 2026</p>
    </div>
```

```
    <div class="tbr">
      <span class="hbg gr"><i class="fas fa-bolt"></i> Groq LLaMA 70B</span>
      <span class="hbg lm"><i class="fas fa-leaf"></i> 2026</span>
      <span id="hT"></span>
    </div>
  </div>

  <div class="loc-bar">
    <div class="loc-ttl"><i class="fas fa-satellite"></i> Ubicación del
análisis</div>
    <div class="loc-grid">
      <div class="fg"><label>Región del Perú</label>
        <select id="regSel" class="fc" onchange="loadDep()">
          <option value="">Seleccionar región...</option>

<option>Costa</option><option>Sierra</option><option>Selva</option>
        </select>
      </div>
      <div class="fg"><label>Departamento</label>
        <select id="depSel" class="fc"><option value="">Seleccionar
departamento...</option></select>
      </div>
      <div class="fg"><label>Localidad (opcional)</label>
        <input id="locIn" class="fc" placeholder="Ej. Valle de Chancay"/>
      </div>
    </div>
  </div>

  <!-- UPLOAD -->
  <section id="uploadPanel" class="panel active">
    <div class="card">
      <div class="card-ttl"><i class="fas fa-file-image"></i> Subir Imagen
para Análisis</div>
      <p class="card-sub">Detecta 9 frutas peruanas (Manzana · Mango · Piña
· Fresa · Plátano · Palta · Papaya · Uva · Naranja) en 3 estados. Análisis
completo con Groq LLaMA 70B.</p>
      <div class="drop" id="dropA"
           onclick="document.getElementById('fIn').click()"
           ondragover="event.preventDefault();this.classList.add('over')"
           ondragleave="this.classList.remove('over')"
           ondrop="onDrop(event)">
        <i class="fas fa-cloud-upload-alt drop-ico"></i>
        <h3>Arrastra tu imagen aquí</h3>
        <p>o haz clic para explorar archivos</p>
        <small>PNG · JPG · JPEG · GIF · BMP · WEBP — Máx. 16 MB</small>
      </div>
      <input type="file" id="fIn" accept="image/*" style="display:none"
onchange="onFS(event)"/>
      <div id="pStr" style="display:none" class="pstrip">
```

```
        <img id="pImg" src="" alt="prev"/>
        <div class="pi"><strong id="pName"></strong><small
id="pSize"></small></div>
        <button class="btn sm bo" onclick="clrUp()" style="margin-
left:auto"><i class="fas fa-times"></i></button>
      </div>
      <div class="row">
        <button class="btn bp" id="procBtn" onclick="procImg()" disabled><i
class="fas fa-magic"></i> Analizar con IA</button>
        <button class="btn bo" onclick="clrUp()"><i class="fas fa-redo"></i>
Limpiar</button>
      </div>
    </div>
    <div id="upRes"></div>
  </section>

    <section id="cameraPanel" class="panel">
    <div class="card">
      <div class="card-ttl"><i class="fas fa-video"></i> Cámara en
Vivo</div>
      <p class="card-sub">Captura y analiza frutas en tiempo real con YOLO +
Groq LLaMA</p>
      <div class="cam-wrap"><video id="vid" autoplay
playsinline></video></div>
      <div class="row" style="justify-content:center">
        <button class="btn bp" id="camBtn" onclick="tCam()"><i class="fas
fa-power-off"></i> Encender Cámara</button>
        <button class="btn bg" id="capBtn" onclick="capImg()" disabled><i
class="fas fa-camera"></i> Capturar y Analizar</button>
      </div>
      <div id="camRes" class="mt2"></div>
    </div>
  </section>

  <section id="libraryPanel" class="panel">
    <div class="card">
      <div class="fx jb ac fw" style="margin-bottom:.85rem;gap:.65rem">
        <div class="card-ttl" style="margin:0"><i class="fas fa-images"></i>
Biblioteca de Análisis</div>
        <div class="fx g1">
          <button class="btn sm bo" onclick="loadLib()"><i class="fas fa-
sync-alt"></i> Actualizar</button>
          <button class="btn sm bd" onclick="clrAll()"><i class="fas fa-
trash"></i> Limpiar todo</button>
        </div>
      </div>
      <div class="lib-ctrl">
        <input class="lib-q" id="libQ" placeholder="🔍 Buscar fruta,
región, estado..."/>
```

```
        <select class="fc" id="libF" onchange="flLib()" style="max-
width:158px">
          <option value="">Todos</option><option value="det">Con
detección</option><option value="no">Sin detección</option>
          <option value="mad">Maduros</option><option
value="ver">Verdes</option><option value="pod">Podridos</option><option
value="groq">Con Groq</option>
        </select>
        <select class="fc" id="libSort" onchange="flLib()" style="max-
width:150px">
          <option value="id">Más recientes</option><option value="iq">Mayor
calidad</option><option value="conf">Mayor confianza</option>
        </select>
      </div>
      <div id="lgrid" class="lgrid"><div class="empty"><i class="fas fa-
images"></i><p>Cargando...</p></div></div>
      <div id="lpag" class="fx g1 mt2" style="justify-content:center;flex-
wrap:wrap"></div>
    </div>
  </section>

  <section id="statsPanel" class="panel">
    <div class="sw">
      <div class="sh">
        <h2><i class="fas fa-chart-line"></i> Estadísticas Avanzadas</h2>
        <div class="fx g1">
          <button class="btn sm bo" onclick="loadSts()"><i class="fas fa-
sync-alt"></i> Actualizar</button>
          <button class="btn sm bd" onclick="rstSys()"><i class="fas fa-
redo"></i> Resetear</button>
        </div>
      </div>
      <div class="kpis">
        <div class="kpi"><div class="kpi-ico">📊</div><div class="kpi-v"
id="kT">—</div><div class="kpi-l">Total</div></div>
        <div class="kpi"><div class="kpi-ico">☑</div><div class="kpi-v"
id="kD">—</div><div class="kpi-l">Detectadas</div></div>
        <div class="kpi"><div class="kpi-ico">📈</div><div class="kpi-v"
id="kTR">—</div><div class="kpi-l">Tasa det.</div></div>
        <div class="kpi"><div class="kpi-ico">🍎</div><div class="kpi-v"
id="kM">—</div><div class="kpi-l">Maduras</div></div>
        <div class="kpi"><div class="kpi-ico">🍸</div><div class="kpi-v"
id="kV">—</div><div class="kpi-l">Verdes</div></div>
        <div class="kpi"><div class="kpi-ico">⚠</div><div class="kpi-v"
id="kP">—</div><div class="kpi-l">Podridas</div></div>
        <div class="kpi"><div class="kpi-ico">🌿</div><div class="kpi-v"
id="kE">—</div><div class="kpi-l">Especies</div></div>
        <div class="kpi"><div class="kpi-ico">⭐</div><div class="kpi-v"
id="kIQ">—</div><div class="kpi-l">Calidad prom.</div></div>
```

```
        <div class="kpi"><div class="kpi-ico">⚡</div><div class="kpi-v"
id="kGPT">—</div><div class="kpi-l">Con Groq</div></div>
      </div>
      <div class="stats-row">
        <div class="sbox"><h5>Región más activa</h5><div class="sv"
id="sReg">—</div><div class="ss">mayor análisis</div></div>
        <div class="sbox"><h5>Fruta más detectada</h5><div class="sv"
id="sFruta">—</div><div class="ss">en biblioteca</div></div>
        <div class="sbox"><h5>Mejor confianza</h5><div class="sv"
id="sConf">—</div><div class="ss">mayor precisión</div></div>
        <div class="sbox"><h5>Departamento líder</h5><div class="sv"
id="sDepto">—</div><div class="ss">más registros</div></div>
      </div>
      <div class="charts">
        <div class="cb"><h4><i class="fas fa-map-marked-alt"></i> Por
Región</h4><div class="cw"><canvas id="rC"></canvas></div></div>
        <div class="cb"><h4><i class="fas fa-apple-alt"></i> Top 8
Frutas</h4><div class="cw"><canvas id="fC"></canvas></div></div>
      </div>
      <div class="charts">
        <div class="cb"><h4><i class="fas fa-calendar-alt"></i> Actividad
Semanal</h4><div class="cw"><canvas id="aC"></canvas></div></div>
        <div class="cb"><h4><i class="fas fa-leaf"></i> Estado
Maduración</h4><div class="cw"><canvas id="mC"></canvas></div></div>
      </div>
      <div class="charts">
        <div class="cb"><h4><i class="fas fa-city"></i> Top
Departamentos</h4><div class="cw"><canvas id="dC"></canvas></div></div>
        <div class="cb"><h4><i class="fas fa-signal"></i> Confianza por
Clase</h4><div class="cw"><canvas id="cC"></canvas></div></div>
      </div>
    </div>
  </section>

  <section id="logsPanel" class="panel">
    <div class="card">
      <div class="fx jb ac" style="margin-bottom:.85rem">
        <div class="card-ttl" style="margin:0"><i class="fas fa-file-
alt"></i> Log del Sistema</div>
        <div class="fx g1">
          <button class="btn sm bo" onclick="loadLogs()"><i class="fas fa-
sync-alt"></i> Actualizar</button>
          <button class="btn sm bo" onclick="testGroq()"><i class="fas fa-
bolt"></i> Test Groq</button>
        </div>
      </div>
      <div class="logbox" id="lb"><em
style="opacity:.3">Cargando...</em></div>
    </div>
```

```
  </section>

  <section id="helpPanel" class="panel">
    <div class="card">
      <div class="card-ttl"><i class="fas fa-question-circle"></i> Centro de
Ayuda</div>
      <p class="card-sub">NeuroAgro v5.2 — YOLO + Groq LLaMA 70B
(gratuito)</p>
      <div class="hgrid">
        <div class="hcard"><h4><i class="fas fa-list-ol"></i> Flujo de
análisis</h4><ul><li>① Selecciona región y departamento</li><li>② Sube
imagen o usa la cámara</li><li>③ YOLO detecta + Groq analiza</li><li>④
Imagen anotada + Detecciones + Índice Calidad + Análisis
Agronómico</li><li>⑤ Biblioteca → "Ver Detalle" → abre panel
completo</li></ul></div>
        <div class="hcard"><h4><i class="fas fa-brain"></i> Campos del
análisis</h4><ul><li>💲 Precio estimado por kg</li><li>✈️ Calidad
exportación</li><li>🥗 Valor nutricional</li><li>🐛 Plagas y
riesgos</li><li>⭐ Índice calidad 0–100</li><li>🌡️ Temperatura
ideal</li><li>📅 Vida útil</li><li>🍸 Variedad detectada</li></ul></div>
        <div class="hcard"><h4><i class="fas fa-apple-alt"></i> Frutas
detectables</h4><div class="chips"><span class="chip">🟤
Manzana</span><span class="chip">🥭 Mango</span><span class="chip">🍍
Piña</span><span class="chip">🍓 Fresa</span><span class="chip">🍌
Plátano</span><span class="chip">🥑 Palta</span><span class="chip">🍈
Papaya</span><span class="chip">🍇 Uva</span><span class="chip">🍊
Naranja</span></div><p style="margin-top:.5rem">Estados: <b
style="color:var(--teal2)">Maduro · Verde · Podrido</b></p></div>
        <div class="hcard"><h4><i class="fas fa-cogs"></i>
Tecnología</h4><ul><li><b>YOLO:</b> YOLOv8s custom frutas
peruanas</li><li><b>IA:</b> Groq LLaMA 3.3 70B
(gratuito)</li><li><b>Preprocesado:</b> CLAHE + HSV boost</li><li><b>DB:</b>
SQLite 30+ campos</li><li><b>Backend:</b> Flask + Python</li></ul></div>
      </div>
    </div>
  </section>
</main>

<div class="ov" id="ov">
  <div class="spin"></div>
  <h3 id="ovH">Analizando...</h3>
  <p id="ovP">Por favor espera</p>
  <div class="ov-steps">
    <div class="ov-step" id="os1">① Preprocesando imagen (CLAHE+HSV)</div>
    <div class="ov-step" id="os2">② Detectando con YOLOv8s</div>
    <div class="ov-step" id="os3">③ Consultando Groq LLaMA 70B</div>
    <div class="ov-step" id="os4">④ Guardando en base de datos</div>
  </div>
</div>
```

```
<div class="toast" id="toast"><i id="tIco" class="fas fa-check-
circle"></i><span id="tMsg"></span></div>

<div class="dm-overlay" id="dmOverlay">
  <div class="dm-shell">
    <div class="dm-bar">
      <h2><i class="fas fa-leaf"></i> <span id="dmTitle">Detalle del
Análisis</span></h2>
      <button class="dm-close" onclick="closeDetail()"><i class="fas fa-
times"></i> Cerrar  <span style="opacity:.45;font-
size:.7rem">ESC</span></button>
    </div>
    <!-- Aquí se inyecta buildFullAnalysis(data, uid) -->
    <div id="dmBody"></div>
  </div>
</div>

<div class="zm-overlay" id="zmOverlay" onclick="closeZoom()">
  <button class="zm-close"><i class="fas fa-times"></i></button>
  <img id="zmImg" src="" alt="Ampliado"/>
</div>

<script>

let camStream=null, libData=[], libPage=1, charts={};
let libCache = {};   // id → datos completos cargados desde /biblioteca

(()=>{const e=document.getElementById('hT');const u=()=>e.textContent=new
Date().toLocaleString('es-
PE',{dateStyle:'short',timeStyle:'medium'});u();setInterval(u,1e3)})();

function sbToggle(){document.getElementById('sb').classList.toggle('on')}
document.addEventListener('click',e=>{
  const sb=document.getElementById('sb');

if(window.innerWidth<=768&&!sb.contains(e.target)&&!e.target.closest('.mbn')
)sb.classList.remove('on');
});

function show(n){

document.querySelectorAll('.panel').forEach(p=>p.classList.remove('active'))
;
  document.querySelectorAll('.ni').forEach(x=>x.classList.remove('active'));
  document.getElementById(n+'Panel').classList.add('active');

document.querySelectorAll('.ni').forEach(x=>{if(x.getAttribute('onclick')?.i
ncludes("'"+n+"'"))x.classList.add('active')});
```

```
  if(n==='library')loadLib();
  if(n==='stats')loadSts();
  if(n==='logs')loadLogs();

if(window.innerWidth<=768)document.getElementById('sb').classList.remove('on
');
}

let toastTimer;
function toast(msg,t='ok'){
  const el=document.getElementById('toast');
  el.className='toast show t'+t;
  document.getElementById('tIco').className='fas fa-'+(t==='ok'?'check-
circle':t==='err'?'times-circle':t==='wrn'?'exclamation-triangle':'info-
circle');
  document.getElementById('tMsg').textContent=msg;

clearTimeout(toastTimer);toastTimer=setTimeout(()=>el.classList.remove('show
'),4500);
}

let stepInt;
function showOv(h,p){
  document.getElementById('ovH').textContent=h||'Analizando...';
  document.getElementById('ovP').textContent=p||'Por favor espera';
  document.getElementById('ov').classList.add('on');

[1,2,3,4].forEach(i=>document.getElementById('os'+i).classList.remove('act',
'done'));
  let s=1;clearInterval(stepInt);
  stepInt=setInterval(()=>{
    if(s>1){const prev=document.getElementById('os'+(s-
1));prev.classList.remove('act');prev.classList.add('done')}
    if(s<=4)document.getElementById('os'+s).classList.add('act');
    s++;if(s>5)clearInterval(stepInt);
  },1900);
}
function
hideOv(){document.getElementById('ov').classList.remove('on');clearInterval(
stepInt)}

function loadDep(){
  const
r=document.getElementById('regSel').value,s=document.getElementById('depSel'
);
  s.innerHTML='<option value="">Seleccionar departamento...</option>';
  if(!r)return;

fetch('/departamentos/'+encodeURIComponent(r)).then(x=>x.json()).then(d=>d.f
```

```
orEach(v=>{const
o=document.createElement('option');o.textContent=v;s.appendChild(o)}));
}

function
openZoom(src){document.getElementById('zmImg').src=src;document.getElementBy
Id('zmOverlay').classList.add('on')}
function
closeZoom(){document.getElementById('zmOverlay').classList.remove('on')}

function openDetail(data){
  const dets=data.detecciones||[];
  document.getElementById('dmTitle').textContent=
    dets.length ? dets.map(d=>d.emoji+' '+d.clase).join(' · ').slice(0,70) :
'Análisis — '+(data.timestamp||'');
  const uid='dm_'+Date.now();
  document.getElementById('dmBody').innerHTML = buildFullAnalysis(data,
uid);
  document.getElementById('dmOverlay').classList.add('on');
  document.body.style.overflow='hidden';
  initTabs(document.getElementById('dmBody'));
  setTimeout(()=>animateBars(document.getElementById('dmBody')), 140);
}
function closeDetail(){
  document.getElementById('dmOverlay').classList.remove('on');
  document.body.style.overflow='';
}
document.addEventListener('keydown',e=>{if(e.key==='Escape'){closeDetail();c
loseZoom()}});

let selFile=null;
function
onDrop(ev){ev.preventDefault();document.getElementById('dropA').classList.re
move('over');const f=ev.dataTransfer.files[0];if(f)setFile(f)}
function onFS(ev){const f=ev.target.files[0];if(f)setFile(f)}
function setFile(f){
  selFile=f;
  document.getElementById('pImg').src=URL.createObjectURL(f);
  document.getElementById('pName').textContent=f.name;

document.getElementById('pSize').textContent=(f.size/1024/1024).toFixed(2)+'
MB';
  document.getElementById('pStr').style.display='flex';
  document.getElementById('procBtn').disabled=false;
}
function clrUp(){
  selFile=null;document.getElementById('fIn').value='';
  document.getElementById('pStr').style.display='none';
  document.getElementById('procBtn').disabled=true;
```

```
  document.getElementById('upRes').innerHTML='';
}
async function procImg(){
  if(!selFile){toast('Selecciona una imagen primero','wrn');return}
  const dep=document.getElementById('depSel').value;
  showOv('Analizando imagen','YOLO + Groq LLaMA 70B...');
  try{
    const fd=new
FormData();fd.append('imagen',selFile);fd.append('departamento',dep);
    const r=await fetch('/upload',{method:'POST',body:fd});
    const d=await r.json();hideOv();
    if(d.error){toast(d.error,'err');return}
    const uid='up_'+Date.now();
    document.getElementById('upRes').innerHTML = buildFullAnalysis(d, uid);
    initTabs(document.getElementById('upRes'));
    setTimeout(()=>animateBars(document.getElementById('upRes')), 140);
    toast(d.procesado_gpt?`☑ ${d.detection_count} det. · Groq OK · IQ:
${d.indice_calidad}`:`☑ ${d.detection_count} det. · análisis local`,'ok');

setTimeout(()=>document.getElementById('upRes').scrollIntoView({behavior:'sm
ooth',block:'start'}),160);
  }catch(e){hideOv();toast('Error: '+e.message,'err')}
}

async function tCam(){
  const
btn=document.getElementById('camBtn'),cap=document.getElementById('capBtn');
  if(camStream){
    camStream.getTracks().forEach(t=>t.stop());camStream=null;
    document.getElementById('vid').srcObject=null;
    btn.innerHTML='<i class="fas fa-power-off"></i> Encender
Cámara';cap.disabled=true;
  }else{
    try{
      camStream=await
navigator.mediaDevices.getUserMedia({video:{facingMode:'environment'}});
      document.getElementById('vid').srcObject=camStream;
      btn.innerHTML='<i class="fas fa-stop-circle"></i> Apagar
Cámara';cap.disabled=false;
      toast('Cámara activa','ok');
    }catch(e){toast('No se pudo acceder a la cámara: '+e.message,'err')}
  }
}
async function capImg(){
  const
vid=document.getElementById('vid'),c=document.createElement('canvas');

c.width=vid.videoWidth;c.height=vid.videoHeight;c.getContext('2d').drawImage
(vid,0,0);
```

```
  const dep=document.getElementById('depSel').value;
  showOv('Procesando captura','YOLO + Groq LLaMA...');
  try{
    const r=await fetch('/captura',{method:'POST',headers:{'Content-
Type':'application/json'},body:JSON.stringify({imagen:c.toDataURL('image/jpe
g',.92),departamento:dep})});
    const d=await r.json();hideOv();
    if(d.error){toast(d.error,'err');return}
    const uid='cam_'+Date.now();
    document.getElementById('camRes').innerHTML = buildFullAnalysis(d, uid);
    initTabs(document.getElementById('camRes'));
    setTimeout(()=>animateBars(document.getElementById('camRes')), 140);
    toast(`☑ ${d.detection_count} det.`,'ok');
  }catch(e){hideOv();toast('Error: '+e.message,'err')}
}

function buildFullAnalysis(d, uid){
  const dets = d.detecciones || [];
  const lv = {maduro:0, verde:0, podrido:0};
  dets.forEach(x=>{ if(x.nivel in lv) lv[x.nivel]++ });

  const pills = `<div class="pills-row">
    ${lv.maduro  ? `<span class="pill M">☑ ${lv.maduro}
madura${lv.maduro>1?'s':''}</span>` : ''}
    ${lv.verde   ? `<span class="pill V">🍸 ${lv.verde}
verde${lv.verde>1?'s':''}</span>`   : ''}
    ${lv.podrido ? `<span class="pill P">⚠ ${lv.podrido}
podrida${lv.podrido>1?'s':''}</span>`: ''}
  </div>`;

  const imgSrc = d.url || d.thumbnail_url || '';
  const tags = [
    `<span class="itag">${dets.length}
detección${dets.length!==1?'es':''}</span>`,
    d.confidence_average ? `<span class="itag g">🎯
${Math.round(d.confidence_average*100)}% conf.</span>` : '',
    d.imagen_width       ? `<span
class="itag">${d.imagen_width}×${d.imagen_height}px</span>` : '',
    d.region_peru        ? `<span class="itag">📍
${d.region_peru}${d.departamento?' · '+d.departamento:''}</span>` : '',
    d.procesado_gpt      ? `<span class="itag r"><i class="fas fa-bolt"></i>
Groq</span>` : '',
  ].filter(Boolean).join('');

  const imgBlock = `<div class="img-box">
    <img src="${imgSrc}" alt="Imagen analizada"
onclick="openZoom('${imgSrc}')"/>
    <div class="ov-top">
      <div class="itags">${tags}</div>
```

```
      <span class="img-ts">${d.timestamp||''}</span>
    </div>
    <div class="ov-bot">
      <span style="font-size:.63rem;color:rgba(255,255,255,.38)"><i
class="fas fa-search-plus"></i> Clic para ampliar</span>
      <button class="zoom-btn" onclick="openZoom('${imgSrc}')"><i class="fas
fa-expand-arrows-alt"></i> Zoom</button>
    </div>
  </div>`;

  let detItems = '';
  if(!dets.length){
    detItems = `<div class="no-det"><i class="fas fa-exclamation-triangle"
style="flex-shrink:0;margin-top:2px"></i>
      <div><b>Sin detecciones YOLO</b><br>No se encontraron frutas
reconocidas. Prueba con mejor iluminación o acerca más la cámara.</div>
    </div>`;
  } else {
    detItems = `<ul class="det-list">${dets.map(x=>`
      <li class="det-item ${x.nivel}">
        <span class="det-emo">${x.emoji}</span>
        <div class="det-body">
          <div class="det-row1">
            <span class="det-name">${x.clase}</span>
            <span class="det-lv ${x.nivel}">${x.estado}</span>
          </div>
          <div class="det-info">${x.conf_pct} confianza  · 
${x.pct_maduracion}% maduración  ·  área
${Math.round((x.area_rel||0)*100)}%</div>
          <div class="cbar-row">
            <div class="cbar-track"><div class="cbar-fill" style="width:0%"
data-target="${(x.conf*100).toFixed(1)}%"></div></div>
            <span class="cbar-pct">${x.conf_pct}</span>
          </div>
        </div>
      </li>`).join('')}</ul>`;
  }
  const detBlock = `<div class="det-panel">
    <div class="det-hdr">
      <h3><i class="fas fa-crosshairs"></i> Detecciones YOLO <span
class="det-badge-cnt">${dets.length}</span></h3>
      <span class="det-sub">YOLOv8s · conf≥0.20</span>
    </div>
    ${detItems}
  </div>`;

  const iq    = d.indice_calidad || 0;
  const iqCls = iq>=70 ? 'hi' : iq>=40 ? 'med' : 'low';
  const iqCol = iq>=70 ? '#22C55E' : iq>=40 ? '#F59E0B' : '#EF4444';
```

```
  const iqTxt = iq>=70
    ? 'Lote en excelentes condiciones — apto para comercializar y exportar.'
    : iq>=40
    ? 'Condiciones aceptables — revisar selección y actuar en los próximos
días.'
    : 'Problemas críticos detectados — acción urgente requerida, riesgo de
pérdida.';

  const iqBlock = `<div class="iq-block">
    <div class="iq-inner">
      <div class="iq-num" style="color:${iqCol}">${iq}<span style="font-
size:1rem;opacity:.38">/100</span></div>
      <div class="iq-right">
        <div class="iq-label"><i class="fas fa-star"
style="color:${iqCol}"></i> Índice de Calidad del Lote</div>
        <div class="iq-track"><div class="iq-fill iq-${iqCls}"
style="width:0%" data-target="${iq}%"></div></div>
        <div class="iq-desc">${iqTxt}</div>
        <div class="iq-stats">
          ${d.detection_count  ? `<span class="iq-stat"><i class="fas fa-
search" style="color:var(--teal)"></i> <b>${d.detection_count}</b>
detección${d.detection_count!==1?'es':''}</span>` : ''}
          ${d.confidence_average ? `<span class="iq-stat"><i class="fas fa-
bullseye" style="color:var(--amber)"></i>
<b>${Math.round(d.confidence_average*100)}%</b> confianza promedio</span>` :
''}
          ${d.porcentaje_maduracion ? `<span class="iq-stat"><i class="fas
fa-leaf" style="color:var(--mad)"></i> <b>${d.porcentaje_maduracion}</b>
maduración</span>` : ''}
        </div>
      </div>
    </div>
  </div>`;

  const precio = d.precio_estimado_kg || '';
  const exp    = d.calidad_exportacion || '';
  const exportable = /sí|si|yes|export/i.test(exp);
  const groqMk = d.procesado_gpt
    ? `<span class="mk-groq"><i class="fas fa-bolt"></i> Groq LLaMA
70B</span>`
    : `<span class="mk-local"><i class="fas fa-database"></i> Análisis
local</span>`;
  const madMk = d.porcentaje_maduracion
    ? `<span class="mk-mad"><i class="fas fa-leaf"></i>
${d.porcentaje_maduracion}</span>` : '';

  const priceHero = precio ? `<div class="price-hero">
    <div>
      <div class="price-big">${precio}</div>
```

```
      <div class="price-sub">Precio estimado por kg · Mercado Perú
2026</div>
    </div>
    <div class="exp-pill ${exportable?'exp-y':'exp-n'}">
      <i class="fas fa-${exportable?'plane':'store'}"></i>
      ${exportable ? 'Exportable' : 'Mercado local'}
    </div>
  </div>` : '';

  const agroBlock = `<div class="agro-wrap">
    <div class="agro-hdr">
      <div class="agro-hdr-left">
        <h2><i class="fas fa-seedling"></i> Análisis Agronómico
Completo</h2>
      </div>
      <div class="agro-meta">${groqMk}${madMk}</div>
    </div>
    ${priceHero}
    <div class="agro-tabs">
      <span class="tab on"  data-p="${uid}_t1">📋 General</span>
      <span class="tab"     data-p="${uid}_t2">💡 Recomendaciones</span>
      <span class="tab"     data-p="${uid}_t3">🌱 Cultivo</span>
      <span class="tab"     data-p="${uid}_t4">💰 Mercado</span>
      <span class="tab"     data-p="${uid}_t5">🧪 Nutrición</span>
      <span class="tab"     data-p="${uid}_t6">📝 Mis Notas</span>
    </div>

    <div id="${uid}_t1" class="pane on">
      <div class="ag3">
        ${d.descripcion_ia      ? `<div class="abox tl span2"><h4><i
class="fas fa-info-circle"></i> Descripción
General</h4><p>${d.descripcion_ia}</p></div>` : ''}
        ${d.variedad_detectada  ? `<div class="abox tl"><h4><i class="fas
fa-dna"></i> Variedad Detectada</h4><p>${d.variedad_detectada}</p></div>` :
''}
        ${d.vida_util_dias      ? `<div class="abox ta"><h4><i class="fas
fa-hourglass-half"></i> Vida Útil
Estimada</h4><p>${d.vida_util_dias}</p></div>` : ''}
        ${d.temperatura_ideal   ? `<div class="abox tc"><h4><i class="fas
fa-thermometer-full"></i> Temperatura
Ideal</h4><p>${d.temperatura_ideal}</p></div>` : ''}
      </div>
    </div>

    <div id="${uid}_t2" class="pane">
      <div class="ag2">
        ${d.recomendaciones  ? `<div class="abox tl span2"><h4><i class="fas
fa-clipboard-check"></i> Recomendaciones
Inmediatas</h4><p>${d.recomendaciones}</p></div>` : ''}
```

```
        ${d.clima_recomendado? `<div class="abox tl"><h4><i class="fas fa-
cloud-sun"></i> Condiciones
Climáticas</h4><p>${d.clima_recomendado}</p></div>` : ''}
        ${d.almacenamiento   ? `<div class="abox ta"><h4><i class="fas fa-
box"></i>
Almacenamiento</h4><p>${d.almacenamiento}${d.temperatura_ideal?'<br><small
style="color:var(--amber)">📌
'+d.temperatura_ideal+'</small>':''}</p></div>` : ''}
      </div>
    </div>

    <div id="${uid}_t3" class="pane">
      <div class="ag2">
        ${d.consejos_cultivo ? `<div class="abox tg span2"><h4><i class="fas
fa-tractor"></i> Consejos para Agricultores
Peruanos</h4><p>${(d.consejos_cultivo||'').replace(/\n/g,'<br>')}</p></div>`
: ''}
        ${d.tiempo_maduracion? `<div class="abox ta"><h4><i class="fas fa-
clock"></i> Tiempo de Maduración</h4><p>${d.tiempo_maduracion}</p></div>` :
''}
        ${d.plagas_riesgos   ? `<div class="abox tc"><h4><i class="fas fa-
bug"></i> Plagas y Riesgos</h4><p>${d.plagas_riesgos}</p></div>` : ''}
      </div>
    </div>

    <div id="${uid}_t4" class="pane">
      ${priceHero}
      <div class="ag2">
        ${d.mercado_local        ? `<div class="abox tl"><h4><i class="fas
fa-store"></i> Mercado Local · Perú
2026</h4><p>${d.mercado_local}</p></div>` : ''}
        ${d.calidad_exportacion  ? `<div class="abox tp"><h4><i class="fas
fa-plane"></i> Exportación y
Destinos</h4><p>${d.calidad_exportacion}</p></div>` : ''}
      </div>
    </div>

    <div id="${uid}_t5" class="pane">
      <div class="ag2">
        ${d.valor_nutricional  ? `<div class="abox tl span2"><h4><i
class="fas fa-heartbeat"></i> Valor
Nutricional</h4><p>${d.valor_nutricional}</p></div>` : ''}
        ${d.temperatura_ideal  ? `<div class="abox ta"><h4><i class="fas fa-
thermometer-half"></i> Temperatura Ideal de
Almacenamiento</h4><p>${d.temperatura_ideal}</p></div>` : ''}
        ${d.vida_util_dias     ? `<div class="abox tc"><h4><i class="fas fa-
calendar-day"></i> Vida Útil Detallada</h4><p>${d.vida_util_dias}</p></div>`
: ''}
      </div>
```

```
    </div>

    <div id="${uid}_t6" class="pane">
      <div class="notas-zone">
        <label><i class="fas fa-pen"></i> Notas del Agricultor — ID
#${d.id||'nuevo'}</label>
        <textarea id="nt_${uid}" placeholder="Observaciones sobre este lote,
fecha de cosecha, campo, condiciones del
terreno...">${d.notas_usuario||''}</textarea>
        <div class="row" style="margin-top:.65rem">
          <button class="btn sm bp" onclick="saveNotas(${d.id||0},
document.getElementById('nt_${uid}').value)">
            <i class="fas fa-save"></i> Guardar notas
          </button>
        </div>
      </div>
    </div>
  </div>`;

  return `<div class="res-wrap">
    ${pills}
    <div class="blk-top">
      ${imgBlock}
      ${detBlock}
    </div>
    ${iqBlock}
    ${agroBlock}
  </div>`;
}

function initTabs(container){
  if(!container) return;
  container.querySelectorAll('.agro-tabs').forEach(bar=>{
    bar.querySelectorAll('.tab').forEach(tab=>{
      tab.onclick = function(){
        const pid = this.dataset.p;
        const sec = this.closest('.agro-wrap');
        sec.querySelectorAll('.tab').forEach(t=>t.classList.remove('on'));
        sec.querySelectorAll('.pane').forEach(p=>p.classList.remove('on'));
        this.classList.add('on');
        const pane = sec.querySelector('#'+pid);
        if(pane) pane.classList.add('on');
      };
    });
  });
}
function animateBars(container){
  if(!container) return;
```

```
  container.querySelectorAll('.cbar-fill[data-target],.iq-fill[data-
target]').forEach(el=>{
    el.style.width = el.dataset.target;
  });
}

async function saveNotas(id, notas){
  if(!id){toast('ID no disponible','wrn');return}
  try{
    const r = await fetch('/notas/'+id,{method:'POST',headers:{'Content-
Type':'application/json'},body:JSON.stringify({notas})});
    toast(r.ok ? '☑ Notas guardadas' : 'Error al guardar',
r.ok?'ok':'err');
  }catch(e){toast('Error de red','err')}
}

async function loadLib(page=1){
  libPage = page;
  try{
    const d = await
fetch(`/biblioteca?page=${page}&per_page=20`).then(r=>r.json());
    libData = d.items || [];
    libData.forEach(x=>{ libCache[x.id] = x; });
    renderLib(libData, d.total, d.page, d.per_page);
  }catch(e){
    document.getElementById('lgrid').innerHTML='<div class="empty"><i
class="fas fa-exclamation-triangle"></i><p>Error al cargar</p></div>';
  }
}

function flLib(){
  const q = (document.getElementById('libQ').value||'').toLowerCase();
  const f = document.getElementById('libF').value;
  const sort = document.getElementById('libSort').value;
  let items = [...libData];
  if(q)         items =
items.filter(x=>JSON.stringify(x).toLowerCase().includes(q));
  if(f==='det') items = items.filter(x=>x.has_detection);
  if(f==='no')  items = items.filter(x=>!x.has_detection);
  if(f==='mad') items =
items.filter(x=>(x.detecciones||[]).some(d=>d.nivel==='maduro'));
  if(f==='ver') items =
items.filter(x=>(x.detecciones||[]).some(d=>d.nivel==='verde'));
  if(f==='pod') items =
items.filter(x=>(x.detecciones||[]).some(d=>d.nivel==='podrido'));
  if(f==='groq')items = items.filter(x=>x.procesado_gpt);
  if(sort==='iq')   items.sort((a,b)=>(b.indice_calidad||0)-
(a.indice_calidad||0));
```

```
  if(sort==='conf') items.sort((a,b)=>(b.confidence_average||0)-
(a.confidence_average||0));
  renderLib(items, items.length, 1, items.length);
}
document.addEventListener('DOMContentLoaded',()=>{
  document.getElementById('libQ').addEventListener('input', flLib);
});

function renderLib(items, total, page, per_page){
  const grid = document.getElementById('lgrid');
  const pag  = document.getElementById('lpag');
  if(!items.length){
    grid.innerHTML='<div class="empty"><i class="fas fa-search"></i><p>Sin
resultados</p></div>';
    pag.innerHTML=''; return;
  }
  grid.innerHTML = items.map(x=>{
    const dets = x.detecciones||[];
    const nivs = [...new Set(dets.map(d=>d.nivel))];
    const iq   = x.indice_calidad||0;
    const iqC  = iq>=70?'#22C55E':iq>=40?'#F59E0B':'#EF4444';
    const frutas = [...new Set(dets.map(d=>d.emoji+' '+d.clase))].join(',
')||'Sin frutas';
    const tags = [
      `<span class="tag
${x.has_detection?'td':'tnd'}">${x.has_detection?'☑ Det.':'✕'}</span>`,
      x.region_peru ? `<span class="tag tr">${x.region_peru}</span>` : '',
      nivs.includes('maduro')  ? '<span class="tag tm">Maduro</span>'  : '',
      nivs.includes('verde')   ? '<span class="tag tv">Verde</span>'   : '',
      nivs.includes('podrido') ? '<span class="tag tp2">Podrido</span>': '',
      x.procesado_gpt ? '<span class="tag tg2">⚡ Groq</span>' : '',
    ].filter(Boolean).join('');

    return `<div class="lcard">
      <div class="lc-img" onclick="openDetailById(${x.id})">
        <img src="${x.thumbnail_url||x.url}" alt="thumb" loading="lazy"/>
        ${iq>0 ? `<span class="lc-iq" style="color:${iqC}">★ ${iq}</span>`
: ''}
      </div>
      <div class="lc-body">
        <div class="lc-title">${frutas.slice(0,50)}</div>
        <div class="lc-meta">
          <span>📅 ${x.timestamp}</span>
          <span>📍 ${x.departamento||x.region_peru||'—'}</span>
          ${x.precio_estimado_kg ? `<span>$
${x.precio_estimado_kg}</span>` : ''}
        </div>
        <div class="lc-meta">${tags}</div>
        <div class="lc-acts">
```

```
          <button class="btn sm bp" onclick="openDetailById(${x.id})">
            <i class="fas fa-eye"></i> Ver Detalle
          </button>
          <button class="btn sm bd"
onclick="event.stopPropagation();delEntry(${x.id},this)">
            <i class="fas fa-trash"></i>
          </button>
        </div>
      </div>
    </div>`;
  }).join('');

  const pages = Math.ceil(total/per_page);
  pag.innerHTML = pages>1
    ? Array.from({length:Math.min(pages,8)},(_,i)=>i+1)
        .map(i=>`<button class="btn sm ${i===page?'bp':'bo'}"
onclick="loadLib(${i})">${i}</button>`)
        .join('')
    : '';
}

async function openDetailById(id){
  let data = libCache[id];

  if(data && !data.descripcion_ia && !data.recomendaciones){
    try{
      const r = await fetch('/biblioteca/'+id);
      if(r.ok){
        const fresh = await r.json();
        if(fresh && !fresh.error){ data = fresh; libCache[id] = fresh; }
      }
    }catch(e){ /* usa datos del cache */ }
  }

  if(data){
    openDetail(data);
  } else {
    toast('Registro no encontrado','wrn');
  }
}

async function delEntry(id, btn){
  if(!confirm('¿Eliminar este registro?')) return;
  btn.disabled = true;
  try{
    const r = await fetch('/delete/'+id, {method:'DELETE'});
    const d = await r.json();
    if(d.message){ toast('Eliminado','ok'); delete libCache[id];
loadLib(libPage); }
```

```
    else{ toast(d.error||'Error','err'); btn.disabled=false; }
  }catch(e){ toast('Error de red','err'); btn.disabled=false; }
}

async function clrAll(){
  if(!confirm('¿Eliminar TODOS los registros?')) return;
  showOv('Reseteando sistema...');
  try{
    const d = await fetch('/reset',{method:'POST'}).then(r=>r.json());
    hideOv(); libCache={}; toast(d.message||d.error,'ok'); loadLib();
  }catch(e){ hideOv(); toast('Error','err'); }
}

async function loadSts(){
  try{
    const d = await fetch('/estadisticas').then(r=>r.json());
    const $ = id=>document.getElementById(id);
    $('kT').textContent=d.total; $('kD').textContent=d.detectadas;
    $('kTR').textContent=(d.tasa_deteccion||0)+'%';
    $('kM').textContent=d.nivel_maduracion?.maduro||0;
    $('kV').textContent=d.nivel_maduracion?.verde||0;
    $('kP').textContent=d.nivel_maduracion?.podrido||0;
    $('kE').textContent=Object.keys(d.clases||{}).length;
    $('kIQ').textContent=(d.indice_calidad_prom||0).toFixed(1);
    $('kGPT').textContent=d.procesadas_con_gpt||0;
    const tR=Object.entries(d.regiones||{}).sort((a,b)=>b[1]-a[1])[0];
    const tF=(d.top_frutas||[])[0];
    const tC=Object.entries(d.avg_conf||{}).sort((a,b)=>b[1]-a[1])[0];
    const tD=Object.entries(d.departamentos||{}).sort((a,b)=>b[1]-a[1])[0];
    $('sReg').textContent   = tR ? tR[0]+' ('+tR[1]+')' : '—';
    $('sFruta').textContent = tF ? tF[0]+' ('+tF[1]+')' : '—';
    $('sConf').textContent  = tC ? tC[0]+' '+Math.round(tC[1]*100)+'%' : '—
';
    $('sDepto').textContent = tD ? tD[0]+' ('+tD[1]+')' : '—';
    buildCharts(d);
  }catch(e){ toast('Error cargando estadísticas','err'); }
}
const
PAL=['#00D4AA','#05F5D0','#BCFF00','#FFB300','#FF4D4D','#A78BFA','#60A5FA','
#FB923C','#34D399'];
const
copts={plugins:{legend:{labels:{color:'rgba(228,242,238,.58)',font:{size:10,
family:'Outfit'}}}}};
function
axOpts(){return{color:'rgba(228,242,238,.32)',grid:{color:'rgba(255,255,255,
.035)'}}}
function mkChart(id,cfg){if(charts[id])charts[id].destroy();const
c=document.getElementById(id);if(!c)return;charts[id]=new Chart(c,cfg)}
function buildCharts(d){
```

```
  const regs=d.regiones||{};

mkChart('rC',{type:'doughnut',data:{labels:Object.keys(regs),datasets:[{data
:Object.values(regs),backgroundColor:PAL,borderWidth:2,borderColor:'rgba(0,0
,0,.5)'}]},options:{...copts,cutout:'62%'}});
  const top=d.top_frutas||[];

mkChart('fC',{type:'bar',data:{labels:top.map(x=>x[0]),datasets:[{data:top.m
ap(x=>x[1]),backgroundColor:PAL,borderRadius:4}]},options:{...copts,indexAxi
s:'y',scales:{x:{...axOpts()},y:{...axOpts()}}}});
  const act=d.actividad_semana||{};

mkChart('aC',{type:'line',data:{labels:Object.keys(act),datasets:[{data:Obje
ct.values(act),borderColor:'#00D4AA',backgroundColor:'rgba(0,212,170,.08)',f
ill:true,tension:.4,pointBackgroundColor:'#00D4AA',pointRadius:4}]},options:
{...copts,scales:{x:{...axOpts()},y:{...axOpts()}}}});
  const niv=d.nivel_maduracion||{};

mkChart('mC',{type:'doughnut',data:{labels:['Maduro','Verde','Podrido'],data
sets:[{data:[niv.maduro||0,niv.verde||0,niv.podrido||0],backgroundColor:['#2
2c55e','#f59e0b','#ef4444'],borderWidth:2,borderColor:'rgba(0,0,0,.5)'}]},op
tions:{...copts,cutout:'58%'}});
  const deps=d.departamentos||{};

mkChart('dC',{type:'bar',data:{labels:Object.keys(deps),datasets:[{data:Obje
ct.values(deps),backgroundColor:PAL,borderRadius:4}]},options:{...copts,scal
es:{x:{...axOpts()},y:{...axOpts()}}}});
  const cTop=Object.entries(d.avg_conf||{}).sort((a,b)=>b[1]-
a[1]).slice(0,8);

mkChart('cC',{type:'bar',data:{labels:cTop.map(x=>x[0]),datasets:[{data:cTop
.map(x=>Math.round(x[1]*100)),backgroundColor:'rgba(0,212,170,.55)',borderRa
dius:4,label:'Conf
%'}]},options:{...copts,scales:{x:{...axOpts()},y:{...axOpts(),min:0,max:100
}}}});
}
async function rstSys(){
  if(!confirm('¿Resetear completamente el sistema?')) return;
  showOv('Reseteando...');
  const d=await fetch('/reset',{method:'POST'}).then(r=>r.json());
  hideOv(); toast(d.message||d.error,'ok'); loadSts();
}

async function loadLogs(){
  const lb=document.getElementById('lb');
  lb.innerHTML='<em style="opacity:.25">Cargando...</em>';
  try{
    const d=await fetch('/logs').then(r=>r.json());
```

```
    if(!d.logs||!d.logs.length){lb.innerHTML='<em style="opacity:.25">Sin
errores registrados.</em>';return}

lb.innerHTML=d.logs.map(l=>`<div>${l.replace(/&/g,'&amp;').replace(/</g,'&lt
;')}</div>`).join('');
    lb.scrollTop=lb.scrollHeight;
  }catch(e){lb.innerHTML='<em style="opacity:.25">Error al cargar</em>'}
}
async function testGroq(){
  toast('Probando Groq desde el servidor...','inf');
  try{
    const d=await fetch('/test-groq').then(r=>r.json());
    toast(d.status==='ok'?'☑ Groq OK — '+d.modelo:'❌ Error:
'+(d.mensaje||'sin respuesta'),d.status==='ok'?'ok':'err');
  }catch(e){toast('Error de red','err')}
}
</script>
</body>
</html>
```

config.py

```
import os
import logging
from logging.handlers import RotatingFileHandler

BASE_DIR         = os.path.dirname(os.path.abspath(__file__))
TEMPLATES_DIR    = os.path.join(BASE_DIR, 'templates')
STATIC_DIR       = os.path.join(BASE_DIR, 'static')
UPLOAD_FOLDER    = os.path.join(BASE_DIR, 'uploads')
PROCESSED_FOLDER = os.path.join(BASE_DIR, 'processed')
THUMBNAIL_FOLDER = os.path.join(BASE_DIR, 'thumbnails')
DB_PATH          = os.path.join(BASE_DIR, 'fruit_classifier.db')
LOG_DIR          = os.path.join(BASE_DIR, 'logs')
MODEL_PATH       = os.path.join(BASE_DIR, 'modelo',
'best_yolov8s_fruits_v6.pt')

for _carpeta in [TEMPLATES_DIR, STATIC_DIR, UPLOAD_FOLDER,
                 PROCESSED_FOLDER, THUMBNAIL_FOLDER, LOG_DIR,
                 os.path.join(BASE_DIR, 'modelo')]:
    os.makedirs(_carpeta, exist_ok=True)

ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif', 'bmp', 'webp'}
MODEL_IMGSZ        = 640
MODEL_CONF         = 0.20
MODEL_IOU          = 0.45
MAX_UPLOAD_MB      = 16

GROQ_API_KEY    = "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"
```

```
GROQ_API_URL     = "https://api.groq.com/openai/v1/chat/completions"
GROQ_MODEL       = "llama-3.3-70b-versatile"   # modelo gratuito y potente
GROQ_TIMEOUT     = 45
GROQ_MAX_TOKENS = 2000

def setup_logging() -> logging.Logger:
    log_file  = os.path.join(LOG_DIR, 'error.log')
    formatter = logging.Formatter(
        '%(asctime)s | %(levelname)-8s | %(module)s:%(lineno)d |
%(message)s',
        datefmt='%Y-%m-%d %H:%M:%S'
    )
    fh = RotatingFileHandler(
        log_file, maxBytes=5*1024*1024, backupCount=5, encoding='utf-8'
    )
    fh.setLevel(logging.WARNING)
    fh.setFormatter(formatter)
    ch = logging.StreamHandler()
    ch.setLevel(logging.INFO)
    ch.setFormatter(logging.Formatter('%(levelname)s: %(message)s'))
    log = logging.getLogger('fruitapp')
    if not log.handlers:
        log.setLevel(logging.DEBUG)
        log.addHandler(fh)
        log.addHandler(ch)
    return log

logger = setup_logging()

CLASES_ES: dict[str, tuple] = {
    'apple_ripe':          ('Manzana',  'Madura',         '🍎',
'maduro',  90),
    'apple_unripe':        ('Manzana',  'Verde/Inmadura', '🍎',
'verde',   20),
    'apple_rotten':        ('Manzana',  'Podrida',        '🍎',
'podrido',  5),
    'mango_ripe':          ('Mango',    'Maduro',         '🥭',
'maduro',  90),
    'mango_unripe':        ('Mango',    'Verde/Inmaduro', '🥭',
'verde',   20),
    'mango_rotten':        ('Mango',    'Podrido',        '🥭',
'podrido',  5),
    'pineapple_ripe':      ('Piña',     'Madura',         '🍍',
'maduro',  90),
    'pineapple_unripe':    ('Piña',     'Verde/Inmadura', '🍍',
'verde',   20),
    'pineapple_rotten':    ('Piña',     'Podrida',        '🍍',
'podrido',  5),
```

```
    'strawberry_ripe':     ('Fresa',    'Madura',         '🍓',
'maduro',  90),
    'strawberry_unripe':   ('Fresa',    'Verde/Inmadura', '🍓',
'verde',   20),
    'strawberry_rotten':   ('Fresa',    'Podrida',        '🍓',
'podrido',  5),
    'banana_ripe':         ('Plátano',  'Maduro',         '🍌',
'maduro',  90),
    'banana_unripe':       ('Plátano',  'Verde/Inmaduro', '🍌',
'verde',   20),
    'banana_rotten':       ('Plátano',  'Podrido',        '🍌',
'podrido',  5),
    'avocado_ripe':        ('Palta',    'Madura',         '🥑',
'maduro',  90),
    'avocado_unripe':      ('Palta',    'Verde/Inmadura', '🥑',
'verde',   20),
    'avocado_rotten':      ('Palta',    'Podrida',        '🥑',
'podrido',  5),
    'papaya_ripe':         ('Papaya',   'Madura',         '🥭',
'maduro',  90),
    'papaya_unripe':       ('Papaya',   'Verde/Inmadura', '🥭',
'verde',   20),
    'papaya_rotten':       ('Papaya',   'Podrida',        '🥭',
'podrido',  5),
    'grape_ripe':          ('Uva',      'Madura',         '🍇',
'maduro',  90),
    'grape_unripe':        ('Uva',      'Verde/Inmadura', '🍇',
'verde',   20),
    'grape_rotten':        ('Uva',      'Podrida',        '🍇',
'podrido',  5),
    'orange_ripe':         ('Naranja',  'Madura',         '🍊',
'maduro',  90),
    'orange_unripe':       ('Naranja',  'Verde/Inmadura', '🍊',
'verde',   20),
    'orange_rotten':       ('Naranja',  'Podrida',        '🍊',
'podrido',  5),
    'no_fruit':            ('Sin fruta','N/A',            '❌',
'none',     0),
}

IGNORAR_CLASES: set[str] = {'no_fruit'}

COLORES_NIVEL: dict[str, tuple] = {
    'maduro':      ( 34, 197,  94),
    'verde':       (234, 179,   8),
    'podrido':     (239,  68,  68),
    'desconocido': ( 99, 102, 241),
    'none':        (156, 163, 175),
}
```

```
REGIONES_PERU: dict[str, dict] = {
    'Costa': {
        'frutas': [
            'limon','mandarina','naranja','palta','platano','mango',
            'uva','maracuya','granadilla','fresa','sandia','melon'
        ],
        'clima': 'Cálido y seco',
        'departamentos': [
            'Lima','La Libertad','Lambayeque','Piura','Ica',
            'Arequipa','Moquegua','Tacna'
        ]
    },
    'Sierra': {
        'frutas': [
            'manzana','pera','durazno','tuna','granadilla','aguaymanto',
            'chirimoya','membrillo','capuli','lucuma','pacae'
        ],
        'clima': 'Templado y frío',
        'departamentos': [
            'Cusco','Puno','Junin','Huanuco','Ancash',
            'Cajamarca','Ayacucho','Apurimac'
        ]
    },
    'Selva': {
        'frutas': [
            'pina','papaya','maracuya','coco','guayaba','camu camu',
            'aguaje','platano','pinon','cacao','aji','castana'
        ],
        'clima': 'Cálido y húmedo',
        'departamentos': [
            'Loreto','Ucayali','San Martin','Madre de Dios','Amazonas'
        ]
    }
}
```

funciones.py

```
import os
import sys
import cv2
import numpy as np
import json
import requests
from datetime import datetime
from ultralytics import YOLO

from config import (
    logger,
```

```
    BASE_DIR, UPLOAD_FOLDER, PROCESSED_FOLDER, THUMBNAIL_FOLDER,
    MODEL_PATH, MODEL_IMGSZ, MODEL_CONF, MODEL_IOU,
    GROQ_API_KEY, GROQ_API_URL, GROQ_MODEL, GROQ_TIMEOUT, GROQ_MAX_TOKENS,
    CLASES_ES, IGNORAR_CLASES, COLORES_NIVEL,
    REGIONES_PERU,
)


def cargar_modelo() -> tuple[YOLO, str]:
    sep = "=" * 65
    print(f"\n{sep}")
    print("  CLASIFICADOR DE FRUTAS PERUANAS v5.2")
    print(sep)
    print(f"  BASE_DIR    : {BASE_DIR}")
    print(f"  Ruta modelo : {MODEL_PATH}")

    existe = os.path.exists(MODEL_PATH)
    print(f"  ¿Existe?    : {'SÍ ✔' if existe else 'NO ✗'}")

    if existe:
        mb    = os.path.getsize(MODEL_PATH) / (1024 * 1024)
        print(f"  Tamaño      : {mb:.1f} MB")
        ruta = MODEL_PATH
    else:
        print(f"  Usando yolov8n.pt (modelo genérico de respaldo)")
        ruta = 'yolov8n.pt'

    try:
        m      = YOLO(ruta)
        clases = list(m.names.values()) if hasattr(m, 'names') else []
        print(f"  Clases ({len(clases)}): {', '.join(clases)}")
        print(f"  imgsz: {MODEL_IMGSZ}  |  conf: {MODEL_CONF}  |  iou:
{MODEL_IOU}")
        print(f"  IA    : Groq {GROQ_MODEL} (llamado desde el servidor)")
        print(sep + "\n")
        logger.info(f"YOLO cargado: {ruta} | {len(clases)} clases | IA: Groq
{GROQ_MODEL}")
        return m, ruta
    except Exception as e:
        print(f"  ERROR CRÍTICO al cargar modelo: {e}")
        logger.critical(f"Error cargando YOLO: {e}", exc_info=True)
        sys.exit(1)


model, MODEL_USADO = cargar_modelo()


def traducir_clase(clase_raw: str) -> dict:
    info = CLASES_ES.get(clase_raw)
```

```
    if info:
        nombre, estado, emoji, nivel, pct = info
        return {'clase_raw': clase_raw, 'nombre': nombre, 'estado': estado,
                'emoji': emoji, 'nivel': nivel, 'pct_maduracion': pct}
    return {'clase_raw': clase_raw, 'nombre': clase_raw.replace('_',' 
').title(),
            'estado': 'Desconocido', 'emoji': '🍎', 'nivel': 'desconocido',
'pct_maduracion': 50}


def preprocesar_imagen(img: np.ndarray) -> np.ndarray:
    lab      = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
    l, a, b  = cv2.split(lab)
    clahe    = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    img_eq   = cv2.cvtColor(cv2.merge((clahe.apply(l), a, b)),
cv2.COLOR_LAB2BGR)
    hsv      = cv2.cvtColor(img_eq, cv2.COLOR_BGR2HSV).astype(np.float32)
    hsv[:,:,1] = np.clip(hsv[:,:,1] * 1.15, 0, 255)
    return cv2.cvtColor(hsv.astype(np.uint8), cv2.COLOR_HSV2BGR)


def inferir_yolo(img_original: np.ndarray) -> tuple[np.ndarray, list]:
    h_orig, w_orig = img_original.shape[:2]
    img_640  = cv2.resize(preprocesar_imagen(img_original), (MODEL_IMGSZ,
MODEL_IMGSZ))
    results  = model(img_640, conf=MODEL_CONF, iou=MODEL_IOU, verbose=False,
imgsz=MODEL_IMGSZ)

    boxes_raw = results[0].boxes
    n_boxes   = len(boxes_raw) if boxes_raw is not None else 0
    print(f"\n[YOLO] conf>={MODEL_CONF} | boxes: {n_boxes}")

    detecciones = []
    annotated   = img_original.copy()

    if boxes_raw is not None and n_boxes > 0:
        for box in boxes_raw:
            cls_id   = int(box.cls[0].item())
            conf_val = float(box.conf[0].item())
            raw_name = model.names[cls_id]
            if raw_name in IGNORAR_CLASES:
                continue

            info  = traducir_clase(raw_name)
            color = COLORES_NIVEL.get(info['nivel'],
COLORES_NIVEL['desconocido'])
            bgr   = (color[2], color[1], color[0])
```

```
            print(f"  ✔ {raw_name} → {info['nombre']}
{info['estado']}  conf={conf_val:.3f}")

            x1, y1, x2, y2 = box.xyxy[0].tolist()
            x1 = int(x1 * w_orig / MODEL_IMGSZ); y1 = int(y1 * h_orig /
MODEL_IMGSZ)
            x2 = int(x2 * w_orig / MODEL_IMGSZ); y2 = int(y2 * h_orig /
MODEL_IMGSZ)

            th = max(2, int(min(w_orig, h_orig) / 200))
            cv2.rectangle(annotated, (x1, y1), (x2, y2), bgr, th)

            # Barra de confianza
            bw = x2 - x1; bh = max(5, int(min(w_orig, h_orig) / 120))
            cv2.rectangle(annotated, (x1, y2+2), (x2, y2+2+bh), (50,50,50),
-1)
            cv2.rectangle(annotated, (x1, y2+2), (x1+int(bw*conf_val),
y2+2+bh), bgr, -1)

            # Etiqueta
            label = f"{info['nombre']} {info['estado']} {conf_val*100:.1f}%"
            fs = max(0.45, min(w_orig, h_orig) / 1000)
            ft = max(1, th - 1)
            (tw, tth), _ = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX,
fs, ft)
            pad = 5; ty = max(y1 - tth - pad*2, 0)
            cv2.rectangle(annotated, (x1, ty), (x1+tw+pad*2, ty+tth+pad*2),
bgr, -1)
            cv2.putText(annotated, label, (x1+pad, ty+tth+pad),
                        cv2.FONT_HERSHEY_SIMPLEX, fs, (255,255,255), ft,
cv2.LINE_AA)

            detecciones.append({
                "clase": info['nombre'], "clase_raw": raw_name, "estado":
info['estado'],
                "emoji": info['emoji'], "nivel": info['nivel'],
                "conf": round(conf_val, 4), "conf_pct":
f"{conf_val*100:.1f}%",
                "pct_maduracion": info['pct_maduracion'], "bbox": [x1, y1,
x2, y2],
                "area_rel": round((x2-x1)*(y2-y1)/(w_orig*h_orig), 4),
            })

    print(f"[YOLO] Detecciones válidas: {len(detecciones)}\n")
    return annotated, detecciones

def determinar_region(nombres_es: list, departamento: str = '') -> str:
    if departamento:
        for region, info in REGIONES_PERU.items():
```

```
            if departamento in info['departamentos']:
                return region
    conteo = {r: 0 for r in REGIONES_PERU}
    for nombre in nombres_es:
        nl = nombre.lower()
        for region, info in REGIONES_PERU.items():
            for fr in info['frutas']:
                if fr in nl or nl in fr:
                    conteo[region] += 1
    return max(conteo, key=conteo.get) if any(conteo.values()) else 'Costa'

def consultar_groq(prompt: str) -> dict | None:
    """
    Llama a Groq LLaMA desde el servidor Flask.
    Retorna dict con el análisis o None si falla.
    """
    try:
        response = requests.post(
            GROQ_API_URL,
            headers={
                "Authorization": f"Bearer {GROQ_API_KEY}",
                "Content-Type":  "application/json",
            },
            json={
                "model":    GROQ_MODEL,
                "messages": [
                    {
                        "role":    "system",
                        "content": (
                            "Eres un experto agrónomo peruano con 20 años de
experiencia en fruticultura. "
                            "Conoces precios de mercado actuales,
exportación SENASA y plagas comunes. "
                            "Responde SOLO con JSON válido, sin markdown ni
texto adicional."
                        )
                    },
                    {"role": "user", "content": prompt}
                ],
                "max_tokens":  GROQ_MAX_TOKENS,
                "temperature": 0.7,
            },
            timeout=GROQ_TIMEOUT
        )

        if response.status_code == 200:
            content = response.json()['choices'][0]['message']['content']
            logger.info(f"Groq OK: {len(content)} chars")
            # Limpiar posible markdown
```

```
            content = content.strip()
            if content.startswith("```"):
                content = content.split("```")[1]
                if content.startswith("json"):
                    content = content[4:]
            return json.loads(content.strip())

        logger.warning(f"Groq HTTP {response.status_code}:
{response.text[:200]}")
        return None

    except requests.exceptions.Timeout:
        logger.warning("Timeout en Groq API")
        return None
    except json.JSONDecodeError as e:
        logger.warning(f"JSON inválido de Groq: {e}")
        return None
    except Exception as e:
        logger.error(f"Error en Groq: {e}", exc_info=True)
        return None

def generar_analisis(detecciones: list, region: str, departamento: str = '')
-> dict:
    """
    Genera análisis usando Groq. Si falla, usa fallback local.
    """
    if not detecciones:
        return _fallback_analisis(region)

    niveles = [d['nivel'] for d in detecciones]
    n_pod = niveles.count('podrido')
    n_mad = niveles.count('maduro')
    n_ver = niveles.count('verde')

    if n_pod > 0:
        estado_gral, pct, iq = "Frutas podridas detectadas — separar
urgente", "10-30%", 25.0
    elif n_mad >= n_ver:
        estado_gral, pct, iq = "Mayormente maduras — cosechar y vender
pronto", "75-95%", 80.0
    else:
        estado_gral, pct, iq = "Mayormente verdes — requieren maduración",
"25-50%", 55.0

    resumen       = '; '.join(f"{d['clase']} ({d['estado']},
{d['conf']:.0%})" for d in detecciones)
    frutas_unicas = list({d['clase'] for d in detecciones})

    prompt = (
```

```
        f"Analiza estas frutas peruanas detectadas por IA (YOLO):\n"
        f"DETECCIONES: {resumen}\n"
        f"REGIÓN: {region} | DEPARTAMENTO: {departamento or 'No
especificado'}\n"
        f"ESTADO GENERAL: {estado_gral}\n"
        f"FRUTAS: {', '.join(frutas_unicas)}\n\n"
        "Responde SOLO con JSON con estas claves exactas (sin markdown):\n"
        '{"descripcion":"contexto peruano 2026 y
variedades","recomendaciones":"qué hacer ahora",'
        '"clima_recomendado":"condiciones ideales en la región",'
        '"consejos_cultivo":"4 tips numerados para agricultores peruanos",'
        '"tiempo_maduracion":"días estimados o listo para cosechar",'
        '"almacenamiento":"temperatura método y duración",'
        '"mercado_local":"precio y demanda en Perú 2026",'
        '"precio_estimado_kg":"rango S/. por kg ej S/. 2.50 - 4.00",'
        '"calidad_exportacion":"si es exportable y destinos principales",'
        '"valor_nutricional":"calorias vitaminas y minerales clave",'
        '"plagas_riesgos":"2-3 plagas comunes y prevención",'
        '"temperatura_ideal":"grados C óptimos de almacenamiento",'
        '"vida_util_dias":"días de vida útil en condición actual",'
        '"variedad_detectada":"variedad más probable según región",'
        '"indice_calidad":75}'
    )

    datos = consultar_groq(prompt)

    if datos:
        try:
            datos['indice_calidad'] = float(datos.get('indice_calidad', iq))
        except (TypeError, ValueError):
            datos['indice_calidad'] = iq
        datos['porcentaje_maduracion'] = pct
        datos['estado_maduracion']     = estado_gral
        datos['procesado_groq']        = True
        fb = _fallback_analisis(region, ', '.join(frutas_unicas))
        for k, v in fb.items():
            if k not in datos or not datos[k]:
                datos[k] = v
        return datos

    fb = _fallback_analisis(region, ', '.join(frutas_unicas))
    fb['porcentaje_maduracion'] = pct
    fb['estado_maduracion']     = estado_gral
    fb['indice_calidad']        = iq
    fb['procesado_groq']        = False
    return fb


def _fallback_analisis(region: str = 'Perú', frutas: str = '') -> dict:
```

```
    clima = REGIONES_PERU.get(region, {}).get('clima', 'variado')
    return {
        'descripcion':           f'{frutas} en región {region}, Perú.',
        'recomendaciones':       'Madurar a temperatura ambiente. Revisar
diariamente.',
        'clima_recomendado':     f'{clima}. Temperatura óptima 18-25°C,
humedad 60-80%.',
        'consejos_cultivo':      '1. Abono orgánico.\n2. Riego por
goteo.\n3. Control de plagas.\n4. Poda sanitaria.',
        'tiempo_maduracion':     '7-15 días según temperatura ambiente.',
        'almacenamiento':        'Cajas ventiladas a 18-22°C. Vida útil 7-10
días.',
        'mercado_local':         'Mercados mayoristas locales y ferias
agroecológicas.',
        'precio_estimado_kg':    'S/. 2.00 - 5.00 por kg',
        'calidad_exportacion':   'Verificar estándares SENASA.',
        'valor_nutricional':     'Rico en vitamina C, fibra y
antioxidantes.',
        'plagas_riesgos':        'Mosca de la fruta, hongos por humedad
excesiva.',
        'temperatura_ideal':     '10-15°C para almacenamiento prolongado.',
        'vida_util_dias':        '5-10 días a temperatura ambiente.',
        'variedad_detectada':    'Variedad local.',
        'porcentaje_maduracion': '50-70%',
        'estado_maduracion':     'Semi-maduro',
        'indice_calidad':        55.0,
        'procesado_groq':        False,
    }

def procesar_imagen(img: np.ndarray, departamento: str,
                    timestamp: str, prefix: str) -> dict:
    """
    Pipeline completo:
      1. YOLO con preprocesamiento
      2. Región peruana
      3. Análisis Groq (desde el servidor)
      4. Guardar archivos
      5. Retornar dict completo
    """
    h_orig, w_orig = img.shape[:2]

    annotated, detecciones = inferir_yolo(img)

    has_detection = len(detecciones) > 0
    confianzas    = [d['conf'] for d in detecciones]
    nombres_es    = [d['clase'] for d in detecciones]
    conf_avg      = round(sum(confianzas) / len(confianzas), 4) if
confianzas else 0.0
```

```
    region   = determinar_region(nombres_es, departamento)
    analisis = generar_analisis(detecciones, region, departamento)

    if detecciones:
        pct_avg        = round(sum(d['pct_maduracion'] for d in detecciones)
/ len(detecciones))
        pct_maduracion = f"{pct_avg}%"
    else:
        pct_maduracion = analisis.get('porcentaje_maduracion', '50%')

    orig_path   =
os.path.join(UPLOAD_FOLDER,    f"{prefix}_{timestamp}_orig.jpg")
    proc_fname  = f"{prefix}_{timestamp}_proc.jpg"
    proc_path   = os.path.join(PROCESSED_FOLDER, proc_fname)
    thumb_fname = f"{prefix}_{timestamp}_thumb.jpg"
    thumb_path  = os.path.join(THUMBNAIL_FOLDER, thumb_fname)

    cv2.imwrite(orig_path,  img)
    cv2.imwrite(proc_path,  annotated, [cv2.IMWRITE_JPEG_QUALITY, 92])
    cv2.imwrite(thumb_path, cv2.resize(img, (220, 220)))

    return {
        "timestamp":             datetime.now().strftime("%d/%m/%Y %H:%M"),
        "detecciones":           detecciones,
        "url":                   f"/processed/{proc_fname}",
        "thumbnail_url":         f"/thumbnails/{thumb_fname}",
        "has_detection":         has_detection,
        "original_filename":     f"{prefix}_{timestamp}",
        "source":                prefix,
        "confidence_average":    conf_avg,
        "detection_count":       len(detecciones),
        "region_peru":           region,
        "departamento":          departamento,
        "descripcion_ia":        analisis.get('descripcion',          ''),
        "recomendaciones":       analisis.get('recomendaciones',      ''),
        "porcentaje_maduracion": pct_maduracion,
        "clima_recomendado":     analisis.get('clima_recomendado',    ''),
        "consejos_cultivo":      analisis.get('consejos_cultivo',     ''),
        "tiempo_maduracion":     analisis.get('tiempo_maduracion',    ''),
        "almacenamiento":        analisis.get('almacenamiento',       ''),
        "mercado_local":         analisis.get('mercado_local',        ''),
        "precio_estimado_kg":    analisis.get('precio_estimado_kg',   ''),
        "calidad_exportacion":   analisis.get('calidad_exportacion',  ''),
        "valor_nutricional":     analisis.get('valor_nutricional',    ''),
        "plagas_riesgos":        analisis.get('plagas_riesgos',       ''),
        "indice_calidad":        float(analisis.get('indice_calidad',  55)),
        "temperatura_ideal":     analisis.get('temperatura_ideal',    ''),
        "vida_util_dias":        analisis.get('vida_util_dias',       ''),
        "variedad_detectada":    analisis.get('variedad_detectada',   ''),
```

```
        "procesado_gpt":        int(analisis.get('procesado_groq', False)),
        "modelo_version":       'v5.2',
        "imagen_width":         w_orig,
        "imagen_height":        h_orig,
    }
```

database.py

```
import sqlite3
import json
from datetime import datetime
from collections import defaultdict

from config import DB_PATH, logger


def init_db() -> None:
    conn   = sqlite3.connect(DB_PATH)
    cursor = conn.cursor()

    cursor.execute('''
        CREATE TABLE IF NOT EXISTS biblioteca (
            id                    INTEGER PRIMARY KEY AUTOINCREMENT,
            timestamp             TEXT,
            detecciones           TEXT,
            url                   TEXT,
            thumbnail_url         TEXT,
            has_detection         BOOLEAN,
            original_filename     TEXT,
            source                TEXT,
            confidence_average    REAL,
            detection_count       INTEGER,
            region_peru           TEXT,
            departamento          TEXT,
            descripcion_ia        TEXT,
            recomendaciones       TEXT,
            porcentaje_maduracion TEXT,
            clima_recomendado     TEXT,
            consejos_cultivo      TEXT,
            tiempo_maduracion     TEXT,
            almacenamiento        TEXT,
            mercado_local         TEXT,
            -- ── NUEVOS CAMPOS v5.0 ──────────────────────────
            precio_estimado_kg    TEXT    DEFAULT '',
            calidad_exportacion   TEXT    DEFAULT '',
            valor_nutricional     TEXT    DEFAULT '',
            plagas_riesgos        TEXT    DEFAULT '',
            indice_calidad        REAL    DEFAULT 0,
            notas_usuario         TEXT    DEFAULT '',
```

```
            temperatura_ideal     TEXT    DEFAULT '',
            vida_util_dias        TEXT    DEFAULT '',
            variedad_detectada    TEXT    DEFAULT '',
            procesado_gpt         BOOLEAN DEFAULT 0,
            modelo_version        TEXT    DEFAULT 'v5.0',
            imagen_width          INTEGER DEFAULT 0,
            imagen_height         INTEGER DEFAULT 0,
            created_at            TIMESTAMP DEFAULT CURRENT_TIMESTAMP
        )
    ''')

    cursor.execute("CREATE INDEX IF NOT EXISTS idx_det ON
biblioteca(has_detection)")
    cursor.execute("CREATE INDEX IF NOT EXISTS idx_reg ON
biblioteca(region_peru)")
    cursor.execute("CREATE INDEX IF NOT EXISTS idx_ts  ON
biblioteca(created_at)")
    cursor.execute("CREATE INDEX IF NOT EXISTS idx_cal ON
biblioteca(indice_calidad)")

    conn.commit()
    conn.close()

    _migrar_columnas()
    logger.info(f"Base de datos lista: {DB_PATH}")


def _migrar_columnas() -> None:
    """Agrega columnas nuevas a tablas existentes (migración segura)."""
    nuevas = [
        ("precio_estimado_kg",  "TEXT    DEFAULT ''"),
        ("calidad_exportacion", "TEXT    DEFAULT ''"),
        ("valor_nutricional",   "TEXT    DEFAULT ''"),
        ("plagas_riesgos",      "TEXT    DEFAULT ''"),
        ("indice_calidad",      "REAL    DEFAULT 0"),
        ("notas_usuario",       "TEXT    DEFAULT ''"),
        ("temperatura_ideal",   "TEXT    DEFAULT ''"),
        ("vida_util_dias",      "TEXT    DEFAULT ''"),
        ("variedad_detectada",  "TEXT    DEFAULT ''"),
        ("procesado_gpt",       "BOOLEAN DEFAULT 0"),
        ("modelo_version",      "TEXT    DEFAULT 'v5.0'"),
        ("imagen_width",        "INTEGER DEFAULT 0"),
        ("imagen_height",       "INTEGER DEFAULT 0"),
    ]
    conn   = sqlite3.connect(DB_PATH)
    cursor = conn.cursor()
    cursor.execute("PRAGMA table_info(biblioteca)")
    existentes = {row[1] for row in cursor.fetchall()}
    for col, tipo in nuevas:
```

```
        if col not in existentes:
            try:
                cursor.execute(f"ALTER TABLE biblioteca ADD COLUMN {col}
{tipo}")
                logger.info(f"Columna migrada: {col}")
            except Exception as e:
                logger.warning(f"No se pudo migrar {col}: {e}")
    conn.commit()
    conn.close()


def get_db() -> sqlite3.Connection:
    conn = sqlite3.connect(DB_PATH)
    conn.row_factory = sqlite3.Row
    return conn

def save_to_db(entry: dict) -> int:
    conn   = get_db()
    cursor = conn.cursor()
    try:
        cursor.execute('''
            INSERT INTO biblioteca (
                timestamp, detecciones, url, thumbnail_url, has_detection,
                original_filename, source, confidence_average,
detection_count,
                region_peru, departamento, descripcion_ia, recomendaciones,
                porcentaje_maduracion, clima_recomendado, consejos_cultivo,
                tiempo_maduracion, almacenamiento, mercado_local,
                precio_estimado_kg, calidad_exportacion, valor_nutricional,
                plagas_riesgos, indice_calidad, notas_usuario,
                temperatura_ideal, vida_util_dias, variedad_detectada,
                procesado_gpt, modelo_version, imagen_width, imagen_height
            ) VALUES
(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)
        ''', (
            entry['timestamp'],
            json.dumps(entry['detecciones'], ensure_ascii=False),
            entry['url'],
            entry['thumbnail_url'],
            entry['has_detection'],
            entry['original_filename'],
            entry['source'],
            entry['confidence_average'],
            entry['detection_count'],
            entry.get('region_peru',           ''),
            entry.get('departamento',          ''),
            entry.get('descripcion_ia',        ''),
            entry.get('recomendaciones',       ''),
            entry.get('porcentaje_maduracion', ''),
```

```
            entry.get('clima_recomendado',     ''),
            entry.get('consejos_cultivo',      ''),
            entry.get('tiempo_maduracion',     ''),
            entry.get('almacenamiento',        ''),
            entry.get('mercado_local',         ''),

            entry.get('precio_estimado_kg',    ''),
            entry.get('calidad_exportacion',   ''),
            entry.get('valor_nutricional',     ''),
            entry.get('plagas_riesgos',        ''),
            entry.get('indice_calidad',         0),
            entry.get('notas_usuario',         ''),
            entry.get('temperatura_ideal',     ''),
            entry.get('vida_util_dias',        ''),
            entry.get('variedad_detectada',    ''),
            entry.get('procesado_gpt',          0),
            entry.get('modelo_version',       'v5.0'),
            entry.get('imagen_width',           0),
            entry.get('imagen_height',          0),
        ))
        eid = cursor.lastrowid
        conn.commit()
        logger.info(f"DB insert id={eid} det={entry['detection_count']}")
        return eid
    except Exception as e:
        logger.error(f"Error guardando en DB: {e}", exc_info=True)
        raise
    finally:
        conn.close()

def get_biblioteca(page: int = 1, per_page: int = 20) -> dict:
    page     = max(1, page)
    per_page = min(50, max(1, per_page))
    offset   = (page - 1) * per_page

    conn   = get_db()
    cursor = conn.cursor()
    cursor.execute('SELECT COUNT(*) FROM biblioteca')
    total = cursor.fetchone()[0]
    cursor.execute(
        'SELECT * FROM biblioteca ORDER BY id DESC LIMIT ? OFFSET ?',
        (per_page, offset)
    )
    rows = cursor.fetchall()
    conn.close()

    items = []
    for row in rows:
        item = dict(row)
```

```
        try:
            item['detecciones'] = json.loads(item['detecciones']) if
item['detecciones'] else []
        except Exception:
            item['detecciones'] = []
        items.append(item)

    return {"items": items, "total": total, "page": page, "per_page":
per_page}


def get_urls_by_id(db_id: int) -> tuple | None:
    conn   = get_db()
    cursor = conn.cursor()
    cursor.execute('SELECT url, thumbnail_url FROM biblioteca WHERE id=?',
(db_id,))
    row = cursor.fetchone()
    conn.close()
    return (row['url'], row['thumbnail_url']) if row else None


def delete_by_id(db_id: int) -> bool:
    conn   = get_db()
    cursor = conn.cursor()
    cursor.execute('DELETE FROM biblioteca WHERE id=?', (db_id,))
    affected = cursor.rowcount
    conn.commit()
    conn.close()
    return affected > 0


def delete_all() -> int:
    conn   = get_db()
    cursor = conn.cursor()
    cursor.execute('DELETE FROM biblioteca')
    affected = cursor.rowcount
    conn.commit()
    conn.close()
    logger.warning(f"DB delete_all: {affected} registros eliminados")
    return affected

def update_notas(db_id: int, notas: str) -> bool:
    conn   = get_db()
    cursor = conn.cursor()
    cursor.execute('UPDATE biblioteca SET notas_usuario=? WHERE id=?',
(notas, db_id))
    affected = cursor.rowcount
    conn.commit()
    conn.close()
```

```
    return affected > 0


def get_estadisticas() -> dict:
    conn   = get_db()
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM biblioteca')
    rows = cursor.fetchall()
    conn.close()

    biblioteca = []
    for row in rows:
        item = dict(row)
        try:
            item['detecciones'] = json.loads(item['detecciones']) if
item['detecciones'] else []
        except Exception:
            item['detecciones'] = []
        biblioteca.append(item)

    total      = len(biblioteca)
    detectadas = sum(1 for e in biblioteca if e['has_detection'])

    conteo      = {}
    conf_map    = {}
    regiones    = {}
    deptos      = {}
    actividad   = defaultdict(int)
    nivel_count = {'maduro': 0, 'verde': 0, 'podrido': 0}
    calidad_sum = 0
    calidad_n   = 0
    gpt_count   = sum(1 for e in biblioteca if e.get('procesado_gpt'))

    dias_es = {
        'Monday':'Lun','Tuesday':'Mar','Wednesday':'Mie',
        'Thursday':'Jue','Friday':'Vie','Saturday':'Sab','Sunday':'Dom'
    }

    semanas = defaultdict(int)

    for entry in biblioteca:
        r = entry.get('region_peru', '') or 'Sin región'
        regiones[r] = regiones.get(r, 0) + 1
        d = entry.get('departamento', '')
        if d:
            deptos[d] = deptos.get(d, 0) + 1

        iq = entry.get('indice_calidad', 0) or 0
        if iq > 0:
```

```
            calidad_sum += iq
            calidad_n   += 1

        for det in entry['detecciones']:
            cl = det.get('clase', det.get('clase_raw', '?'))
            conteo[cl] = conteo.get(cl, 0) + 1
            conf_map.setdefault(cl, []).append(det['conf'])
            nv = det.get('nivel', '')
            if nv in nivel_count:
                nivel_count[nv] += 1

        try:
            dt  = datetime.strptime(entry['timestamp'], "%d/%m/%Y %H:%M")
            dia = dias_es.get(dt.strftime('%A'), dt.strftime('%a'))
            actividad[dia] += 1
            # semana del año
            wk = dt.strftime('%Y-W%U')
            semanas[wk] += 1
        except Exception:
            pass

    dias_ord   = ['Lun','Mar','Mie','Jue','Vie','Sab','Dom']
    top_frutas = sorted(conteo.items(), key=lambda x: x[1],
reverse=True)[:10]
    top_deptos = sorted(deptos.items(),  key=lambda x: x[1],
reverse=True)[:5]
    avg_conf   = {c: round(sum(v)/len(v), 3) for c, v in conf_map.items()}

    indice_calidad_prom = round(calidad_sum / calidad_n, 1) if calidad_n > 0
else 0

    tasa_deteccion = round((detectadas / total * 100), 1) if total > 0 else
0

    semanas_sorted = sorted(semanas.items())[-6:]

    return {
        "total":              total,
        "detectadas":         detectadas,
        "no_detectadas":      total - detectadas,
        "tasa_deteccion":     tasa_deteccion,
        "clases":             conteo,
        "avg_conf":           avg_conf,
        "regiones":           regiones,
        "departamentos":      dict(top_deptos),
        "top_frutas":         top_frutas,
        "actividad_semana":   {d: actividad.get(d, 0) for d in dias_ord},
        "nivel_maduracion":   nivel_count,
        "indice_calidad_prom": indice_calidad_prom,
```

```
        "procesadas_con_gpt":  gpt_count,
        "tendencia_semanal":   dict(semanas_sorted),
    }
```

entrenamiento/entrenamiento.py

```
from google.colab import drive
import os

drive.mount('/content/drive')

zip_path = "/content/drive/MyDrive/fruits_datasets.zip"

dest_folder = "/content/drive/MyDrive/dataset_frutas_fijo"

if not os.path.exists(dest_folder):
    os.makedirs(dest_folder, exist_ok=True)
    print("Descomprimiendo... esto tardará unos minutos en Drive.")
    !unzip -q "{zip_path}" -d "{dest_folder}"
    print("¡Descompresión terminada!")
else:
    print("El dataset ya está en Drive. Saltando descompresión.")

!pip install -q ultralytics
from ultralytics import YOLO

yaml_path = f"{dest_folder}/fruits_datasets/data.yaml"

model = YOLO('yolov8s.pt')
results = model.train(
    data='/content/drive/MyDrive/dataset_frutas_fijo/fruits_datasets/data.ya
ml',
    epochs=60
    patience=10,
    batch=16,
    imgsz=640,
    device=0,
    save_period=10,
    cache=False,
    project='/content/drive/MyDrive/yolo_runs',
    name='fruits_v8s',
    exist_ok=True
)


!pip install -q ultralytics
from google.colab import drive
from ultralytics import YOLO
```

```
import os

drive.mount('/content/drive')

last_checkpoint =
'/content/drive/MyDrive/yolo_runs/fruits_v8s/weights/last.pt'

if os.path.exists(last_checkpoint):
    print("Retomando entrenamiento desde el último punto...")
    model = YOLO(last_checkpoint)
    model.train(resume=True)
else:
    print("No se encontró el archivo last.pt. Verifica la ruta en tu
Drive.")

from google.colab import drive, files

from ultralytics import YOLO
import cv2
import matplotlib.pyplot as plt

drive.mount('/content/drive')

model_path = "/content/drive/MyDrive/best_yolov8s_fruits_v6.pt"
model = YOLO(model_path)
print(" Modelo cargado correctamente desde:", model_path)


print("\n📤 Sube una o varias imágenes para probar el modelo...")
uploaded = files.upload()


for filename in uploaded.keys():
    print(f"\n Procesando imagen: {filename}")
    results = model.predict(
        source=filename,
        conf=0.6,
        show=False
    )

    res_plotted = results[0].plot()
    plt.figure(figsize=(8, 8))
    plt.imshow(cv2.cvtColor(res_plotted, cv2.COLOR_BGR2RGB))
    plt.axis('off')
    plt.title(f"Predicción: {filename}")
    plt.show()
```

entrenamiento/extraer_y_mantener2.py

```
import os
import zipfile
import shutil

zip_name = "concha.zip"
dataset_dir = os.path.splitext(zip_name)[0]
num_imagenes = 450
clase_objetivo = "0"
nuevo_id = "11"
carpetas_buscar = ["train", "valid", "test"]
destino = "subset_filtered"

if zip_name.endswith(".zip"):
    if not os.path.exists(dataset_dir):
        with zipfile.ZipFile(zip_name, 'r') as zip_ref:
            zip_ref.extractall(dataset_dir)
        print(f"☑ ZIP extraído en: {dataset_dir}")
    else:
        print(f"⚠ Ya existe la carpeta: {dataset_dir}")
else:
    print(f"🗁 Usando dataset ya descomprimido: {dataset_dir}")

dst_images = os.path.join(destino, "images")
dst_labels = os.path.join(destino, "labels")
os.makedirs(dst_images, exist_ok=True)
os.makedirs(dst_labels, exist_ok=True)

copiadas = 0
for subset in carpetas_buscar:
    src_images = os.path.join(dataset_dir, subset, "images")
    src_labels = os.path.join(dataset_dir, subset, "labels")

    if not os.path.exists(src_images) or not os.path.exists(src_labels):
        continue

    print(f"🔎 Buscando en {subset}...")

    for lbl_name in sorted(os.listdir(src_labels)):
        if not lbl_name.endswith(".txt"):
            continue

        lbl_path = os.path.join(src_labels, lbl_name)
        with open(lbl_path, "r") as f:
            lineas = f.readlines()

        nuevas = []
        for linea in lineas:
```

```
            partes = linea.strip().split()
            if partes and partes[0] == clase_objetivo:
                partes[0] = nuevo_id
                nuevas.append(" ".join(partes))

        if nuevas:
            base = os.path.splitext(lbl_name)[0]
            posibles_ext = [".jpg", ".png", ".jpeg"]
            img_path = None
            for ext in posibles_ext:
                ruta = os.path.join(src_images, base + ext)
                if os.path.exists(ruta):
                    img_path = ruta
                    break

            if img_path:
                shutil.copy(img_path, os.path.join(dst_images,
os.path.basename(img_path)))

                with open(os.path.join(dst_labels, lbl_name), "w") as f:
                    f.write("\n".join(nuevas))

                copiadas += 1

            if copiadas >= num_imagenes:
                break
    if copiadas >= num_imagenes:
        break

print(f"\n☑ Se copiaron {copiadas} imágenes con solo la clase
{clase_objetivo} (guardadas en {destino}/)")
```

entrenamiento/preparar.py

```
import os
import shutil
import random
from pathlib import Path
import yaml

RUTA_ORIGEN = "datasets"
RUTA_DESTINO = "fruits_datasets"
CLASES = [
    "apple_ripe", "apple_unripe", "apple_rotten",
    "mango_ripe", "mango_unripe", "mango_rotten",
    "pineapple_ripe", "pineapple_unripe", "pineapple_rotten",
    "strawberry_ripe", "strawberry_unripe", "strawberry_rotten",
    "banana_ripe", "banana_unripe", "banana_rotten",
```

```
    "avocado_ripe", "avocado_unripe", "avocado_rotten",
    "papaya_ripe", "papaya_unripe", "papaya_rotten",
    "grape_ripe", "grape_unripe", "grape_rotten",
    "orange_ripe", "orange_unripe", "orange_rotten",
    "no_fruit"
]

for split in ["train", "valid", "test"]:
    os.makedirs(os.path.join(RUTA_DESTINO, split, "images"), exist_ok=True)
    os.makedirs(os.path.join(RUTA_DESTINO, split, "labels"), exist_ok=True)

pares = []
for fruta in os.listdir(RUTA_ORIGEN):
    fruta_path = os.path.join(RUTA_ORIGEN, fruta)
    if os.path.isdir(fruta_path):
        for clase in os.listdir(fruta_path):
            clase_path = os.path.join(fruta_path, clase)
            img_dir = os.path.join(clase_path, "images")
            lbl_dir = os.path.join(clase_path, "labels")
            if os.path.exists(img_dir) and os.path.exists(lbl_dir):
                imagenes = [f for f in os.listdir(img_dir) if
f.lower().endswith(('.jpg', '.png', '.jpeg'))]
                for img in imagenes:
                    label = img.rsplit('.', 1)[0] + ".txt"
                    if os.path.exists(os.path.join(lbl_dir, label)):
                        pares.append((os.path.join(img_dir, img),
os.path.join(lbl_dir, label)))

random.shuffle(pares)
n_total = len(pares)
n_train = int(n_total * 0.8)
n_valid = int(n_total * 0.1)

splits = {
    "train": pares[:n_train],
    "valid": pares[n_train:n_train + n_valid],
    "test": pares[n_train + n_valid:]
}

for split, lista in splits.items():
    for img, lbl in lista:
        shutil.copy(img, os.path.join(RUTA_DESTINO, split, "images"))
        shutil.copy(lbl, os.path.join(RUTA_DESTINO, split, "labels"))

data = {
    "train": "./train/images",
    "val": "./valid/images",
    "test": "./test/images",
    "nc": len(CLASES),
```

```
    "names": CLASES
}

with open(os.path.join(RUTA_DESTINO, "data.yaml"), "w") as f:
    yaml.dump(data, f)

print(f"☑ Dataset listo en: {RUTA_DESTINO}")
print("🗀 Estructura YOLO creada correctamente.")
```

UNID
UNIVERSIDAD INTERAMERICANA PARA EL DESARROLLO