TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TLAXIACO

Desarrollo de una interfaz gráfica de Windows Forms para una Calculadora Estándar utilizando Eventos de un Botton Personalizado en C#.

Asignatura: Tópicos Avanzados de Programación

Carrera: Ingeniera en Sistemas Computacionales.

Profesor: Ing. José Alfredo Román Cruz.

Alumno: Noelia Natividad González Sánchez

Grupo: 4UIS

Contenido

TALA DE IMÁGENES

OBJETIVO.

Realizar la implementación de un componente personalizado para su implementación como control en Visual Studio.

OBJETIVOS SECUNDARIOS.

1. Diseñar una biblioteca DLL para la implementación del control personalizado.
2. Declarar campos y propiedades heredadas de un control nativo de Visual Studio
3. Compilar la biblioteca para poder reutilizarla en nuevos proyectos de Visual Studio.
4. Realizar una aplicación de escritorio utilizando la nueva aplicación de escritorio.

PROCEDIMIENTO

1. Elaboramos nuestra clase biblioteca dll, abrimos el visual studio donde podremos hacer nuestro trabajo utilizamos la última versión disponible del .Net Framework.

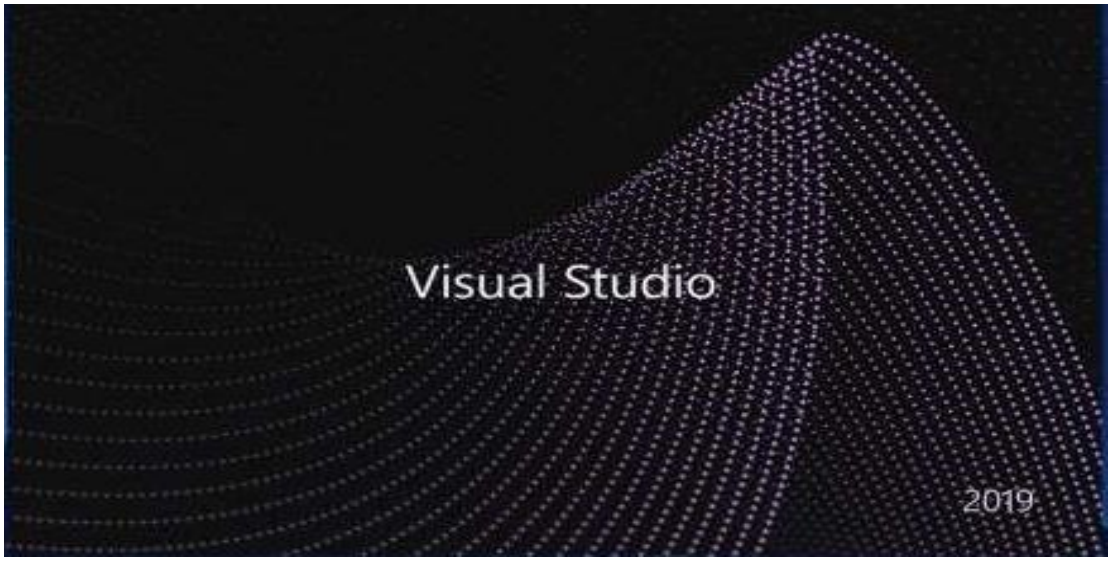

Imagen #1 Abrimos el programa Visual Studio.

2. Creamos un nuevo proyecto

Imagen #2 creación de un proyecto nuevo

3. El nuevo proyecto será de tipo biblioteca de clases C#

Imagen #3 Selección de la plantilla del proyecto tipo biblioteca de clases(.NET Framework)

4. Le damos siguiente para continuar con el nombre que le daremos a la biblioteca de clases.

Imagen #4 Seguir con el proyecto

5. Utilizamos el nombre de BotonPersonalizado para nuestro proyecto, seleccionamos el lugar donde deseamos guardarlo, podemos de igual forma seleccionar la versión del Framework.

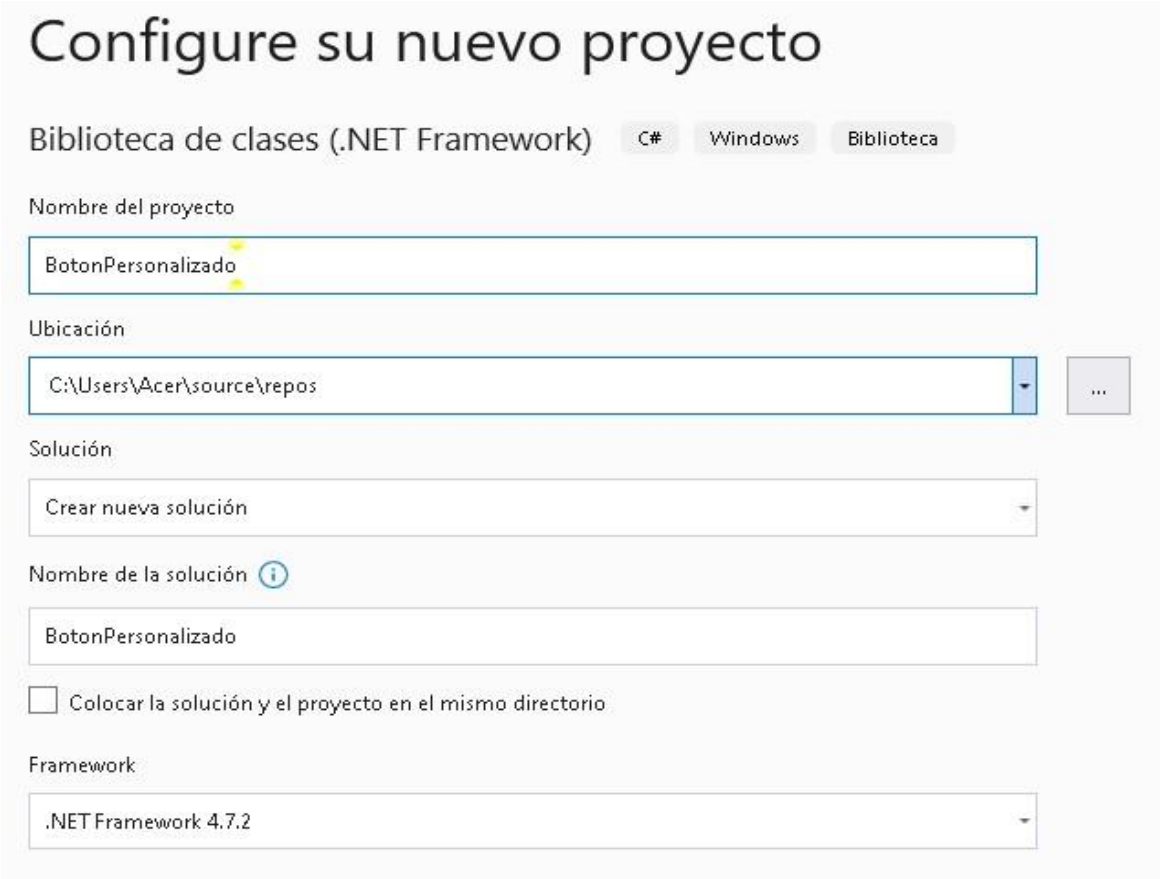

Imagen #5 Nombre del proyecto, su ubicación y versión.

6. El siguiente paso será crear la clase, en la parte del Explorador de soluciones daremos clic derecho en la ventana que se nos abre, colocamos el ratón encima y de damos en donde dice componente.

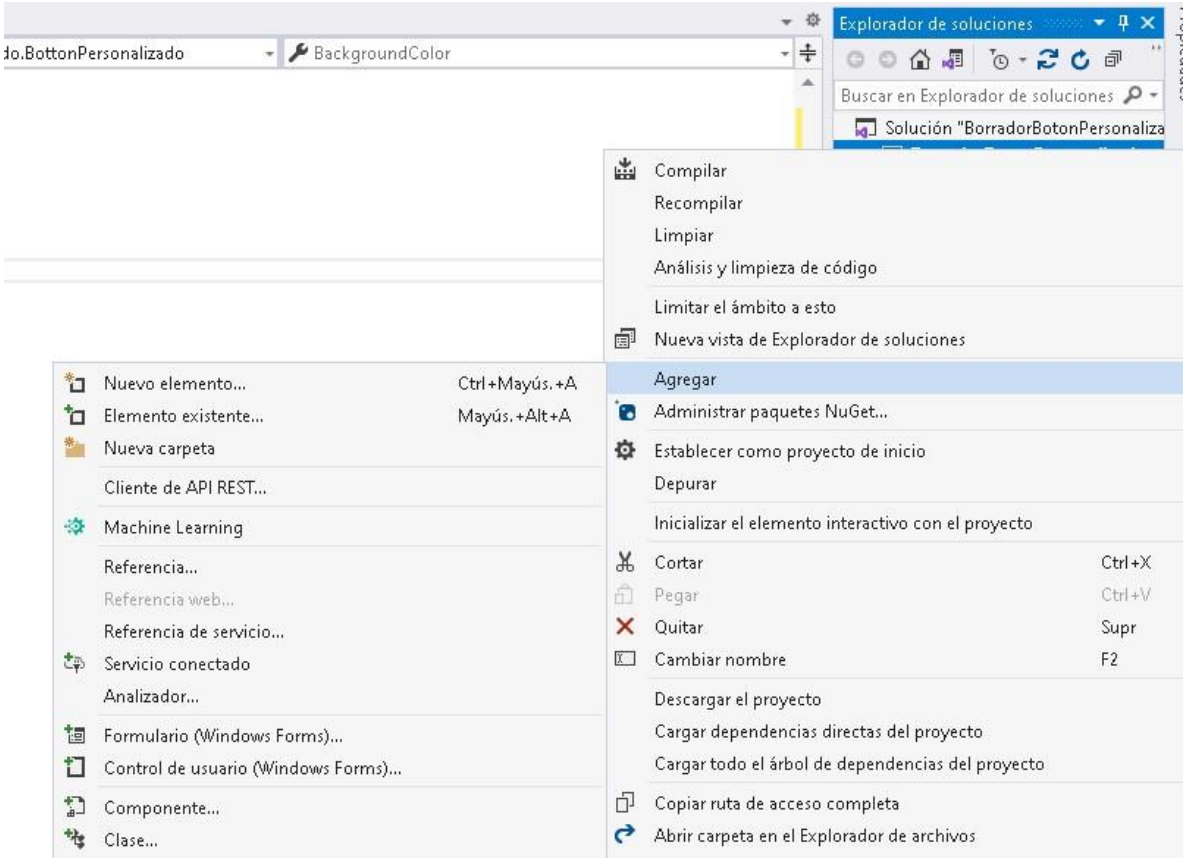

Imagen #6 Creación de un componente nuevo de tipo clase

7. Agregamos las librerías de Windows Forms.

```
7   using System.Windows.Forms;
8   using System.Drawing;
9   using System.Drawing.Drawing2D;
10  using System.ComponentModel;
```

Imagen #7 Importación de las librerías de windows Forms

8. Heredamos a la nuestra clase la clase del Botton.

```
14      public class BottonPersonalizado : Button
15      {
16
17      }
```

Imagen #8 Heredacion de la clase Botton en la nuestra.

9. Declaramos las propiedades que tendrá nuestro botón inicializando cada uno.

```
16          //Declaramos campos para la apariencia del Botton,asignando sus valores predeterminados
17          //ejemplo color de fondo y palanca en estado encendido y apagado.
18          //radio y tamaño del  bode
19          private Color onBackColor = Color.MediumSlateBlue;
20          private Color onToggleColor = Color.WhiteSmoke;
21          private Color offBackColor = Color.Gray;
22          private Color offToggleColor = Color.Gainsboro;
23          private bool solidStyle = true;
24          private int borderSize = 0;
25          private int borderRadius = 0;
26          private Color borderColor = Color.PaleVioletRed;
27
```

Imagen #9 Declaración de las propiedades nuevas

10. Generamos los métodos de las propiedades los cuales permitirán su uso y manipulación

```
29          //Generamos propiedades para exponer los campos anteriores declarados.
30          [Category("RJ Code Advance")]
31          public Color OnBackColor
32          {
33              get { return onBackColor; }
34              set
35              {
36                  onBackColor = value;
37                  this.Invalidate();
38              }
39          }
40
41          [Category("RJ Code Advance")]
42          public Color OnToggleColor
43          {
44              get { return onToggleColor; }
45              set
46              {
47                  onToggleColor = value;
48                  this.Invalidate();
49              }
50          }
51
52          [Category("RJ Code Advance")]
53          public Color OffBackColor
54          {
55              get { return offBackColor; }
56              set
57              {
58                  offBackColor = value;
59                  this.Invalidate();
60              }
61          }
```

Imagen #10 Métodos de las propiedades de apariencia del Botton

```
62
63          [Category("RJ Code Advance")]
64          public Color OffToggleColor
65          {
66              get { return offToggleColor; }
67              set
68              {
69                  offToggleColor = value;
70                  this.Invalidate();
71              }
72          }
73
74          [Browsable(false)]
75          public override string Text
76          {
77              get { return base.Text; }
78              set { }
79          }
80
81          [Category("RJ Code Advance")]
82          [DefaultValue(true)]
83          public bool SolidStyle
84          {
85              get { return solidStyle; }
86              set
87              {
88                  solidStyle = value;
89                  this.Invalidate();
90              }
91          }
```

Imagen #11 Métodos de las propiedades de apariencia del Botton

```
93          public int BorderSize
94          {
95              get { return borderSize; }
96              set
97              {
98                  borderSize = value;
99                  this.Invalidate();
100             }
101         }
102
103         [Category("RJ Code Advance")]
104         public int BorderRadius
105         {
106             get { return borderRadius; }
107             set
108             {
109                 borderRadius = value;
110                 this.Invalidate();
111             }
112         }
113
114         [Category("RJ Code Advance")]
115         public Color BorderColor
116         {
117             get { return borderColor; }
118             set
119             {
120                 borderColor = value;
121                 this.Invalidate();
122             }
123         }
```

Imagen #12 Métodos de las propiedades de apariencia del Botton

```
125         [Category("RJ Code Advance")]
126         public Color BackgroundColor
127         {
128             get { return this.BackColor; }
129             set { this.BackColor = value; }
130         }
131
132         [Category("RJ Code Advance")]
133         public Color TextColor
134         {
135             get { return this.ForeColor; }
136             set { this.ForeColor = value; }
137         }
```

Imagen #13 Métodos de las propiedades de apariencia del Botton

11. generamos el constructor de nuestra clase

```
139         //Constructor
140         public BottonPersonalizado()
141         {
142             this.MinimumSize = new Size(45, 22);
143         }
```

Imagen #14 Constructor de la clase

12. Creamos el método que creara nuestra figura Botton, para ello se agregan 2 arcos para el lado izquierdo (En el eje X=0 y eje Y=0 con tamaño arcSize, comenzando en un ángulo de 90 grados con un rango de 180 grados) y lado derecho (En el eje X=[Ancho de control – arcSize -2] y eje Y=0 con tamaño arcSize, comenzando en un ángulo de 270 grados con un rango de 180 grados).

```
150         //Methods
151         private GraphicsPath GetFigurePath(Rectangle rect, int radius)
152         {
153
154             GraphicsPath path = new GraphicsPath();
155             float curveSize = radius * 2F;
156
157             path.StartFigure();
158             path.AddArc(rect.X, rect.Y, curveSize, curveSize, 180, 90);
159             path.AddArc(rect.Right - curveSize, rect.Y, curveSize, curveSize, 270, 90);
160             path.AddArc(rect.Right - curveSize, rect.Bottom - curveSize, curveSize, curveSize, 0, 90);
161             path.AddArc(rect.X, rect.Bottom - curveSize, curveSize, curveSize, 90, 90);
162             path.CloseFigure();
163             return path;
164         }
```

Imagen #15 Método que dibuja nuestro Botton con dos parámetros

```
165         //Methods
166         private GraphicsPath GetFigurePath()
167         {
168             int arcSize = this.Height - 1;
169             Rectangle leftArc = new Rectangle(0, 0, arcSize, arcSize);
170             Rectangle rightArc = new Rectangle(this.Width - arcSize - 2, 0, arcSize, arcSize);
171
172             GraphicsPath path = new GraphicsPath();
173             path.StartFigure();
174             path.AddArc(leftArc, 90, 180);
175             path.AddArc(rightArc, 270, 180);
176             path.CloseFigure();
177
178             return path;
179         }
```

Imagen #16 Método que dibuja nuestro Botton sobrecargado sin parámetros

13. Se anula el método OnPaint

```
protected override void OnPaint(PaintEventArgs pevent)
{
    base.OnPaint(pevent);

    int toggleSize = this.Height - 5;
    pevent.Graphics.SmoothingMode = SmoothingMode.AntiAlias;
    pevent.Graphics.Clear(this.Parent.BackColor);
    Rectangle rectSurface = this.ClientRectangle;
    Rectangle rectBorder = Rectangle.Inflate(rectSurface, -borderSize, -borderSize);
    int smoothSize = 2;

    if (borderSize > 0)
        smoothSize = borderSize;

    if (borderRadius > 2) //Rounded button
    {
        using (GraphicsPath pathSurface = GetFigurePath(rectSurface, borderRadius))
        using (GraphicsPath pathBorder = GetFigurePath(rectBorder, borderRadius - borderSize))
        using (Pen penSurface = new Pen(this.Parent.BackColor, smoothSize))
        using (Pen penBorder = new Pen(borderColor, borderSize))
        {
            pevent.Graphics.SmoothingMode = SmoothingMode.AntiAlias;
            //Button surface
            this.Region = new Region(pathSurface);
            //Draw surface border for HD result
            pevent.Graphics.DrawPath(penSurface, pathSurface);

            //Button border
            if (borderSize >= 1)
                //Draw control border
                pevent.Graphics.DrawPath(penBorder, pathBorder);
        }
    }
```

Imagen #17 Anulamos OnPaint

```
    else //Normal button
    {
        pevent.Graphics.SmoothingMode = SmoothingMode.None;
        //Button surface
        this.Region = new Region(rectSurface);
        //Button border
        if (borderSize >= 1)
        {
            using (Pen penBorder = new Pen(borderColor, borderSize))
            {
                penBorder.Alignment = PenAlignment.Inset;
                pevent.Graphics.DrawRectangle(penBorder, 0, 0, this.Width - 1, this.Height - 1);
            }
        }
    }

    if (this.Checked) //ON
    {
        //Draw the control surface
        if (solidStyle)
            pevent.Graphics.FillPath(new SolidBrush(onBackColor), GetFigurePath());
        else pevent.Graphics.DrawPath(new Pen(onBackColor, 2), GetFigurePath());
        //Draw the toggle
        pevent.Graphics.FillEllipse(new SolidBrush(onToggleColor),
          new Rectangle(this.Width - this.Height + 1, 2, toggleSize, toggleSize));
    }
    else //OFF
    {
        //Draw the control surface
        if (solidStyle)
            pevent.Graphics.FillPath(new SolidBrush(offBackColor), GetFigurePath());
        else pevent.Graphics.DrawPath(new Pen(offBackColor, 2), GetFigurePath());
        //Draw the toggle
        pevent.Graphics.FillEllipse(new SolidBrush(offToggleColor),
          new Rectangle(2, 2, toggleSize, toggleSize));
    }
```

Imagen #18 Anulamos OnPaint

14. Código con métodos de características para el Botton

```
252         protected override void OnHandleCreated(EventArgs e)
253         {
254             base.OnHandleCreated(e);
255             this.Parent.BackColorChanged += new EventHandler(Container_BackColorChanged);
256         }
257
258         private void Container_BackColorChanged(object sender, EventArgs e)
259         {
260             this.Invalidate();
261         }
262         private void Button_Resize(object sender, EventArgs e)
263         {
264             if (borderRadius > this.Height)
265                 borderRadius = this.Height;
266         }
```

Imagen #19 Métodos de características del Botton

RESULTADOS

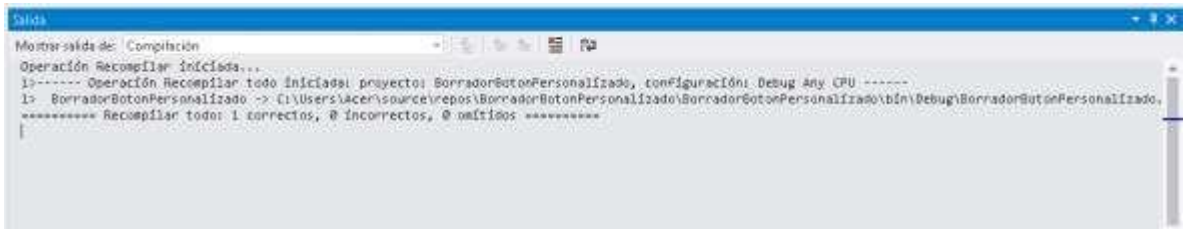

Imagen #20 Compilación del código exitosamente

Imagen #21 Diseño con los botones en funcionamiento

CONCLUSIONES

El diseño de un botón personalizado hace que nuestra interfaz gráfica se vea más llamativa, al diseñar un botón podemos realizarlo de manera que sea más entendible para nosotros puesto que al personalizarlo adaptamos a nuestro gusto los controles, nuestro diseño será único, con nuevas propiedades y características, las librerías dinámicas nos dan una ventaja en tiempo, los botones se pueden utilizar en cualquier app sin problemas.