Herramientas de Programación 1

Tema Nº2:

Diseño de formularios

Indicador de logro Nº2:los formularios y controles comunes de Windows, configurando correctamente sus propiedades y manipulando adecuadamente sus eventos.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº2:**

DISEÑO DE FORMULARIOS.

Los formularios de Windows (WinForms) es una biblioteca de clase de interfaz gráfica de usuario (GUI) que se incluye en .Net Framework. Su objetivo principal es proporcionar una interfaz fácil para desarrollar las aplicaciones de computadoras de escritorio. WinForms solo se puede usar para desarrollar aplicaciones de formularios Windows Forms, no aplicaciones web. Las aplicaciones WinForms pueden contener diferentes tipos de controles como formularios, etiquetas, cuadros de lista, menús, etc.

Formularios: FORM

Un formulario Form es una representación de cualquier ventana mostrada en una aplicación. La clase Form se puede utilizar para crear ventanas estándar y otras de características y funcionalidades más complejas.

Propiedad y eventos.

Las propiedades más comunes de un formulario son:

|  |  |
| --- | --- |
| Propiedad | Descripción |
| BackColor | Obtiene o establece el color de fondo del control. |
| BackgroundImage | Obtiene o establece la imagen de fondo que se muestra en el control. |
| ControlBox | Obtiene o establece un valor que indica si se muestra un cuadro de control en la barra de título del formulario. |
| FormBorderStyle | Obtiene o establece el estilo del borde del formulario. |
| Name | Obtiene o establece el nombre del control. |
| StartPosition | Obtiene o establece la posición inicial del formulario en tiempo de ejecución. |
| Text | Obtiene o establece el texto asociado al control. |

Las propiedades de formularios y controles se pueden cambiar durante el diseño o mediante la edición de código. En el ejemplo que continúa veremos cómo actualizar las propiedades de un formulario usando ambas formas.

Eventos

Los objetos tienen la posibilidad de transmitirnos sus reacciones generando eventos. Los eventos se utilizan mucho en el diseño de la interfaz gráfica, ya que nos permiten obtener datos relativos a las acciones efectuadas por el usuario de la aplicación. Un ejemplo clásico es el evento OnClick, que se desencadena (dispara) cuando hacemos click dentro del formulario o control.

Cada control tiene un conjunto de eventos que les son característicos, en el ejemplo utilizaremos dos eventos para los formularios que vamos a programar.

Los eventos más comunes de un formulario son:

|  |  |
| --- | --- |
| Evento | Descripción |
| Closed | Tiene lugar cuando el formulario está cerrado. |
| Closing | Tiene lugar cuando se cierra el formulario. |
| Load | Se produce antes de que se muestre un formulario por primera vez. |

Control: LABEL

El control label representa a una etiqueta estándar de Windows.

Propiedades

Las propiedades más comunes de una etiqueta son:

|  |  |
| --- | --- |
| Propiedad | Descripción |
| Font | Obtiene o establece la fuente del texto que muestra el control. |
| ForeColor | Obtiene o establece el color de primer plano del control. |
| Name | Obtiene o establece el nombre del control. |
| Text | Obtiene o establece el texto asociado al control. |
| Visible | Obtiene o establece un valor que indica si se muestran el control y todos sus controles secundarios. |

Eventos

Generalmente no se trabajan con los eventos de este tipo de controles por lo que no vamos a listarlos. Si en temas posteriores lo necesitamos vamos a explicarlo previamente.

Manejo de Controles parte 1

Control: TEXTBOX

Los cuadros de texto en formularios Windows Forms se utilizan para obtener entradas del usuario o para mostrar texto. El control TextBox se utiliza generalmente para el texto que se puede editar, aunque también puede configurarse como control de sólo lectura.

Propiedades

Las propiedades más comunes de un cuadro de texto son:

|  |  |
| --- | --- |
| Propiedad | Descripción |
| Multiline | Obtiene o establece un valor que indica si es un control TextBox multilínea. |
| PasswordChar | Obtiene o establece el carácter utilizado para enmascarar caracteres de una contraseña en una sola línea TextBox control. |
| ReadOnly | Obtiene o establece un valor que indica si el texto en el cuadro de texto es de solo lectura. |
| MaxLength | Obtiene o establece el número máximo de caracteres que el usuario puede escribir o pegar en el control de cuadro de texto. |
| Name | Obtiene o establece el nombre del control. |
| TabIndex | Obtiene o establece el orden de tabulación del control en su contenedor. |
| Text | Obtiene o establece el texto asociado al control. |

Eventos

Los eventos más comunes de un cuadro de texto son:

|  |  |
| --- | --- |
| Evento | Descripción |
| Click | Se produce cuando se hace clic en el control. |
| Leave | Se produce cuando el foco de entrada deja el control. |
| TextChanged | Se produce cuando cambia el valor de la propiedad Text. |

Control: BUTTON

* El control Button de los formularios Windows Forms permite al usuario hacer click en él para ejecutar una acción. Cada vez que el usuario hace click en un botón, se invoca al controlador del evento Click. El código se ubica en el controlador del evento Click para ejecutar la acción deseada.

Propiedades

* Las propiedades más comunes de un botón son:

|  |  |
| --- | --- |
| Propiedad | Descripción |
| [AcceptButton](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.form.acceptbutton?view=netframework-4.8) | Nombre del botón que responderá (a su evento Click) cuando se presione la tecla Enter |
| BackColor | Obtiene o establece el color de fondo del control. |
| CancelButton | Nombre del botón que responderá (a su evento Click) cuando se presione la tecla ESC |
| Font | Obtiene o establece la fuente del texto que muestra el control. |
| ForeColor | Obtiene o establece el color de primer plano del control. |
| Name | Obtiene o establece el nombre del control. |
| TabIndex | Obtiene o establece el orden de tabulación del control en su contenedor. |
| Text | Obtiene o establece el texto asociado al control. |

Eventos

* Como vimos previamente el evento más común del botón es:

|  |  |
| --- | --- |
| Evento | Descripción |
| Click | Se produce cuando se hace clic en el control. |

**Ejemplos:**

**RECURSOS**

1. Hardware

* Una computadora con 4 GB e RAM y procesador de 2.0 GHz.

1. Software

* Visual Studio Professional 2015 / 2017

**PROCEDIMIENTO**

**Laboratorio 2:**

2.1 Trabajar con formularios de Windows.

**Objetivos**

* Desarrollar aplicaciones utilizando formularios, modificar sus propiedades y entender sus eventos.

**Introducción**

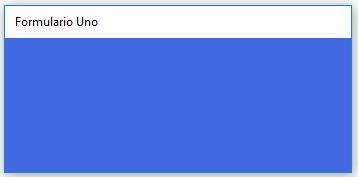
Durante esta actividad, tendrá un primer acercamiento a los formularios de Windows.

Paso 1: Creamos el proyecto Semana02 de la misma forma que creamos el proyecto Semana01.

Paso 2: Cambiamos las siguientes propiedades del formulario:

|  |  |  |  |
| --- | --- | --- | --- |
| Control | Propiedad | Valor | |
| Form1 | (Name) | | frmUno |
|  | BackColor | | RoyalBlue [Web] |
|  | ControlBox | | False |
|  | FormBorderStyle | | Fixed3D |
|  | StartPosition | | CenterScreen |
|  | Text | | Formulario Uno |

Paso 3: Ejecutamos la aplicación, obtenemos:



Podemos validar que se cumplen las propiedades que cambiamos a este formulario.

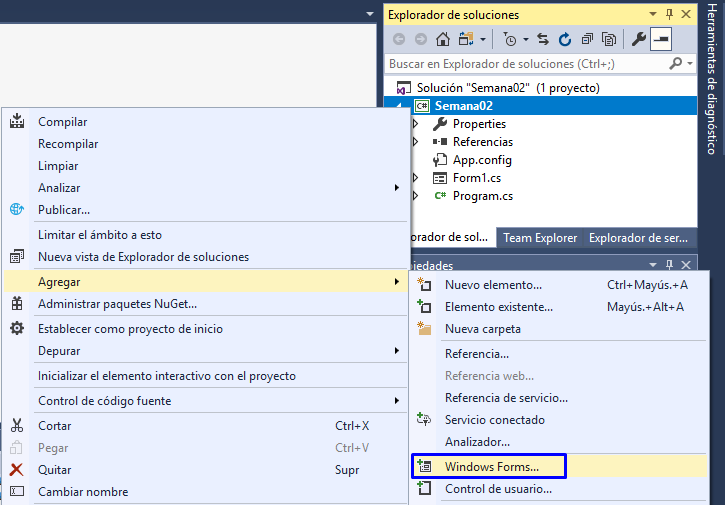
Paso 4: Cerramos la aplicación.

A través de la GUI no existe forma de cerrar la aplicación, entonces hacemos uso de una combinación de dos teclas para cerrarlo: Alt+F4. De esta forma se puede cerrar cualquier aplicación que se ejecuta sobre Windows.

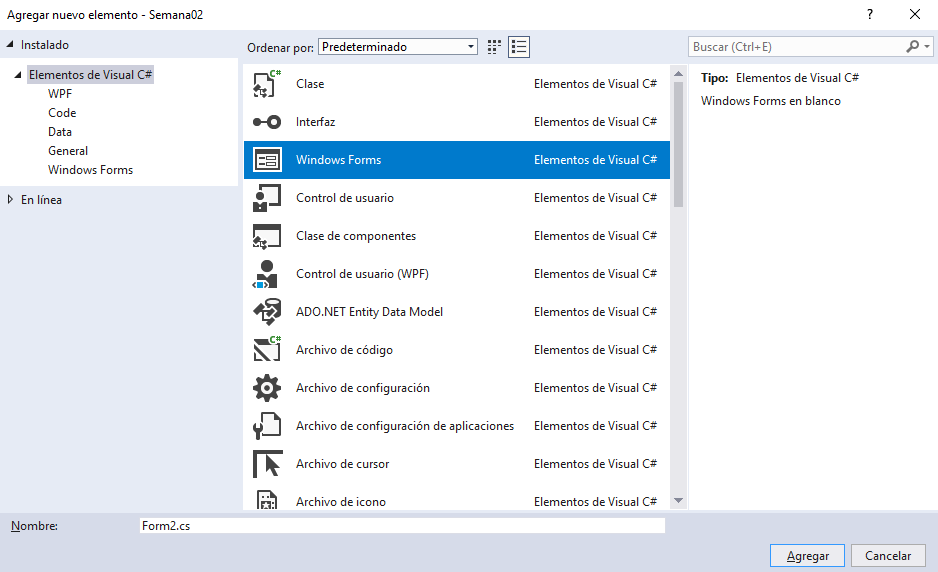
Paso 5: Cambiar las propiedades mediante la edición de código.

Para aplicar las mismas propiedades a través de la edición de código agregamos un nuevo formulario, para ello hacemos lo siguiente:

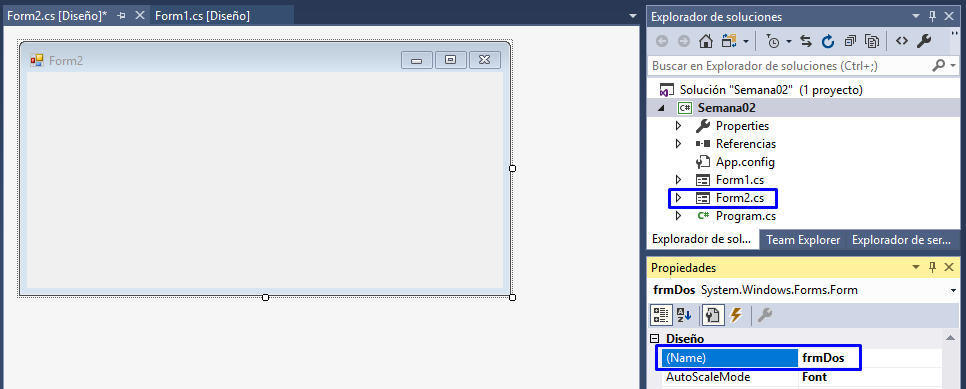
1. Seleccionamos el proyecto en el panel Explorador de soluciones, hacemos click derecho para que se muestre el menú desplegable y seleccionamos: Agregar > Windows Forms…



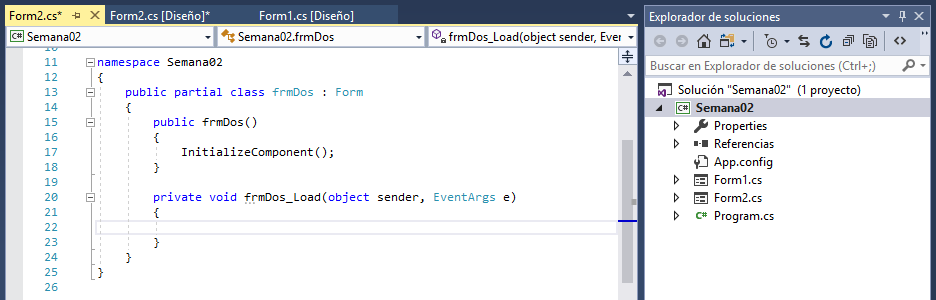
En la ventana emergente le damos las opciones que se muestran y presionamos en el botón Aceptar



En diseño solo cambiaremos la propiedad Name, le daremos el valor “frmDos”.



Para editar código del formulario hacemos doble click en cualquier parte dentro del diseño del formulario, obtenemos:



La función Load() se ejecuta cada vez que se carga un formulario, una vez ejecutado el código que está dentro de la función recién se muestra el formulario en pantalla.

Es ideal entonces que cambiemos por código las propiedades del nuevo formulario dentro de la función Load(). Para ello modificamos el código de acuerdo a:

namespace Semana02

{

public partial class frmDos : Form

{

public frmDos()

{

InitializeComponent();

}

private void frmDos\_Load(object sender, EventArgs e)

{

this.BackColor = Color.RoyalBlue;

this.ControlBox = false;

this.FormBorderStyle = FormBorderStyle.Fixed3D;

this.StartPosition = FormStartPosition.CenterScreen;

this.Text = "Formulario Dos";

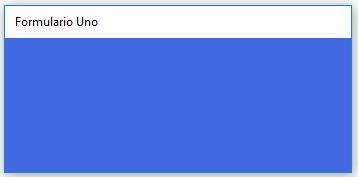
}

}

}

La sentencia this identifica al formulario que estamos editando, entonces las propiedades que estamos modificando corresponden al formulario frmDos.

1. Ejecutamos la aplicación.

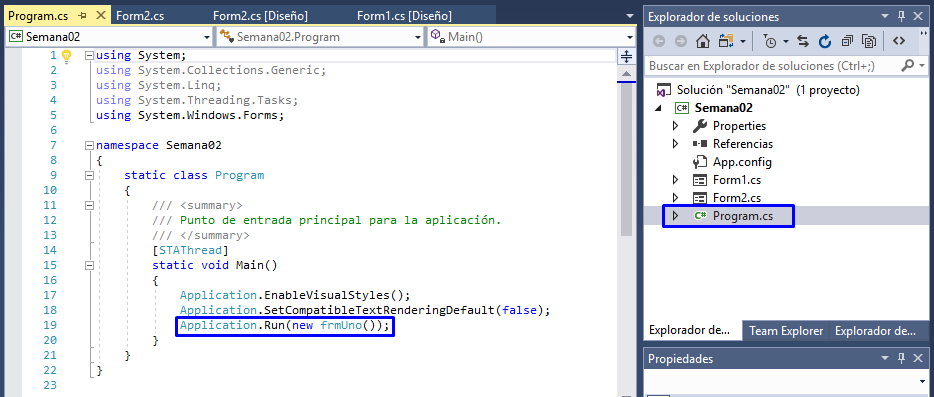


Observamos que no se muestra el Formulario Dos sino el Formulario Uno.

Culminamos la ejecución con Alt+F4.

Para seleccionar que formulario mostrar cuando se inicie la ejecución de la aplicación debemos modificar el código que está en el archivo Program.cs. Cabe indicar que cuando se ejecuta la aplicación, el primer código que se ejecuta es el código contenido en este archivo y luego el código del formulario que se indique.

1. Para editar este archivo hacemos doble click en el archivo Program.cs que se muestra en el panel Explorador de soluciones.



1. Lo modificamos de acuerdo a:

namespace Semana02

{

static class Program

{

/// <summary>

/// Punto de entrada principal para la aplicación.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

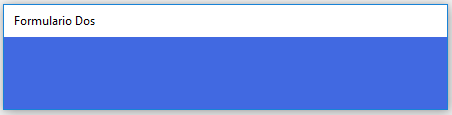
Application.Run(new frmDos());

}

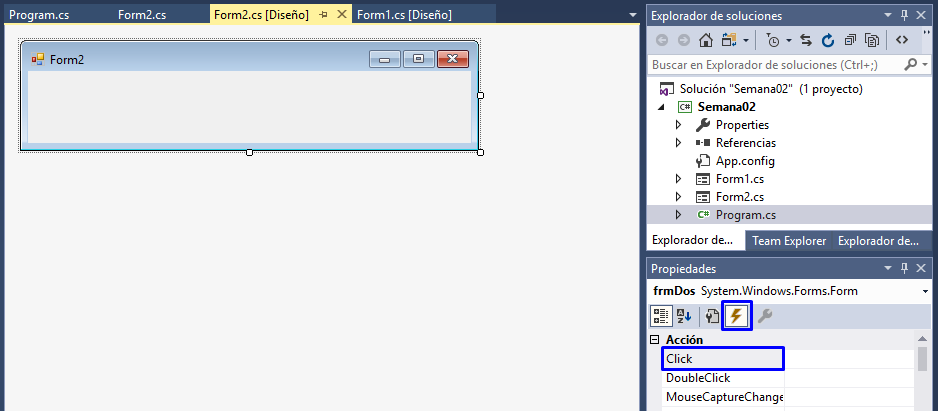
}

}

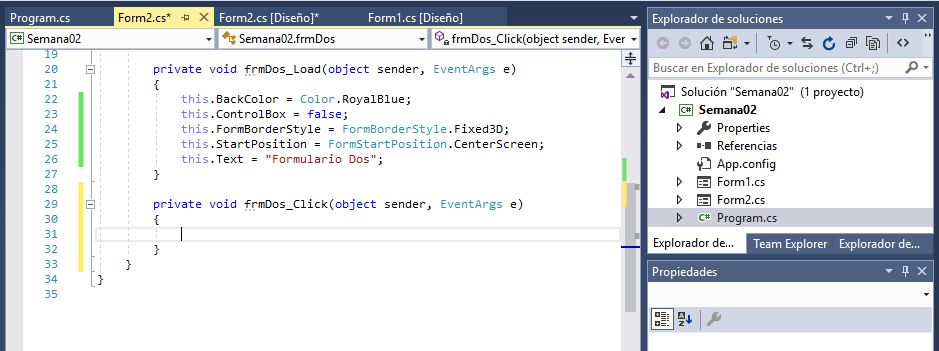
1. Ahora ejecutamos la aplicación y observamos que se muestra el Formulario Dos, también podemos validar las propiedades que cambiamos por código.



Ahora vamos a agregar código al evento Click del nuevo formulario, para ello en el panel Propiedades hacemos doble click en la Acción Click.



1. Obtenemos:



1. Entonces, la función que se ejecutara cuando se dispare el evento Click del formulario frmDos será la función frmDos\_Click(), modificamos el código de esta función de acuerdo a:

namespace Semana02

{

public partial class frmDos : Form

{

public frmDos()

{

InitializeComponent();

}

private void frmDos\_Load(object sender, EventArgs e)

{

this.BackColor = Color.RoyalBlue;

this.ControlBox = false;

this.FormBorderStyle = FormBorderStyle.Fixed3D;

this.StartPosition = FormStartPosition.CenterScreen;

this.Text = "Formulario Dos";

}

private void frmDos\_Click(object sender, EventArgs e)

{

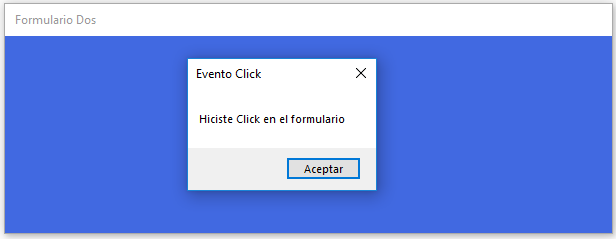
MessageBox.Show("Hiciste Click en el formulario","Evento Click");

}

}

}

1. Ejecutamos la aplicación y hacemos click dentro del formulario, obtenemos:



1. Presionamos el botón Aceptar y cerramos la aplicación.

2.2 Trabajar con controles LABEL.

Objetivos

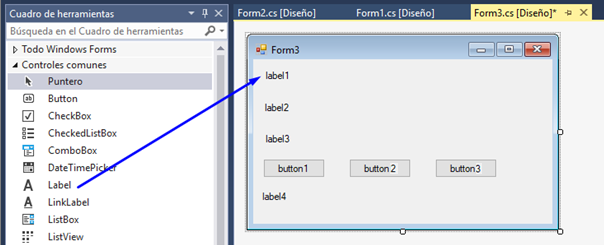
* Desarrollar aplicaciones utilizando etiquetas, modificar sus propiedades y entender sus eventos.

Introducción

Durante esta actividad, entenderá el uso de etiquetas en los formularios de Windows.

Paso 1: En el proyecto Semana02 agregamos un nuevo formulario y le ponemos como propiedad Name el valor “frmTres”.

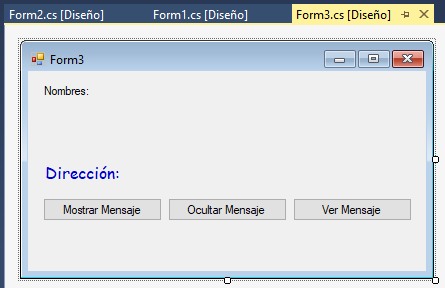
Paso 2: En este formulario agregamos cuatro etiquetas y tres botones.



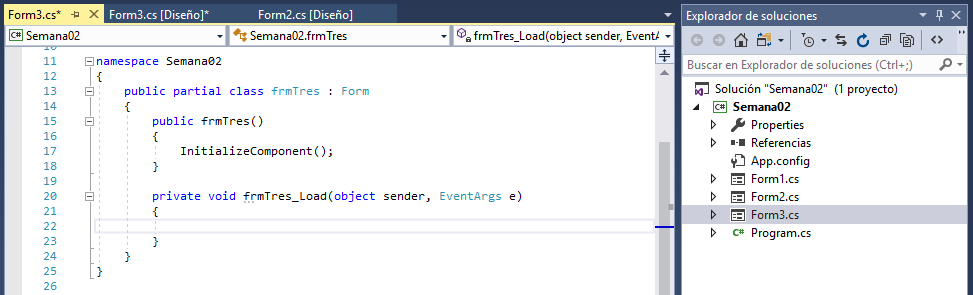
Paso 3: Cambiamos las propiedades de los controles de acuerdo a:

|  |  |  |  |
| --- | --- | --- | --- |
| Control | Propiedad | Valor | |
| Label1 | (Name) | | lblUno |
|  | Text | | Nombres: |
| Label2 | (Name) | | lblDos |
|  | Text | | [vacío] |
| Label3 | (Name) | | lblTres |
|  | Font > Name | | Comic Sans MS |
|  | Font > Size | | 12 |
|  | ForeColor | | MediumBlue [Web] |
|  | Text | | Dirección: |
| Label4 | (Name) | | lblCuatro |
|  | ForeColor | | red [Web] |
|  | Text | | [vacío] |
| Button1 | (Name) | | btnMostrarMensaje |
|  | Text | | Mostrar Mensaje |
| Button2 | (Name) | | btnOcultarMensaje |
|  | Text | | Ocultar Mensaje |
| Button3 | (Name) | | btnVerMensaje |
|  | Text | | Ver Mensaje |

El diseño debe quedar de esta forma:



Paso 4: Para agregar código al evento Load del formulario hacemos doble click dentro del formulario.



Modificamos el código de acuerdo a:

namespace Semana02

{

public partial class frmTres : Form

{

public frmTres()

{

InitializeComponent();

}

private void frmTres\_Load(object sender, EventArgs e)

{

this.Text = "Formulario Tres";

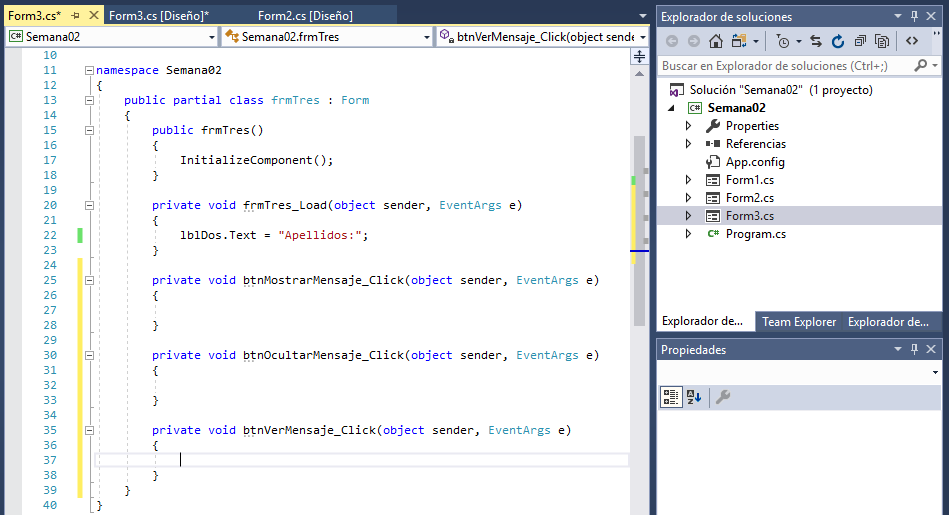
lblDos.Text = "Apellidos:";

}

}

}

1. Para agregar código al evento Click de cada botón haremos doble click en cada uno de ellos.



1. Modificamos el código de acuerdo a:

namespace Semana02

{

public partial class frmTres : Form

{

public frmTres()

{

InitializeComponent();

}

private void frmTres\_Load(object sender, EventArgs e)

{

this.Text = "Formulario Tres";

lblDos.Text = "Apellidos:";

}

private void btnMostrarMensaje\_Click(object sender, EventArgs e)

{

lblCuatro.ForeColor = Color.Red;

lblCuatro.Text = "El campo 'Apellidos' es un dato obligatorio.";

}

private void btnOcultarMensaje\_Click(object sender, EventArgs e)

{

lblCuatro.Visible = false;

}

private void btnVerMensaje\_Click(object sender, EventArgs e)

{

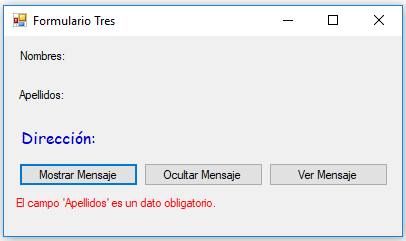
lblCuatro.Visible = true;

}

}

}

1. Modificamos el archivo Program.cs para que al ejecutar se muestre el formulario frmTres.
2. Ejecutamos la aplicación.



**2.3 Trabajar con controles TEXTBOX.**

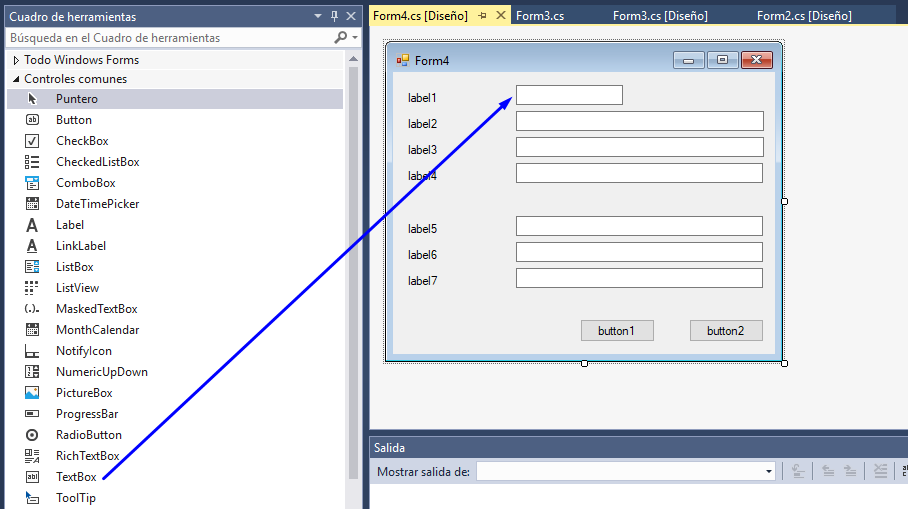
**Objetivos**

* Desarrollar aplicaciones utilizando cuadros de texto, modificar sus propiedades y entender sus eventos.

**Introducción**

* Durante esta actividad, entenderá el uso de cuadros de texto en los formularios de Windows.

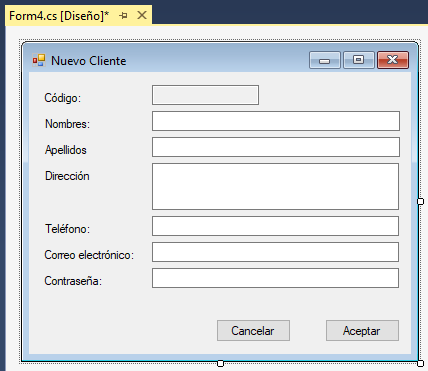
Paso 1: En el proyecto Semana02 agregamos un nuevo formulario y agregamos los controles de acuerdo a la siguiente figura.



Paso 2: Cambiamos las propiedades de los controles de acuerdo a:

|  |  |  |  |
| --- | --- | --- | --- |
| Control | Propiedad | Valor | |
| Form4 | (Name) | | frmCuatro |
|  | StartPosition | | CenterScreen |
|  | Text | | Nuevo Cliente |
| Label1 | (Name) | | Label1 |
|  | Text | | Código: |
| Label2 | (Name) | | Label2 |
|  | Text | | Nombres: |
| Label3 | (Name) | | Label3 |
|  | Text | | Apellidos: |
| Label4 | (Name) | | Label4 |
|  | Text | | Dirección: |
| Label5 | (Name) | | Label5 |
|  | Text | | Teléfono: |
| Label6 | (Name) | | Label6 |
|  | Text | | Correo electrónico: |
| Label7 | (Name) | | Label7 |
|  | Text | | Contraseña: |
| TextBox1 | (Name) | | txbCodigo |
|  | ReadOnly | | True |
|  | TabIndex | | 9 |
| TextBox2 | (Name) | | txbNombres |
|  | TabIndex | | 1 |
| TextBox3 | (Name) | | txbApellidos |
|  | TabIndex | | 2 |
| TextBox4 | (Name) | | txbDireccion |
|  | Multiline | | True |
|  | TabIndex | | 3 |
| TextBox5 | (Name) | | txbTelefono |
|  | MaxLength | | 14 |
|  | TabIndex | | 4 |
| TextBox6 | (Name) | | txbEmail |
|  | TabIndex | | 5 |
| TextBox7 | (Name) | | txbPassword |
|  | PasswordChar | | ? |
|  | TabIndex | | 6 |
| Button1 | (Name) | | btnCancelar |
|  | Text | | Cancelar |
|  | TabIndex | | 7 |
| Button2 | (Name) | | btnAceptar |
|  | Text | | Aceptar |
|  | TabIndex | | 8 |

Paso 3: El diseño debe quedar de esta forma:



Paso 4: Agregamos código en el evento Load del formulario de acuerdo a:

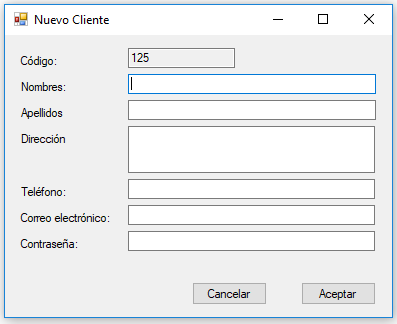
private void frmCuatro\_Load(object sender, EventArgs e)

{

txbCodigo.Text = "125";

}

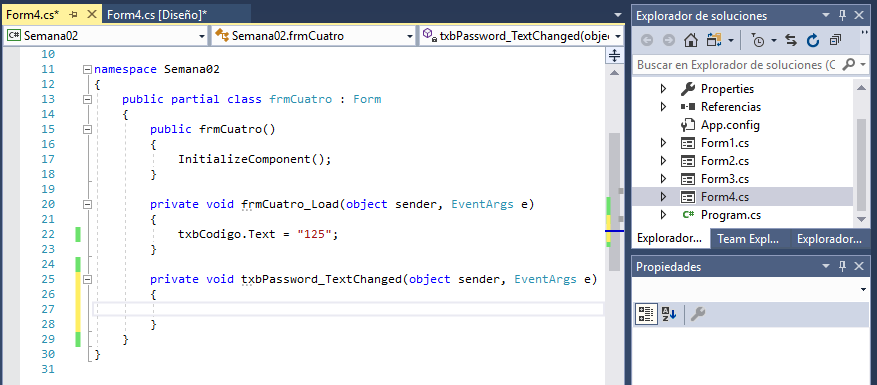
1. Modificamos el archivo Program.cs para que al ejecutar se muestre el formulario frmCuatro.
2. Ejecutamos la aplicación.



1. Comprobamos que el cuadro de texto txbCodigo ya tiene el valor 125 al mostrar el formulario.

También vemos que el cuadro de texto txbNombres tiene el foco inicialmente, esto se debe a que la propiedad TabIndex de este control es 1. Si pulsamos la tecla Tab veremos que el foco pasa de un control a otro de acuerdo al orden que les pusimos en la propiedad TabIndex de cada control.

1. Vamos a trabajar con el evento TextChanged del cuadro de texto txbPassword, para ingresar código a este evento, en diseño, hacemos doble click en el cuadro de texto txbPassword, obtenemos:



1. Agregamos código al evento TextChanged de acuerdo a:

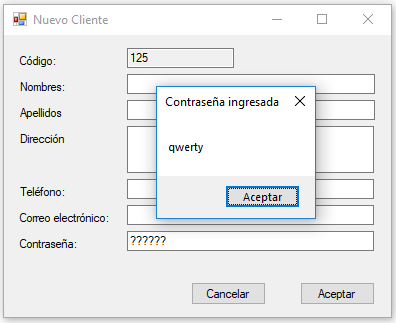
private void txbPassword\_TextChanged(object sender, EventArgs e)

{

MessageBox.Show(txbPassword.Text, "Contraseña ingresada");

}

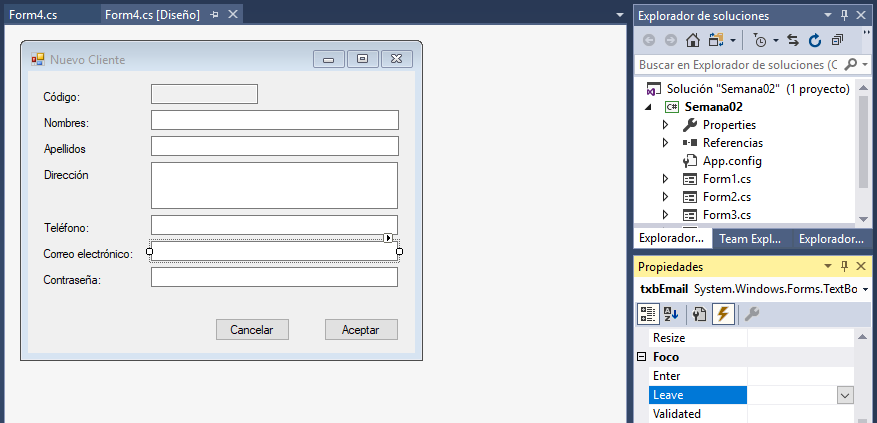
1. Ejecutamos la aplicación.



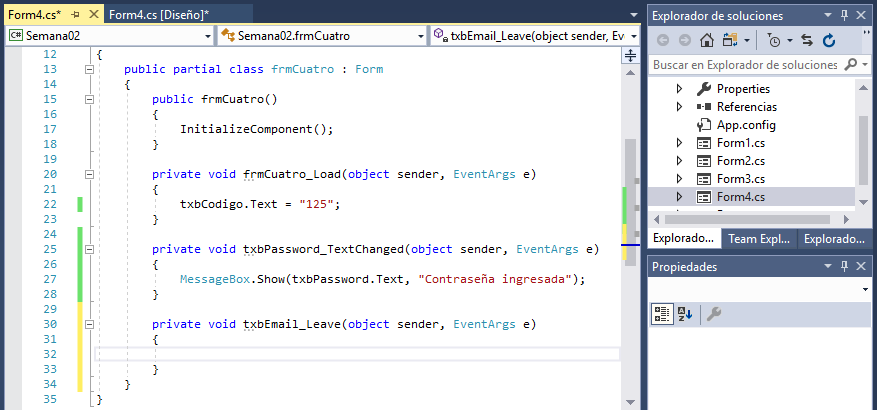
Observamos que el evento se ejecuta cada vez que agregamos o borramos un caracter en el cuadro de texto.

Paso 5: Vamos a trabajar con el evento Leave del cuadro de texto txbEmail, para ingresar código a este evento, en diseño no debemos hacer doble click en el cuadro de texto txbEmail ya que se creará la función para el evento TextChanged.Lo que debemos hacer es:

1. Seleccionar el control txbEmail
2. En la ventana Propiedades seleccionamos el icono de eventos 
3. Hacemos doble click en el evento Leave



1. Obtenemos:



1. Agregamos código al evento Leave de acuerdo a:

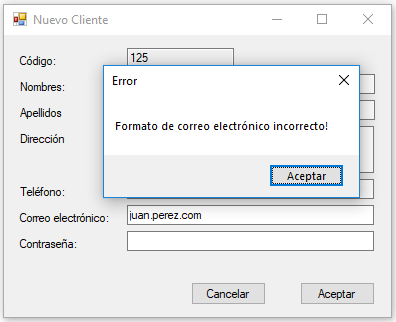
private void txbEmail\_Leave(object sender, EventArgs e)

{

MessageBox.Show("Formato de correo electrónico incorrecto!", "Error");

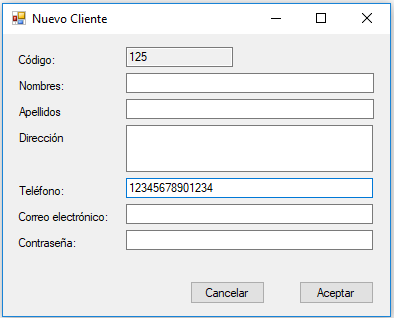
}

1. Ejecutamos la aplicación.



Observamos que el evento se ejecuta cada vez que quitamos el foco del cuadro de texto de texto.

1. En el cuadro de texto txbTelefono podemos validar que no podemos escribir más de 14 caracteres, esto se debe al valor de la propiedad MaxLength de este cuadro de texto (14).



**2.4 Trabajar con controles BUTTON.**

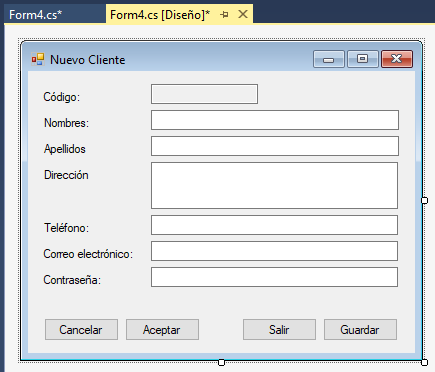
**Objetivo**

* Desarrollar aplicaciones utilizando botones, modificar sus propiedades y entender sus eventos.

**Introducción**

Durante esta actividad, entenderá el uso de botones en los formularios de Windows.

Paso1: Vamos a seguir trabajando en el formulario frmCuatro del proyecto Semana02.Agregamos dos botones: Salir y Guardar.



Paso 2: Agregamos las siguientes propiedades:

|  |  |  |  |
| --- | --- | --- | --- |
| Control | Propiedad | Valor | |
| frmCuatro | CancelButton | | btnCancelar |
|  | [AcceptButton](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.form.acceptbutton?view=netframework-4.8) | | btnAceptar |
| Button3 | (Name) | | btnSalir |
|  | Text | | Salir |
| Button4 | (Name) | | btnGuardar |
|  | Text | | Guardar |

Paso 3: Agregamos código en el evento Click de los botones btnCancelar y btnAceptar de acuerdo a:

private void btnCancelar\_Click(object sender, EventArgs e)

{

MessageBox.Show("Presionó el botón 'Cancelar'", "Cancelar");

}

private void btnAceptar\_Click(object sender, EventArgs e)

{

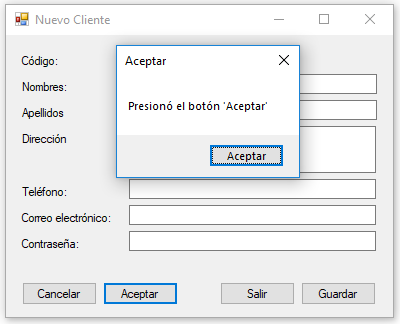
MessageBox.Show("Presionó el botón 'Aceptar'", "Aceptar");

}

Ejecutamos la aplicación.

Verificamos en el formulario que cuando presionamos la tecla ESC se obtiene la misma reacción que como si hubiéramos hecho click en el botón Cancelar.

1. Verificamos en el formulario que cuando presionamos la tecla Enter se obtiene la misma reacción que como si hubiéramos hecho click en el botón Aceptar.



1. Agregamos código en el evento Click de los botones btnSalir y btnGuardar de acuerdo a:

private void btnSalir\_Click(object sender, EventArgs e)

{

this.Close();

}

private void btnGuardar\_Click(object sender, EventArgs e)

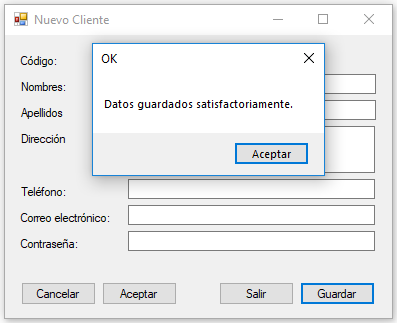
{

MessageBox.Show("Datos guardados satisfactoriamente.", "OK");

this.Close();

}

1. Comprobamos que se cierra el formulario cuando presionamos el botón Salir y también el botón Guardar, en este último caso previamente se muestra un mensaje.



**Actividad:**

* Ingresa a la plataforma virtual, revisa los siguientes enlaces luego realiza las actividades propuestas
* Form Class. Propiedades, métodos y eventos de un formulario

<https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.form?view=netframework-4.8>

* Introducción a Windows Form

https://www.youtube.com/watch?v=mFMTQyOQrqA&list=PLhbcXfA7vHX64kzwPvdHGOsiPh9q3IOhF&index=22

* Propiedades de una Ventana

https://www.youtube.com/watch?v=Iu2xk3NMtHU&list=PLhbcXfA7vHX64kzwPvdHGOsiPh9q3IOhF&index=23

* Diseñando una ventana de Login

<https://www.youtube.com/watch?v=tGnZmoPtjDo&list=PLhbcXfA7vHX64kzwPvdHGOsiPh9q3IOhF&index=24>

* Label Class. Propiedades, métodos y eventos de una etiqueta

<https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.label?view=netframework-4.8>

* TextBox Class. Propiedades, métodos y eventos de un cuadro de texto

<https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.textbox?view=netframework-4.8>

* Button Class. Propiedades, métodos y eventos de un botón

<https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.button?view=netframework-4.8>

1. Por cada tipo de control hacer ejemplos con 3 propiedades que no se han explicado en el laboratorio.
2. Por cada tipo de control hacer ejemplos con 2 eventos de no se han explicado en el laboratorio.