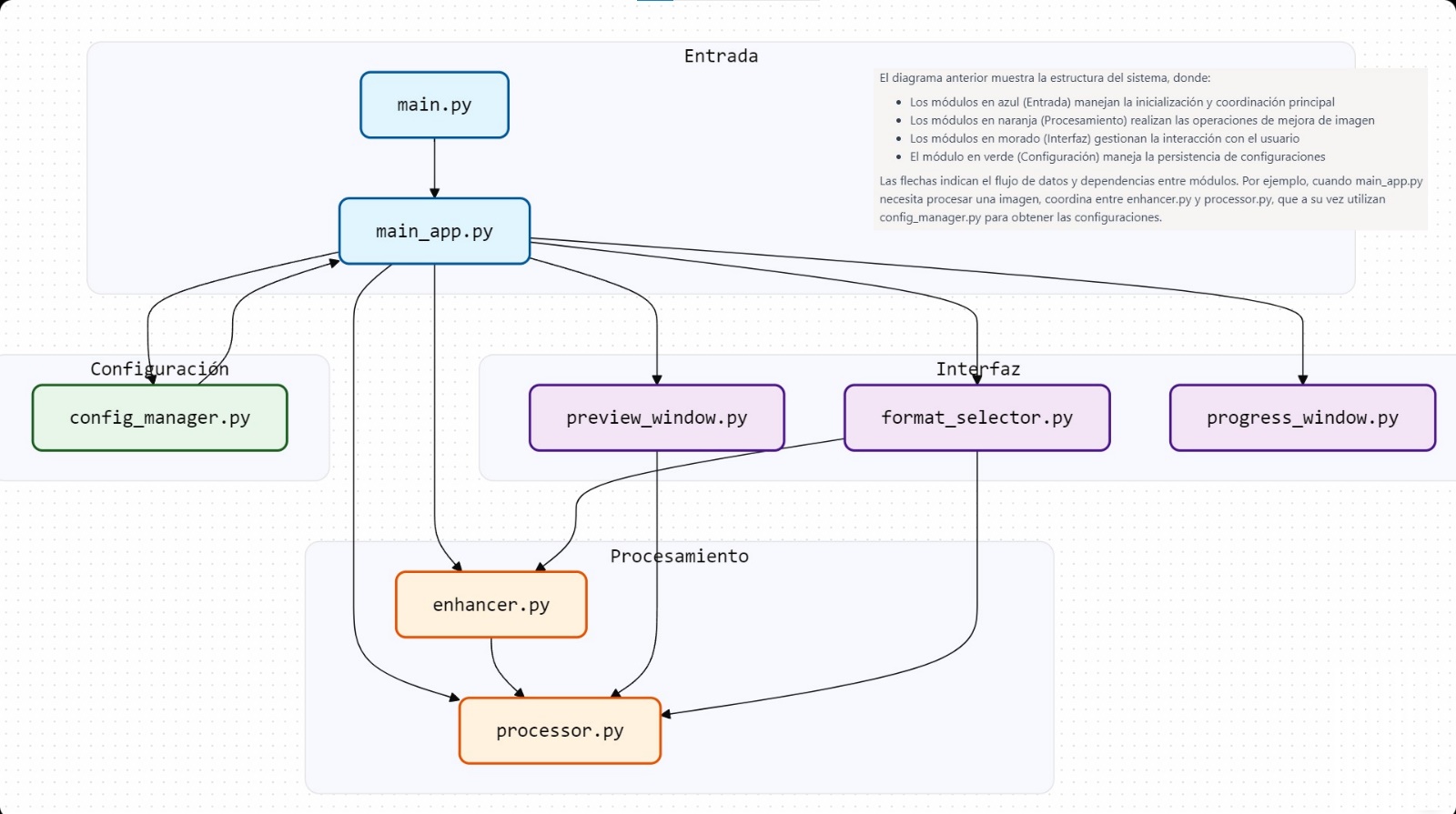
****

**MÓDULOS**

1. **# main.py**

import sys

import os

from pathlib import Path

import tkinter as tk

from tkinter import messagebox

from main\_app import MainApp

# Configuración crítica para PyInstaller y Tesseract

if getattr(sys, 'frozen', False):

    # 1. Configura Tesseract

    tessdata\_path = Path(sys.executable).parent / 'tessdata'

    os.environ['TESSDATA\_PREFIX'] = str(tessdata\_path)

    # 2. Añade rutas de bibliotecas empaquetadas

    sys.path.append(str(Path(sys.executable).parent / 'basicsr'))

    sys.path.append(str(Path(sys.executable).parent / 'numba'))

if \_\_name\_\_ == "\_\_main\_\_":

    try:

        root = tk.Tk()

        app = MainApp(root)

        root.mainloop()

    except Exception as e:

        try:

            messagebox.showerror("Error", f"Ha ocurrido un error al iniciar la aplicación: {e}")

        except:

            print(f"Error crítico: {e}")

1. **# main\_app.py**

import os

import cv2

import time

import subprocess

from datetime import datetime

from tkinter import filedialog, messagebox

from config\_manager import ConfigManager

from enhancer import ImageEnhancer

from preview\_window import SharpnessPreviewWindow

from format\_selector import FormatSelectorWindow

from processor import ZONA\_HORARIA

class MainApp:

    def \_\_init\_\_(self, root):

        self.root = root

        self.root.withdraw()

        self.config\_manager = ConfigManager()

        self.enhancer = ImageEnhancer()

        self.nitidez = self.config\_manager.get("nitidez", float)

        self.nitidez\_texto = self.config\_manager.get("nitidez\_texto", float)

        self.deteccion\_texto = self.config\_manager.get("deteccion\_texto", bool)

        self.modo\_debug = self.config\_manager.get("modo\_debug", bool)

        self.abrir\_carpetas = self.config\_manager.get("abrir\_carpetas", bool)

        self.borrar\_origen = self.config\_manager.get("borrar\_origen", bool)

        self.entrada\_previa = self.config\_manager.get("entrada\_reciente")

        self.salida\_previa = self.config\_manager.get("salida\_reciente")

        self.\_iniciar()

    def \_iniciar(self):

        carpeta\_entrada = filedialog.askdirectory(

            title="Selecciona la carpeta de entrada",

            initialdir=self.entrada\_previa if self.entrada\_previa and os.path.exists(self.entrada\_previa) else os.getcwd()

        )

        if not carpeta\_entrada:

            return

        archivos = [f for f in os.listdir(carpeta\_entrada) if f.lower().endswith((".png", ".jpg", ".jpeg"))]

        if not archivos:

            messagebox.showerror("Error", "No hay imágenes válidas en la carpeta seleccionada.")

            return

        imagen\_prueba = os.path.join(carpeta\_entrada, archivos[0])

        def despues\_de\_nitidez(nit, nit\_txt, det\_texto, debug):

            carpeta\_salida = filedialog.askdirectory(

                title="Selecciona la carpeta de salida",

                initialdir=self.salida\_previa if self.salida\_previa and os.path.exists(self.salida\_previa) else os.getcwd()

            )

            if not carpeta\_salida:

                return

            # Guardar configuración

            self.config\_manager.save({

                "nitidez": nit,

                "nitidez\_texto": nit\_txt,

                "deteccion\_texto": int(det\_texto),

                "abrir\_carpetas": int(self.abrir\_carpetas),

                "borrar\_origen": int(self.borrar\_origen),

                "entrada\_reciente": carpeta\_entrada,

                "salida\_reciente": carpeta\_salida,

                "modo\_debug": int(debug),

                "minimo\_palabras\_texto": 3

            })

            # Mostrar selector de formato (nueva ventana)

            FormatSelectorWindow(

                root=self.root,

                archivos=archivos,

                carpeta\_entrada=carpeta\_entrada,

                carpeta\_salida=carpeta\_salida,

                nitidez=nit,

                nitidez\_texto=nit\_txt,

                deteccion\_texto=det\_texto,

                modo\_debug=debug,

                abrir\_carpetas=self.abrir\_carpetas,

                borrar\_origen=self.borrar\_origen,

                enhancer=self.enhancer

            )

        # Lanzar vista previa

        SharpnessPreviewWindow(

            self.root, imagen\_prueba,

            self.nitidez, self.nitidez\_texto, self.deteccion\_texto, self.modo\_debug,

            callback=despues\_de\_nitidez

        )

1. **# config\_manager.py**

import configparser

import os

class ConfigManager:

    def \_\_init\_\_(self, path\_ini="mejora\_imagenes\_IA.ini"):

        self.path = path\_ini

        self.config = configparser.ConfigParser()

        self.default\_config()

        self.load()

    def default\_config(self):

        self.config["CONFIG"] = {

            "nitidez": "1.0",

            "nitidez\_texto": "1.5",

            "abrir\_carpetas": "1",

            "borrar\_origen": "0",

            "entrada\_reciente": "",

            "salida\_reciente": "",

            "deteccion\_texto": "1",

            "modo\_debug": "0",

            "minimo\_palabras\_texto": "2",

            "idiomas": "spa+eng"

        }

    def load(self):

        if os.path.exists(self.path):

            try:

                self.config.read(self.path, encoding='utf-8')

                print(f"🔧 Configuración cargada: {dict(self.config['CONFIG'])}")

            except Exception as e:

                print(f"Error al leer configuración: {e}")

    def save(self, valores):

        for k, v in valores.items():

            self.config["CONFIG"][k] = str(v)

        try:

            with open(self.path, "w", encoding='utf-8') as f:

                self.config.write(f)

            print(f"✅ Configuración guardada: {dict(self.config['CONFIG'])}")

        except Exception as e:

            print(f"❌ Error al guardar configuración: {e}")

    def get(self, clave, tipo=str):

        valor = self.config["CONFIG"].get(clave)

        if tipo == bool:

            return bool(int(valor))

        elif tipo == float:

            return float(valor)

        elif tipo == int:

            return int(valor)

        return valor

1. **# preview\_window.py**

import tkinter as tk

from tkinter import ttk, messagebox

from PIL import Image, ImageTk

import cv2

import numpy as np

from processor import ImageProcessor

class SharpnessPreviewWindow:

    def \_\_init\_\_(self, root, imagen\_path, nitidez\_inicial, nitidez\_texto\_inicial,

                 deteccion\_texto\_inicial, modo\_debug\_inicial, callback):

        self.callback = callback

        self.imagen\_path = imagen\_path

        self.nitidez\_val = tk.DoubleVar(value=nitidez\_inicial)

        self.nitidez\_texto\_val = tk.DoubleVar(value=nitidez\_texto\_inicial)

        self.detectar\_texto\_val = tk.BooleanVar(value=deteccion\_texto\_inicial)

        self.modo\_debug\_val = tk.BooleanVar(value=modo\_debug\_inicial)

        self.texto\_detectado\_var = tk.StringVar()

        self.ventana = tk.Toplevel(root)

        self.ventana.title("Configuración de nitidez")

        self.ventana.geometry("600x650")

        self.ventana.protocol("WM\_DELETE\_WINDOW", self.\_cerrar\_ventana)

        self.\_construir\_interfaz()

    def \_cerrar\_ventana(self):

        """Maneja el cierre seguro de la ventana"""

        if hasattr(self, 'imagen\_cv'):

            cv2.destroyAllWindows()

        self.ventana.destroy()

    def \_construir\_interfaz(self):

        main\_frame = tk.Frame(self.ventana)

        main\_frame.pack(fill=tk.BOTH, expand=1)

        canvas = tk.Canvas(main\_frame)

        scrollbar = ttk.Scrollbar(main\_frame, orient=tk.VERTICAL, command=canvas.yview)

        scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

        canvas.pack(side=tk.LEFT, fill=tk.BOTH, expand=1)

        canvas.configure(yscrollcommand=scrollbar.set)

        inner\_frame = tk.Frame(canvas)

        canvas.create\_window((0, 0), window=inner\_frame, anchor=tk.NW)

        def \_on\_mousewheel(event):

            if canvas.winfo\_exists():  # Solo si el canvas existe

                try:

                    canvas.yview\_scroll(int(-1 \* (event.delta / 120)), "units")

                except:

                    pass  # Ignorar errores si el canvas ya no responde

        canvas.bind\_all("<MouseWheel>", \_on\_mousewheel)

        self.ventana.bind("<Destroy>", lambda e: canvas.unbind\_all("<MouseWheel>"))

        try:

            self.imagen\_cv = cv2.imread(self.imagen\_path)

            if self.imagen\_cv is None:

                raise ValueError("No se pudo leer la imagen")

            self.imagen\_original = cv2.cvtColor(self.imagen\_cv, cv2.COLOR\_BGR2RGB)

            self.tiene\_texto = ImageProcessor.detectar\_texto(self.imagen\_cv, debug=False)

            self.texto\_detectado\_var.set(f"{'✓ Texto detectado' if self.tiene\_texto else '✗ No se detectó texto'}")

        except Exception as e:

            messagebox.showerror("Error", f"Error al cargar imagen: {str(e)}")

            self.ventana.destroy()

            return

        frame\_img = tk.Frame(inner\_frame)

        frame\_img.pack(pady=10)

        self.label\_img = tk.Label(frame\_img)

        self.label\_img.pack()

        frame\_controles = tk.LabelFrame(inner\_frame, text="Configuración de nitidez")

        frame\_controles.pack(pady=5, padx=20, fill=tk.X)

        tk.Scale(frame\_controles, from\_=0.0, to=3.0, resolution=0.1,

                 orient=tk.HORIZONTAL, label="Nitidez estándar",

                 variable=self.nitidez\_val, length=400).pack(pady=5)

        tk.Scale(frame\_controles, from\_=0.0, to=3.0, resolution=0.1,

                 orient=tk.HORIZONTAL, label="Nitidez para texto",

                 variable=self.nitidez\_texto\_val, length=400).pack(pady=5)

        tk.Label(frame\_controles, textvariable=self.texto\_detectado\_var,

                 font=('Arial', 10, 'bold')).pack(pady=5)

        tk.Checkbutton(frame\_controles, text="Habilitar detección automática de texto",

                       variable=self.detectar\_texto\_val).pack(anchor=tk.W, pady=2)

        tk.Checkbutton(frame\_controles, text="Modo debug",

                       variable=self.modo\_debug\_val).pack(anchor=tk.W, pady=2)

        self.nitidez\_val.trace\_add('write', lambda \*\_: self.\_actualizar\_preview())

        self.nitidez\_texto\_val.trace\_add('write', lambda \*\_: self.\_actualizar\_preview())

        frame\_botones = tk.Frame(inner\_frame)

        frame\_botones.pack(pady=10)

        tk.Button(frame\_botones, text="Probar detección de texto",

                  command=self.\_probar\_deteccion).pack(side=tk.LEFT, padx=5)

        tk.Button(frame\_botones, text="Confirmar configuración",

                  command=self.\_confirmar).pack(side=tk.LEFT, padx=5)

        self.\_actualizar\_preview()

        inner\_frame.update\_idletasks()

        canvas.config(scrollregion=canvas.bbox("all"))

    def \_actualizar\_preview(self):

        nivel\_nitidez = self.nitidez\_texto\_val.get() if self.tiene\_texto else self.nitidez\_val.get()

        sharpened = ImageProcessor.aplicar\_sharpen(self.imagen\_original, nivel\_nitidez, self.tiene\_texto)

        img\_pil = Image.fromarray(sharpened)

        img\_pil.thumbnail((500, 500))

        img\_tk = ImageTk.PhotoImage(img\_pil)

        self.label\_img.configure(image=img\_tk)

        self.label\_img.image = img\_tk

    def \_probar\_deteccion(self):

        self.tiene\_texto = ImageProcessor.detectar\_texto(self.imagen\_cv, debug=self.modo\_debug\_val.get())

        self.texto\_detectado\_var.set(f"{'✓ Texto detectado' if self.tiene\_texto else '✗ No se detectó texto'}")

        self.\_actualizar\_preview()

    def \_confirmar(self):

        self.ventana.destroy()

        self.callback(

            self.nitidez\_val.get(),

            self.nitidez\_texto\_val.get(),

            self.detectar\_texto\_val.get(),

            self.modo\_debug\_val.get()

        )

1. **# enhancer.py**

import os

import sys

import cv2

import numpy as np

from PIL import Image

from realesrgan import RealESRGANer

from basicsr.archs.rrdbnet\_arch import RRDBNet

def resource\_path(relative\_path):

    """

    Obtiene la ruta absoluta al recurso. Funciona para desarrollo y para PyInstaller.

    Versión mejorada con múltiples estrategias de búsqueda.

    """

    try:

        # 1. Intento: PyInstaller crea una carpeta temporal en \_MEIPASS

        base\_path = sys.\_MEIPASS

        full\_path = os.path.join(base\_path, relative\_path)

        if os.path.exists(full\_path):

            return full\_path

    except Exception:

        pass

    try:

        # 2. Intento: Directorio del ejecutable

        base\_path = os.path.dirname(sys.executable)

        full\_path = os.path.join(base\_path, relative\_path)

        if os.path.exists(full\_path):

            return full\_path

    except Exception:

        pass

    try:

        # 3. Intento: Directorio de trabajo actual

        base\_path = os.path.abspath(".")

        full\_path = os.path.join(base\_path, relative\_path)

        if os.path.exists(full\_path):

            return full\_path

    except Exception:

        pass

    try:

        # 4. Intento: Ruta relativa al paquete realesrgan

        import realesrgan

        base\_path = os.path.dirname(realesrgan.\_\_file\_\_)

        full\_path = os.path.join(base\_path, relative\_path)

        if os.path.exists(full\_path):

            return full\_path

    except Exception:

        pass

    # 5. Intento: Ruta absoluta directa como último recurso

    full\_path = os.path.abspath(relative\_path)

    if os.path.exists(full\_path):

        return full\_path

    raise FileNotFoundError(f"No se pudo encontrar el recurso: {relative\_path}")

class ImageEnhancer:

    def \_\_init\_\_(self, weight\_path='weights/RealESRGAN\_x4plus.pth'):

        """

        Inicializa el mejorador de imágenes.

        Args:

            weight\_path (str): Ruta relativa al archivo de pesos del modelo.

                              Por defecto: 'weights/RealESRGAN\_x4plus.pth'

        """

        self.device = 'cuda' if (cv2.cuda.getCudaEnabledDeviceCount() > 0) else 'cpu'

        self.model\_path = resource\_path(weight\_path)

        self.model = None

        self.available\_models = {

            'x4plus': resource\_path('weights/RealESRGAN\_x4plus.pth'),

            'x4plus\_2': resource\_path('weights/RealESRGAN\_x4plus\_2.pth'),

            'general\_x4v3': resource\_path('weights/realesr-general-x4v3.pth')

        }

    def load\_model(self, model\_name='x4plus'):

        """

        Carga el modelo seleccionado.

        Args:

            model\_name (str): Nombre del modelo a cargar. Opciones:

                             - 'x4plus' (por defecto)

                             - 'x4plus\_2'

                             - 'general\_x4v3'

        Raises:

            FileNotFoundError: Si no se encuentra el archivo de pesos

            Exception: Si hay errores al cargar el modelo

        """

        if model\_name not in self.available\_models:

            raise ValueError(f"Modelo desconocido: {model\_name}. Opciones válidas: {list(self.available\_models.keys())}")

        self.model\_path = self.available\_models[model\_name]

        if not os.path.exists(self.model\_path):

            raise FileNotFoundError(

                f"No se encontró el modelo en: {self.model\_path}\n"

                f"Por favor asegúrese que el archivo .pth está en la carpeta weights/"

            )

        try:

            model\_rrdb = RRDBNet(

                num\_in\_ch=3,

                num\_out\_ch=3,

                num\_feat=64,

                num\_block=23,

                num\_grow\_ch=32,

                scale=4

            )

            self.model = RealESRGANer(

                scale=4,

                model\_path=self.model\_path,

                model=model\_rrdb,

                tile=0,

                tile\_pad=10,

                pre\_pad=0,

                half=False,

                device=self.device

            )

        except Exception as e:

            raise Exception(f"Error al cargar el modelo: {str(e)}")

    def enhance(self, image\_cv2, model\_name='x4plus'):

        """

        Mejora la imagen utilizando el modelo seleccionado.

        Args:

            image\_cv2 (numpy.ndarray): Imagen en formato OpenCV (BGR)

            model\_name (str): Nombre del modelo a usar (por defecto: 'x4plus')

        Returns:

            numpy.ndarray: Imagen mejorada en formato OpenCV (BGR)

        Raises:

            Exception: Si hay errores durante el procesamiento

        """

        try:

            if self.model is None or model\_name not in self.available\_models:

                self.load\_model(model\_name)

            # Convertir de BGR (OpenCV) a RGB (PIL)

            imagen\_pil = Image.fromarray(cv2.cvtColor(image\_cv2, cv2.COLOR\_BGR2RGB))

            imagen\_np = np.array(imagen\_pil)

            # Aplicar mejora

            imagen\_mejorada, \_ = self.model.enhance(imagen\_np)

            # Convertir de RGB a BGR

            return cv2.cvtColor(imagen\_mejorada, cv2.COLOR\_RGB2BGR)

        except Exception as e:

            raise Exception(f"Error durante la mejora de imagen: {str(e)}")

1. **# format\_selector.py**

import os

import time

import threading

import subprocess

import tkinter as tk

from tkinter import ttk, messagebox

from datetime import datetime

import cv2

from processor import ZONA\_HORARIA, ImageProcessor

from progress\_window import ProgressWindow

class FormatSelectorWindow:

    def \_\_init\_\_(self, root, archivos, carpeta\_entrada, carpeta\_salida,

                 nitidez, nitidez\_texto, deteccion\_texto, modo\_debug,

                 abrir\_carpetas, borrar\_origen, enhancer):

        self.archivos = archivos

        self.carpeta\_entrada = carpeta\_entrada

        self.carpeta\_salida = carpeta\_salida

        self.nitidez = nitidez

        self.nitidez\_texto = nitidez\_texto

        self.deteccion\_texto = deteccion\_texto

        self.modo\_debug = modo\_debug

        self.enhancer = enhancer

        self.var\_abrir = tk.BooleanVar(value=abrir\_carpetas)

        self.var\_borrar = tk.BooleanVar(value=borrar\_origen)

        self.formato\_val = tk.StringVar(value="auto")

        self.\_construir\_ventana(root)

    def \_construir\_ventana(self, root):

        self.ventana = tk.Toplevel(root)

        self.ventana.title("Mejorador de Imágenes Real-ESRGAN")

        self.ventana.geometry("400x450")

        self.ventana.resizable(False, False)

        tk.Label(self.ventana, text="Formato de salida:", font=("Arial", 12)).pack(pady=5)

        marco\_formatos = tk.LabelFrame(self.ventana)

        marco\_formatos.pack(pady=5, padx=20, fill="both")

        opciones = [

            ("Mantener formato de entrada", "auto"),

            ("PNG", "png"), ("JPG", "jpg"),

            ("BMP", "bmp"), ("TIFF", "tiff"), ("WEBP", "webp")

        ]

        for texto, valor in opciones:

            tk.Radiobutton(marco\_formatos, text=texto, variable=self.formato\_val, value=valor).pack(anchor="w", padx=20)

        marco\_nitidez = tk.LabelFrame(self.ventana, text="Configuración de nitidez")

        marco\_nitidez.pack(pady=5, padx=20, fill="both")

        tk.Label(marco\_nitidez, text=f"Nitidez estándar: {self.nitidez}", font=("Arial", 10)).pack(anchor="w", padx=20)

        tk.Label(marco\_nitidez, text=f"Nitidez para texto: {self.nitidez\_texto}", font=("Arial", 10)).pack(anchor="w", padx=20)

        tk.Label(marco\_nitidez,

                text=f"Detección automática: {'Activada' if self.deteccion\_texto else 'Desactivada'}",

                font=("Arial", 10)).pack(anchor="w", padx=20)

        tk.Checkbutton(self.ventana,

                      text="Abrir carpetas de Entrada y Salida al finalizar",

                      variable=self.var\_abrir).pack(anchor="w", padx=30)

        tk.Checkbutton(self.ventana,

                      text="Borrar archivos de origen procesados correctamente",

                      variable=self.var\_borrar).pack(anchor="w", padx=30)

        estado = tk.Label(self.ventana, text="Esperando para comenzar...", font=("Arial", 10))

        estado.pack(pady=10)

        frame\_boton = tk.Frame(self.ventana)

        frame\_boton.pack(side="bottom", fill="x", pady=15)

        tk.Button(frame\_boton, text="Iniciar", command=self.\_iniciar,

                 font=("Arial", 12), bg="green", fg="white", width=10, height=1).pack(side="bottom")

        self.ventana.attributes("-topmost", True)

        self.ventana.update()

        self.ventana.attributes("-topmost", False)

        self.ventana.update\_idletasks()

        self.estado = estado

    def \_iniciar(self):

        if not self.archivos:

            messagebox.showerror("Error", "No se encontraron imágenes válidas.")

            return

        self.ventana.withdraw()

        # Usamos un hilo normal (no daemon) para evitar cierre prematuro

        self.proceso = threading.Thread(

            target=self.\_procesar\_imagenes,

            args=(self.formato\_val.get(), self.var\_abrir.get(), self.var\_borrar.get())

        )

        self.proceso.start()

    def \_procesar\_imagenes(self, formato\_salida, abrir\_carpetas, borrar\_originales):

        progreso = ProgressWindow(self.ventana)

        sufijo = datetime.now().strftime("\_mejorado\_%Y-%m-%d\_%H-%M")

        log\_path = os.path.join(self.carpeta\_salida, datetime.now().strftime("Imagenes\_Procesadas\_%Y-%m-%d\_%H-%M.log"))

        log\_lineas = ["Nombre\_Imagen\_Original;Nombre\_Imagen\_Procesada;Fecha;Hora\_Inicio;Hora\_Fin;Tiempo\_Transcurrido (hh:mm:ss);Contiene\_Texto;Nitidez\_Aplicada"]

        try:

            self.enhancer.load\_model()

        except Exception as e:

            messagebox.showerror("Error", f"No se pudo cargar el modelo: {e}")

            return

        try:

            for i, nombre\_archivo in enumerate(self.archivos):

                if progreso.cancelar:

                    break

                while progreso.pausar:

                    time.sleep(0.5)

                progreso.actualizar\_contador(i + 1, len(self.archivos))

                progreso.actualizar\_estado(f"Procesando: {nombre\_archivo[:20]}...")

                progreso.actualizar\_progreso(10)

                ruta\_entrada = os.path.join(self.carpeta\_entrada, nombre\_archivo)

                imagen = cv2.imread(ruta\_entrada)

                if imagen is None:

                    continue

                hora\_inicio = datetime.now(ZONA\_HORARIA)

                contiene\_texto = False

                if self.deteccion\_texto:

                    progreso.actualizar\_estado("Analizando texto...")

                    progreso.actualizar\_progreso(30)

                    contiene\_texto = ImageProcessor.detectar\_texto(imagen, min\_palabras=3, debug=self.modo\_debug)

                nivel\_nitidez = self.nitidez\_texto if contiene\_texto else self.nitidez

                progreso.actualizar\_estado("Mejorando imagen...")

                progreso.actualizar\_progreso(60)

                imagen\_mejorada = self.enhancer.enhance(imagen)

                imagen\_mejorada = ImageProcessor.aplicar\_sharpen(imagen\_mejorada, nivel\_nitidez, contiene\_texto)

                hora\_fin = datetime.now(ZONA\_HORARIA)

                tiempo = hora\_fin - hora\_inicio

                ext = os.path.splitext(nombre\_archivo)[1] if formato\_salida == "auto" else f".{formato\_salida}"

                nombre\_salida = os.path.splitext(nombre\_archivo)[0] + sufijo + ext

                ruta\_salida = os.path.join(self.carpeta\_salida, nombre\_salida)

                try:

                    progreso.actualizar\_estado("Guardando resultado...")

                    progreso.actualizar\_progreso(90)

                    cv2.imwrite(ruta\_salida, imagen\_mejorada)

                    log\_lineas.append(f"{nombre\_archivo};{nombre\_salida};{hora\_inicio.strftime('%d/%m/%Y')};{hora\_inicio.strftime('%H:%M:%S')};{hora\_fin.strftime('%H:%M:%S')};{str(tiempo)};{'Sí' if contiene\_texto else 'No'};{nivel\_nitidez}")

                    if borrar\_originales and os.path.exists(ruta\_salida):

                        os.remove(ruta\_entrada)

                except Exception as e:

                    log\_lineas.append(f"{nombre\_archivo};ERROR\_AL\_GUARDAR;{hora\_inicio.strftime('%d/%m/%Y')};{hora\_inicio.strftime('%H:%M:%S')};;;{contiene\_texto};{nivel\_nitidez}")

                progreso.actualizar\_progreso(100)

                time.sleep(0.2)

            with open(log\_path, "w", encoding="utf-8") as log:

                log.write("\n".join(log\_lineas))

            if abrir\_carpetas:

                self.\_abrir\_carpetas\_robusto()

        except Exception as e:

            print(f"Error durante el procesamiento: {e}")

            progreso.finalizar(f"⛔ Error: {str(e)}")

            return

        finally:

            progreso.finalizar("✅ Proceso completado")

            self.ventana.after(100, self.ventana.destroy)

    def \_abrir\_carpetas\_robusto(self):

        """Método ultra-reforzado para apertura de carpetas"""

        print("\n=== INICIANDO APERTURA DE CARPETAS ===")

        # Configuración de carpetas a abrir

        carpetas = [

            ("SALIDA", self.carpeta\_salida),

            ("ENTRADA", self.carpeta\_entrada)

        ]

        # Todos los métodos de apertura posibles

        metodos = [

            {

                "nombre": "os.startfile",

                "funcion": lambda r: os.startfile(r),

                "delay": 1.0

            },

            {

                "nombre": "explorer directo",

                "funcion": lambda r: subprocess.run(['explorer', os.path.normpath(r)], check=True),

                "delay": 1.0

            },

            {

                "nombre": "explorer con shell",

                "funcion": lambda r: subprocess.run(f'explorer "{os.path.normpath(r)}"', shell=True, check=True),

                "delay": 1.5

            }

        ]

        for nombre, ruta in carpetas:

            if not os.path.exists(ruta):

                print(f"× [ERROR] Carpeta {nombre} no existe: {ruta}")

                continue

            print(f"\n● [PROCESANDO] Carpeta {nombre}: {ruta}")

            exito = False

            for metodo in metodos:

                try:

                    print(f"→ [INTENTO] Método: {metodo['nombre']}")

                    metodo['funcion'](ruta)

                    print(f"✓ [ÉXITO] Abierta con {metodo['nombre']}")

                    exito = True

                    time.sleep(metodo['delay'])  # Pausa específica para cada método

                    break

                except Exception as e:

                    print(f"× [FALLO] {metodo['nombre']}: {str(e)}")

                    time.sleep(0.5)  # Pausa entre intentos

            if not exito:

                print(f"‼ [FALLO CRÍTICO] No se pudo abrir {nombre} con ningún método")

            else:

                print(f"✔ [COMPLETADO] Procesamiento de {nombre}")

        print("\n=== APERTURA DE CARPETAS FINALIZADA ===")

1. **# processor.py**

import cv2

import numpy as np

import pytesseract

from pytesseract import Output

import re

import pytz

import os

import sys

from pathlib import Path

ZONA\_HORARIA = pytz.timezone('America/Argentina/Buenos\_Aires')

def get\_tesseract\_cmd():

    """Gestión inteligente de rutas para Tesseract para funcionar en desarrollo y en .exe"""

    try:

        # 1. Intento: Ruta empaquetada (PyInstaller)

        if getattr(sys, 'frozen', False):

            base\_path = Path(sys.\_MEIPASS)

            tesseract\_path = base\_path / 'Tesseract-OCR' / 'tesseract.exe'

            if tesseract\_path.exists():

                return str(tesseract\_path)

        # 2. Intento: Ruta relativa al ejecutable

        exec\_path = Path(sys.executable).parent if getattr(sys, 'frozen', False) else Path(\_\_file\_\_).parent

        tesseract\_path = exec\_path / 'Tesseract-OCR' / 'tesseract.exe'

        if tesseract\_path.exists():

            return str(tesseract\_path)

        # 3. Intento: Ruta de desarrollo (venv)

        venv\_path = Path(sys.prefix) / 'Scripts' / 'tesseract.exe'

        if venv\_path.exists():

            return str(venv\_path)

        # 4. Intento: Ruta de instalación estándar

        default\_paths = [

            r'C:\Program Files\Tesseract-OCR\tesseract.exe',

            r'C:\Program Files (x86)\Tesseract-OCR\tesseract.exe'

        ]

        for path in default\_paths:

            if Path(path).exists():

                return path

        # 5. Intento: Buscar en PATH del sistema

        tesseract\_cmd = pytesseract.pytesseract.tesseract\_cmd

        if tesseract\_cmd and Path(tesseract\_cmd).exists():

            return tesseract\_cmd

    except Exception as e:

        print(f"Error al buscar Tesseract: {e}")

    raise FileNotFoundError(

        "No se encontró Tesseract-OCR. Por favor instálelo o "

        "asegúrese de incluir la carpeta Tesseract-OCR junto al ejecutable."

    )

# Configura la ruta de Tesseract al importar el módulo

try:

    pytesseract.pytesseract.tesseract\_cmd = get\_tesseract\_cmd()

except Exception as e:

    print(f"Advertencia: {str(e)}")

class ImageProcessor:

    @staticmethod

    def aplicar\_sharpen(imagen\_cv2, nitidez=1.0, tiene\_texto=False):

        """

        Aplica filtro de sharpening adaptativo según si la imagen contiene texto

        Args:

            imagen\_cv2: Imagen en formato OpenCV (BGR)

            nitidez: Nivel de nitidez (0.0 a 3.0)

            tiene\_texto: Booleano que indica si la imagen contiene texto

        Returns:

            Imagen procesada con filtro de sharpening

        """

        if tiene\_texto:

            kernel = np.array([

                [0, -0.5, 0],

                [-0.5, 2 + 1.5 \* nitidez, -0.5],

                [0, -0.5, 0]

            ])

        else:

            kernel = np.array([

                [0, -0.2, 0],

                [-0.2, 1 + 0.8 \* nitidez, -0.2],

                [0, -0.2, 0]

            ])

        return cv2.filter2D(imagen\_cv2, -1, kernel, borderType=cv2.BORDER\_REFLECT)

    @staticmethod

    def detectar\_texto(imagen, min\_palabras=2, debug=False):

        """

        Detecta texto en imágenes usando múltiples configuraciones de Tesseract

        Args:

            imagen: Imagen en formato OpenCV (BGR)

            min\_palabras: Mínimo de palabras válidas para considerar que hay texto

            debug: Modo depuración para mostrar información detallada

        Returns:

            Booleano indicando si se detectó texto suficiente

        """

        try:

            # Preprocesamiento de imagen

            gris = cv2.cvtColor(imagen, cv2.COLOR\_BGR2GRAY)

            # Ajuste de gamma para mejorar contraste

            gamma = 1.5

            inv\_gamma = 1.0 / gamma

            table = np.array([((i / 255.0) \*\* inv\_gamma) \* 255 for i in np.arange(0, 256)]).astype("uint8")

            gris = cv2.LUT(gris, table)

            # Binarización adaptativa

            binaria = cv2.adaptiveThreshold(

                gris, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,

                cv2.THRESH\_BINARY\_INV, 41, 10

            )

            # Configuraciones múltiples para mejorar detección

            configs = [

                ('--psm 6 --oem 3 -c tessedit\_char\_whitelist="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyzÁÉÍÓÚáéíóú0123456789-.,:()/°"', 'spa'),

                ('--psm 11 --oem 3', 'spa'),

                ('--psm 4 --oem 3', 'spa+eng')

            ]

            palabras\_validas = []

            for config, lang in configs:

                try:

                    data = pytesseract.image\_to\_data(

                        binaria, config=config,

                        lang=lang, output\_type=Output.DICT

                    )

                    for i, word in enumerate(data['text']):

                        word = word.strip()

                        try:

                            conf = int(data['conf'][i])

                        except (ValueError, TypeError):

                            conf = 0

                        # Validación de palabra detectada

                        if (len(word) >= 3 and conf > 70 and

                            any(c.isalpha() for c in word) and

                            not re.fullmatch(r'^\W+$', word)):

                            palabras\_validas.append(word)

                except Exception as e:

                    if debug:

                        print(f"Error con configuración {config}: {e}")

            # Verificación de resultados

            if debug:

                print("=== DEBUG DETALLADO ===")

                print(f"Configuraciones probadas: {len(configs)}")

                print(f"Palabras únicas detectadas: {list(set(palabras\_validas))}")

                cv2.imshow("Imagen preprocesada", binaria)

                cv2.waitKey(0)

                cv2.destroyAllWindows()

            return len(set(palabras\_validas)) >= min\_palabras

        except Exception as e:

            print(f"Error en detección de texto: {e}")

            if debug:

                raise  # Relanza la excepción en modo debug

            return False

1. **# progress\_window.py**

import tkinter as tk

from tkinter import ttk

import os

class ProgressWindow:

    def \_\_init\_\_(self, master):

        self.master = master

        self.window = tk.Toplevel(master)

        self.window.title("Procesando imágenes...")

        self.window.geometry("450x210")

        self.cancelar = False

        self.pausar = False

        self.\_crear\_widgets()

    def \_crear\_widgets(self):

        self.frame = tk.Frame(self.window)

        self.frame.pack(pady=15, padx=20, fill=tk.BOTH, expand=True)

        tk.Label(self.frame, text="Progreso del archivo actual:", font=('Arial', 9)).pack(anchor=tk.W)

        self.progress\_archivo = ttk.Progressbar(self.frame, orient=tk.HORIZONTAL, length=400, mode='determinate')

        self.progress\_archivo.pack(pady=5)

        self.lbl\_contador = tk.Label(self.frame, text="Archivo 0 de 0", font=('Arial', 9))

        self.lbl\_contador.pack(pady=5)

        self.lbl\_estado = tk.Label(self.frame, text="Preparando...", font=('Arial', 10))

        self.lbl\_estado.pack(pady=5)

        frame\_botones = tk.Frame(self.frame)

        frame\_botones.pack(pady=10)

        self.btn\_pausar = tk.Button(frame\_botones, text="Pausar", width=10, command=self.\_toggle\_pausa)

        self.btn\_pausar.pack(side=tk.LEFT, padx=20)

        self.btn\_finalizar = tk.Button(frame\_botones, text="Finalizar", width=10, command=self.\_cerrar)

        self.btn\_finalizar.pack(side=tk.RIGHT, padx=20)

        self.window.protocol("WM\_DELETE\_WINDOW", self.\_cerrar)

    def \_toggle\_pausa(self):

        self.pausar = not self.pausar

        self.btn\_pausar.config(text="Continuar" if self.pausar else "Pausar")

    def \_cerrar(self):

        self.cancelar = True

        try:

            self.window.destroy()

        except:

            pass

        os.\_exit(0)

    def actualizar\_estado(self, mensaje):

        self.lbl\_estado.config(text=mensaje)

        self.window.update()

    def actualizar\_contador(self, actual, total):

        self.lbl\_contador.config(text=f"Archivo {actual} de {total}")

        self.window.update()

    def actualizar\_progreso(self, valor):

        self.progress\_archivo["value"] = valor

        self.window.update()

    def finalizar(self, mensaje\_final):

        self.actualizar\_estado(mensaje\_final)

        self.btn\_pausar.config(state="disabled")

        self.btn\_finalizar.config(text="Cerrar", command=self.\_cerrar)

        self.window.wait\_window()

**ARCHIVOS DE CONFIGURACIÓN**

1. ***mejora\_imagenes\_IA.ini***

[CONFIG]

nitidez = 0.8

nitidez\_texto = 0.6

abrir\_carpetas = 1

borrar\_origen = 0

entrada\_reciente = Ruta a la carpeta de “Entrada” (donde están las imágenes a procesar)

salida\_reciente = Ruta a la carpeta de “Salida” (donde quedarán las imágenes a procesadas)

deteccion\_texto = 1

modo\_debug = 0

minimo\_palabras\_texto = 3

idiomas = spa+eng

1. ***requirements.txt***

basicSR

addict

future

lmdb

numpy>=2.2.6

opencv-python>=4.9.0

Pillow

pyyaml

requests

scikit-image

scipy

tb-nightly

torch>=1.7

torchvision

tqdm

yapf

pytesseract==0.3.13

pytz

realesrgan==0.3.0

1. ***INSTRUCCIONES INICIALES (LEER.TXT)***
2. ***Colocar los modelos de RealESRGAN dentro de la carpeta 'weights':***

***“C:\Users\xxxxx\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\_POO\weights”:*** *RealESRGAN\_x4plus.pth; RealESRGAN\_x4plus\_2.pth; realesr-general-x4v3.pth*

1. ***ENTORNO***

***RUTA A LOS MÓDULOS:***

“C:\Users\xxxxx\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\_POO”

***RUTA A BASICSR:***

***Instalar en*** *“****C:\Users\xxxxx\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\****”en la carpeta “****BasicSR-master****”*

***RUTA AL ENTORNO:***

“C:\Users\xxxxx\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv”

***RUTA AL PYTHON:***

“C:\Users\xxxxx\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Scripts\python.exe”

***RUTA A REALESRGAN:***

realesrgan in “c:\users\xxxxx\desktop\ciencia de datos e ia\cuatrimestre 5\procesamiento de imagenes\tpf\venv\lib\site-packages”

***RUTA A LOS MODELOS REALESRGAN:***

*“C:\Users\xxxxx\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\_POO\weights”: RealESRGAN\_x4plus.pth; RealESRGAN\_x4plus\_2.pth; realesr-general-x4v3.pth*

***RUTA AL TESSERACT:***

“C:\Program Files\Tesseract-OCR”

***RUTA A PREENTRENADOS DE TESSERACT e IDIOMAS:***

“C:\Program Files\Tesseract-OCR\tessdata”

1. ***Activar entorno virtual:***

python -m venv venv

..\TPF\venv\Scripts\activate

1. ***Instalar dependencias:*** *pip install -r requirements.txt o bien: pip install -r requirements.txt --no-deps (El flag --no-deps evita que pip intente reinstalar todo desde cero, pero instala los paquetes que faltan según el archivo.)*
2. ***Ejecutar script:*** *python main.py*

***BIBLIOTECAS INSTALADAS***

|  |  |  |
| --- | --- | --- |
| **Paquete** | **Versión** | **Ruta** |
| absl-py | 2.2.2 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| addict | 2.4.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| altgraph | 0.17.4 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| auto-py-to-exe | 2.46.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| basicsr | 1.4.2 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| bottle | 0.13.3 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| bottle-websocket | 0.2.9 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| certifi | 2025.4.26 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| cffi | 1.17.1 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| charset-normalizer | 3.4.2 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| colorama | 0.4.6 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| contourpy | 1.3.2 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| cycler | 0.12.1 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| Eel | 0.18.1 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| facexlib | 0.3.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| filelock | 3.18.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| filterpy | 1.4.5 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| fonttools | 4.57.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| fsspec | 2025.3.2 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| future | 1.0.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| gevent | 25.5.1 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| gevent-websocket | 0.10.1 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| gfpgan | 1.3.8 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| greenlet | 3.2.2 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| grpcio | 1.71.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| idna | 3.10 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| imageio | 2.37.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| Jinja2 | 3.1.6 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| kiwisolver | 1.4.8 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| lazy\_loader | 0.4 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| llvmlite | 0.44.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| lmdb | 1.6.2 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| Markdown | 3.8 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| MarkupSafe | 3.0.2 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| matplotlib | 3.10.3 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| mpmath | 1.3.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| networkx | 3.4.2 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| Nuitka | 2.7.3 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| numba | 0.61.2 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| numpy | 2.2.6 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| opencv-contrib-python | 4.11.0.86 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| opencv-python | 4.11.0.86 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| ordered-set | 4.1.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| packaging | 25.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| pefile | 2023.2.7 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| pillow | 11.2.1 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| pip | 25.1.1 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| platformdirs | 4.3.8 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| protobuf | 6.30.2 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| pycparser | 2.22 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| pyinstaller | 6.13.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| pyinstaller-hooks-contrib | 2025.4 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| pyparsing | 3.2.3 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| pytesseract | 0.3.13 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| python-dateutil | 2.9.0.post0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| pytz | 2025.2 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| pywin32-ctypes | 0.2.3 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| PyYAML | 6.0.2 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| realesrgan | 0.3.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| requests | 2.32.3 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| scikit-image | 0.25.2 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| scipy | 1.15.3 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| setuptools | 80.3.1 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| six | 1.17.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| sympy | 1.14.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| tb-nightly | 2.20.0a20250509 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| tensorboard-data-server | 0.7.2 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| tifffile | 2025.3.30 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| tk | 0.1.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| torch | 2.7.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| torchvision | 0.22.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| tqdm | 4.67.1 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| urllib3 | 2.4.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| Werkzeug | 3.1.3 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| wheel | 0.45.1 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| yapf | 0.43.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| zope.event | 5.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| zope.interface | 7.2 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |
| zstandard | 0.23.0 | C:\Users\jfauc\Desktop\CIENCIA DE DATOS e IA\CUATRIMESTRE 5\Procesamiento de Imagenes\TPF\venv\Lib\site-packages |

