```
#-------------------------------------------------
# Alumno: José Roberto Escamilla Meza

# ->Proyecto LLM - 18092025
# Generación de imágenes gente desaparecida

# ->Objetivo:
# Solicitar una imagen de una persona desaparecida hace un tiempo (rasgos, ropa, años de desaparecid@)

# ->Resultados:
# -Imagen de persona desaparecida, de como seria a la fecha actual (teniendo en cuenta los años transcurridos) - X
# -Imagen de la persona desaparecida con la ropa descrita al momento de la desaparición
#-------------------------------------------------
```

```
# ============================================
# CELDA 1: INSTALACIÓN DE DEPENDENCIAS
# ============================================
print("🔧 Instalando dependencias necesarias...")#manipulación de imágenes, Cliente oficial para usar la API de OpenAI
!pip install openai pillow opencv-python-headless numpy -q
print("✅ Dependencias instaladas correctamente\n")
```

```
🔧 Instalando dependencias necesarias...
✅ Dependencias instaladas correctamente
```

```
# ============================================
# CELDA 2: IMPORTS Y CONFIGURACIÓN
# ============================================
import base64 #Codificar/decodificar datos en Base64
import requests #Realizar peticiones HTTP
from openai import OpenAI #Cliente oficial de OpenAI para Python
from PIL import Image #Manipulación de imágenes, Se usa para abrir, editar, redimensionar, convertir formatos y guardar imágenes
import numpy as np #Computación numérica y manejo de arrays
from io import BytesIO #Permite manejar bytes en memoria como si fueran archivos
import cv2 #Visión por computadora, Se utiliza específicamente para la detección de rostros
from google.colab import files #Utilidad de Google Colab para manejo de archivos
from IPython.display import display, Image as IPImage #Módulos de IPython para mostrar contenido en el notebook
import os #Sistema operativo
import time #Manejo de tiempo
from datetime import datetime #fechas y horas

print("📚 Librerías importadas correctamente")

# CONFIGURACIÓN DE API KEY (HARDCODEADA)
# ⚠️ IMPORTANTE: Reemplaza con tu API key real
API_KEY = ""

print("🔑 API Key configurada")
print("=" * 50)
```

```
📚 Librerías importadas correctamente
🔑 API Key configurada
==================================================
```

```
# ============================================
# VERSIÓN CORREGIDA DEFINITIVA
# ============================================

class ProgresionEdadColab:
    def __init__(self, api_key):
        """Inicializa el sistema de progresión de edad"""
        try:
            print("🚀 Inicializando sistema de progresión de edad...")
            self.client = OpenAI(api_key=api_key)
            self.api_key = api_key

            # Verificar que la API key sea válida
            if not api_key or api_key.startswith("sk-XXX"):
                raise ValueError("⚠️ Por favor, configura una API key válida")

            print("✅ Sistema inicializado correctamente")
```

```
            except Exception as e:
                print(f"❌ Error al inicializar: {str(e)}")
                raise

        def cargar_imagen_colab(self):
            """Permite cargar una imagen desde tu computadora"""
            try:
                print("\n📤 Por favor, selecciona una imagen de tu computadora:")
                uploaded = files.upload()

                if not uploaded:
                    raise ValueError("No se seleccionó ninguna imagen")

                # Obtener el primer archivo subido
                filename = list(uploaded.keys())[0]
                print(f"✅ Imagen '{filename}' cargada correctamente")

                # Guardar temporalmente
                with open(filename, 'wb') as f:
                    f.write(uploaded[filename])

                # Abrir y convertir a PNG
                img = Image.open(filename)
                print(f"📐 Dimensiones originales: {img.size}")
                print(f"📷 Formato original: {img.format if img.format else 'Unknown'}")

                # Asegurar que sea RGB o RGBA
                if img.mode not in ['RGB', 'RGBA']:
                    img = img.convert('RGBA')

                # Guardar como PNG con nombre fijo
                png_filename = "original_image.png"
                img.save(png_filename, "PNG", optimize=False)
                print(f"✅ Imagen guardada como PNG: {png_filename}")

                # Verificar que el PNG se guardó correctamente
                test = Image.open(png_filename)
                print(f"✅ Verificación: Formato={test.format}, Modo={test.mode}")

                # Mostrar preview
                display_img = img.copy()
                display_img.thumbnail((300, 300))
                display(display_img)

                return png_filename

            except Exception as e:
                print(f"❌ Error al cargar imagen: {str(e)}")
                return None

        def detectar_rostro(self, imagen_array):
            """Detecta el rostro en la imagen usando OpenCV"""
            try:
                print("👤 Detectando rostro en la imagen...")

                # Convertir a escala de grises
                gray = cv2.cvtColor(imagen_array, cv2.COLOR_RGB2GRAY)

                # Cargar el clasificador
                face_cascade = cv2.CascadeClassifier(
                    cv2.data.haarcascades + 'haarcascade_frontalface_default.xml'
                )

                # Detectar rostros
                faces = face_cascade.detectMultiScale(
                    gray,
                    scaleFactor=1.1,
                    minNeighbors=5,
                    minSize=(30, 30)
                )

                if len(faces) == 0:
                    print("⚠️ No se detectaron rostros, usando área central")
                    # Retornar área central como rostro
                    h, w = gray.shape
                    face_w = w // 2
                    face_h = h // 2
```

```
                x = w // 4
                y = h // 4
                return (x, y, face_w, face_h)

            print(f"✅ {len(faces)} rostro(s) detectado(s)")
            return faces[0]

        except Exception as e:
            print(f"⚠️ Error en detección: {str(e)}")
            # Retornar área central por defecto
            return (256, 256, 512, 512)

    def preparar_imagen_para_edicion(self, imagen_path):
        """Prepara imagen y máscara para DALL-E 2"""
        try:
            print("\n🎨 Preparando imagen para edición...")

            # Cargar imagen original
            img = Image.open(imagen_path)

            # Asegurar que sea RGBA
            if img.mode != 'RGBA':
                img = img.convert('RGBA')

            # Redimensionar a 1024x1024
            img_resized = img.resize((1024, 1024), Image.Resampling.LANCZOS)
            print(f"  - Imagen redimensionada a 1024x1024")

            # Guardar imagen redimensionada
            imagen_final_path = "imagen_para_editar.png"
            img_resized.save(imagen_final_path, "PNG")
            print(f"  - Imagen guardada como: {imagen_final_path}")

            # Crear máscara
            print("  - Creando máscara...")

            # Convertir a array para detección
            img_array = np.array(img_resized.convert("RGB"))

            # Detectar rostro
            face = self.detectar_rostro(img_array)
            x, y, w, h = face

            # Expandir área del rostro
            expansion = 0.3
            x = max(0, int(x - w * expansion/2))
            y = max(0, int(y - h * expansion/2))
            w = min(1024 - x, int(w * (1 + expansion)))
            h = min(1024 - y, int(h * (1 + expansion)))

            print(f"  - Área de edición: x={x}, y={y}, w={w}, h={h}")

            # Crear máscara con PIL
            # La máscara debe ser RGBA donde el área transparente es donde se editará
            mask = Image.new('RGBA', (1024, 1024), (0, 0, 0, 255))

            # Crear un rectángulo transparente en el área del rostro
            for i in range(x, min(x + w, 1024)):
                for j in range(y, min(y + h, 1024)):
                    mask.putpixel((i, j), (0, 0, 0, 0))

            # Guardar máscara
            mascara_path = "mascara_edicion.png"
            mask.save(mascara_path, "PNG")
            print(f"  - Máscara guardada como: {mascara_path}")

            # Mostrar preview
            preview = mask.copy()
            preview.thumbnail((200, 200))
            print("\n👀 Preview de la máscara:")
            display(preview)

            print("✅ Preparación completada")

            return imagen_final_path, mascara_path

        except Exception as e:
```

```
            print(f"❌ Error preparando imagen: {str(e)}")
            raise

    def crear_archivo_png_bytes(self, imagen_path):
        """Crea un objeto BytesIO con formato PNG correcto"""
        try:
            # Abrir la imagen
            img = Image.open(imagen_path)

            # Asegurar que sea RGBA
            if img.mode != 'RGBA':
                img = img.convert('RGBA')

            # Crear BytesIO object
            output = BytesIO()

            # Guardar como PNG en el BytesIO
            img.save(output, format='PNG')

            # Obtener los bytes
            output.seek(0)

            return output

        except Exception as e:
            print(f"❌ Error creando bytes: {str(e)}")
            raise

    def editar_edad_dalle2(self, imagen_path, años_adicionales):
        """Usa DALL-E 2 para modificar edad"""
        try:
            print(f"\n⏳ Iniciando proceso de envejecimiento (+{años_adicionales} años)...")
            print("=" * 50)

            # Preparar imagen y máscara
            img_path, mask_path = self.preparar_imagen_para_edicion(imagen_path)

            # Descripciones de edad
            # descripciones_edad = {
            #     range(5, 10): "young child, soft features",
            #     range(10, 15): "pre-teen, youthful appearance",
            #     range(15, 20): "teenager, young adult features",
            #     range(20, 30): "adult, mature features",
            #     range(30, 40): "mature adult, slight aging signs",
            #     range(40, 50): "middle aged, visible wrinkles, some grey hair",
            #     range(50, 60): "senior, pronounced aging, grey hair",
            #     range(60, 70): "elderly, deep wrinkles, white hair",
            #     range(70, 100): "very elderly, heavily aged features"
            # }

            descripciones_edad = {
                range(5, 20): "portrait of person with youthful appearance, artistic rendering",
                range(20, 30): "professional portrait, adult person, natural appearance",
                range(30, 40): "mature professional portrait, refined features",
                range(40, 50): "distinguished portrait, experienced appearance, wisdom lines",
                range(50, 60): "senior professional portrait, silver highlights in hair",
                range(60, 70): "distinguished elder portrait, graceful aging",
                range(70, 100): "wise elder portrait, silver hair, dignified appearance"
            }

            edad_desc = "mature adult"
            for rango, desc in descripciones_edad.items():
                if años_adicionales in rango:
                    edad_desc = desc
                    break

            print(f"📝 Descripción objetivo: {edad_desc}")

            # Prompt mejorado
            prompt = f"Photorealistic portrait of the same person but {años_adicionales} years older, {edad_desc}, natural aging,

            print(f"🎯 Prompt: {prompt[:100]}...")

            try:
                print("\n🌐 Conectando con OpenAI...")

                # MÉTODO CORRECTO: Abrir archivos como file objects
```

```
                with open(img_path, 'rb') as img_file:
                    with open(mask_path, 'rb') as mask_file:

                        print("📦 Enviando a DALL-E 2...")
                        print("⏳ Por favor espera (10-20 segundos)...")

                        start_time = time.time()

                        # Llamada a la API con file objects directos
                        response = self.client.images.edit(
                            model="dall-e-2",
                            image=img_file,
                            mask=mask_file,
                            prompt=prompt,
                            n=1,
                            size="1024x1024"
                        )

                        elapsed = time.time() - start_time
                        print(f"✅ Imagen generada en {elapsed:.2f} segundos")

                # Procesar resultado
                image_url = response.data[0].url
                print("📥 Descargando resultado...")

                img_response = requests.get(image_url)
                if img_response.status_code != 200:
                    raise Exception(f"Error descargando: HTTP {img_response.status_code}")

                img_resultado = Image.open(BytesIO(img_response.content))

                # Guardar con timestamp
                timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
                filename = f"progresion_{años_adicionales}años_{timestamp}.png"
                img_resultado.save(filename, "PNG")
                print(f"💾 Guardado como: {filename}")

                # Limpiar temporales
                for temp_file in [img_path, mask_path]:
                    if os.path.exists(temp_file):
                        os.remove(temp_file)

                # Mostrar resultado
                print("\n🖼️ Resultado:")
                display(img_resultado)

                return img_resultado, filename

            except Exception as api_error:
                print(f"\n❌ Error de API: {str(api_error)}")

                # Si el error persiste, intentar método alternativo
                print("\n🔄 Intentando método alternativo...")
                return self.metodo_alternativo_variacion(imagen_path, años_adicionales)

        except Exception as e:
            print(f"❌ Error general: {str(e)}")
            return None, None

    def metodo_alternativo_variacion(self, imagen_path, años_adicionales):
        """Método alternativo usando variations si edit falla"""
        try:
            print("\n🔄 Usando método de variaciones...")

            # Cargar y preparar imagen
            img = Image.open(imagen_path)
            if img.mode != 'RGBA':
                img = img.convert('RGBA')

            # Redimensionar
            img = img.resize((1024, 1024), Image.Resampling.LANCZOS)

            # Guardar temporalmente
            temp_path = "temp_variation.png"
            img.save(temp_path, "PNG")

            # Crear variación
```

```
                with open(temp_path, 'rb') as img_file:
                    print("📤 Generando variación...")

                    response = self.client.images.create_variation(
                        model="dall-e-2",
                        image=img_file,
                        n=1,
                        size="1024x1024"
                    )

                # Obtener resultado
                image_url = response.data[0].url
                img_response = requests.get(image_url)
                img_resultado = Image.open(BytesIO(img_response.content))

                # Guardar
                timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
                filename = f"variacion_{timestamp}.png"
                img_resultado.save(filename, "PNG")

                print(f"✅ Variación guardada como: {filename}")
                print("⚠️ Nota: Las variaciones no controlan la edad específicamente")

                # Limpiar
                if os.path.exists(temp_path):
                    os.remove(temp_path)

                display(img_resultado)
                return img_resultado, filename

            except Exception as e:
                print(f"❌ Error en método alternativo: {str(e)}")
                return None, None

        def descargar_resultado(self, filename):
            """Descarga el resultado"""
            try:
                print(f"\n💾 Descargando '{filename}'...")
                if os.path.exists(filename):
                    files.download(filename)
                    print("✅ Descargado correctamente")
                else:
                    print(f"❌ Archivo no encontrado")
            except Exception as e:
                print(f"❌ Error: {str(e)}")

    # # ============================================
    # # FUNCIÓN ALTERNATIVA SI NADA FUNCIONA
    # # ============================================

    # def usar_solo_variaciones():
    #     """Función simplificada que solo usa variaciones"""
    #     print("🎨 MODO SIMPLE: Solo Variaciones")
    #     print("=" * 50)

    #     try:
    #         if API_KEY.startswith("sk-XXX"):
    #             print("⚠️ Configura tu API Key primero")
    #             return

    #         client = OpenAI(api_key=API_KEY)

    #         # Cargar imagen
    #         print("📤 Selecciona una imagen:")
    #         uploaded = files.upload()

    #         if not uploaded:
    #             return

    #         filename = list(uploaded.keys())[0]

    #         # Preparar imagen
    #         img = Image.open(filename)
    #         img = img.convert("RGBA")
    #         img = img.resize((1024, 1024))

    #         temp_file = "temp_var.png"
```

```
#         img.save(temp_file, "PNG")

#         # Mostrar original
#         print("\n📸 Imagen original:")
#         display(img)

#         # Generar variaciones
#         print("\n⏳ Generando 4 variaciones...")

#         with open(temp_file, 'rb') as f:
#             response = client.images.create_variation(
#                 model="dall-e-2",
#                 image=f,
#                 n=4,
#                 size="1024x1024"
#             )

#         # Guardar resultados
#         for i, data in enumerate(response.data):
#             img_response = requests.get(data.url)
#             img_result = Image.open(BytesIO(img_response.content))

#             var_filename = f"variacion_{i+1}.png"
#             img_result.save(var_filename)

#             print(f"\n📸 Variación {i+1}:")
#             display(img_result)

#             # Descargar
#             files.download(var_filename)

#         print("\n✅ Proceso completado")

#     except Exception as e:
#         print(f"❌ Error: {str(e)}")

# Si el método principal falla, descomentar y ejecutar:
# usar_solo_variaciones()
```

```
# ==========================================
# CELDA 4: FUNCIÓN PRINCIPAL DE EJECUCIÓN
# ==========================================
def ejecutar_progresion_edad():
    """Función principal que ejecuta todo el proceso"""

    print("=" * 60)
    print("🎯 SISTEMA DE PROGRESIÓN DE EDAD CON DALL-E 2")
    print("=" * 60)

    try:
        # Verificar API Key
        if API_KEY.startswith("sk-XXX"):
            print("⚠️ ATENCIÓN: Debes configurar tu API Key de OpenAI")
            print("   Ve a la celda 2 y reemplaza 'sk-XXX...' con tu key real")
            return

        # Inicializar sistema
        sistema = ProgresionEdadColab(api_key=API_KEY)

        # Cargar imagen
        imagen_path = sistema.cargar_imagen_colab()

        if not imagen_path:
            print("❌ No se pudo cargar la imagen")
            return

        # Solicitar años a agregar
        print("\n" + "=" * 50)
        años = int(input("¿Cuántos años deseas agregar? (5-70): "))

        if años < 5 or años > 70:
            print("⚠️ Por favor ingresa un valor entre 5 y 70 años")
            return

        print(f"📊 Configuración: +{años} años")

        # Procesar imagen
```

```
        img_resultado, filename = sistema.editar_edad_dalle2(imagen_path, años)

        if img_resultado and filename:
            print("\n" + "=" * 50)
            print("🎉 ¡PROCESO COMPLETADO EXITOSAMENTE!")
            print("=" * 50)

            # Preguntar si desea descargar
            descargar = input("\n¿Deseas descargar la imagen? (s/n): ")
            if descargar.lower() == 's':
                sistema.descargar_resultado(filename)

            print("\n✨ ¡Gracias por usar el sistema!")
        else:
            print("\n❌ El proceso no se completó correctamente")

    except KeyboardInterrupt:
        print("\n\n⚠️ Proceso interrumpido por el usuario")
    except Exception as e:
        print(f"\n❌ Error inesperado: {str(e)}")
        print("\n💡 Sugerencia: Verifica tu API key y conexión a internet")
```

```
# ============================================
# CELDA 5: EJECUTAR EL PROGRAMA
# ============================================

# Ejecutar el programa principal
ejecutar_progresion_edad()
```

```
Empieza a programar o a crear código con IA.
```

```
🚀 Inicializando sistema de progresión de edad...
✅ Sistema inicializado correctamente

📤 Por favor, selecciona una imagen de tu computadora:
```

Elegir archivos NiñoWero1.jpg

**NiñoWero1.jpg**(image/jpeg) - 17239 bytes, last modified: 17/9/2025 - 100% done

```
Saving NiñoWero1.jpg to NiñoWero1 (5).jpg
✅ Imagen 'NiñoWero1 (5).jpg' cargada correctamente
📐 Dimensiones originales: (612, 408)
📷 Formato original: JPEG
✅ Imagen guardada como PNG: original_image.png
✅ Verificación: Formato=PNG, Modo=RGB
```