

C# Y .NET 10

Parte 15. Imágenes

2026-01

Rafael Alberto Moreno Parra
ramsoftware@gmail.com

Contenido

Tabla de ilustraciones.....	4
Acerca del autor.....	6
Licencia de este libro	6
Licencia del software	6
Marcas registradas	7
Dedicatoria	8
Iniciando con imágenes.....	9
Convertir a escala de grises	20
Disminuir el tamaño de la imagen	21
Reflejo Horizontal.....	23
Reflejo Vertical	24
Giro de la imagen.....	25
Filtro blanco y negro.....	26
Invierte colores.....	27
Filtro Kodachrome	28
Filtro Sepia	29
Filtro Lomograph.....	30
Filtro Polaroid	31
Filtro umbral	32
Filtro borde	33
Filtro pixelado.....	34
Filtro de saturación de color	35
Filtro viñeta.....	36
Cambiar pixel a pixel	37
Invertir colores	38
Aplicando un núcleo o “kernel”	40
Una aplicación WPF que carga una imagen	42
Una aplicación WPF que aplica varios filtros	53
Información imágenes	62
Dibujando en imágenes.....	63
Dibujar una línea	66
Dibujar un círculo	68
Dibujar un polígono	70

Dibujar Líneas.....	72
Dibujar Círculos	74
Dibujar rectángulos al azar	76
Dibujar letras.....	78
Texto girado	80

Tabla de ilustraciones

Ilustración 1: Inicia proyecto en Visual Studio 2026	9
Ilustración 2: Aplicación de consola	10
Ilustración 3: Ponerle un nombre al proyecto	11
Ilustración 4: Seleccionar .NET 10	12
Ilustración 5: Aplicación de consola de ejemplo	13
Ilustración 6: Dar clic en "Herramientas" y luego en "Administrador de paquetes NuGet"	14
Ilustración 7: Instalación de SixLabors.ImageSharp	15
Ilustración 8: Se instalan las librerías en el proyecto	16
Ilustración 9: Versión de la librería	17
Ilustración 10: Licencia de la librería	18
Ilustración 11: Paquete instalado	19
Ilustración 12: Escala de grises	20
Ilustración 13: Cambio de tamaño de imagen	22
Ilustración 14: Reflejo horizontal	23
Ilustración 15: Reflejo vertical	24
Ilustración 16: Giro de la imagen	25
Ilustración 17: Filtro blanco y negro	26
Ilustración 18: Invierte colores	27
Ilustración 19: Filtro Kodachrome	28
Ilustración 20: Filtro Sepia	29
Ilustración 21: Filtro Lomograph	30
Ilustración 22: Filtro Polaroid	31
Ilustración 23: Filtro umbral	32
Ilustración 24: Filtro borde	33
Ilustración 25: Filtro pixelado	34
Ilustración 26: Filtro de saturación de color	35
Ilustración 27: Filtro viñeta	36
Ilustración 28: Modificando pixel a pixel	37
Ilustración 29: Invertir colores	39
Ilustración 30: Aplicando un núcleo	41
Ilustración 31: Aplicación WPF	42
Ilustración 32: Nombre del proyecto	43
Ilustración 33: Seleccionar .NET 10	44
Ilustración 34: Aplicación WPF típica	45
Ilustración 35: Código en C#	46
Ilustración 36: Carga las librerías	47
Ilustración 37: Se pone el objeto "Image" en la ventana	48
Ilustración 38: Se pone nombre a ese objeto "Image"	49
Ilustración 39: Ejecución del proyecto	52
Ilustración 40: Una ventana de diálogo para escoger el archivo de imagen	53
Ilustración 41: Imagen cargada	54
Ilustración 42: Aplicando el filtro sepia	55
Ilustración 43: Aplicando reflejo horizontal y doble vez Kodachrome	56
Ilustración 44: Información de la imagen	62
Ilustración 45: Se agrega SixLabors.ImageSharp.Drawing	63

Ilustración 46: Agregar Drawing	64
Ilustración 47: Licencia de esa librería	65
Ilustración 48: Dibuja una línea en imagen.....	67
Ilustración 49: Dibuja círculo en imagen	69
Ilustración 50: Dibuja un polígono dentro de una imagen	71
Ilustración 51: Dibuja líneas en una imagen	73
Ilustración 52: Dibuja círculos en una imagen	75
Ilustración 53: Rectángulos rellenos al azar	77

Acerca del autor

Rafael Alberto Moreno Parra

ramsoftware@gmail.com o enginelifemail@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 2026 ® Enlace: <https://visualstudio.microsoft.com/es/vs/>

Dedicatoria

A mis padres, a mi hermana....

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

Iniciando con imágenes

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

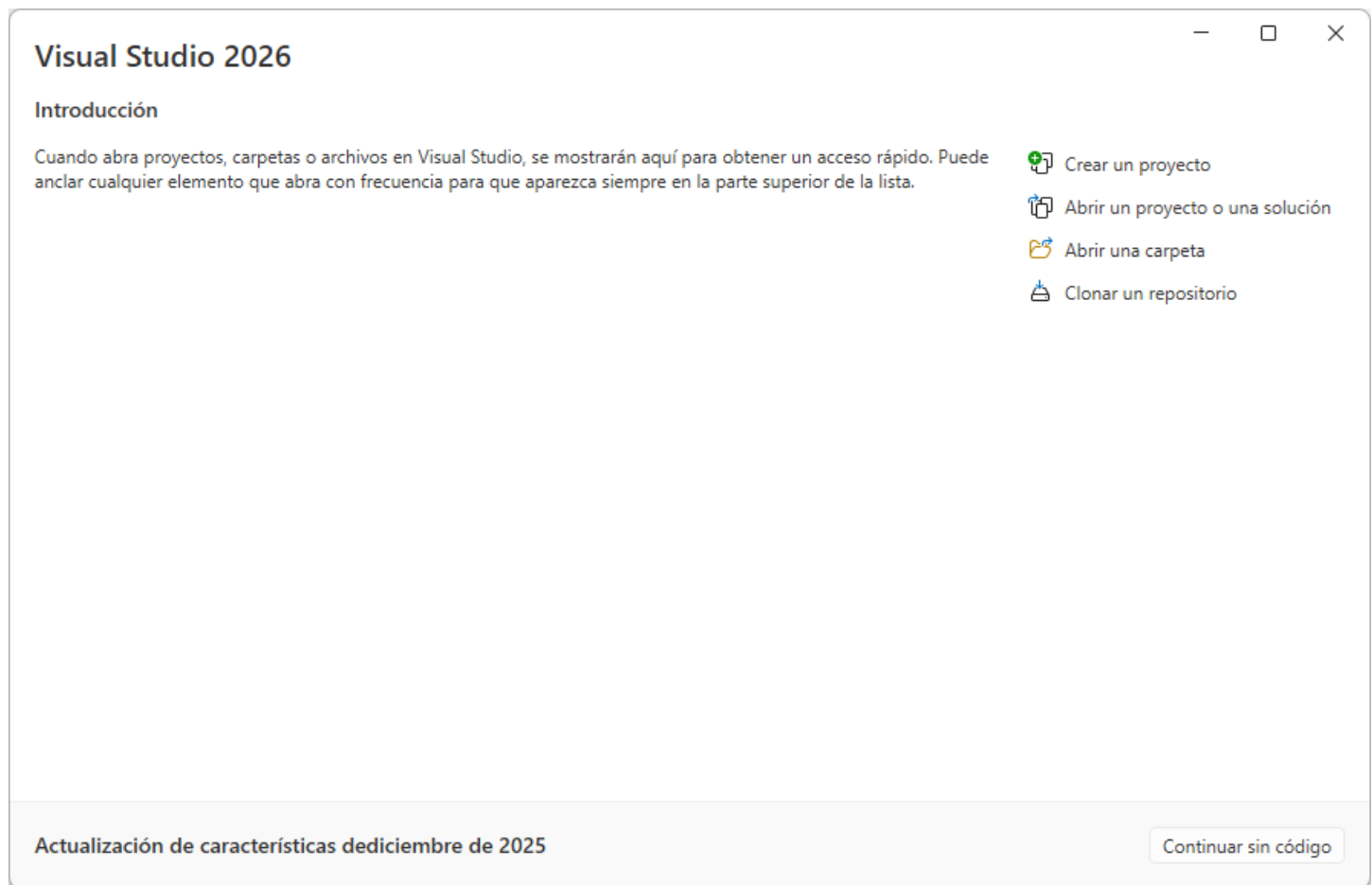


Ilustración 1: Inicia proyecto en Visual Studio 2026

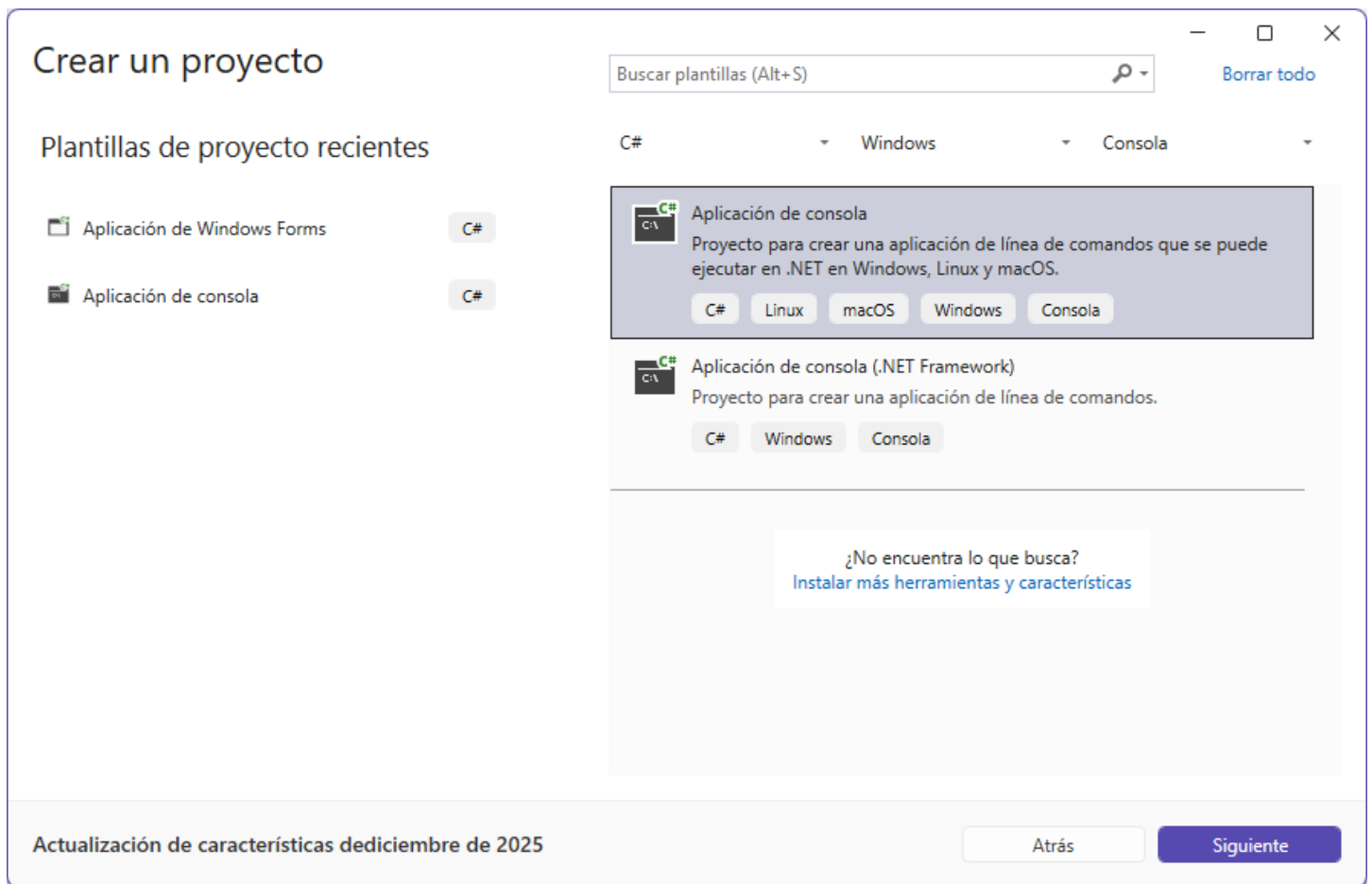


Ilustración 2: Aplicación de consola

—□×

Configure su nuevo proyecto

Aplicación de consola

C#LinuxmacOSWindowsConsola

Nombre del proyecto

Ejemplo

Ubicación

C:\Users\engin\source\repos

...

Nombre de la solución ⓘ

Ejemplo

☐ Colocar la solución y el proyecto en el mismo directorio

Proyecto se creará en "C:\Users\engin\source\repos\Ejemplo\Ejemplo"

Actualización de características dediciembre de 2025

Atrás

Siguiente

Ilustración 3: Ponerle un nombre al proyecto

—□×

Información adicional

Aplicación de consola

C#LinuxmacOSWindowsConsola

Framework ⓘ

.NET 10.0 (Compatibilidad a largo plazo) ▾

☒ No usar instrucciones de nivel superior ⓘ
☐ Habilitar publicación de AOT native ⓘ

Actualización de características dediciembre de 2025

AtrásCrear

Ilustración 4: Seleccionar .NET 10

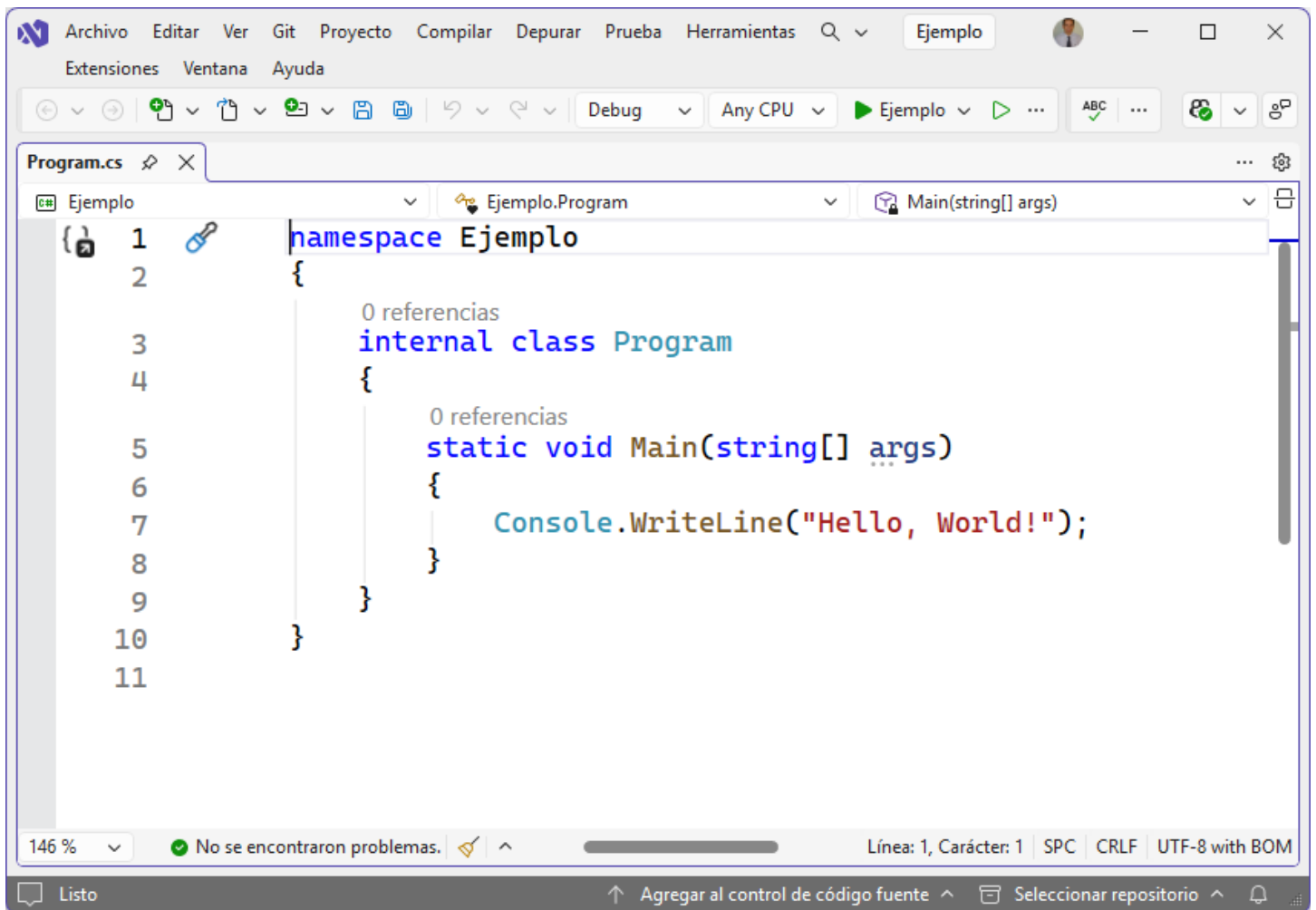


Ilustración 5: Aplicación de consola de ejemplo

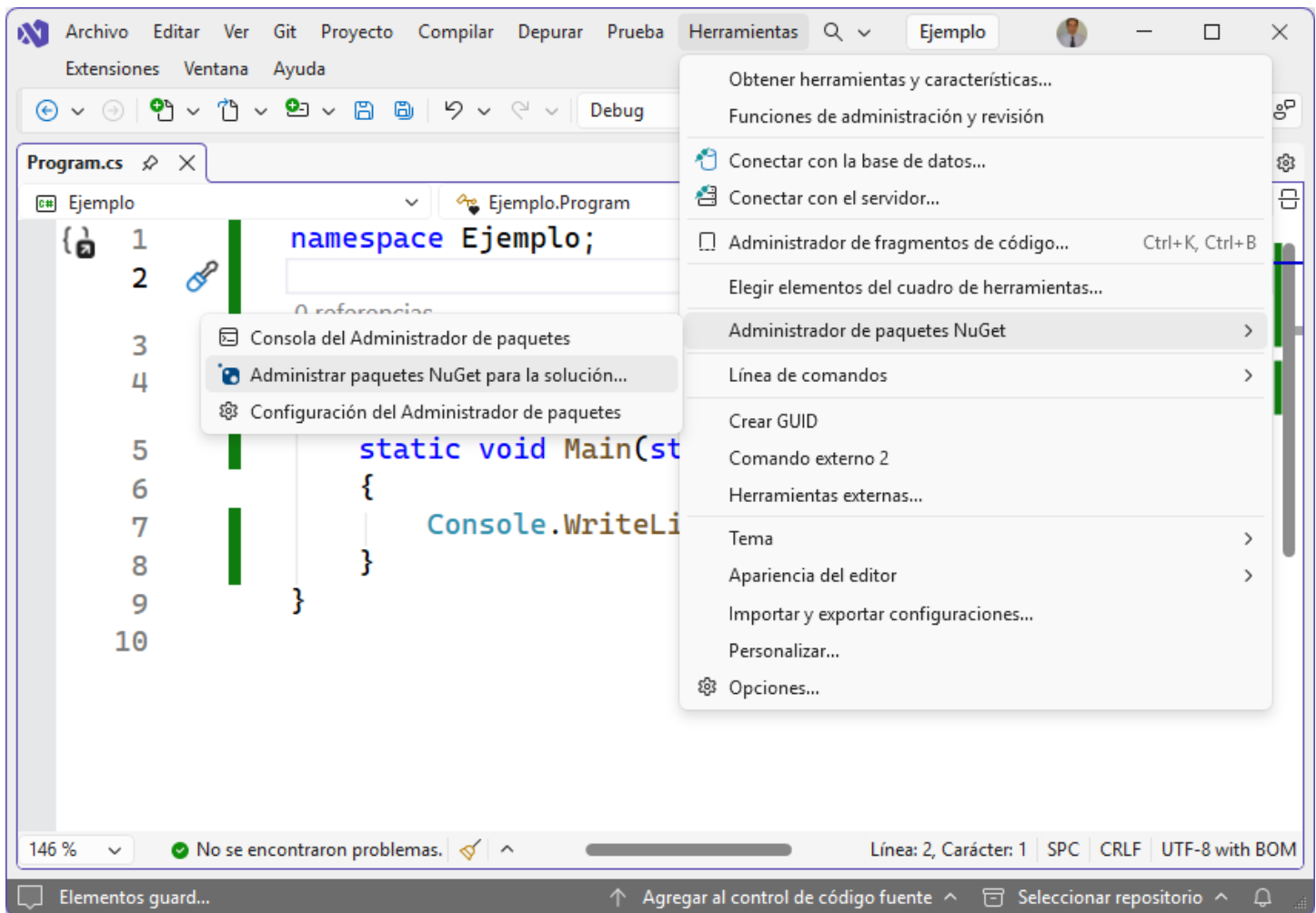


Ilustración 6: Dar clic en "Herramientas" y luego en "Administrador de paquetes NuGet"

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

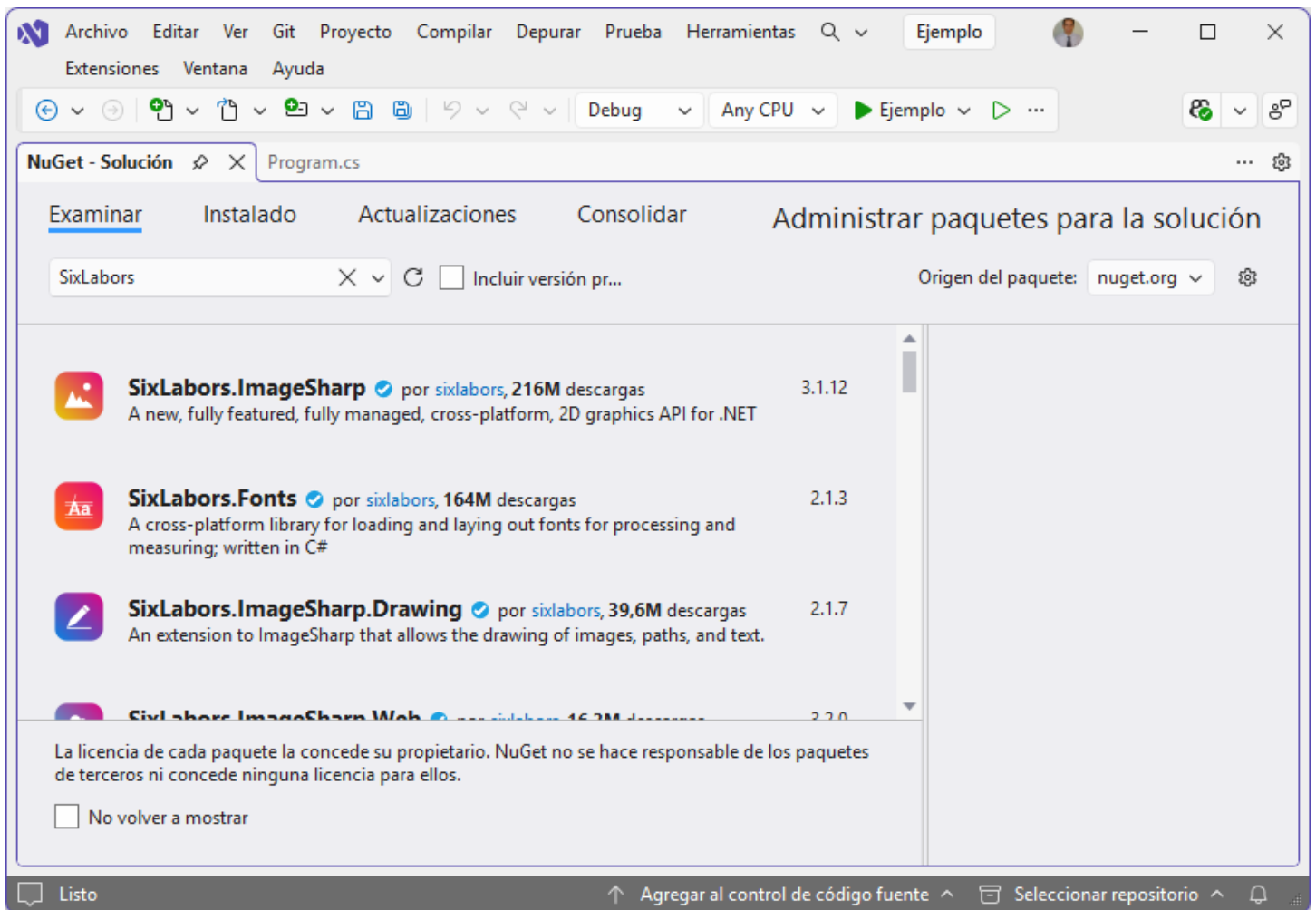


Ilustración 7: Instalación de SixLabors.ImageSharp

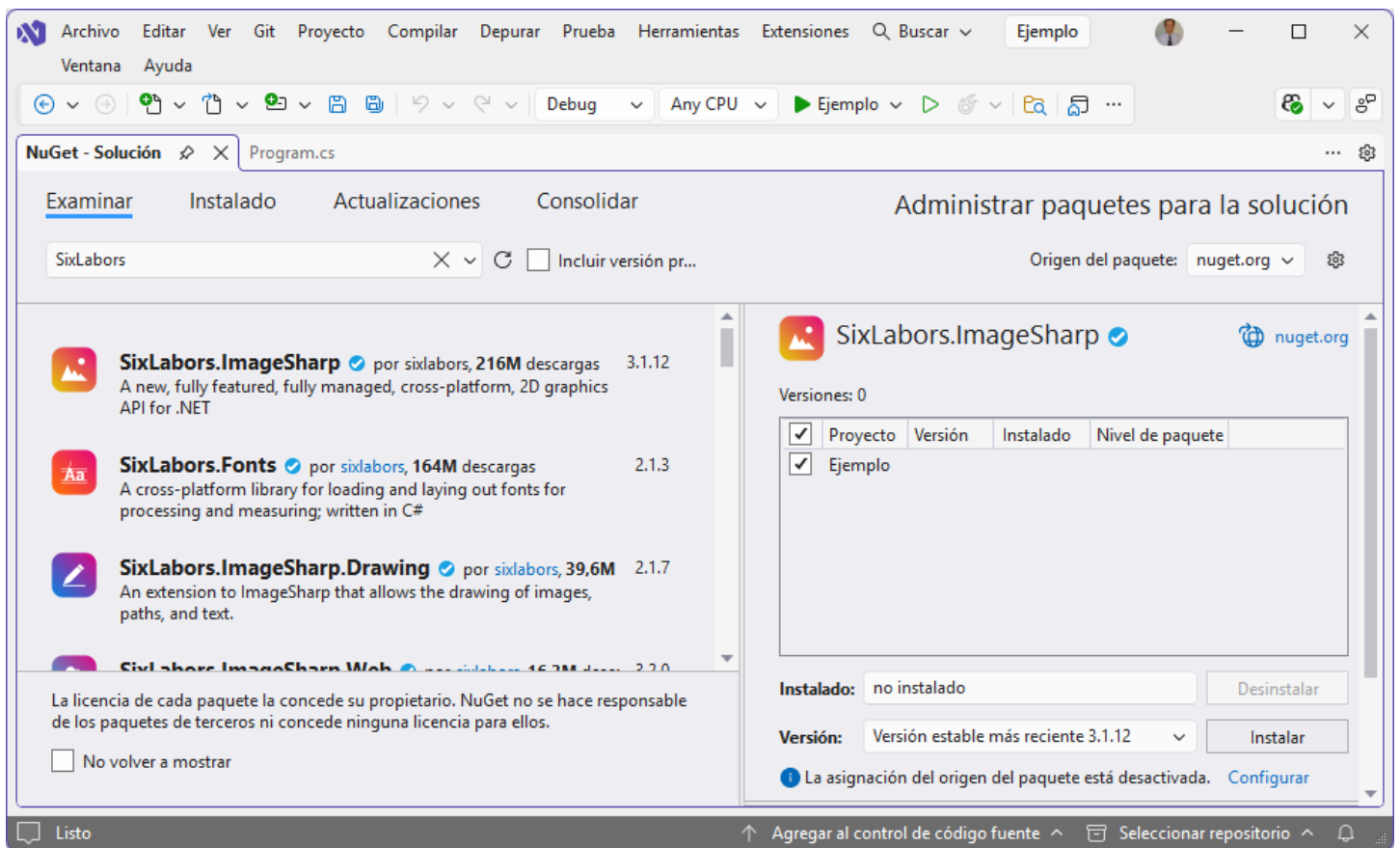


Ilustración 8: Se instalan las librerías en el proyecto

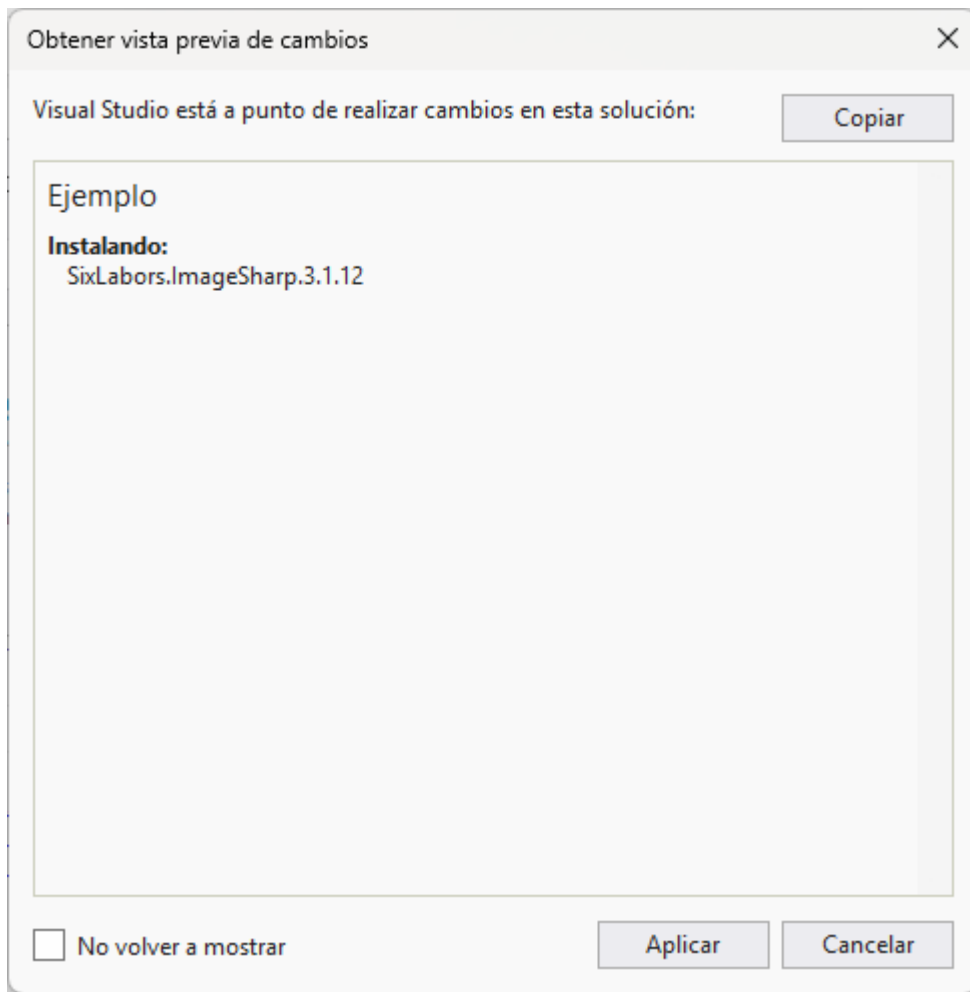


Ilustración 9: Versión de la librería

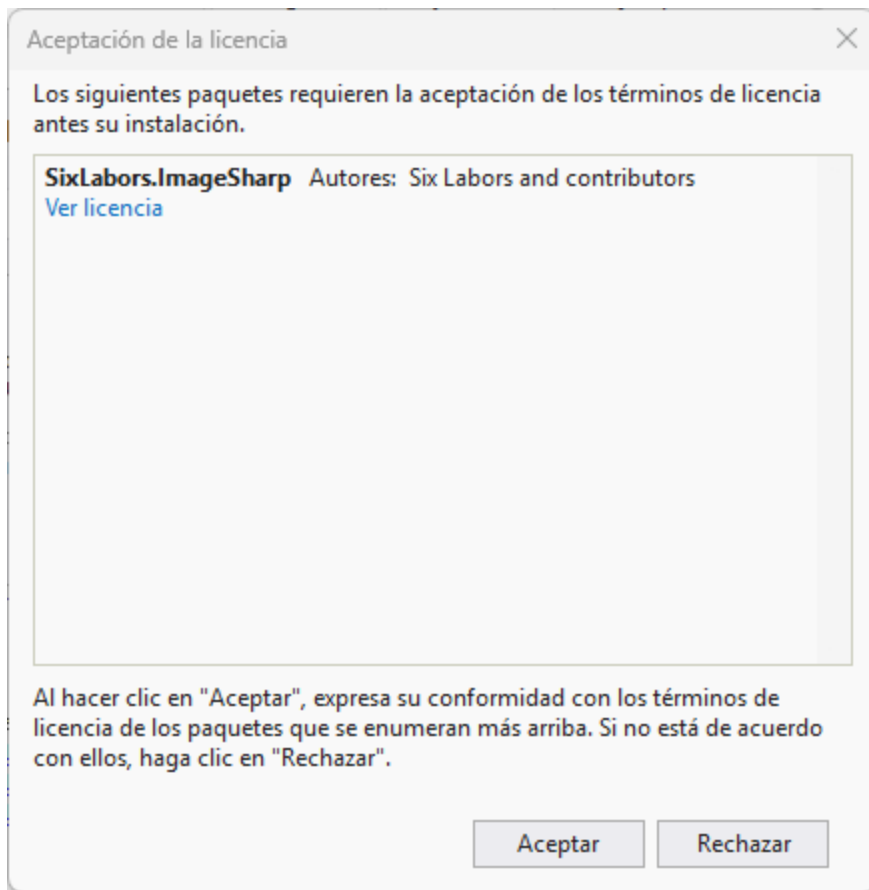


Ilustración 10: Licencia de la librería

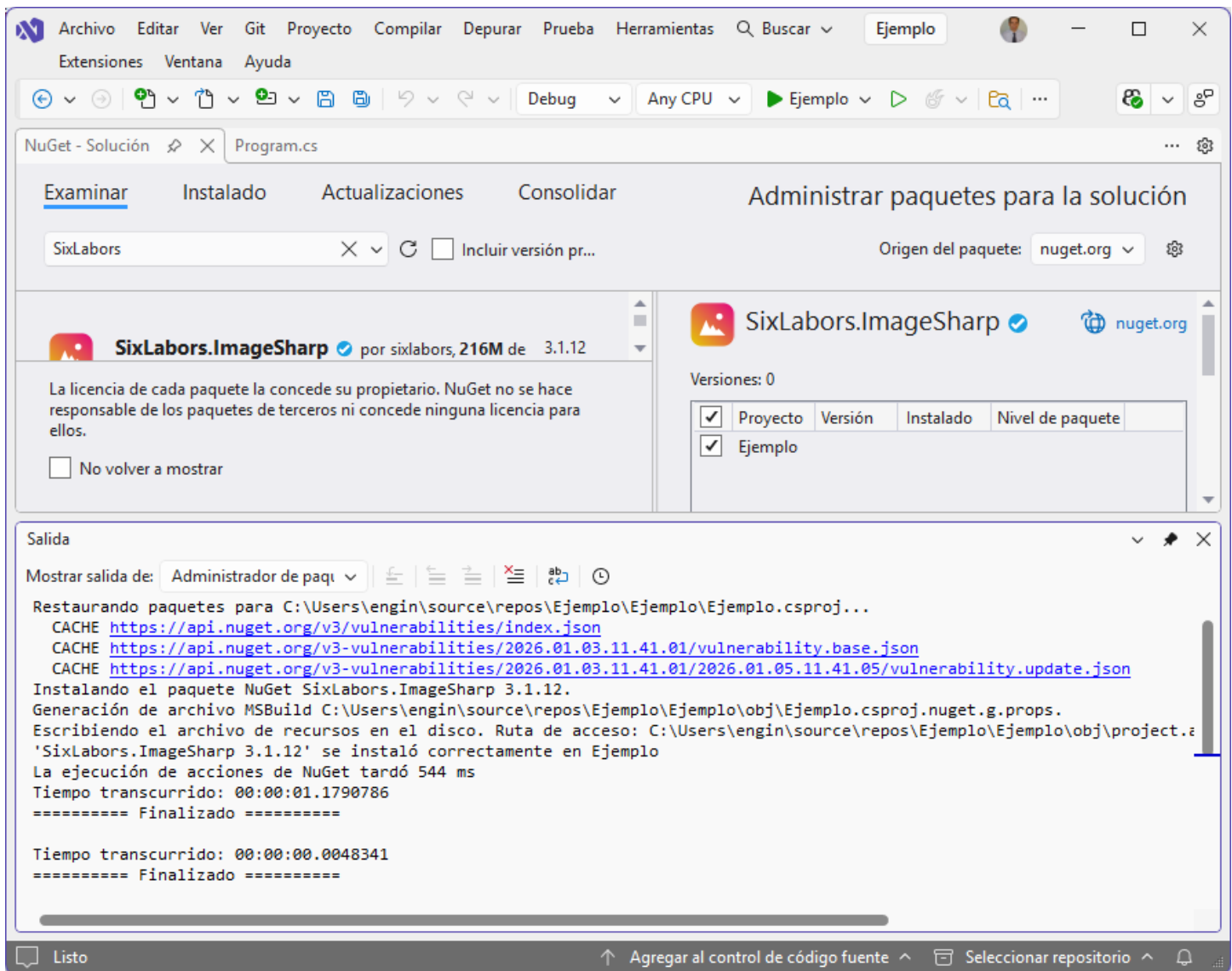


Ilustración 11: Paquete instalado

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.");
    }
}
```

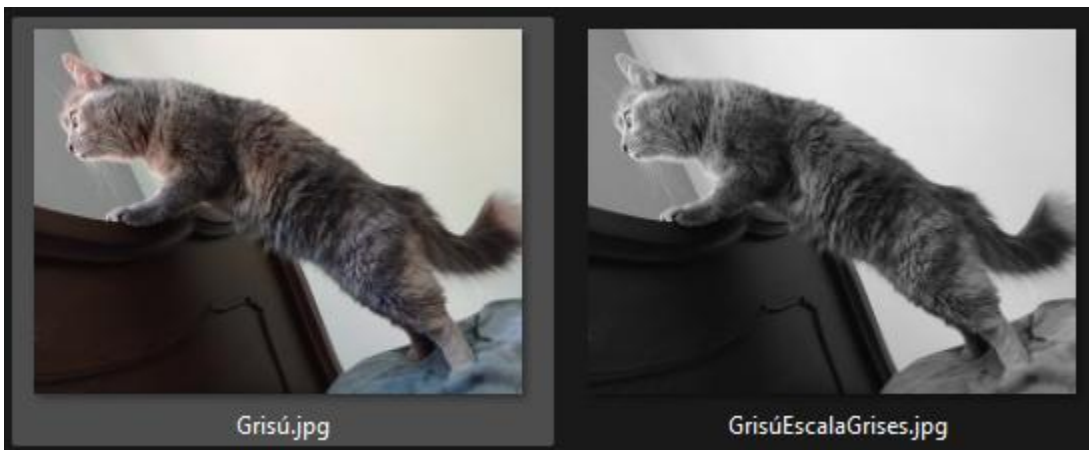


Ilustración 12: Escala de grises

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.");
    }
}
```

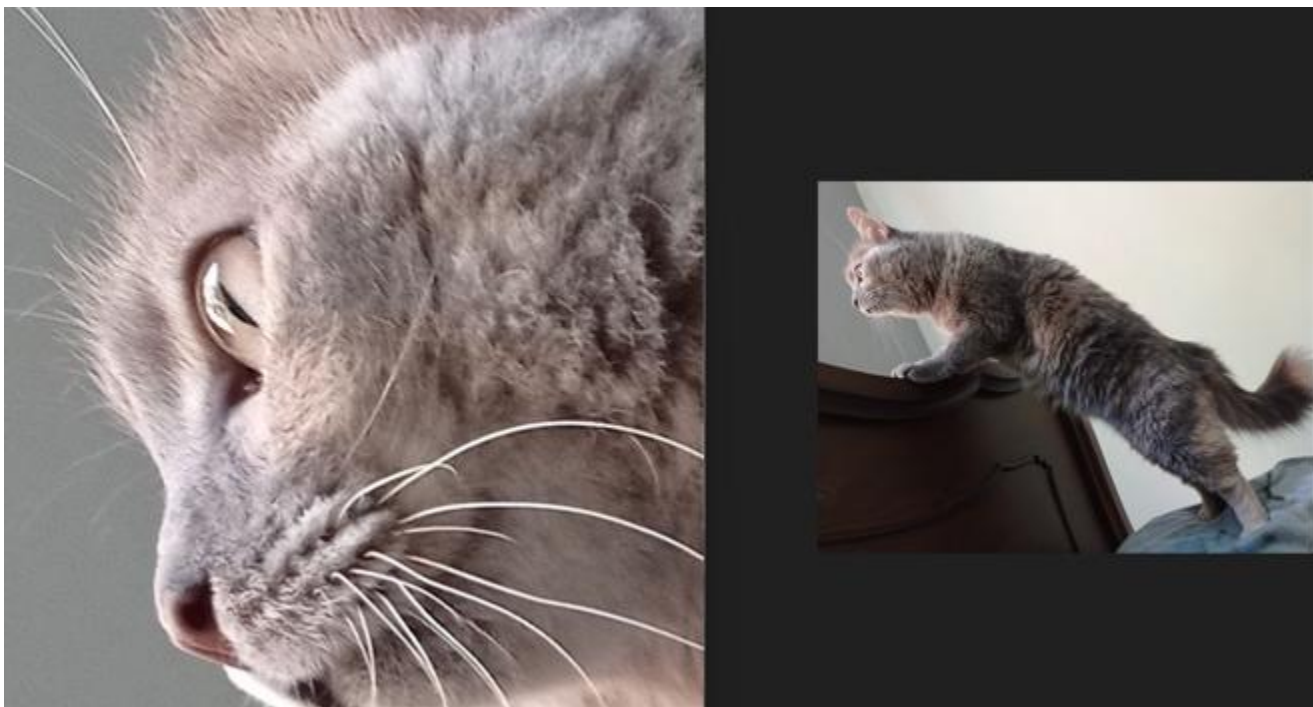


Ilustración 13: Cambio de tamaño de imagen

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 14: 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.");
    }
}
```

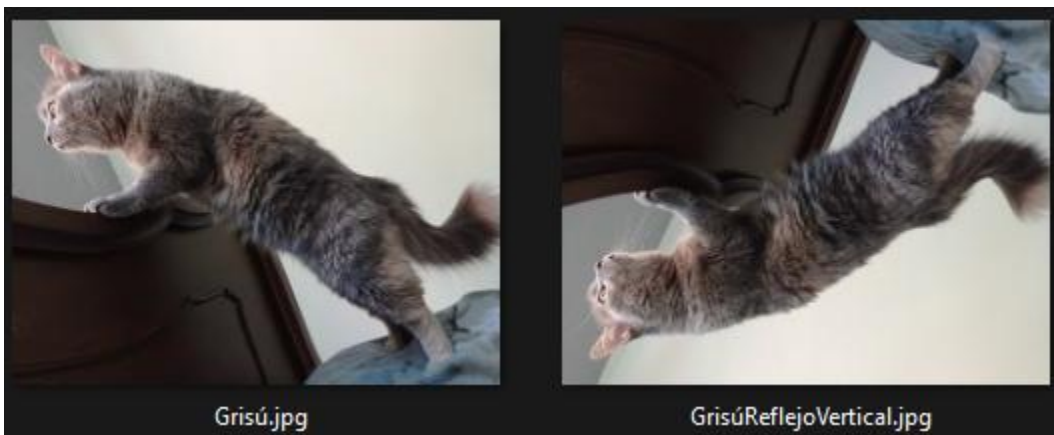


Ilustración 15: Reflejo vertical

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 16: 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.");
    }
}
```

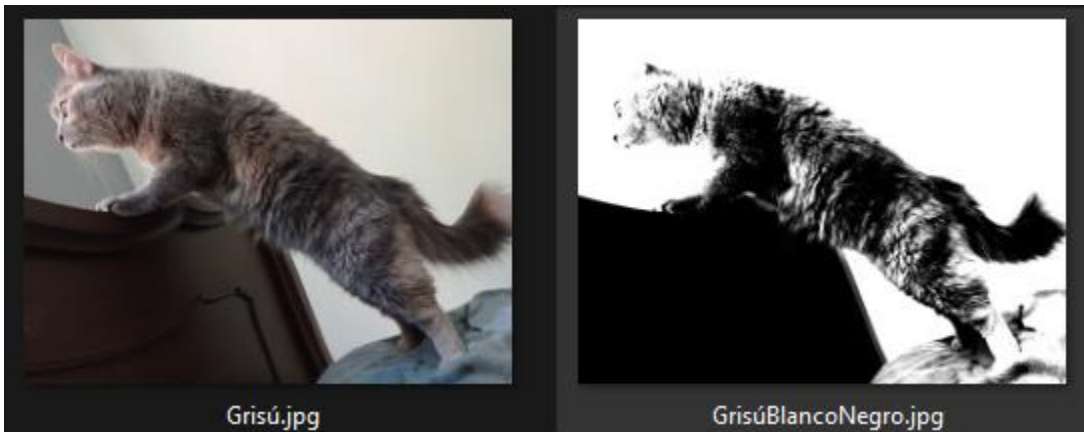


Ilustración 17: Filtro blanco y negro

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.");
    }
}
```

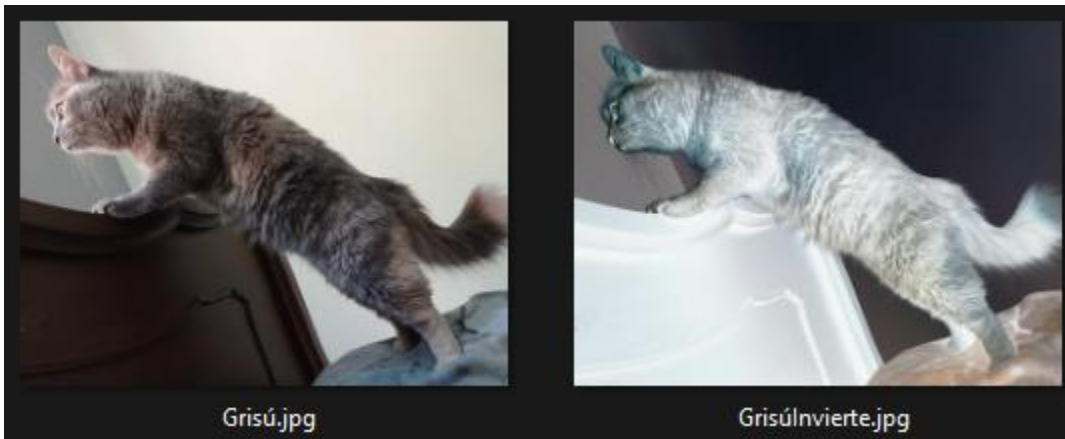


Ilustración 18: Invierte colores

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.");
    }
}
```

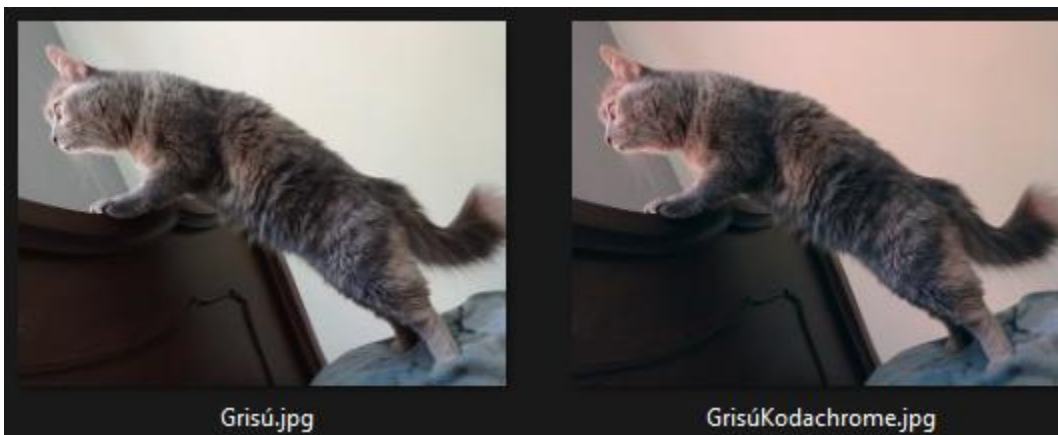


Ilustración 19: Filtro Kodachrome

```
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.");
    }
}
```



Grisú.jpg



GrisúSepia.jpg

Ilustración 20: Filtro Sepia

Filtro Lomograph

O/010.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 Lomograph
            Foto.Mutate(x => x.Lomograph());

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

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



Ilustración 21: 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.");
    }
}
```



Grisú.jpg



GrisúPolaroid.jpg

Ilustración 22: Filtro Polaroid

Filtro umbral

O/012.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 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 23: 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 24: 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 25: 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.");
    }
}
```



Grisú.jpg



GrisúSaturado.jpg

Ilustración 26: 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.");
    }
}
```



Grisú.jpg



GrisúVineta.jpg

Ilustración 27: 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");
    }
}
```



Grisú.jpg



GrisúPixelGris.jpg

Ilustración 28: 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 pixel 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");
    }
}
```



Grisú.jpg



GrisúInvierte.jpg

Ilustración 29: Invertir colores

Aplicando un núcleo o “kernel”

Un kernel es una matriz cuadrada pequeña (por ejemplo, 3×3 o 5×5) que contiene valores numéricos.

Se utiliza en la operación llamada convolución, donde cada píxel de la imagen se recalcula combinando los valores de sus vecinos según los coeficientes del kernel.

El resultado depende de los valores dentro del kernel: algunos resaltan bordes, otros suavizan, otros enfocan.

Tipo de operación	Matriz	Resultado
Identidad	$\begin{vmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{vmatrix}$	No cambia la imagen
Detección de bordes	$\begin{vmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{vmatrix}$	Resalta los bordes
Enfoque	$\begin{vmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{vmatrix}$	Aumenta nitidez

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];
        }
    }
}
```



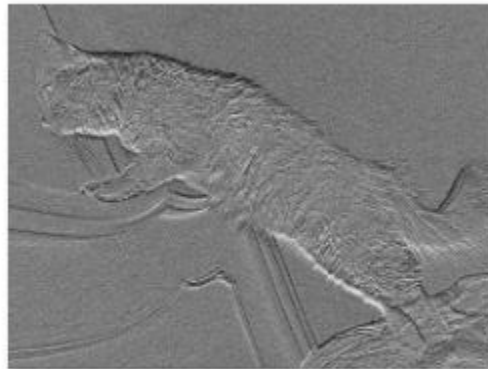
```

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");
}
}

```



Grisú.jpg



GrisúAplicaNucleo.jpg

Ilustración 30: Aplicando un núcleo

Una aplicación WPF que carga una imagen

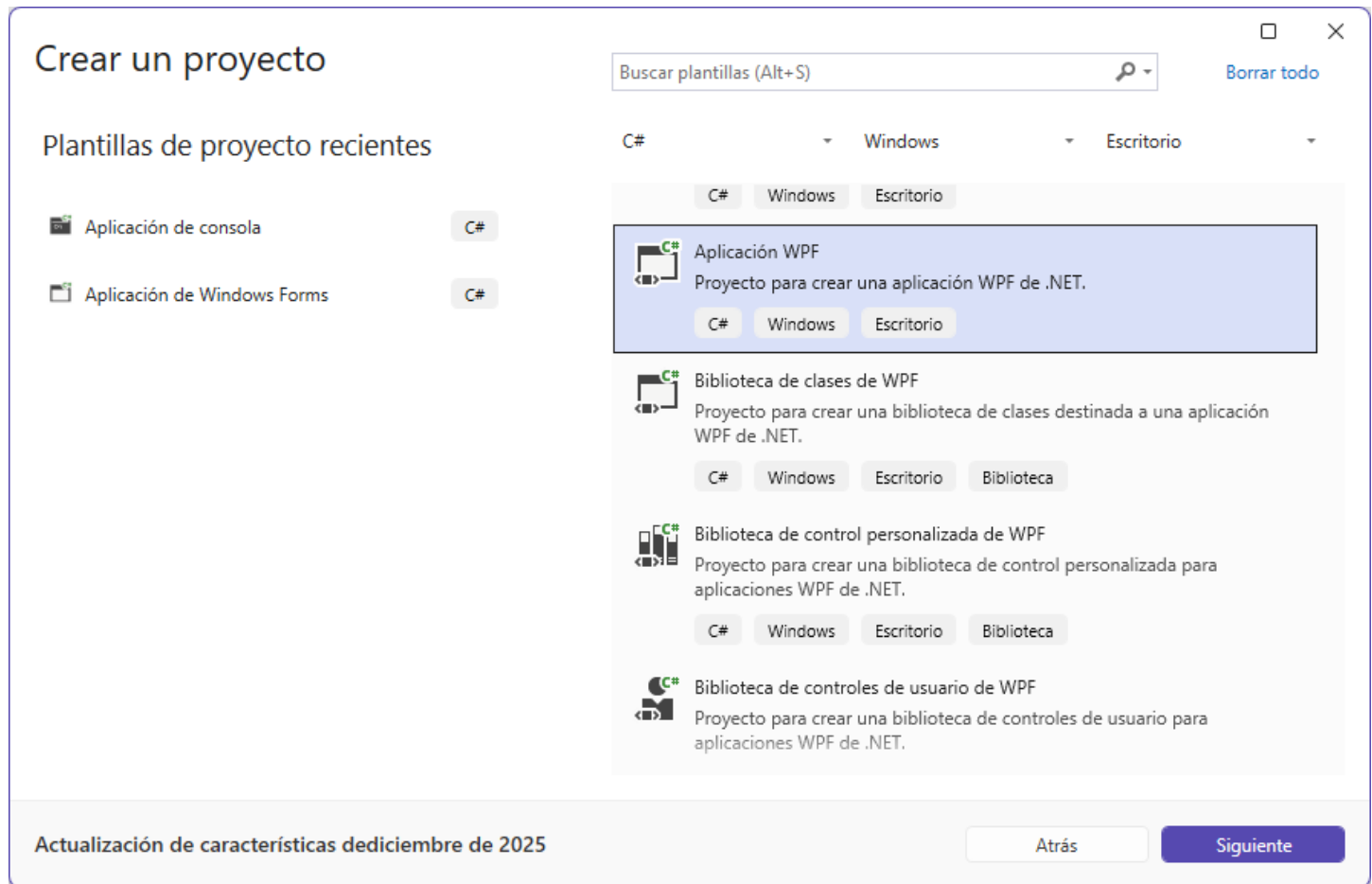


Ilustración 31: Aplicación WPF

Configure su nuevo proyecto

Aplicación WPF

C#

Windows

Escritorio

Nombre del proyecto

Sharplimagen

Ubicación

C:\Users\engin\source\repos

...

Nombre de la solución ⓘ

Sharplimagen

☐ Colocar la solución y el proyecto en el mismo directorio

Proyecto se creará en "C:\Users\engin\source\repos\Sharplimagen\Sharplimagen\"

Actualización de características dediciembre de 2025

Atrás

Siguiente

Ilustración 32: Nombre del proyecto

Información adicional

Aplicación WPF

C#WindowsEscritorio

Framework ⓘ

.NET 10.0 (Compatibilidad a largo plazo)

Actualización de características dediciembre de 2025

Atrás

Crear

Ilustración 33: Seleccionar .NET 10

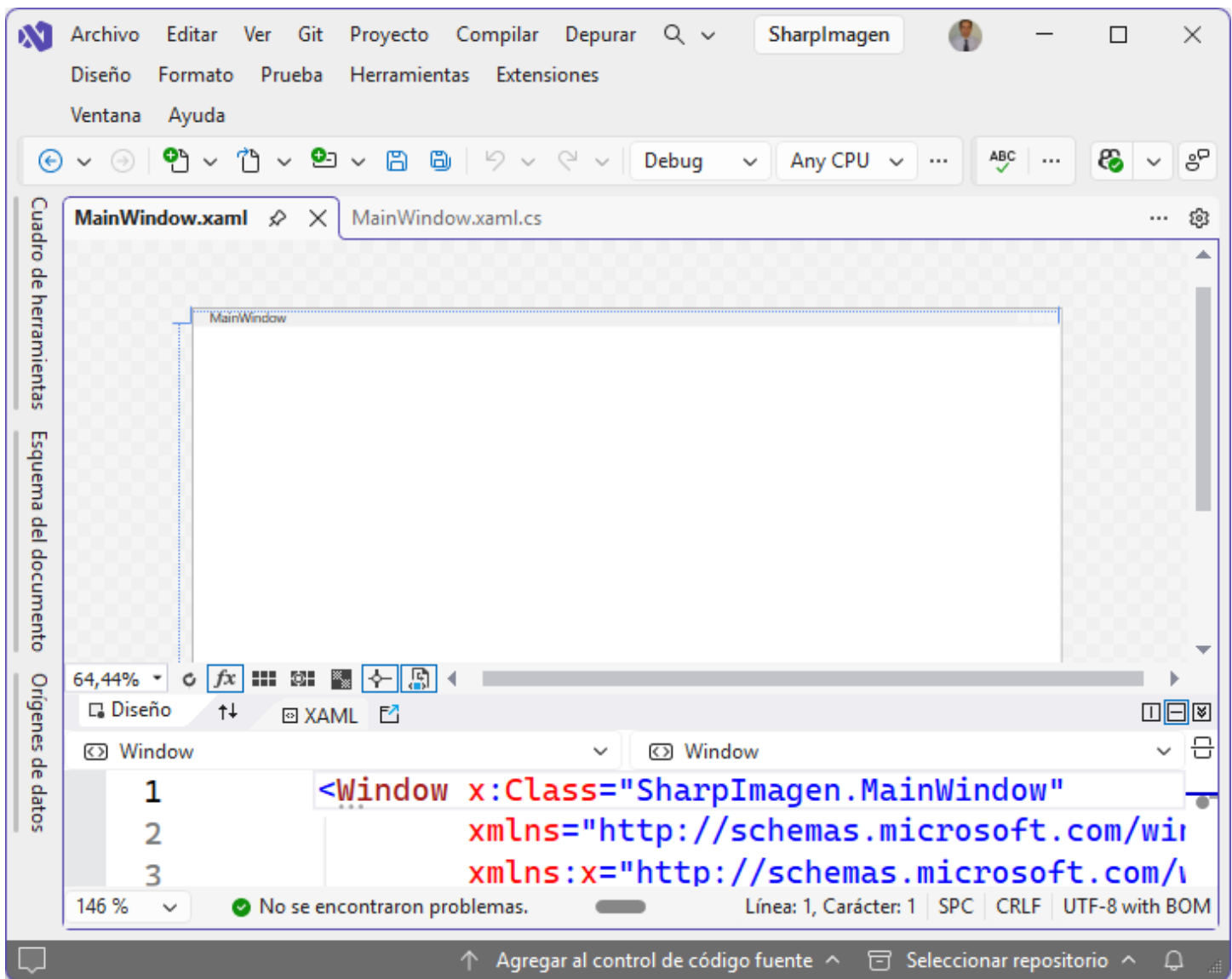


Ilustración 34: Aplicación WPF típica

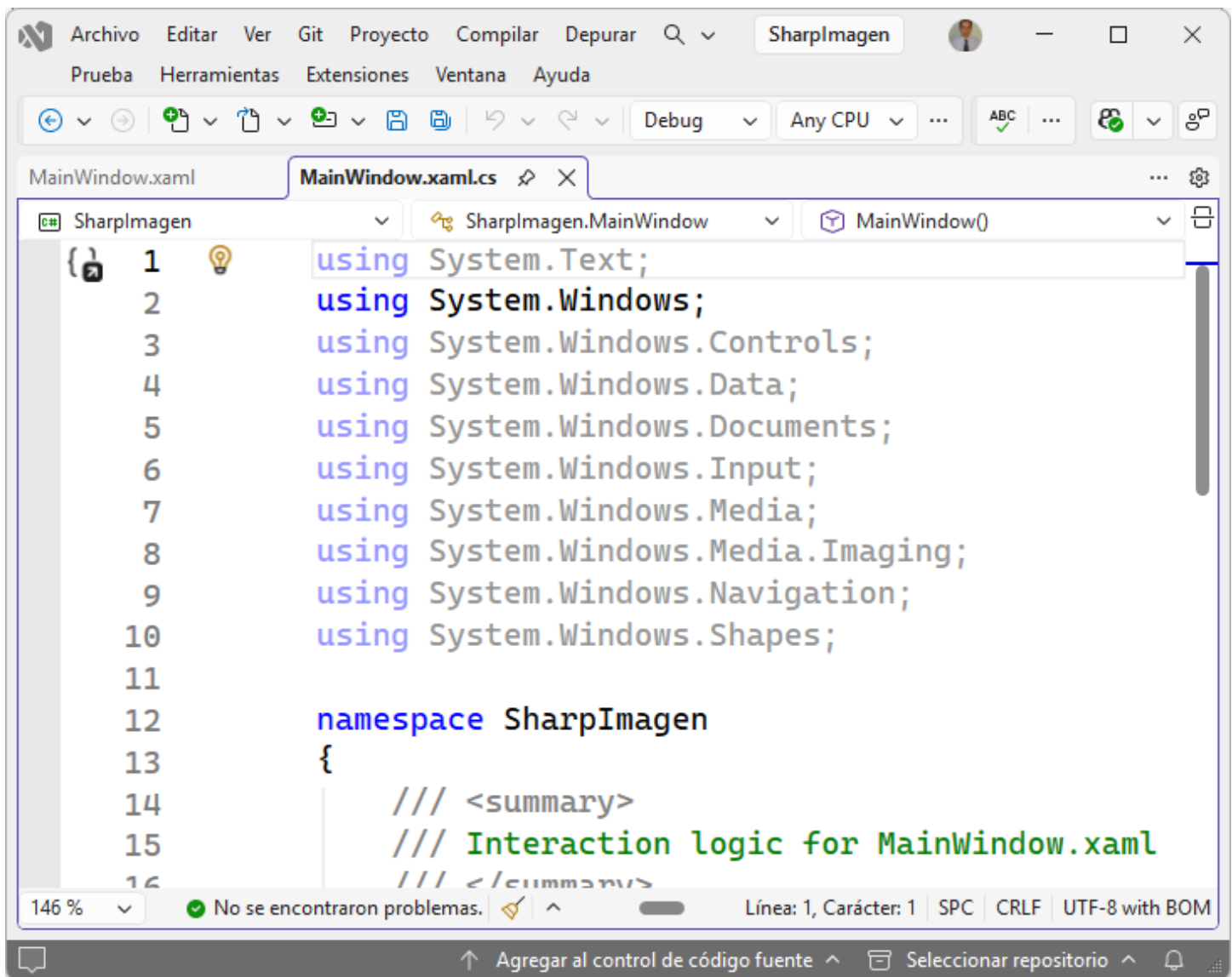


Ilustración 35: Código en C#

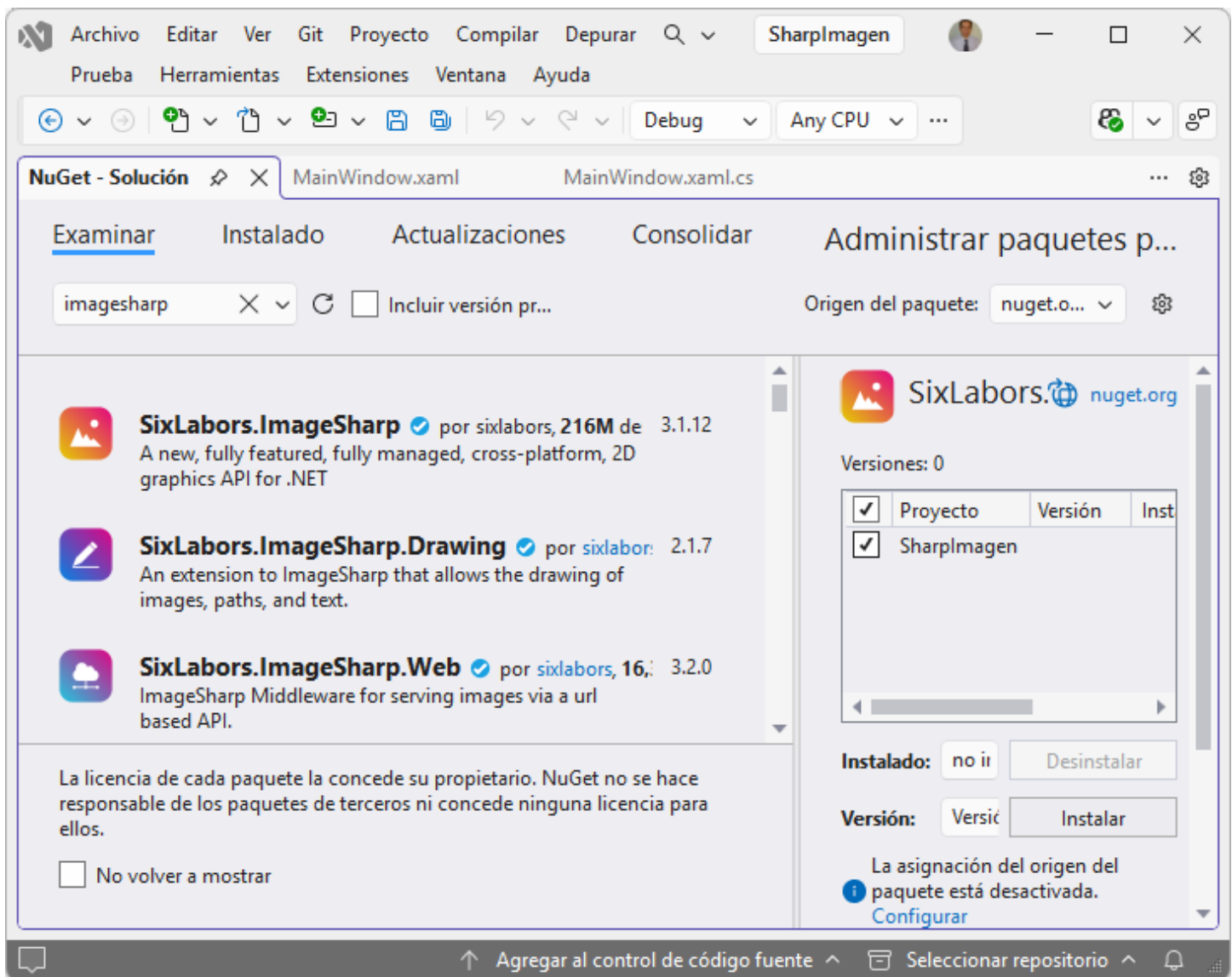


Ilustración 36: Carga las librerías

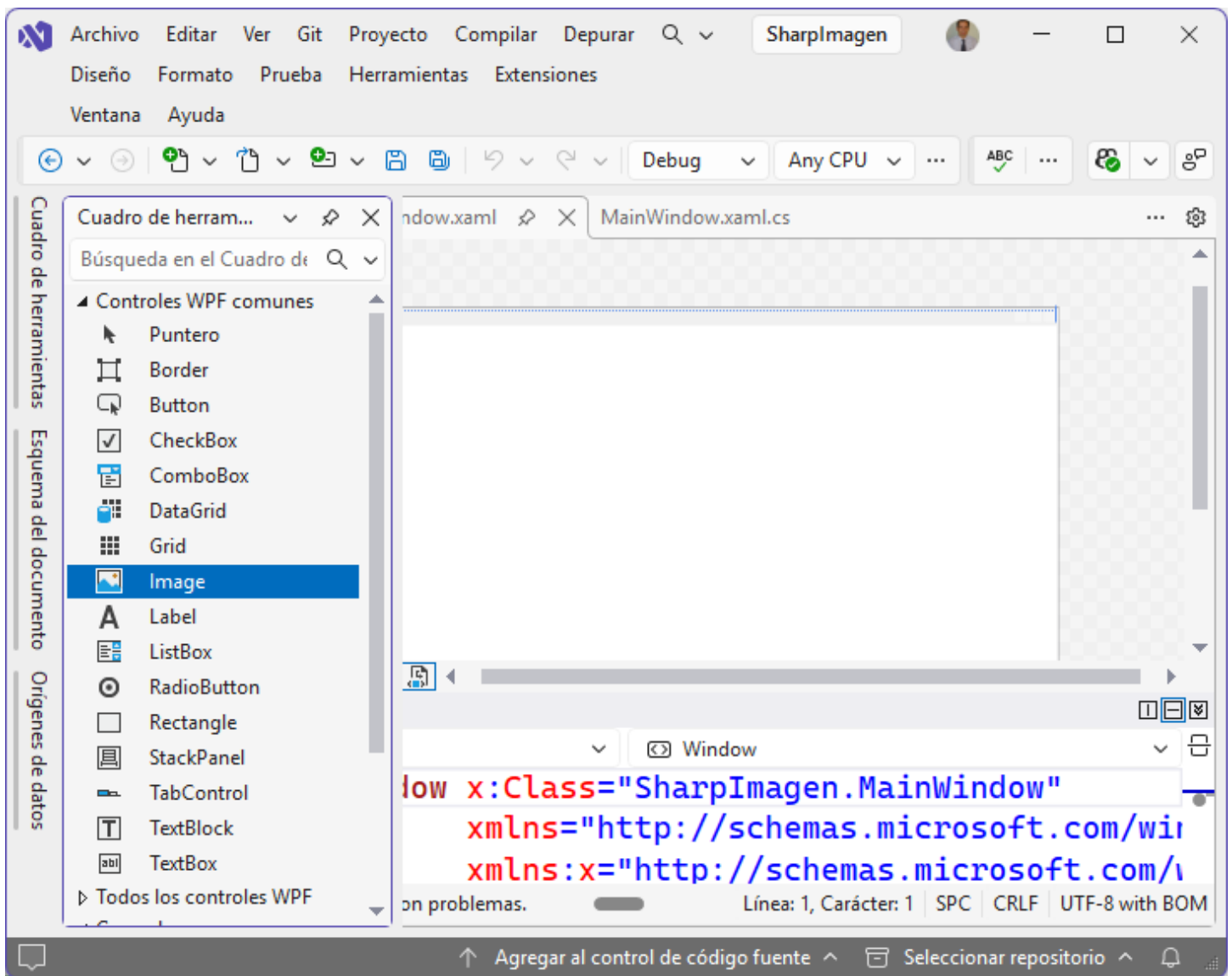


Ilustración 37: Se pone el objeto "Image" en la ventana

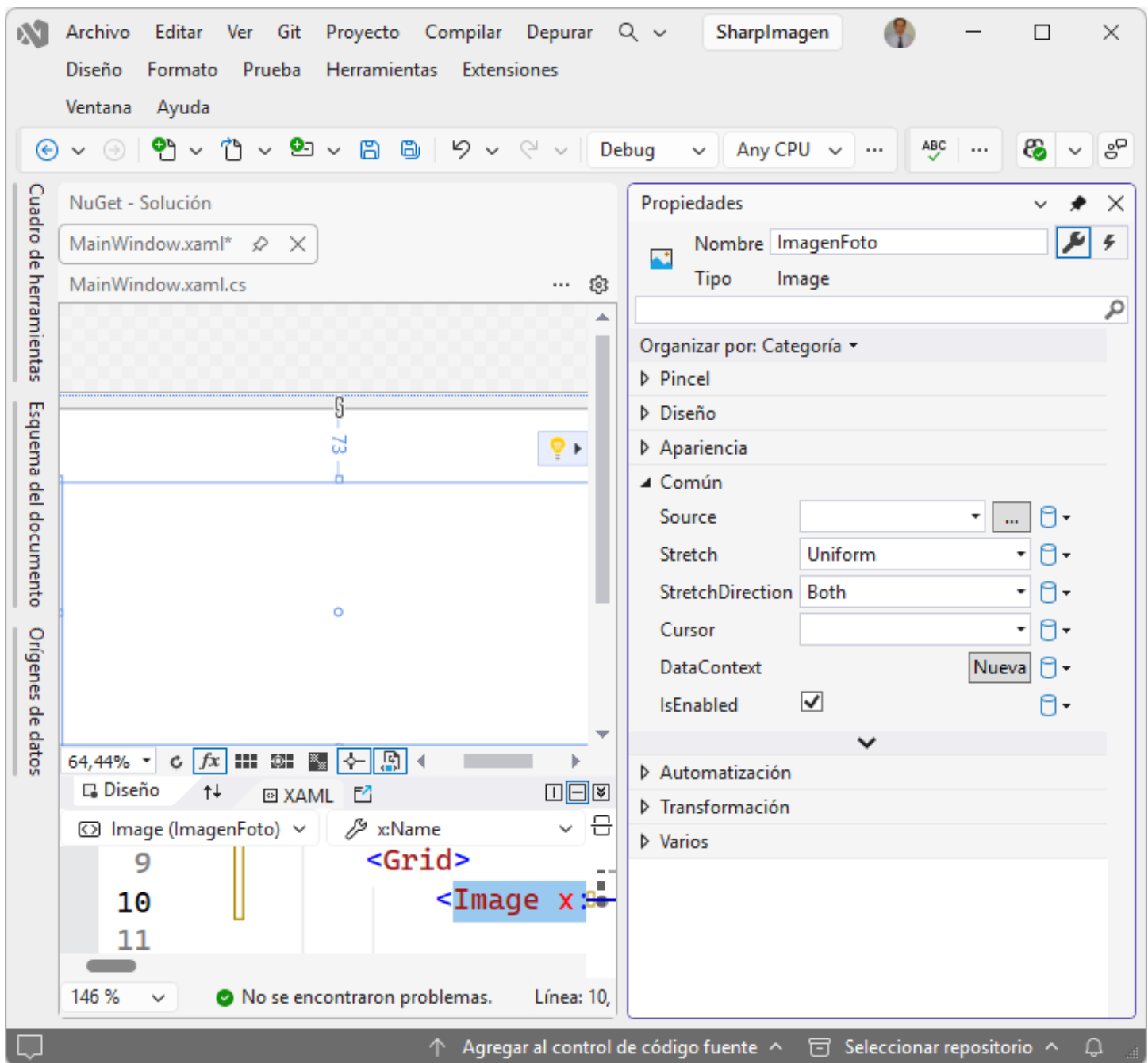


Ilustración 38: Se pone nombre a ese objeto "Image"

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.zip

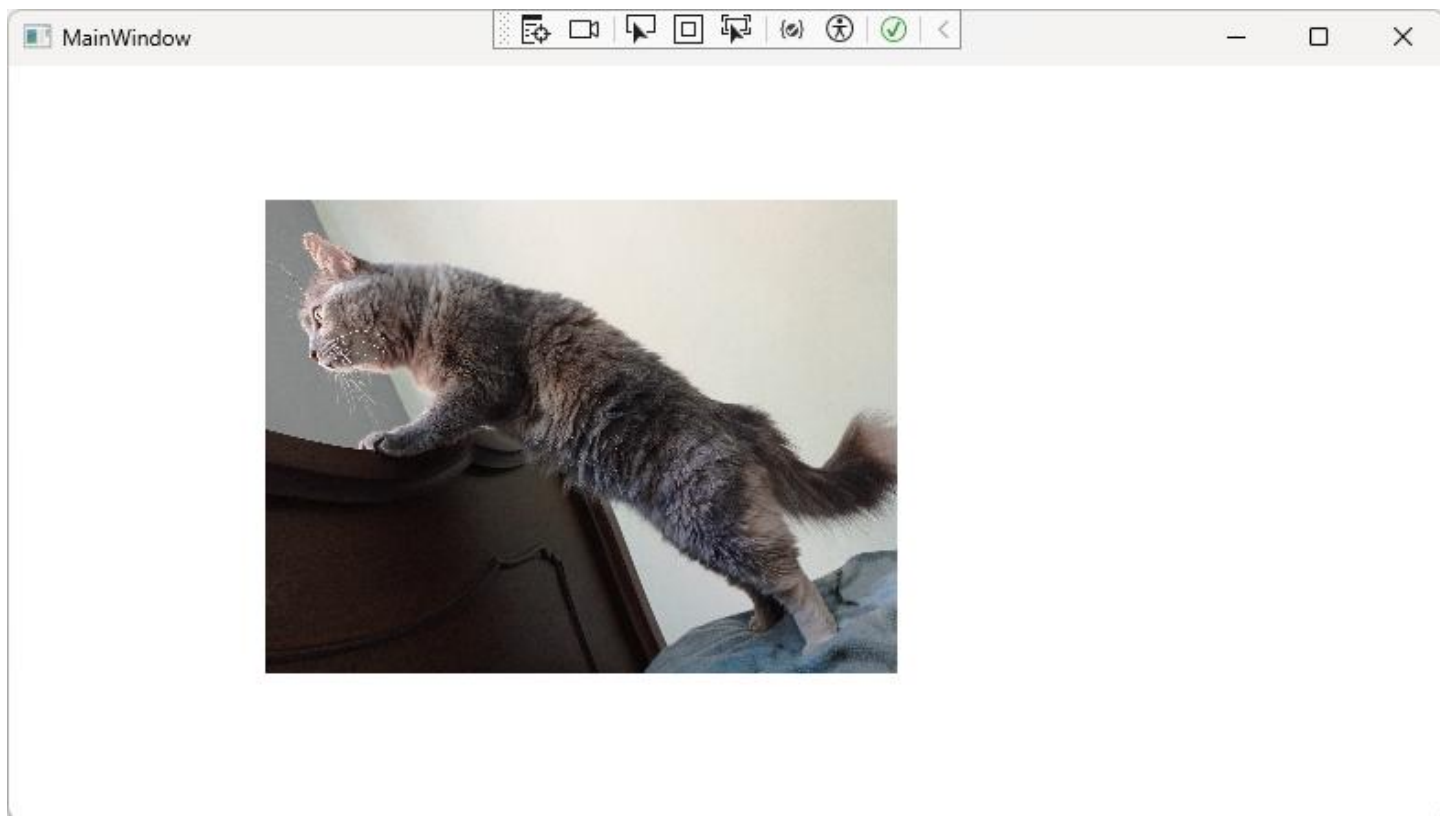


Ilustración 39: Ejecución del proyecto

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.zip

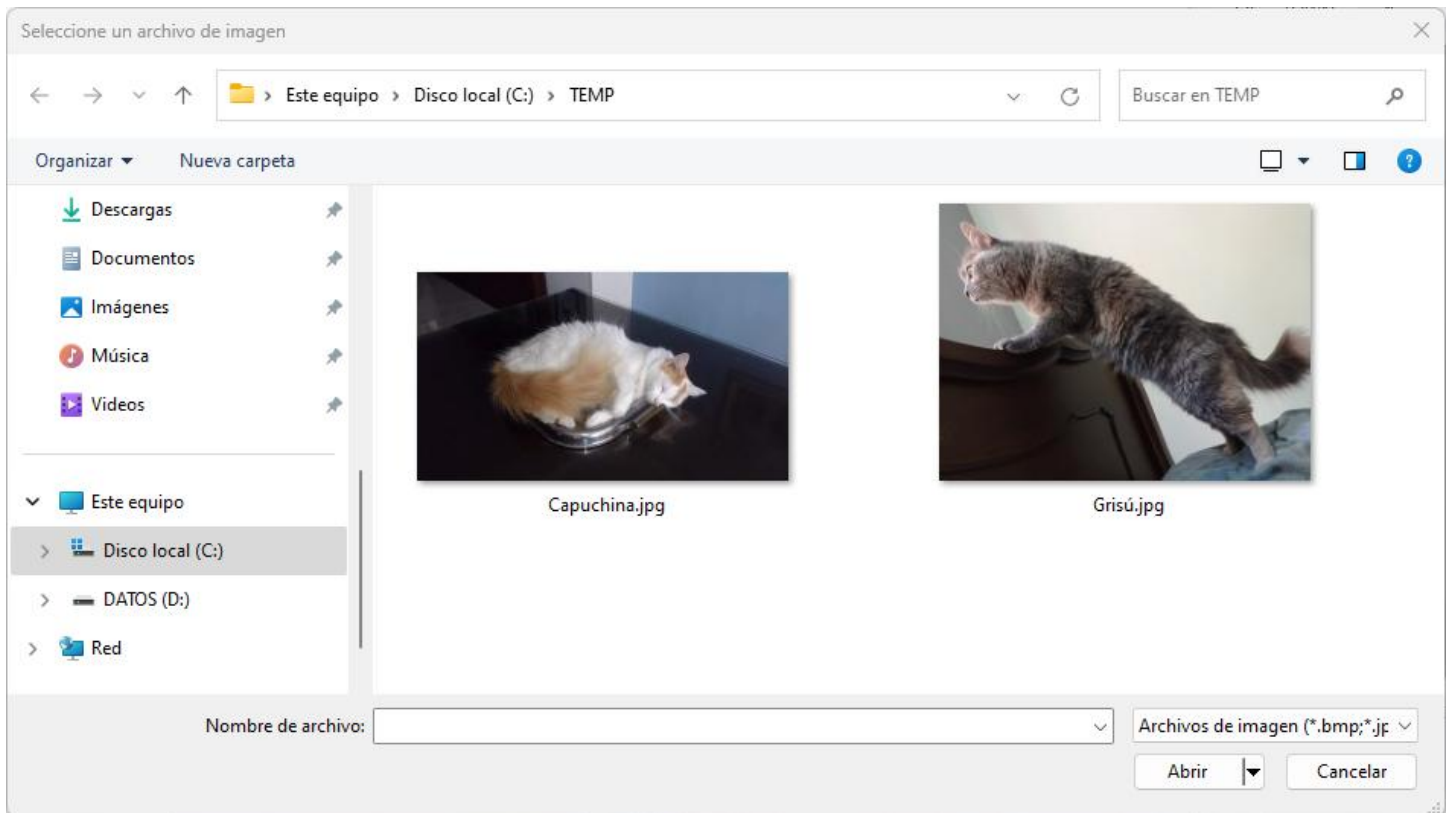


Ilustración 40: Una ventana de diálogo para escoger el archivo de imagen

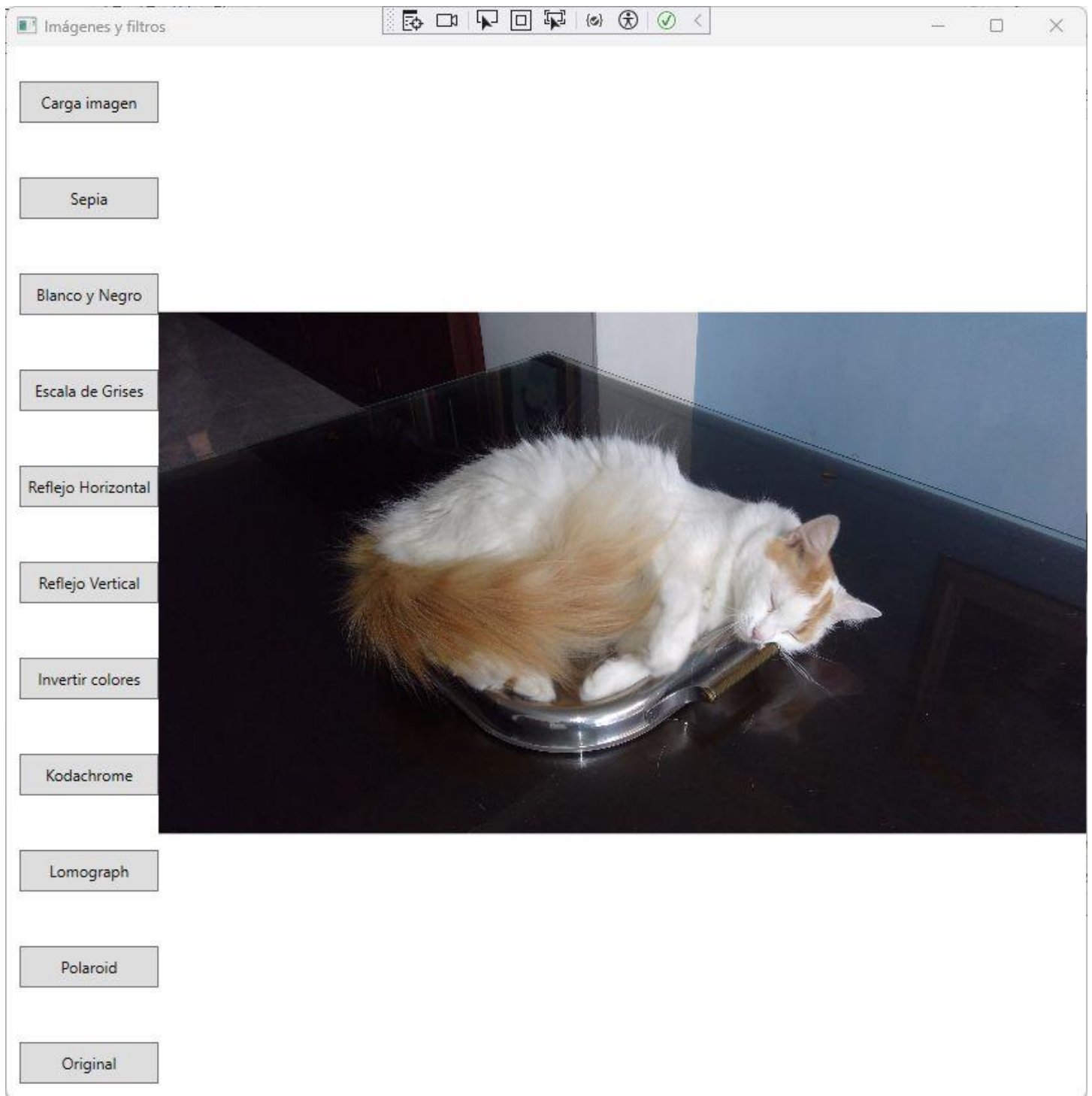


Ilustración 41: Imagen cargada

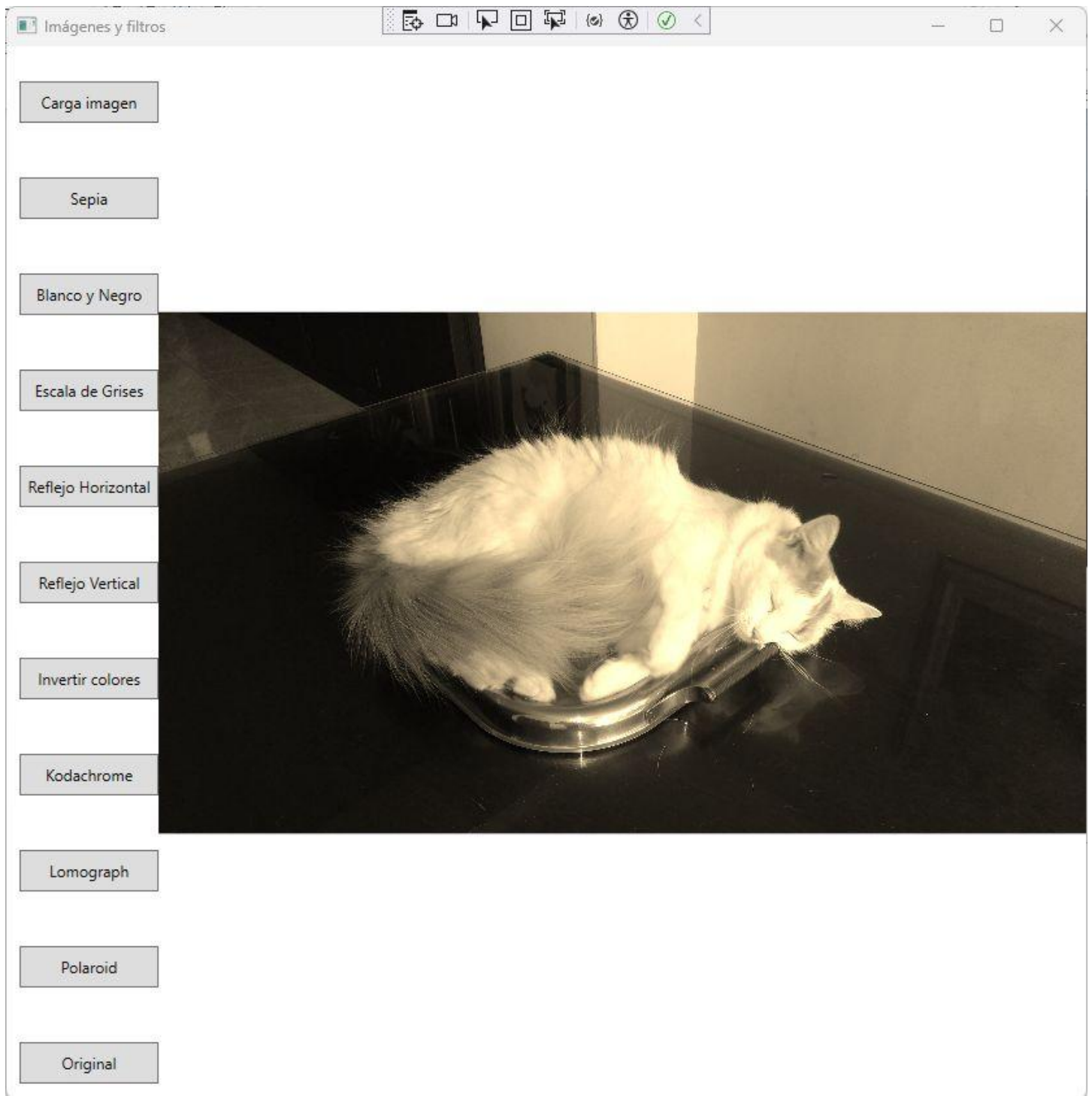


Ilustración 42: Aplicando el filtro sepia

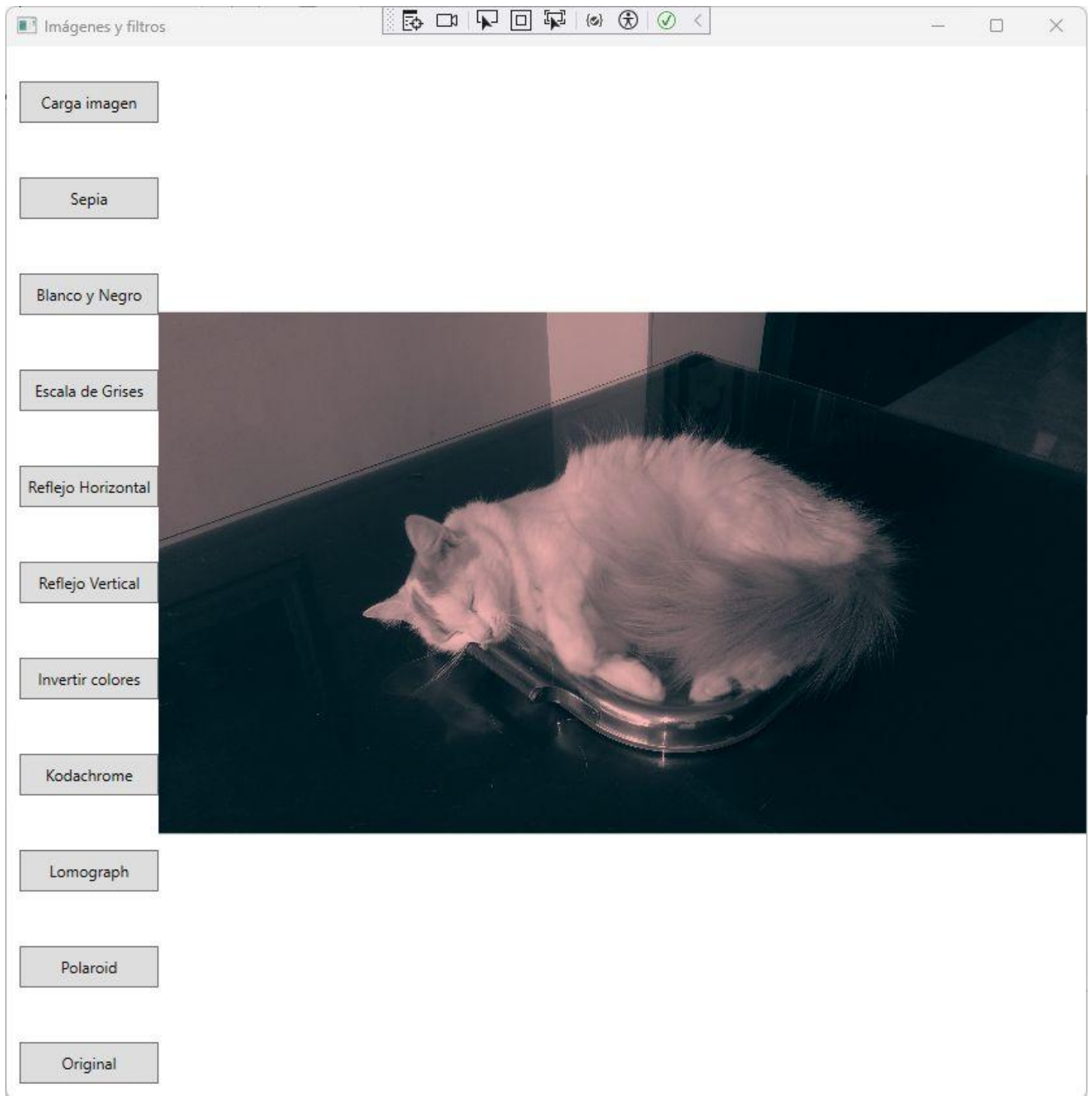


Ilustración 43: Aplicando reflejo horizontal y doble vez Kodachrome


```

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 btnGrises_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}");
    }
}
}
}
}
}

```

```

<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="btnGris_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>

```

```
using SixLabors.ImageSharp;

namespace Ejemplo;

class Program {
    static void Main() {
        ImageInfo InformacionImagen = Image.Identify(@"C:\\TEMP\\Grisú.jpg");
        Console.WriteLine($"Ancho: {InformacionImagen.Width}");
        Console.WriteLine($"Alto: {InformacionImagen.Height}");
    }
}
```

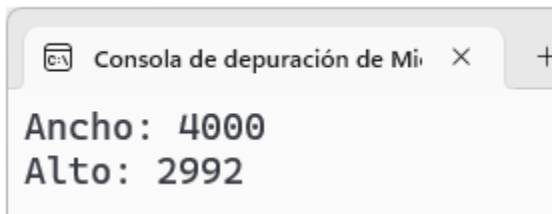


Ilustración 44: Información de la imagen

Dibujando en imágenes

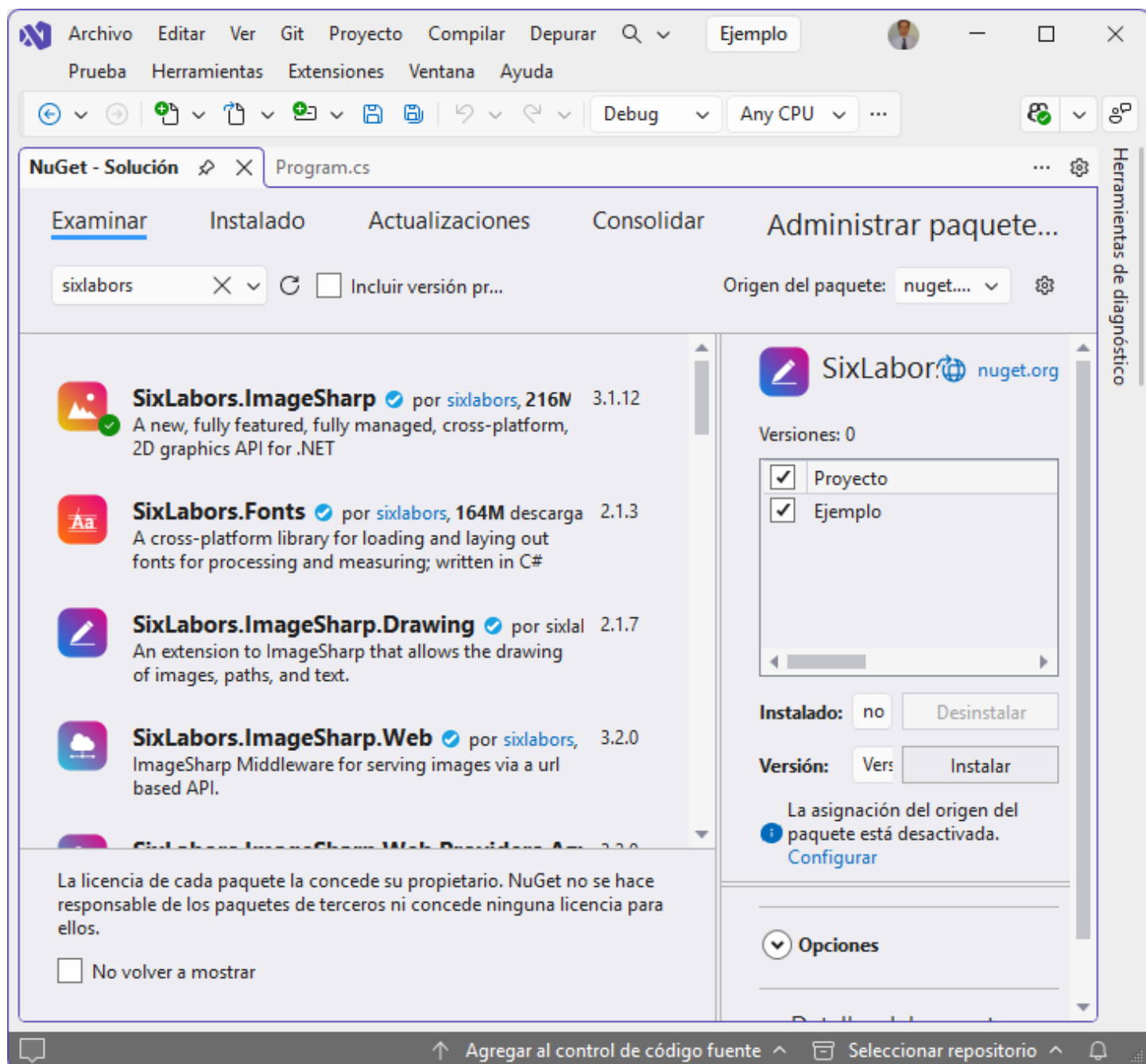


Ilustración 45: Se agrega SixLabors.ImageSharp.Drawing

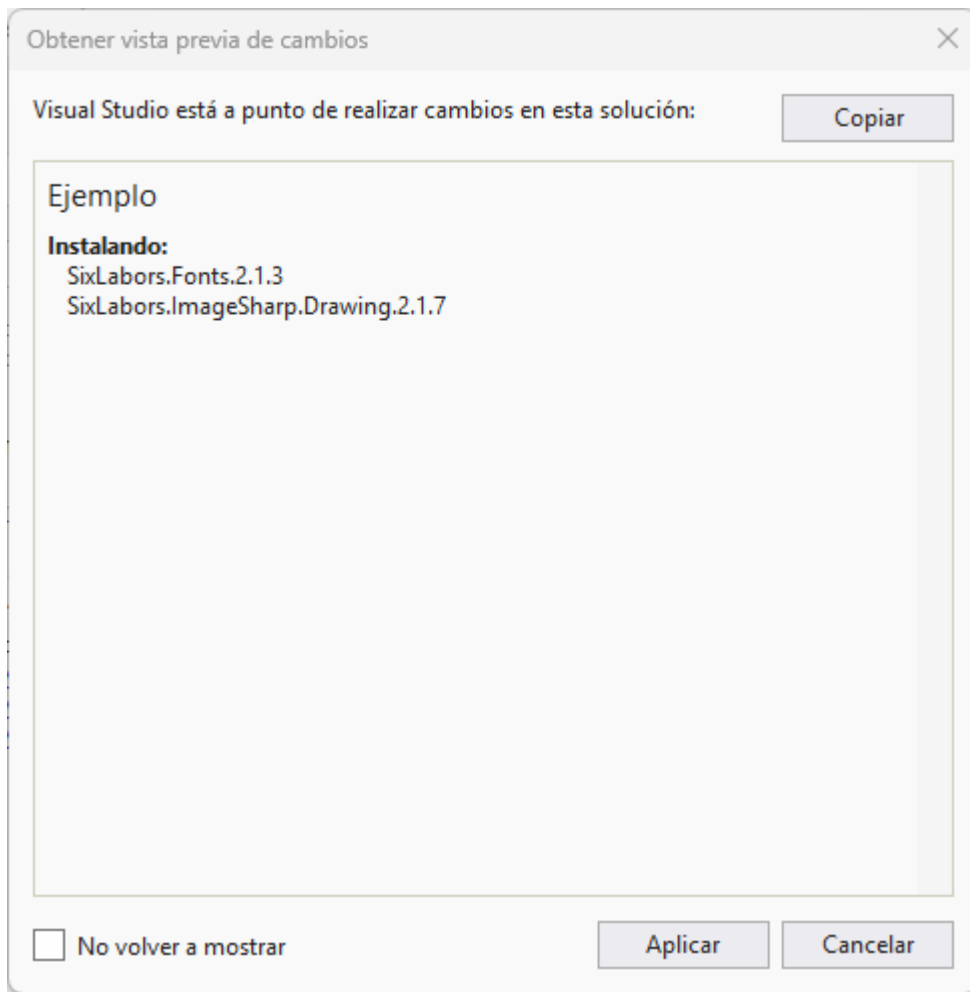


Ilustración 46: Agregar Drawing

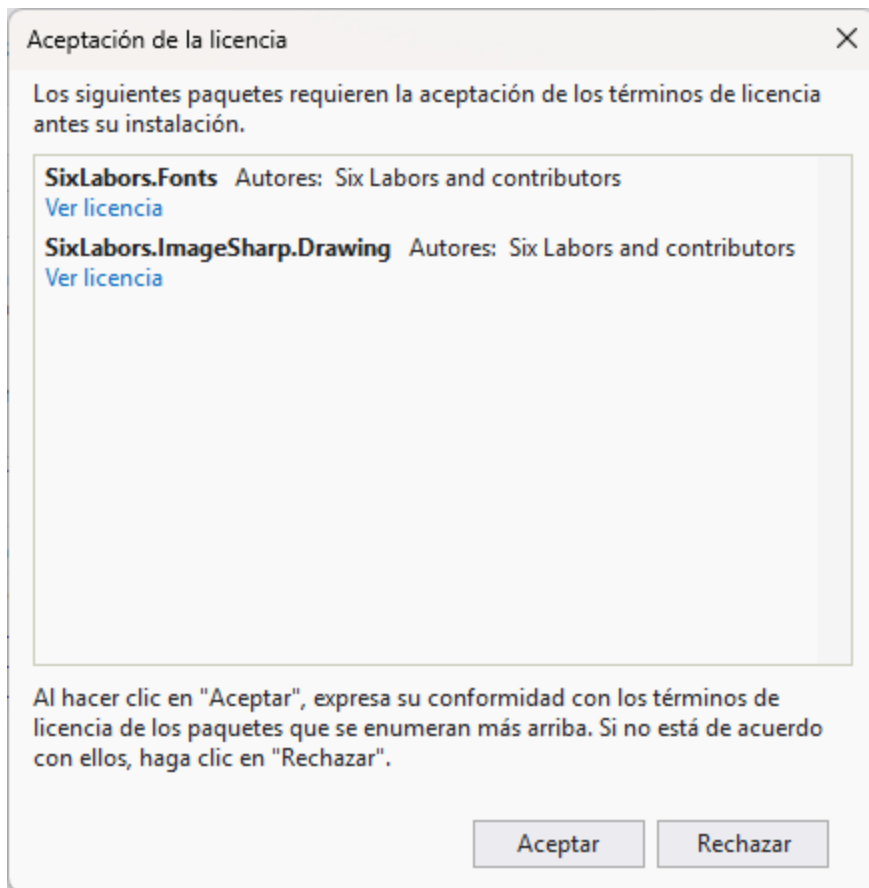


Ilustración 47: Licencia de esa librería

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

namespace Ejemplo;

class Program {
    static void Main() {
        // Crear una imagen de 500x500 píxeles con fondo blanco
        using (var NuevaImagen = new Image<Rgba32>(500, 500)) {
            NuevaImagen.Mutate(Lienzo => {
                // Fondo blanco
                Lienzo.Fill(Color.White);

                // Crear un lapiz azul con grosor de 5 píxeles
                var Lapis = Pens.Solid(Color.Blue, 5);

                // Dibujar una línea desde (50, 50) hasta (450, 450)
                var Linea = new SixLabors.ImageSharp.Drawing.PathBuilder()
                    .AddLine(new PointF(50, 50), new PointF(450, 450))
                    .Build();

                Lienzo.Draw(Lapis, Linea);
            });

            // Guardar la imagen
            NuevaImagen.Save("linea_azul.png");
        }
    }
}
```

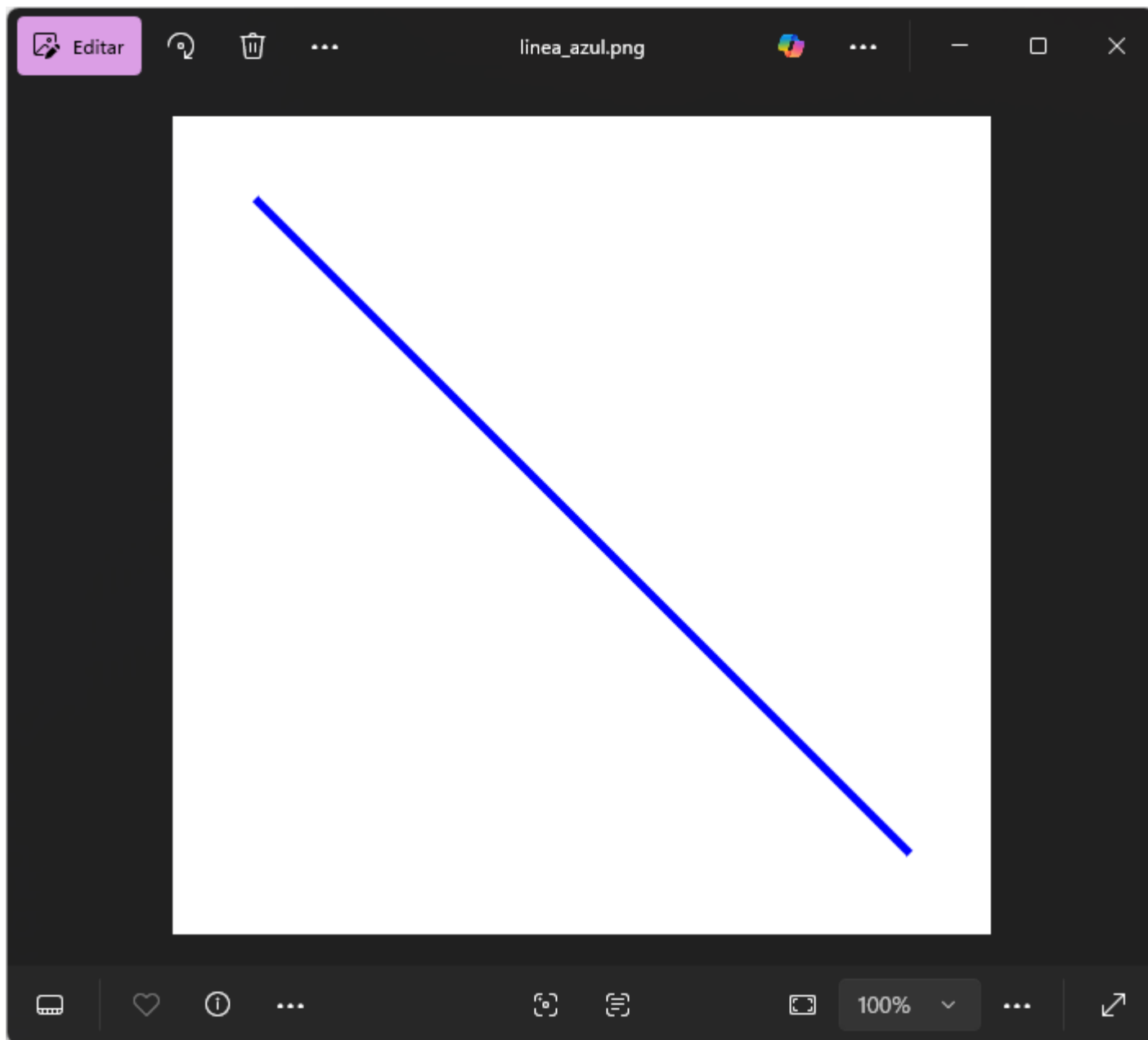


Ilustración 48: Dibuja una línea en imagen

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

namespace Ejemplo;

class Program {
    static void Main() {
        // Crear una imagen de 500x500 píxeles con fondo blanco
        using (var NuevaImagen = new Image<Rgba32>(500, 500)) {
            NuevaImagen.Mutate(Lienzo => {
                // Fondo blanco
                Lienzo.Fill(Color.White);

                // Crear un lapiz azul con grosor de 5 píxeles
                var Lapiz = Pens.Solid(Color.Blue, 5);

                // Dibujar un círculo centrado en (250, 250) con radio 100
                var Circulo = new EllipsePolygon(new PointF(250, 250), 100);

                Lienzo.Draw(Lapiz, Circulo);
            });

            // Guardar la imagen
            NuevaImagen.Save("circulo_azul.png");
        }
    }
}
```

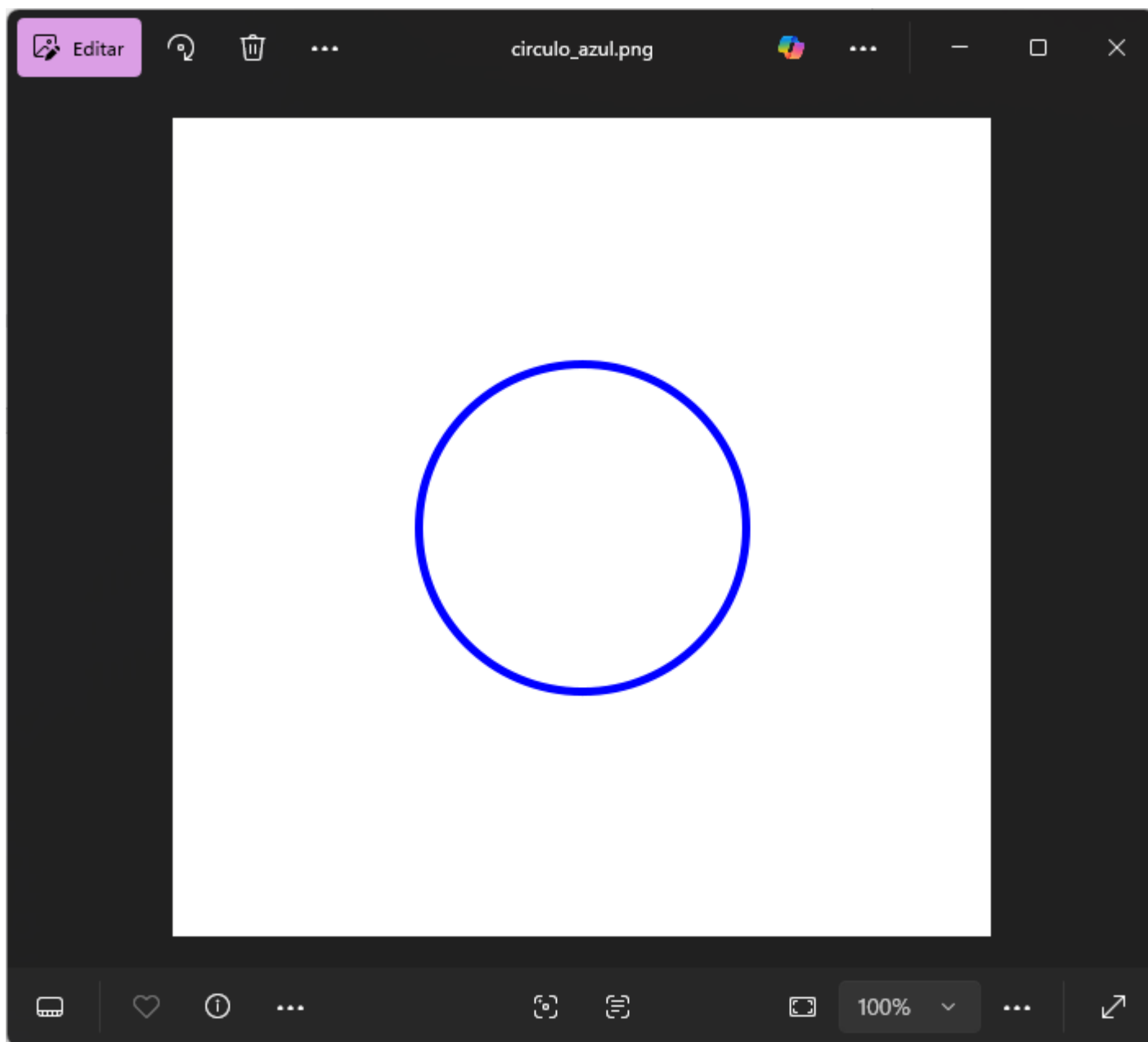


Ilustración 49: Dibuja círculo en imagen

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

namespace Ejemplo;

class Program {
    static void Main() {
        // Crear una imagen de 500x500 píxeles con fondo blanco
        using (var NuevaImagen = new Image<Rgba32>(500, 500)) {
            NuevaImagen.Mutate(Lienzo => {
                // Fondo blanco
                Lienzo.Fill(Color.White);

                // Crear un lapiz azul con grosor de 5 píxeles
                var Lapiz = Pens.Solid(Color.Blue, 5);

                // Definir los puntos del polígono (triángulo en este caso)
                var Poligono = new Polygon(new LinearLineSegment(
                    new PointF(250, 100),
                    new PointF(100, 400),
                    new PointF(400, 400),
                    new PointF(250, 100) // cerrar el triángulo
                ));

                Lienzo.Draw(Lapiz, Poligono);
            });

            // Guardar la imagen
            NuevaImagen.Save("poligono_azul.png");
        }
    }
}
```

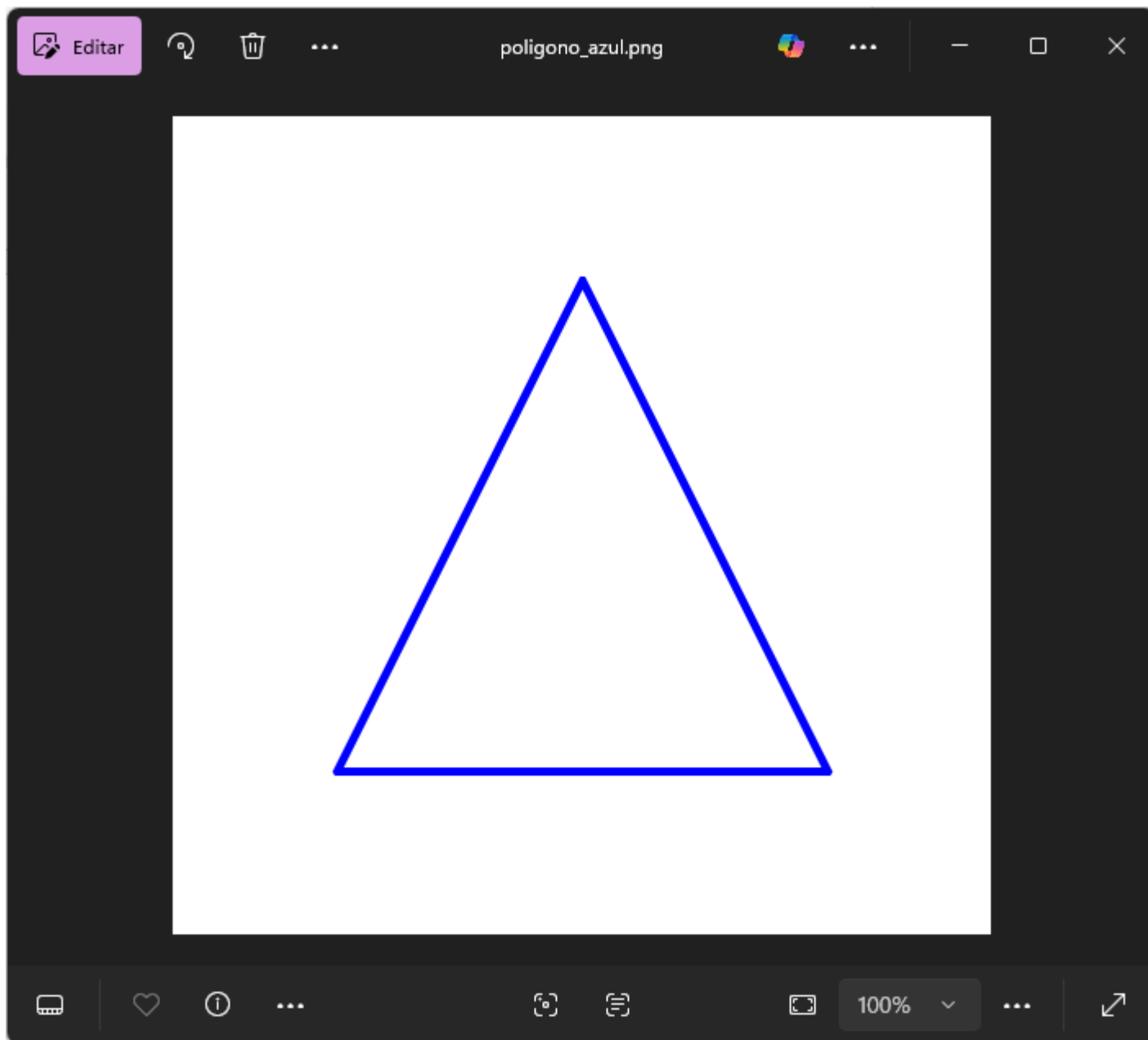


Ilustración 50: Dibuja un polígono dentro de una imagen

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

namespace Ejemplo;

class Program {
    static void Main() {
        int Ancho = 500;
        int Alto = 500;
        int cantidadLineas = 20;
        Random Azar = new();

        // Crear una imagen de 500x500 píxeles con fondo blanco
        using (var NuevaImagen = new Image<Rgba32>(500, 500)) {
            NuevaImagen.Mutate(Lienzo => {
                // Fondo blanco
                Lienzo.Fill(Color.White);

                // Crear un lapiz azul con grosor de 5 píxeles
                var Lapiz = Pens.Solid(Color.Blue, 5);

                for (int i = 0; i < cantidadLineas; i++) {
                    // Generar puntos aleatorios dentro del área de la imagen
                    var x1 = Azar.Next(Ancho);
                    var y1 = Azar.Next(Alto);
                    var x2 = Azar.Next(Ancho);
                    var y2 = Azar.Next(Alto);

                    var Linea = new SixLabors.ImageSharp.Drawing.PathBuilder()
                        .AddLine(new PointF(x1, y1), new PointF(x2, y2))
                        .Build();

                    Lienzo.Draw(Lapiz, Linea);
                }
            });

            // Guardar la imagen
            NuevaImagen.Save("Lineas.png");
        }
    }
}
```

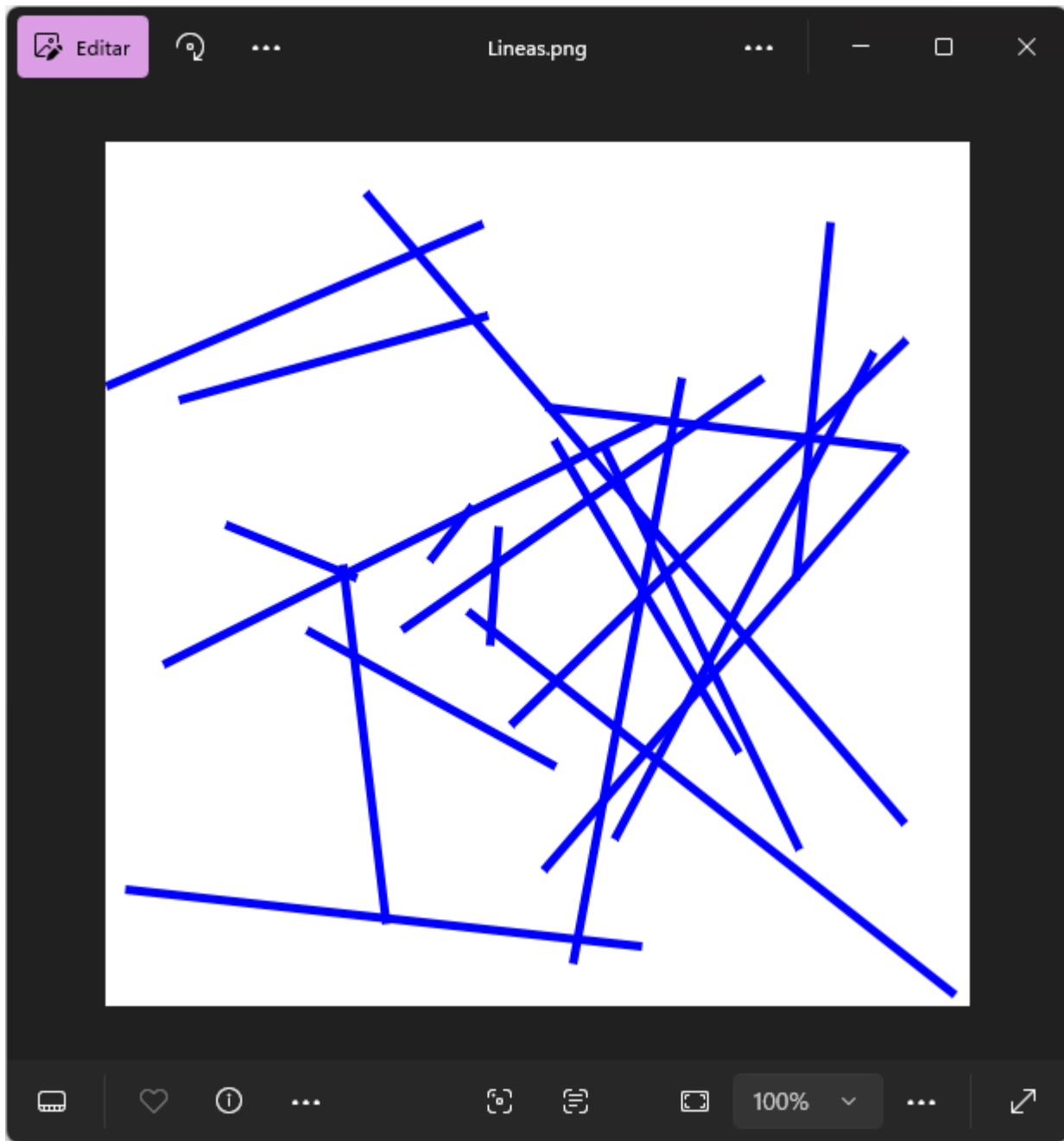



Ilustración 51: Dibuja líneas en una imagen

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

namespace Ejemplo;

class Program {
    static void Main() {
        int Ancho = 500;
        int Alto = 500;
        int cantidadCirculos = 20;
        Random Azar = new();

        // Crear una imagen de 500x500 píxeles con fondo blanco
        using (var NuevaImagen = new Image<Rgba32>(500, 500)) {
            NuevaImagen.Mutate(Lienzo => {
                // Fondo blanco
                Lienzo.Fill(Color.White);

                // Crear un lapiz azul con grosor de 5 píxeles
                var Lapiz = Pens.Solid(Color.Blue, 5);

                for (int i = 0; i < cantidadCirculos; i++) {
                    // Radio aleatorio entre 10 y 50
                    float radio = Azar.Next(10, 51);

                    // Posición aleatoria dentro de los límites de la imagen
                    float x = Azar.Next((int)radio, Ancho - (int)radio);
                    float y = Azar.Next((int)radio, Alto - (int)radio);

                    var circle = new EllipsePolygon(new PointF(x, y), radio);
                    Lienzo.Draw(Lapiz, circle);
                }
            });

            // Guardar la imagen
            NuevaImagen.Save("Circulos.png");
        }
    }
}
```

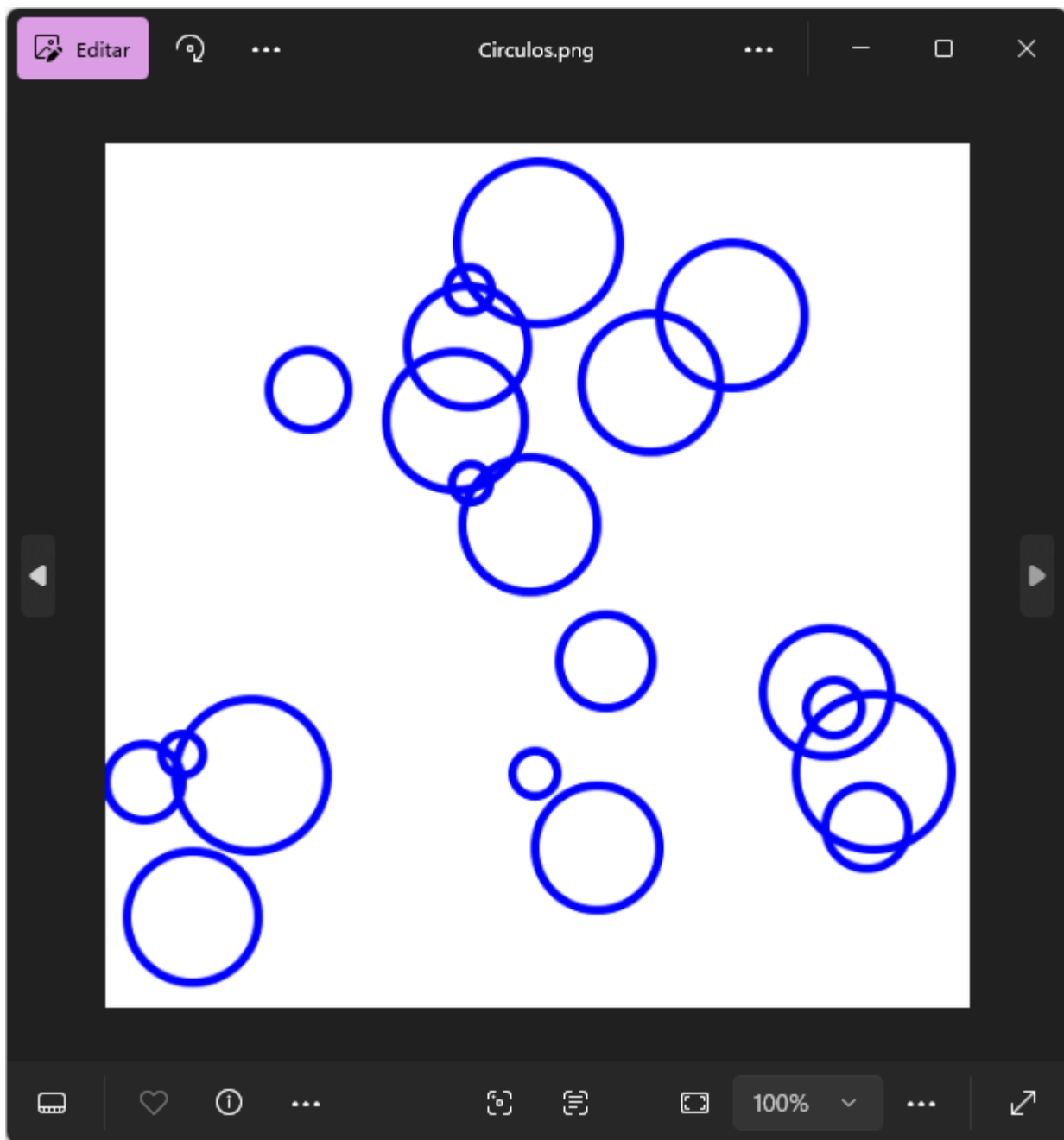


Ilustración 52: Dibuja círculos en una imagen

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

namespace Ejemplo;

class Program {
    static void Main() {
        int Ancho = 500;
        int Alto = 500;
        int cantidadRectangulos = 20;
        Random Azar = new();

        // Crear una imagen de 500x500 píxeles con fondo blanco
        using (var NuevaImagen = new Image<Rgba32>(500, 500)) {
            NuevaImagen.Mutate(Lienzo => {
                // Fondo blanco
                Lienzo.Fill(Color.White);

                // Crear un lapiz azul con grosor de 5 píxeles
                var Lapiz = Pens.Solid(Color.Blue, 5);

                for (int i = 0; i < cantidadRectangulos; i++) {
                    // Generar posición y tamaño aleatorios
                    int w = Azar.Next(40, 200);
                    int h = Azar.Next(40, 200);
                    int x = Azar.Next(0, Ancho - w);
                    int y = Azar.Next(0, Alto - h);

                    var rectangulo = new RectangularPolygon(x, y, w, h);

                    // Dibujar relleno azul
                    Lienzo.Fill(Color.Blue, rectangulo);

                    // Dibujar borde opcional
                    Lienzo.Draw(Color.DarkBlue, 3f, rectangulo);
                }
            });

            // Guardar la imagen
            NuevaImagen.Save("Rectangulos.png");
        }
    }
}
```

}

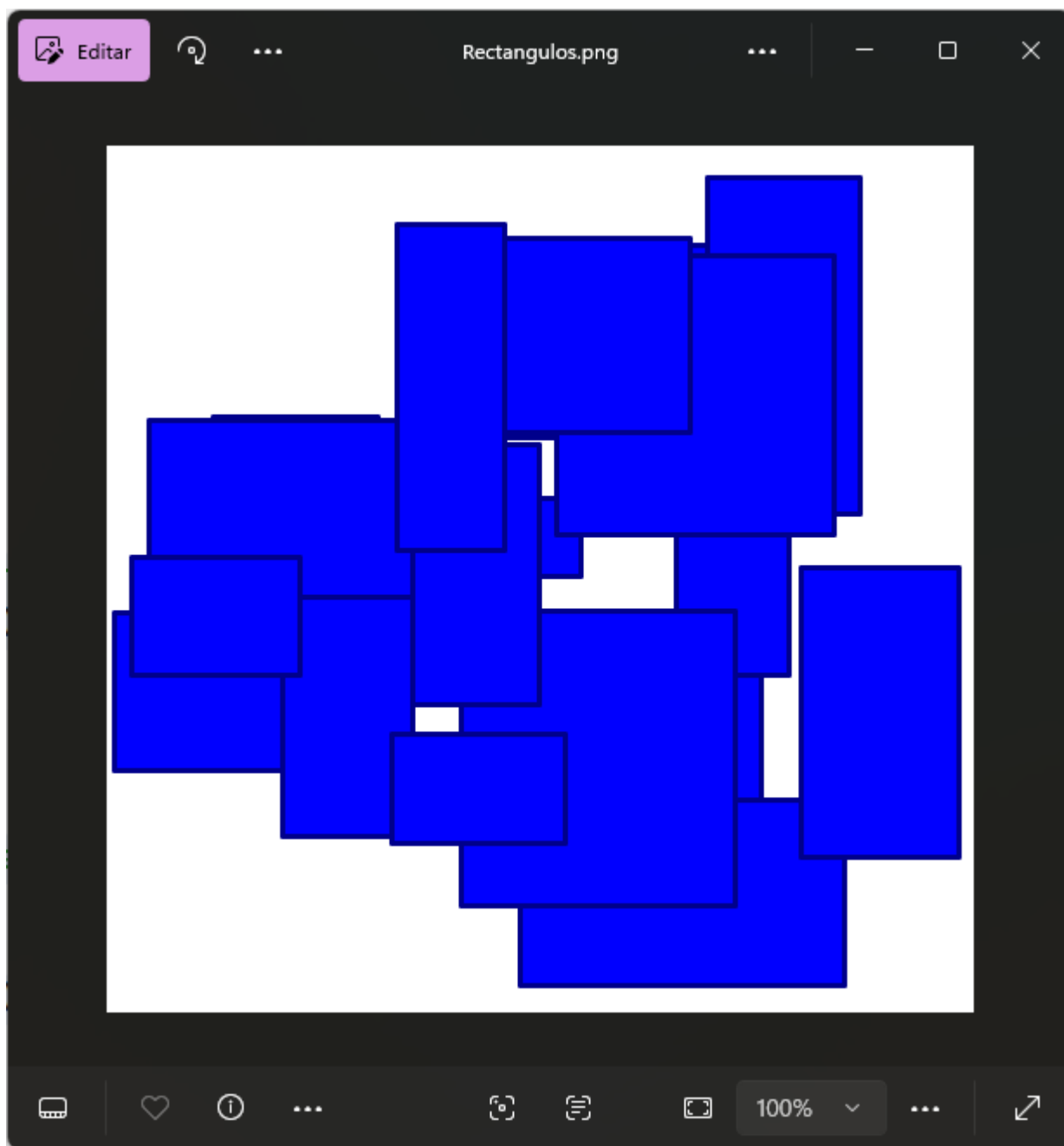


Ilustración 53: Rectángulos rellenos al azar

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

namespace Ejemplo;

class Program {
    static void Main() {
        int width = 800, height = 600;

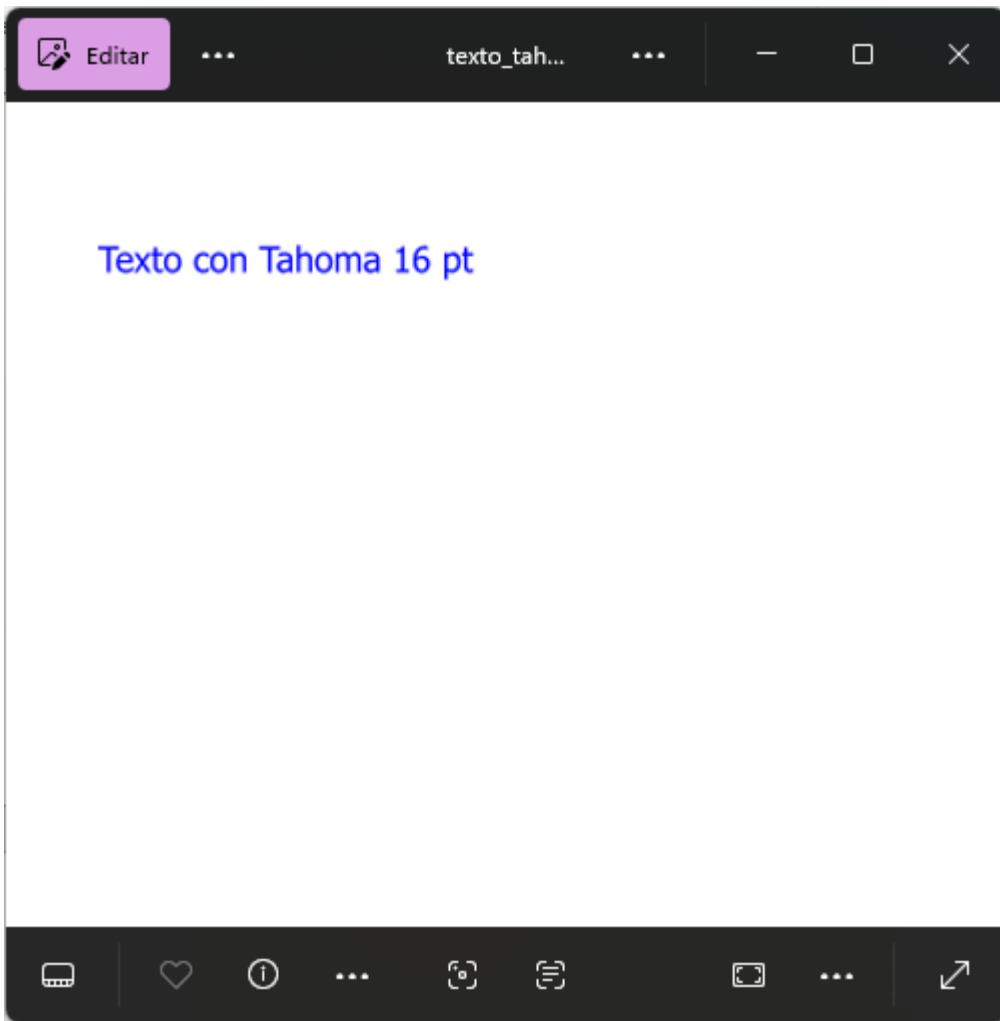
        using var img = new Image<Rgba32>(width, height, Color.White);

        // Intentar obtener Tahoma desde fuentes del sistema
        Font font;
        if (SystemFonts.TryGet("Tahoma", out FontFamily tahomaFamily)) {
            font = tahomaFamily.CreateFont(16, FontStyle.Regular);
        }
        else {
            throw new InvalidOperationException("La fuente 'Tahoma' no está disponible en el sistema.");
        }

        string texto = "Texto con Tahoma 16 pt";
        var color = Color.Blue; // Cambia el color según necesites
        var position = new PointF(50, 80); // Posición superior izquierda del texto

        img.Mutate(ctx => {
            // Dibujar texto
            ctx.DrawText(texto, font, color, position);
        });

        img.Save("texto_tahoma.png");
        Console.WriteLine("Imagen generada: texto_tahoma.png");
    }
}
```



```

using SixLabors.Fonts;
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.Drawing.Processing;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;

namespace Ejemplo;

class Program {
    static void Main() {
        int width = 800, height = 600;

        using var img = new Image<Rgba32>(width, height, Color.White);

        // Intentar obtener Tahoma desde fuentes del sistema
        Font fuenteLetra;
        if (SystemFonts.TryGet("Tahoma", out FontFamily tahomaFamily)) {
            fuenteLetra = tahomaFamily.CreateFont(31, FontStyle.Regular);
        }
        else {
            throw new InvalidOperationException("La fuente 'Tahoma' no está
disponible en el sistema.");
        }

        string Texto = "Texto con Tahoma 31 pt";

        img.Mutate(x => x
            .DrawText(
                new DrawingOptions {
                    Transform = Matrix3x2Extensions.CreateRotationDegrees(45, new
PointF(50, 50))
                },
                Texto,
                fuenteLetra,
                Color.Black,
                new PointF(50, 50)));

        img.Save("texto_tahoma.png");
        Console.WriteLine("Imagen generada: texto_tahoma.png");
    }
}

```