LUG MIII U6 2018

Lenguajes De Última Generación (Universidad Abierta Interamericana)

Carrera: **Analista Programador**

Lenguaje de última generación

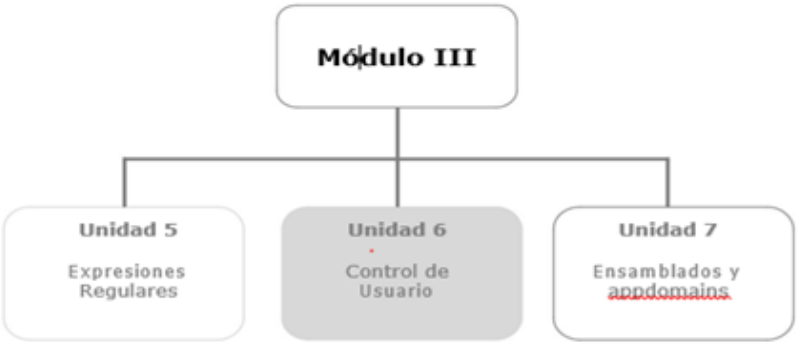
Módulo III

Bases para construir aplicaciones empresariales

Unidad 6

Control de Usuario

Autor de contenidos: Ing. Mauricio Prinzo



Presentación

En esta unidad abordaremos superficialmente algunos conocimientos para el desarrollo de aplicaciones en la web utilizando controles de usuarios. No nos detendremos demasiado porque sencillamente, no es contenido específico de esta asignatura.

Igual que en la unidad anterior, le propondremos la rememoración y aplicación de algunos conceptos de los lenguajes de programación que son pilares fundamentales en el desarrollo de aplicaciones empresariales.

Tal como lo señalamos en el cierre de la unidad, usted comprobará que a medida que avanza en el desarrollo de aplicaciones aparecerán funcionalidades repetidas. Esperamos que encuentre estas regularidades para poder generar recursos propios que le permitan tener un mayor control sobre los desarrollos asegurando aplicaciones escalables y seguras

Esperamos que a través del estudio de esta unidad, adquiera capacidad para:

- Ampliar los controles de la caja de herramientas
- Personalizar las funcionalidades de los controles
- Comprender las estructuraras de datos utilizadas en un lenguaje de programación
- Trabajar con estructuras planas de datos que faciliten el intercambio de datos.

A continuación, le presentamos un detalle de los contenidos y actividades que integran esta unidad. Usted deberá ir avanzando en el estudio y profundización de los diferentes temas, realizando las lecturas requeridas y elaborando las actividades propuestas, algunas de desarrollo individual y otras para resolver en colaboración con otros estudiantes y con su profesor tutor.

Contenidos y Actividades

1. Entorno Web

- **1.1. Diferencia entre páginas estáticas y dinámicas**
- **1.2. Proyecto Web**

2. Controles de Usuario

- **2.1. Almacenamiento en Cache**

3. Controles Personalizados

- **3.1. Controles compuestos**

4. Control Personalizados para WinForm

- **4.1. Tipos de controles**

Lectura Requerida

- Deitel, Harvey M.; Deitel, Paul J.; Vidal Romero Elizondo, Alfonso(Traductor); y otros. **Cómo programar en C#.** 2a.ed.--México, DF.

Cierre de la unidad

Trabajo colaborativo/Foro

- Foro de la Unidad 6.

Para el estudio de estos contenidos usted deberá consultar la bibliografía que aquí se menciona:

BIBLIOGRAFÍA OBLIGATORIA

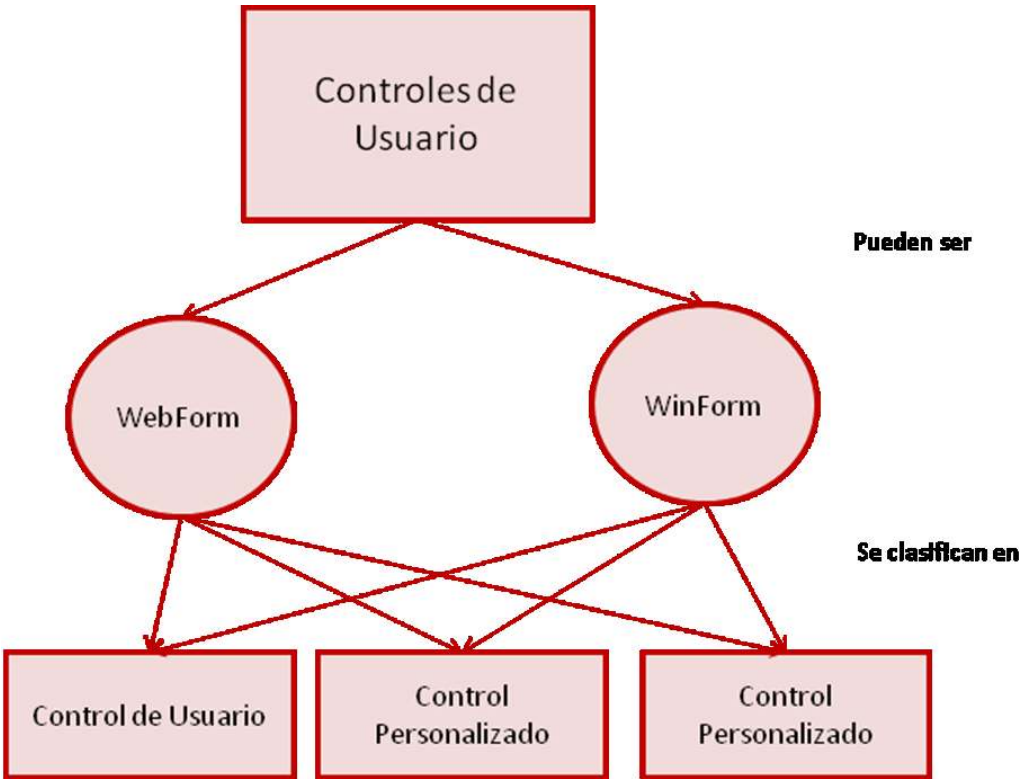
- Deitel, Harvey M.; Deitel, Paul J.; Vidal Romero Elizondo, Alfonso(Traductor); y otros. **Cómo programar en C#.** 2a.ed.-- México, DF.
- Chappell, David; Garza Marín, David. **Aplique .NET**.-- México, DF: Pearson Educación de México, c2003. Xiv.

Links a temas de interés

- http://Biblioteca.vaneduc.edu.ar/
  - **El portal de la biblioteca universitaria con material y links de interés, además del asesoramiento de un referencista especializado.**

Organizador Gráfico

El siguiente esquema le permitirá visualizar la interrelación entre los conceptos que a continuación abordaremos.



Lo/a invitamos ahora a comenzar con el estudio de los contenidos que conforman esta unidad.

1. Entorno Web

Un nuevo ámbito para desarrollar aplicaciones es el entorno Web. Básicamente, potencia el uso de interfaces livianas. El proceso es muy sencillo: necesita un computador que oficie de servidor, en el servidor se encuentra la página web y el usuario solicita por http el archivo alojado en el servidor.

Por ejemplo, cuando navegamos en internet estamos usando este tipo de servicios. El visual Studio tiene la posibilidad de emular un servidor web en un ámbito local.

Pero, **¿qué es Internet?**

El origen de Internet se remonta a la década del '60 y en sus primeros pasos aparece vinculado al desarrollo de la red ARPA que planteaba la necesidad de interactuar entre varias computadoras y facilitar la posibilidad de compartir información.

Hoy, estamos conectados todo el tiempo a la Red para ver los correos o para buscar información. En 1989 nace lo que hoy conocemos como Internet, fue Tim Bernes Lee quien basando en un estándar de IBM desarrolló el primero lenguaje de marcas para crear páginas Web.

Pero no nos vamos a extender mucho más con este tema. Desarrollaremos a continuación aquellos necesarios para desarrollar un proyecto Web.

1.1. Diferencia entre paginas estáticas y dinámicas

Existen dos tipos de páginas Web, unas estáticas y otras dinámicas.

Las **páginas estáticas** son aquellas que solo muestran información donde el contenido no varía en el tiempo. Frecuentemente se desarrollan usando un lenguaje de marcas llamado HTML.

Un ejemplo elemental de una página Web estática donde sólo se muestra su esqueleto se muestra a continuación:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title></title>
</head>
<body>

</body>
</html>
```

Las **páginas dinámicas**, se usan para interactuar en su mayoría con base de datos. Para entender, si necesitamos mostrar el resumen de una cuenta corriente usaremos una página dinámica.

El lenguaje que se usa en la mayoría de estos proyectos es ASP.NET aunque es posible que usemos también una combinación entre páginas HTML y lenguaje de programación como ASP.NET. La diferencia, y el lado fascinante, es que en el ámbito de ASP.NET trabajaremos con objetos.

El ejemplo se muestra a continuación:

```
namespace WebApplication1
{
   public partial class WebForm1 : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {

      }
    }
```

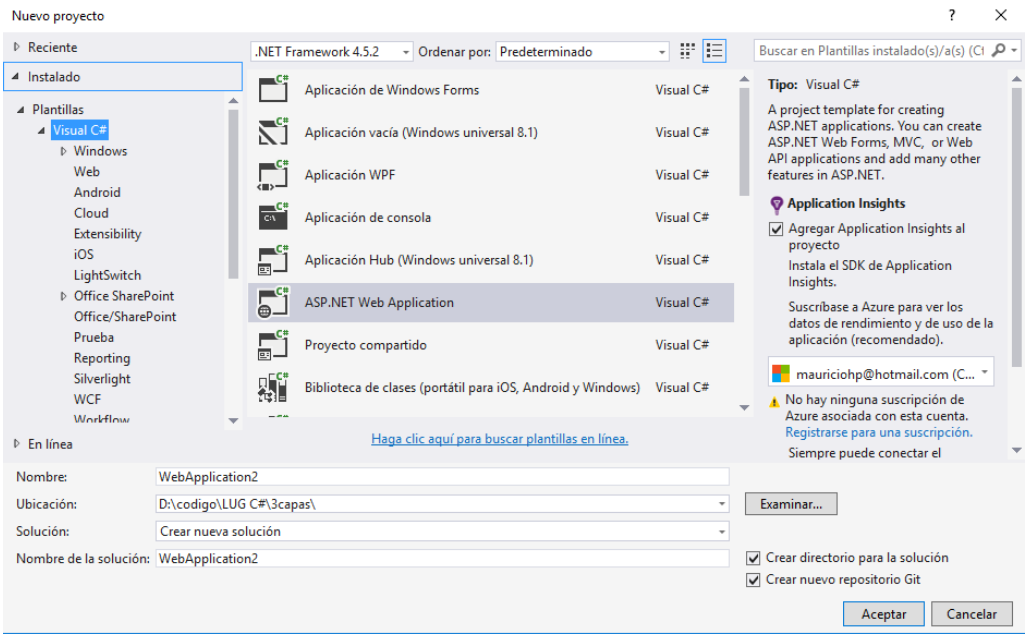
Veamos a continuación cómo desarrollar un proyecto web y ver estas páginas en acción.

1.2. Proyecto Web

Para crear un proyecto Web seguimos los siguientes pasos:
1- Entrar en Visual Studio

2- Seleccionar el lenguaje, en este caso Visual Basic

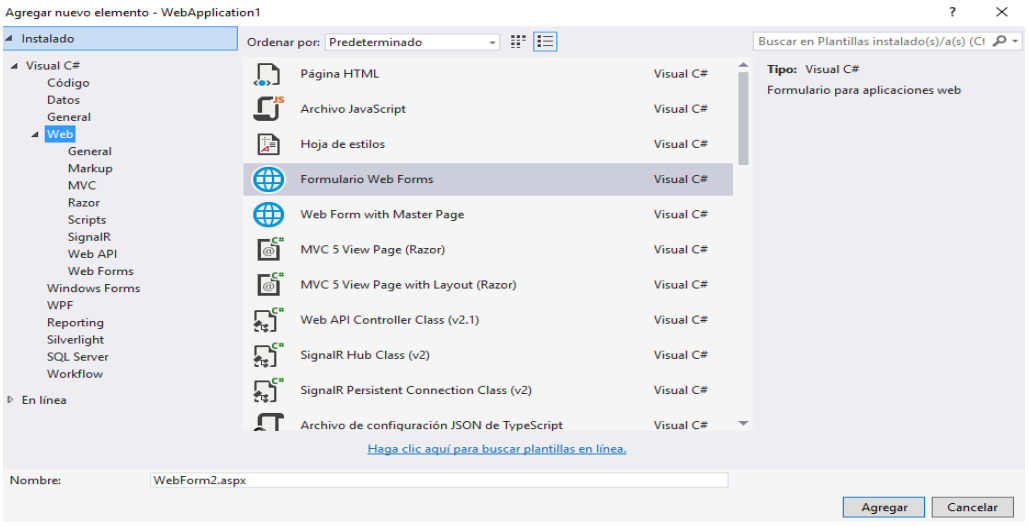
3- Seleccionar Proyecto Web: Aplicación web vacía



4- Listo, ya podemos desarrollar una aplicación web

5- Agregaremos una página **WebForm**, en el menú principal

Proyecto > Agregar nuevo elemento



6- Seleccionamos Formulario **Web Forms**

¿Cómo desarrollamos una página estática?

Seguimos los pasos anteriores pero seleccionamos el elemento página HTML. Luego, todo es cuestión de usar etiquetas HTML (no nos detendremos en los detalles de este lenguaje pero si recuerda algo de XML se dará una idea acerca de qué es una etiqueta de HTML).

Como en XML las etiquetas tiene apertura y cierre, por ejemplo

```
<H1>Título</H1>
```

La etiqueta del esqueleto representa diferentes partes de una página web, las cuales se muestran a continuación:

¿Cómo programamos una página dinámica?

Las páginas se programan de la misma manera que los formularios Windows. Pero cuando llega del lado del cliente siempre tenemos HTML.

La ventaja de este escenario es que podemos aprovecharnos del Frame-Work, tal como lo hacemos en una aplicación Windows.

Las partes del proyecto Web son:

- **WebForms**: uno o más archivos con extensiones **.aspx**

- **Code-Behind**: Archivos asociados a **WebForms** que contiene el código el lado del servidor.

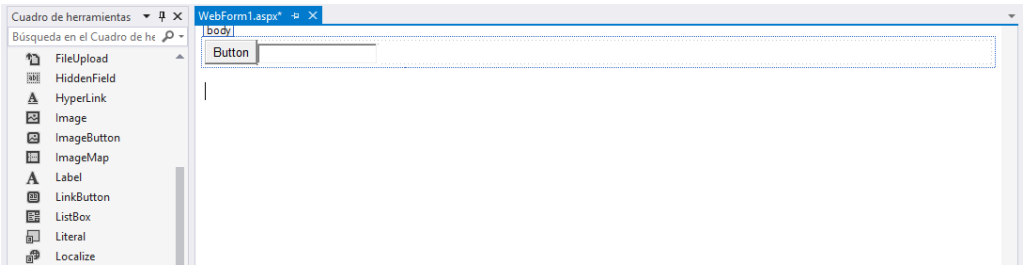
2. Controles de usuario

Los controles de usuario son muy fáciles de desarrollar y se programan en ASP.net. Son muy parecidos a las páginas .aspx que pueden contener un bloque de texto HTML.

¿Cómo creamos un control de usuario?

El proceso de creación es muy sencillo, primero, crear una página .aspx con un grupo de controles sencillos y luego, convertirlo a controles de usuario.

Luego de agregar una página .aspx, movemos desde la caja de herramientas dos controles HTML. En este caso una caja de texto y un botón. Nos quedaría la siguiente imagen:



A continuación, modificamos la directiva que se encuentra en la cabecera:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBe-
hind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
   <div>

      <asp:Button ID="Button1" runat="server" Text="Button" />
      <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

   </div>
   </form>
</body>
</html>
```

E incluimos el siguiente código:

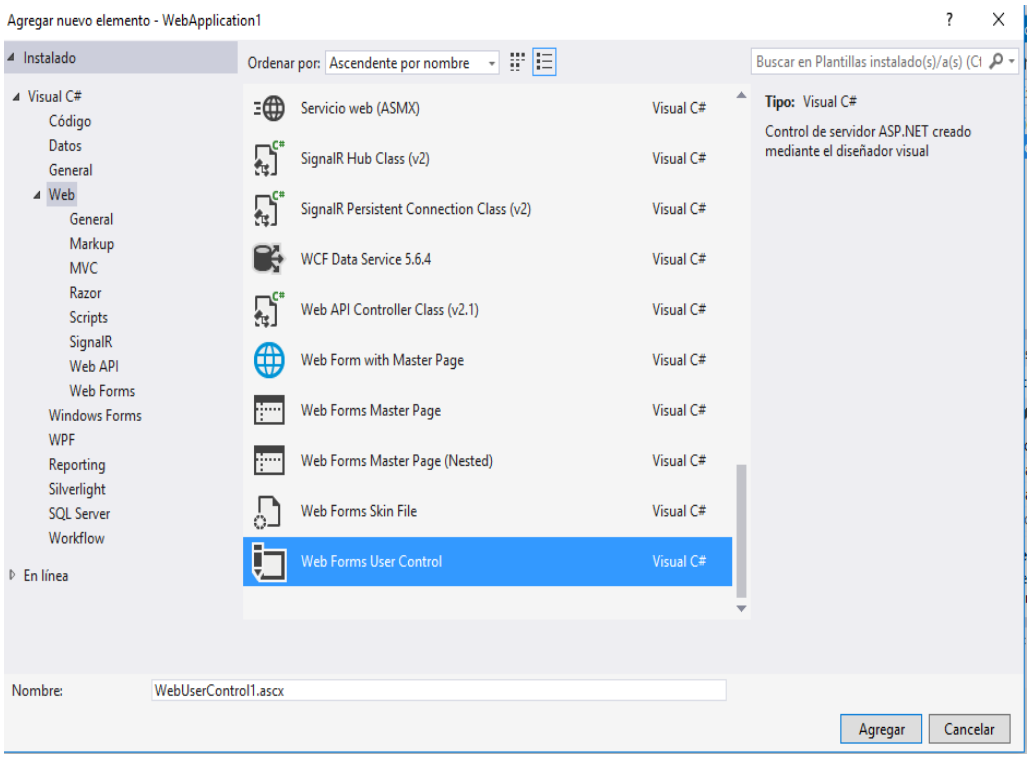
```
<%@ Cotrol Language="C#" %>

      <asp:Button ID="Button1" runat="server" Text="Button" />
      <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
```

Como puede comprobar es muy sencillo, solo nos resta guardar el archivo con el nombre **control.ascx**.

Ya tenemos creado nuestro control de usuario. Solo queda arrastrar el control **ascx** al cuerpo de una página **.aspx.** La ventaja que esto tiene es que no importa cuántas veces se necesite, solo hay que arrastrar el control y listo.

Otra forma de realizar lo mismo es agregando un elemento nuevo y seleccionando Web Form User Control



Después se repiten los pasos anteriores, arrastramos los controles de HTML al cuerpo del control .ascx y tenemos un nuevo control.

2.1. Almacenamiento en el cache

Para complementar los controles, podemos almacenar total o parcialmente su estado para que funcionen más rápido. Esto se resuelve con el uso de la directiva **@OutputCache**, que almacena sólo las partes de la página, es decir, el HTML generado por el control.

```
<%@ OutputCache Duration="2000" VaryByParam="*" %>
```

En **VaryByControl** se asignan las propiedades del control, utilizando la dupla nombre y valor separado por una coma. Se utiliza cuando se omite **VaryByParam**.

```
<%@ OutputCache Duration="2000" VaryByControl="Name,Value" %>
```

3. Controles personalizados

Los **controles personalizados** extienden las funcionalidades de los controles básicos de la plataforma de programación. Se trata de una clase que hereda de:

```
public partial class WebUserControl1 : System.Web.UI.UserControl
```

Se sobre escriben las funciones que permiten dibujar el control en la página web, tal como se muestra en el código siguiente:

```
public class Class1 : System.Web.UI.UserControl {

    protected override void Render(System.Web.UI.HtmlTextWriter writer) {
        base.Render(writer);
    }

    protected override void OnPreRender(System.EventArgs e) {
        base.OnPreRender(e);
    }
}
```

Los controles personalizados, puede ampliar y mejorar las funciones de un control básico o crear un control Nuevo que reproduzca código HTML

```
public class CajaTextoHTML : System.Web.UI.WebControls.WebControl {

   protected override void OnPreRender(System.EventArgs e) {
      if (!Page.ClientScript.IsClientScriptBlockRegistered("CajaTExtoHTML")) {
         Pa-
ge.ClientScript.RegisterClientScriptBlock(Page.GetType, "CajaTextoHTML", Script);
      }

      base.OnPreRender(e);
   }

   protected override void Render(System.Web.UI.HtmlTextWriter writer) {
      writer.Write(HTML);
      // MyBase.Render(writer)
   }

   private void HTML() {
      string T = "";
      T = (T + "<input type=\"text\" ID=\"CajaTExto\" name=\'CajaTExto\' />");
      T = (T + "<input type=\"button\" ID=\"butValidar\" name=\'butValidar\' onclick
=\'Validar(\'CajaTexto\')\' />");
      return T;
   }

   private void Script() {
      string T = "";
      T = (T + "<script language=\'javascript\' type=\'text/javascript\'>");
      T = (T + "function Validar(NombreControl) {");
      T = (T + "var Control = document.getElementById(NombreControl);");
      T = (T + "if (Control.value == \")");
      T = (T + "{ alert(\"upss!\") } ");
      T = (T + "}</script>");
      return T;
   }
}
```

3.1. Controles compuestos

Cuando creamos un control personalizado que enlaza más de un control, se asocian en un mismo contenedor dando lugar a un control compuesto.

La ventaja de los controles compuestos es que se pueden enlazar los controles internos con mucha facilidad creando recursos ricos para nuestros desarrollos.

Le proponemos un poco de práctica y de puesta a prueba de la comprensión de los conceptos.

Actividades para la facilitación de los aprendizajes (1)

Este trabajo le permitirá poner en uso los conceptos básicos para crear y usar con eficacia los controles desarrollados por el usuario.

Comenzaremos con un caso simple donde podrá experimentar el uso de controles de usuario y controles personalizados

Consigna

1. En primer lugar, debemos crear un proyecto web vacio al cual le agregamos dos paginas **aspx**.

En la primera página aspx le agregamos un control textbox, un label y un control button. Nuestra aplicación deberá mostrar un saludo cuando presionamos el botón, tomando el nombre que colocamos en la caja de texto para completar el mensaje.

Una vez completado este procedimiento, realizaremos un doble click sobre el control botón para pasar al ámbito de código. Allí escribiremos lo siguiente:

```
public class WebUserControl1 : System.Web.UI.UserControl {

    protected void Page_Load(object sender, System.EventArgs e) {
    }

    protected void Button1_Click(object sender, EventArgs e) {
        this.Label1.Text = ("Hola " + this.TextBox1.Text);
    }
}
```

2. Luego, complementaremos el ejercicio anterior creando un control **textbox** personalizado. El control nuevo que generaremos extendiendo la clase **textbox**, tendrá una propiedad que lo habilitará o lo bloqueará cuando la caja de texto este vacía.

El código del control personalizado será

```
public class textboxnuevo : System.Web.UI.WebControls.TextBox {

    enum enumEstado {
        Bloqueado,
        Habilitado,
    }

    property enumEstado {get; set; }

    protected override void Render(System.Web.UI.HtmlTextWriter writer) {
        if ((this.Text.Trim == "")) {
            this.Estado = enumEstado.Bloqueado;
        }
        else {
            this.Estado = enumEstado.Habilitado;
        }
    }
}



    protected override void OnPreRender(System.EventArgs e) {
        base.OnPreRender(e);

    }
```

Luego de guardarlo notará que aparece un nuevo control en la caja de herramientas.

El código del control de usuario cambiará al siguiente

```
public class WebUserControl1 : System.Web.UI.UserControl {

   protected void Page_Load(object sender, System.EventArgs e) {
      this.textboxnuevo1.Text = "";
   }

   protected void Button1_Click(object sender, EventArgs e) {
      if ((this.textboxnuevo1.Estado == textboxnuevo.enumEstado.Habilitado))
{
         this.Label1.Text = ("Hola " + this.textboxnuevo1.Text);
      }
      else {
         this.Label1.Text = "No puso un nombre";
      }

   }
}
Endclass Unknown {
}


   protected void textboxnuevo1_TextChanged(object sender, EventArgs e) {
   }
```

Si tiene alguna duda acerca de la resolución de este ejercicio o si se presenta alguna dificultad durante la programación, por favor, consulte a sus pares y tutor.

4. Controles personalizados para Winform

Siguiendo la línea de los desarrollos anteriores, cuando desarrollamos aplicaciones de escritorio, podemos crear controles personalizados para mejorar la calidad de nuestros desarrollos..

La ventaja de personalizar los controles es que se pueden extender las funcionalidades de los controles nativos de la plataforma de programación.

La clase **Control** es la clase base de los controles de formularios **WinForms**. Ofrece la interface necesaria de los controles básicos de .NET y toda la lógica para la presentación visual del control.

Las tareas son las siguientes según las fuentes de MSDN:

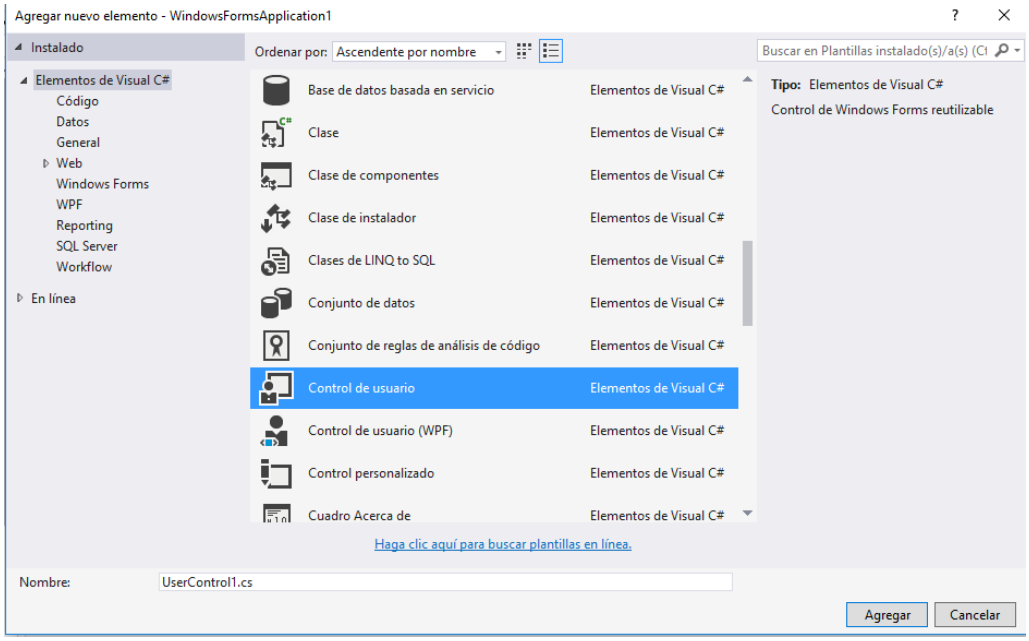
1. Expone un controlador de ventanas.
2. Administra el enrutamiento de mensajes.
3. Proporciona eventos del mouse (ratón) y del teclado y muchos otros eventos de la interfaz de usuario.
4. Proporciona características de diseño avanzadas.
5. Contiene muchas propiedades específicas a la presentación visual, como **ForeColor**, **BackColor**, **Height** y **Width**.
6. Proporciona la seguridad y compatibilidad para subprocesos necesarias para que un control de formularios Windows Forms actúe como un control de Microsoft® ActiveX®.

4.1. Tipos de controles

Se pueden desarrollar dos tipos de controles: compuesto y ampliado.
Controles compuestos

Se trata de un grupo de controles de formularios WinForms dentro de un contenedor común. En algunas bibliografías puede llamarse control de usuario.

Desde la opción Proyecto del menú y un nuevo elemento puede agregar un control de usuario



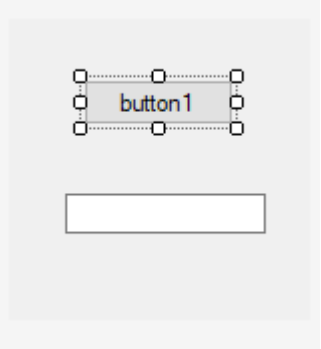
Veamos el siguiente código ejemplo:

```
public partial class UserControl1 : UserControl
   {
      public UserControl1()
      {
         InitializeComponent();
      }

      private void UserControl1_Load(object sender, EventArgs e)
      {

      }
   }
```

Agregando los controles como se muestra en la siguiente figura



Control Ampliado

Es cuando queremos preservar la funcionalidad de un control existente pero además le agregamos nuevas funcionalidades

```
public class cajaTextoNueva : System.Windows.Forms.TextBox {

   protect-
ed override void OnPaint(System.Windows.Forms.PaintEventArgs e)
{
      base.OnPaint(e);
   }
}
```

A medida que nos adentramos en el desarrollo de aplicaciones iremos notando que muchas veces las funcionalidades se repiten. Un buen programador se alerta de estas vivencias y toma cartas en el asunto. Crear recursos propios nos permite tener un mayor control y nos asegura aplicaciones escalables y seguras.

Le proponemos la siguiente lectura. En ella podrá ampliar y profundizar los contenidos presentados aquí, a modo de primera aproximación.

Lectura Requerida

- Deitel, Harvey M.; Deitel, Paul J.; Vidal Romero Elizondo, Alfonso(Traductor); y otros. **Cómo programar en C#.** 2a.ed.--México, DF.

Cierre de la unidad

Le proponemos compartir experiencias con sus pares y tutores a través de este espacio de construcción compartida de conocimientos.

Trabajo colaborativo/Foro

- Foro de la Unidad 6.