

C# Y .NET 8

Parte 15. Imágenes

2024-11

Rafael Alberto Moreno Parra
ramsoftware@gmail.com

Contenido

Tabla de ilustraciones.....	3
Acerca del autor.....	4
Licencia de este libro	4
Licencia del software	4
Marcas registradas	5
Dedicatoria	6
Iniciando con imágenes.....	7
Convertir a escala de grises	9
Disminuir el tamaño de la imagen	10
Reflejo Horizontal.....	12
Reflejo Vertical	14
Giro de la imagen.....	16
Filtro blanco y negro.....	18
Invierte colores.....	20
Filtro Kodachrome	22
Filtro Sepia	24
Filtro Lomograph.....	26
Filtro Polaroid	28
Filtro umbral	30
Filtro borde	32
Filtro pixelado.....	34
Filtro de saturación de color.....	36
Filtro viñeta.....	38
Cambiar pixel a pixel	40
Invertir colores	42
Aplicando un núcleo o “kernel”	44
Una aplicación WPF que carga una imagen	46
Una aplicación WPF que aplica varios filtros	49

Tabla de ilustraciones

Ilustración 1: Inicia proyecto en Visual Studio 2022	7
Ilustración 2: Instalación de SixLabors.ImageSharp	8
Ilustración 3: Paquete instalado.....	8
Ilustración 4: Escala de grises	9
Ilustración 5: Cambio de tamaño de imagen	11
Ilustración 6: Reflejo horizontal	13
Ilustración 7: Reflejo vertical	15
Ilustración 8: Giro de la imagen.....	17
Ilustración 9: Filtro blanco y negro.....	19
Ilustración 10: Invierte colores	21
Ilustración 11: Filtro Kodachrome	23
Ilustración 12: Filtro Sepia.....	25
Ilustración 13: Filtro Lomograph	27
Ilustración 14: Filtro Polaroid	29
Ilustración 15: Filtro umbral.....	31
Ilustración 16: Filtro borde	33
Ilustración 17: Filtro pixelado	35
Ilustración 18: Filtro de saturación de color	37
Ilustración 19: Filtro viñeta.....	39
Ilustración 20: Modificando pixel a pixel.....	41
Ilustración 21: Invertir colores	43
Ilustración 22: Aplicando un núcleo	45
Ilustración 23: Aplicación WPF que carga una imagen	48
Ilustración 24: Una ventana de diálogo para escoger el archivo de imagen	49
Ilustración 25: Imagen cargada	50
Ilustración 26: Aplicando el filtro sepia	51
Ilustración 27: Aplicando reflejo horizontal y doble vez Kodachrome	52

Acerca del autor

Rafael Alberto Moreno Parra

ramsoftware@gmail.com o enginelifelife@hotmail.com

Sitio Web: <http://darwin.50webs.com> (dedicado a la investigación de algoritmos evolutivos y vida artificial).

Github: <https://github.com/ramsoftware>

Youtube: <https://www.youtube.com/@RafaelMorenoP>

Licencia de este libro



Licencia del software

Todo el software desarrollado aquí tiene licencia LGPL "Lesser General Public License" [1]



Marcas registradas

En este libro se hace uso de las siguientes tecnologías registradas:

Microsoft ® Windows ® Enlace: <http://windows.microsoft.com/en-US/windows/home>

Microsoft ® Visual Studio 2022 ® Enlace: <https://visualstudio.microsoft.com/es/vs/>

Dedicatoria

A mis padres, a mi hermana....

Y a mi tropa gatuna: Sally, Suini, Grisú, Capuchina, Milú,
Arián, Frac y mis recordados Tinita, Tammy, Vikingo y
Michu.

Iniciando con imágenes

Para trabajar con gráficos, se debe crear un proyecto de consola

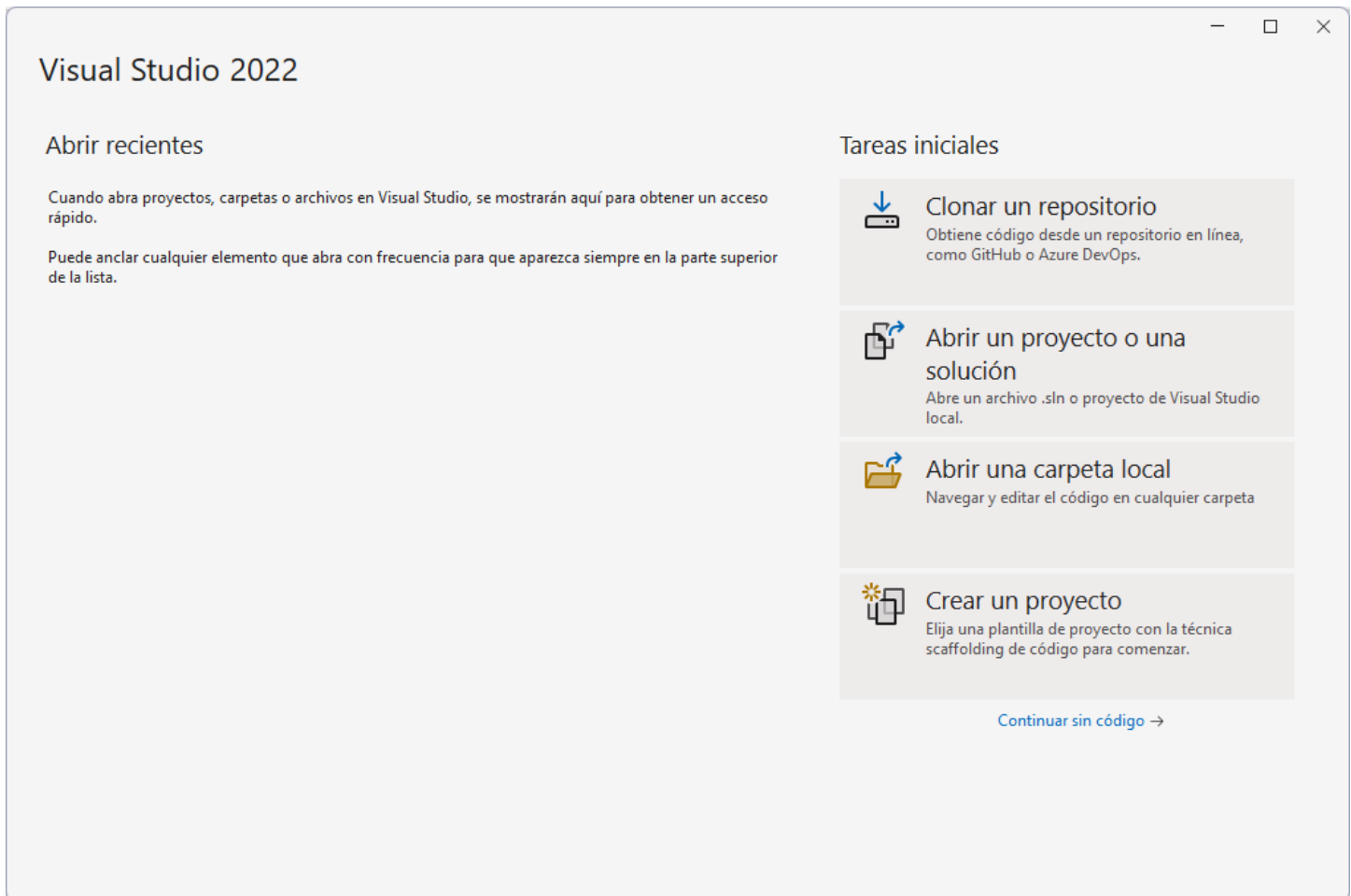


Ilustración 1: Inicia proyecto en Visual Studio 2022

Se requiere el paquete "SIX LABORS – ImageShark", esta es su instalación:

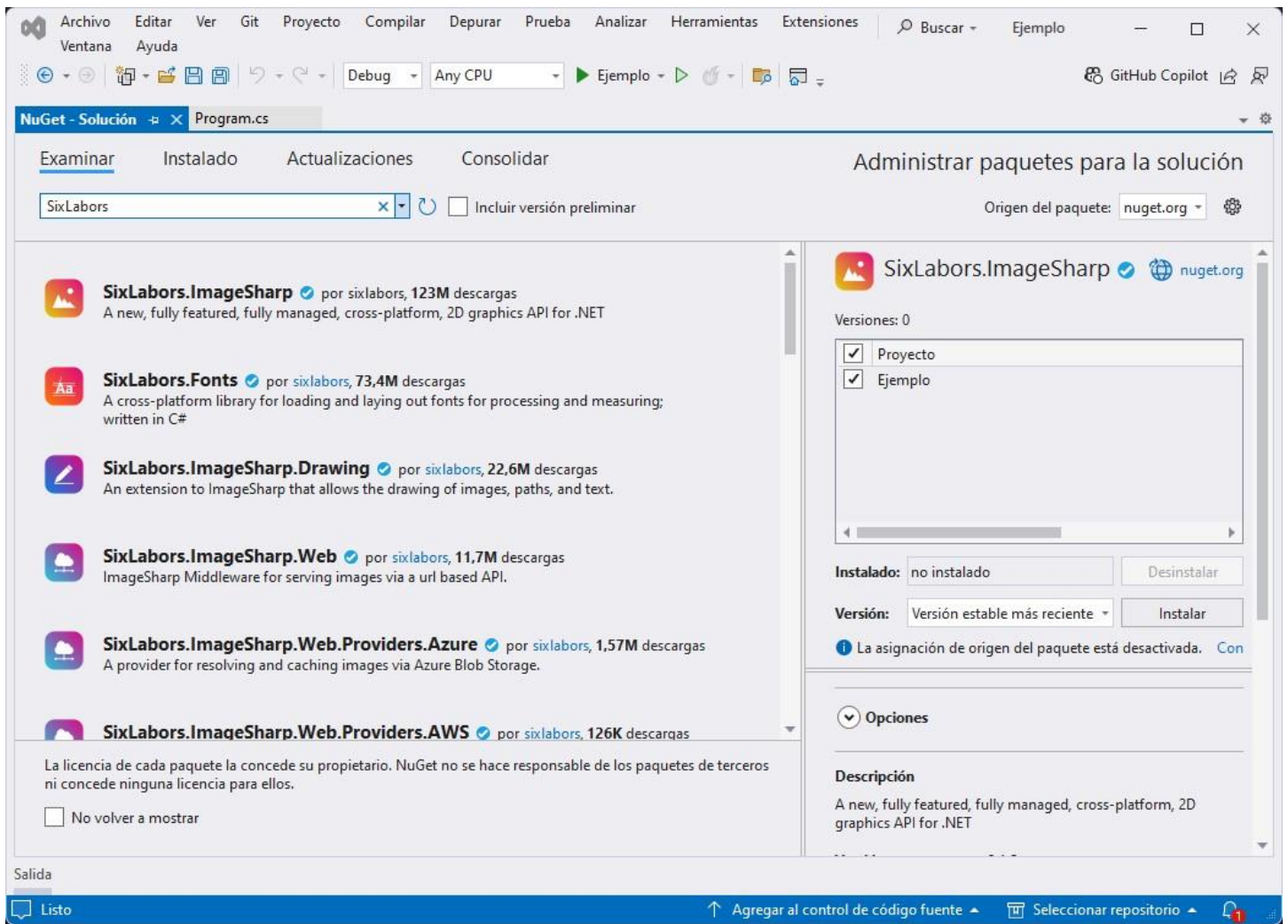


Ilustración 2: Instalación de SixLabors.ImageSharp

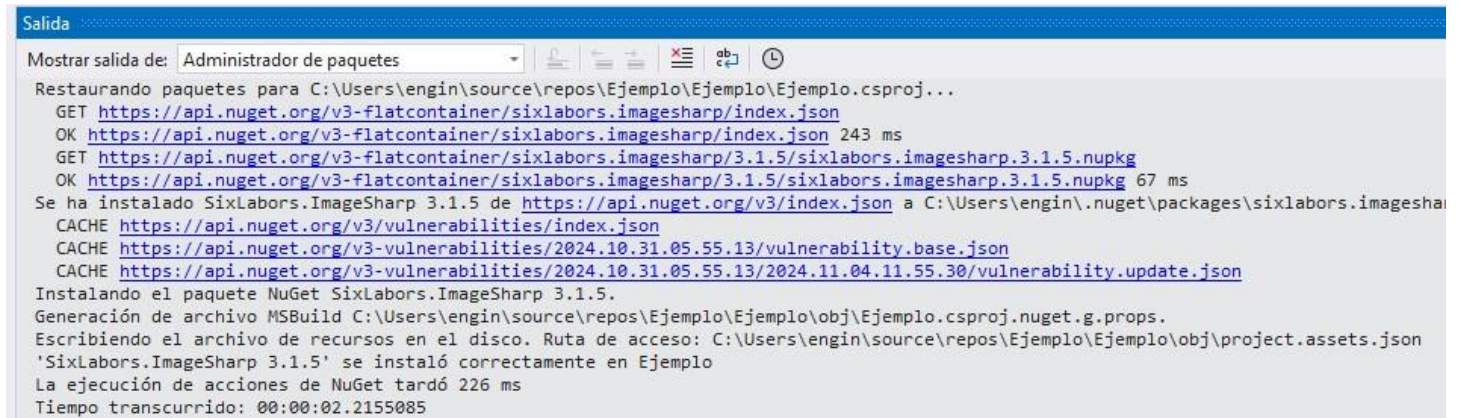


Ilustración 3: Paquete instalado

Convertir a escala de grises

O/001.cs

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Carga imagen original
            string Entrada = "C:\\\\TEMP\\\\Grisú.jpg";
            using (Image<Rgba32> Foto = Image.Load<Rgba32>(Entrada)) {
                //Aplica el filtro de escala de grises
                Foto.Mutate(x => x.Grayscale());

                //Guarda la nueva imagen de escala de grises
                string Salida = "C:\\\\TEMP\\\\GrisúEscalaGrises.jpg";
                Foto.Save(Salida);
            }

            Console.WriteLine("Conversión terminada.");
        }
    }
}
```

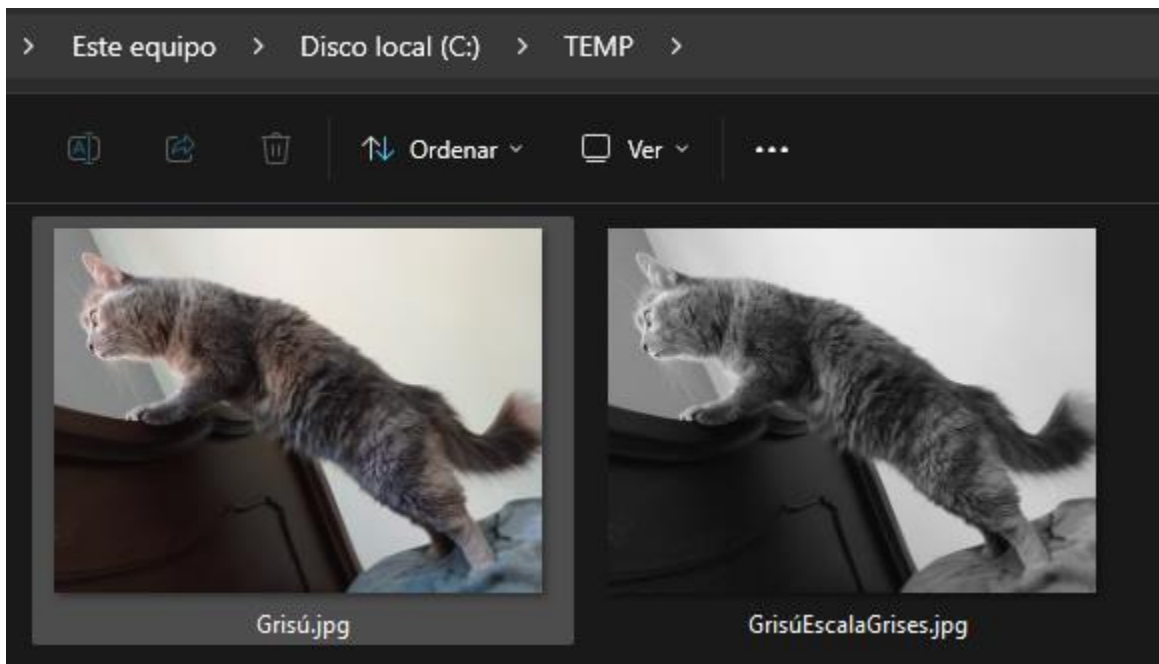


Ilustración 4: Escala de grises

Disminuir el tamaño de la imagen

O/002.cs

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Carga imagen original
            string Entrada = "C:\\TEMP\\Grisú.jpg";
            using (Image<Rgba32> Foto = Image.Load<Rgba32>(Entrada)) {
                //Aplica el filtro para disminuir
                Foto.Mutate(x => x.Resize(Foto.Width / 10, Foto.Height / 10));

                //Guarda la nueva imagen
                string Salida = "C:\\TEMP\\GrisúEscala.jpg";
                Foto.Save(Salida);
            }

            Console.WriteLine("Conversión terminada.");
        }
    }
}
```

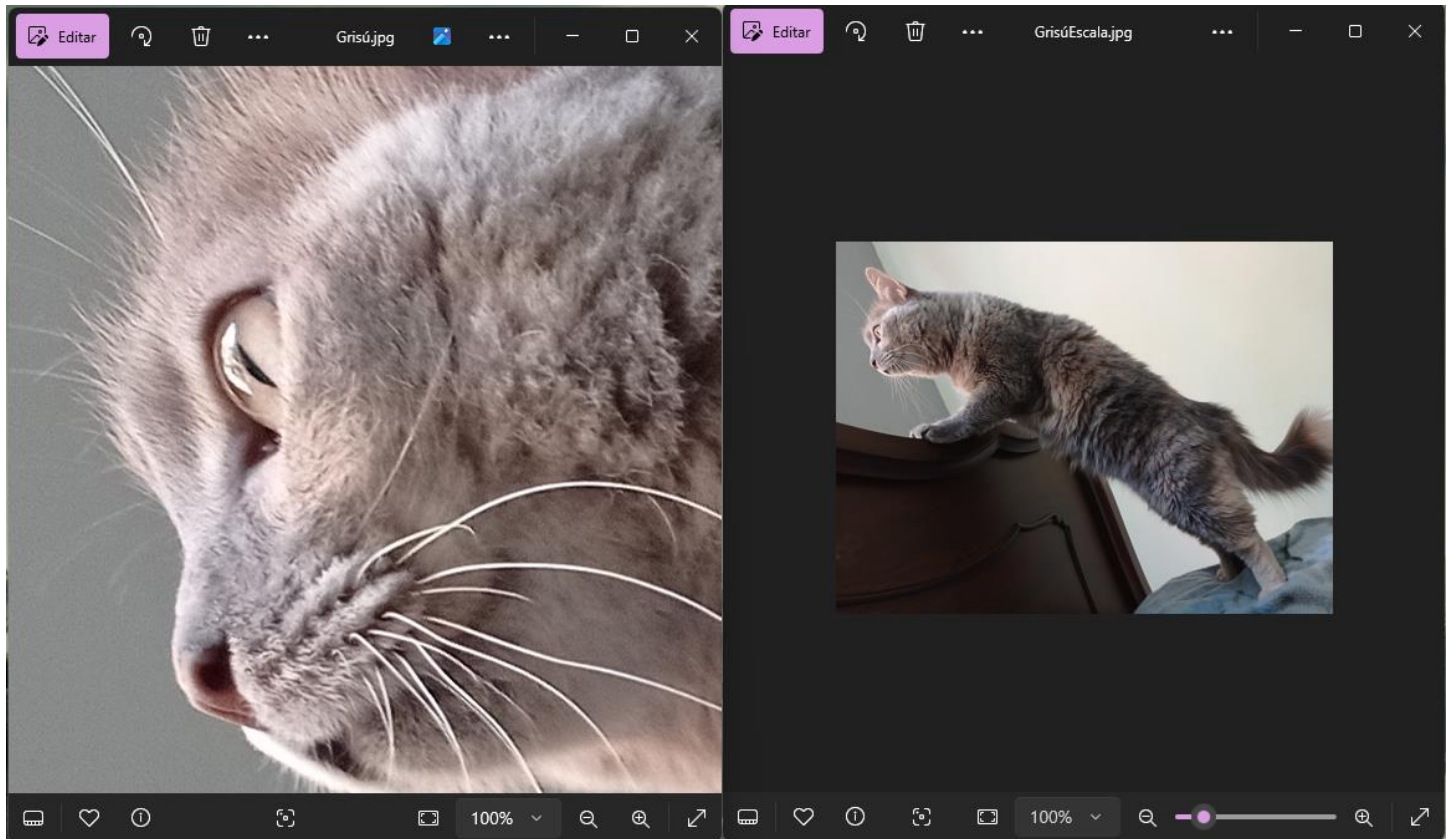


Ilustración 5: Cambio de tamaño de imagen

Reflejo Horizontal

O/003.cs

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Carga imagen original
            string Entrada = "C:\\TEMP\\Grisú.jpg";
            using (Image<Rgba32> Foto = Image.Load<Rgba32>(Entrada)) {
                // Aplicar el reflejo horizontal
                Foto.Mutate(x => x.Flip(FlipMode.Horizontal));

                //Guarda la nueva imagen
                string Salida = "C:\\TEMP\\GrisúReflejoHorizontal.jpg";
                Foto.Save(Salida);
            }

            Console.WriteLine("Conversión terminada.");
        }
    }
}
```

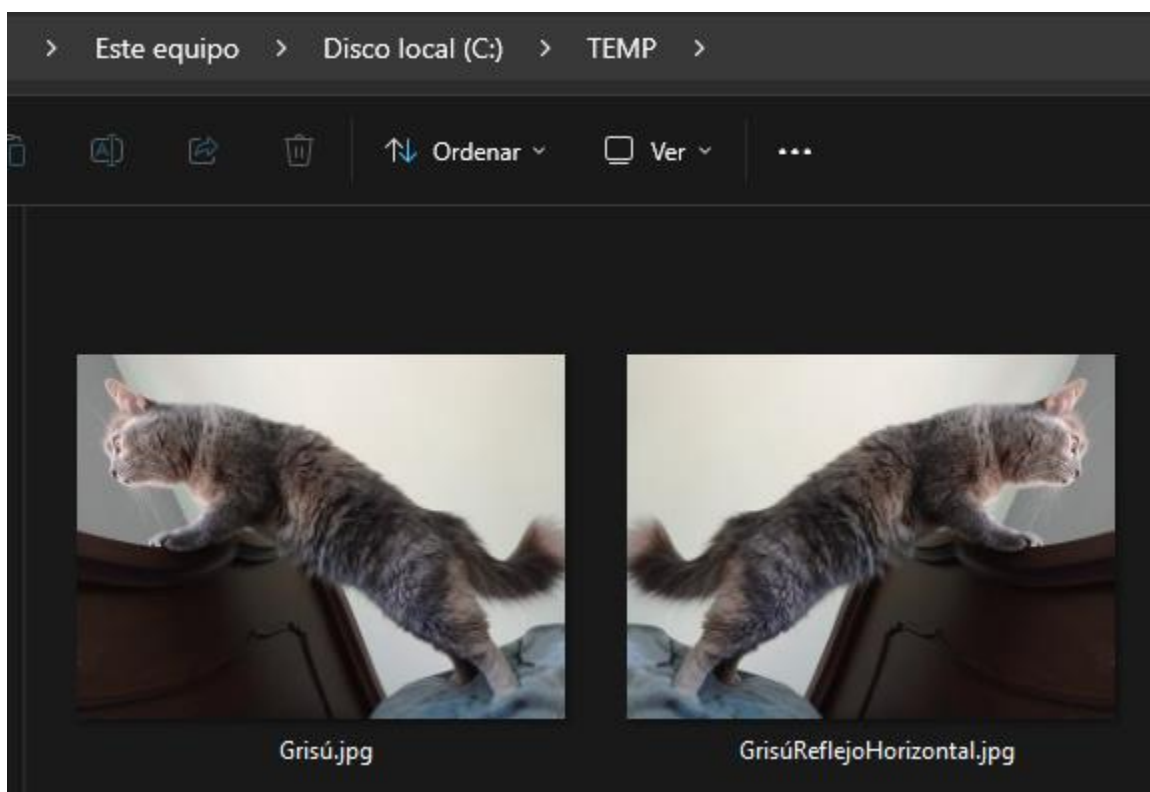


Ilustración 6: Reflejo horizontal

Reflejo Vertical

O/004.cs

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Carga imagen original
            string Entrada = "C:\\\\TEMP\\\\Grisú.jpg";
            using (Image<Rgba32> Foto = Image.Load<Rgba32>(Entrada)) {
                // Aplicar el reflejo vertical
                Foto.Mutate(x => x.Flip(FlipMode.Vertical));

                //Guarda la nueva imagen
                string Salida = "C:\\\\TEMP\\\\GrisúReflejoVertical.jpg";
                Foto.Save(Salida);
            }

            Console.WriteLine("Conversión terminada.");
        }
    }
}
```

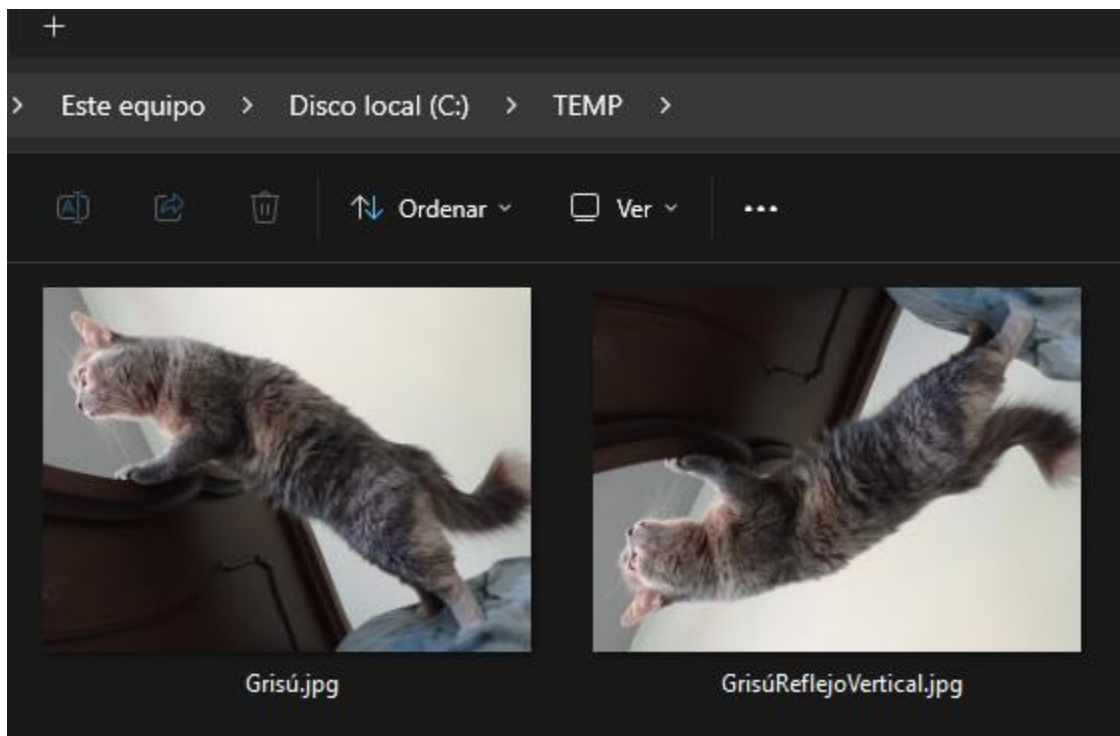


Ilustración 7: Reflejo vertical

Giro de la imagen

O/005.cs

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Carga imagen original
            string Entrada = "C:\\TEMP\\Grisú.jpg";
            using (Image<Rgba32> Foto = Image.Load<Rgba32>(Entrada)) {
                // Aplicar el giro en 45 grados
                Foto.Mutate(x => x.Rotate(45));

                //Guarda la nueva imagen
                string Salida = "C:\\TEMP\\GrisúGiro.jpg";
                Foto.Save(Salida);
            }

            Console.WriteLine("Conversión terminada.");
        }
    }
}
```

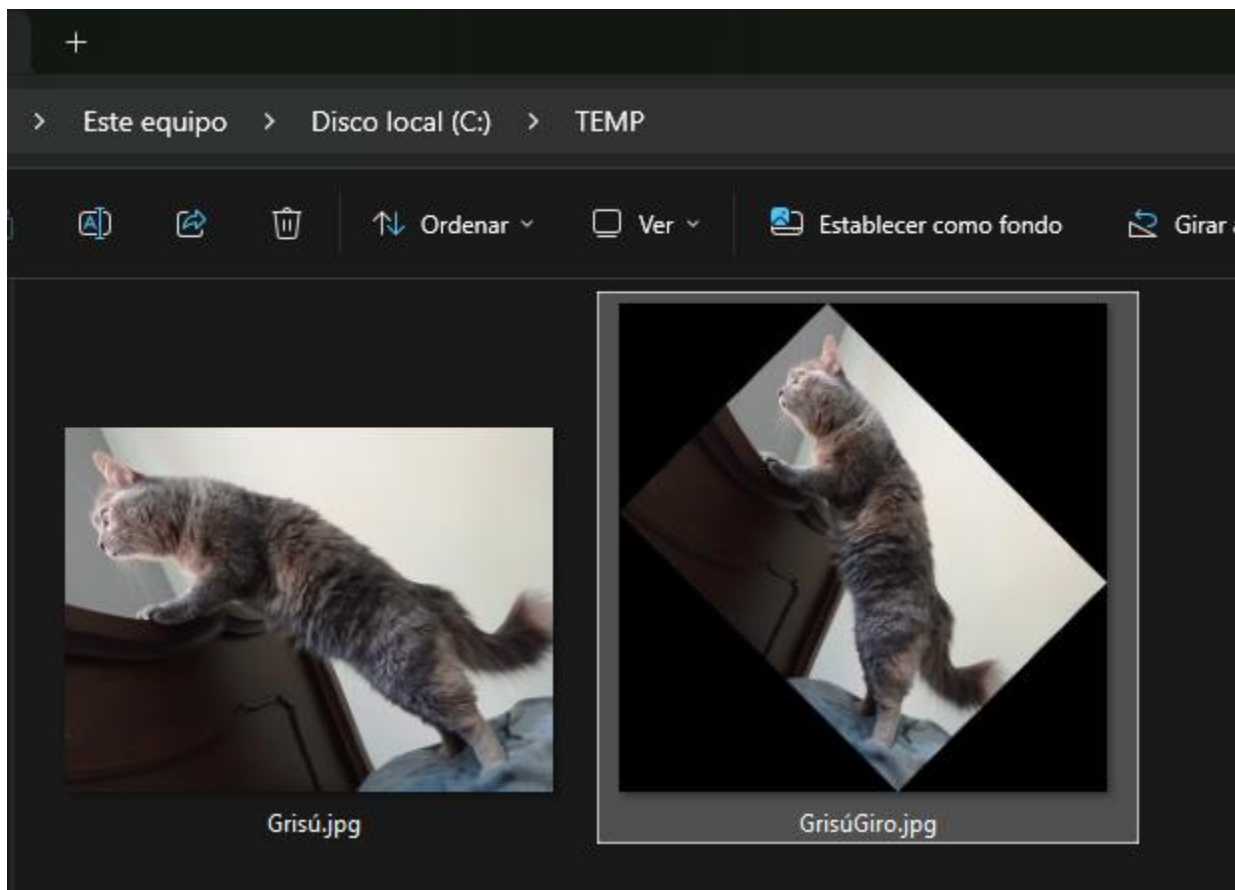



Ilustración 8: Giro de la imagen

Filtro blanco y negro

O/006.cs

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Carga imagen original
            string Entrada = "C:\\\\TEMP\\\\Grisú.jpg";
            using (Image<Rgba32> Foto = Image.Load<Rgba32>(Entrada)) {
                //Aplica el filtro de blanco y negro
                Foto.Mutate(x => x.BlackWhite());

                //Guarda la nueva imagen
                string Salida = "C:\\\\TEMP\\\\GrisúBlancoNegro.jpg";
                Foto.Save(Salida);
            }

            Console.WriteLine("Conversión terminada.");
        }
    }
}
```

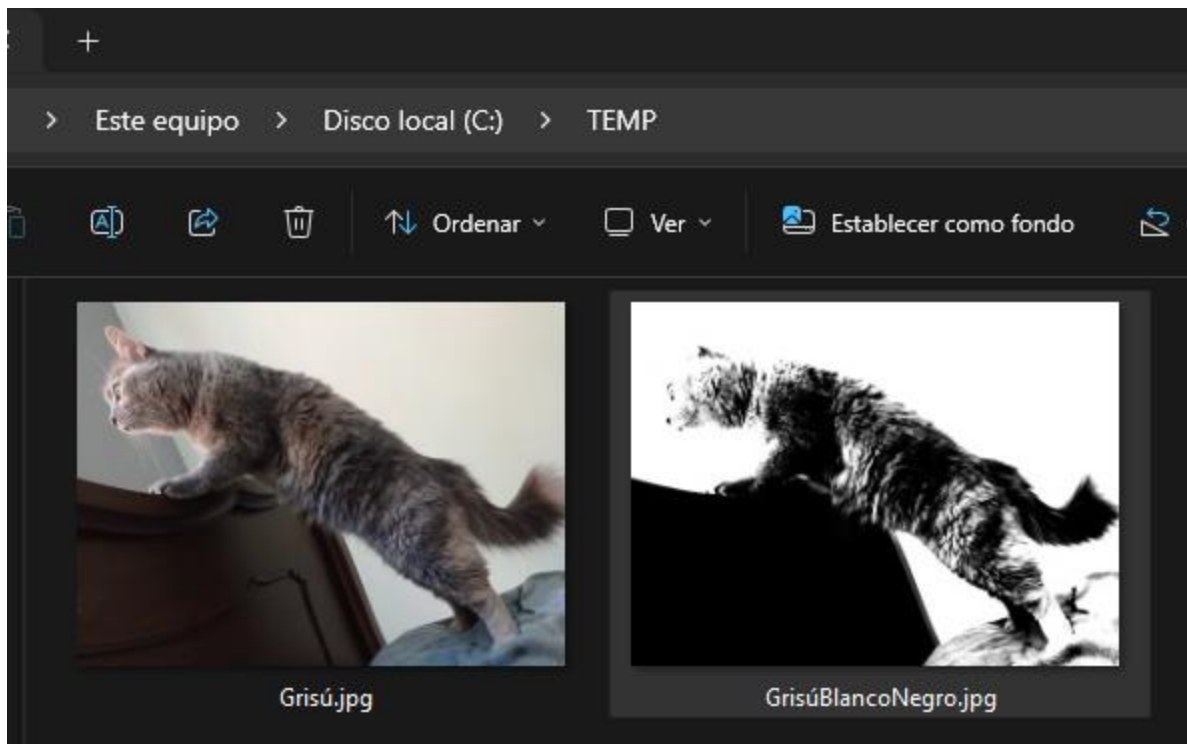


Ilustración 9: Filtro blanco y negro

Invierte colores

O/007.cs

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Carga imagen original
            string Entrada = "C:\\TEMP\\Grisú.jpg";
            using (Image<Rgba32> Foto = Image.Load<Rgba32>(Entrada)) {
                //Aplica el filtro de invertir colores
                Foto.Mutate(x => x.Invert());

                //Guarda la nueva imagen
                string Salida = "C:\\TEMP\\GrisúInvierte.jpg";
                Foto.Save(Salida);
            }

            Console.WriteLine("Conversión terminada.");
        }
    }
}
```

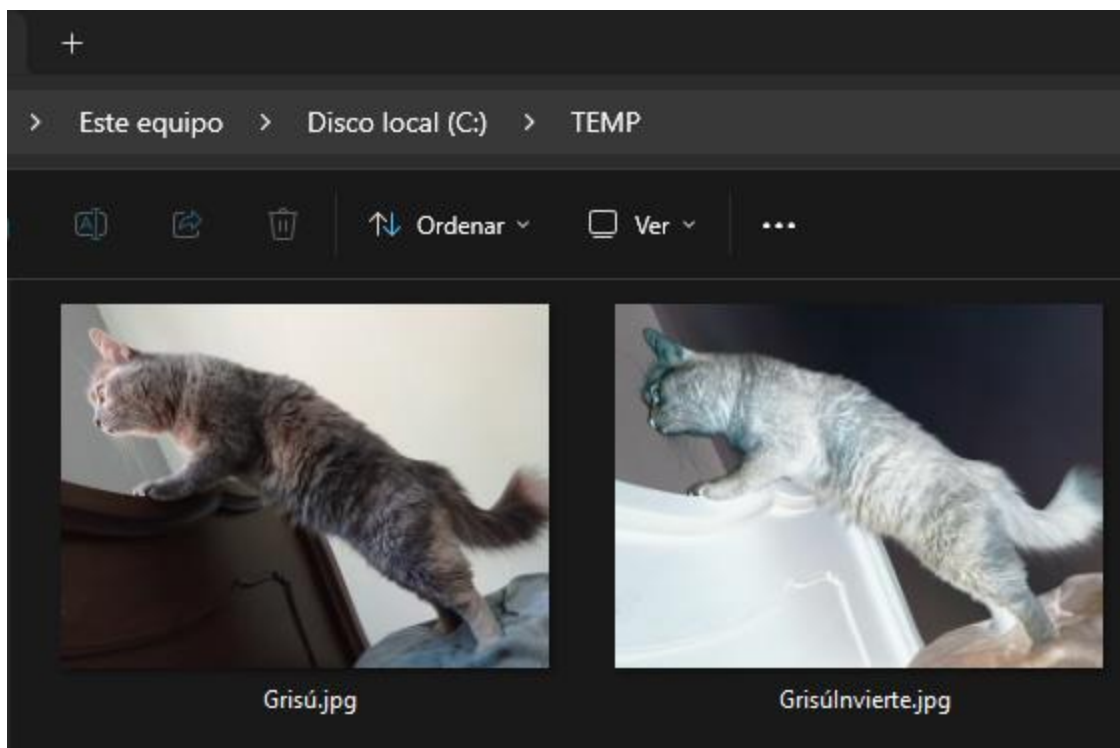


Ilustración 10: Invierte colores

Filtro Kodachrome

Aplica un efecto similar a las cámaras antiguas Kodachrome.

O/008.cs

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Carga imagen original
            string Entrada = "C:\\TEMP\\Grisú.jpg";
            using (Image<Rgba32> Foto = Image.Load<Rgba32>(Entrada)) {
                //Aplica un efecto similar a las cámaras antiguas Kodachrome
                Foto.Mutate(x => x.Kodachrome());

                //Guarda la nueva imagen
                string Salida = "C:\\TEMP\\GrisúKodachrome.jpg";
                Foto.Save(Salida);
            }

            Console.WriteLine("Conversión terminada.");
        }
    }
}
```

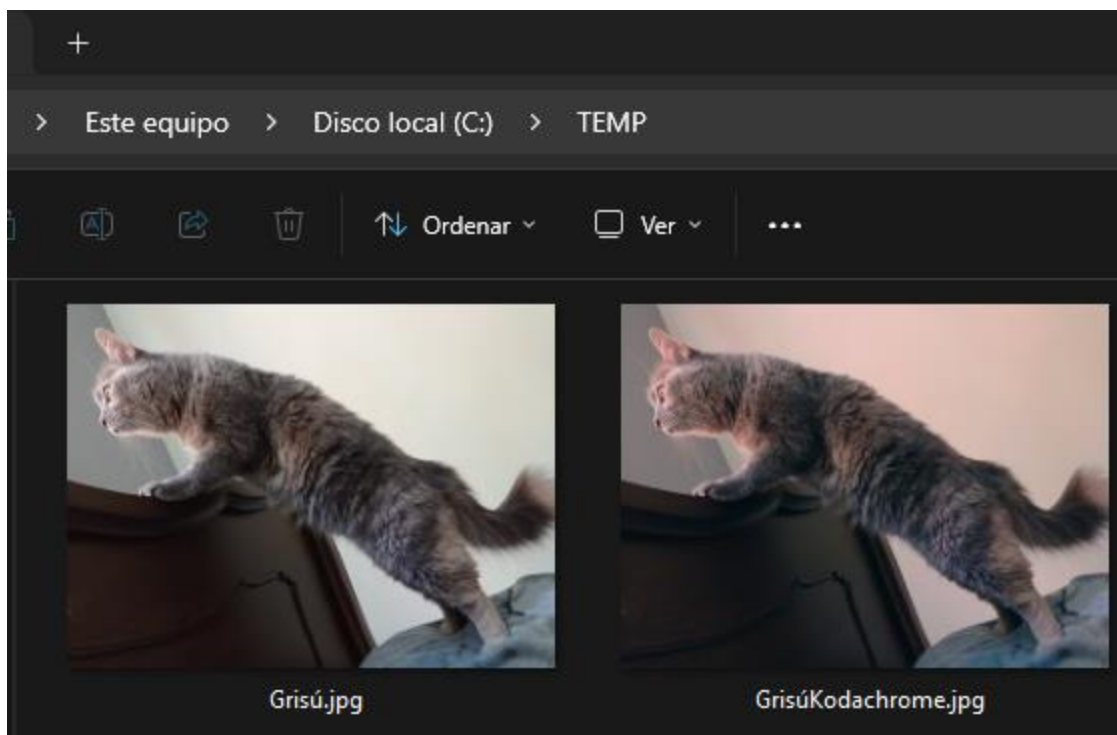


Ilustración 11: Filtro Kodachrome

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Carga imagen original
            string Entrada = "C:\\\\TEMP\\\\Grisú.jpg";
            using (Image<Rgba32> Foto = Image.Load<Rgba32>(Entrada)) {
                //Aplica el filtro sepia
                Foto.Mutate(x => x.Sepia());

                //Guarda la nueva imagen
                string Salida = "C:\\\\TEMP\\\\GrisúSepia.jpg";
                Foto.Save(Salida);
            }

            Console.WriteLine("Conversión terminada.");
        }
    }
}
```

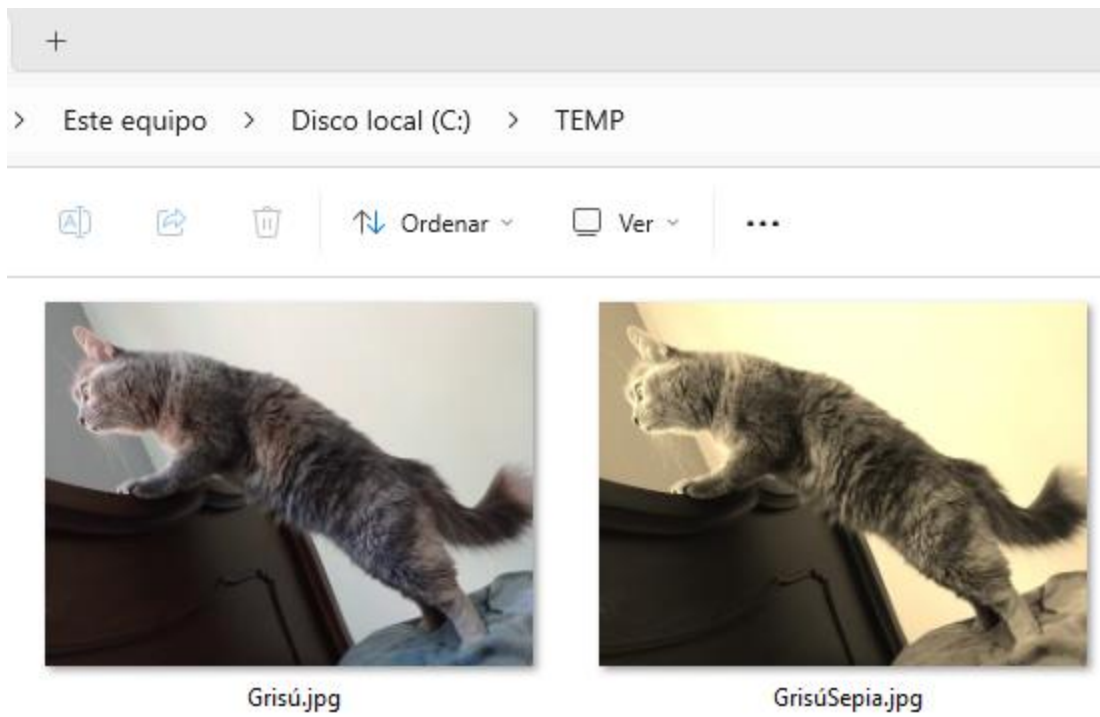



Ilustración 12: Filtro Sepia

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Carga imagen original
            string Entrada = "C:\\\\TEMP\\\\Grisú.jpg";
            using (Image<Rgba32> Foto = Image.Load<Rgba32>(Entrada)) {
                //Aplica el filtro Lomograph
                Foto.Mutate(x => x.Lomograph());

                //Guarda la nueva imagen
                string Salida = "C:\\\\TEMP\\\\GrisúLomograph.jpg";
                Foto.Save(Salida);
            }

            Console.WriteLine("Conversión terminada.");
        }
    }
}
```

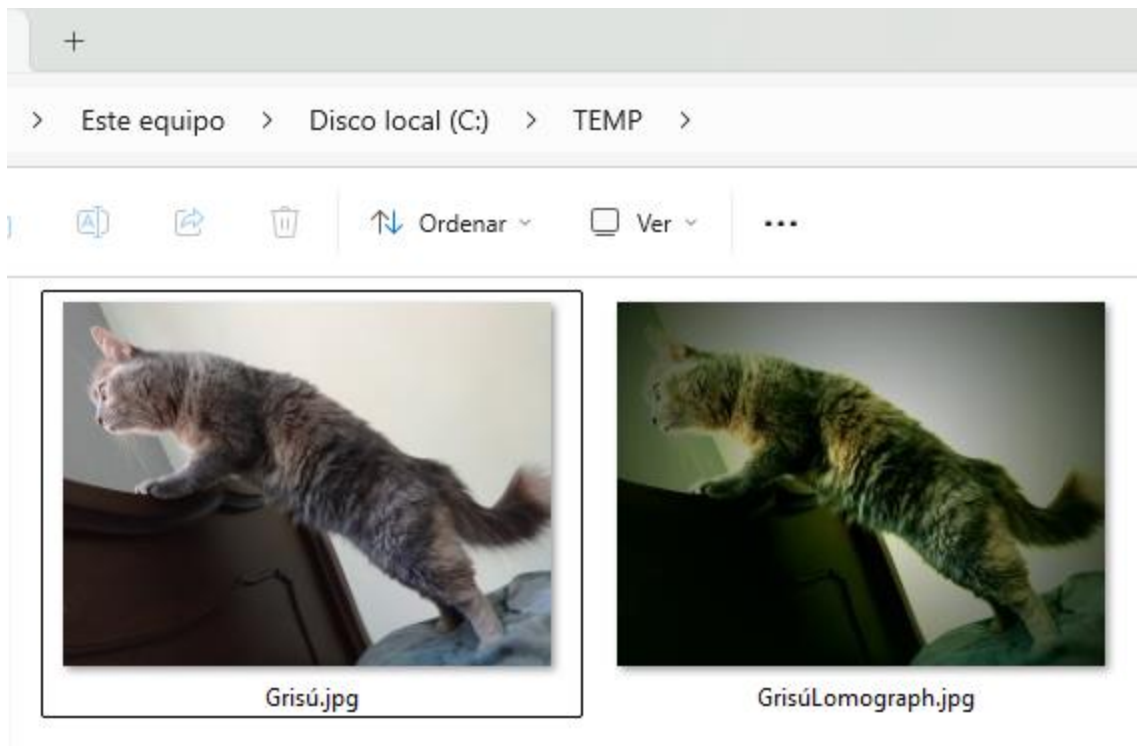


Ilustración 13: Filtro Lomograph

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Carga imagen original
            string Entrada = "C:\\\\TEMP\\\\Grisú.jpg";
            using (Image<Rgba32> Foto = Image.Load<Rgba32>(Entrada)) {
                //Aplica el filtro Polaroid
                Foto.Mutate(x => x.Polaroid());

                //Guarda la nueva imagen
                string Salida = "C:\\\\TEMP\\\\GrisúPolaroid.jpg";
                Foto.Save(Salida);
            }

            Console.WriteLine("Conversión terminada.");
        }
    }
}
```

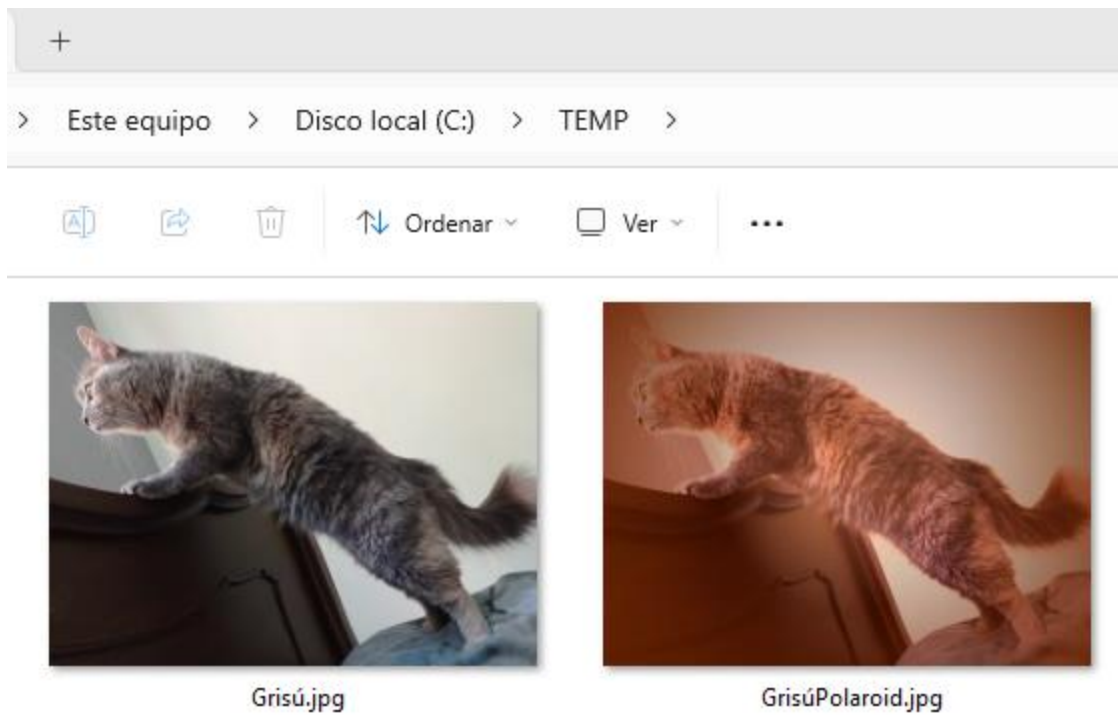


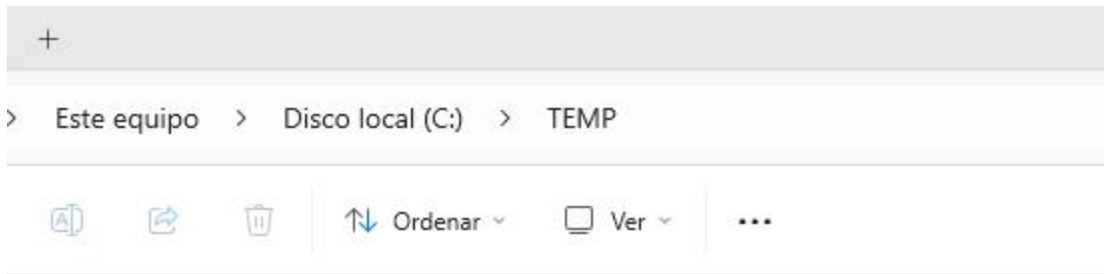
Ilustración 14: Filtro Polaroid

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Carga imagen original
            string Entrada = "C:\\\\TEMP\\\\Grisú.jpg";
            using (Image<Rgba32> Foto = Image.Load<Rgba32>(Entrada)) {
                //Aplica el filtro Threshold
                Foto.Mutate(x => x.AdaptiveThreshold());

                //Guarda la nueva imagen
                string Salida = "C:\\\\TEMP\\\\GrisúThreshold.jpg";
                Foto.Save(Salida);
            }

            Console.WriteLine("Conversión terminada.");
        }
    }
}
```



Grisú.jpg



GrisúThreshold.jpg

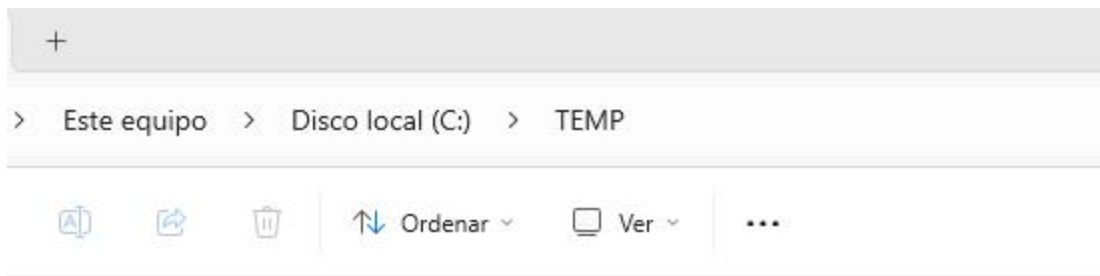
Ilustración 15: Filtro umbral

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Carga imagen original
            string Entrada = "C:\\\\TEMP\\\\Grisú.jpg";
            using (Image<Rgba32> Foto = Image.Load<Rgba32>(Entrada)) {
                //Aplica el filtro detección de bordes
                Foto.Mutate(x => x.DetectEdges());

                //Guarda la nueva imagen
                string Salida = "C:\\\\TEMP\\\\GrisúBordes.jpg";
                Foto.Save(Salida);
            }

            Console.WriteLine("Conversión terminada.");
        }
    }
}
```

Grisú.jpg



GrisúBordes.jpg

Ilustración 16: Filtro borde

Filtro pixelado

Pixelado de la imagen, el parámetro es el tamaño del pixel, entre más alto, más se pixela.

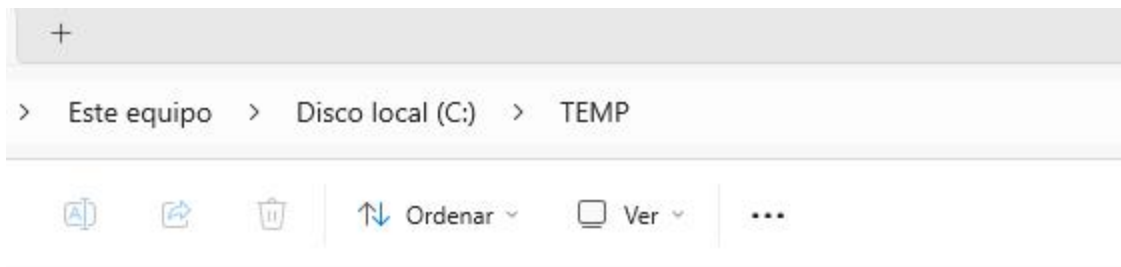
O/014.cs

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Carga imagen original
            string Entrada = "C:\\\\TEMP\\\\Grisú.jpg";
            using (Image<Rgba32> Foto = Image.Load<Rgba32>(Entrada)) {
                //Aplica el filtro pixelado
                Foto.Mutate(x => x.Pixelate(100));

                //Guarda la nueva imagen
                string Salida = "C:\\\\TEMP\\\\GrisúPixelado.jpg";
                Foto.Save(Salida);
            }

            Console.WriteLine("Conversión terminada.");
        }
    }
}
```



Grisú.jpg



GrisúPixelado.jpg

Ilustración 17: Filtro pixelado

Filtro de saturación de color

Recibe un parámetro entero, entre más alto, más satura el color.

O/015.cs

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Carga imagen original
            string Entrada = "C:\\TEMP\\Grisú.jpg";
            using (Image<Rgba32> Foto = Image.Load<Rgba32>(Entrada)) {
                //Aplica el filtro de saturación de color
                Foto.Mutate(x => x.Saturate((float)10));

                //Guarda la nueva imagen
                string Salida = "C:\\TEMP\\GrisúSaturado.jpg";
                Foto.Save(Salida);
            }

            Console.WriteLine("Conversión terminada.");
        }
    }
}
```

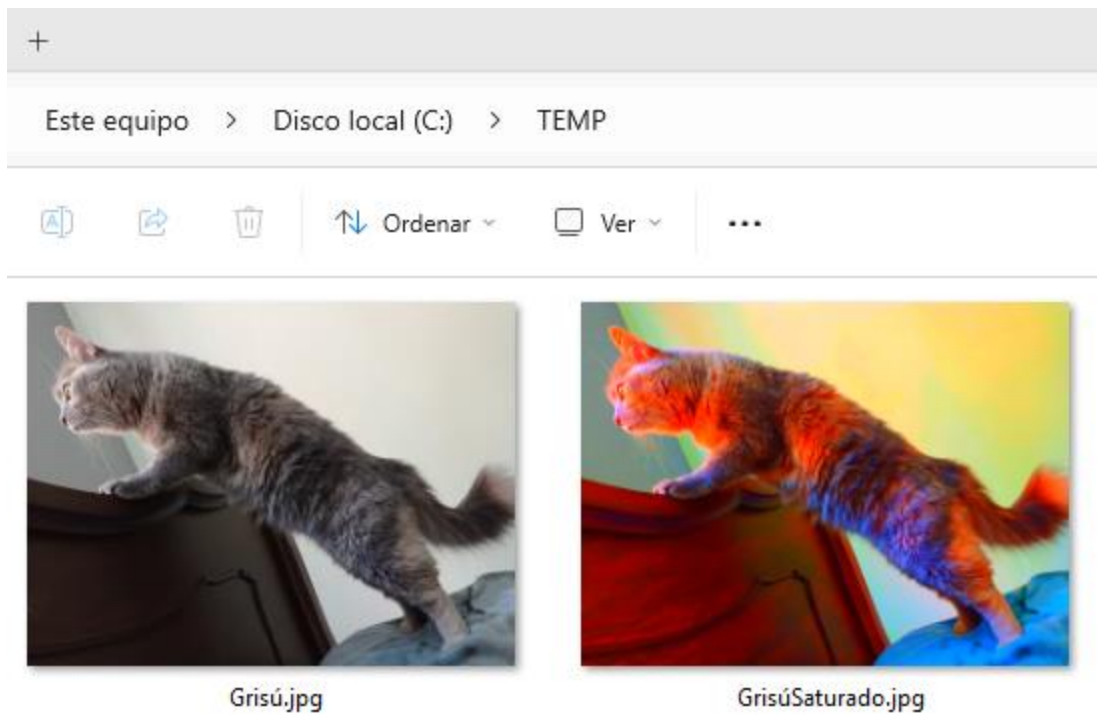


Ilustración 18: Filtro de saturación de color

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Carga imagen original
            string Entrada = "C:\\\\TEMP\\\\Grisú.jpg";
            using (Image<Rgba32> Foto = Image.Load<Rgba32>(Entrada)) {
                //Aplica el filtro viñeta
                Foto.Mutate(x => x.Vignette(Color.Blue));

                //Guarda la nueva imagen
                string Salida = "C:\\\\TEMP\\\\GrisúVineta.jpg";
                Foto.Save(Salida);
            }

            Console.WriteLine("Conversión terminada.");
        }
    }
}
```

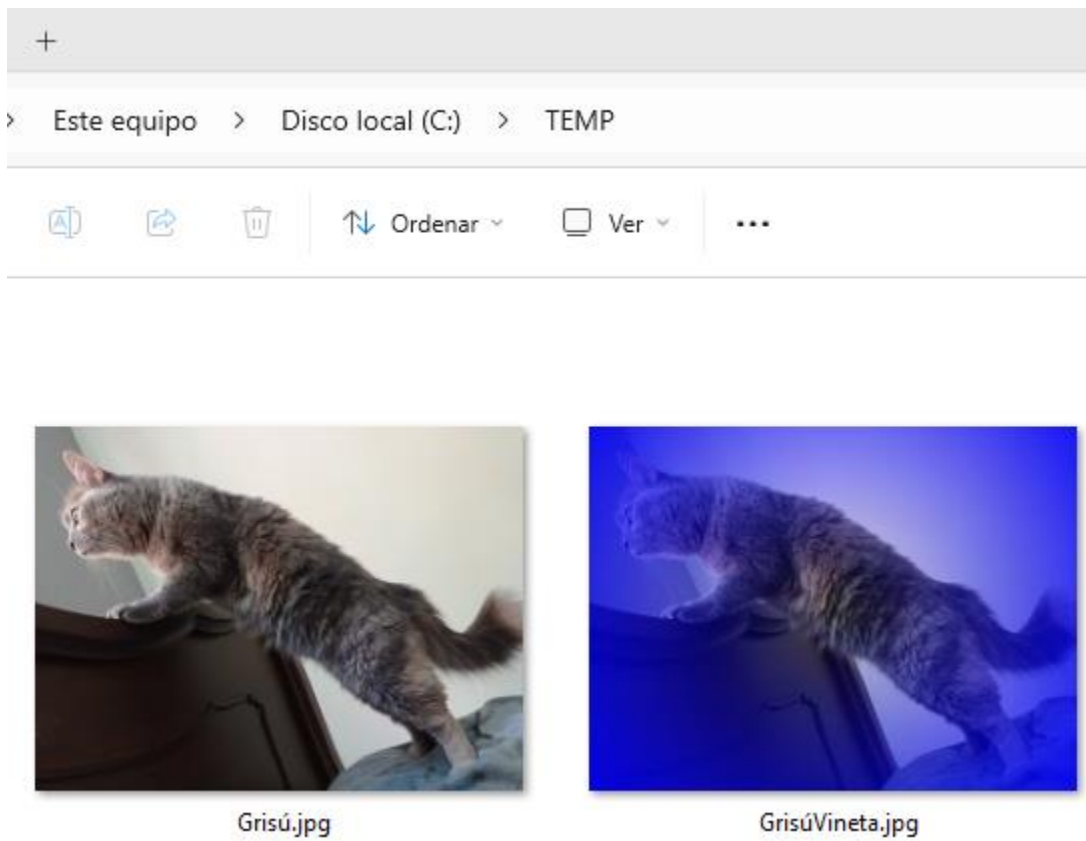


Ilustración 19: Filtro viñeta

Cambiar pixel a pixel

O/017.cs

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;

namespace Ejemplo {
    internal class Program {
        static void Main() {
            // Cargar la imagen original
            using (var Foto = Image.Load<Rgba32>("C:\\TEMP\\Grisú.jpg")) {
                // Recorrer cada píxel y convertirlo a escala de grises
                for (int y = 0; y < Foto.Height; y++) {
                    for (int x = 0; x < Foto.Width; x++) {
                        var pixel = Foto[x, y];
                        // Calcular el valor de gris usando la fórmula de luminancia
                        byte gris = (byte)(0.3 * pixel.R + 0.59 * pixel.G + 0.11 *
pixel.B);
                        var Pixelgris = new Rgba32(gris, gris, gris, pixel.A);
                        Foto[x, y] = Pixelgris;
                    }
                }

                // Guardar la imagen en escala de grises
                Foto.Save("C:\\TEMP\\GrisúPixelGris.jpg");
            }
            Console.WriteLine("Proceso terminado");
        }
    }
}
```

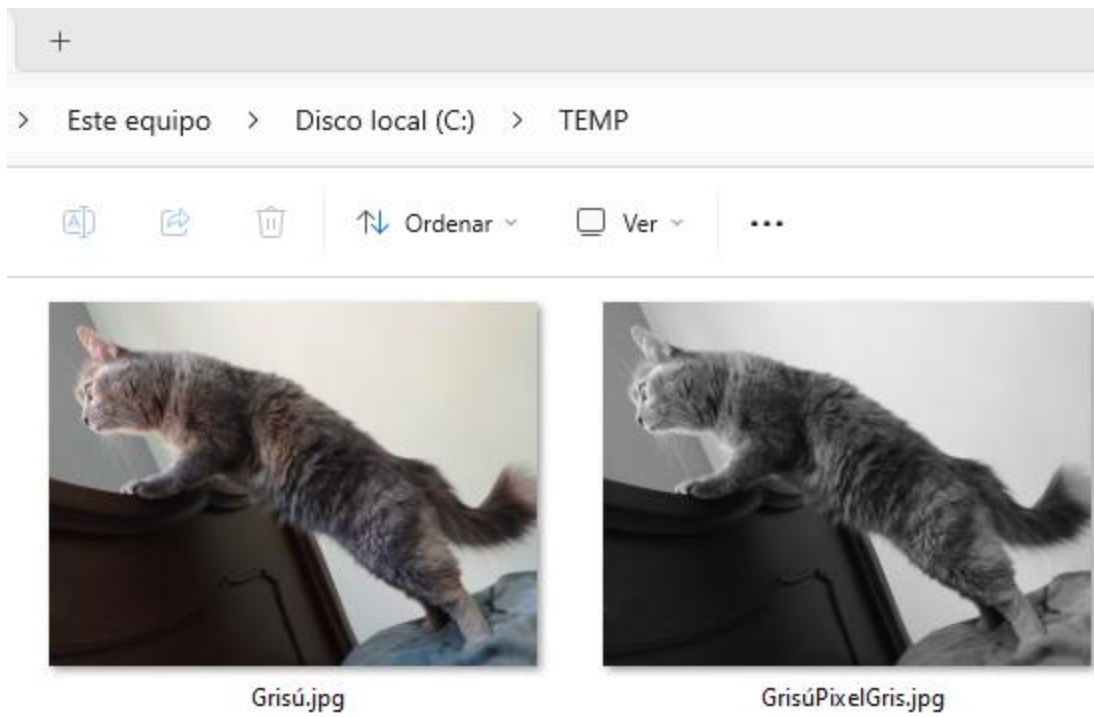



Ilustración 20: Modificando pixel a pixel

Invertir colores

O/018.cs

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;

namespace Ejemplo {
    internal class Program {
        static void Main() {
            // Cargar la imagen original
            using (var Foto = Image.Load<Rgba32>("C:\\TEMP\\Grisú.jpg")) {

                // Recorrer cada píxel y convertirlo
                for (int y = 0; y < Foto.Height; y++) {
                    for (int x = 0; x < Foto.Width; x++) {
                        var pixel = Foto[x, y];

                        //Invierte el color
                        byte NuevoR = (byte)(255 - pixel.R);
                        byte NuevoG = (byte)(255 - pixel.G);
                        byte NuevoB = (byte)(255 - pixel.B);

                        //El píxel con el nuevo color
                        var PixelNuevo = new Rgba32(NuevoR, NuevoG, NuevoB, pixel.A);

                        //Cambia la imagen
                        Foto[x, y] = PixelNuevo;
                    }
                }

                // Guardar la imagen en escala de grises
                Foto.Save("C:\\TEMP\\GrisúInvierte.jpg");
            }
            Console.WriteLine("Proceso terminado");
        }
    }
}
```

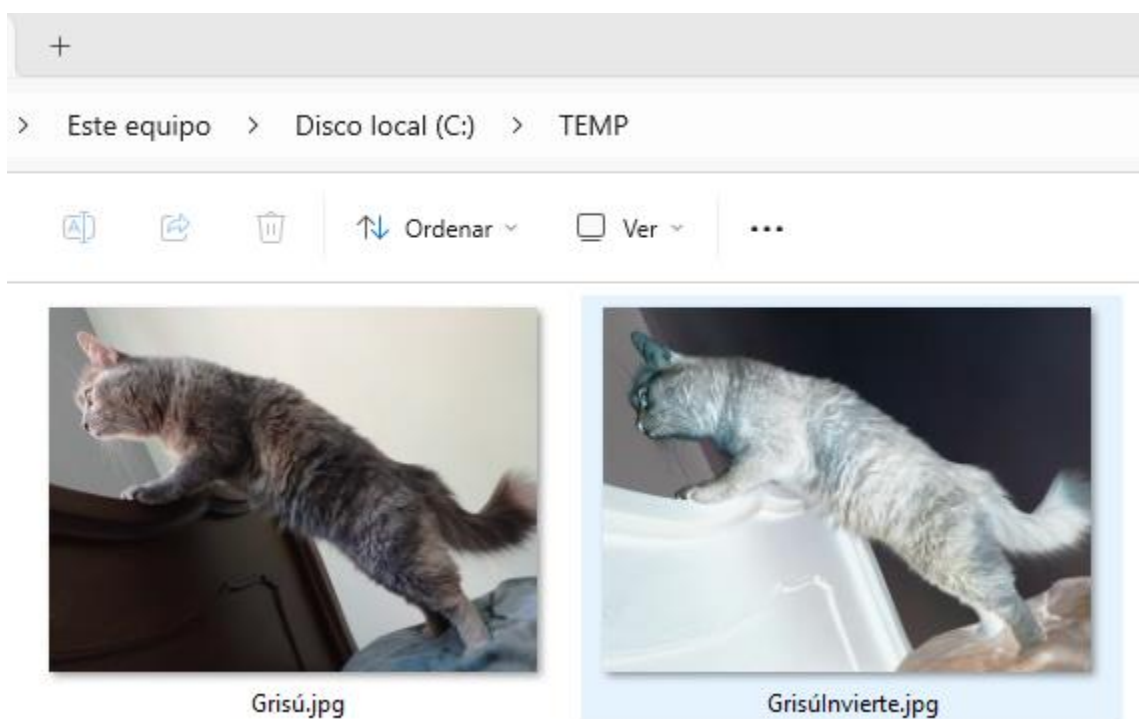


Ilustración 21: Invertir colores

Aplicando un núcleo o “kernel”

O/019.cs

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;

namespace Ejemplo {
    class Program {
        static void Main() {
            int[,] Nucleo = {
                { 1, 0, -1 },
                { 1, 0, -1 },
                { 1, 0, -1 }
            };

            int nAlto = Nucleo.GetLength(0);
            int nAncho = Nucleo.GetLength(1);

            using (var Foto = Image.Load<Rgba32>("C:\\TEMP\\Grisú.jpg")) {
                var pixel = Foto[0, 0];
                for (int y = 0; y < Foto.Height - nAlto + 1; y++) {
                    for (int x = 0; x < Foto.Width - nAncho + 1; x++) {
                        int Acumula = 0;
                        for (int nY = 0; nY < nAlto; nY++) {
                            for (int nX = 0; nX < nAncho; nX++) {
                                pixel = Foto[x + nX, y + nY];
                                int gris = (int) (0.3 * pixel.R + 0.59 * pixel.G + 0.11 *
pixel.B);
                                Acumula += gris * Nucleo[nX, nY];
                            }
                        }
                        byte suma = (byte) Acumula;
                        Foto[x, y] = new Rgba32(suma, suma, suma, pixel.A);
                    }
                }
                // Guardar la imagen que se le aplicó el núcleo
                Foto.Save("C:\\TEMP\\GrisúAplicaNucleo.jpg");
            }
            Console.WriteLine("Proceso terminado");
        }
    }
}
```

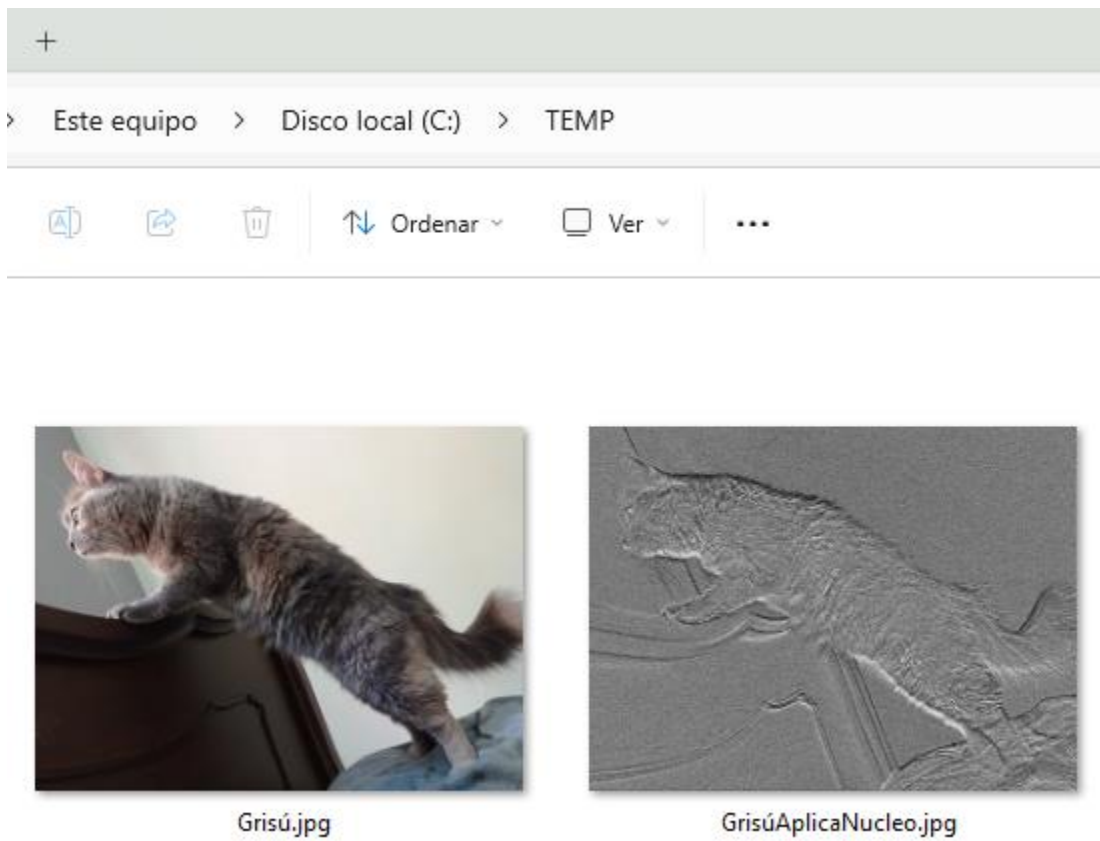


Ilustración 22: Aplicando un núcleo

Una aplicación WPF que carga una imagen

O/020.cs

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;
using System.IO;
using System.Windows;
using System.Windows.Media.Imaging;

namespace SharpImagen {
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window {
        public MainWindow() {
            InitializeComponent();
            LoadImage("C:\\TEMP\\Grisú.jpg");
        }

        //Carga la imagen y la muestra en el control
        private void LoadImage(string RutaImagen) {
            try {
                using (var image =
SixLabors.ImageSharp.Image.Load<Rgba32>(RutaImagen)) {
                    using (var FlujoMemoria = new MemoryStream()) {
                        image.SaveAsBmp(FlujoMemoria);

                        /* Mover la posición actual dentro del flujo de memoria
                        (FlujoMemoria) a un punto específico.
                        FlujoMemoria es usado para almacenar temporalmente la imagen
                        en memoria.
                        Seek(0, SeekOrigin.Begin): Es el método que mueve la
                        posición actual dentro del flujo.
                        Este método toma dos parámetros:
                        0: Este es el desplazamiento en bytes desde la posición
                        especificada por el segundo parámetro.
                        En este caso, 0 significa que no se está moviendo
                        desde la posición especificada, simplemente se está
                        estableciendo la posición en el inicio del flujo.
                        SeekOrigin.Begin: Este es un enumerador que especifica
                        el punto de referencia desde el cual
                        se calcula la nueva posición.
                        SeekOrigin.Begin indica que el punto
                        de referencia es el comienzo del flujo. */
                        FlujoMemoria.Seek(0, SeekOrigin.Begin);

                        var bitmap = new BitmapImage();
```

```

/* BeginInit() marca el inicio de un bloque de inicialización
 * para el objeto BitmapImage. Durante este bloque, se puede
 * establecer varias propiedades del BitmapImage sin que el
 * objeto intente cargarse o procesarse inmediatamente */
bitmap.BeginInit();
bitmap.StreamSource = FlujoMemoria;

/* Se utiliza para configurar cómo se almacena en caché la
 imagen cuando se carga en un objeto BitmapImage.
 bitmap: Es el objeto BitmapImage que se está utilizando
 para mostrar la imagen en la aplicación WPF.
 CacheOption: Es una propiedad del BitmapImage que determina
 cómo se almacena en caché la imagen.
 Esto afecta el rendimiento y el uso de memoria
 de la aplicación.
 BitmapCacheOption.OnLoad: Es uno de los valores del enumerador
 BitmapCacheOption. Especifica que la
 imagen debe cargarse completamente en
 memoria cuando se llama al método
 EndInit(). Esto significa que toda
 la imagen se carga y se almacena en
 memoria de una vez.
 */
bitmap.CacheOption = BitmapCacheOption.OnLoad;

/* En este punto, el BitmapImage se carga y se procesa utilizando
 * las propiedades que se establecieron durante el bloque
 * de inicialización. */
bitmap.EndInit();
ImagenFoto.Source = bitmap;
}
}
}
catch (Exception ex) {
    MessageBox.Show($"Error al cargar la imagen: {ex.Message}");
}
}
}
}
}

```

O/020.xaml

```

<Window x:Class="SharpImagen.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

```

```
xmlns:local="clr-namespace:SharpImagen"  
mc:Ignorable="d"  
Title="MainWindow" Height="450" Width="800">  
<Grid>  
    <Image x:Name="ImagenFoto" HorizontalAlignment="Left" Height="333"  
Margin="45,41,0,0" VerticalAlignment="Top" Width="676"/>  
</Grid>  
</Window>
```

El proyecto completo se puede descargar en [O/020.7z](#)

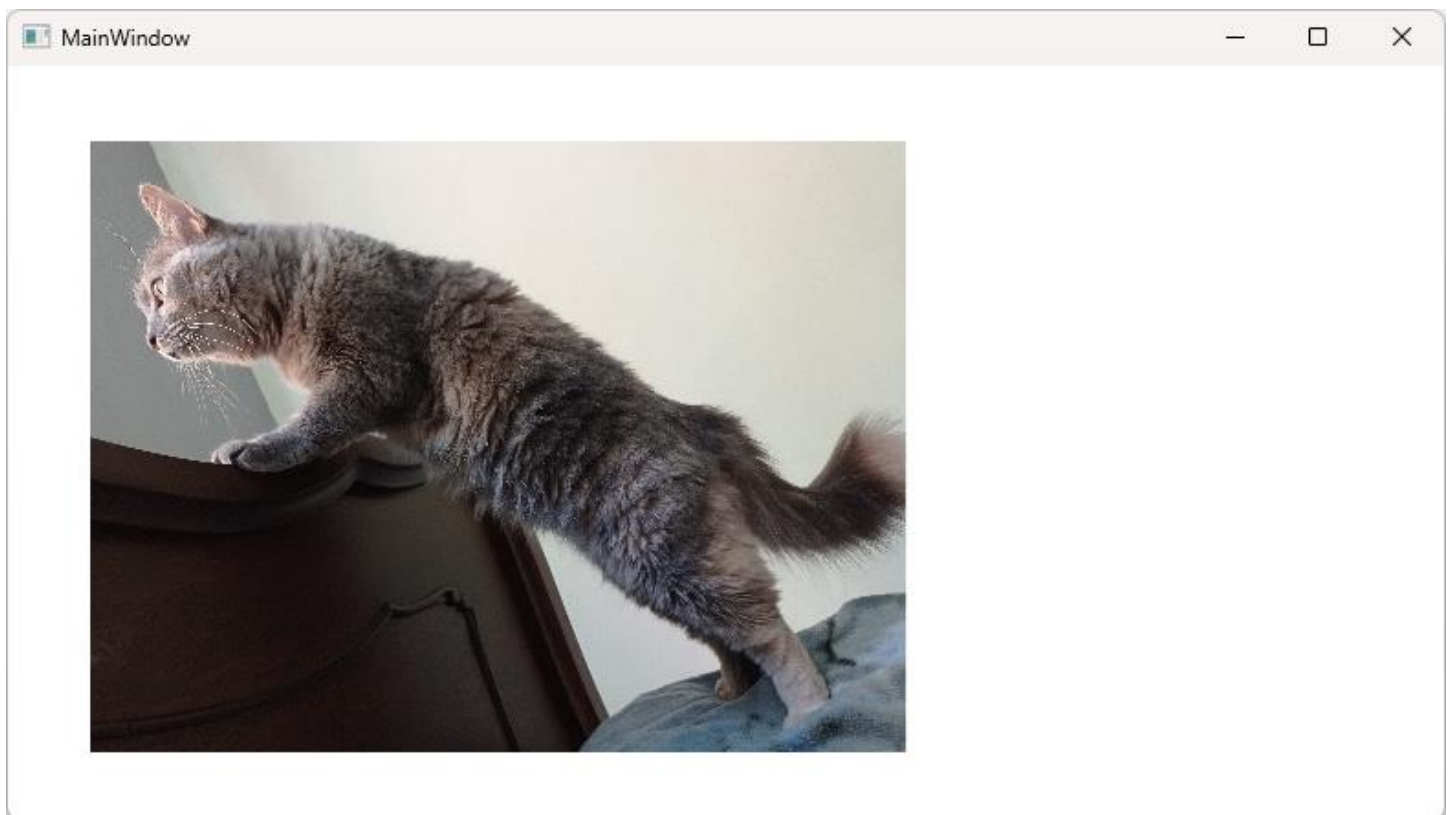


Ilustración 23: Aplicación WPF que carga una imagen

Una aplicación WPF que aplica varios filtros

Se muestra una aplicación en WPF que permite al usuario cargar una imagen (bmp, jpg, jpeg, gif, webp, tga) y luego aplicarle varios filtros, inclusive combinándolos. El proyecto completo se puede descargar en [O/021.7z](#)

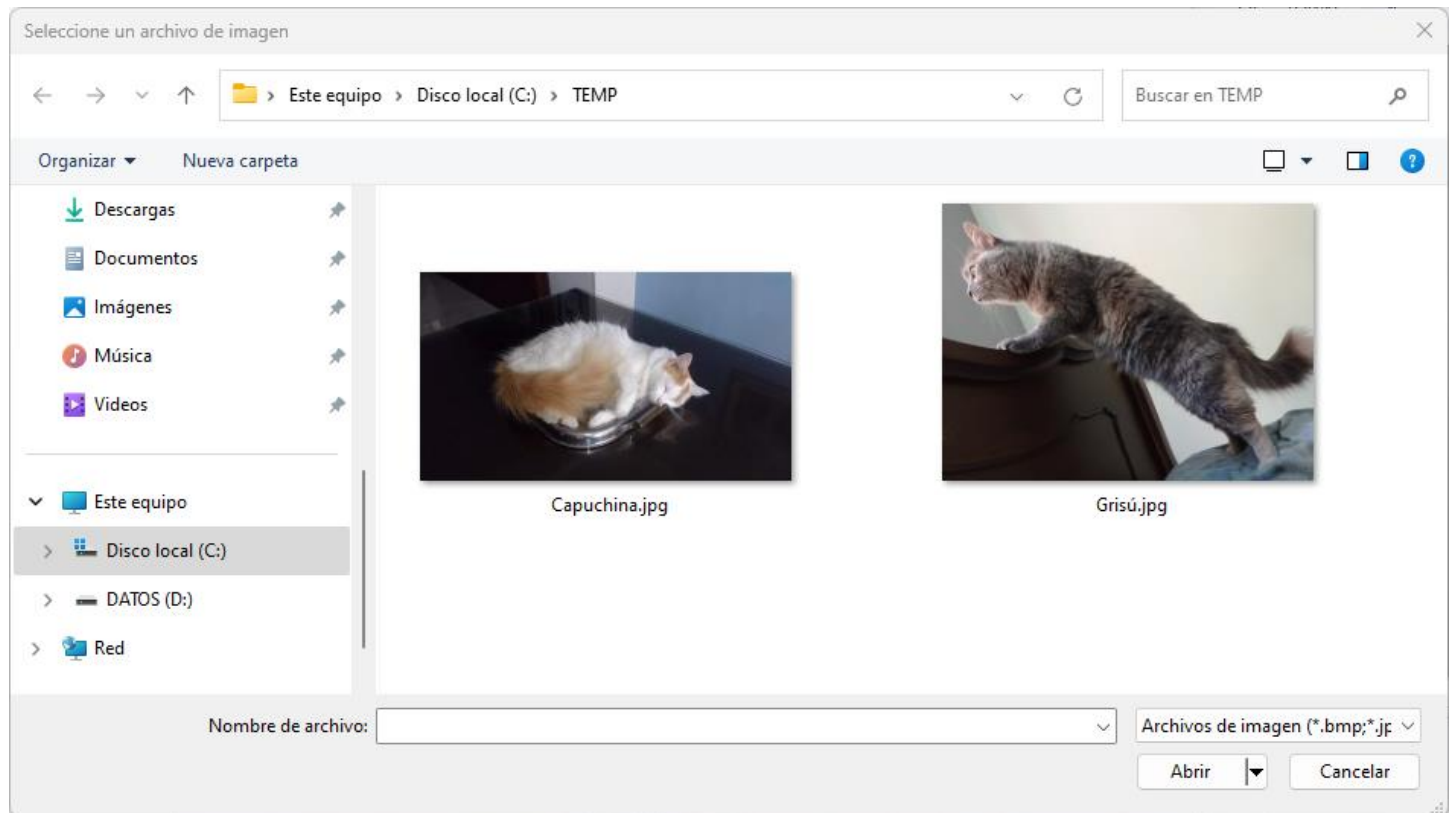


Ilustración 24: Una ventana de diálogo para escoger el archivo de imagen

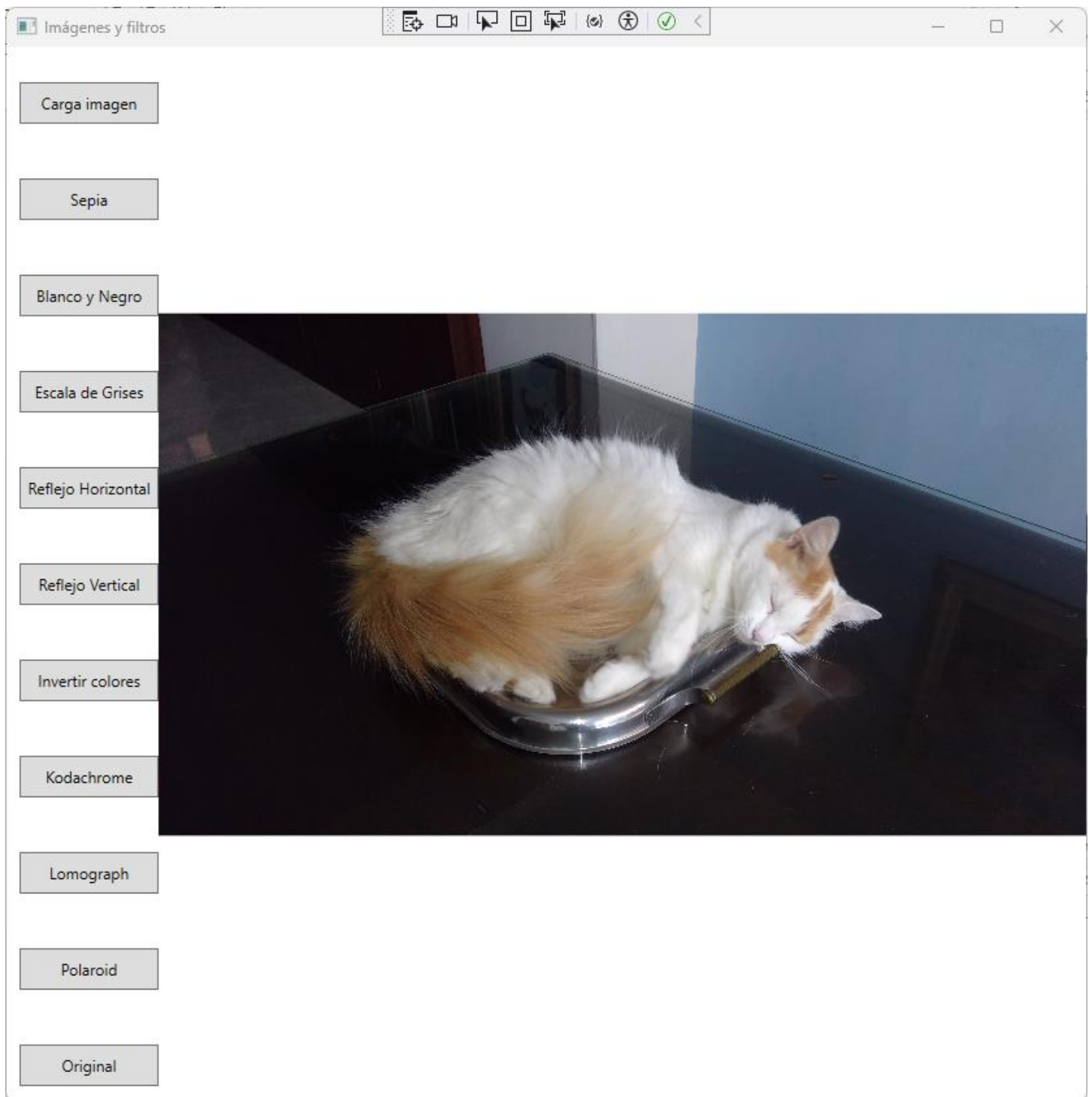


Ilustración 25: Imagen cargada

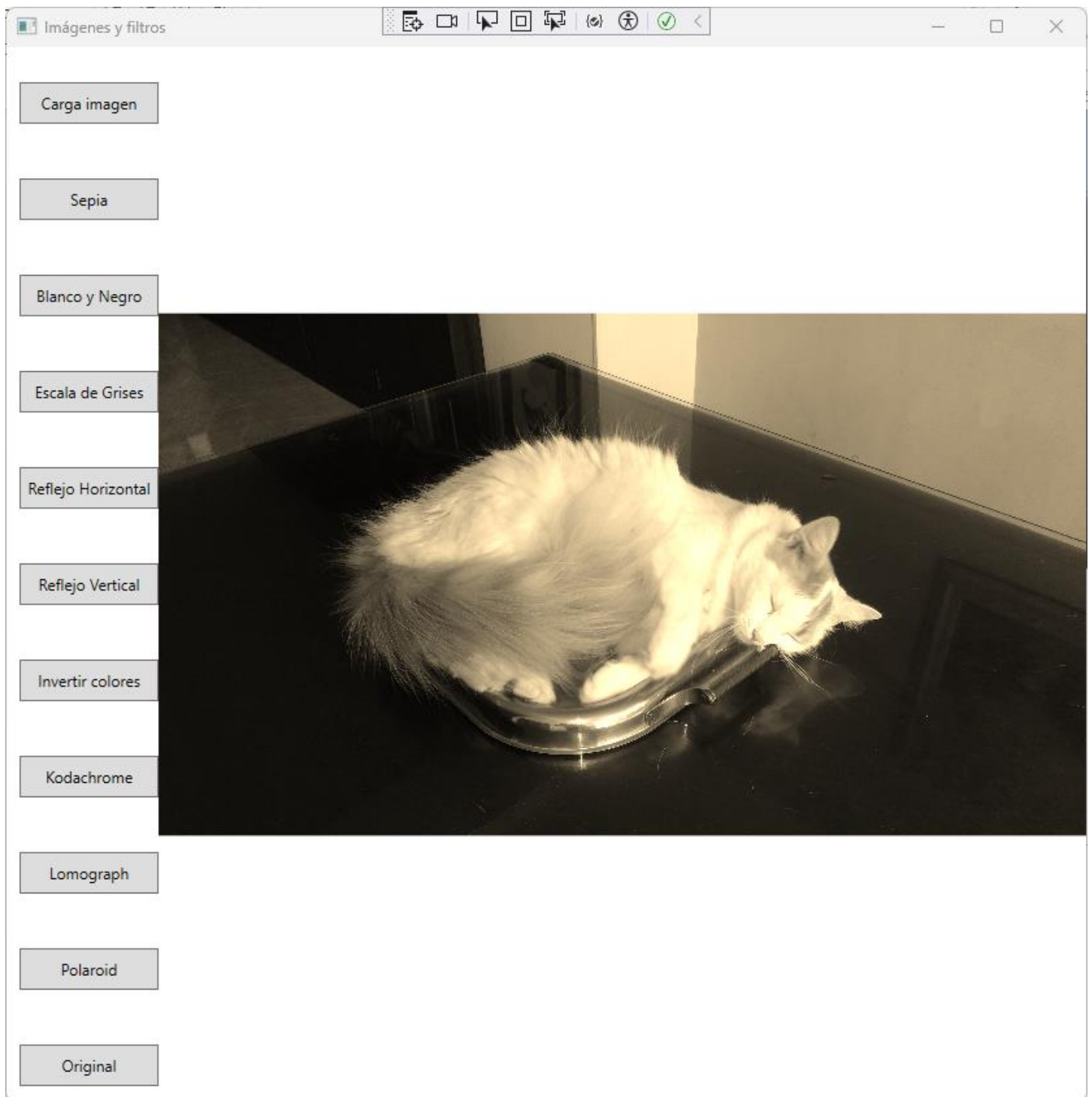


Ilustración 26: Aplicando el filtro sepia

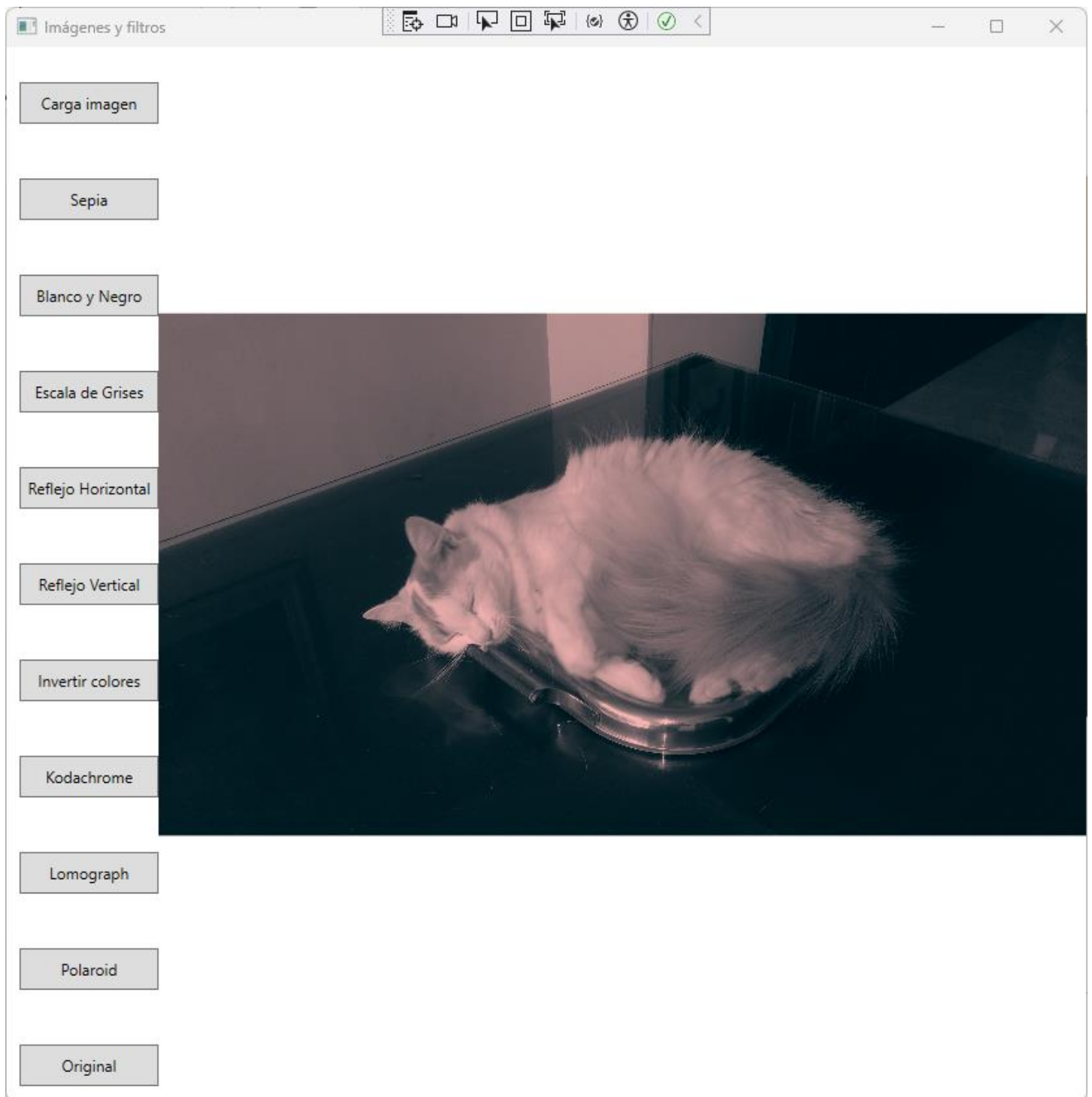


Ilustración 27: Aplicando reflejo horizontal y doble vez Kodachrome

```

using Microsoft.Win32;
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;
using System.IO;
using System.Windows;
using System.Windows.Media.Imaging;

namespace SharpImagen {
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window {
        SixLabors.ImageSharp.Image Foto, Copia;

        public MainWindow() {
            InitializeComponent();
        }

        //Carga la imagen y la muestra en el control
        private void MuestraFoto() {
            var FlujoMemoria = new MemoryStream();
            Copia.SaveAsBmp(FlujoMemoria);

            /* Mover la posición actual dentro del flujo de memoria
               (FlujoMemoria) a un punto específico.
               FlujoMemoria es usado para almacenar temporalmente la imagen
               en memoria.
               Seek(0, SeekOrigin.Begin): Es el método que mueve la
               posición actual dentro del flujo.
               Este método toma dos parámetros:
               0: Este es el desplazamiento en bytes desde la posición
                  especificada por el segundo parámetro.
                  En este caso, 0 significa que no se está moviendo
                  desde la posición especificada, simplemente se está
                  estableciendo la posición en el inicio del flujo.
               SeekOrigin.Begin: Este es un enumerador que especifica
                  el punto de referencia desde el cual
                  se calcula la nueva posición.
                  SeekOrigin.Begin indica que el punto
                  de referencia es el comienzo del flujo. */
            FlujoMemoria.Seek(0, SeekOrigin.Begin);

            var bitmap = new BitmapImage();

```

```

/* BeginInit() marca el inicio de un bloque de inicialización
 * para el objeto BitmapImage. Durante este bloque, se puede
 * establecer varias propiedades del BitmapImage sin que el
 * objeto intente cargarse o procesarse inmediatamente */
bitmap.BeginInit();
bitmap.StreamSource = FlujoMemoria;

/* Se utiliza para configurar cómo se almacena en caché la
 imagen cuando se carga en un objeto BitmapImage.
 bitmap: Es el objeto BitmapImage que se está utilizando
 para mostrar la imagen en la aplicación WPF.
 CacheOption: Es una propiedad del BitmapImage que determina
 cómo se almacena en caché la imagen.
 Esto afecta el rendimiento y el uso de memoria
 de la aplicación.
 BitmapCacheOption.OnLoad: Es uno de los valores del enumerador
 BitmapCacheOption. Especifica que la
 imagen debe cargarse completamente en
 memoria cuando se llama al método
 EndInit(). Esto significa que toda
 la imagen se carga y se almacena en
 memoria de una vez.
 */
bitmap.CacheOption = BitmapCacheOption.OnLoad;

/* En este punto, el BitmapImage se carga y se procesa utilizando
 * las propiedades que se establecieron durante el bloque
 * de inicialización. */
bitmap.EndInit();
ImagenFoto.Source = bitmap;

}

private void btnSepia_Click(object sender, RoutedEventArgs e) {
    Copia.Mutate(x => x.Sepia());
    MuestraFoto();
}

private void btnBlancoNegro_Click(object sender, RoutedEventArgs e) {
    Copia.Mutate(x => x.BlackWhite());
    MuestraFoto();
}

private void btnGrisés_Click(object sender, RoutedEventArgs e) {
    Copia.Mutate(x => x.Grayscale());
    MuestraFoto();
}

```

```

private void btnOriginal_Click(object sender, RoutedEventArgs e) {
    Copia = Foto.CloneAs<Rgba32>();
    MuestraFoto();
}

private void btnReflejoH_Click(object sender, RoutedEventArgs e) {
    Copia.Mutate(x => x.Flip(FlipMode.Horizontal));
    MuestraFoto();
}

private void btnReflejoV_Click(object sender, RoutedEventArgs e) {
    Copia.Mutate(x => x.Flip(FlipMode.Vertical));
    MuestraFoto();
}

private void btnInvierte_Click(object sender, RoutedEventArgs e) {
    Copia.Mutate(x => x.Invert());
    MuestraFoto();
}

private void btnKodachrome_Click(object sender, RoutedEventArgs e) {
    Copia.Mutate(x => x.Kodachrome());
    MuestraFoto();
}

private void btnLomograph_Click(object sender, RoutedEventArgs e) {
    Copia.Mutate(x => x.Lomograph());
    MuestraFoto();
}

private void btnPolaroid_Click(object sender, RoutedEventArgs e) {
    Copia.Mutate(x => x.Polaroid());
    MuestraFoto();
}

private void btnCarga_Click(object sender, RoutedEventArgs e) {
    OpenFileDialog dlgAbrir = new OpenFileDialog();
    dlgAbrir.Filter = "Archivos de
imagen|*.bmp;*.jpg;*.jpeg;*.gif;*.webp;*.tga";
    dlgAbrir.Title = "Seleccione un archivo de imagen";

    if (dlgAbrir.ShowDialog() == true) {
        try {
            Foto = SixLabors.ImageSharp.Image.Load<Rgba32>(dlgAbrir.FileName);
            Copia = Foto.CloneAs<Rgba32>();
            MuestraFoto();
            btnBlancoNegro.IsEnabled = true;
            btnGrises.IsEnabled = true;
        }
    }
}

```

```

        btnSepia.IsEnabled = true;
        btnOriginal.IsEnabled = true;
        btnReflejoH.IsEnabled = true;
        btnReflejoV.IsEnabled = true;
        btnInvierte.IsEnabled = true;
        btnKodachrome.IsEnabled = true;
        btnLomograph.IsEnabled = true;
        btnPolaroid.IsEnabled = true;
    }
    catch (Exception ex) {
        MessageBox.Show($"Error al cargar la imagen: {ex.Message}");
    }
}
}
}
}
}

```

O/021.xaml debe renombrarse a MainWindow.xaml

```

<Window x:Class="SharpImagen.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:SharpImagen"
    mc:Ignorable="d"
    Title="Imágenes y filtros" Height="850" Width="800"
    WindowStartupLocation="CenterScreen" WindowState="Maximized">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto"></ColumnDefinition>
            <ColumnDefinition Width="*"></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <Image x:Name="ImagenFoto" Grid.Column="1"/>
        <Button x:Name="btnCarga" Content="Carga imagen"
            HorizontalAlignment="Left" Height="32" Margin="10,27,0,0"
            VerticalAlignment="Top" Width="107" Click="btnCarga_Click"/>
        <Button x:Name="btnSepia" Content="Sepia" HorizontalAlignment="Left"
            Height="32" Margin="10,101,0,0" VerticalAlignment="Top" Width="107"
            IsEnabled="False" Click="btnSepia_Click"/>
        <Button x:Name="btnBlancoNegro" Content="Blanco y Negro"
            HorizontalAlignment="Left" Height="32" Margin="10,175,0,0"
            VerticalAlignment="Top" Width="107" IsEnabled="False"
            Click="btnBlancoNegro_Click"/>
        <Button x:Name="btnGris" Content="Escala de Grises"
            HorizontalAlignment="Left" Height="32" Margin="10,249,0,0"

```



```

VerticalAlignment="Top" Width="107" IsEnabled="False"
Click="btnGrises_Click"/>
    <Button x:Name="btnReflejoH" Content="Reflejo Horizontal"
HorizontalAlignment="Left" Height="32" Margin="10,323,0,0"
VerticalAlignment="Top" Width="107" IsEnabled="False"
Click="btnReflejoH_Click"/>
    <Button x:Name="btnReflejoV" Content="Reflejo Vertical"
HorizontalAlignment="Left" Height="32" Margin="10,397,0,0"
VerticalAlignment="Top" Width="107" IsEnabled="False"
Click="btnReflejoV_Click"/>
    <Button x:Name="btnInvierte" Content="Invertir colores"
HorizontalAlignment="Left" Height="32" Margin="10,471,0,0"
VerticalAlignment="Top" Width="107" IsEnabled="False"
Click="btnInvierte_Click"/>
    <Button x:Name="btnKodachrome" Content="Kodachrome"
HorizontalAlignment="Left" Height="32" Margin="10,545,0,0"
VerticalAlignment="Top" Width="107" IsEnabled="False"
Click="btnKodachrome_Click"/>
    <Button x:Name="btnLomograph" Content="Lomograph"
HorizontalAlignment="Left" Height="32" Margin="10,619,0,0"
VerticalAlignment="Top" Width="107" IsEnabled="False"
Click="btnLomograph_Click"/>
    <Button x:Name="btnPolaroid" Content="Polaroid"
HorizontalAlignment="Left" Height="32" Margin="10,693,0,0"
VerticalAlignment="Top" Width="107" IsEnabled="False"
Click="btnPolaroid_Click"/>
    <Button x:Name="btnOriginal" Content="Original"
HorizontalAlignment="Left" Height="32" Margin="10,767,0,0"
VerticalAlignment="Top" Width="107" IsEnabled="False"
Click="btnOriginal_Click"/>
</Grid>
</Window>

```