2024

C#: Aplicaciones Windows

Contenido

Otros libros del autor

Página web del autor y canal en Youtube

Investigación sobre Vida Artificial: http://darwin.50webs.com

Canal en Youtube: http://www.youtube.com/user/RafaelMorenoP (dedicado principalmente al desarrollo en C#)

Sitio en GitHub

El código fuente se puede descargar en https://github.com/ramsoftware/CSharpDinamica

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 2019 ® Enlace: https://visualstudio.microsoft.com/es/vs/

Dedicatoria

A mis padres, a mi hermana….

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

Introducción

Este libro se concentra en el desarrollo de aplicaciones nativas Windows Forms.

Iniciando

Esta es la pantalla de inicio de Microsoft Visual Studio 2022 Community Edition

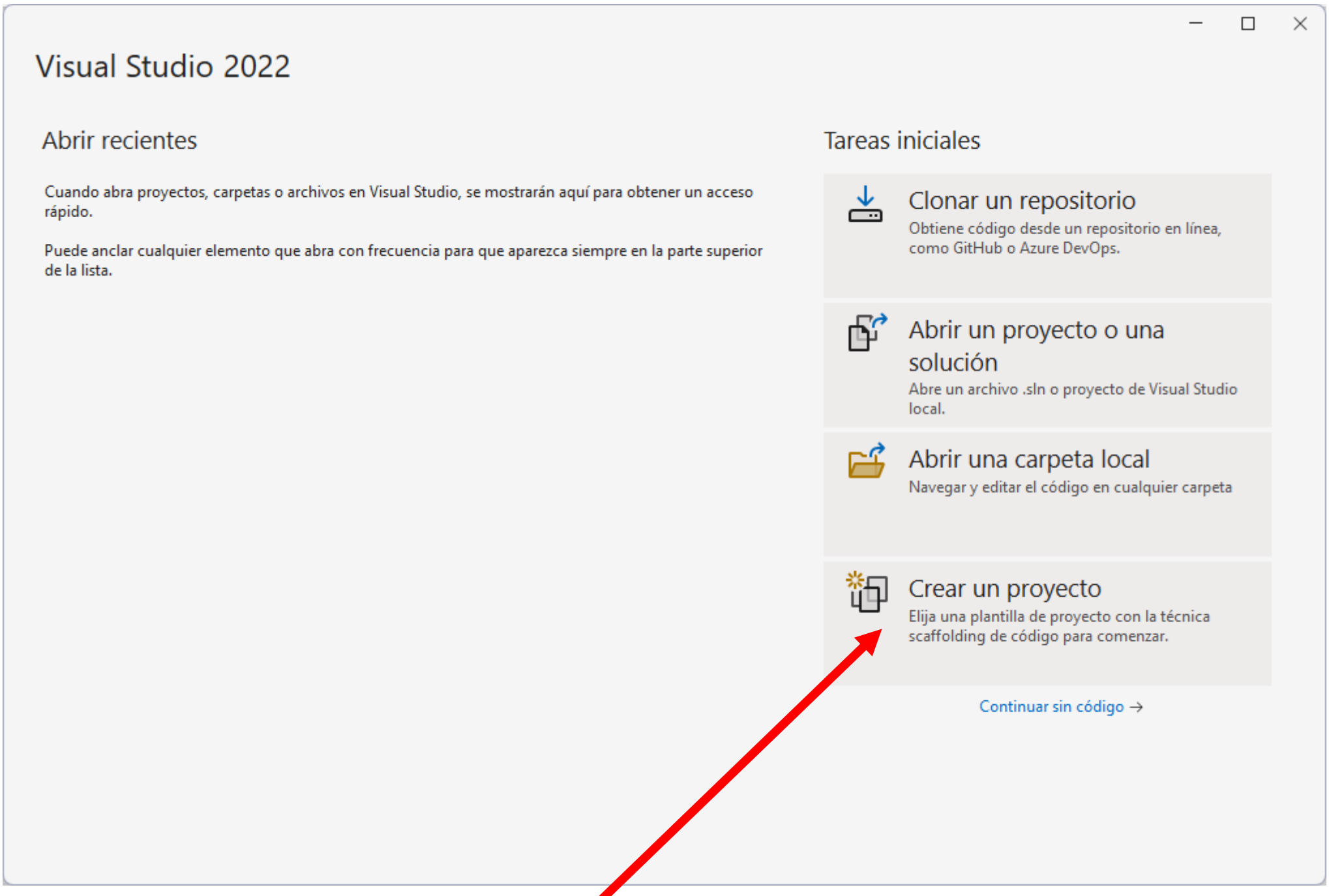


Ilustración 1: Pantalla inicial de Microsoft Visual Studio 2022 Community Edition

El siguiente paso es dar clic en “Crear un proyecto”

Pantalla que sigue:

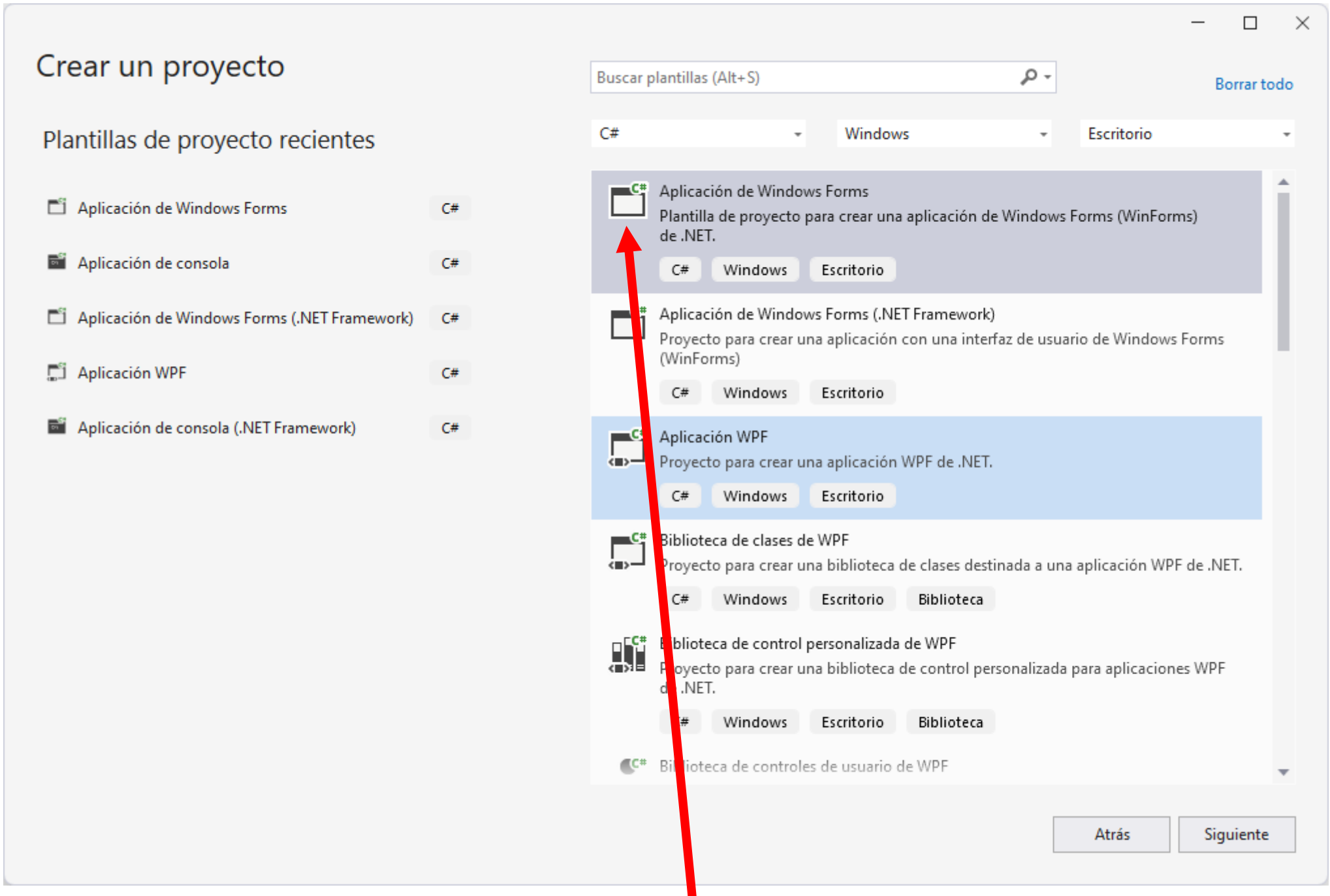
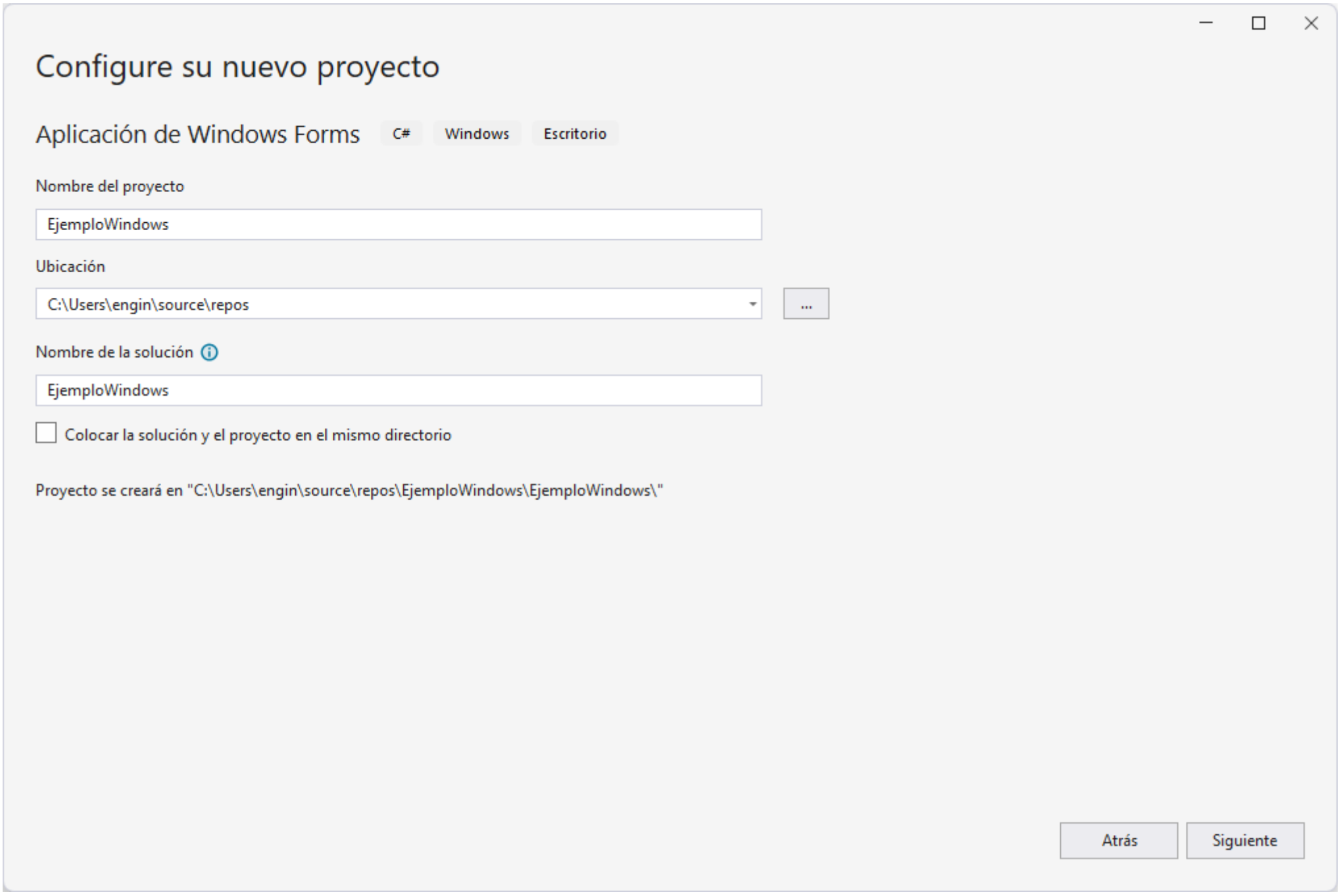
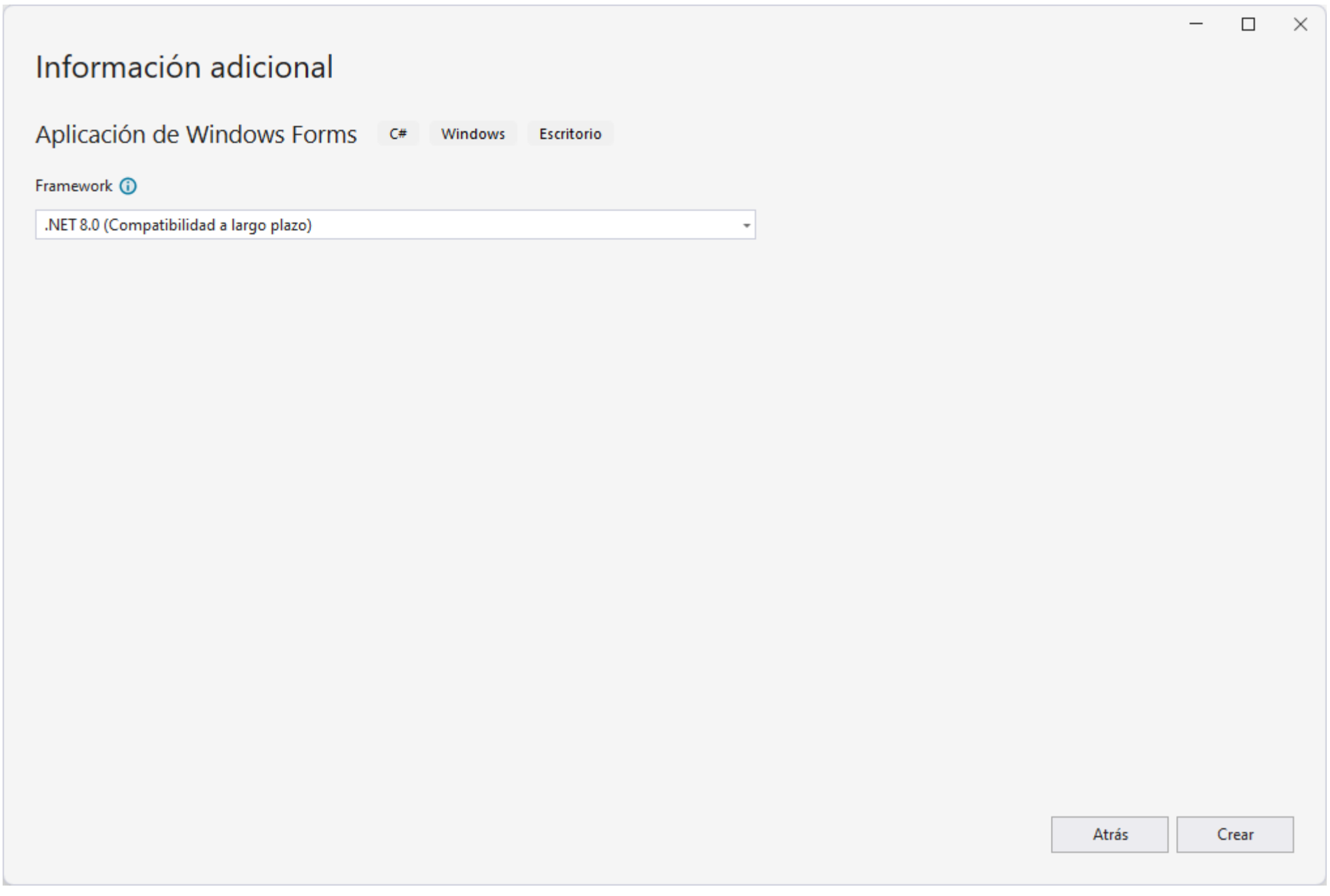


Ilustración 2: Tipos de proyecto que se pueden elaborar en este IDE. A usar los filtros:

Una vez seleccionado “Aplicación de Windows Forms se da clic en Siguiente





Y se da clic en “Crear”

La pantalla de trabajo que se presenta:

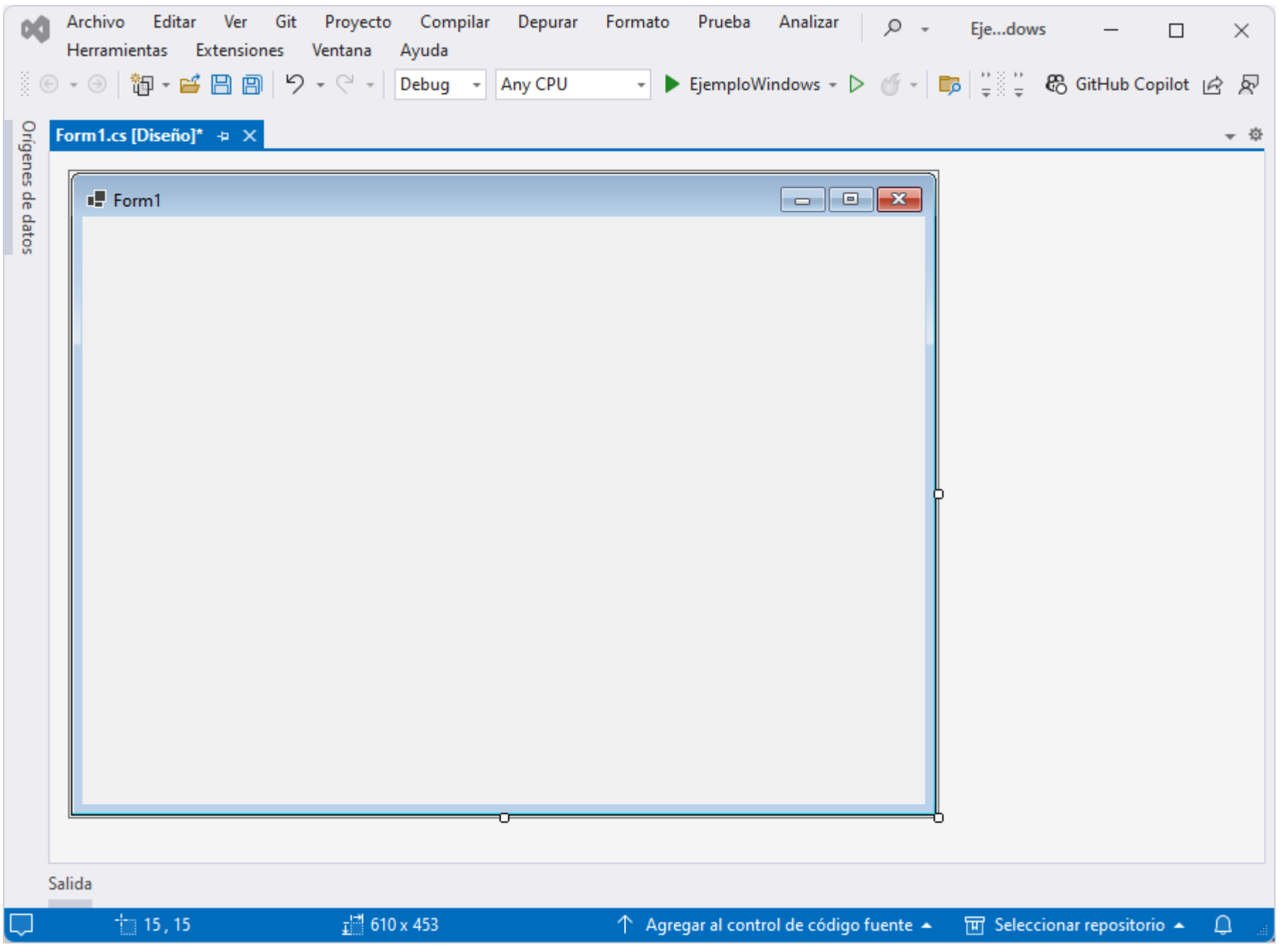


Ilustración 3: Pantalla que nos recibe para hacer una aplicación Windows

Personalizando el entorno

Guardar y cerrar todas las ventanas para dejar el escritorio limpio.

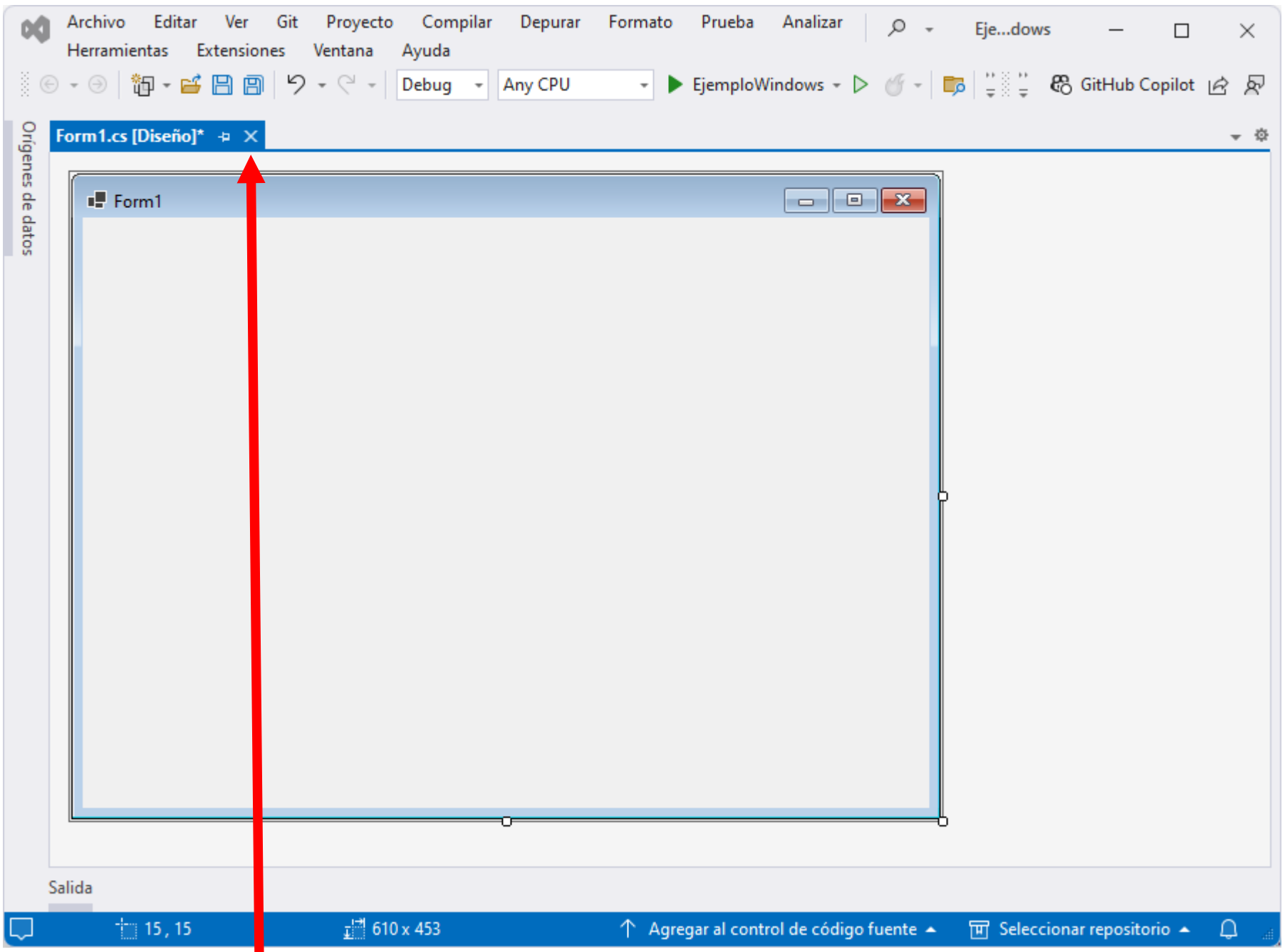


Ilustración 4: Cerrando todo

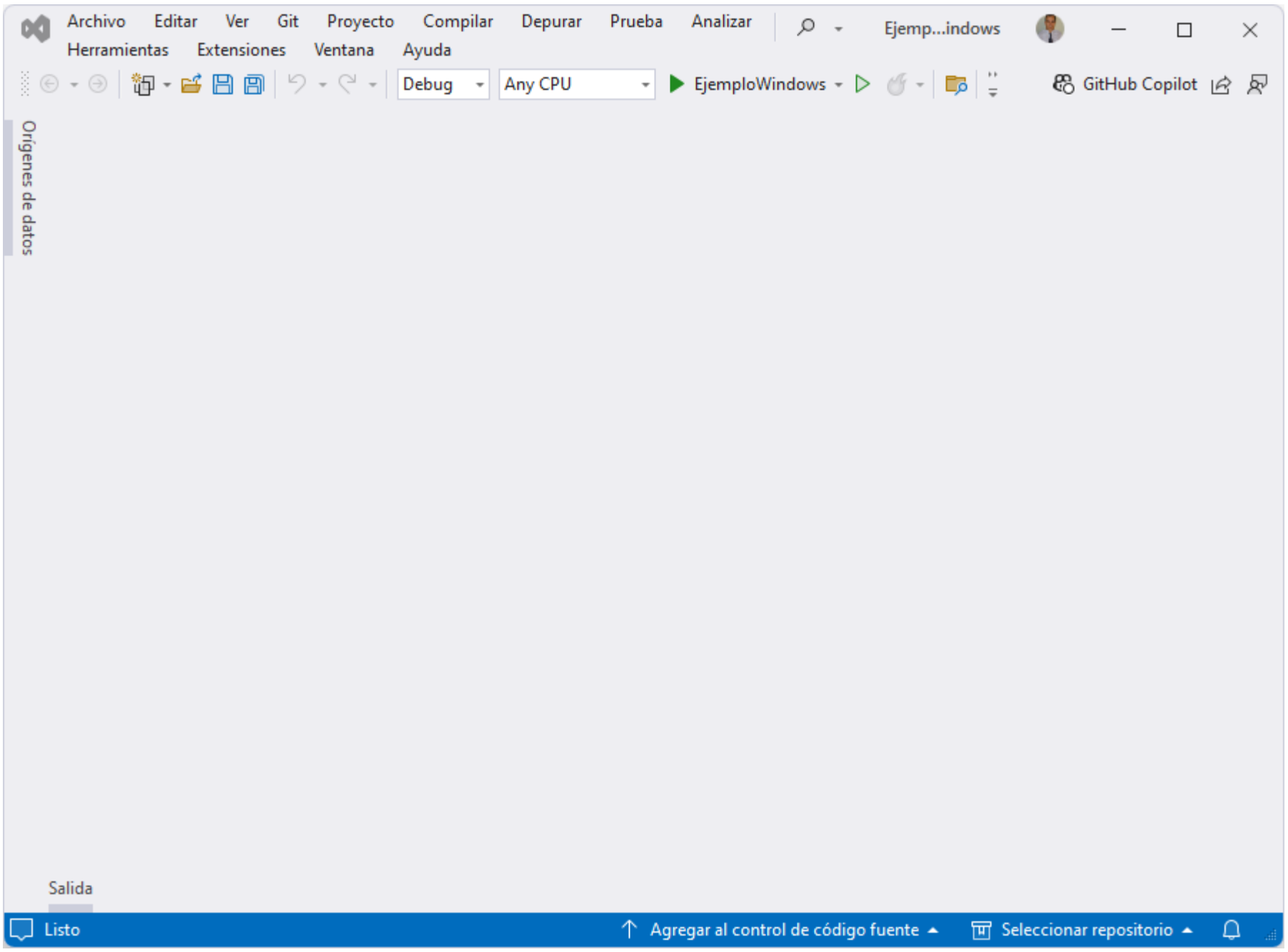


Ilustración 5: Todo cerrado.

Ahora se activa el "Explorador de soluciones"

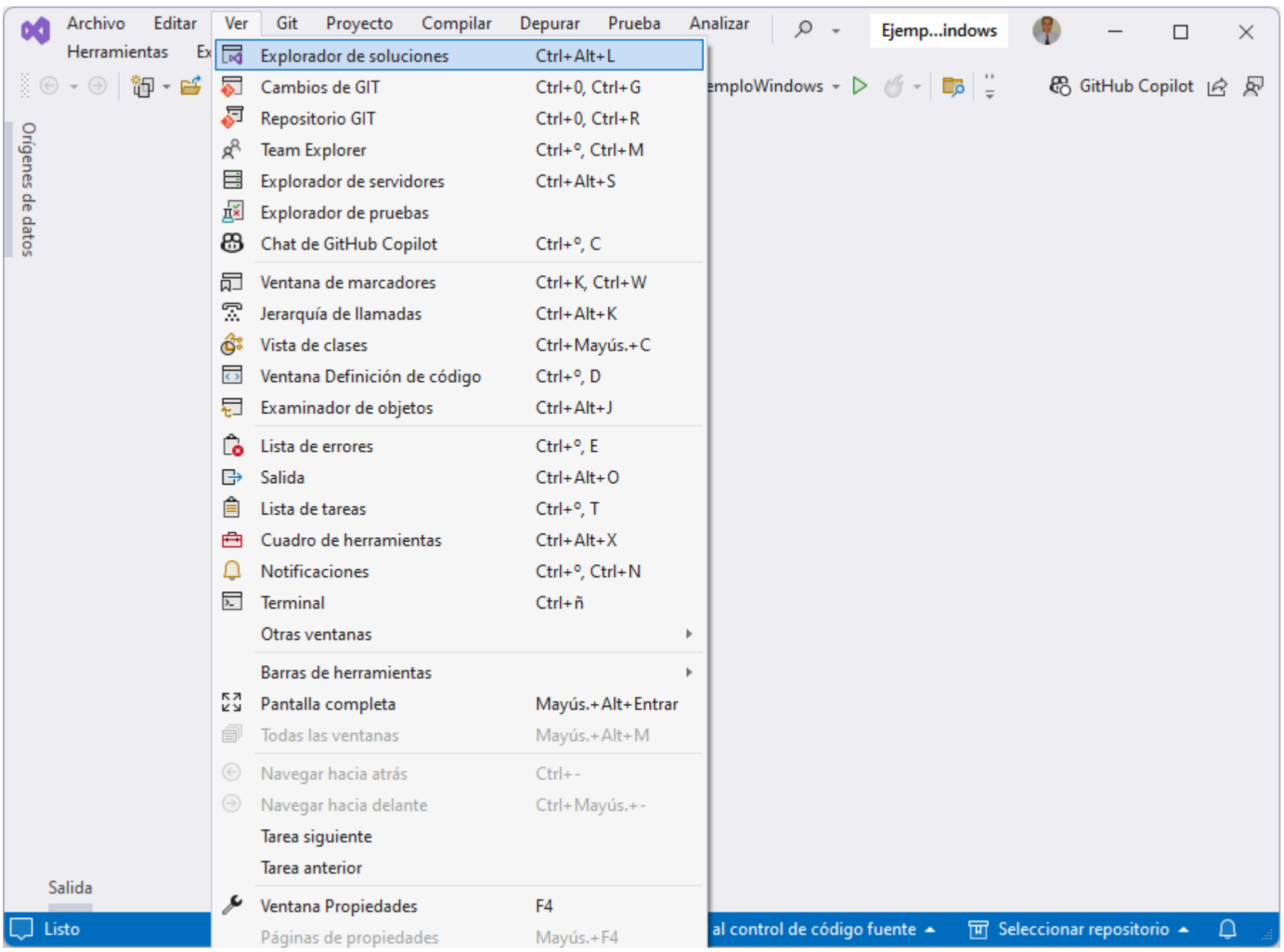


Ilustración 6: Ver -> Explorador de soluciones

Ilustración 7: La solución es visible, el siguiente paso es dar doble clic en Form1.cs

Y se abre la ventana

Ilustración 8: La ventana inicial de la solución

Accediendo a los diferentes componentes gráficos

Ver → Cuadro de herramientas

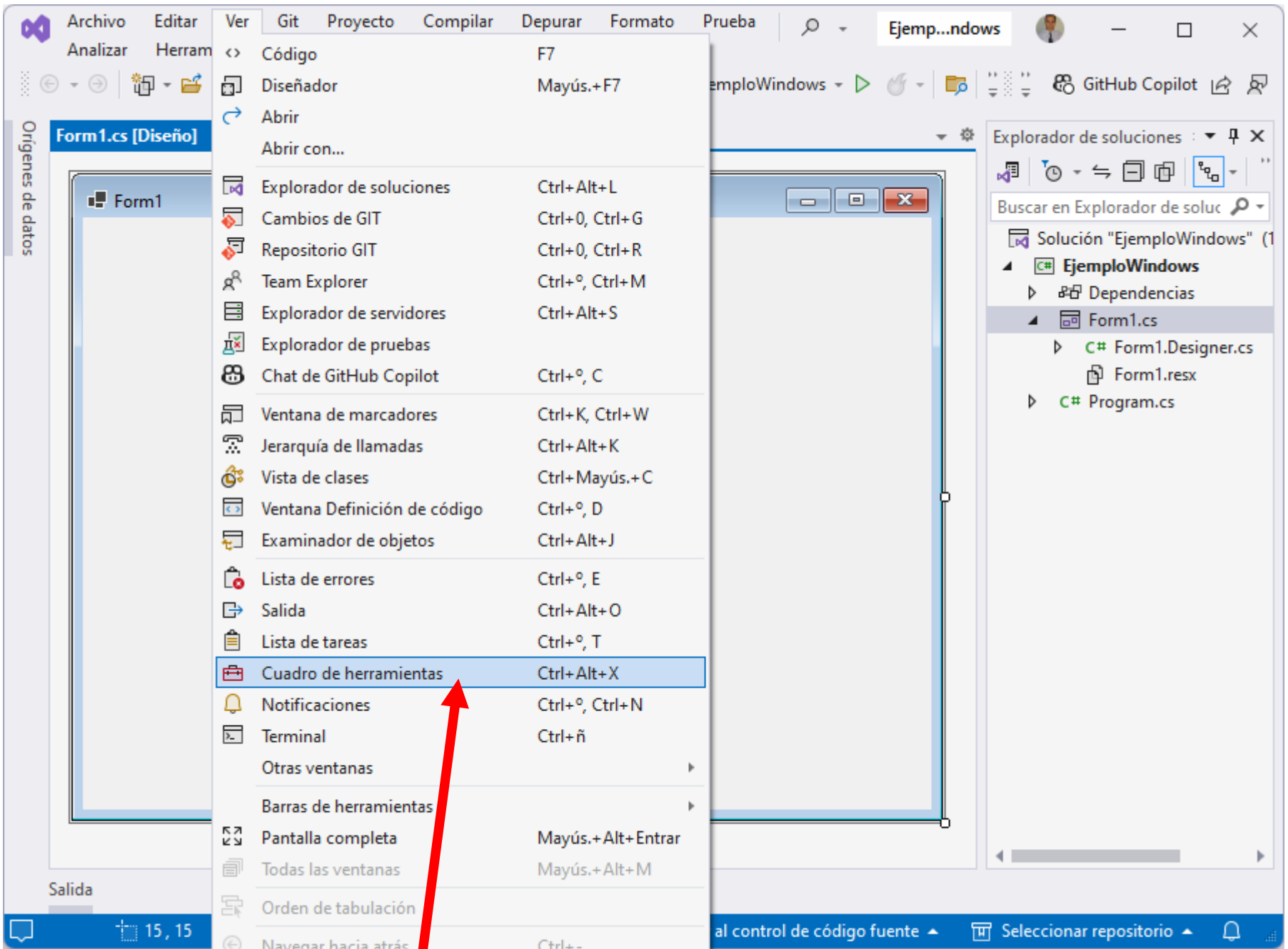


Ilustración 9: Ver → Cuadro de herramientas

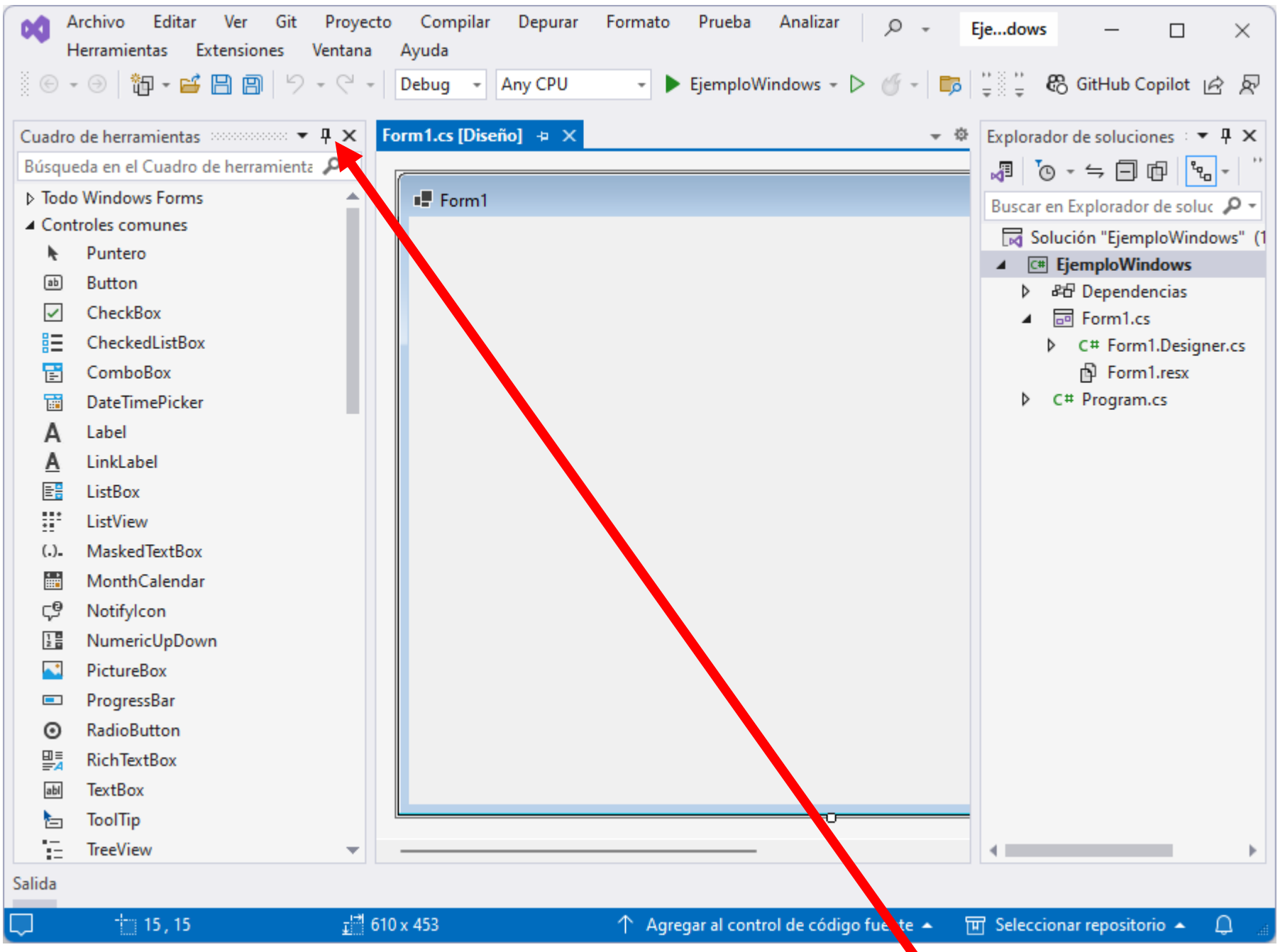


Ilustración 10: "Cuadro de herramientas", de clic en el pincho para que quede vertical y así fijar esa barra

El formulario o ventana

Al iniciar un proyecto de escritorio Windows en Visual Studio 2022, hay mínimo una ventana o formulario. De aquí en adelante se le llamará formulario.

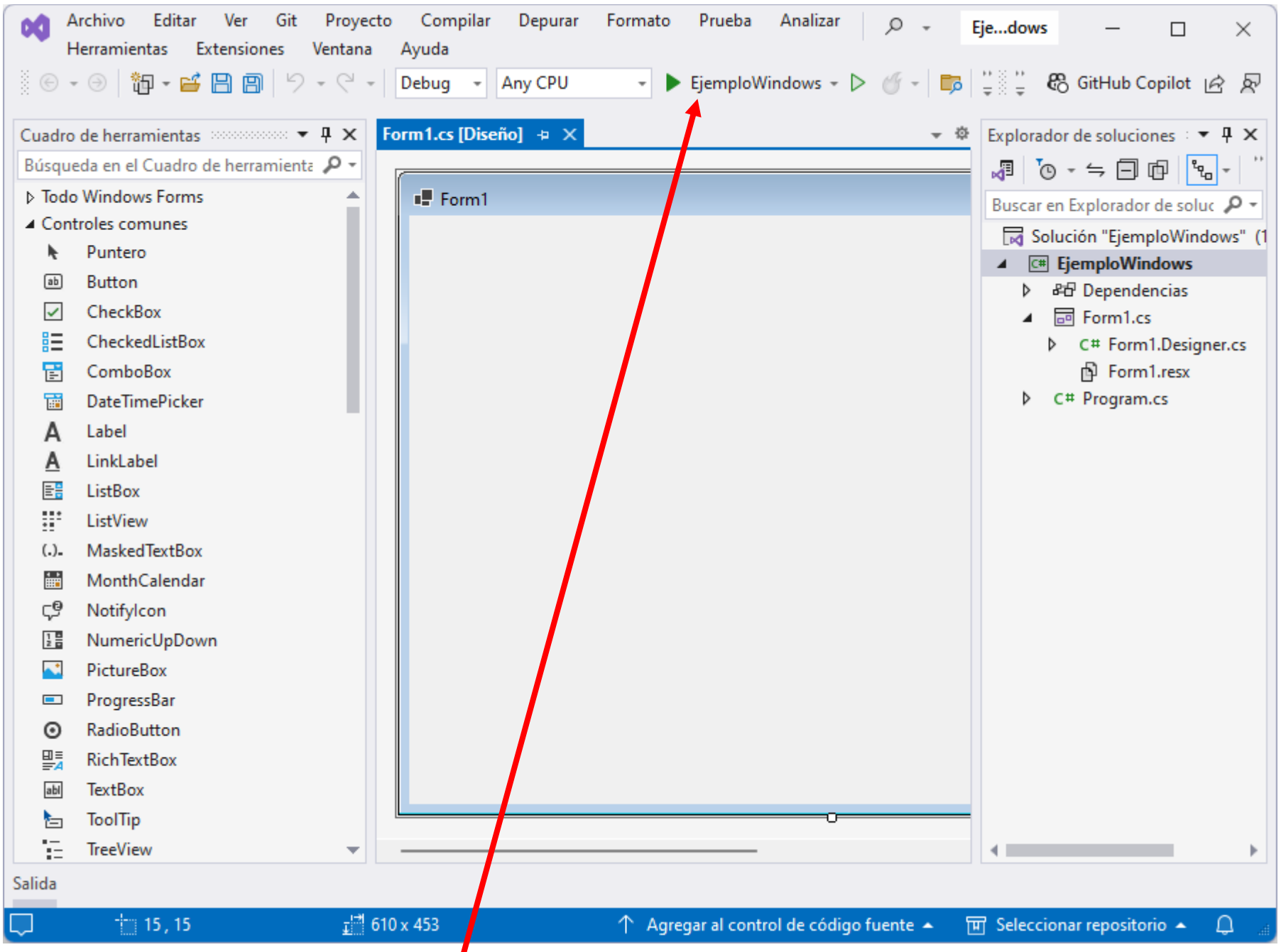


Ilustración 11: Inicia con una ventana o formulario

Se puede ejecutar la aplicación dando clic en el botón “Play” obteniendo una ventana vacía, pero con las funcionalidades de moverla, minimizarla, maximizarla y cambiarle el tamaño.

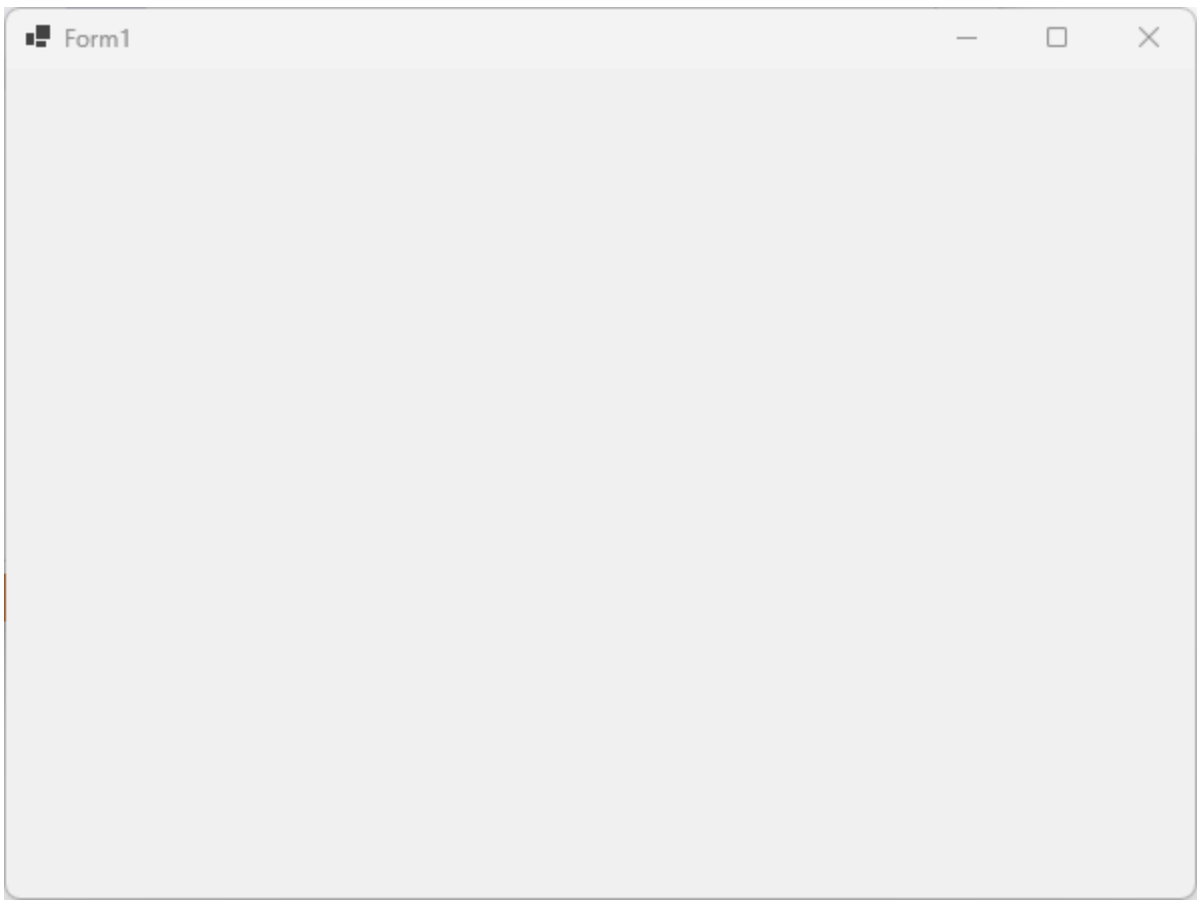


Ilustración 12: Ventana o formulario que se muestra al ejecutar el programa

¡OJO! Al ejecutar la aplicación y hacer modificaciones

Nota: Un error común al programar una aplicación de escritorio Windows es que se intenta programar cuando la aplicación está ejecutando, por lo que al tratar de programar o hacer cambios es imposible. Hay que verificar que **no** esté ejecutando la aplicación.

En este caso, la aplicación está ejecutando, se puede saber fácilmente viendo este botón.

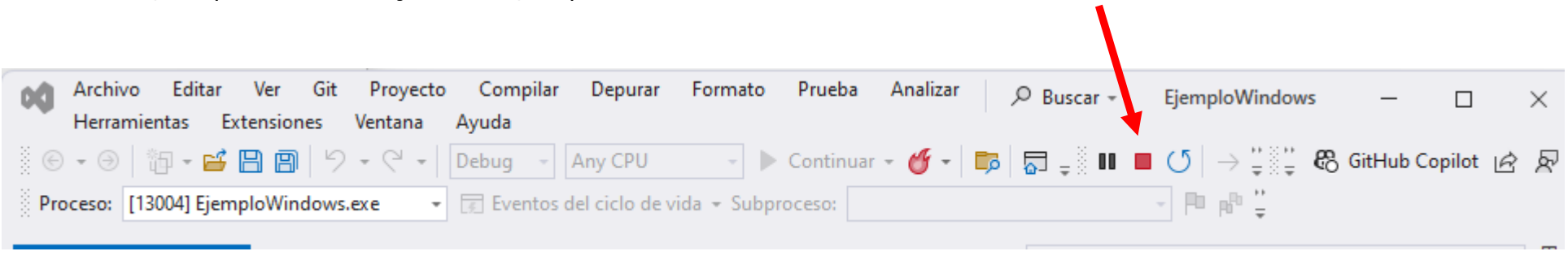


Ilustración 13: Nuestra aplicación ejecutando, debe cerrar antes de seguir programando

Hay varias formas de detener la ejecución de la aplicación, la primera es cerrando la ventana de la aplicación y la segunda es presionar ese botón rojo señalado.

Accediendo a las propiedades, métodos y eventos

Esa ventana o formulario (Form1) que se observa al inicio es un objeto, una instancia de System.Windows.Forms.Form, y como todo objeto hay acceso a sus atributos (propiedades), métodos y eventos. Para lograr eso, hay que asegurarse que la aplicación no esté ejecutando, es decir, que esté en modo de desarrollo el Visual Studio.

Al poner el cursor del ratón en algún lugar al interior de la ventana y dando clic con el botón derecho, se obtiene el siguiente menú:

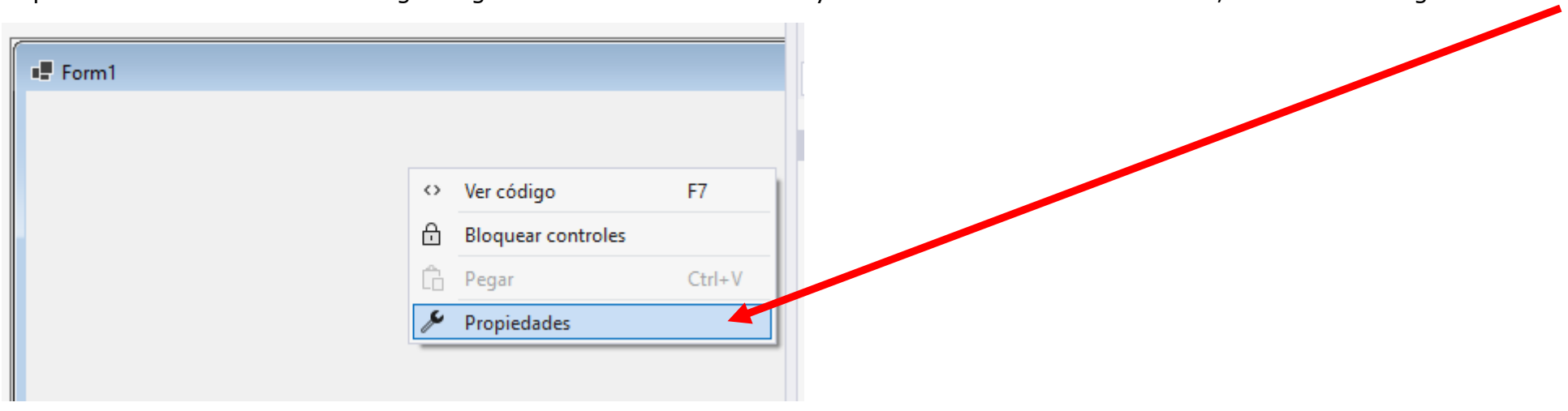


Ilustración 14: Menú contextual para acceder a las propiedades y eventos del objeto Form1. Se escoge "Propiedades"

Se selecciona propiedades y se obtiene un listado de propiedades y sus valores de ese objeto Form1

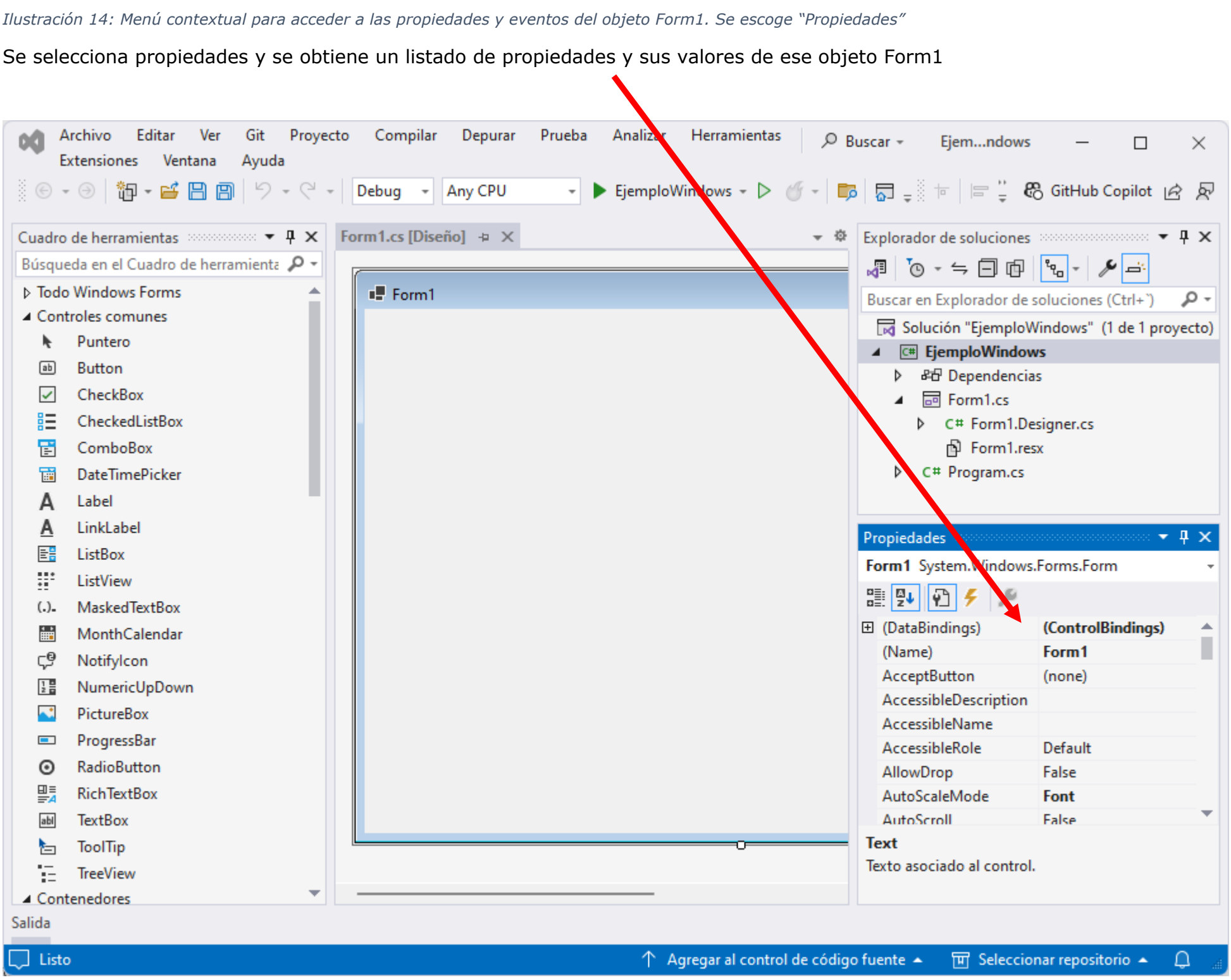


Ilustración 15: Listado de propiedades y sus valores para ese objeto Form1

Propiedades de Apariencia

Aquí aparecen propiedades que son:

BackColor

Color de fondo de la ventana. Al dar clic allí, aparece un listado con tres columnas: Personalizado, Web y Sistema

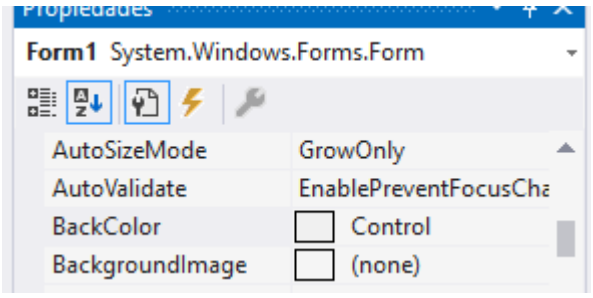


Ilustración 16: Color de fondo. Categoría: Sistema

En color Web, se tiene acceso a colores predefinidos

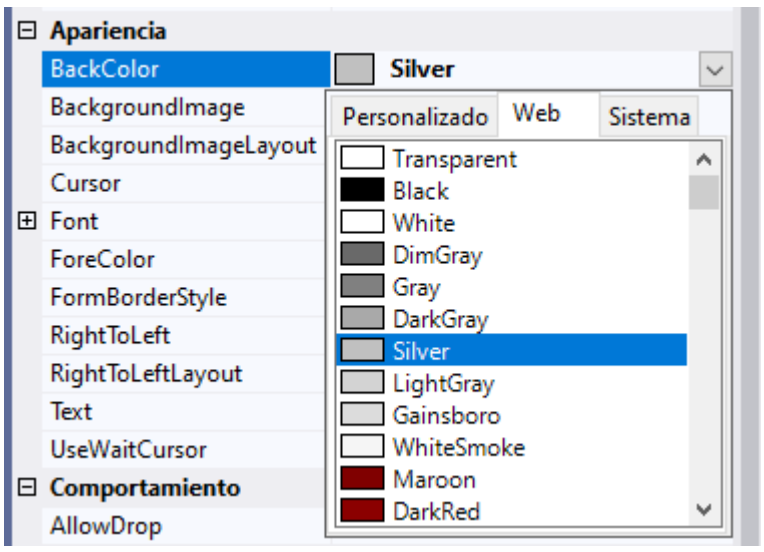


Ilustración 17: Colores fijos Web

En color Personalizado, se tiene acceso a una paleta de colores:

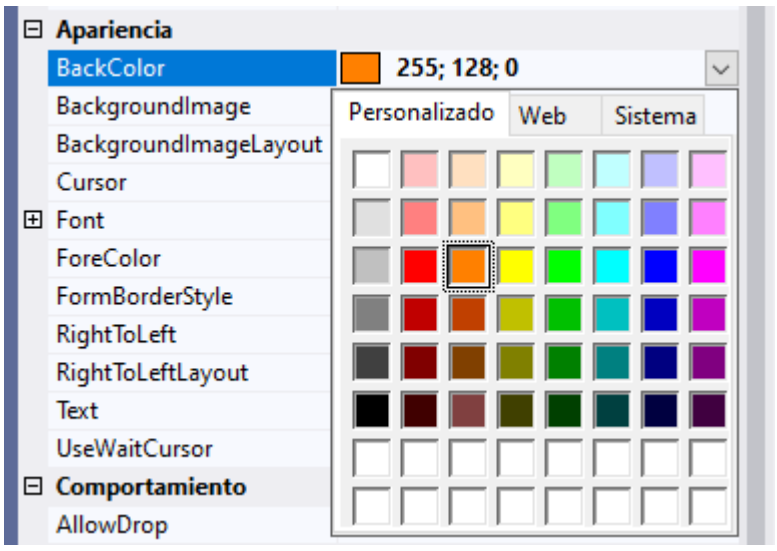


Ilustración 18: Paleta de colores y para personalizar

Al posar el cursor del ratón en alguna de las casillas blancas al final y dar clic botón derecho, aparece este cuadro de diálogo:

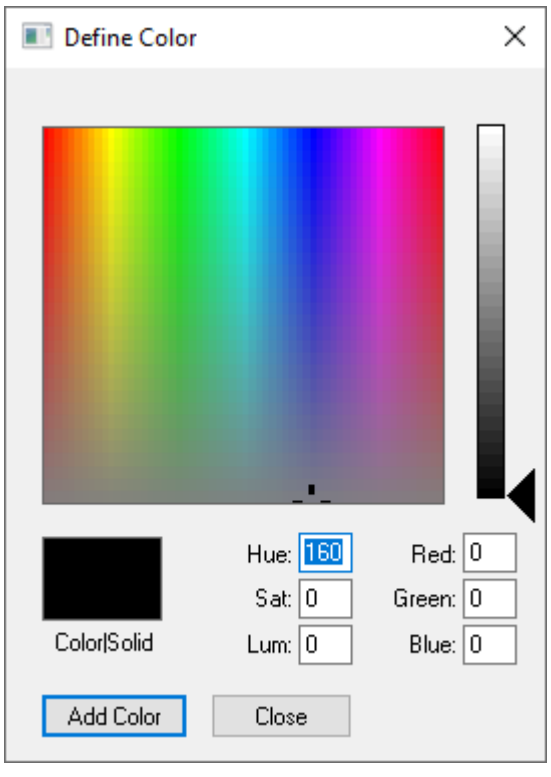


Ilustración 19: Para seleccionar algún color al gusto

Allí se puede escoger el color deseado y posteriormente se presiona el botón “Add Color”, y luego “Close”

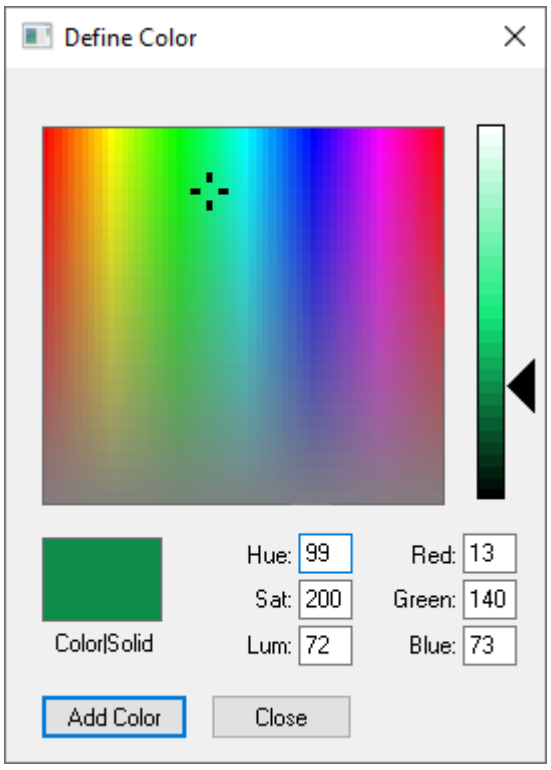


Ilustración 20: Se selecciona un color

El color personalizado queda guardado en la paleta para un uso posterior

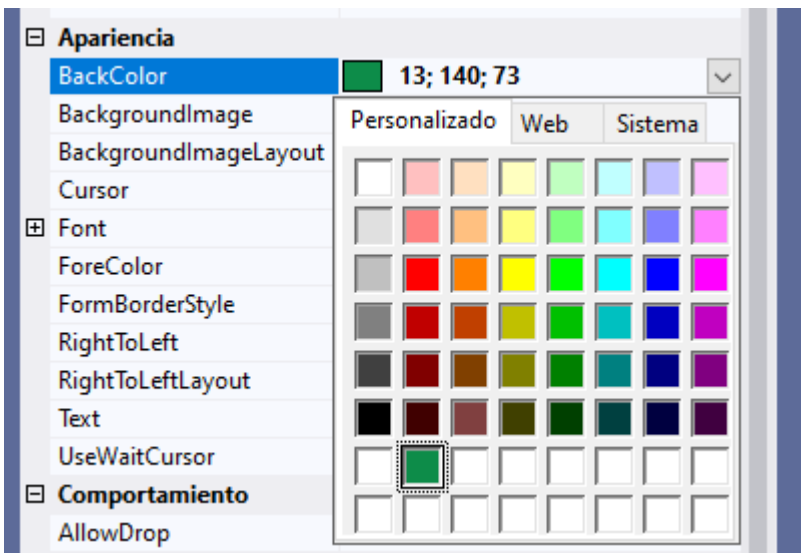


Ilustración 21: El nuevo color personalizado queda en la paleta

Y la ventana queda con ese color

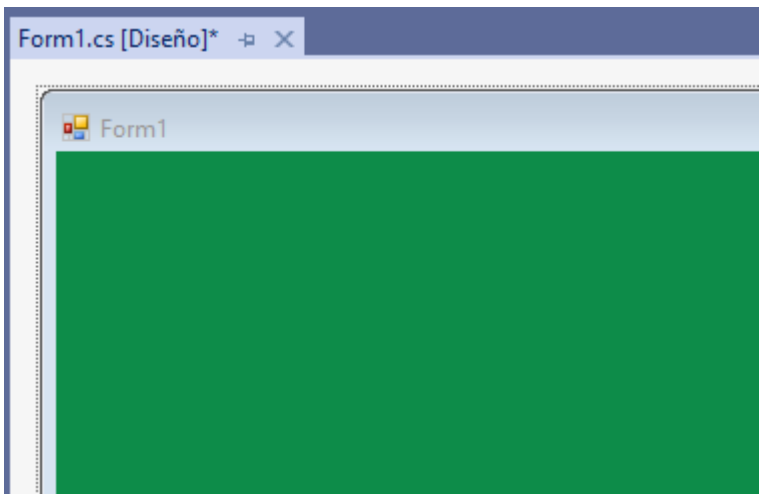


Ilustración 22: Y la ventana tendrá un fondo con el color personalizado

BackgroundImage

Permite poner una imagen de fondo. Es una propiedad que se debe tener especial cuidado porque mal usada pueden enlentecer considerablemente la aplicación y hacer que el ejecutable crezca muchísimo.

Tenemos por ejemplo esta imagen:

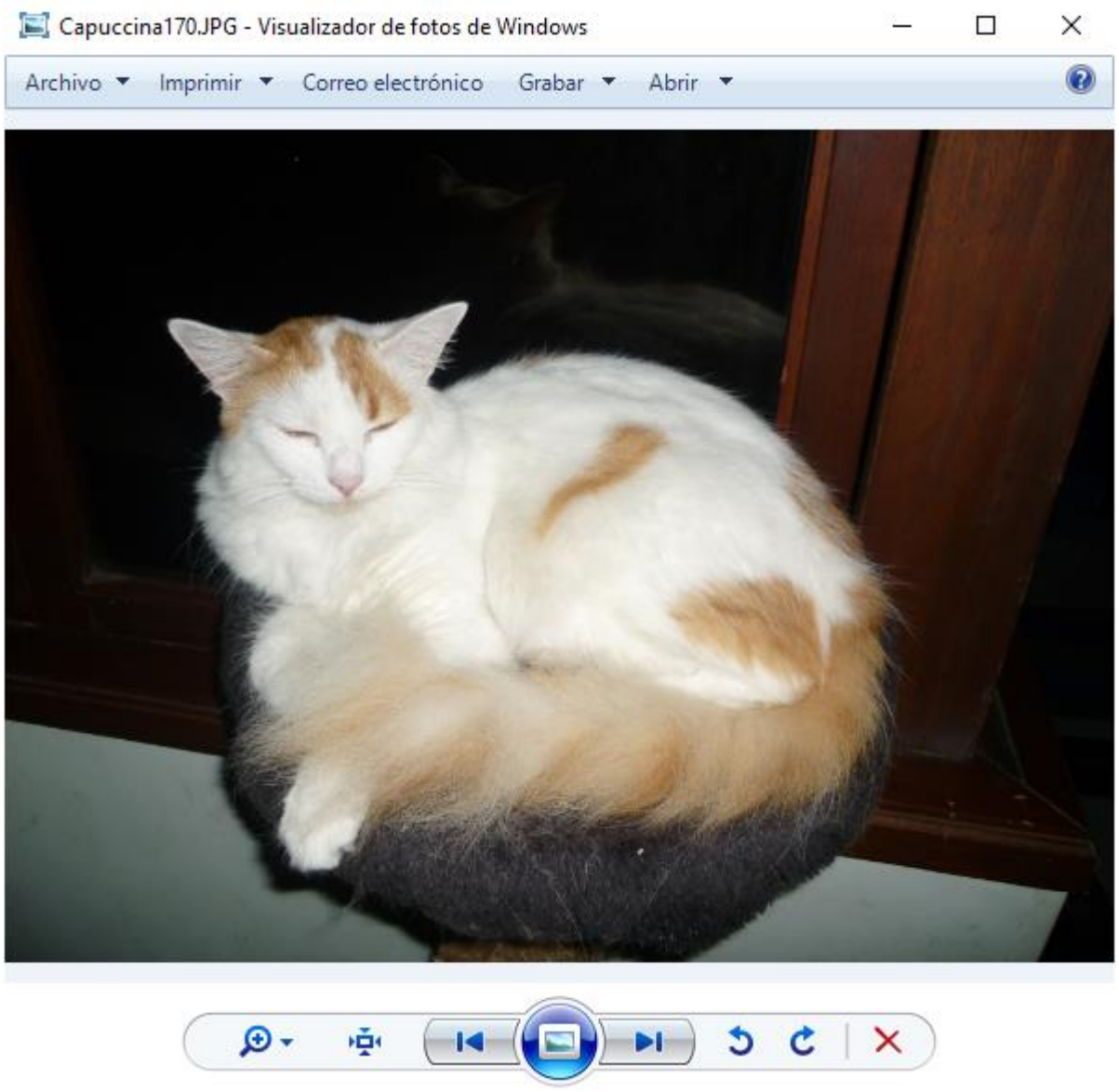


Ilustración 23: Una imagen. En este caso mi gata "Capuchina"

Estos son los datos de esa imagen. El tamaño de archivo es grande.

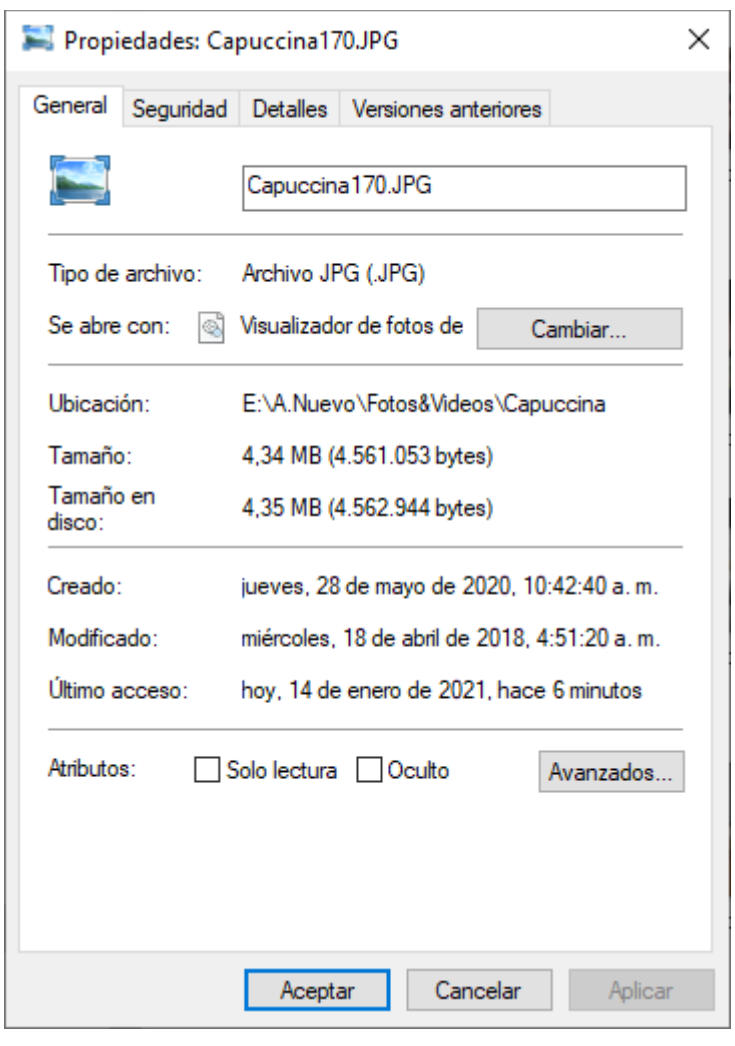


Ilustración 24: Datos de la imagen.

Procedemos a ponerla como imagen de fondo

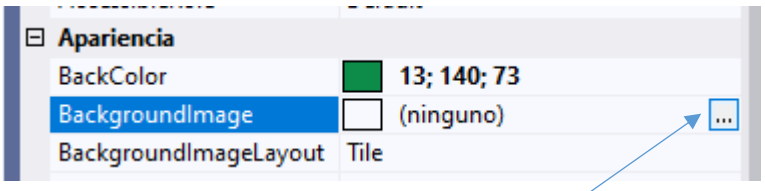


Ilustración 25: Imagen de fondo para el control

Al presionar el botón de la derecha se muestra la siguiente ventana:

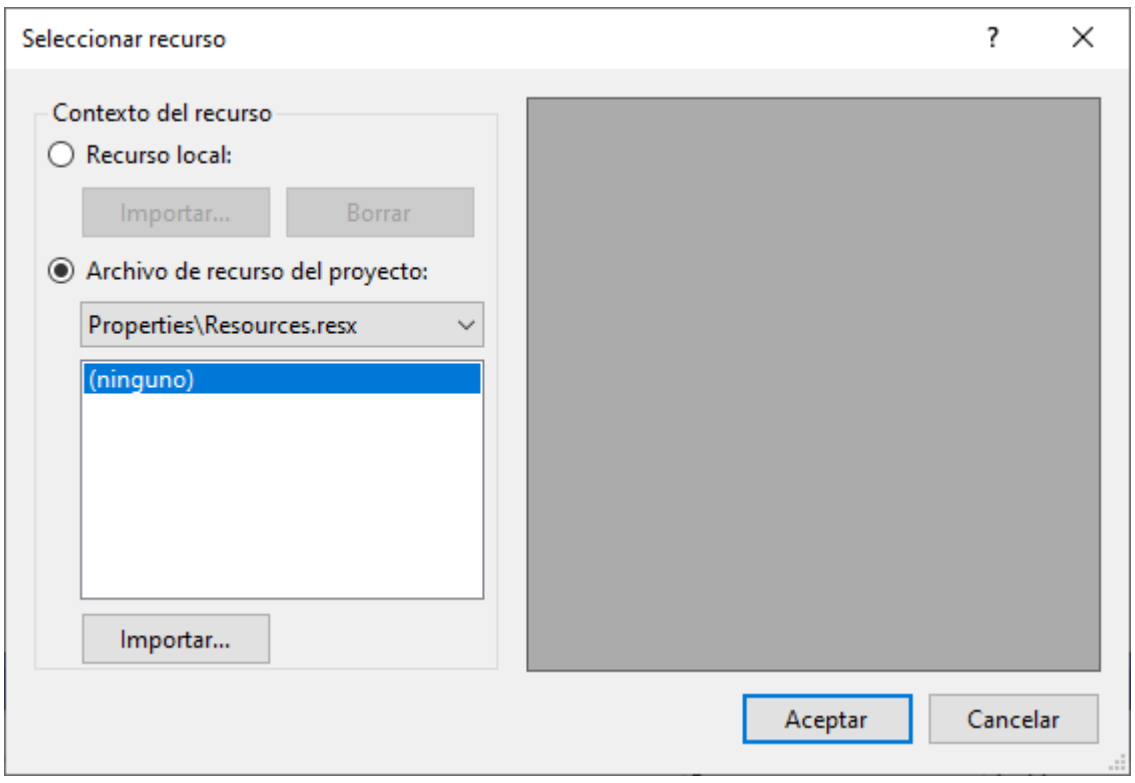


Ilustración 26: Ventana de diálogo para importar imagen

Hay dos opciones: "Recurso local:" y "Archivo de recurso del proyecto:". Si seleccionamos "Recurso local:", se activa el botón de "Importar…" que nos lleva al siguiente cuadro de diálogo para importar la imagen:

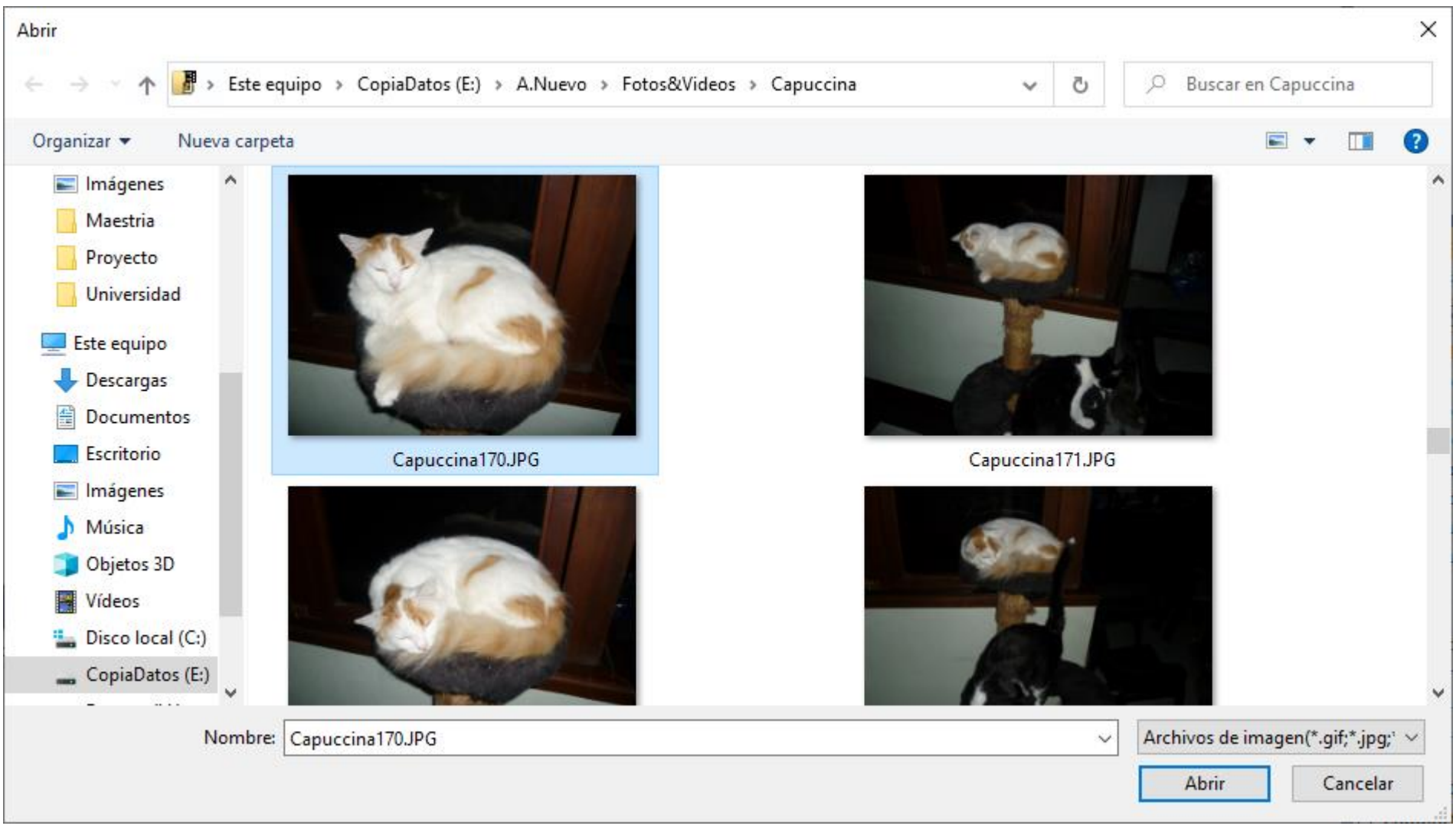


Ilustración 27: Listado de imágenes

Se da clic en "Abrir" y la imagen es importada.

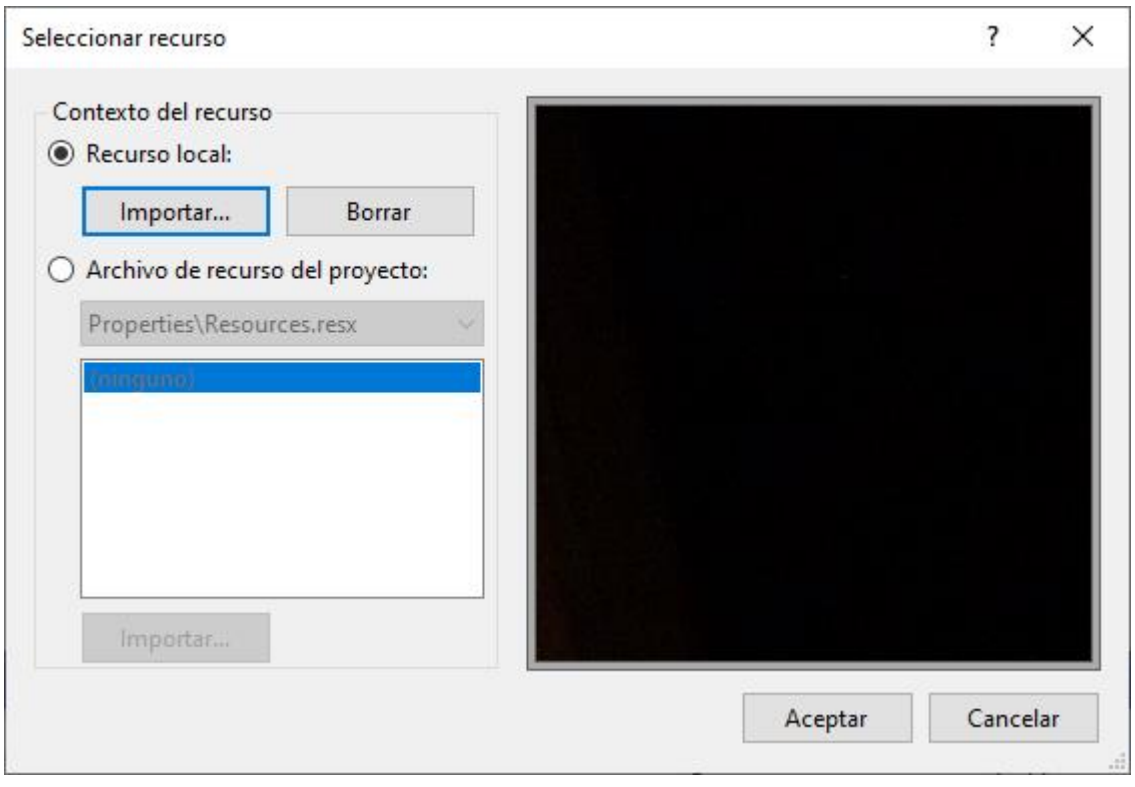


Ilustración 28: Se ha importado la imagen

El problema es que la imagen es gigantesca y sólo se ve la esquina superior izquierda en la vista previa. Eso ya es una advertencia que estamos trabajando con un recurso de imagen muy pesado.

Una forma de ver el impacto de importar un recurso tan pesado es en el directorio del proyecto donde podemos observar el archivo "Form1.resx" con un pequeño tamaño de 6kb

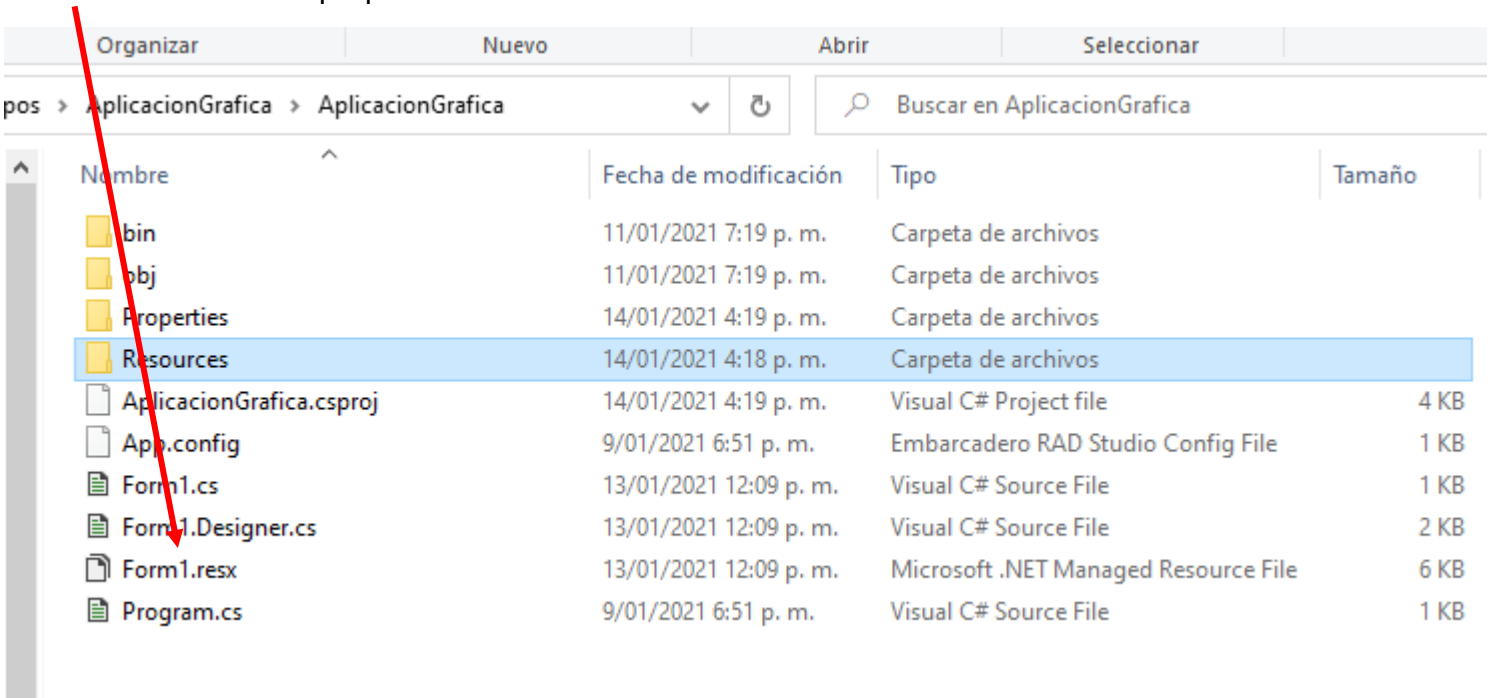


Ilustración 29: Archivo "Form1.resx" con un tamaño de 6Kb

Pero al importar el archivo, obtenemos que la imagen no cabe en la ventana, sólo se está viendo la esquina superior izquierda de la imagen

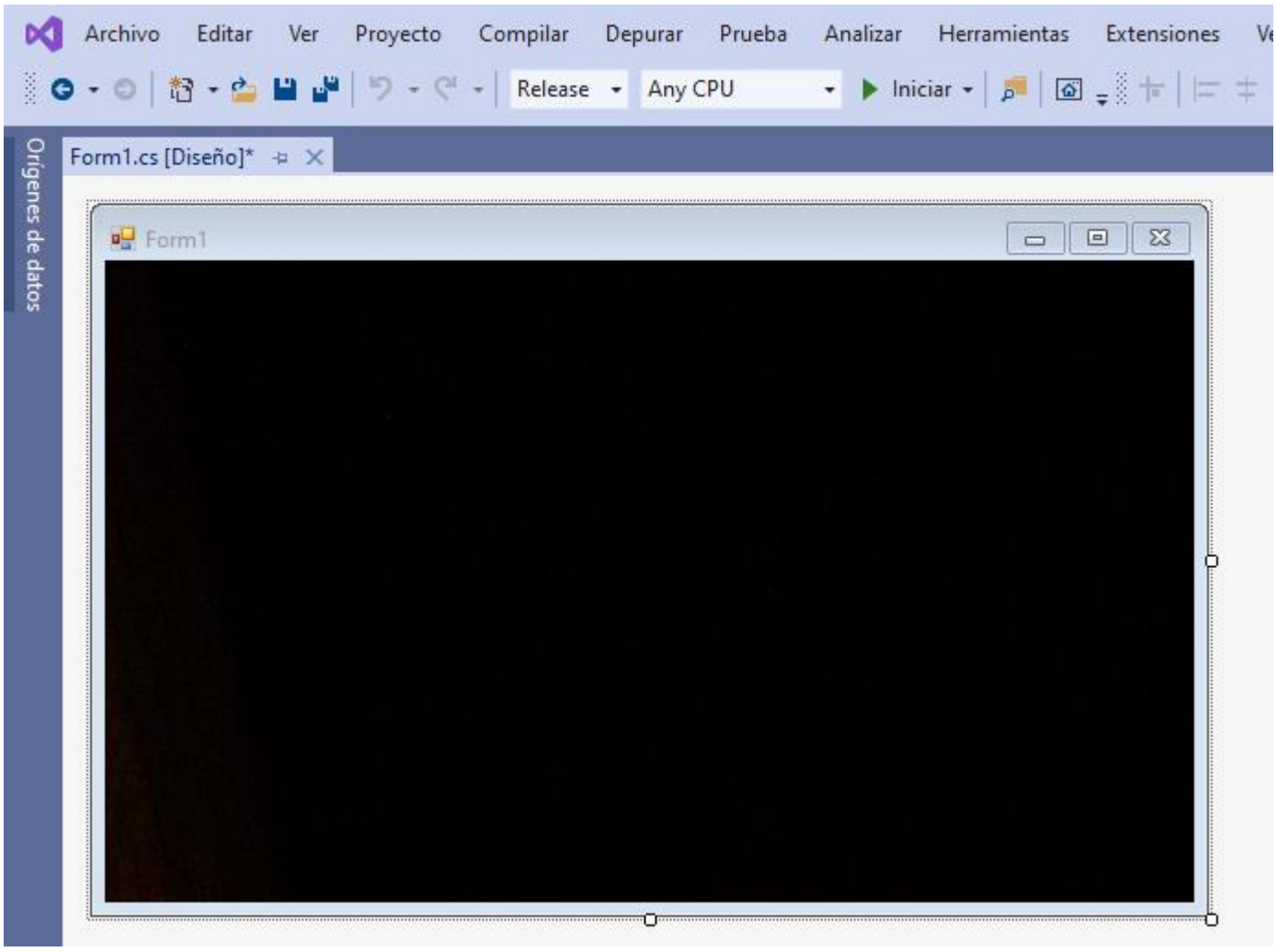


Ilustración 30: Imagen de fondo en la ventana. Sólo se ve la esquina superior izquierda.

Y el archivo “Form1.resx” ha crecido considerablemente

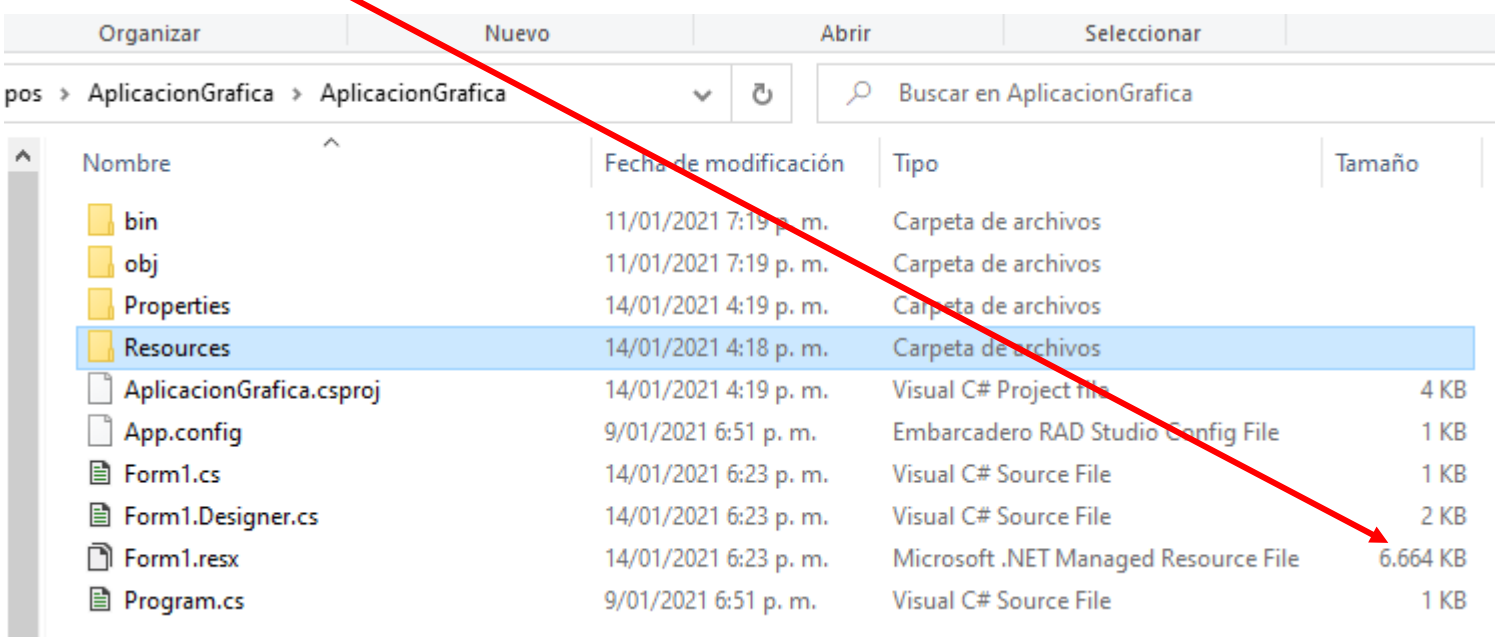


Ilustración 31: El archivo "Form1.resx" crece considerablemente

Adelantándonos un poco, si se quiere ver la imagen, debemos modificar la propiedad “BackgroundImageLayout” y escoger la opción “Stretch”

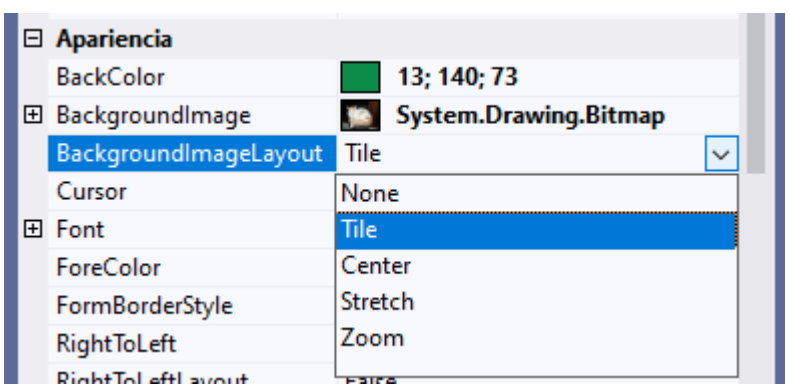


Ilustración 32: Se pueden hacer transformaciones de la presentación de la imagen de fondo con esta propiedad

Y se obtiene el siguiente resultado:

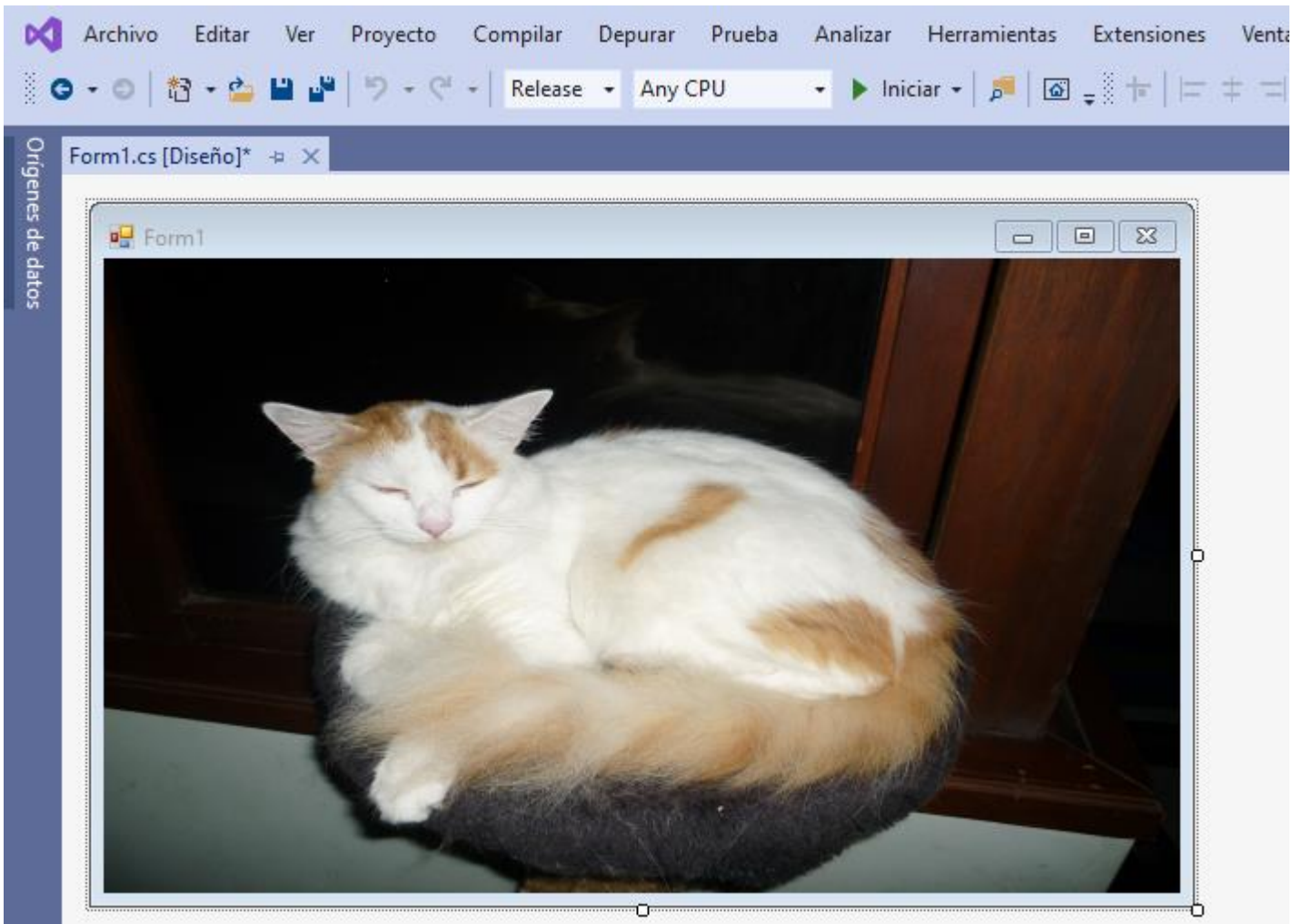


Ilustración 33: Imagen de fondo aplicando el valor "Stretch"

La imagen es visible ahora mismo, pero el problema del peso persiste y eso hará nuestra aplicación muy lenta, y esto es visible al generar el ejecutable en “Release”, que es muy grande.

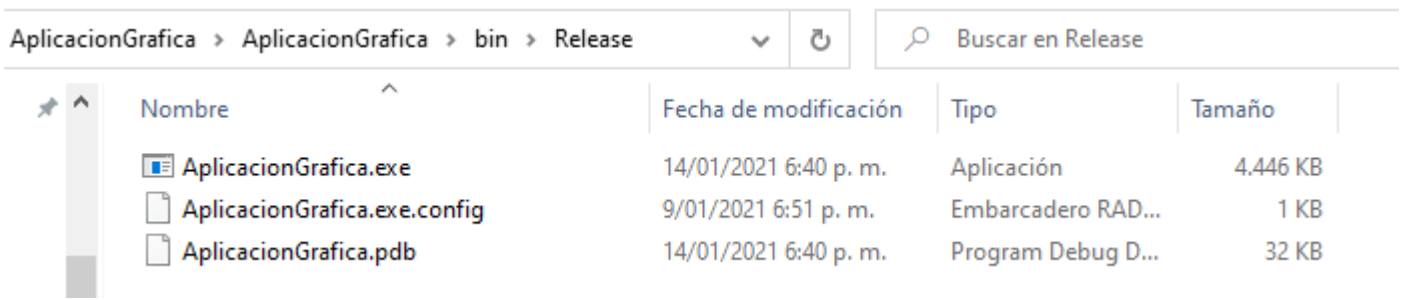


Ilustración 34: Tamaño del ejecutable inflado por la imagen de fondo

La conclusión es NO usar imágenes con archivos muy pesados. Primero hay que hacerle edición con algún software especializado que le reduzca el tamaño, para después si hacer uso como imagen de fondo.

Primero borramos el “Recurso local” y luego se usa la otra forma “Archivo de recurso del proyecto” y damos clic en “Importar”

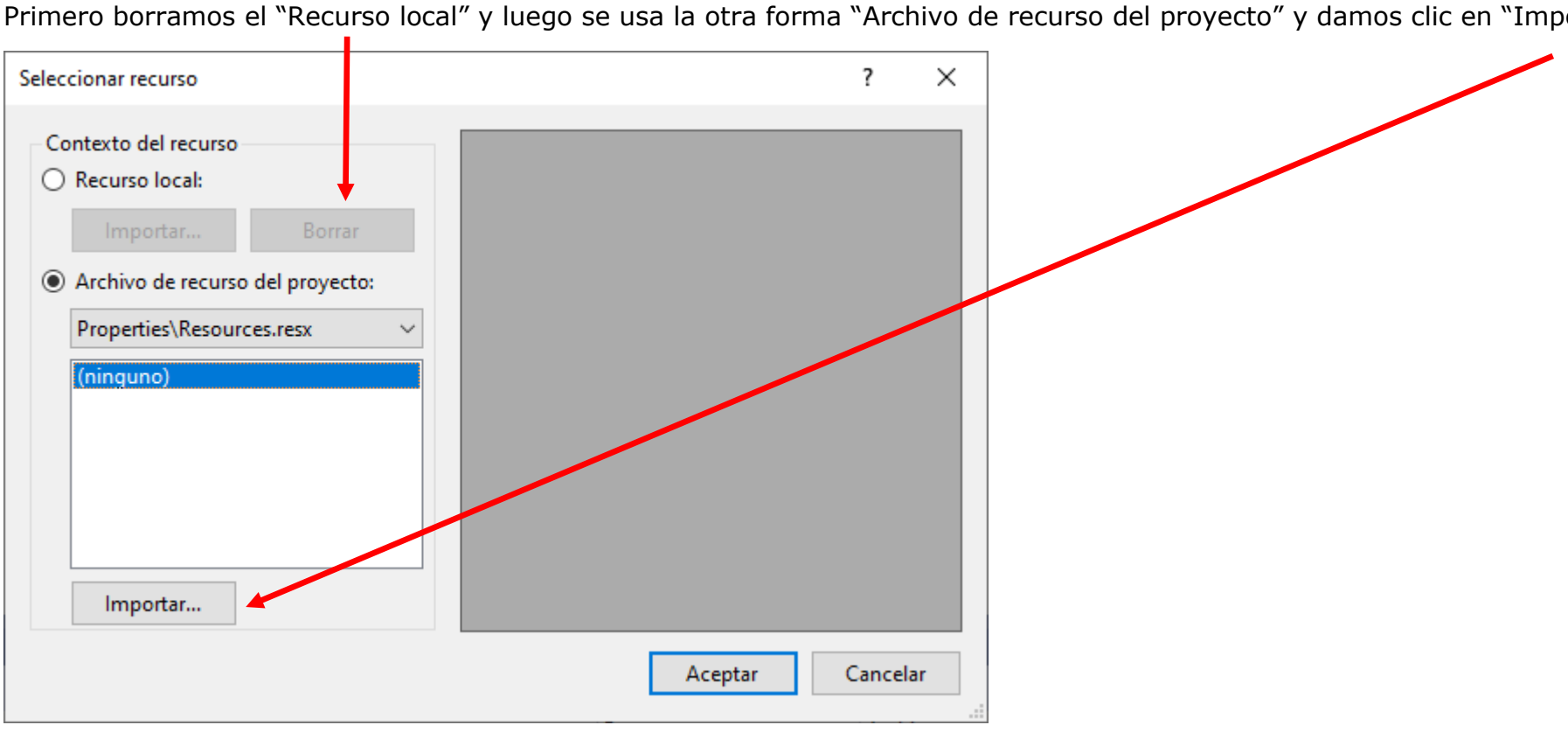


Ilustración 35: Segunda forma de usar una imagen de fondo

Es similar, se selecciona alguna imagen

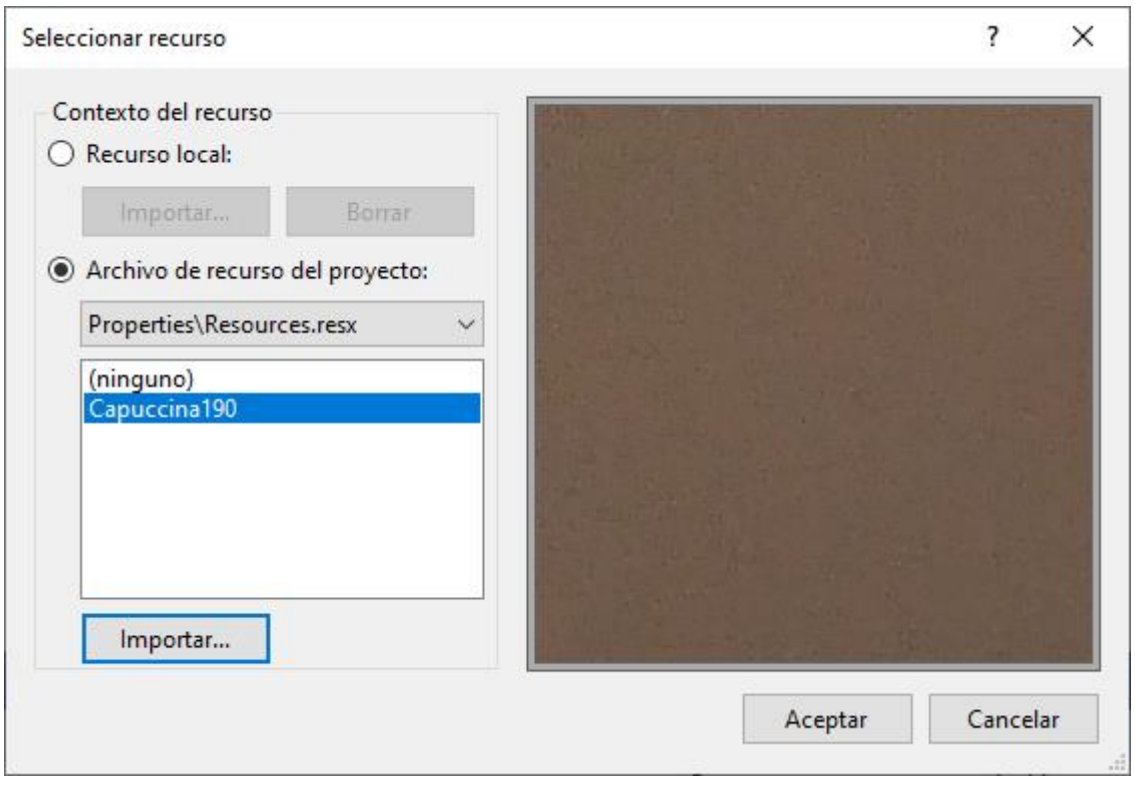


Ilustración 36: Uso de importar "Archivo de recurso del proyecto:"

Y aparece la imagen como fondo de la ventana, por supuesto, la propiedad "BackgroundImageLayout" está con el valor "Stretch"

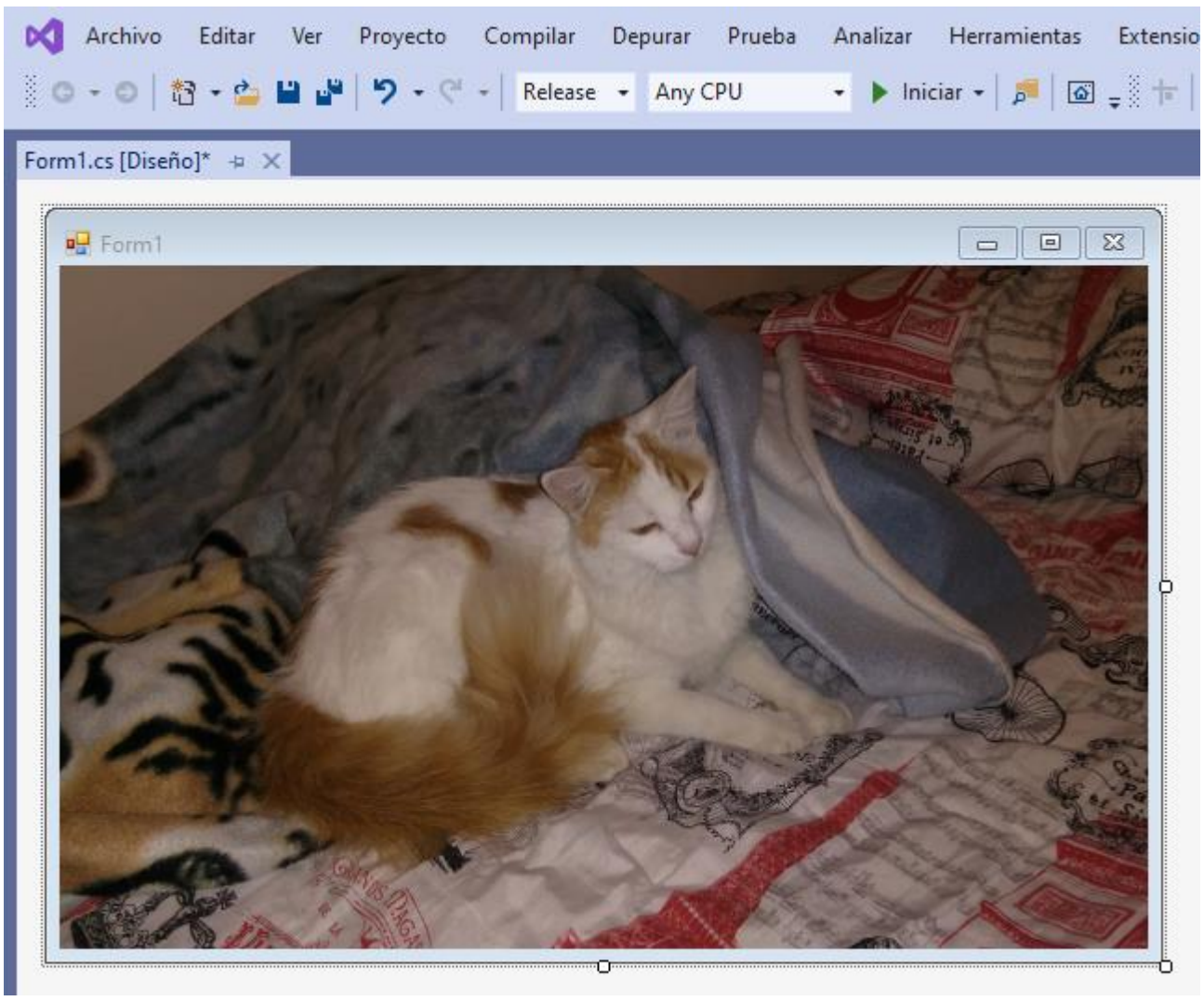


Ilustración 37: La imagen de fondo

La diferencia es que en el proyecto se almacena una copia del archivo de imagen en la carpeta resources

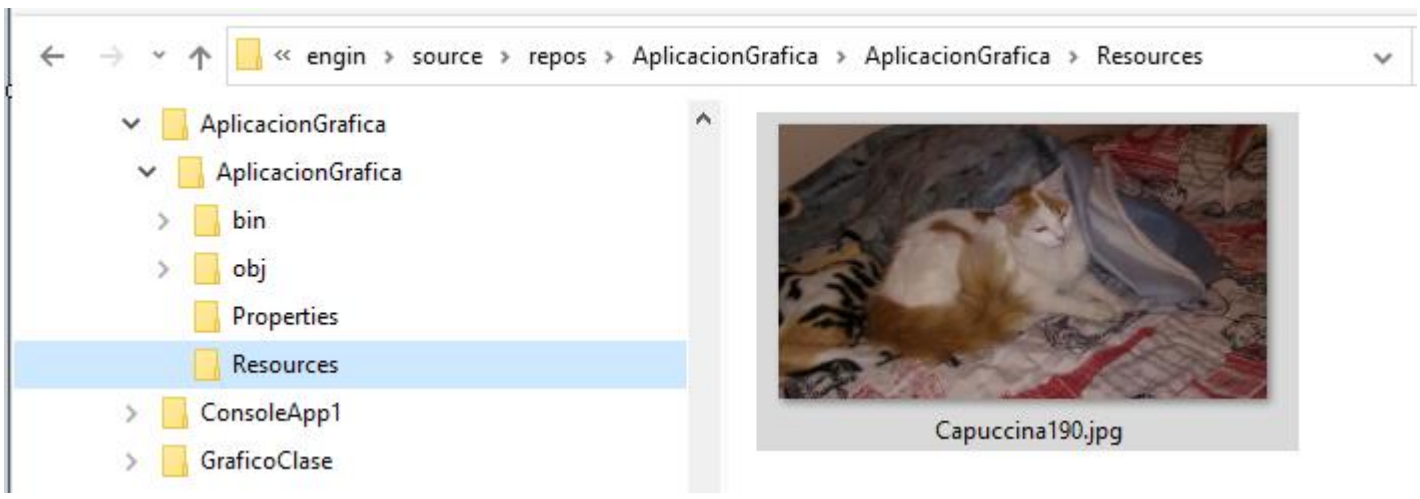


Ilustración 38: Una copia del archivo de imagen queda en "Resources" dentro de la carpeta de la solución

Y esa imagen se puede apreciar en el “Explorador de soluciones”

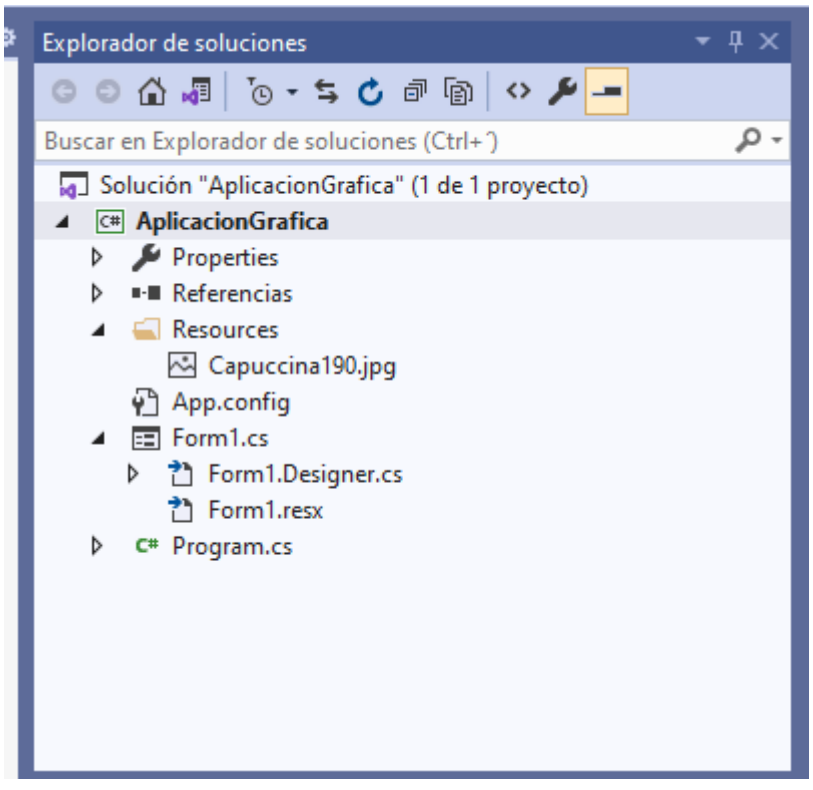
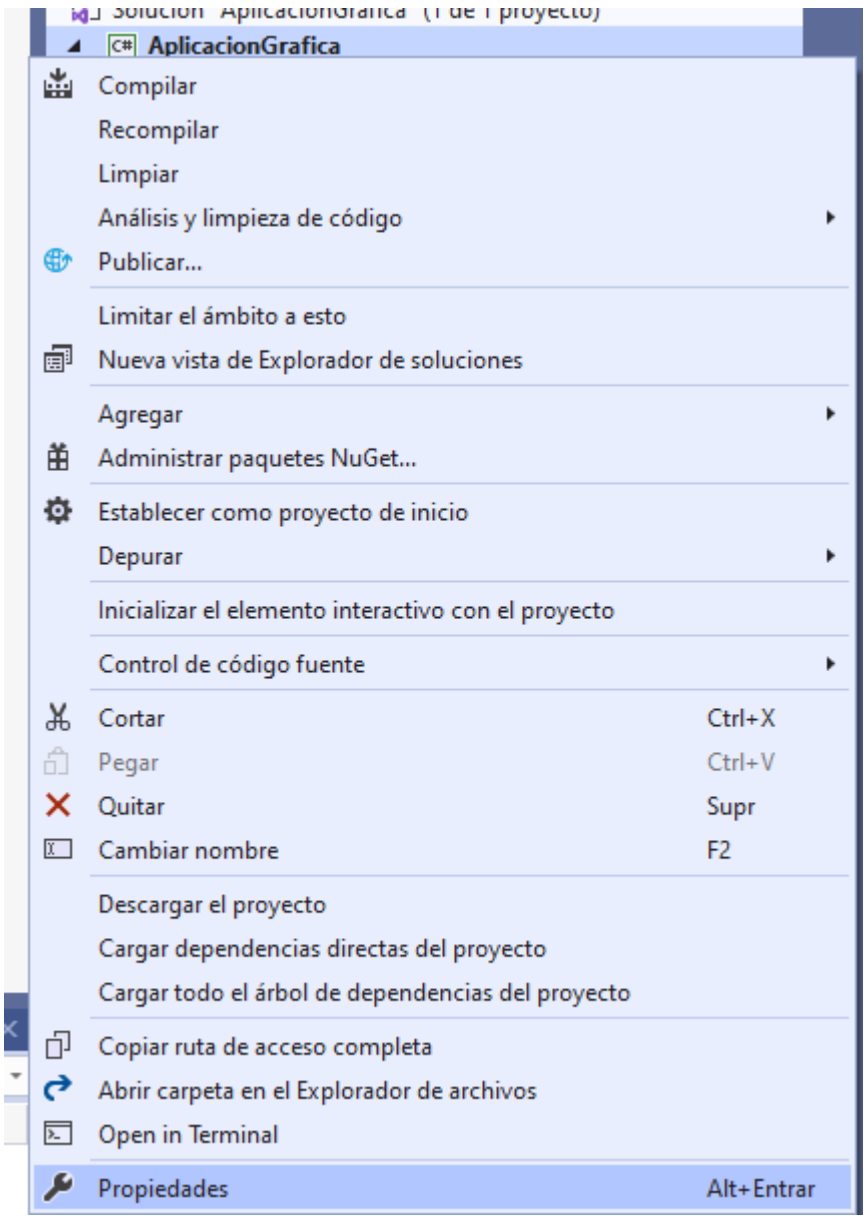


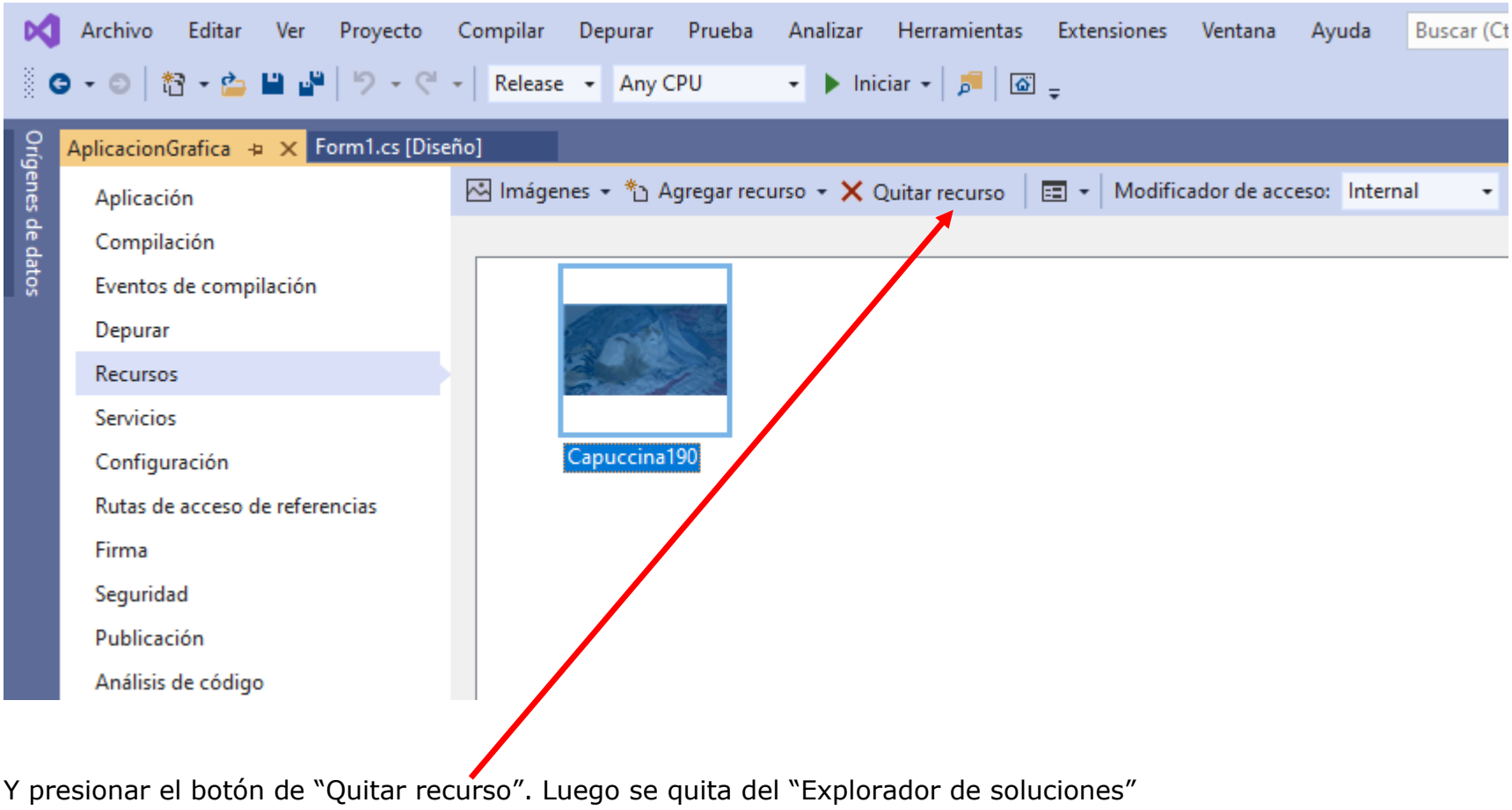
Ilustración 39: En el explorador de soluciones se observa el archivo de imagen

De resto es un comportamiento similar, en el que la imagen queda dentro del ejecutable (.exe) y presentaría los mismos problemas de lentitud.

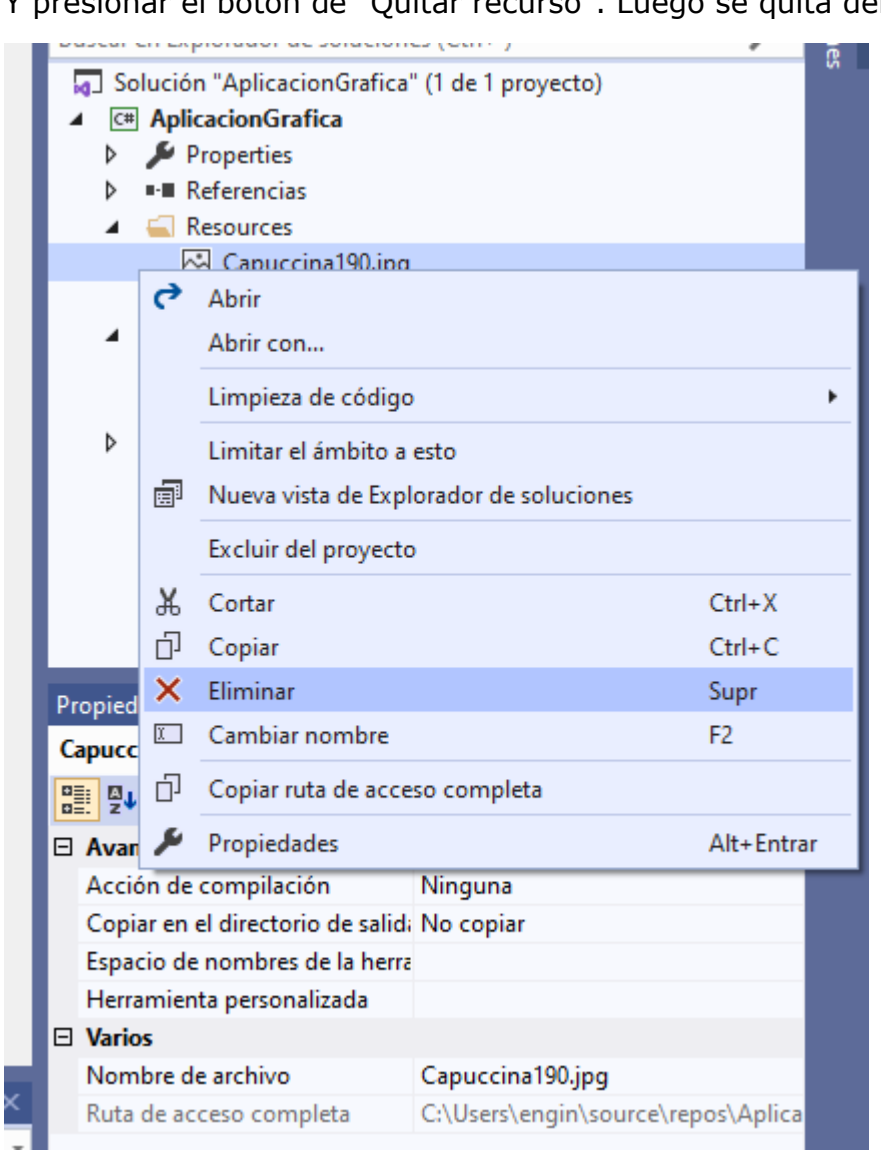
Para quitar correctamente esa imagen como recurso debe dar clic botón derecho sobre el nombre de la solución y seleccionar “Propiedades”:

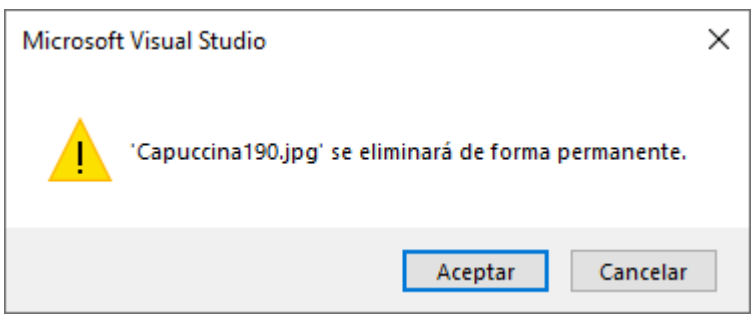


En la ventana que se abre, debe seleccionar “Recursos”

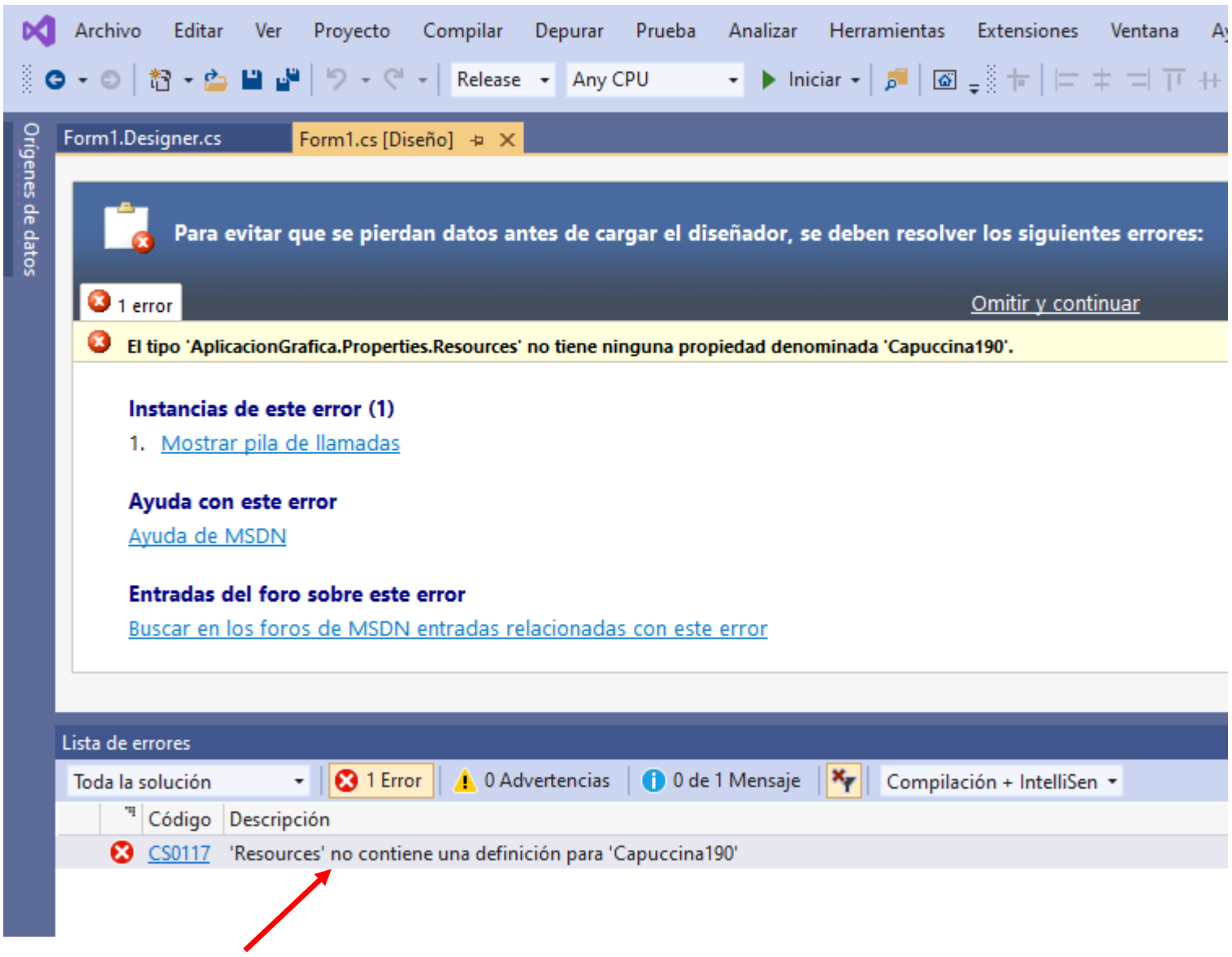


Y presionar el botón de “Quitar recurso”. Luego se quita del “Explorador de soluciones”

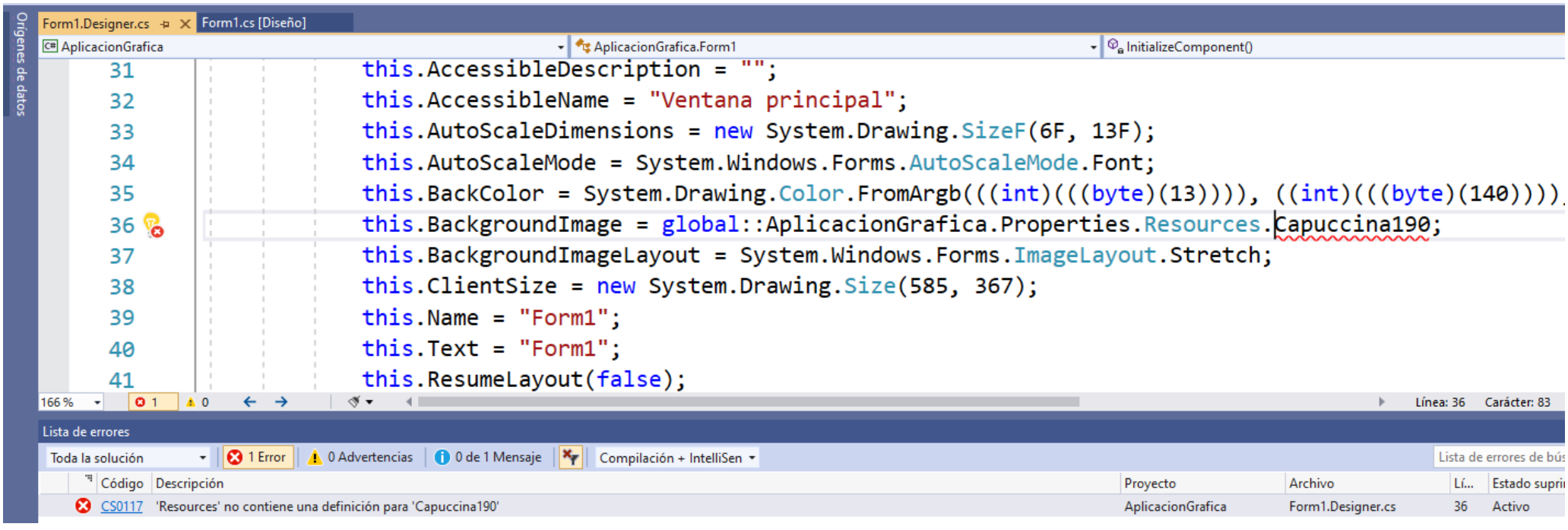




Nos puede aparecer este error:



Damos doble clic en el error (no en CS0117) y nos llevará al archivo Form1.Designer.cs, que sería el archivo "detrás de escena" que controla el diseño gráfico.



Simplemente se borra esa línea:

```
this.BackgroundImage = global::AplicacionGrafica.Properties.Resources.Capuccina190;
```

Y eso es todo, volvemos al diseño normalmente.

BackgroundImageLayout

Para decidir como mostrar la imagen de fondo. Ejemplo: La imagen original mostrada por “Vista Previa”

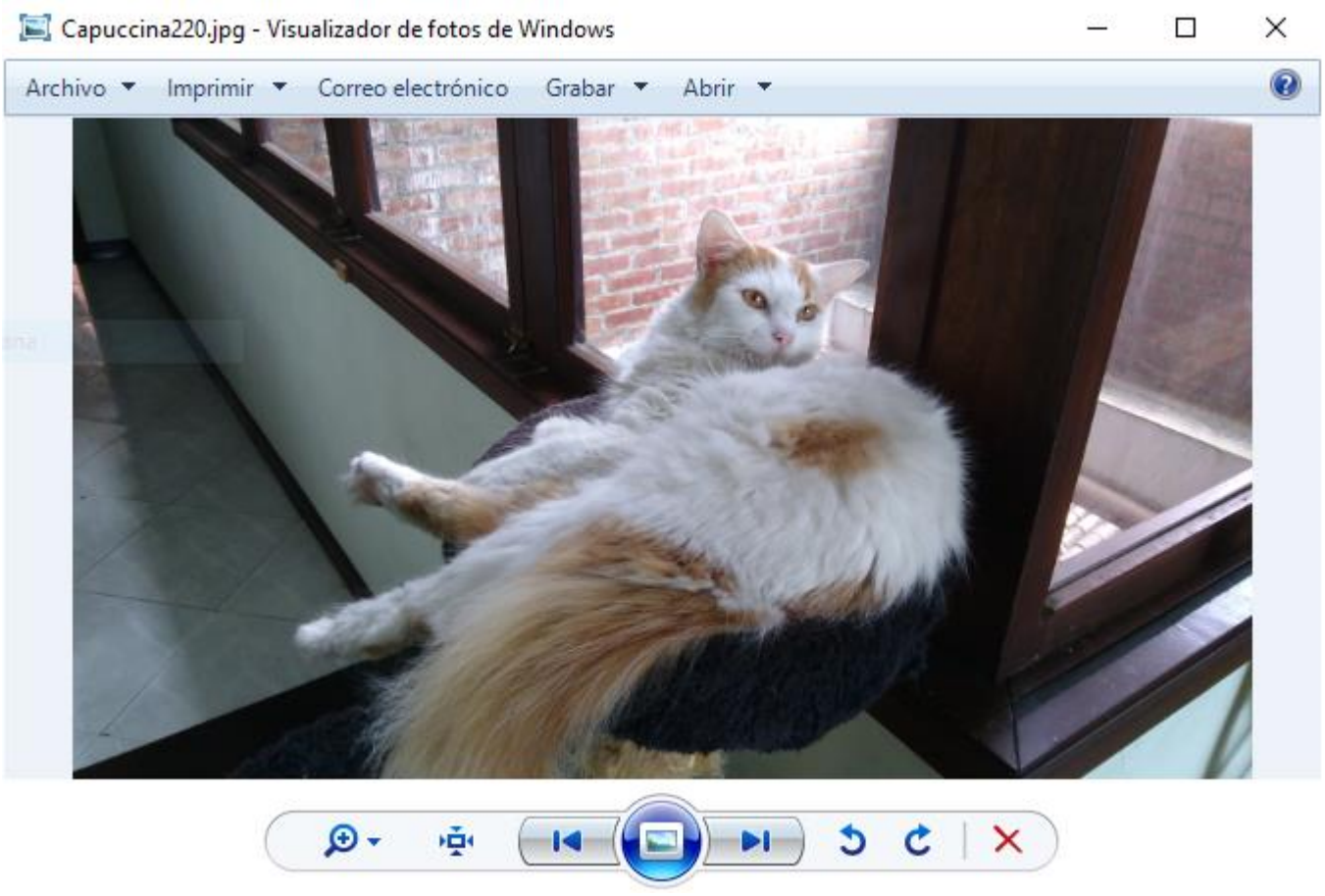


Ilustración 40: Imagen original

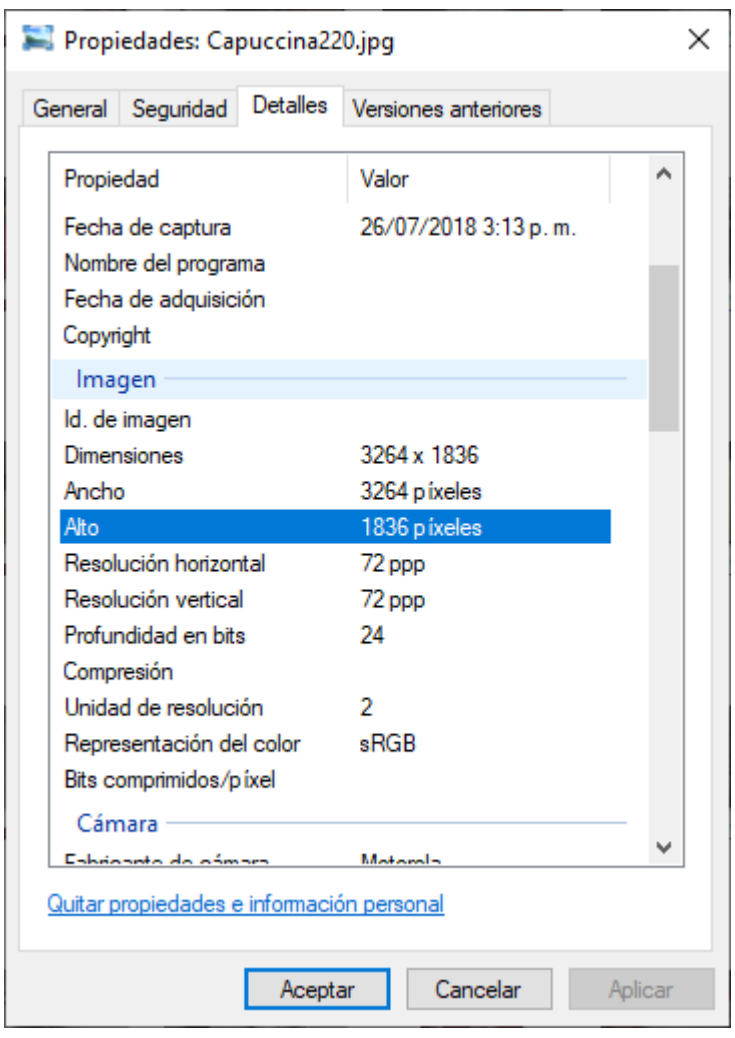


Ilustración 41: Características de esa imagen

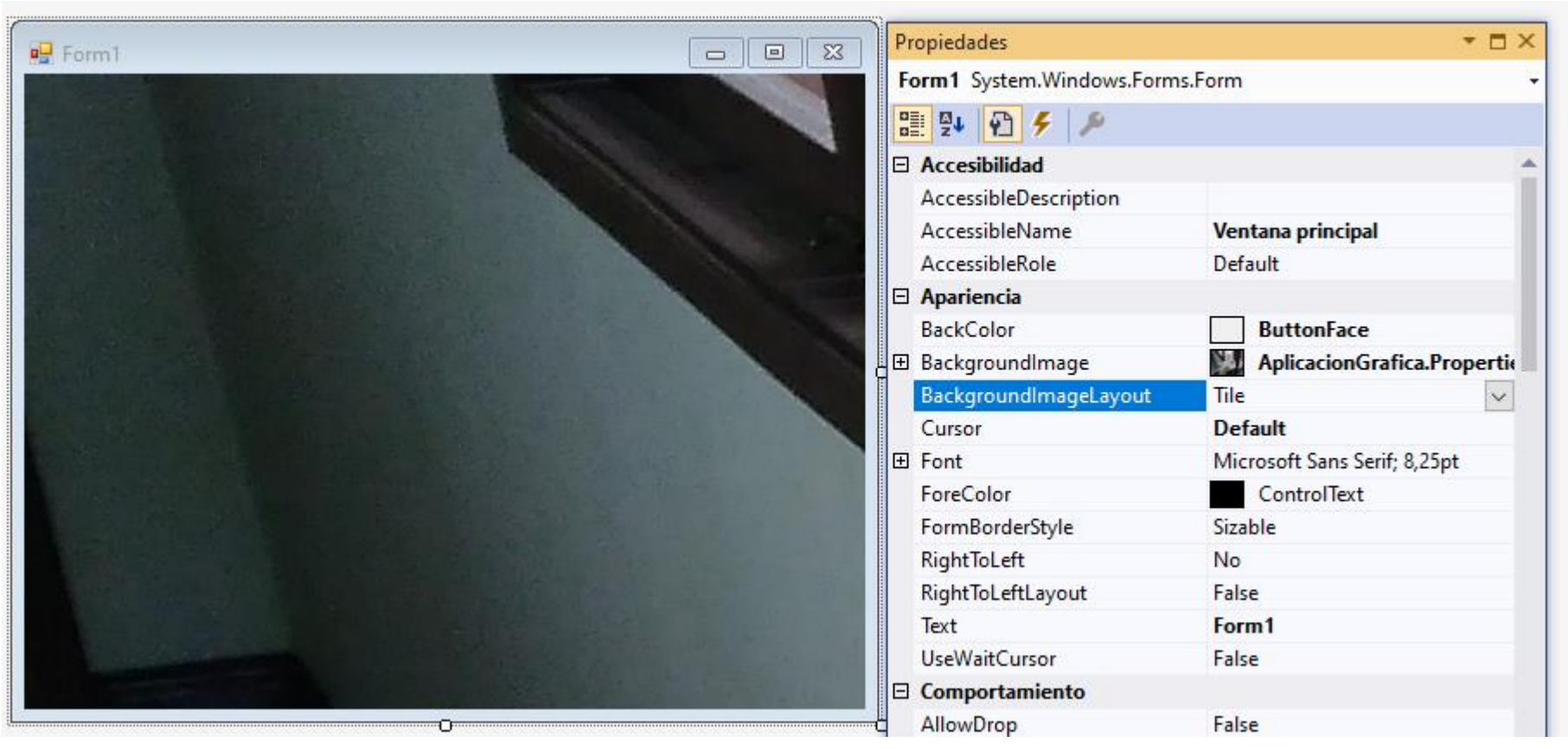


Ilustración 42: Valor "Tile" para BackgroundImageLayout

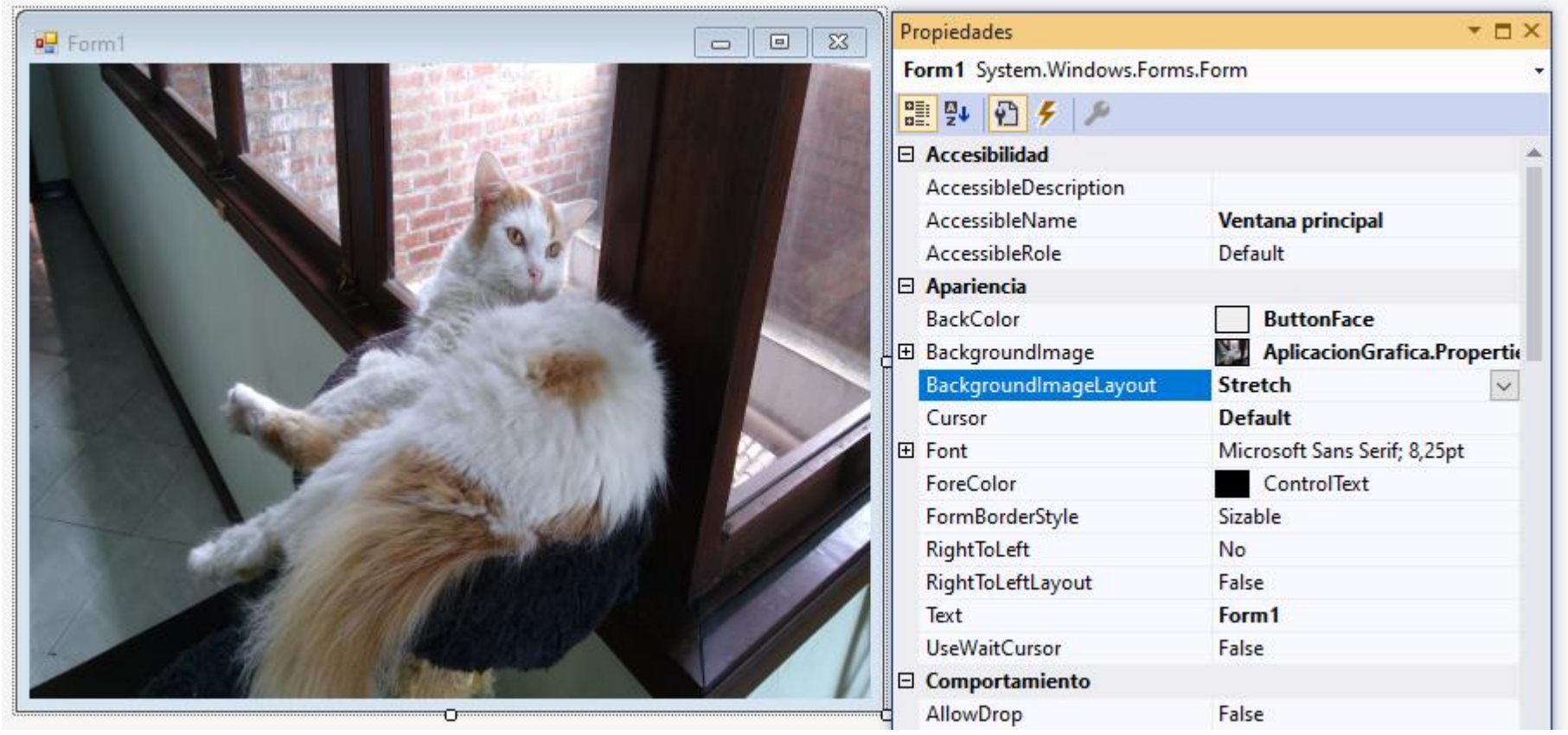


Ilustración 43: Valor "Stretch" para BackgroundImageLayout

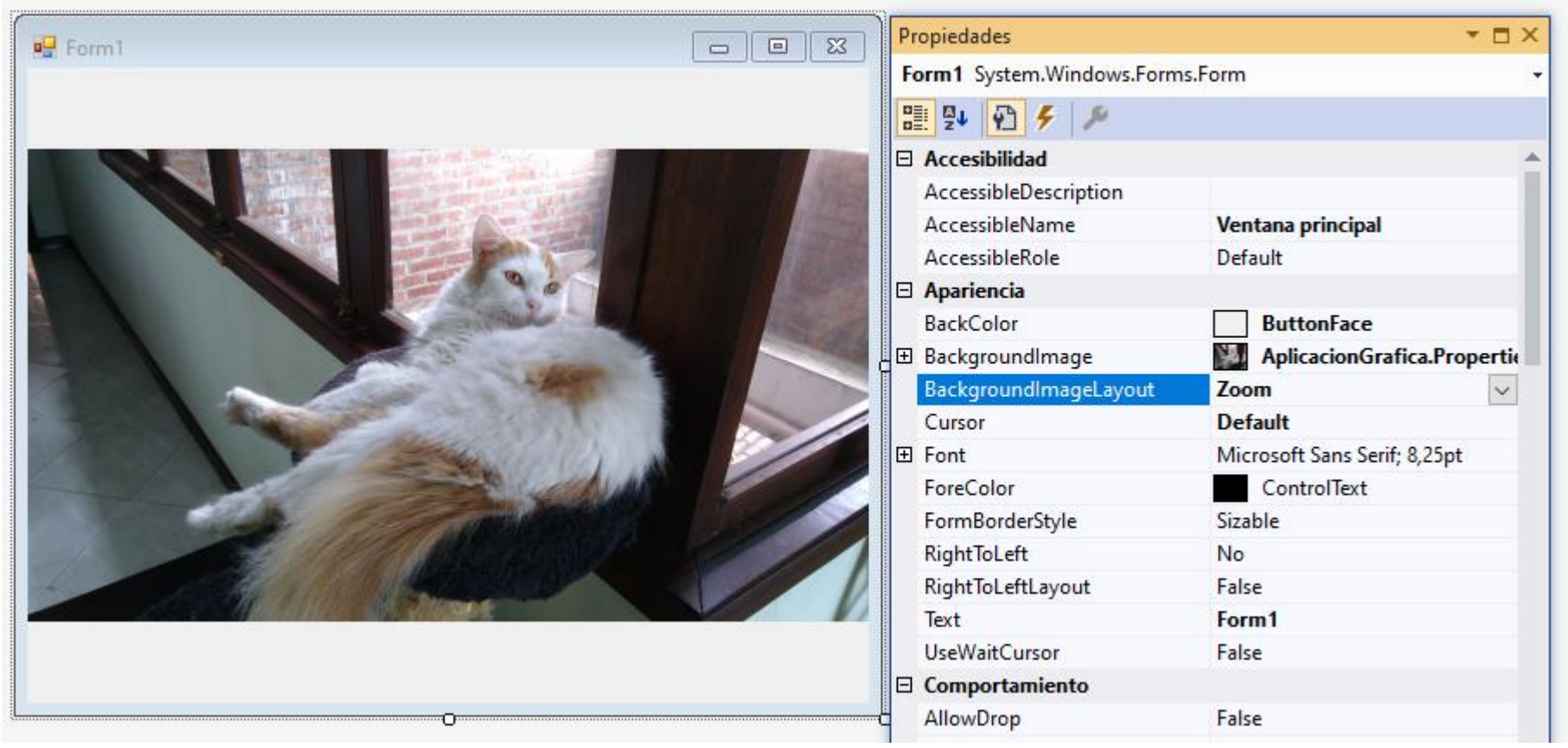


Ilustración 44: Valor "Zoom" para BackgroundImageLayout

Si la imagen de fondo es más pequeña que la ventana, se puede usar el valor “Center” de esa propiedad, que pondría la imagen en todo el centro de la ventana.

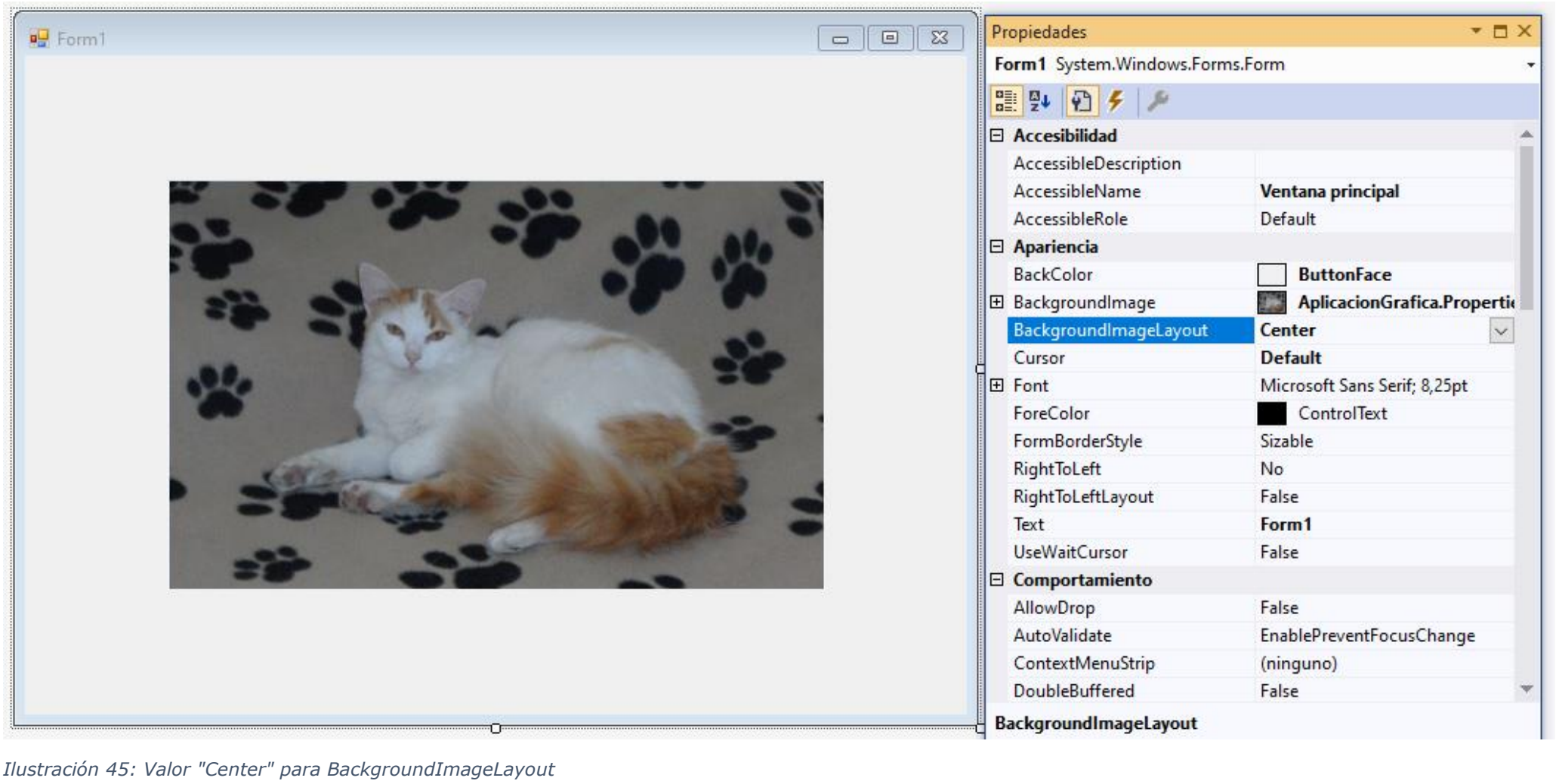


Ilustración 45: Valor "Center" para BackgroundImageLayout

Cursor

Cambia la forma del cursor cuando el puntero del mouse pasa por encima de la ventana

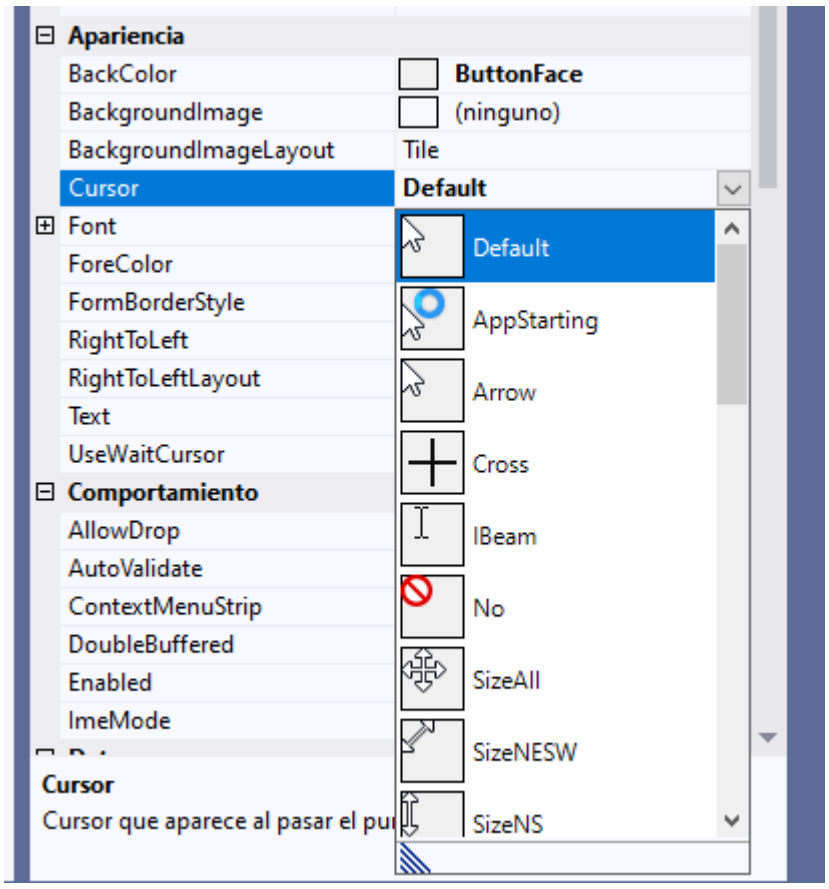


Ilustración 46: Tipos de cursor

Font

Controla el tipo de letra que se mostrará por defecto en todos los controles al interior de la ventana.

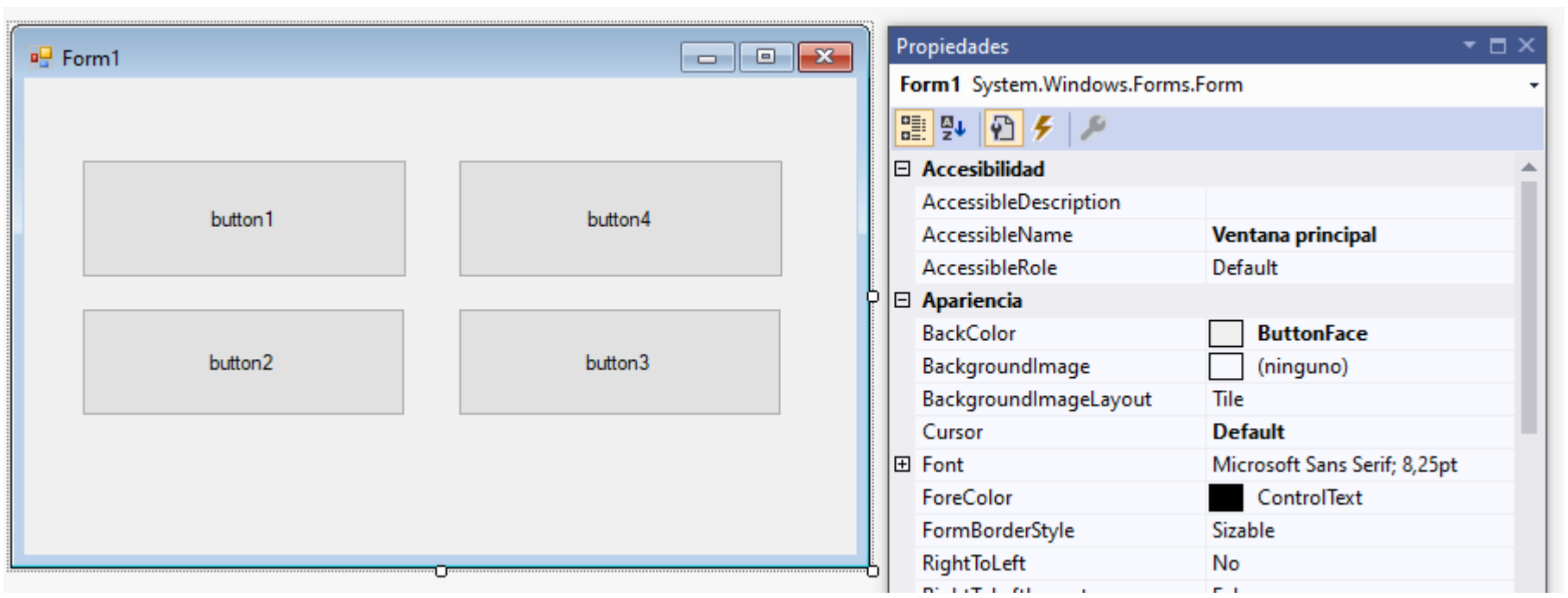


Ilustración 47: Propiedad "Font" afecta al tipo de letra de los botones

Pero cuidado, el cambio de fuente de letra afecta el tamaño de la ventana y los controles gráficos internos.

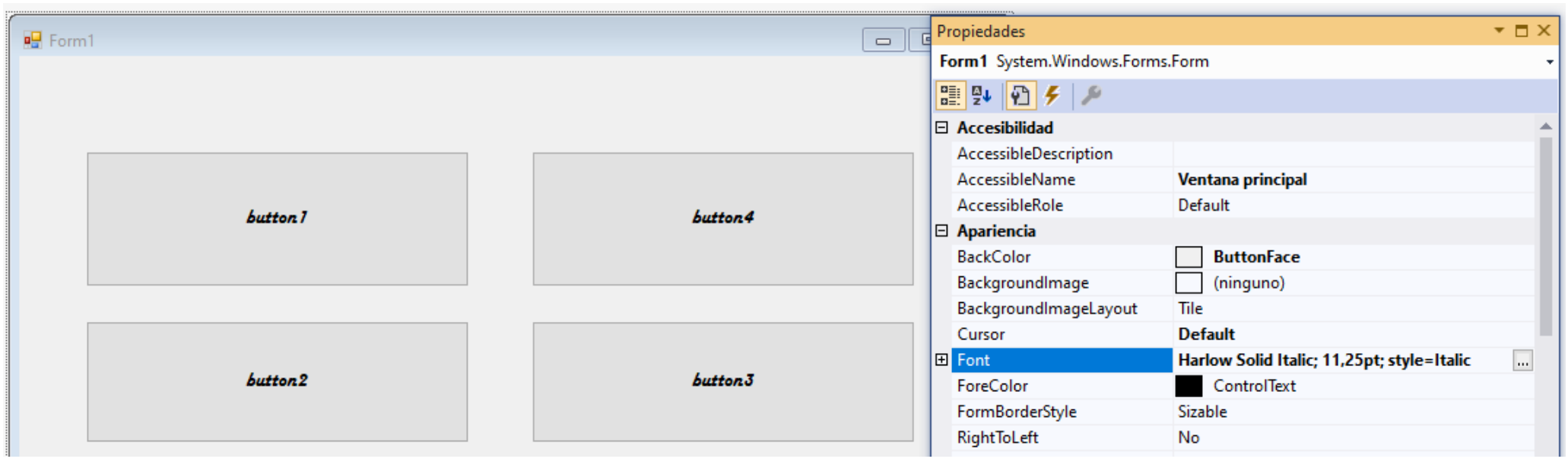


Ilustración 48: El cambio no solo afecta el tipo de letra sino el tamaño de la ventana y los controles gráficos

ForeColor

Ese color afectaría al texto de varios controles gráficos internos

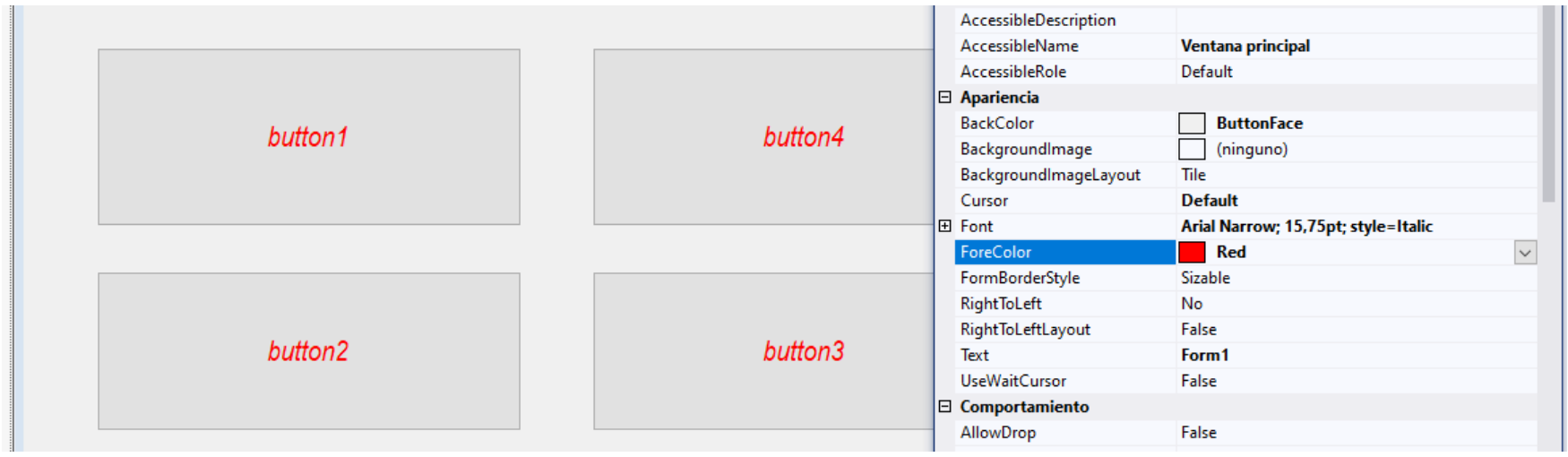


Ilustración 49: ForeColor afecta al color del texto de los controles gráficos internos

FormBorderStyle

Determina como se comportará el borde de la ventana. Tiene los siguientes valores:

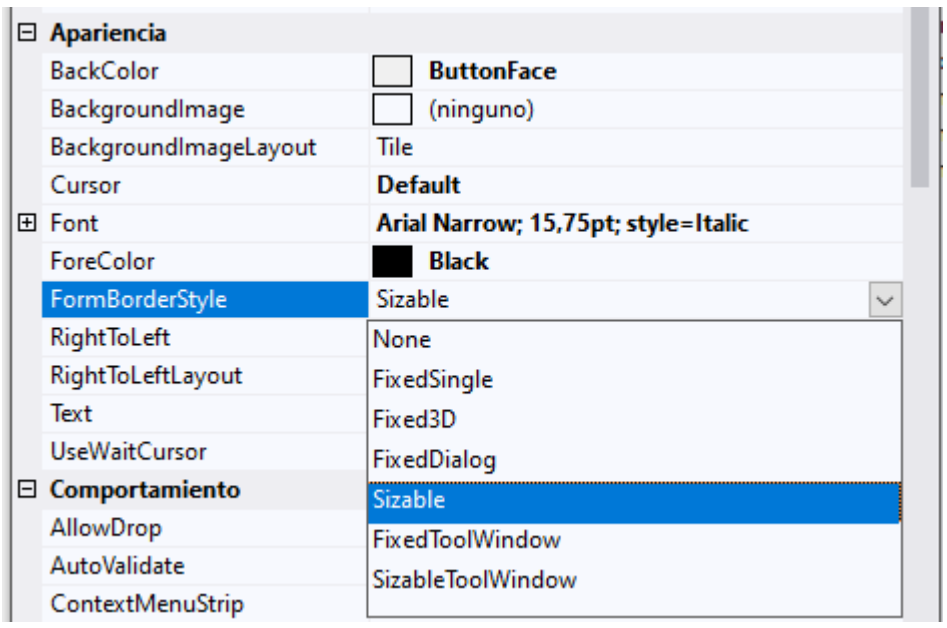


Ilustración 50: Estilo de borde de la ventana

El valor “Sizable” hace que la ventana se puede crecer o decrecer como el usuario lo desee usando el ratón

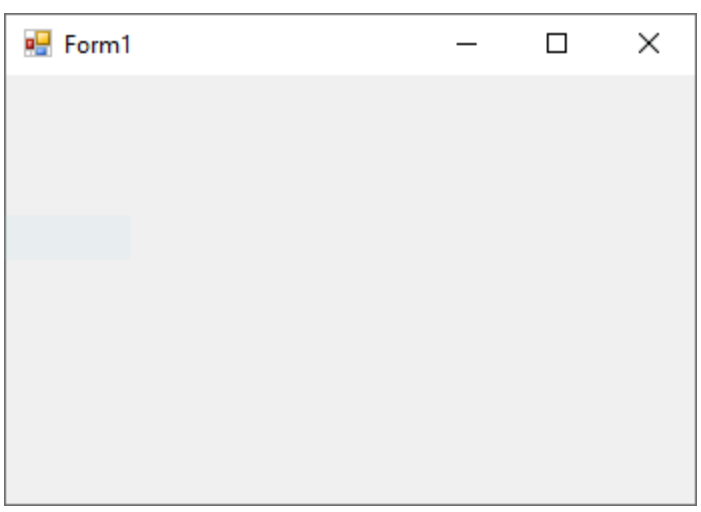


Ilustración 51: Una ventana normal

El valor “None” elimina los bordes y la barra superior de la ventana. Queda un rectángulo fijo que no se puede mover, ni cerrar

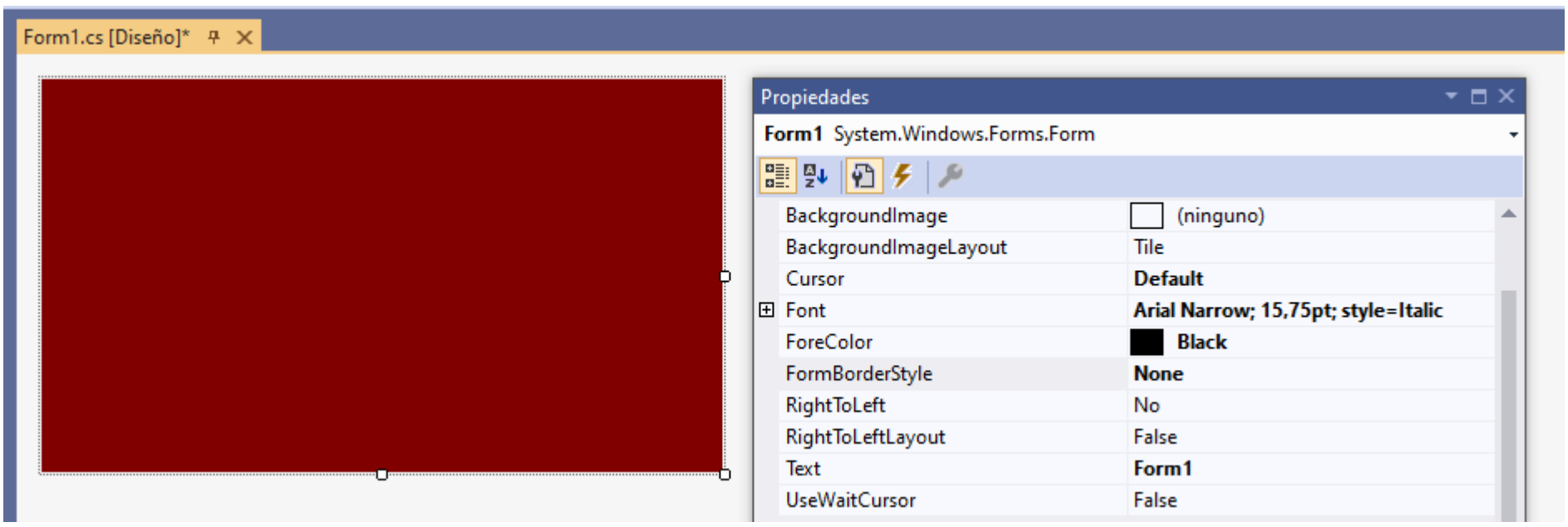


Ilustración 52: Una ventana sin bordes, ni barra superior, ni se puede mover

El valor “FixedSingle” o “FixedDialog” o “Fixed3d” genera una ventana que no se puede redimensionar

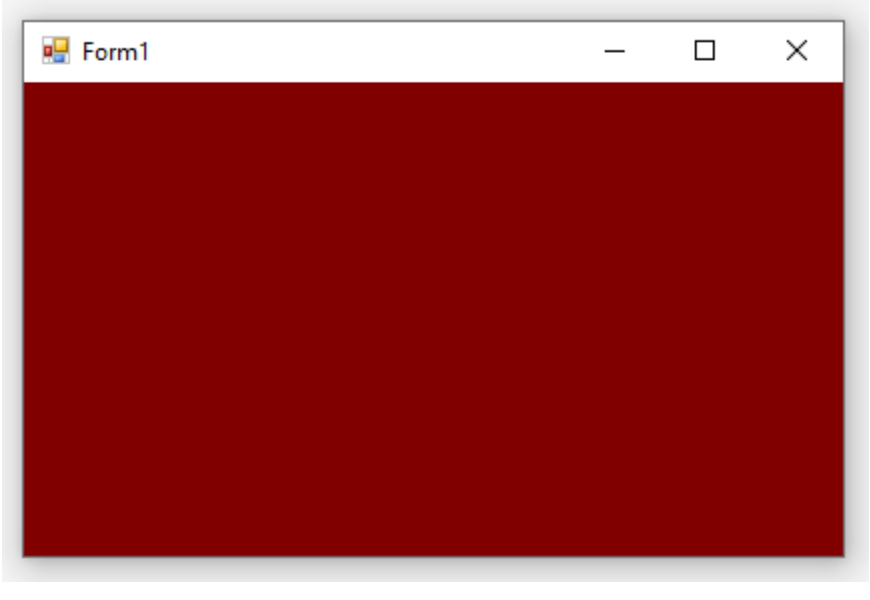
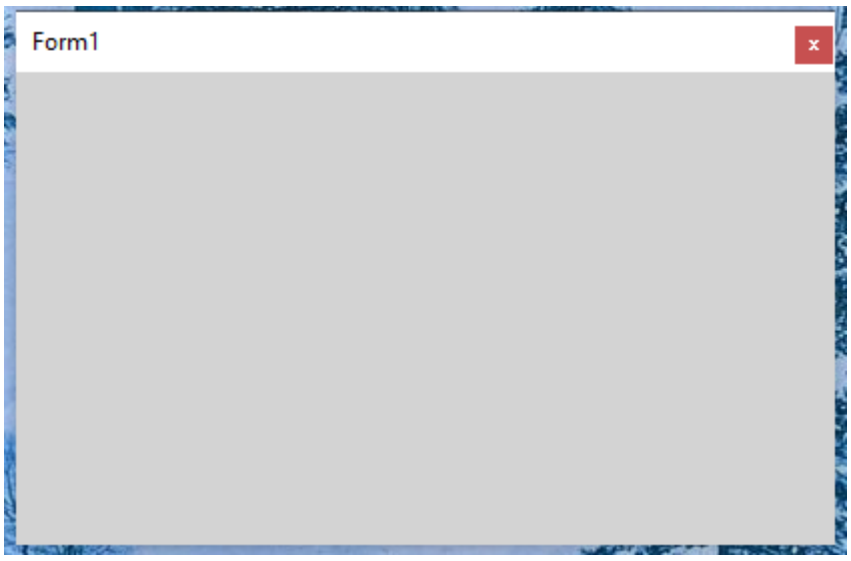


Ilustración 53: "FixedSingle" no permite redimensionar

“FixedToolWindow” retira los botones de minimizar y maximizar, además la ventana no se puede redimensionar. Muy parecida es “SizableToolWindow” pero esta si permite redimensionar la ventana.



RightToLeft y RigthToLeftLayout

Ambas propiedades trabajan juntas, si la primera está en “Yes” y la segunda en “True”, los controles de ventana invierten su posición

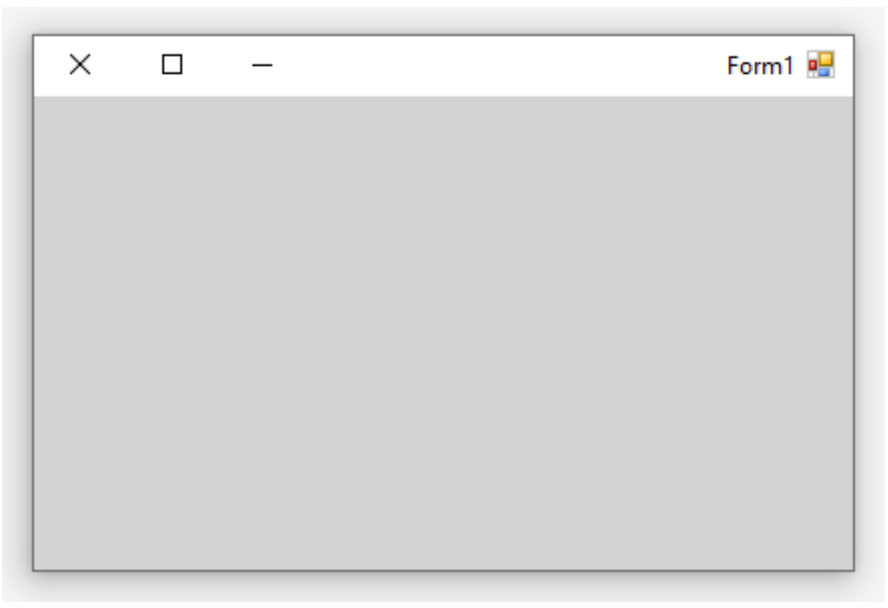


Ilustración 54: Posición invertida de los controles de ventana

Text

Cambia el título de la ventana

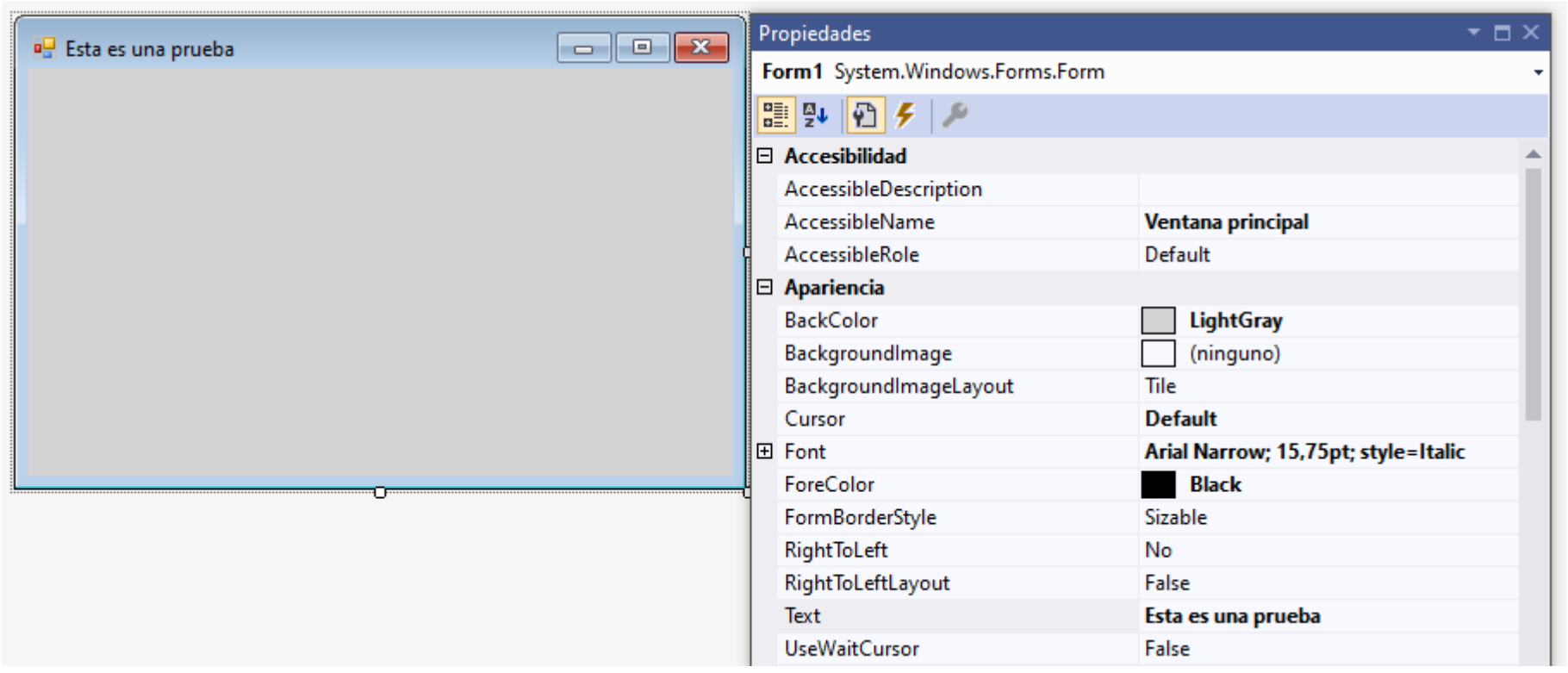


Ilustración 55: Cambiar el título de la ventana

UseWaitCursor

Al poner su valor en “True” hace que el cursor del ratón cambie a la forma de proceso en espera cuando el usuario pone el cursor en el interior de la ventana.

Propiedades de Diseño

(Name)

Una de las propiedades más importantes, porque este nombre único en todo el formulario será usado cuando se escriban las líneas de código

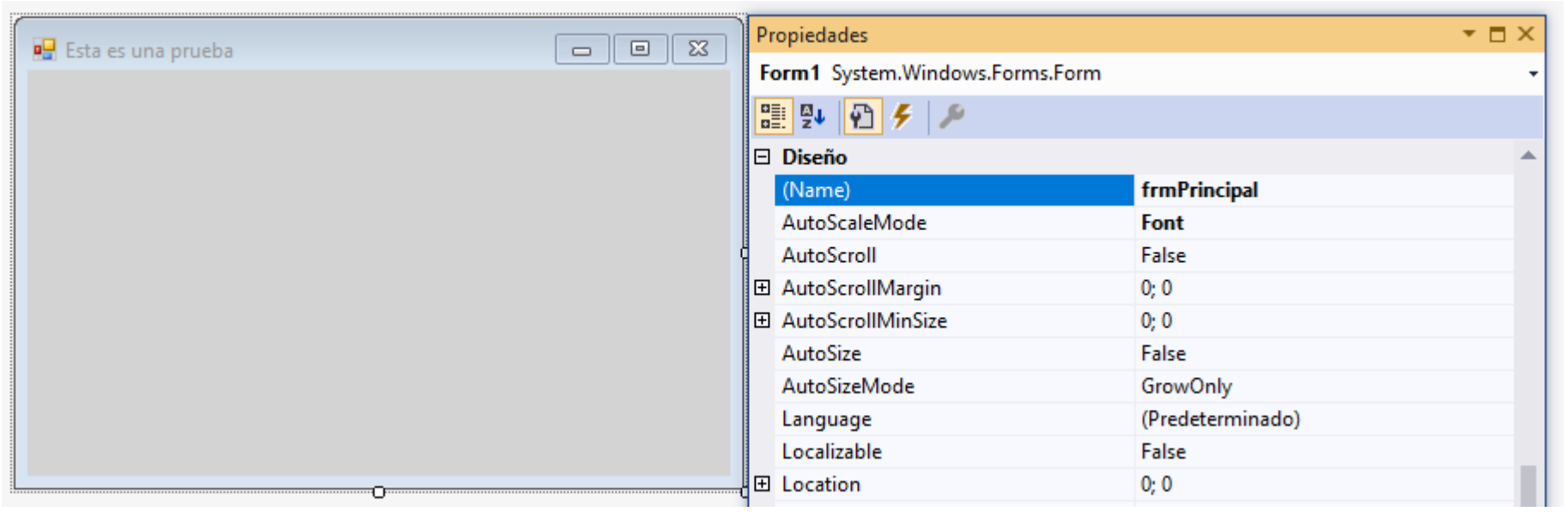


Ilustración 56: (Name) nombre del objeto ventana

Las ventanas o formularios deberían iniciar con las letras "frm" seguido de una letra en mayúscula.

AutoScroll

Si la ventana se reduce de tamaño y deja no visible algunos controles, entonces si es puesto el AutoScroll a True entonces se mostrarán barras de desplazamiento vertical u horizontal.

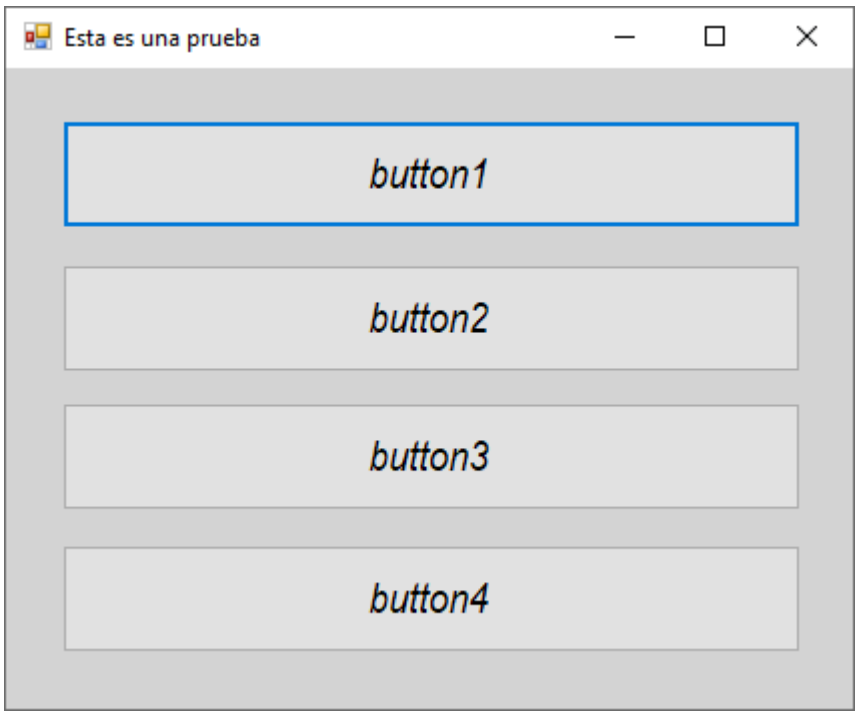


Ilustración 57:Tamaño normal

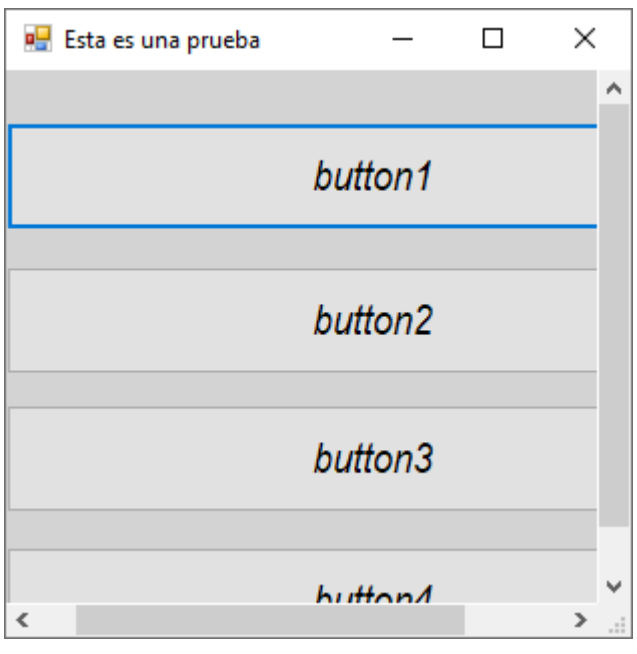
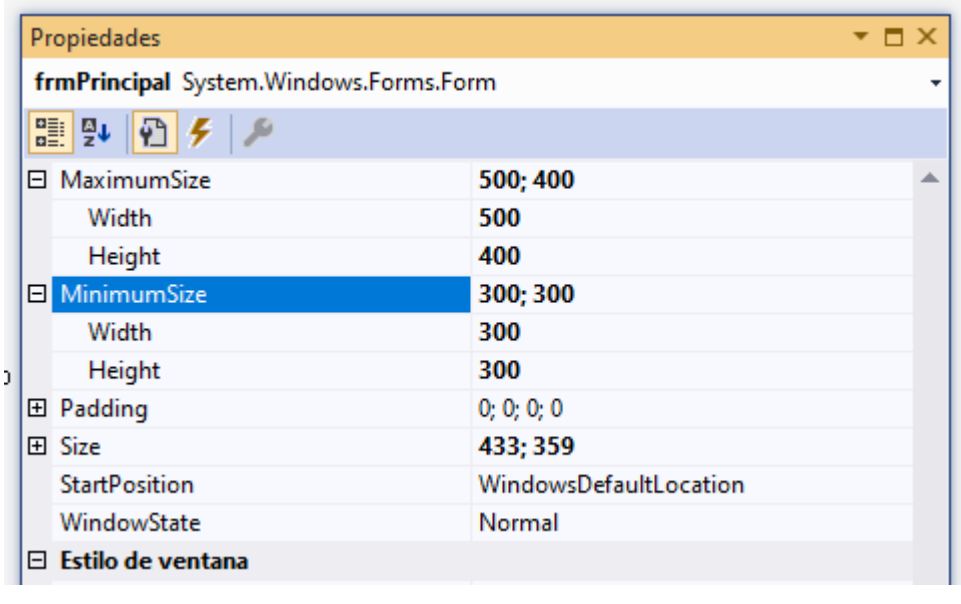


Ilustración 58: Tamaño variado y aparecen barras de desplazamiento

MaximumSize y MinimumSize

Por defecto, ambos están en 0;0 pero al darles un valor de ancho;alto hace que el usuario sólo pueda agrandar la ventana hasta cierto tamaño o empequeñecerla hasta cierto tamaño. Inclusive el botón de maximizar ventana sólo puede escalar la ventana hasta el tamaño máximo configurado.



Size

Tamaño de la ventana originalmente.

StartPosition

Cuando inicie la aplicación ¿Dónde aparecerá? Si es donde la puso el desarrollador, entonces se selecciona el valor “WindowsDeafultLocation”, si lo quiere en el centro de la pantalla, entonces es “CenterScreen”

Ejemplo 1. Uso de los componentes gráficos. Calculando el área de un triángulo dada la medida de sus lados.

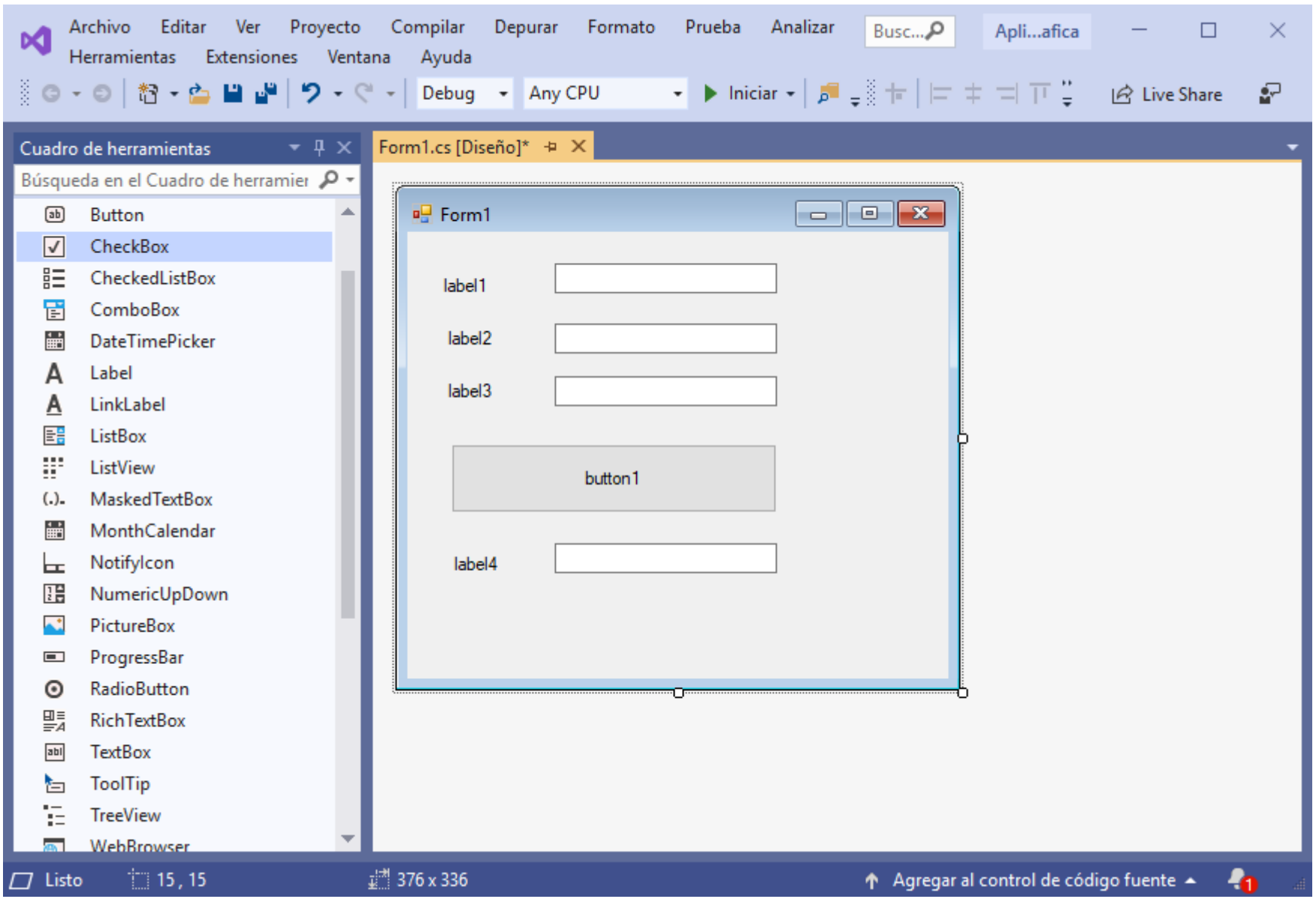


Ilustración 59: Componentes usados "Label", "TextBox", "Button"

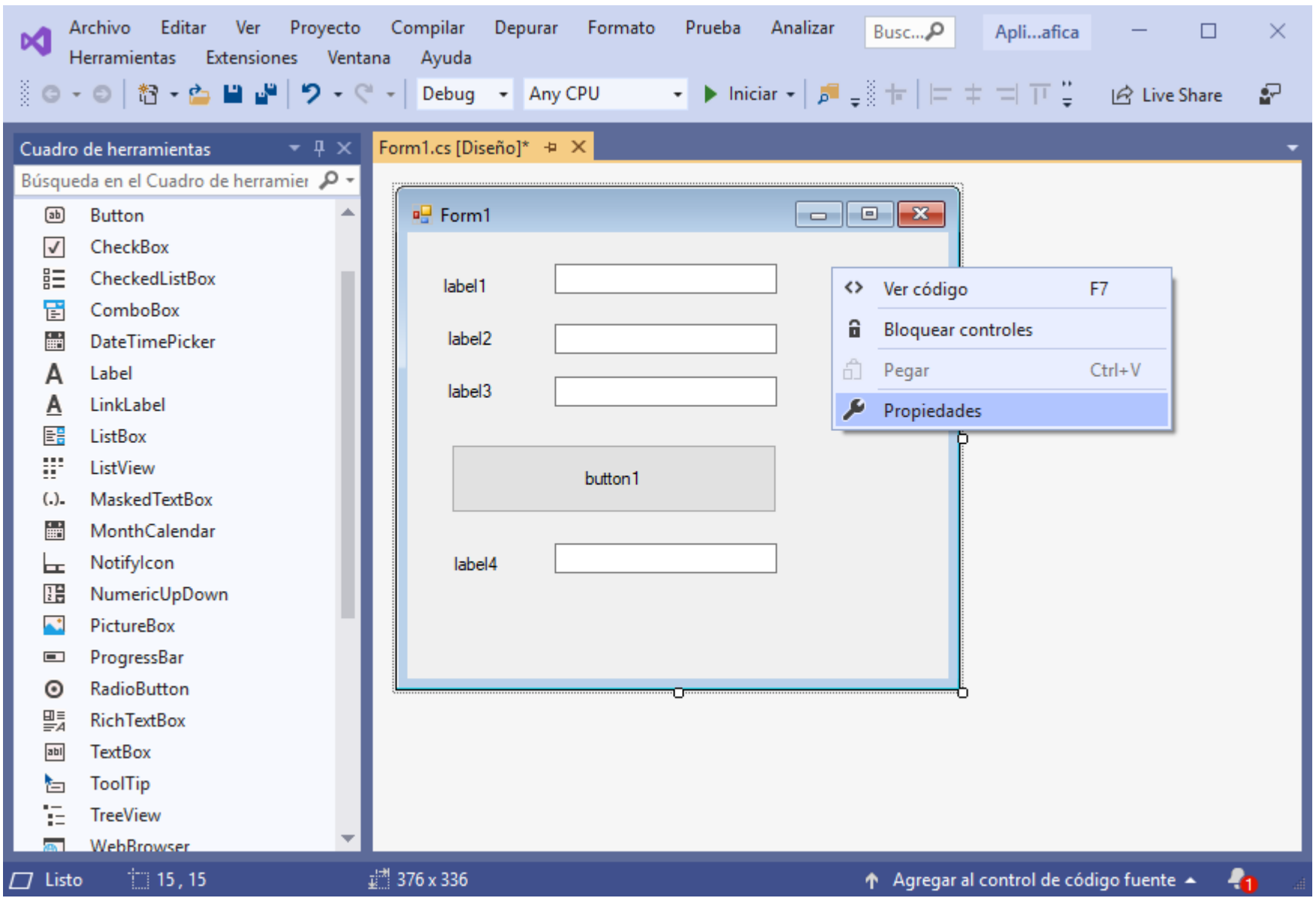


Ilustración 60: De clic botón derecho en la ventana y seleccionamos "Propiedades"

Las propiedades (atributos) se despliegan para **cada** componente gráfico.

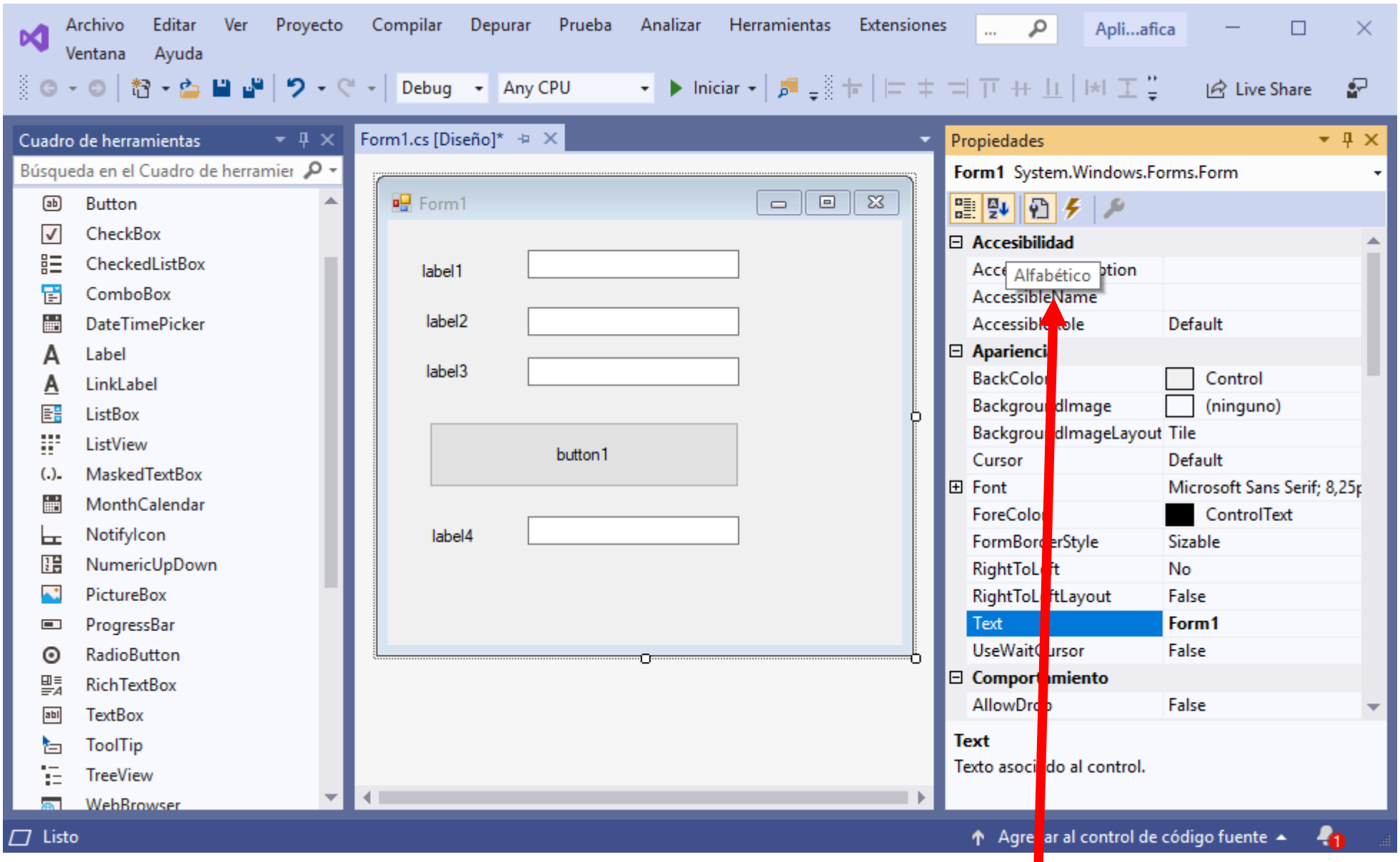


Ilustración 61: Propiedades de cada componente se desplegarán en esta ventana. Seleccione Alfabético para facilitar.

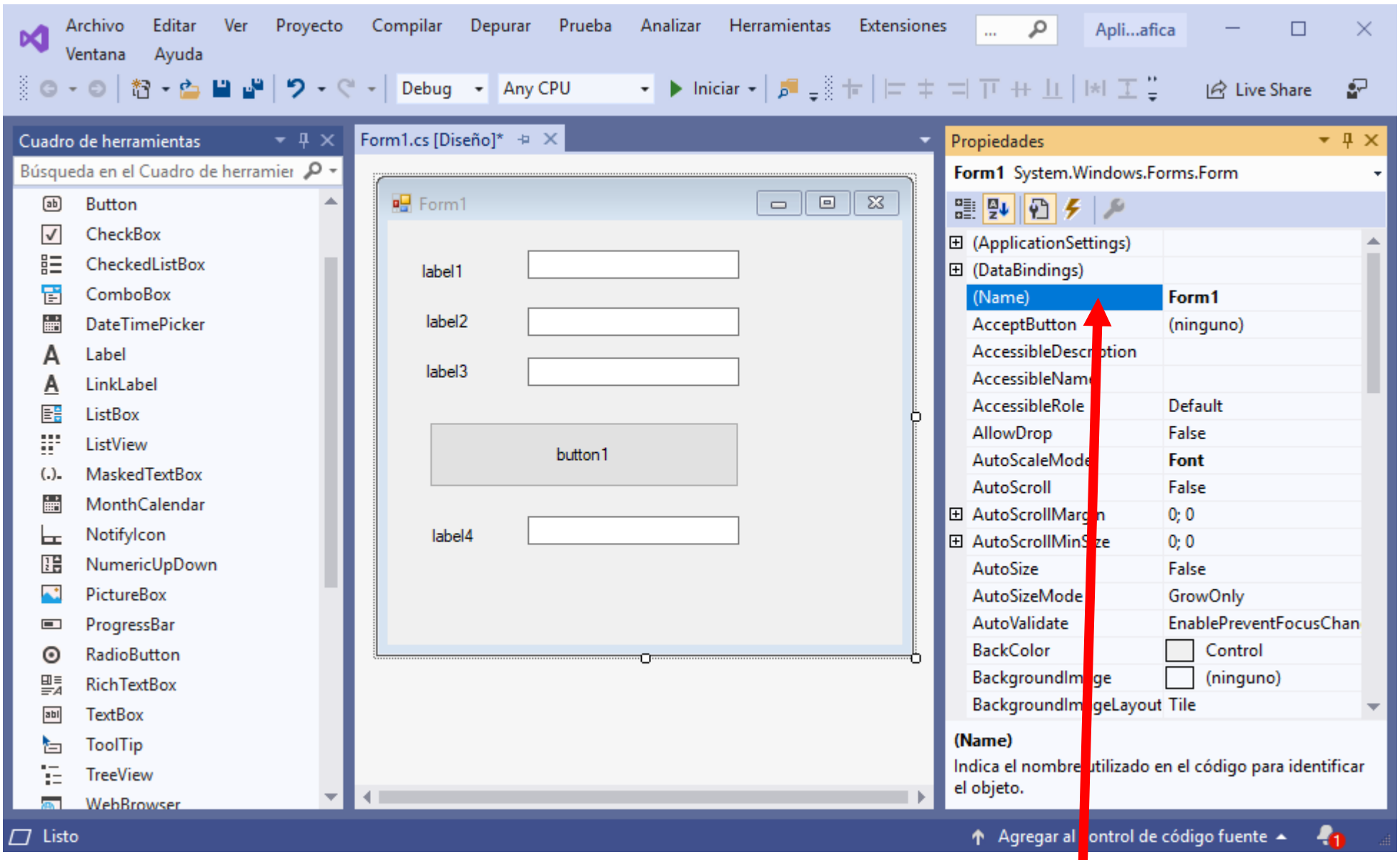


Ilustración 62: Propiedades en orden alfabético de ese componente formulario. La propiedad (Name) es la más importante.

Se deben poner nombres en (Name) a cada componente. Este paso es muy importante y hay hasta una recomendación https://gist.github.com/andyyou/3052671

En el caso del ejemplo sería:

frmAreaTriangulo, para el formulario

lblLadoA, lblLadoB, lblLadoC, lblArea, para las etiquetas (label)

txtLadoA, txtLadoB, txtLadoC, txtArea, para las cajas de texto (textbox)

btnCalcular, para el botón (Button)

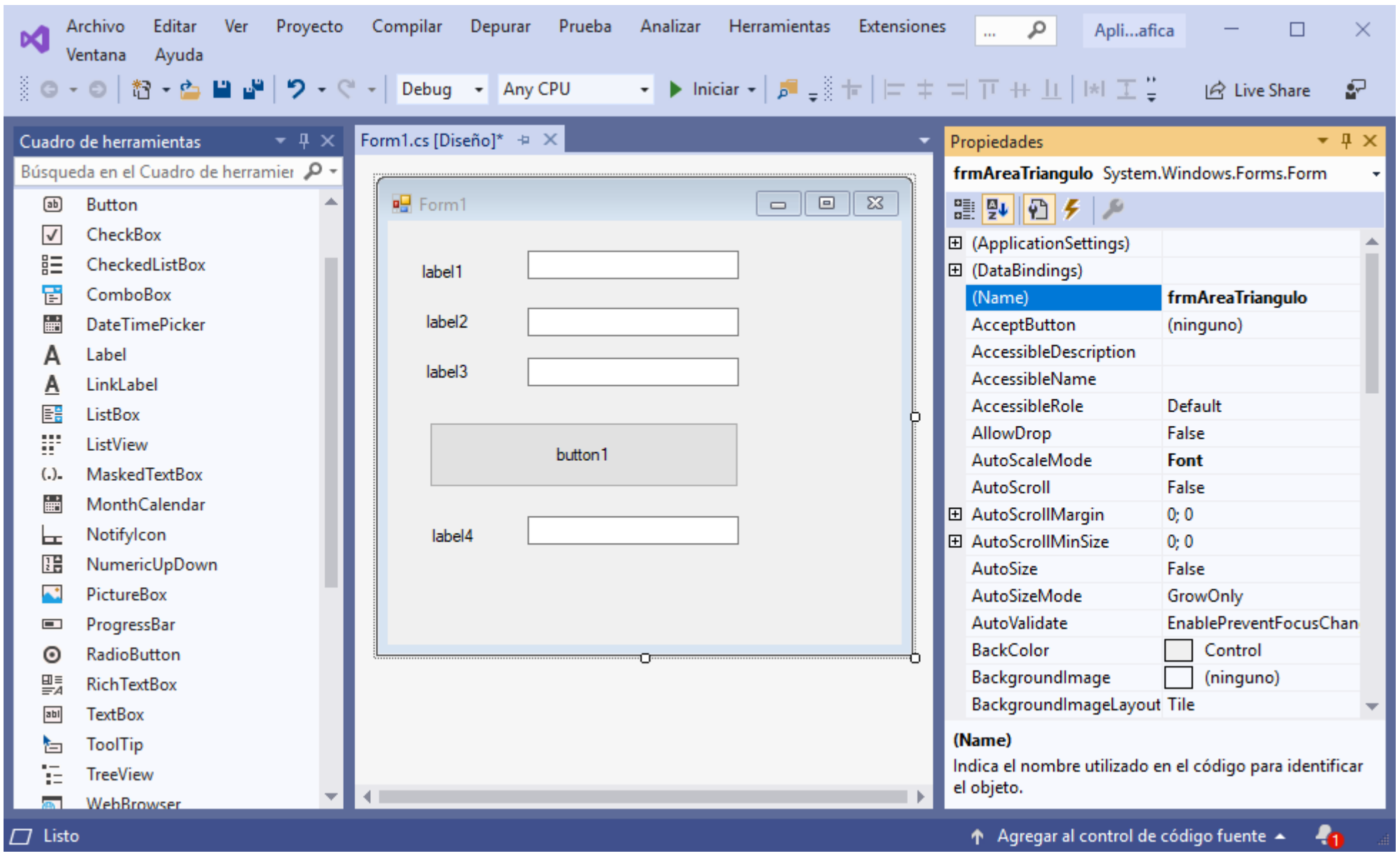


Ilustración 63: Se cambia el (Name) de cada componente usado

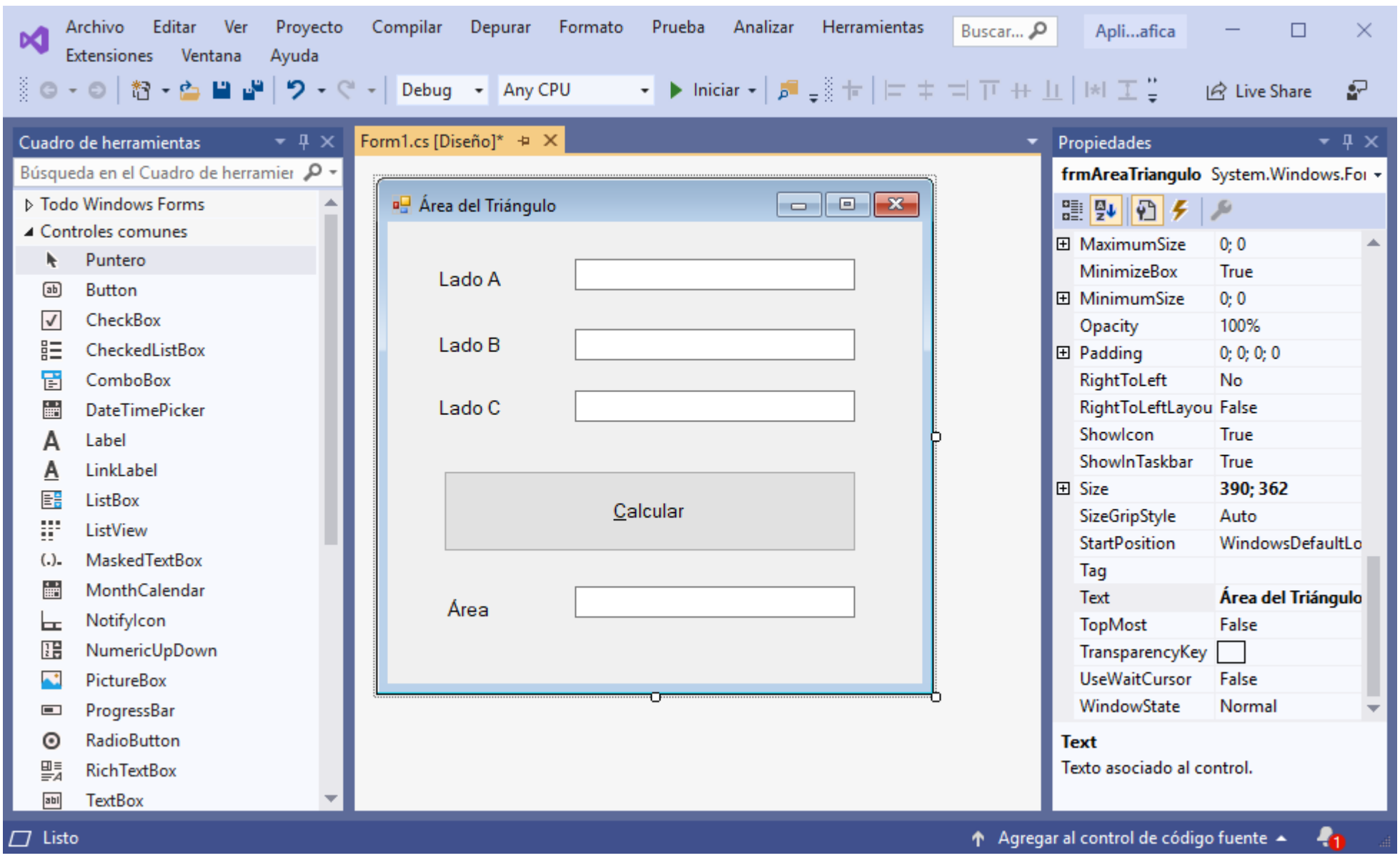


Ilustración 64: Modificando las propiedades como "Text" y "Font" de cada componente. Ahora se da doble clic en el botón

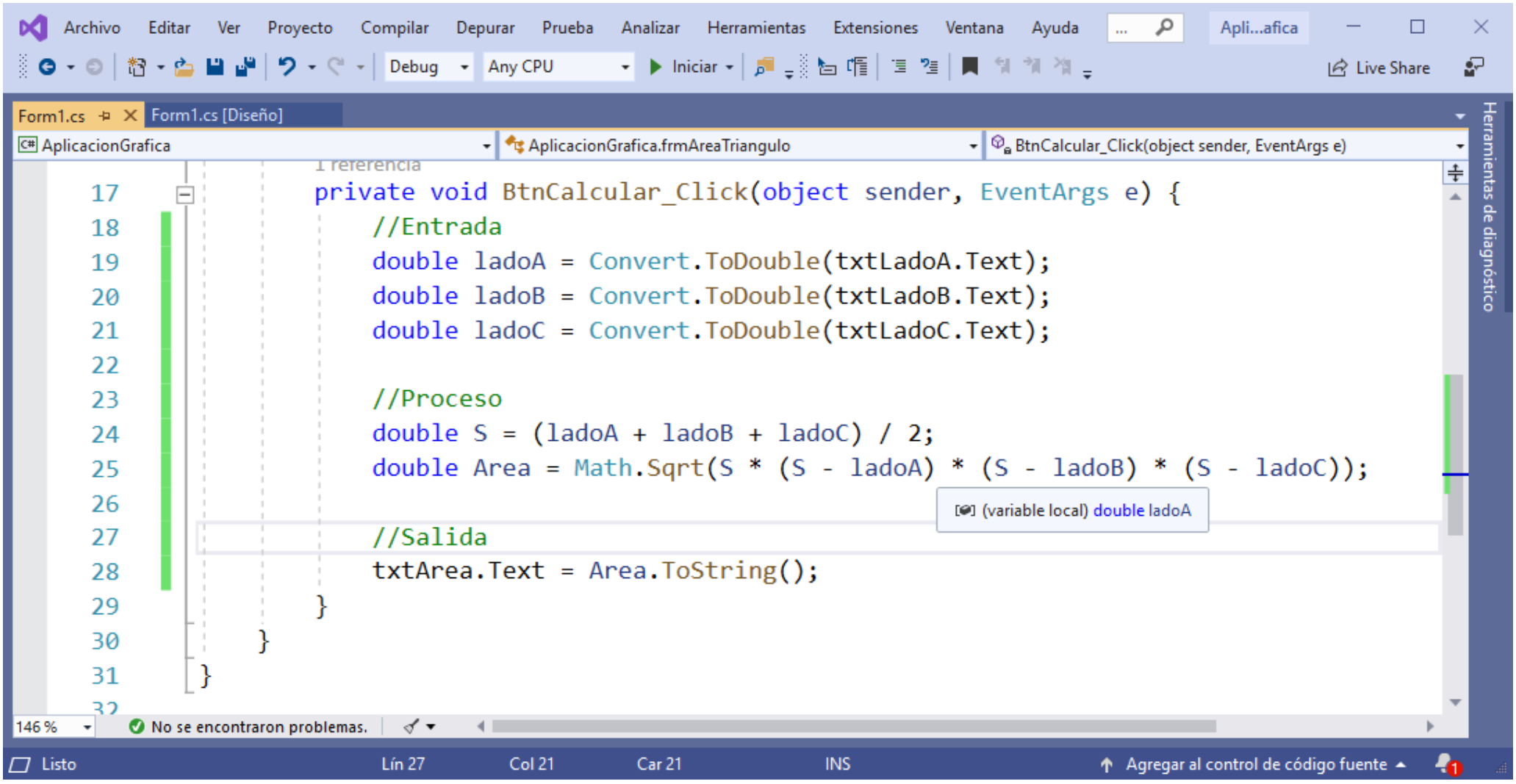


Ilustración 65: Se escribe el código. Atento al uso de las propiedades.

```
            //Entrada
            double ladoA = Convert.ToDouble(txtLadoA.Text);
            double ladoB = Convert.ToDouble(txtLadoB.Text);
            double ladoC = Convert.ToDouble(txtLadoC.Text);

            //Proceso
            double S = (ladoA + ladoB + ladoC) / 2;
            double Area = Math.Sqrt(S * (S - ladoA) * (S - ladoB) * (S - ladoC));

            //Salida
            txtArea.Text = Area.ToString();
```

Ejemplo 2. Uso del checkbox

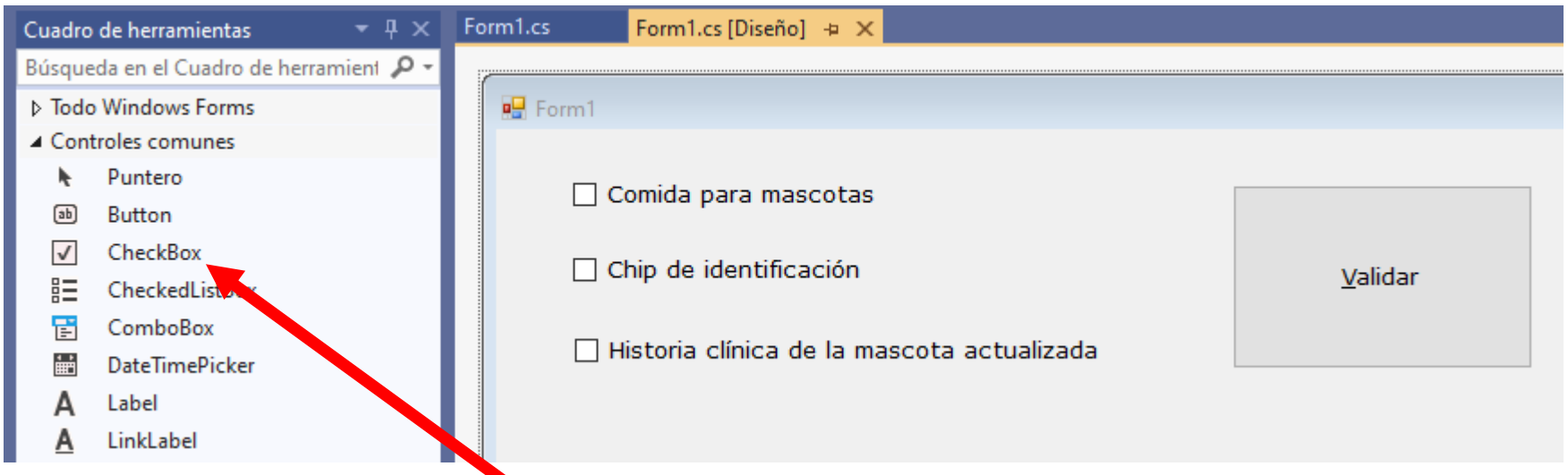


Ilustración 66: Uso del componente checkbox

Se generan tres(3) checkbox y un botón con los siguientes nombres (Name)

chkComida, chkChip, chkHistoria

btnValidar para el botón

Este es el código al presionar el botón

```
        private void btnValidar_Click(object sender, EventArgs e) {
            //Valida si los checkbox están chuleados
            if (chkComida.Checked) {
                MessageBox.Show("Comida para gatos ESTÁ chuleado");
            }

            if (chkChip.Checked) {
                DialogResult resultado = MessageBox.Show("Importante con las mascotas", "¿La mascota tiene chip?",
                                MessageBoxButtons.YesNo,
                                MessageBoxIcon.Question,
                                MessageBoxDefaultButton.Button2);
                if (resultado == DialogResult.Yes) {
                    MessageBox.Show("¡Muy bien!");
                }
                else {
                    MessageBox.Show("¡Un chip puede salvar a su mascota!");
                }
            }

            if (chkHistoria.Checked) {
                MessageBox.Show("La historia clínica está actualizada");
            }
        }
```

Ejemplo 3. Uso del combobox

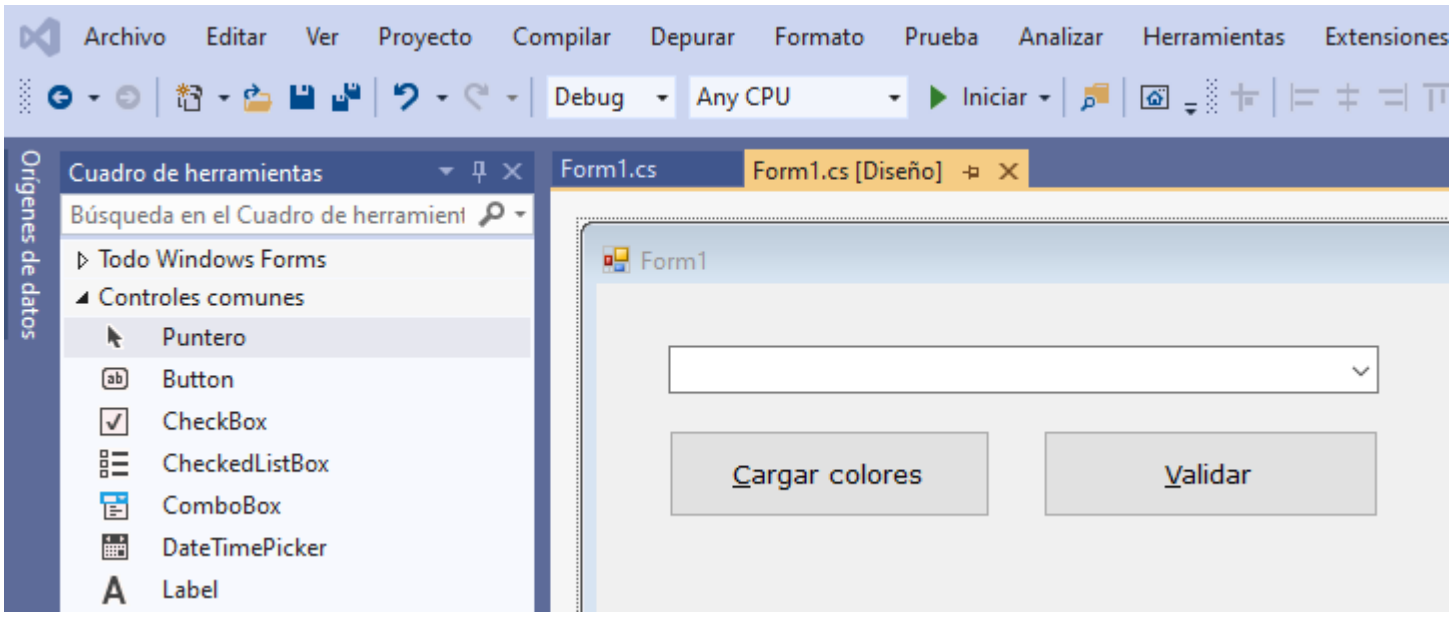


Ilustración 67: Adicionar un combobox y dos botones

Se generan un combobox y dos botones con los siguientes nombres (Name)

cboColores para el combobox

btnCargar, btnValidar para los botones

Este es el código al presionar el botón btnCargar

```
        private void btnCargar_Click(object sender, EventArgs e) {
            cboColores.Items.Clear();
            cboColores.Items.Add("Negro");
            cboColores.Items.Add("Azul");
            cboColores.Items.Add("Marrón");
            cboColores.Items.Add("Gris");
            cboColores.Items.Add("Verde");
            cboColores.Items.Add("Naranja");
            cboColores.Items.Add("Rosa");
            cboColores.Items.Add("Púrpura");
            cboColores.Items.Add("Rojo");
            cboColores.Items.Add("Blanco");
            cboColores.Items.Add("Amarillo");
            cboColores.Items.Add("Turquesa");
            cboColores.Items.Add("Verde Oliva");
            cboColores.Items.Add("Borgoña");
            cboColores.Items.Add("Lavanda");
            cboColores.Items.Add("Magenta");
            cboColores.Items.Add("Salmón");
            cboColores.Items.Add("Beige");
            cboColores.Items.Add("Rosado");
            cboColores.Items.Add("Verde Oscuro");
            cboColores.Items.Add("Aceituna");
            cboColores.Items.Add("Lila");
            cboColores.Items.Add("Amarillo pálido");
            cboColores.Items.Add("Fucsia");
            cboColores.Items.Add("Mostaza");
            cboColores.Items.Add("Ocre");
            cboColores.Items.Add("Trullo");
            cboColores.Items.Add("Malva");
            cboColores.Items.Add("Púrpura oscuro");
            cboColores.Items.Add("Verde lima");
            cboColores.Items.Add("Verde claro");
            cboColores.Items.Add("Ciruela");
            cboColores.Items.Add("Azul claro");
            cboColores.Items.Add("Melocotón");
            cboColores.Items.Add("Violeta");
            cboColores.Items.Add("Tan");
            cboColores.Items.Add("Granate");
            cboColores.Items.Add("Cian");
         }
```

Este es el código al presionar el botón btnValidar

```
        private void btnValidar_Click(object sender, EventArgs e) {
            //Muestra que item del combobox fue seleccionado
            if (cboColores.SelectedItem != null) {
                MessageBox.Show(cboColores.SelectedItem.ToString());
            }
            else {
                MessageBox.Show("No ha seleccionado color");
            }
        }
```

Ejemplo 4. Uso del TreeView

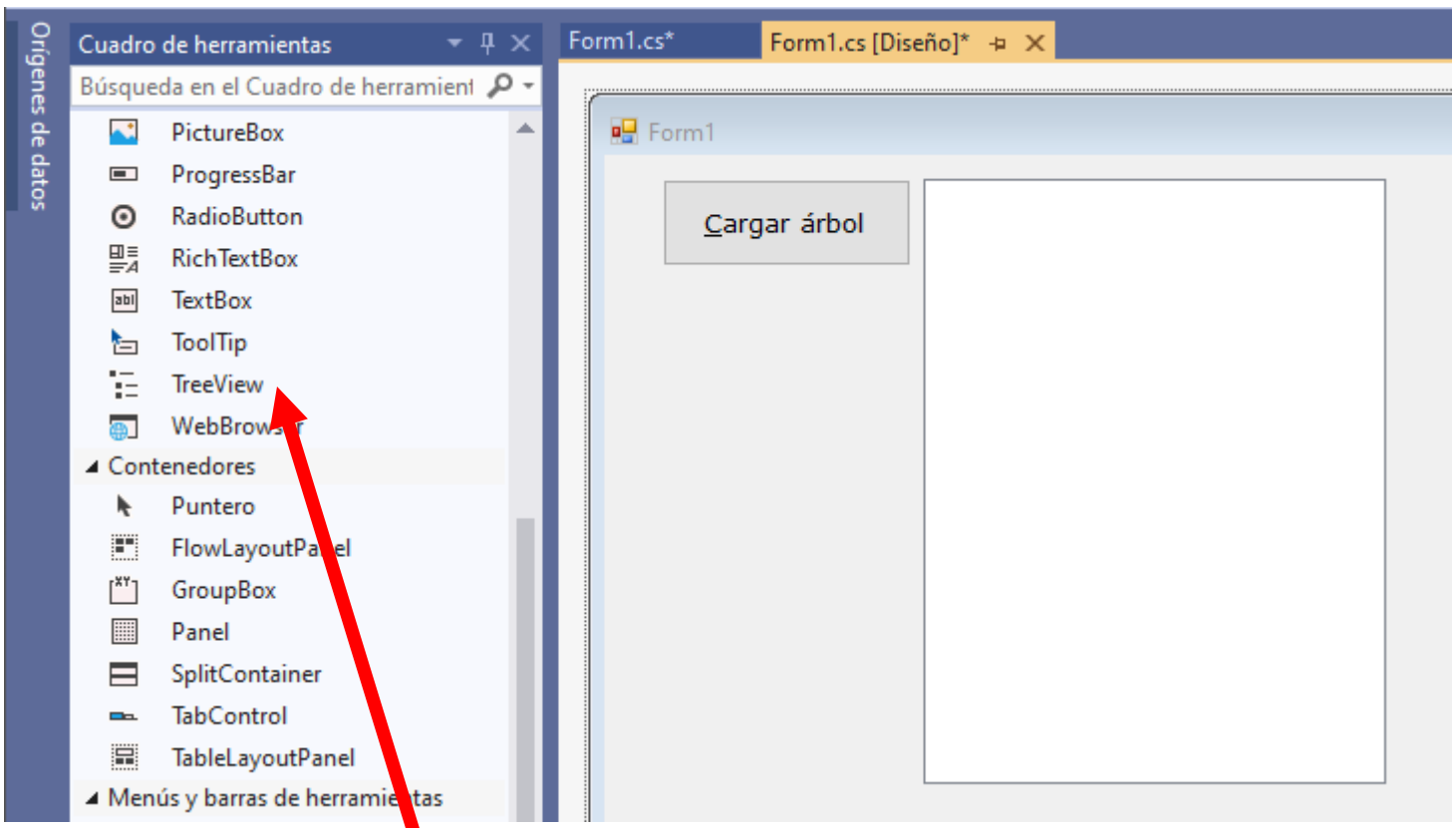


Ilustración 68: Uso del TreeView

Se genera un TreeView y un botón con los siguientes nombres (Name):

treeArbol para el TreeView

btnCargar para el botón

Este es el código del botón

```
        private void btnCargar_Click(object sender, EventArgs e) {
            treeArbol.BeginUpdate();
            treeArbol.Nodes.Add("Raíz");
            treeArbol.Nodes[0].Nodes.Add("Rama 1");
            treeArbol.Nodes[0].Nodes.Add("Rama 2");
            treeArbol.Nodes[0].Nodes[1].Nodes.Add("SubRama");
            treeArbol.Nodes[0].Nodes[1].Nodes[0].Nodes.Add("Sub-SubRama");
            treeArbol.EndUpdate();
        }
```

Este es el código del evento double-click del ratón del componente TreeView

```
        private void treeArbol_NodeMouseDoubleClick(object sender, TreeNodeMouseClickEventArgs e) {
            MessageBox.Show(e.Node.Text);
        }
```

Ejemplo 5. Su propio navegador

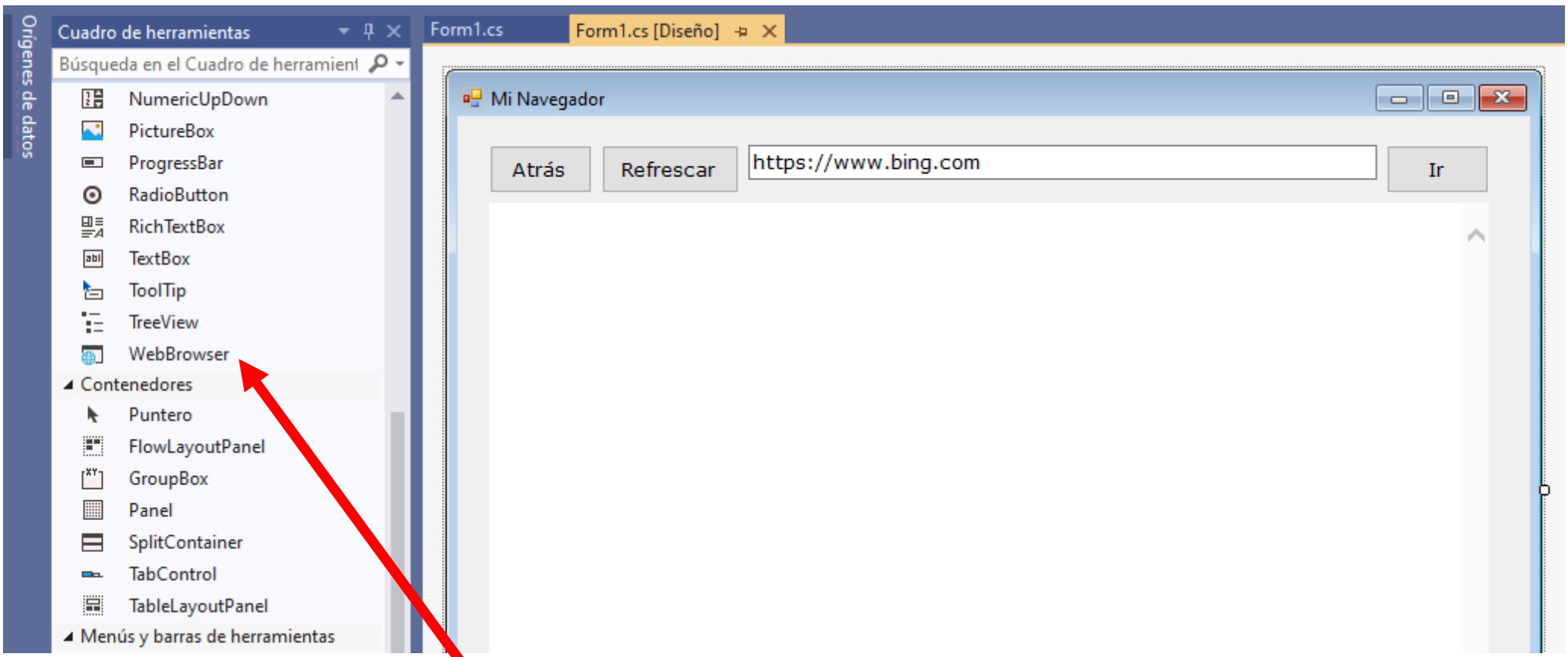


Ilustración 69: Uso del componente WebBrowser

Se generan tres botones, un textbox y un WebBrowser con los siguientes nombres (Name):

btnAtras, btnRefrescar, btnIr para los botones

txtURL para el textbox y en la propiedad Text poner una URL completa como https://darwin.50webs.com

webNavegador para el WebBrowser

Este es el código para los tres botones en sus respectivos eventos

```
        private void btnIr_Click(object sender, EventArgs e) {
            webNavegador.Navigate(txtURL.Text);
        }

        private void btnRefrescar_Click(object sender, EventArgs e) {
            webNavegador.Refresh();
        }

        private void btnAtras_Click(object sender, EventArgs e) {
            webNavegador.GoBack();
        }
```