

07 DE ENERO DE 2025

PROYECTO: ROTACIÓN DE IMÁGENES



PRESENTAN:

- AVILA ISLAS IRVING
- DEL CASTILLO AGUIRRE NANCY ASHANTI
- MARTINEZ GONZALES LUIS FERNANDO
- MOCTEZUMA SILVESTRE GRECIA

ÍNDICE

o o o o

- 1 Objetivo del proyecto
- 2 Descripción del proyecto
- 3 Arquitectura del sistema
- 4 Código
- 5 Conclusión

Objetivo del proyecto

o o o o

El proyecto tiene como propósito central:

El propósito del proyecto es diseñar e implementar un sistema interactivo que permita realizar transformaciones geométricas en imágenes, incluyendo rotaciones, escalados, reflexiones y traslaciones, utilizando operaciones matriciales.



Esto busca proporcionar una herramienta educativa y funcional para entender cómo las matrices pueden manipular imágenes en aplicaciones gráficas y de visión computacional.

Descripción del proyecto

Se desarrolló una aplicación web con el framework Flask que permite al usuario cargar una imagen, aplicar diversas transformaciones geométricas y visualizar los resultados en tiempo real.

Características del sistema

- Entrada del usuario

1. Entrada Manual:

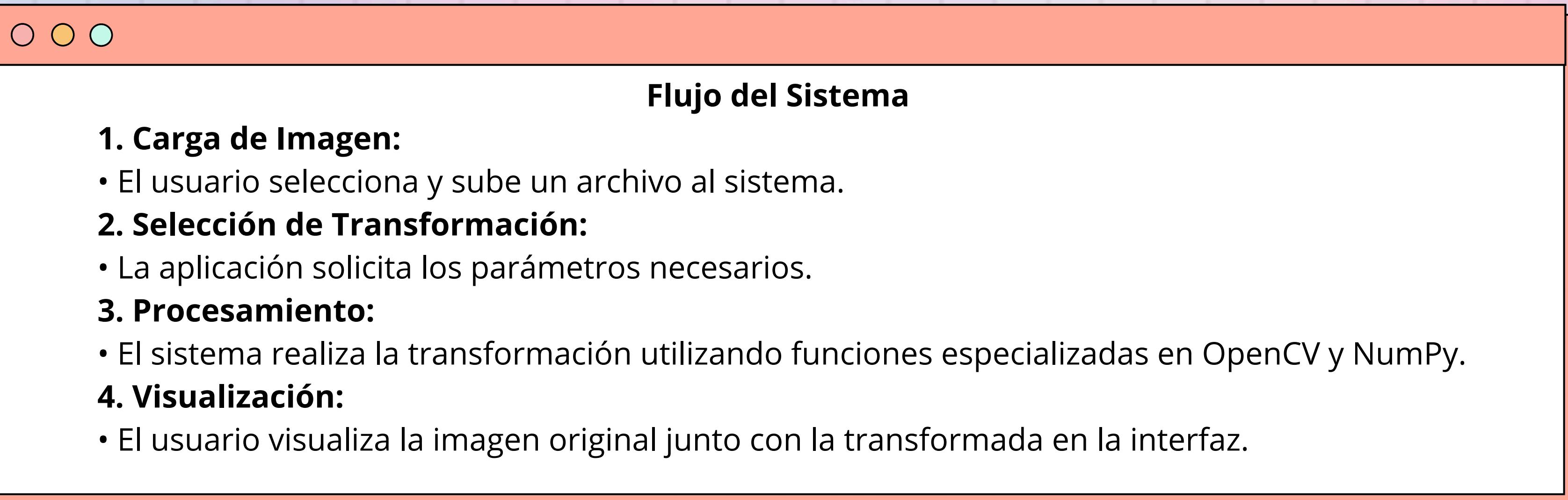
- El usuario puede cargar una imagen en formatos comunes como .jpg o .png.
- Seleccionar el tipo de transformación a aplicar (rotación, escalado, reflexión, traslación).
- Proveer parámetros específicos para cada transformación (e.g., ángulo de rotación, factor de escala).

2. Resultados Visuales:

- Mostrar la imagen original y la transformada en una interfaz web..

Arquitectura del sistema

○ ○ ○ ○



Flujo del Sistema

- 1. Carga de Imagen:**
 - El usuario selecciona y sube un archivo al sistema.
- 2. Selección de Transformación:**
 - La aplicación solicita los parámetros necesarios.
- 3. Procesamiento:**
 - El sistema realiza la transformación utilizando funciones especializadas en OpenCV y NumPy.
- 4. Visualización:**
 - El usuario visualiza la imagen original junto con la transformada en la interfaz.

Transformaciones Implementadas

1. Rotación:

- Calcula la matriz de rotación y ajusta el lienzo para evitar recortes.

2. Escalado:

- Permite aumentar o reducir el tamaño de la imagen según un factor.

3. Reflexión:

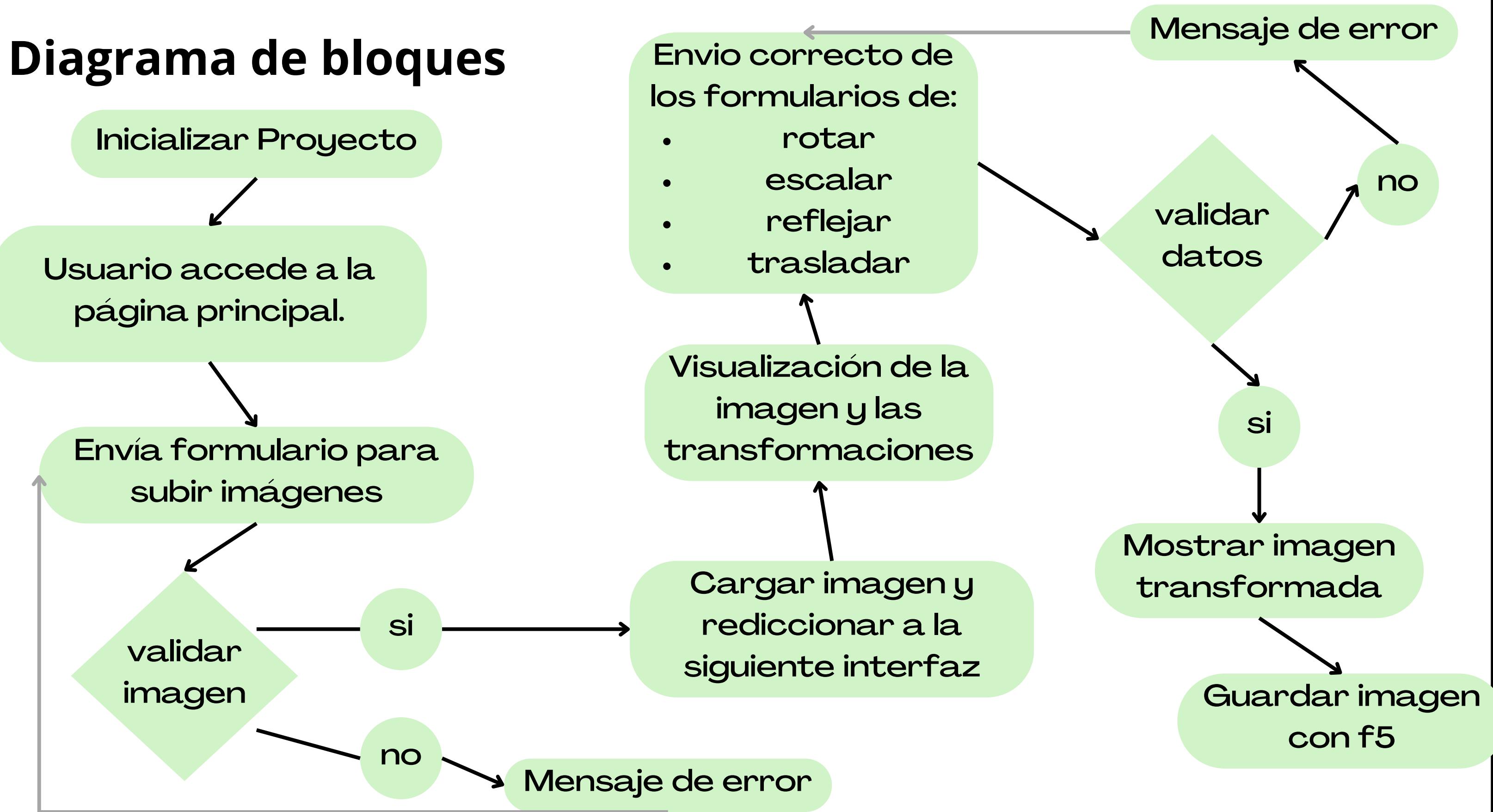
- Genera imágenes reflejadas en los ejes horizontal o vertical.

4. Traslación:

- Desplaza la imagen según un vector dado.

o o o o

Diagrama de bloques



Código

o o o o



El sistema está basado en Flask y utiliza librerías de visión computacional (OpenCV) y matemáticas (NumPy). Algunos aspectos destacados del código son:

1. Rotación con Lienzo Expandido:

Evita recortes al calcular dinámicamente el tamaño requerido del lienzo.

2. Validación de Entrada:

Se asegura de que los parámetros sean válidos antes de realizar la transformación.

3. Interfaz Visual:

Utiliza HTML y CSS para proporcionar una experiencia amigable al usuario.



```
1  from flask import Flask, render_template, request, redirect, url_for, send_from_directory  
2  import cv2  
3  import numpy as np  
4  import os
```

El sistema está basado en Flask y utiliza librerías de visión computacional (OpenCV) y matemáticas (NumPy). Algunos aspectos destacados del código son: 1. Rotación con Lienzo Expandido: Evita recortes al calcular dinámicamente el tamaño requerido del lienzo. 2. Validación de Entrada: Se asegura de que los parámetros sean válidos antes de realizar la transformación. 3. Interfaz Visual: Utiliza HTML y CSS para proporcionar una experiencia amigable al usuario.

```
6  app = Flask(__name__)  
7  UPLOAD_FOLDER = 'static/images/'  
8  app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
```

CONFIGURACIÓN INICIAL: Se crea una instancia de la aplicación Flask y se configura una carpeta para almacenar imágenes cargadas.

```
11 @app.route('/')  
12 def index():  
13     return render_template('index.html')
```

RTA DE LA PAGINA PRINCIPAL: Define la ruta principal (/) que renderiza la plantilla index.html.



```
16 @app.route('/upload', methods=['POST'])
17 def upload_image():
18     if 'file' not in request.files:
19         return redirect(request.url)
20     file = request.files['file']
21     if file.filename == '':
22         return redirect(request.url)
23     if file:
24         original_filename = file.filename
25         filepath = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
26         file.save(filepath)
27         return redirect(url_for('transform_image', filename=file.filename))
```

RUTA PARA SUBIR IMÁGENES: Maneja la carga de imágenes a través de un formulario POST. Guarda la imagen en el servidor y redirige a la página de transformación.



```
@app.route('/transform/<filename>', methods=['GET', 'POST'])
def transform_image(filename):
    filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
    image = np.array(Image.open(filepath).convert("RGB"))

    output_filename = 'output_' + filename
    output_path = os.path.join(app.config['UPLOAD_FOLDER'], output_filename)

    if request.method == 'POST':
        transform_type = request.form['transformation']
        if transform_type == 'rotate':
            angle = float(request.form['angle'])
            image = manual_rotate_image(image, angle)
        elif transform_type == 'scale':
            factor = float(request.form['factor'])
            image = manual_scale_image(image, factor)
        elif transform_type == 'reflect':
            axis = request.form['axis']
            image = manual_reflect_image(image, axis)
        elif transform_type == 'translate':
            dx = int(request.form['dx'])
            dy = int(request.form['dy'])
            image = manual_translate_image(image, dx, dy)

    output_image = Image.fromarray(image.astype('uint8'))
    output_image.save(output_path)

return render_template('transform.html', original_filename=filename, transformed_filename=output_filename)
```

RUTA PARA TRANSFORMAR IMÁGENES: Procesa y aplica transformaciones a la imagen cargada. Soporta rotación, escalado, reflexión y traslación basándose en datos del formulario.



```
61     @app.route('/static/images/<filename>')
62     def show_image(filename):
63         return send_from_directory(app.config['UPLOAD_FOLDER'], filename)
```

RUTA PARA MOSTRAR IMÁGENES: Muestra la imagen solicitada desde el directorio de carga.



```
def manual_rotate_image(image, angle):      You, 2 hours ago • Uncommitted changes
    angle_rad = np.deg2rad(angle)
    cos_val = np.cos(angle_rad)
    sin_val = np.sin(angle_rad)

    rows, cols, _ = image.shape

    #nuevas dimensiones
    diagonal = int(np.sqrt(rows**2 + cols**2))
    new_rows = new_cols = diagonal
    expanded_image = np.ones((new_rows, new_cols, 3), dtype=image.dtype) * 255 #blanco

    # Centrar imagen
    center_x, center_y = new_cols // 2, new_rows // 2
    offset_x, offset_y = (new_cols - cols) // 2, (new_rows - rows) // 2
    expanded_image[offset_y:offset_y + rows, offset_x:offset_x + cols] = image

    rotated_image = np.ones_like(expanded_image) * 255

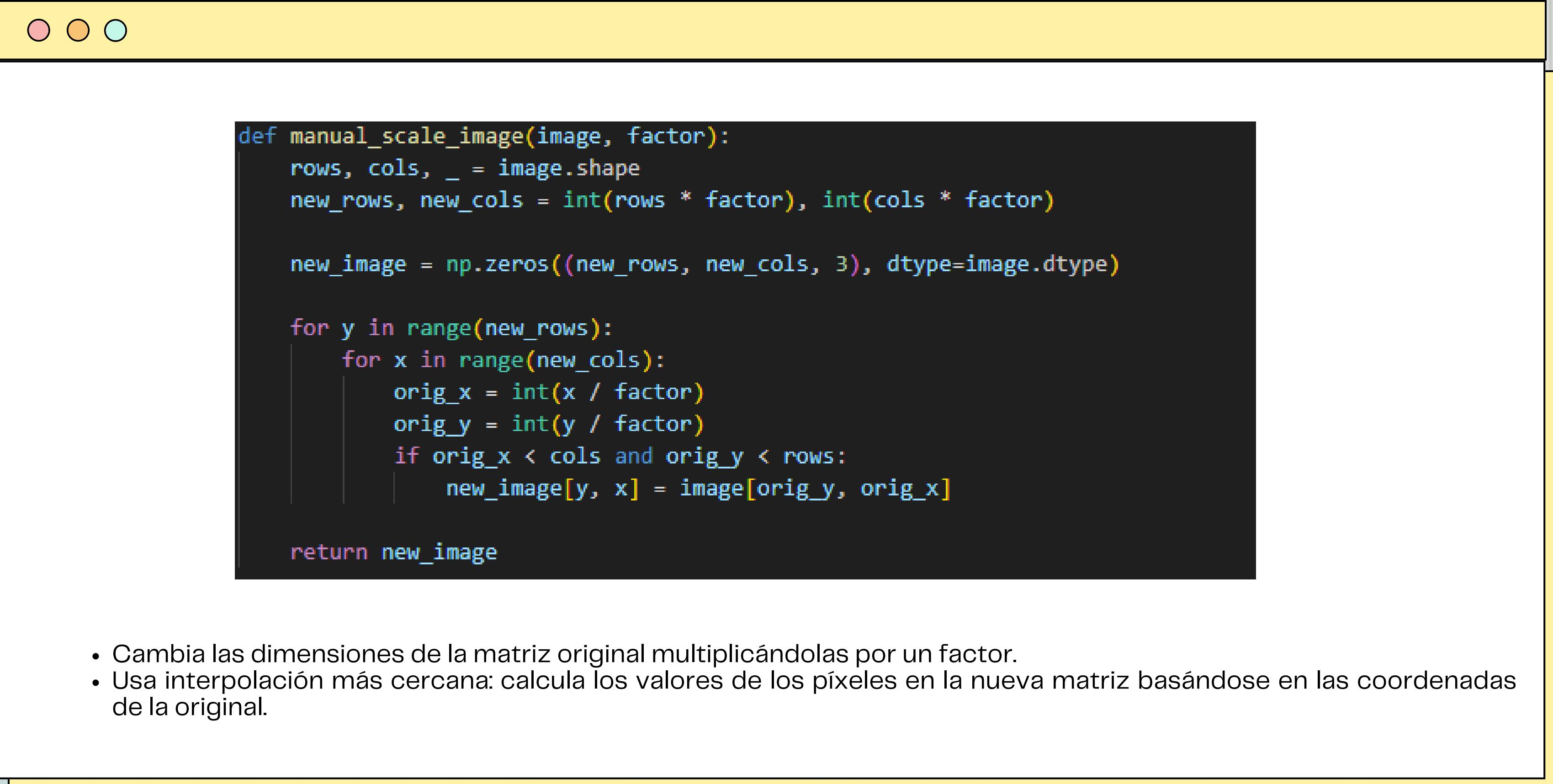
    # Aplicar la matriz de rotación
    for y in range(new_rows):
        for x in range(new_cols):
            x_shifted = x - center_x
            y_shifted = y - center_y

            orig_x = int(cos_val * x_shifted - sin_val * y_shifted + cols // 2)
            orig_y = int(sin_val * x_shifted + cos_val * y_shifted + rows // 2)

            if 0 <= orig_x < cols and 0 <= orig_y < rows:
                rotated_image[y, x] = expanded_image[orig_y + offset_y, orig_x + offset_x]

    return rotated_image
```

- Usa operaciones matemáticas (como seno y coseno) para calcular las nuevas posiciones de los píxeles después de rotar.
- Itera sobre cada píxel de la matriz y calcula su nueva ubicación.



```
def manual_scale_image(image, factor):
    rows, cols, _ = image.shape
    new_rows, new_cols = int(rows * factor), int(cols * factor)

    new_image = np.zeros((new_rows, new_cols, 3), dtype=image.dtype)

    for y in range(new_rows):
        for x in range(new_cols):
            orig_x = int(x / factor)
            orig_y = int(y / factor)
            if orig_x < cols and orig_y < rows:
                new_image[y, x] = image[orig_y, orig_x]

    return new_image
```

- Cambia las dimensiones de la matriz original multiplicándolas por un factor.
- Usa interpolación más cercana: calcula los valores de los píxeles en la nueva matriz basándose en las coordenadas de la original.



```
def manual_reflect_image(image, axis):
    rows, cols, _ = image.shape
    new_image = np.zeros_like(image)

    if axis == 'horizontal':
        for y in range(rows):
            for x in range(cols):
                new_image[y, cols - 1 - x] = image[y, x]
    elif axis == 'vertical':
        for y in range(rows):
            for x in range(cols):
                new_image[rows - 1 - y, x] = image[y, x]

    return new_image
```

Invierte las filas o columnas de la matriz según el eje seleccionado:

- Horizontal: Los valores de las columnas se intercambian.
- Vertical: Los valores de las filas se invierten.



```
def manual_translate_image(image, dx, dy):
    rows, cols, _ = image.shape

    #estetica
    expanded_rows = rows + abs(dy)
    expanded_cols = cols + abs(dx)
    expanded_image = np.ones((expanded_rows, expanded_cols, 3), dtype=image.dtype) * 255 # Fondo blanco

    offset_y = abs(dy) if dy < 0 else 0
    offset_x = abs(dx) if dx < 0 else 0
    expanded_image[offset_y:offset_y + rows, offset_x:offset_x + cols] = image

    #nueva imagen trasladada
    translated_image = np.ones_like(expanded_image) * 255 # Fondo blanco

    for y in range(expanded_rows):
        for x in range(expanded_cols):
            orig_x = x - dx
            orig_y = y - dy

            if 0 <= orig_x < cols + abs(dx) and 0 <= orig_y < rows + abs(dy):
                translated_image[y, x] = expanded_image[orig_y, orig_x]

    return translated_image
```

- Crea una nueva matriz más grande si es necesario para acomodar el desplazamiento.
- Copia los píxeles de la matriz original a nuevas posiciones según los valores dx (horizontal) y dy (vertical).



```
107 if __name__ == '__main__':
108     app.run(debug=True, port=5002)
```

EJECUCIÓN DE LA APLICACIÓN: Inicia la aplicación Flask en modo de depuración en el puerto 5002.

Conclusión

o o o o

Este proyecto muestra cómo las transformaciones geométricas pueden aplicarse de manera interactiva utilizando operaciones matriciales. Es un ejemplo práctico del uso de matemáticas en aplicaciones tecnológicas cotidianas, y su diseño modular facilita su ampliación para incluir nuevas funcionalidades.

Al finalizar se espera haber construido una herramienta robusta que facilite la comprensión del concepto de rotación de matrices aplicándolo en un contexto visual y práctico.

¡Gracias!

○ ○ ○ ○

Que tengas un gran
día por delante.