Universidad Nacional Autónoma de México

Facultad de Ciencias

Proyecto 2

Modelado y Programación

Johann Ramón Gordillo Guzmán - 418046090

José Jhovan Gallardo Valdez - 310192815

Proyecto presentado como parte del curso de **Modelado y Programación** impartido por el profesor **José de Jesús Galaviz Casas**.

07 de Septiembre del 2019

Link al código fuente: https://github.com/JohannGordillo/

1. Definición del Problema

El cliente, el diseñador gráfico Alexander Rybak, CEO de la empresa **RED & BLUE** nos ha solicitado elaborar un programa para poder aplicar filtros de colores a sus imagenes, además de poder ver los cambios directamente al aplicar los filtros.

El cliente necesita cuatro filtros diferentes:

- Filtro Rojo.

- Filtro Azul.

- Filtro Verde.

- Filtro Mosaico.

En el filtro mosaico nos pide dividir la imagen en casillas, y obtener el color promedio de la misma, para posteriormente aplicarle ese color a toda la casilla.

2. Análisis del Problema

1. **Requisitos Funcionales**

   Dada una imagen, se debe mostrar en una interfaz gráfica la misma, y cuatro botones botones para que al presionar cada uno de ellos se apliquen los cuatro filtros de colores requeridos por el usuario: Rojo, Verde, Azul y el filtro mosaico. Los cambios deben ser visibnles al usuario en la mencionada interfaz gráfica.

   Se debe llevar a cabo la obtención de dichas imagenes del sistema de archivos del usuario.

2. **Requisitos No Funcionales**

   El cliente como diseñador gráfico se supone capacitado para manejar el sistema de archivos de sus computadora, o por lo menos capaz de encontrar una imagen guardada por el mismo u otros usuarios. Por lo que se implementará una interfaz gráfica donde uno de los botones le permitirá acceder a dicho sistema de archivos y seleccionar la imagen a editar. Dicha interfaz será **amigable** y sencilla de usar para el diseñador. Permitirá seleccionar la imagen a editar y aplicar los filtros de colores de una manera sencilla e intuitiva.

   El programa deberá poseer **tolerancia a fallas**, pues el diseñador provee sus servicios de edición de imagenes a clientes exigentes y quiere que éstos estén satisfechos con el trabajo realizado por el diseñador.

   El programa será **escalable**, pues posteriormente se podrán implementar filtros de más colores y más funcionalidades dentro de la interfaz gráfica del proyecto.

   Se cumplirá con la **eficiencia** requerida por el programador, asegurándose el programa de que los filtros sean aplicados a las imagenes de una manera rápida y los cambios en la imagen puedan ser mostrados al usuario de la misma manera.

De una manera más particular, podemos dividir hacer el analísis del problema de la siguiente manera:

2.1. Entrada

Una imagen, en formato .jpg. La imagen se seleccionará del directorio de archivos del usuario con ayuda de una interfaz gráfica (GUI) amigable e intuitiva.

1. **Formato**
   Imagenes con formato .jpg.

2. **Tamaño**
   El cliente no ha restringido el tamaño de la imagen.

3. **Cantidad**
   Una imagen.

4. **Fuente**
   Directorio de archivos del usuario.

2.2. Salida

Se mostrará en la interfaz gráfica la imagen seleccionada por el usuario, con el filtro seleccionado por el mismo, aplicado en la imagen.

1. **Formato**
   Una imagen en formato .jpg en la interfaz gráfica.

2. **Cantidad**
   Una imagen ya editada.

3. **Destino**
   La sección destinada a la imagen en la interfaz gráfica.

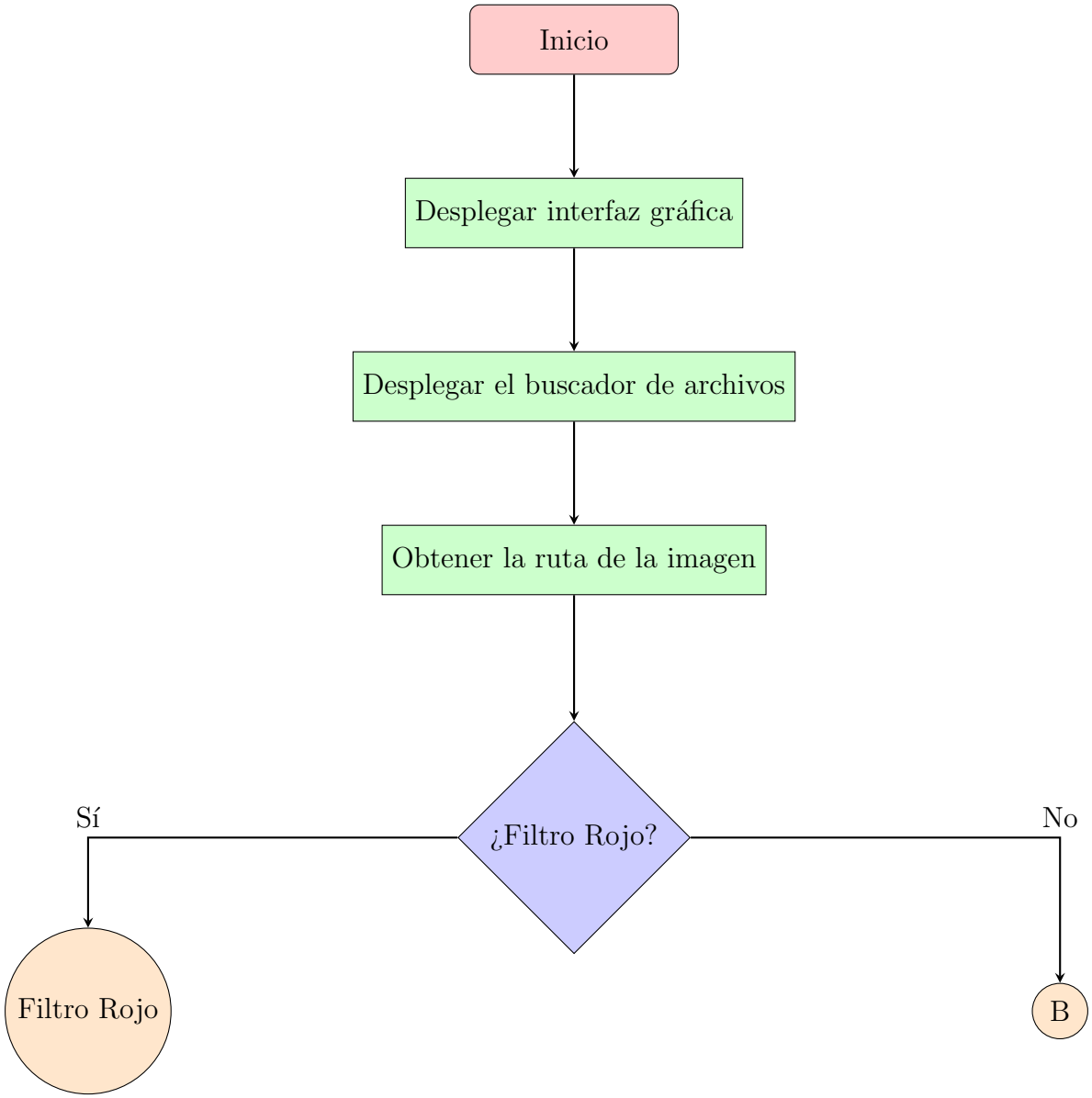
3. Selección de la mejor alternativa

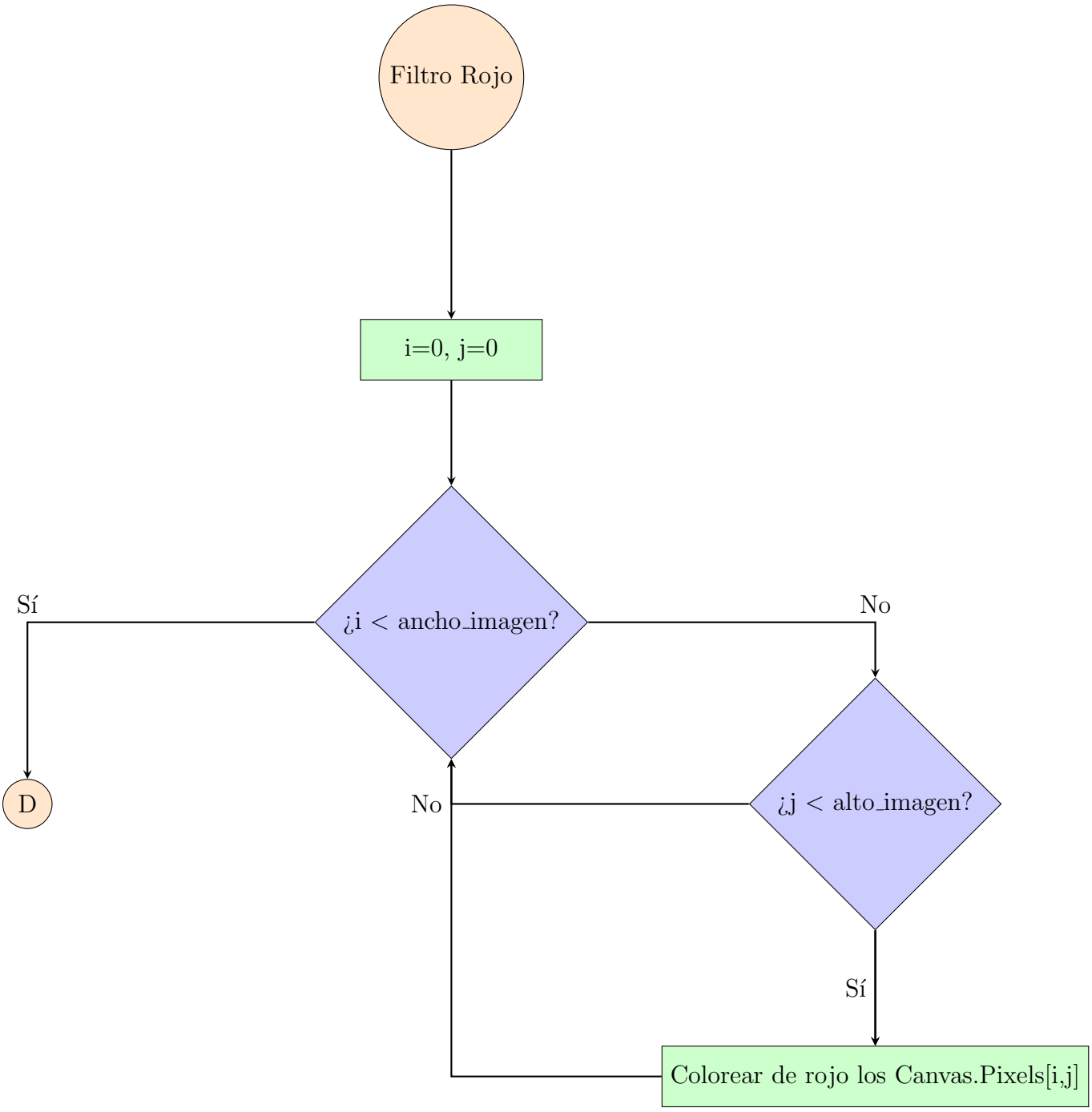
Se ha decidido usar el lenguaje de programación **Delphi** para la elaboración del proyecto, sobre otras alternativas. Delphi es un lenguaje muy fácil de usar basado en Pascal, que nos permitirá crear interfaces graficas de una manera extremadamente sencilla, pues basta con arrastrar las componentes (botones, menús, espacios para imagenes, etc) a un formulario. Además de que nos permite usar un paradigma de **programación orientado a objetos** y al mismo tiempo nos proveé la posibilidad de utilizar funciones.

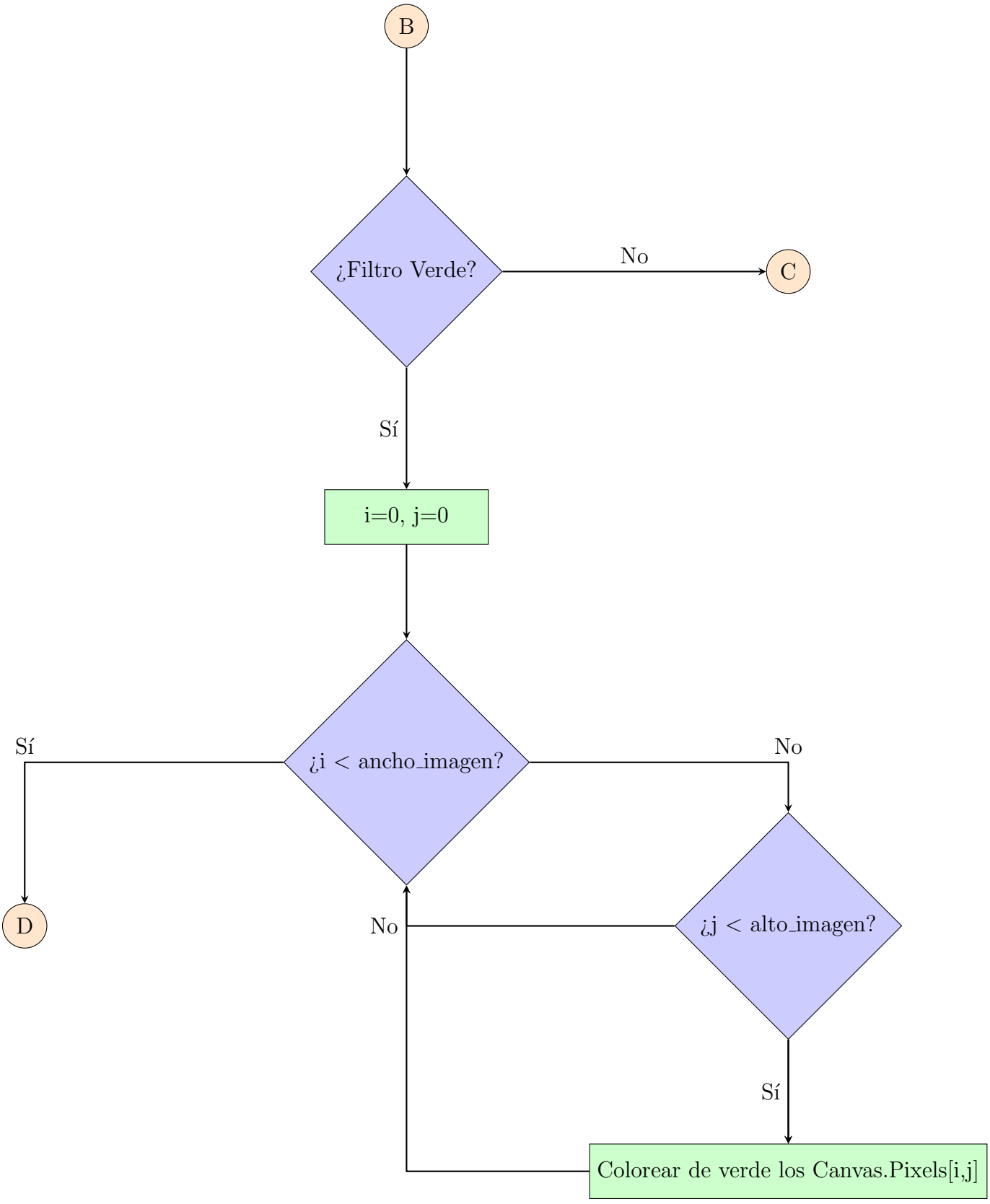
El modelo con el cual trabajar elegido se ajusta a los requisitos funcionales y al esbozo general de la solución y el programador no ve una mejor solución por el momento. Sin embargo, si en algún futuro encuentra una mejor manera de hacerlo, notificará al cliente de la misma.
[**véase Mantenimiento**].

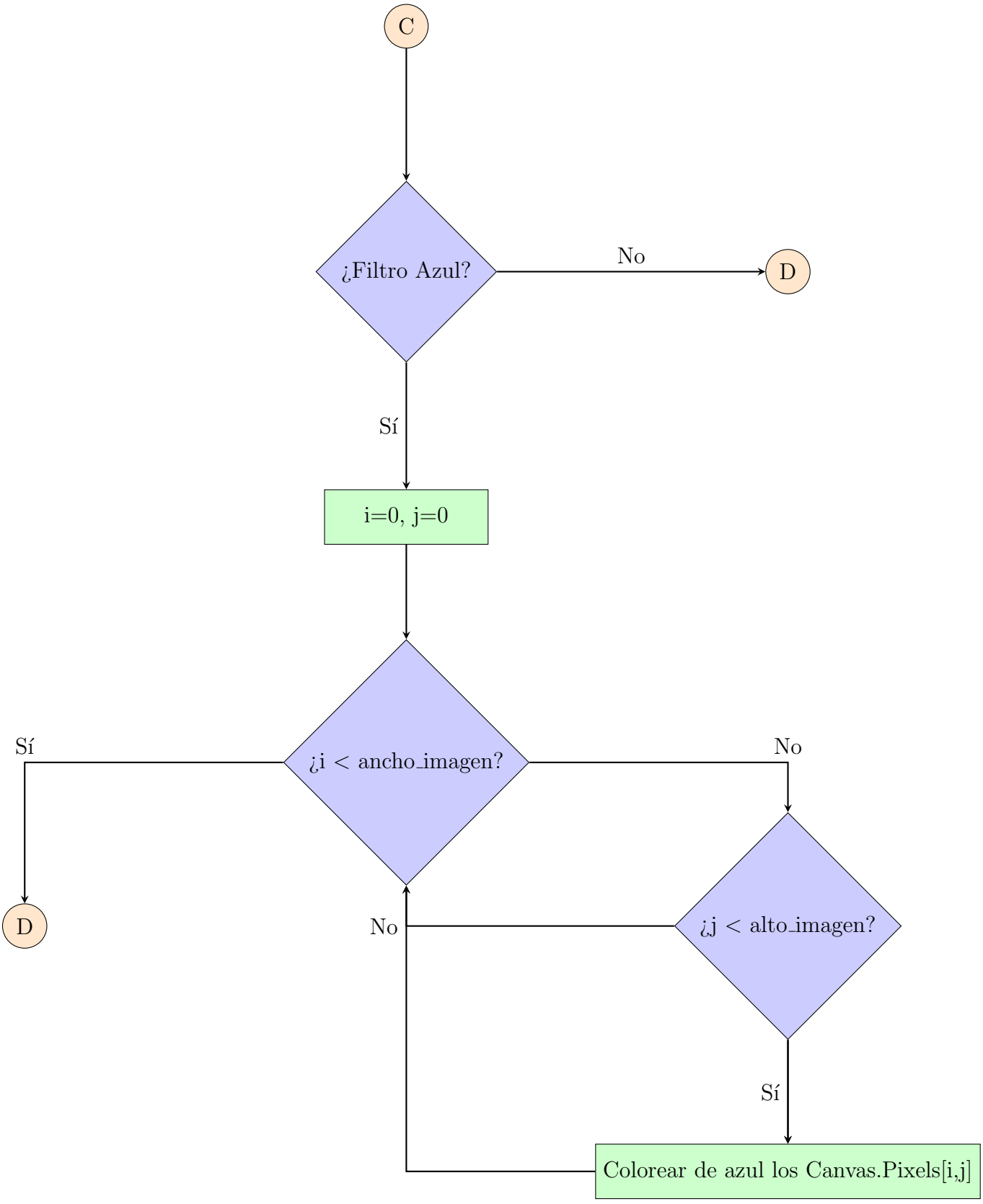
Como se mencionó anteriormente en este documento, se utilizará **interfaz gráfica**. Para ello, se ha elegido usar el diseñador de interfaces gráficas incluido en la IDE **Lazarus** sobre otras alternativas, por su facilidad de uso y su excelente documentación.

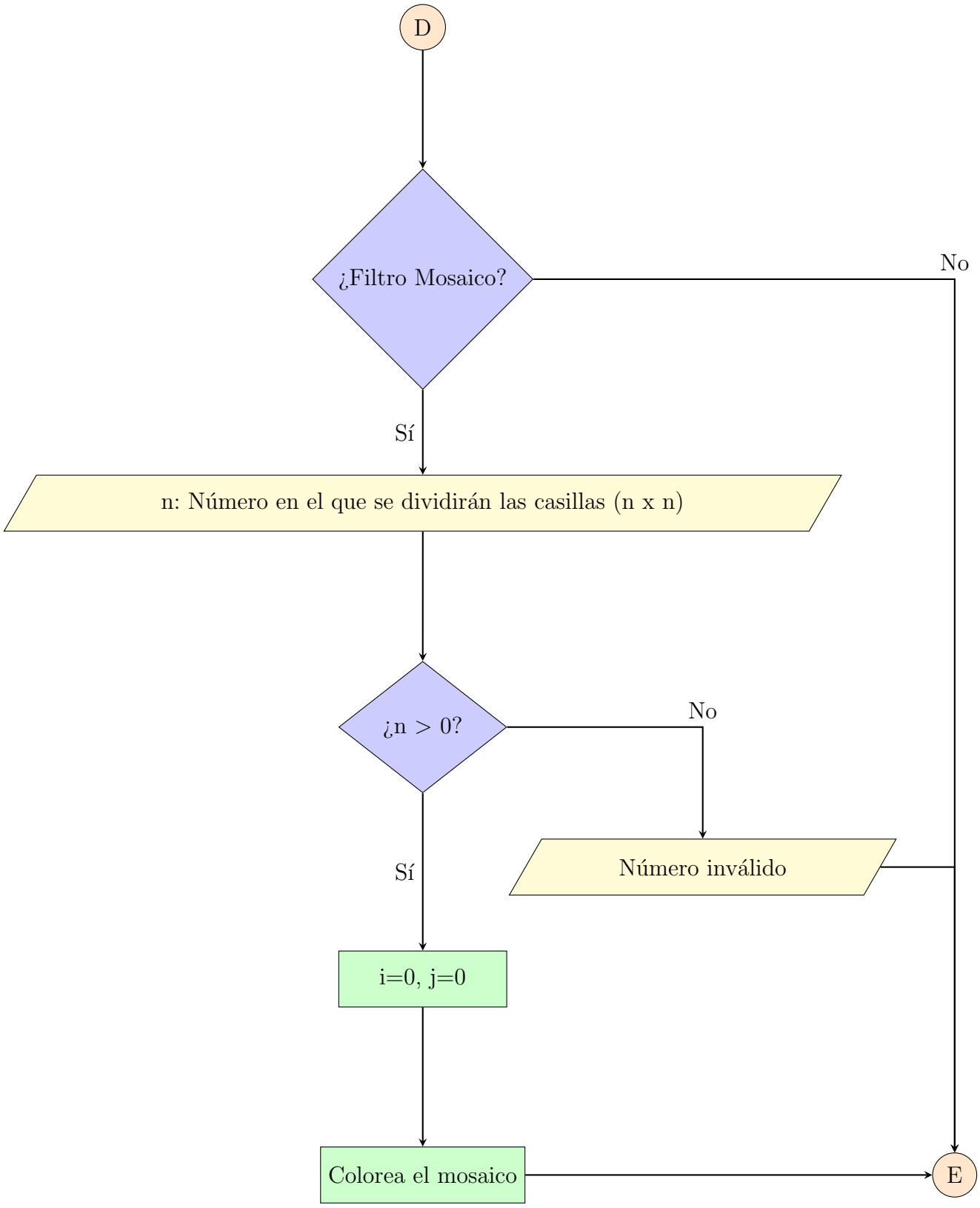
4. Diagrama de Flujo

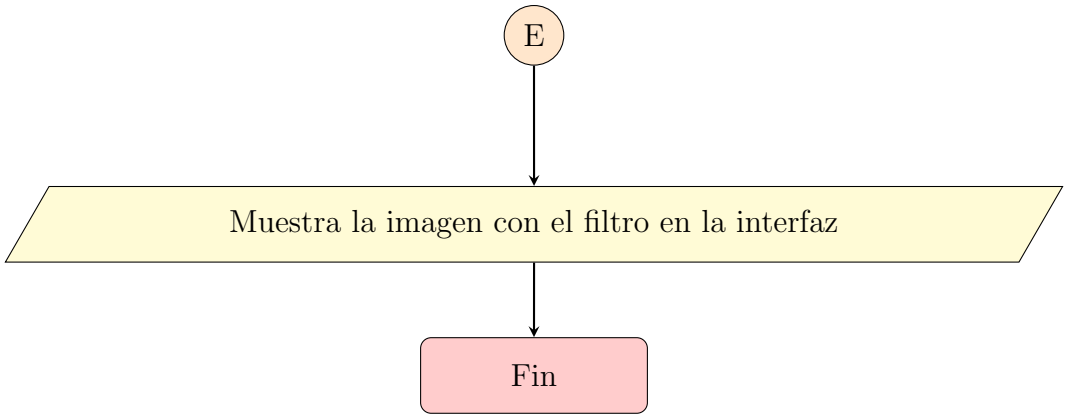



Filtro Rojo
i=0, j=0
¿i < ancho_imagen?
Sí
No
D
No
¿j < alto_imagen?
Sí
Colorear de rojo los Canvas.Pixels[i,j]


B
¿Filtro Verde?
No
C
Sí
i=0, j=0
¿i < ancho_imagen?
Sí
No
D
No
¿j < alto_imagen?
Sí
Colorear de verde los Canvas.Pixels[i,j]


C
¿Filtro Azul?
No
D
Sí
i=0, j=0
¿i < ancho_imagen?
Sí
D
No
¿j < alto_imagen?
No
Sí
Colorear de azul los Canvas.Pixels[i,j]


D
¿Filtro Mosaico?
No
Sí
n: Número en el que se dividirán las casillas (n x n)
¿n > 0?
No
Sí
Número inválido
i=0, j=0
Colorea el mosaico
E



4.1. Explicación

1. Iniciamos la ejecución del programa.

2. Primero mostramos al usuario una amigable interfaz gráfica, donde se incluyan botones para los filtros Azul, Rojo, Verde y Mosaico. Además del botón que permita al usuario buscar la imagen a editar.

3. El siguiente paso es obtener ruta de la imagen, para ello se le mostrará al usuario un gestor de archivos.

4. Ahora esperamos a que el usuario aplique algún filtro presionando un botón. Si el usuario presiona el botón de filtro rojo, pasa a «**5**». En caso contrario, pasa a «**8**».

5. Inicializamos las variables i=0 y j=0.

6. Si i es menor al ancho de la imagen a procesar, pasa a «**7**». En caso contrario, pasa a «**13**».

7. Si j es menor a la altura de la imagen a procesar, coloreamos de rojo los Canvas.Pixels[i,j]. Luego, regresamos a «**6**».

8. Si el usuario presiona el botón para aplicar el filtro verde, repetimos los pasos de «**5**» a «**7**» pero coloreamos de verde los Canvas.Pixels[i,j]. En caso contrario, pasa a «**9**»

9. Si el usuario presiona el botón para aplicar el filtro azul, repetimos los pasos de «**5**» a «**7**» pero coloreamos de azul los Canvas.Pixels[i,j]. En caso contrario, pasa a «**10**»

10. Si el usuario presiona el botón para aplicar el filtro mosaico, pedimos el número n en el cual dividir la imagen en n x n mosaicos.

11. Si n es menor o igual a cero, lanzamos un mensaje de error y terminamos.

12. Coloreamos uno por uno los mosaicos obteniendo el color promedio de cada uno de ellos. Cuando se hayan coloreado todos los mosaicos pasa a «**13**».

13. Mostramos la imagen en la interfaz gráfica.

14. Finalizamos el programa.

4.2. Descripción del trabajo de equipo

El proyecto fue largo y complicado de desarrollar, por lo que los desarrolladores se repartieron el trabajo de la siguiente manera:

Filtro Rojo - Jhovan.

Filtro Mosaico - Jhovan.

Filtro Azul - Johann.

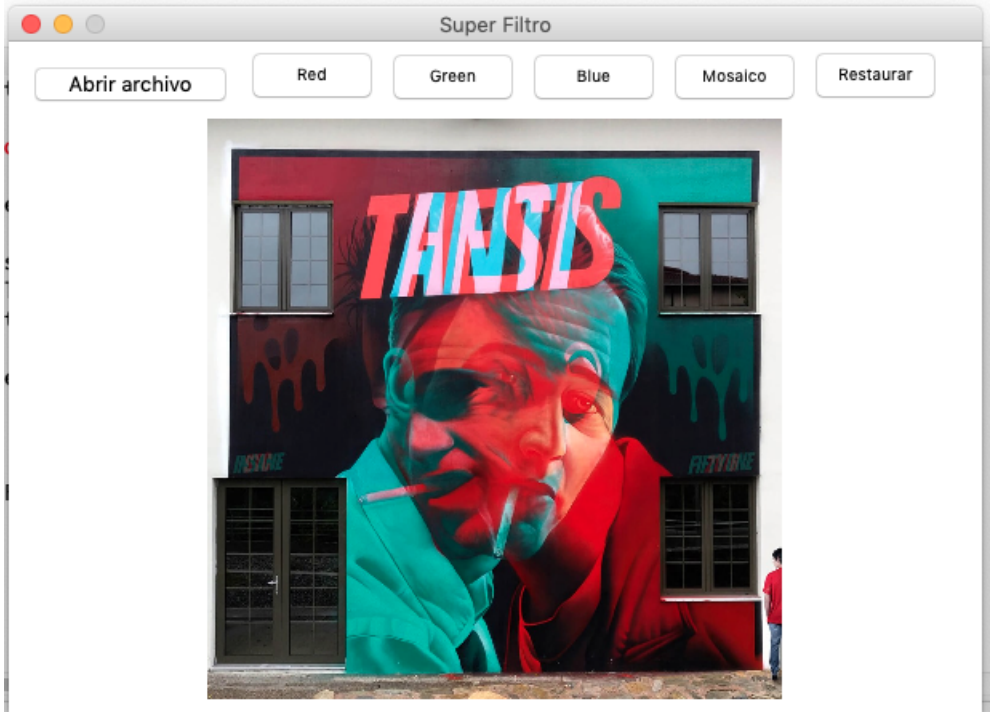
Filtro Verde - Johann.

Interfaz Gráfica - Jhovan y Johann.

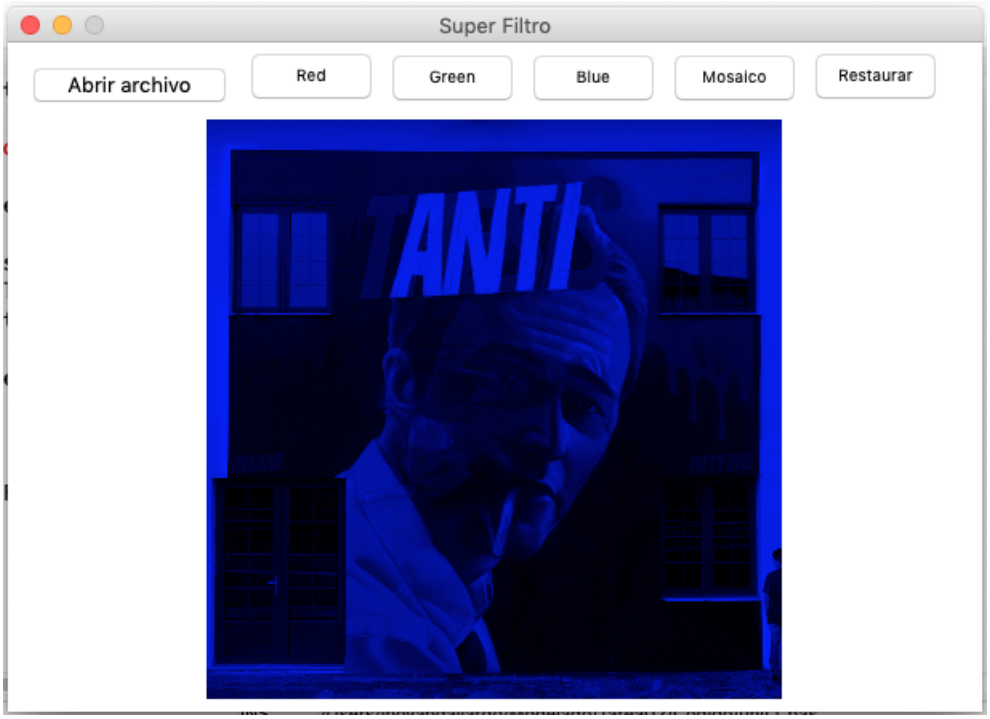
Pruebas - Jhovan y Johann.

4.3. Pruebas

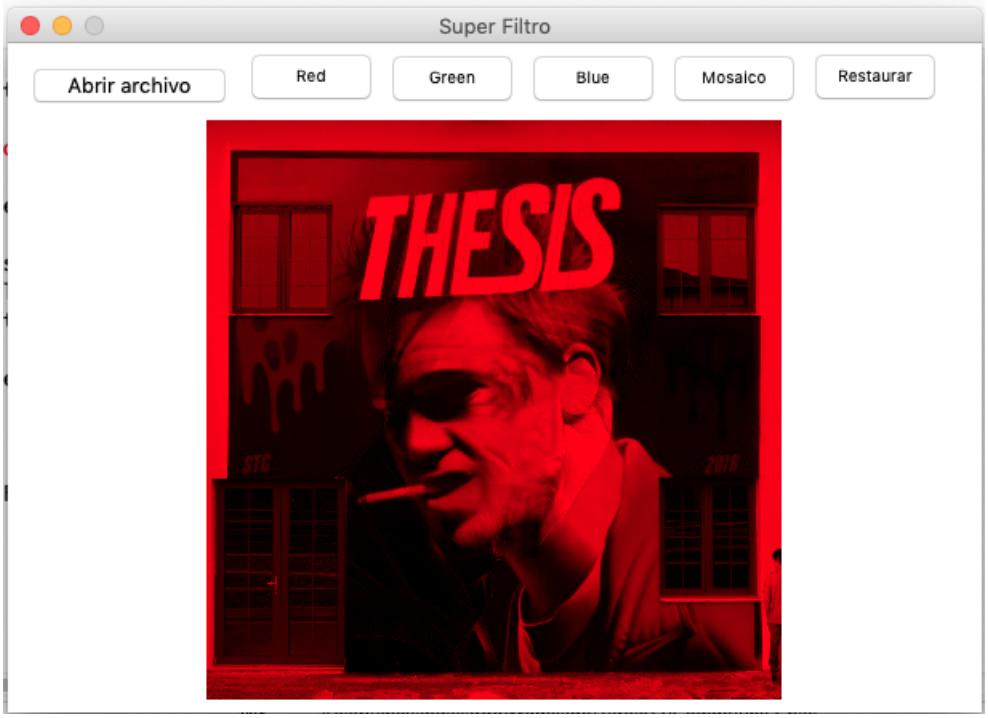
El programa funciona sin problemas y a continuación se presentan pruebas de su correcto funcionamiento:
Funcionalidad para seleccionar una imagen:



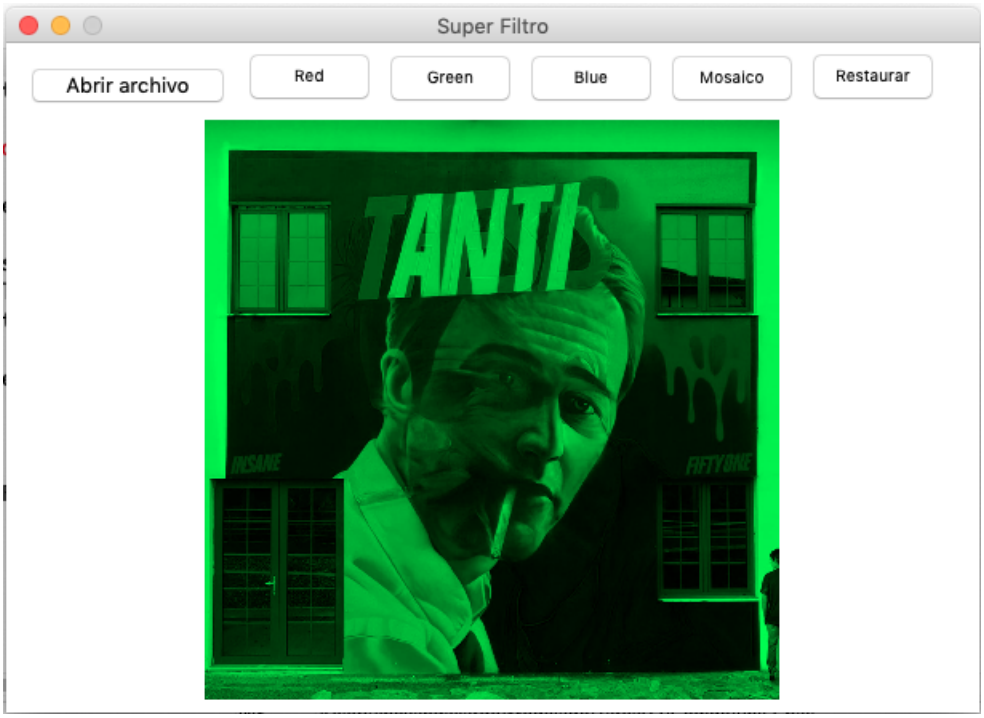
Al aplicar el Filtro Azul:



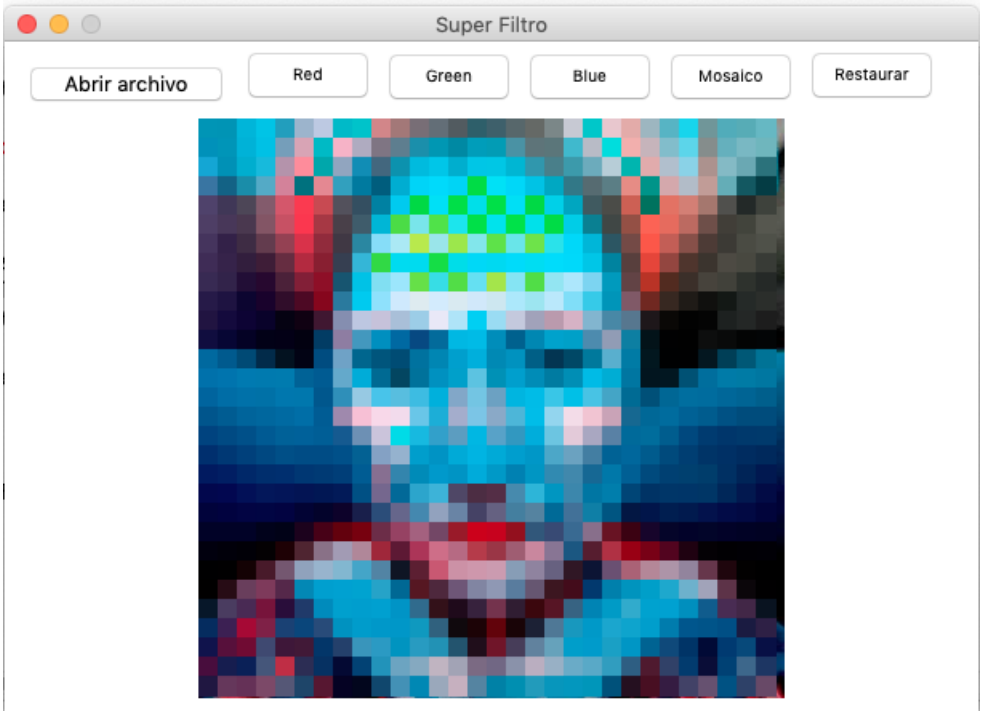
Al aplicar el Filtro Rojo:



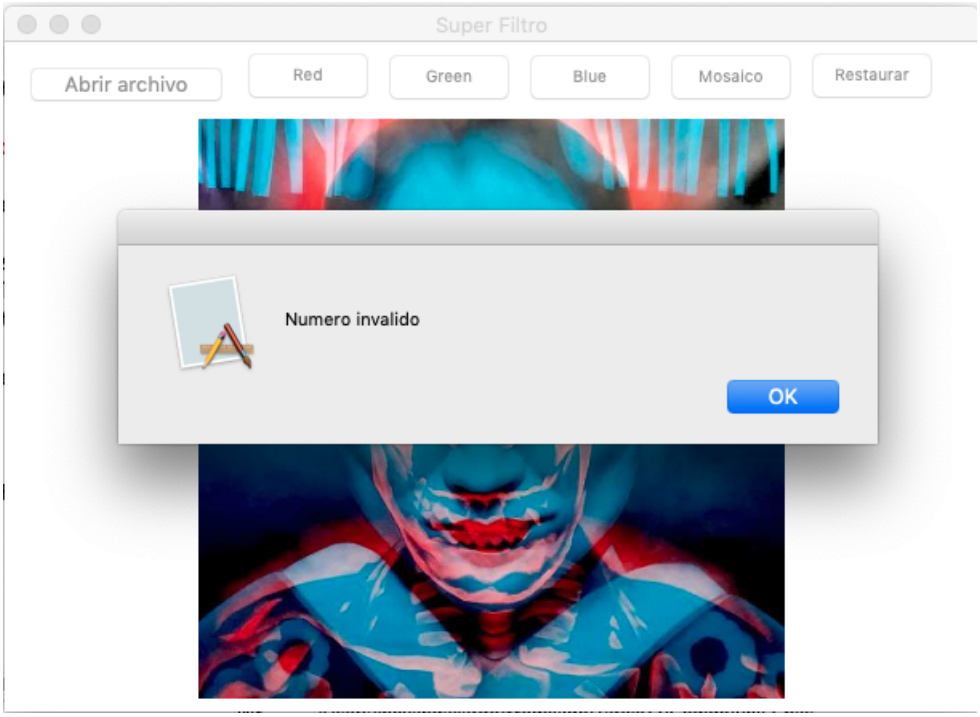
Al aplicar el Filtro Verde:



Al aplicar el Filtro Mosaico:



Prueba de que el Filtro Mosaico lanza una excepción al ingresar un número inválido para dividir la imagen en casillas n x n:



5. Sobre el mantenimiento del código

5.1. Nuevas funcionalidades

El programa solicitado por el cliente necesita únicamente cuatro filtros aplicables a imagenes de entrada: un filtro color rojo, uno verde, uno azul y un filtro de mosaico, en el que la imagen se divide en casillas y cada casilla se pinta con el color promedio de esa misma casilla; pero el equipo de desarrolladores está consciente de que se pueden realizar más filtros, por lo que si el cliente así lo desea, se añadirán los mismos al proyecto.

5.2. Bugs

Ningún programa es libre de bugs, y el equipo de desarrolladores está conciente de ello, por lo que ante descubrimientos de bugs y errores en el código, los programadores se encargarán de corregirlos.

5.3. Eficiencia

Si los desarrolladores descubren una manera más eficiente de implementar la solución al problema, notificará al cliente para estimar el costo de nuevas implementaciones. Por ejemplo, si se encuentra un algoritmo que permita reducir el tiempo en el que se aplica un filtro a una imagen cualquiera.

5.4. Escalabilidad

Si el cliente necesita que se editen más imagenes al mismo tiempo, esto será posible de realizar, pues el programa se ha realizado de tal manera que sea escalable.

6. Costo final del proyecto

El costo del proyecto fue calculado con base en las horas que el equipo de desarrolladores del mismo tuvo que invertir para verificar que la solución al problema planteado por el usuario, **RED & BLUE**, fuera eficiente; y el código de la implementación, robusto.

Se implementaron pruebas para que en un futuro, desarrolladores que trabajen en el proyecto puedan implementar nuevas funcionalidades y aún así verificar si el código sigue funcionando correctamente bajo ciertos casos planteados estrategicamente por los desarrolladores.

El costo final es de **$3,000.00** pesos mexicanos.

Para aclaraciones del proyecto y del costo del mismo contactar con
jhovan.gallardo@ciencias.unam.mx o con **jgordillo@ciencias.unam.mx**