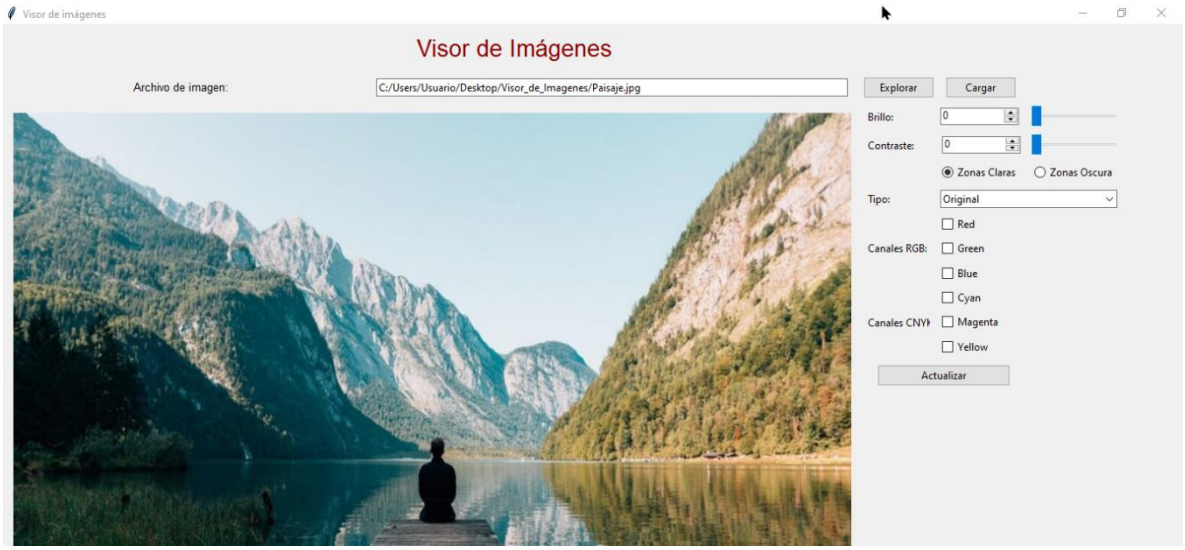
Proyecto: Desarrollo de un Visor de Imágenes en Python



Objetivo:

Desarrollar una aplicación de escritorio en Python utilizando la biblioteca tkinter o QT para la interfaz gráfica. El visor de imágenes permitirá al usuario cargar, visualizar, aplicar ajustes básicos a la imagen y realizar operaciones avanzadas como rotación, zoom, binarización, visualización del histograma y fusión de dos imágenes.

Requerimientos:

1. **Interfaz de Usuario:**
    - Crear una ventana principal con un título adecuado, p. ej., "Visor de Imágenes".
    - Incluir un campo de texto para mostrar la ruta del archivo de imagen cargado.
    - Botones:
        - **Explorar:** Abre un cuadro de diálogo para seleccionar una imagen del sistema de archivos.
        - **Cargar:** Carga la imagen seleccionada y la muestra en la ventana principal.

        - **Actualizar:** Aplica los ajustes seleccionados a la imagen mostrada.
        - **Guardar:** Permite guardar la imagen modificada con un nombre nuevo y en una ubicación diferente.
        - **Fusionar Imágenes:** Permite seleccionar una segunda imagen y fusionarla con la imagen principal cargada, aplicando una transparencia ajustable.
    - Controles deslizantes para ajustar:
        - **Brillo**: Aumentar o disminuir el brillo de la imagen.
        - **Contraste**: Ajustar el contraste de la imagen.
        - **Rotación**: Permite rotar la imagen desde 0° hasta 360°.
        - **Transparencia de Fusión**: Ajustar la transparencia de la segunda imagen (de 0% a 100%) para fusionar con la imagen principal.
    - Opciones de filtro de color:
        - **Zonas Claras** y **Zonas Oscuras**: Permitir la aplicación de un filtro para resaltar las zonas claras u oscuras de la imagen.
        - **Canales RGB**: Permitir al usuario seleccionar o desactivar los canales de color (Rojo, Verde, Azul) de la imagen.
        - **Canales CMY**: Permitir al usuario activar/desactivar los canales Cian, Magenta y Amarillo.
    - Funciones adicionales:
        - **Zoom:** Permitir seleccionar un área de la imagen utilizando coordenadas iniciales (X, Y) y ajustar el zoom sobre la región seleccionada.
        - **Binarizar Imagen:** Convertir la imagen a blanco y negro utilizando un umbral ajustable.
        - **Visualizar Histograma:** Mostrar el histograma de la imagen cargada con la distribución de cada canal de color.
        - **Negativo Imagen:** Saca el negativo de una imagen

        - **Fusión de Imágenes:** Permitir al usuario cargar una segunda imagen y combinarla con la imagen principal, aplicando un valor de transparencia ajustable.

2. **Funcionalidad:**
    - **Cargar y Mostrar Imágenes:**
        - Permitir al usuario cargar imágenes en formato JPG, PNG y BMP.
        - Mostrar la imagen cargada en el área de visualización de la ventana principal.
    - **Ajustes de Imagen:**
        - Aplicar las transformaciones de brillo y contraste utilizando la biblioteca que se está realizando en la clase.
        - Permitir la rotación de la imagen desde 0° hasta 360°.
        - Implementar un zoom sobre la imagen a partir de un punto inicial (X, Y) y un factor de escala determinado por el usuario.
        - Binarizar la imagen aplicando un umbral para convertirla en blanco y negro.
    - **Fusión de Imágenes:**
        - Permitir la carga de una segunda imagen desde el sistema de archivos.
        - Ajustar la transparencia de la segunda imagen y fusionarla con la imagen principal.
        - La transparencia se ajustará utilizando un control deslizante, variando desde 0 (completamente opaca) hasta 1 (completamente transparente).
    - **Visualización del Histograma:**
        - Generar y mostrar un histograma que muestre la distribución de los canales de color (R, G, B) y la intensidad de luminosidad.
    - **Actualización Dinámica:**
        - Actualizar la vista de la imagen cada vez que se realice un cambio en los controles.

3. **Detalles de Implementación:**
    - Usar tkinter o Qt para la creación de la interfaz gráfica.
    - Usar matplotlib para la generación y visualización del histograma.
    - Seguir buenas prácticas de programación, separando la lógica de la interfaz de usuario de la manipulación de imágenes.
4. **Criterios de Evaluación:**
    - La interfaz gráfica es intuitiva y permite al usuario realizar las operaciones descritas.
    - La aplicación es capaz de cargar y mostrar imágenes sin errores.
    - Los ajustes de brillo, contraste, rotación y zoom se aplican correctamente a la imagen.
    - Los filtros de color y binarización se aplican de manera adecuada y producen los resultados esperados.
    - La fusión de imágenes se realiza de manera correcta y visualmente coherente, de acuerdo con el valor de transparencia definido por el usuario.
    - El histograma se genera y muestra correctamente en una ventana adicional o en un área designada dentro de la aplicación.
    - Se incluyen comentarios y documentación en el código para facilitar la comprensión del mismo.
5. **Recomendaciones:**
    - Utilizar tkinter.Frame para organizar los elementos de la interfaz gráfica.
    - Crear clases separadas para manejar la lógica del procesamiento de imágenes y la interfaz de usuario.
    - Probar la aplicación con diferentes tipos y tamaños de imágenes para asegurar su funcionamiento.
    - Implementar la funcionalidad de zoom de manera que permita ajustar tanto la región inicial como el factor de escala.

Funcionalidades Avanzadas (Opcionales):

- Implementar la funcionalidad de **Deshacer Cambios** para restaurar la imagen a su estado original.
- Añadir un botón de **Restaurar** para devolver la imagen a sus valores por defecto (sin ajustes aplicados).
- Incluir la opción de **Rotación Libre**, donde el usuario puede rotar la imagen con el ratón o utilizando el teclado.
- Permitir seleccionar áreas con el ratón para aplicar el zoom dinámicamente.
- Implementar la **fusión de imágenes con diferentes modos de superposición** (e.g., multiplicación, adición, diferencia).