**VISUALIZADOR DE IMÁGENES**

Implementar una interfaz gráfica que permita la carga y visualización de imágenes. Esta debe tener la opción aplicar algún filtro sobre la imagen cargada.

El diseño de la interfaz puede ser como el que se muestra a continuación:

Graphical user interface, application

Description automatically generated

**Consideraciones:**

* Los filtros que se vayan a aplicar sobre la imagen se pueden implementar del módulo *scipy.ndimage.*
* El botón “Filtros” solo se puede usar cuando se ha cargado una imagen.
* La ventana que se abre cuando se oprime el botón de “Filtros” no bloquea la funcionalidad de la ventana principal.

**Solución:**

Para el desarrollo de la interfaz gráfica se empleo Qt Designer para el diseño de la interfaz. En este se promovió la clase Graphics View a ImageView, esto se realizó como se mostró en el ejercicio anterior. El diseño que se obtuvo de la ventana principal fue:

Graphical user interface, application, Word

Description automatically generated

El botón Filtros se selecciona que esté por defecto desactivado. Esto porque no hay imagen cargada.

Con respecto a la ventana de diálogo para aplicar los filtros, este fue su diseño:

Graphical user interface, application

Description automatically generated

En esta se selecciona que sea no modal, esto con el fin de que no interrumpa la ejecución de la ventana principal.

En este caso solo se explicará la lógica de interfaz de usuario. No se mostrarán las líneas de código que se suelen repetir cuando se desarrollan aplicaciones en PyQt.

**Métodos constructores de las ventanas involucradas:**

|  |  |
| --- | --- |
|  |  |

Con respecto a la ventana principal, además de los atributos asociados a los widgets dentro de esta, se define también el atributo ruta, el cual se encarga de almacenar la ruta que contiene la imagen seleccionada. Además, se establece un atributo llamado imgCargada, el cual se encarga de decirle a la clase si se encuentra ingresada, o no, una imagen.

A los dos botones que se tienen, se les asocia el evento de clickear con sus respectivos métodos. Al botón de cargar imagen se le asigna el método graficar, y el botón de abrir Filtros el método abrirFiltros.

Con respecto a la ventana de diálogo se debe tener en cuenta que se debe alguna forma de comunicación entre esta y la ventana principal. En este caso se empleó un método que no es recomendado cuando la aplicación es muy compleja. En este caso se asocia la ventana de diálogo (Filtros) con la ventana principal por medio de un atributo, de manera que cuando se genera la ventana de diálogo (Filtros) esta recibe como argumento la clase desde dónde se genera, asignando este argumento entonces al atributo ventanaPrincipal. Una forma de evitar este tipo de prácticas es usando arquitecturas de software como lo es el modelo-vista-controlador (MVC).

Con respecto a los eventos de cada botón de esta ventana, se asocian a diferentes métodos propios de la ventana de diálogo. En estos métodos se llama a una determinada funcionalidad de la ventana principal. Esto ser verá a continuación.

**Carga de imagen**



Para cargar y asignar la imagen al objeto ImageView se llama el método graficar. En este lo primero que se ejecuta es lo que está en verde, que es el método que se encarga de la carga de la imagen. Para esto se usa un widget llamado QFileDialog, el cual por medio del método getOpenFileName ejecuta una ventana que permite al usuario seleccionar el archivo del que se desea capturar su ruta. Esta ruta es el primer elemento que retorna este método, el reto de atributos no importa, por lo que se usa ‘\_’ para no tenerlos en cuenta.

En función de lo que haya, o no, seleccionado el usuario se evalúa un condicional. En este, primero se verifica si el usuario sí seleccionó un archivo; si no lo hizo, no pasa nada y el código que está resaltado en azul no se ejecuta. Ahora bien, si el usuario seleccionó un archivo se verifica si este permite cargar una imagen, esto se hace por medio de la estructura try y llamando la función que permite leer imágenes ofrecida por matplotlib. Si se genera alguna excepción significa que el nombre del archivo no captura una imagen, entonces se le muestra al usuario una ventana de alerta mencionando que “ingrese una imagen”. Si la ruta sí contiene una imagen, esta se almacena en el atributo img y el atributo que valida que haya una imagen cargada toma el valor de True.

Al haber una imagen cargada se ejecuta el código resaltado en azul del método graficar, en este se activa el botón que permite llamar a la ventana de diálogo de los filtros y luego se grafica la imagen en el objeto ImageView, esto por medio del método setImage, el cual grafica la matriz que tiene una imagen asociada. Se establecen el orden de los ejes con el segundo argumento del método, esto para que la imagen sea cargada correctamente y no boca abajo.

**Aplicación de Filtros**

****

|  |  |
| --- | --- |
| **Métodos Filtros ventana Principal** | **Métodos Filtros ventana de Diálogo** |

Cuando se decide aplicar los filtros sobre la imagen cargada se debe abrir la ventana de diálogo, cuando sucede eso se invoca el método abrirFiltros, en este se genera la ventana y se le asocia como atributo de la ventana principal. La generación de la ventana Filtros recibe como argumento self, el cual es la palabra que hace referencia a toda la funcionalidad de la ventana principal, esto permitirá que desde la ventana de diálogo sea posible invocar los métodos que se encuentran en la izquierda.

Por lo que el orden de aplicación de un filtro sería: Al oprimirse un botón de la ventana de diálogo este llama a un método de esta. Por ejemplo, el botón Filtro 1 llama el método appFilt1 de la ventana de diálogo (Código de la derecha resaltado en amarillo), dentro de esta se ejecuta el método appFilt1 pero del atributo ventanaPrincipal (Código resaltado de la izquierda). Como este último contiene toda la funcionalidad de la ventana principal, entonces se llama el método appFilt1 de la izquierda, en este ya se genera la operación sobre el objeto ImageView, el cual es aplicar un filtro gaussiano con desviación estándar de 10. Esto último se realiza por medio del paquete ndimage de Scipy.

La documentación de los diferentes filtros aplicados se encuentra en los siguientes links:

* <https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.gaussian_filter.html#scipy.ndimage.gaussian_filter>
* <https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.maximum_filter.html#scipy.ndimage.maximum_filter>
* <https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.uniform_filter.html#scipy.ndimage.uniform_filter>

**Funcionalidad de la aplicación:**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
| Esta última muestra la funcionalidad que viene por defecto en la clase ImageView. Es posible interactuar con la imagen haciendo zoom o moviéndose sobre esta. Además, esta clase tiene implementada la funcionalidad de la derecha que permite cambiar tonos oscuros y claros de la imagen. | |