# Programación Visual con C# .NET

Guion de la práctica 1

## OBJETIVOS

· Iniciarse en el desarrollo de interfaces gráficas
· Ir familiarizándose con el lenguaje C#

## TEMPORIZACIÓN

| | |
|---|---|
| Recogida del enunciado: | Semana del 20 de septiembre |
| Desarrollo de la práctica: | **1 semana** |
| Fecha de entrega: | Semana del 25 de octubre junto con las prácticas 2, 3 y 4 |
| Fecha límite de entrega: | Semana del 1 de noviembre |

# PRÁCTICA 1
# Ventana principal, gráficos y diálogos

## TABLA DE CONTENIDOS:

# Introducción

Se realizará una aplicación con el siguiente aspecto:


En el punto 1 se creará una aplicación que muestre la hora del sistema en una ventana. En la misma ventana habrá un botón que el usuario tendrá que pulsar cuando quiera que se actualice la hora mostrada.

En el punto 2 se añadirá a la aplicación otra ventana que muestre un reloj analógico. De esta forma la hora, además de aparecer en la caja de texto (reloj digital), se dibujará en forma de reloj de agujas. Esto requerirá ciertos cálculos trigonométricos sencillos. Además, se mejorará el programa para que actualice la hora cada segundo, de forma que el reloj parecerá estar en marcha. Finalmente, se añadirá la funcionalidad necesaria para que el usuario pueda cambiar la hora pulsando con el ratón sobre el reloj analógico.

Nota: si está utilizando el entorno de desarrollo Visual C# Express, asegúrese de que tienen activada la configuración para expertos: menú *Herramientas* > *Configuración* > *Configuración para expertos*.

# 1. Crear la aplicación

## *1.1 Crear un nuevo proyecto*

Ejecute Visual Studio .NET. Abra un proyecto tipo *Visual C# - Aplicación para Windows* (vea anexo, al final del enunciado, para la creación y configuración del proyecto con Visual Studio 2022). Asigne un nombre al proyecto, y guárdelo en la carpeta que desee. Una vez hecho esto, se mostrará una ventana llamada "Form1"; se trata de un objeto de la clase `Form1` derivada de `System.Windows.Forms.Form`.

Asigne al formulario el nombre (**Name**) "RelojDigital" (obsérvese que este nombre será también el nombre de la clase) y el título (**Text**) "Reloj digital".

**Compilar y ejecutar para ver el resultado**.

## *1.2 Creación de un menú*

Asigne al formulario `RelojDigital` una barra de menús. Ésta constará de dos menús, "Archivo" y "Ayuda", y éstos a su vez de las órdenes "Salir" y "Acerca de", respectivamente. Para ello, en la vista de diseño, elija del cuadro de herramientas el control **MenuStrip**, y añádalo a la ventana debajo de la barra de título. Una vez hecho esto, añada los menús arriba comentados usando el editor integrado.

Por último, implemente la orden *Salir*. Para ello, añada un manejador que responda al evento **Click** de este elemento y complételo para que al ejecutarse invoque al método `Close` del propio formulario.

**Compilar y ejecutar para ver el resultado**.

## *1.3 Pequeños ajustes*

Personalice la apariencia de esta ventana principal. Por ejemplo, evite que el usuario pueda modificar o maximizar el tamaño de la ventana, estableciendo las propiedades adecuadas del formulario. También puede cambiar el icono de la aplicación a uno que considere conveniente.

**Compilar y ejecutar para ver el resultado**.

## *1.4 Creación del diálogo "Acerca de"*

Habrá comprobado que al ejecutar el programa y hacer clic sobre la orden "Acerca de" del menú, ésta no hace nada. Esto es lógico ya que no se ha implementado el manejador para responder al evento **Click** de ese elemento. Este manejador deberá mostrar la ventana "Acerca de". Por lo tanto, añada un nuevo formulario (objeto de la clase `Form`) denominado `AcercaDe` y siga, para su diseño, los pasos indicados a continuación.

Por cuestión de apariencia, deberá ajustar el tamaño de este formulario a uno que considere adecuado. Además, deberá modificar las propiedades necesarias para que el formulario no se pueda maximizar, minimizar o redimensionar, y para que no aparezca en la barra de tareas del sistema operativo.

Añada al formulario `AcercaDe` un botón `bt_Aceptar`, que permitirá al usuario cerrar el formulario. Modifique su propiedad **DialogResult** a `OK`, y su propiedad **Text** para que el botón muestre "Aceptar". A continuación, asigne el nombre de dicho botón a la propiedad **AcceptButton** del formulario `AcercaDe`.

Ahora sólo falta añadir el método que maneje el evento **Click** de la orden "Acerca de". Este método tiene que crear un objeto de la clase `AcercaDe` y mostrarlo. Esto se realiza de la siguiente manera:

```
AcercaDe dlg = new AcercaDe();
dlg.ShowDialog();
```

Añada una etiqueta de texto (control de la clase **Label**) para almacenar en la misma los créditos de la aplicación. Este texto será cargado desde un recurso de tipo "cadena de caracteres" (clase *Resources*, archivo *Resources.resx*) denominado **textoAcercaDe**. Opcionalmente puede añadir una imagen (control **PictureBox**). Por ejemplo:


Nota: este diálogo es *modal*, es decir, es necesario cerrarlo para poder seguir interactuando con la aplicación

**Compilar y ejecutar para ver el resultado**.

## *1.5 Una caja de edición y una etiqueta*

Edite el formulario principal `RelojDigital`. Añada una caja de texto, `ct_HoraActual`, que muestre el texto centrado y sea de sólo lectura. Después, añada encima una etiqueta con el texto "Hora actual:".

**Compilar y ejecutar para ver el resultado**.

## *1.6 Dar la hora (al principio)*

Añada en el constructor de `RelojDigital` el siguiente código, después de la llamada a `InitializeComponent`:

```
DateTime hora = DateTime.Now;
ct_HoraActual.Text = hora.ToLongTimeString();
```

**Compilar y ejecutar para ver el resultado**.

## *1.7 Dar la hora (cuando lo pida el usuario). Un botón*

Utilizando la herramienta de refactorización *Extraer método* de Visual C#, convierta las dos últimas sentencias que añadió, en una llamada a un método privado `MostrarHoraActual`. Edite el formulario `RelojDigital`. Debajo de la caja de texto añada un botón `bt_Actualizar` con texto "Actualizar".

Añada un manejador `bt_Actualizar_Click` para responder al evento **Click** del botón y llame desde su interior a `MostrarHoraActual`.

**Compilar y ejecutar para ver el resultado**.

## *1.8 Desfase horario*

Añada a `RelojDigital` el siguiente campo privado:

```
private TimeSpan m_DesfaseHorario = new TimeSpan(0);
```

Cambie la primera línea de `MostrarHoraActual` del formulario principal por:

```
DateTime hora = DateTime.Now + m_DesfaseHorario;
```

**Compilar y ejecutar para ver el resultado**.

# 2. Reloj analógico. Una caja de diálogo

## *2.1 Crear la caja de diálogo no modal*

Añada a la aplicación un nuevo formulario de nombre "RelojAnalógico", y con título "Reloj analógico". Modifique las propiedades necesarias para que la barra de título no muestre ningún botón, y para que el diálogo no aparezca en la barra de tareas del sistema operativo. Añada a la clase `RelojDigital` un campo `m_RelojAnalógico` de tipo `RelojAnalógico`, y asígnele un objeto de su clase en el mismo sitio de su declaración. Hágalo visible como se muestra a continuación, al final del constructor de `RelojDigital`:

```
m_RelojAnalógico.Show(this);
```

Nota: este diálogo es *no modal*, es decir, no es necesario cerrarlo para poder seguir interactuando con la aplicación.

**Compilar y ejecutar para ver el resultado**.

## *2.2 Situar el diálogo no modal en una posición determinada*

Añada en la clase `RelojDigital` el manejador para el evento **Shown** del formulario y edite el manejador así:

```
private void RelojDigital_Shown(object sender, EventArgs e)
{
  m_RelojAnalogico.Location = new Point(
      this.Location.X + ancho reloj digital + 10,
      coordenada Y del reloj digital);
}
```

**Compilar y ejecutar para ver el resultado.**

## *2.3 Evitar que el diálogo no modal se cierre con Alt+F4*

El atajo de teclado *Alt+F4* cierra un formulario. Para evitar que pueda cerrar el reloj analógico cancelaremos su evento *FormClosing* que se genera cuando se cierra el formulario. Añada en la clase `RelojAnalógico` el manejador para el evento **FormClosing** del formulario y edite el manejador así:

```
private void RelojAnalógico_FormClosing(object sender,
                                        FormClosingEventArgs e)
{
   if (si el diálogo RelojAnalógico está enfocado)
     cancelar evento;
}
```

**Compilar y ejecutar para ver el resultado.**

## *2.4 Datos del reloj analógico: centro, tamaño y hora*

Añada los siguientes campos privados a la clase `RelojAnalógico`:

```
Point m_Centro = new Point();
int m_Radio;

DateTime m_Hora;
```

Añada también la siguiente propiedad pública:

```
public DateTime Hora
{
   set
   {
      m_Hora = value;
      Invalidate();
   }
}
```

Inicie `m_Hora` con la hora actual, y el resto de los campos de la forma indicada a continuación, en el constructor de `RelojAnalógico`:

```
m_Centro.X = this.ClientSize.Width / 2;
m_Centro.Y = this.ClientSize.Height / 2;
m_Radio = Math.Min(m_Centro.X,m_Centro.Y);
```

Añada a la clase `RelojAnalógico` un manejador para el evento **Resize**. Utilizando la herramienta de refactorización, convierta las líneas del constructor de `RelojAnalógico` que inicializan los campos `m_Centro.X/Y` y `m_Radio` en una llamada a un método privado `ActualizarDimensiones`. Al final del manejador que acaba de añadir, incluya una llamada a `ActualizarDimensiones`.

Añada la siguiente línea al final del método `MostarHoraActual` de la clase `RelojDigital` que permitirá (una vez que se hayan pintado las manecillas del reloj analógico) actualizar la hora del reloj analógico desde el reloj digital, cada vez que se muestre la hora actual en el reloj digital.

```
m_RelojAnalógico.Hora = hora;
```

**Compilar y ejecutar para ver el resultado**.

## *2.5 Pintar las manecillas. Líneas*

Añada un manejador para el evento **Paint** del formulario `RelojAnalógico`. A continuación, complételo así:

```
private void RelojAnalógico_Paint(object sender, PaintEventArgs e)
{
   if (m_Radio <= 10)
      return;

   Graphics gfx = e.Graphics;
```

```
        Pen lápizMuyGordoAzul = new Pen(Color.Blue, 4);
        float alfa, x, y;

        // Manecilla de las horas
        alfa = (float) ((m_Hora.Hour % 12) * Math.PI * 2 / 12);
        alfa += (float)((m_Hora.Minute) * Math.PI * 2 / 12 / 60);

        x = (float) Math.Sin(alfa) * m_Radio;
        y = ((float)Math.Cos(alfa) * m_Radio;

        gfx.DrawLine(lápizMuyGordoAzul, m_Centro.X, m_Centro.Y,
                         m_Centro.X + (int)(x * .5),
                         m_Centro.Y + (int)(y * .5));

        // Manecilla de los minutos
        alfa = (float)((m_Hora.Minute % 60) * Math.PI * 2 / 60);
        alfa += (float)((m_Hora.Second) * Math.PI * 2 / 60 / 60);
        x = (float)(Math.Sin(alfa) * m_Radio);
        y = (float)(Math.Cos(alfa) * m_Radio);

        gfx.DrawLine(lápizMuyGordoAzul, m_Centro.X, m_Centro.Y,
                         m_Centro.X + (int)(x * .75),
                         m_Centro.Y + (int)(y * .75));


        // Manecilla de los segundos
        alfa = (float)((m_Hora.Second % 60) * Math.PI * 2 / 60);
        x = (float)(Math.Sin(alfa) * m_Radio);
        y = (float)(Math.Cos(alfa) * m_Radio);

        gfx.DrawLine(lápizMuyGordoAzul, m_Centro.X, m_Centro.Y,
                         m_Centro.X + (int)(x * .9),
                         m_Centro.Y + (int)(y * .9));

    }
```

**Compilar y ejecutar para ver el resultado**.

## *2.6 Pintar la esfera con sus marcas. Pinceles, brochas y elipses*

Modifique en `RelojAnalógico` el método privado `RelojAnalógico_Paint` como se indica a continuación. Se dibujan las marcas del reloj y su esfera, además del punto central mediante el uso de pinceles y brochas, estableciéndose el origen de coordenadas en el centro; la coordenada Y aumenta hacia arriba.

```
    private void RelojAnalógico_Paint(object sender, PaintEventArgs e)
    {
        if (m_Radio <= 10)
            return;

        Graphics gfx = e.Graphics;

        Pen lápizNormal = new Pen(Color.Black, 1);
        Pen lápizGordoNegro = new Pen(Color.Black, 2);
        Pen lápizGordoRojo = new Pen(Color.Red, 2);
        Pen lápizMuyGordoAzul = new Pen(Color.Blue, 4);
        HatchBrush brochaGris =
            new HatchBrush(HatchStyle.Sphere, Color.Gray, Color.Gray);
        float alfa, x, y;
```

```
        // Llevamos el origen de coordenadas al centro de la ventana
        // y hacemos que el eje "y" aumente hacia arriba
        Matrix matriz = new Matrix(1, 0, 0, -1,
                                   m_Centro.X, m_Centro.Y);
        gfx.Transform = matriz;

        //Esfera
        float radioEsfera = m_Radio * .95f;
        gfx.DrawEllipse(lápizNormal, -radioEsfera, -radioEsfera,
                                    radioEsfera * 2, radioEsfera * 2);

        // Marcas de los minutos
        for (int i = 0; i < 60; i++)
        {
           alfa = (float)(i * Math.PI * 2 / 60);
           x = (float)(Math.Sin(alfa) * m_Radio);
           y = (float)(Math.Cos(alfa) * m_Radio);

           if (i % 5 == 0)
              gfx.DrawLine(lápizGordoNegro, x * .85f,y * .85f,
                                            x * .95f,y * .95f);
           else
              gfx.DrawLine(lápizNormal,x *.9f,y *.9f,
                                       x * .95f,y * .95f);
        }

        // Manecilla de las horas
        alfa = (float)((m_Hora.Hour % 12) * Math.PI * 2 / 12);

        alfa += (float)((m_Hora.Minute) * Math.PI * 2 / 12 / 60);
        x = (float)(Math.Sin(alfa) * m_Radio);
        y = (float)(Math.Cos(alfa) * m_Radio);

        gfx.DrawLine(lápizMuyGordoAzul, 0, 0, x * .5f, y * .5f);


        // Manecilla de los minutos con lápiz negro...

        // Manecilla de los segundos con lápiz rojo...


        // Botón Central
        float radioBotón = m_Radio * .1f / 2;

        gfx.DrawEllipse(lápizGordoNegro, -radioBotón, -radioBotón,
                        radioBotón * 2, radioBotón * 2);
        gfx.FillEllipse(brochaGris, -radioBotón, -radioBotón,
                        radioBotón * 2, radioBotón*2);
    }
```

Hará falta poner `using System.Drawing.Drawing2D` al principio de `RelojAnalógico` para poder usar clases como `HatchBrush` o `Matrix`.

**Compilar y ejecutar para ver el resultado**.

## *2.7 Repintar al cambiar de tamaño. Invalidate()*

Cambie, si procede, el valor de la propiedad **FormBorderStyle** del formulario `RelojAnalógico` a **Sizable**.

**Compilar y ejecutar para ver el resultado** al cambiar el tamaño de la ventana del reloj analógico.

Añada una llamada a `Invalidate` al final del método `RelojAnalógico_Resize`.

**Compilar y ejecutar para ver el resultado**.

## *2.8 Dar la hora todo el tiempo. Temporizador*

Este proceso requiere del uso de un temporizador. Para ello, estando en modo diseño, seleccione en el cuadro de herramientas el control **Timer** y añádalo al formulario `RelojDigital` y habilítelo. Este temporizador deberá generar un evento cada segundo. El manejador que responda a los eventos generados por este temporizador se limitará a llamar a `MostrarHoraActual`.

**Compilar y ejecutar para ver el resultado**.

## *2.9 Mostrar/ocultar el reloj analógico con un botón*

Elimine el botón "Actualizar" del formulario principal, así como el código asociado. Cree un nuevo botón llamado "bt_Mostrar" que muestre y oculte alternativamente el reloj analógico. Cuando el reloj analógico no esté visible, el botón mostrará el título "Mostrar analógico", y cuando sí lo esté, "Ocultar analógico".

Nota: cuando una propiedad de un componente es de tipo **bool**, para asignar a esta propiedad un valor en función del valor de la misma u otras propiedades es más simple hacerlo como indican los ejemplos siguientes que utilizando sentencias **if … else**.

```
obj01.prop01 = !obj01.prop01;
obj02.prop02 = obj03.prop03;
obj04.prop04 = n < m;
```

**Compilar y ejecutar para ver el resultado**.

## *2.10 Doble búffer*

Para evitar que la imagen del reloj analógico parpadee, se deberá activar la propiedad **DoubleBuffered** en las propiedades del formulario.

## *2.11 Cambiar la hora con el ratón*

Añadir a `RelojAnalógico` manejadores para los eventos **MouseDown** y **MouseMove**. A continuación, edítelos para que invoquen a un método `ArrastrarAgujas`:

```
ArrastrarAgujas(sender, e);
```

Haga clic con el botón derecho sobre la línea anterior en cualquiera de los dos métodos. En el menú contextual, seleccione *Generar código auxiliar del método*. Edite el método generado de la siguiente manera:

```
private void ArrastrarAgujas(object sender, MouseEventArgs e)
{
  bool botonDerIzq = e.Button == MouseButtons.Right ||
                                 e.Button == MouseButtons.Left;
  bool pulsadoCentro = (e.X == m_Centro.X && e.Y == m_Centro.Y);

    if (m_Radio < 10 || !botonDerIzq || pulsadoCentro)
        return;

    double alfa;
    int horas, minutos;

    alfa = Math.Atan2(m_Centro.X - e.X, e.Y - m_Centro.Y);

    minutos = (int)((alfa / Math.PI / 2 * 12 + 6) * 60);
    horas = (minutos / 60)%12;
    minutos %= 60;

    if (e.Button == MouseButtons.Right)
        horas += 12;

    Console.WriteLine(horas + ":" + minutos + ":" + m_Hora.Second);
}
```

Observe que con el botón izquierdo del ratón puede ajustar la hora entre las 00:00 y las 11:59 y con el botón derecho entre las 12:00 y las 23:59.

La última línea muestra que hemos variado simplemente las horas y los minutos. Esto facilita los cálculos.

**Compilar y ejecutar para ver el resultado**. El resultado de `WriteLine` lo podrá ver en la ventana de resultados si ejecuta la aplicación en modo depuración (**F5**), aunque todavía no es posible ver las agujas del reloj digital moviéndose.

A continuación, añadiremos la funcionalidad necesaria para que al arrastrar las agujas del reloj analógico cambie también la hora del reloj digital, mediante el método `MostrarHoraActual` que también mostrará la hora en el reloj analógico, visualizando en ese momento el movimiento de agujas generado por el arrastre con el ratón. En el método `ArrastrarAgujas` de la clase `RelojAnalógico`, sustituya la llamada final a `Console.WriteLine` por las siguientes líneas:

```
RelojDigital relojD = (RelojDigital)this.Owner;
relojD.CambiarHora(horas, minutos, m_Hora.Second);
```

Utilice de nuevo la herramienta *código auxiliar del método* para crear el esqueleto del método `CambiarHora` anterior. Edite dicho método como se muestra a continuación:

```
internal void CambiarHora(int horas, int minutos, int segundos)
```

```
{
    DateTime actual = DateTime.Now;
    DateTime hora = new DateTime(actual.Year, actual.Month,
                                 actual.Day, horas, minutos, segundos);

    m_DesfaseHorario = hora - actual;

    MostrarHoraActual();
}
```

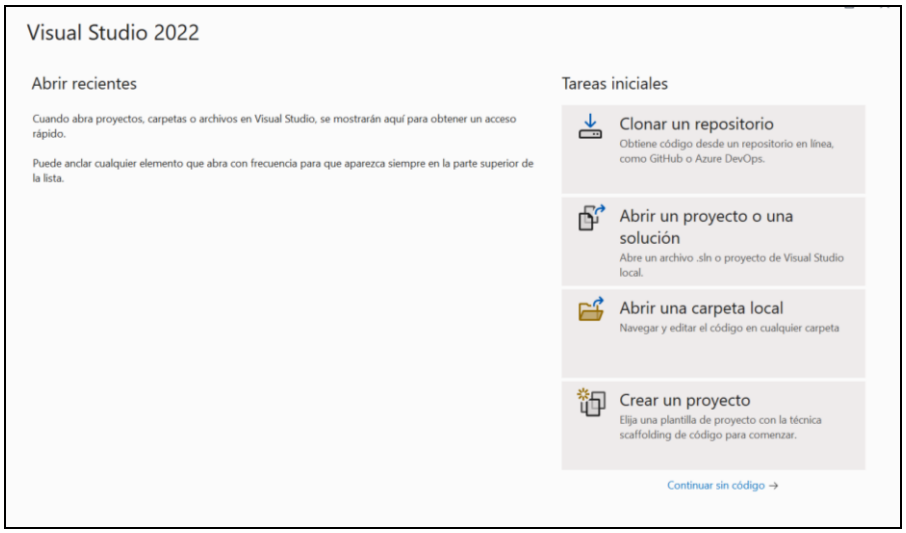
El modificador `internal` hace que este método sea accesible sólo para las clases pertenecientes a este ensamblado (proyecto).

**Compilar y ejecutar para ver el resultado**.

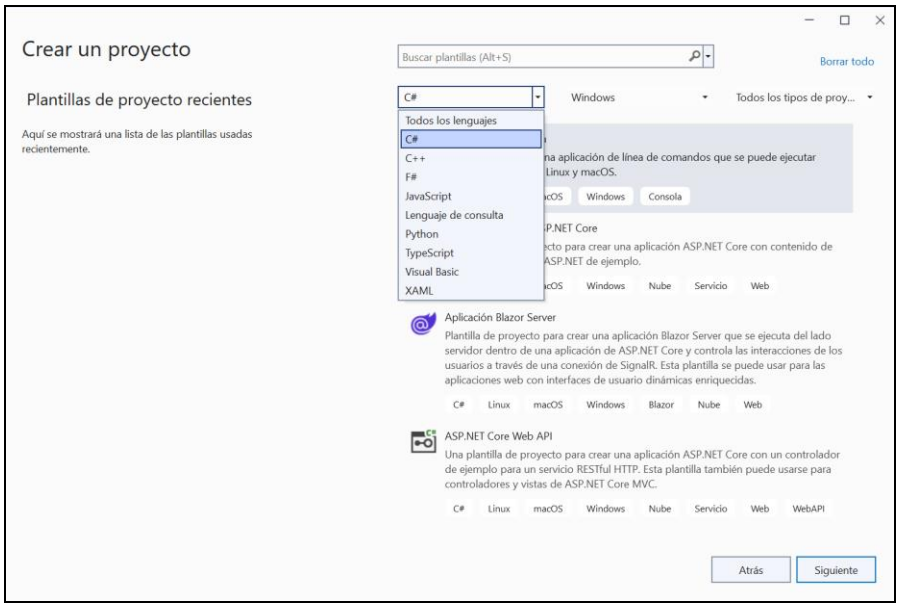
## ANEXO

Creación y configuración de un proyecto de tipo *Aplicación para Windows con Visual Studio 2022*. Las siguientes capturas de pantalla guían el proceso.

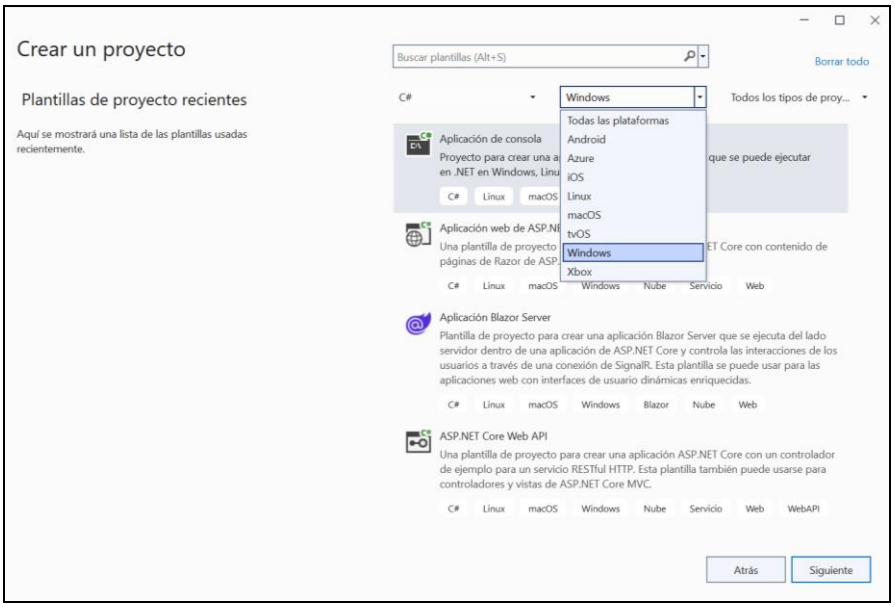
1. Pulse la opción “Crear proyecto”



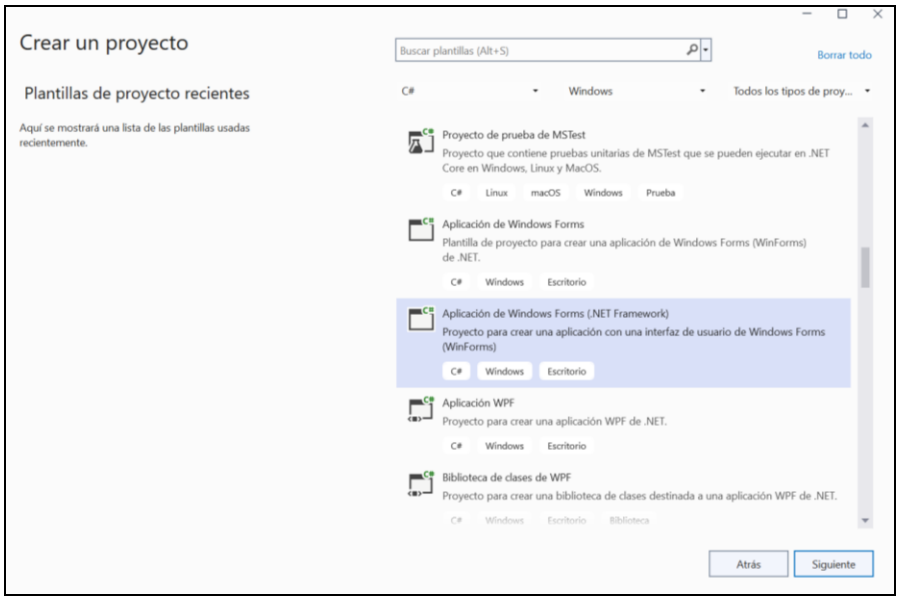

2. Seleccione el lenguaje de programación (C#)




3. Seleccione la plataforma (Windows)




4. Seleccione el tipo de proyecto (Aplicación de Windows Form (.NET Framework))




5. Rellene los datos en la ventana de configuración del proyecto. Coloque la solución y el proyecto en la misma carpeta.