Herramientas de Programación 1

Tema Nº4:

Cuadros de Diálogo

Indicador de logro Nº4:Realiza aplicaciones en Windows con cuadros de diálogo personalizados y gestiona los eventos del teclado cumpliendo con los requisitos de diseño del usuario

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº4:**

CUADROS DE DIÁLOGO – FORMULARIOS MODALES Y NO MODALES

Cuadros de Diálogo Personalizados

A la hora de interactuar con los usuarios, la programación nos permite varias opciones, la más común y popular es mediante los Mensajes de Dialogo.

Estos mensajes se pueden personalizar de tal forma que podemos mostrar distintos botones y agregar un icono de acuerdo al mensaje que queremos mostrar.

Formularios Modales y No Modales

Cuando mostramos formularios lo podemos hacer de dos formas, la elección depende de la funcionalidad que queremos darle al formulario.

Si queremos que un formulario no permita interactuar con otro hasta que cumpla su funcionalidad lo mostraremos como Modal.

Si queremos que el formulario permita la interacción con otros debemos mostrarlo como No Modal.

Eventos del Teclado

Existen circunstancias donde es necesario capturar algún evento del teclado para ejecutar cierta funcionalidad, veremos cómo capturar estos eventos y ejecutar cierto código.

De los eventos de un teclado destacan dos por su utilidad:

|  |  |
| --- | --- |
| Evento | Descripción |
| KeyPress | Tiene lugar cuando el control tiene el foco y el usuario presiona y suelta una tecla. |
| KeyDown | Tiene lugar cuando se presiona una tecla. |

Ambos se disparan cuando se presiona una tecla, sin embargo cada evento nos proporcionan información distinta de la(s) teclas(s) presionada(s). Lo entenderemos con los ejemplos que se explican más adelante.

**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 4:**

4.1 Mostrar cuadros de dialogo personalizados.

**Objetivos**

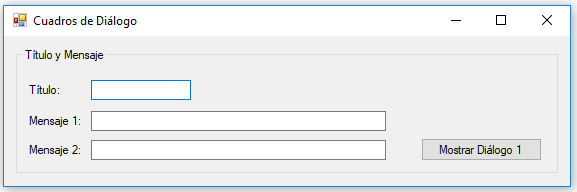
* Desarrollar aplicaciones utilizando cuadros de dialogo personalizados, modificar sus propiedades y entender sus eventos.

**Introducción**

Durante esta actividad, podrá configurar y mostrar cuadros de dialogo personalizados en las aplicaciones de escritorio.

Paso 1. Título y Mensaje.

1. Para mostrar el cuadro de diálogo más sencillo, pero personalizado, vamos a crear el proyecto Semana04, al formulario le damos el siguiente diseño:



|  |  |  |  |
| --- | --- | --- | --- |
| Control | Propiedad | Valor | |
| Form1 | (Name) | | frmUno |
|  | Text | | Cuadros de Diálogo |
| GroupBox1 | (Name) | | GroupBox1 |
|  | Text | | Título y Mensaje |
| Label1 | (Name) | | Label1 |
|  | Text | | Título: |
| Label2 | (Name) | | Label2 |
|  | Text | | Mensaje: |
| TextBox1 | (Name) | | txbTitulo |
| TextBox2 | (Name) | | txbMensaje1 |
| TextBox3 | (Name) | | txbMensaje2 |
| Button1 | (Name) | | btnMostrarDialogo1 |
|  | Text | | Mostrar Diálogo 1 |

1. Agregamos código al evento Click del control btnMostrarDialogo1 de acuerdo a:

private void btnMostrarDialogo1\_Click(object sender, EventArgs e)

{

String strTitulo = txbTitulo.Text;

String strMensaje = txbMensaje1.Text + "\n" + txbMensaje2.Text;

MessageBox.Show(strMensaje, strTitulo);

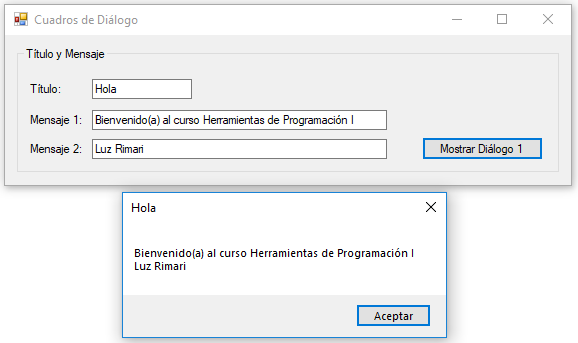
}

En este ejemplo vamos empezar a utilizar variables, strTitulo y strMensaje son las variables.

En C# una variable debe tener un tipo específico, de todos los tipos vamos a usar los tipos String, este tipo identifica a las cadenas de caracteres (o texto).

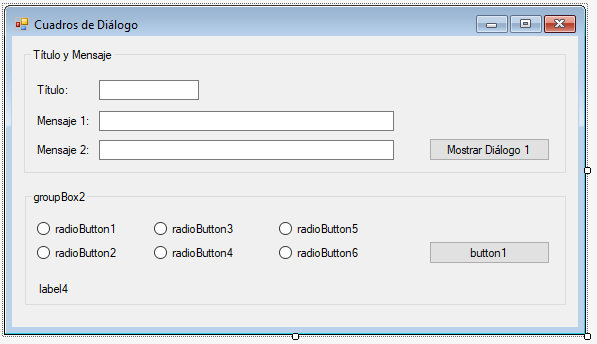
El valor de la variable strMensaje es la unión (concatenación) de tres cadenas de texto, lo que hace la cadena de texto “\n” (salto de línea) es mostrar en una línea aparte el texto que viene después del salto de línea.

1. Ejecutamos la aplicación.



Paso 2: Botones

1. Podemos agregar distintos juegos de botones al cuadro de dialogo, para mostrar un ejemplo de ello agregamos al formulario los controles del GroupBox2:



|  |  |  |  |
| --- | --- | --- | --- |
| Control | Propiedad | Valor | |
| GroupBox2 | (Name) | | GroupBox2 |
|  | Text | | Botones |
| RadioButton1 | (Name) | | rbtOk |
|  | Text | | Ok |
| RadioButton2 | (Name) | | rbtOkCancel |
|  | Text | | Ok Cancel |
| RadioButton3 | (Name) | | rbtYesNo |
|  | Text | | Yes No |
| RadioButton4 | (Name) | | rbtYesNoCancel |
|  | Text | | Yes No Cancel |
| RadioButton5 | (Name) | | rbtRetryCancel |
|  | Text | | Retry Cancel |
| RadioButton6 | (Name) | | rbtAbortRetryIgnore |
|  | Text | | Abort Retry Ignore |
| Label4 | (Name) | | lblResultado |
|  | Text | | … |
|  | ForeColor | | Blue [Web] |
| Button1 | (Name) | | btnMostrarDialogo2 |
|  | Text | | Mostrar Diálogo 2 |

1. Primero vamos agregar código al evento Load del formulario de acuerdo a:

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

{

txbTitulo.Text = "Hola";

txbMensaje1.Text = "Bienvenido Juan Perez al curso:";

txbMensaje2.Text = "Herramientas de Programación I";

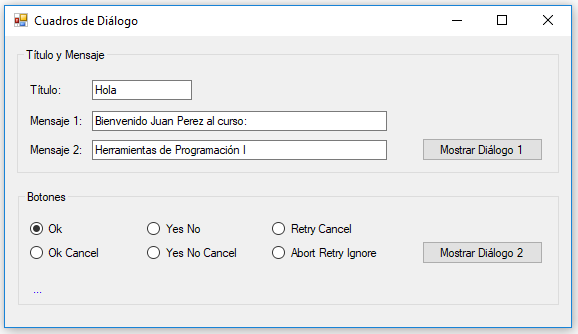
rbtOk.Select();

//Tambin se puede usar:

//rbtOk.Checked = true;

}

1. Ejecutamos la aplicación.



1. Para mostrar los distintos juegos de botones de un cuadro de diálogo vamos agregar el siguiente código en el evento Click del botón btnMostrarDialogo2:

private void btnMostrarDialogo2\_Click(object sender, EventArgs e)

{

String strTitulo = txbTitulo.Text;

String strMensaje = txbMensaje1.Text + "\n" + txbMensaje2.Text;

if(rbtOk.Checked)

{

MessageBox.Show(strMensaje, strTitulo, MessageBoxButtons.OK);

}

if (rbtOkCancel.Checked)

{

MessageBox.Show(strMensaje, strTitulo, MessageBoxButtons.OKCancel);

}

if (rbtYesNo.Checked)

{

MessageBox.Show(strMensaje, strTitulo, MessageBoxButtons.YesNo);

}

if (rbtYesNoCancel.Checked)

{

MessageBox.Show(strMensaje, strTitulo, MessageBoxButtons.YesNoCancel);

}

if (rbtRetryCancel.Checked)

{

MessageBox.Show(strMensaje, strTitulo, MessageBoxButtons.RetryCancel);

}

if (rbtAbortRetryIgnore.Checked)

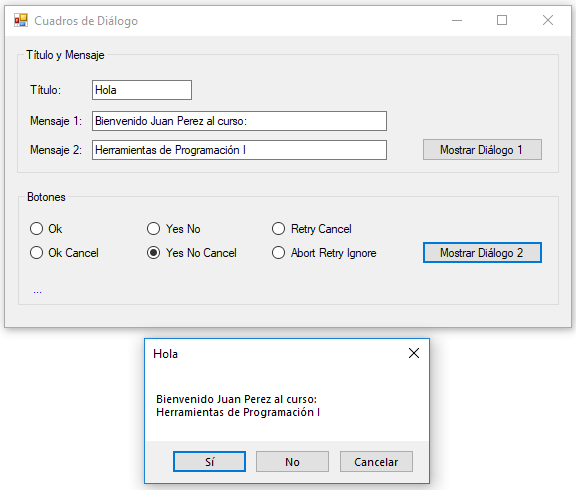
{

MessageBox.Show(strMensaje, strTitulo, MessageBoxButtons.AbortRetryIgnore);

}

}

1. Ejecutamos la aplicación.



Validamos que en el cuadro de dialogo se muestra el juego de botones que elegimos en los RadioButtons.

Generalmente es útil identificar cuál de los botones se ha presionado, para de acuerdo a ello ejecutar alguna tarea específica.

1. En el siguiente ejemplo identificaremos el botón que se ha presionado, para ello modificamos el código en el evento Click del botón btnMostrarDialogo2 de acuerdo a:

private void btnMostrarDialogo2\_Click(object sender, EventArgs e)

{

String strTitulo = txbTitulo.Text;

String strMensaje = txbMensaje1.Text + "\n" + txbMensaje2.Text;

DialogResult drBoton = DialogResult.OK;

lblResultado.Text = "";

if (rbtOk.Checked)

{

drBoton = MessageBox.Show(strMensaje, strTitulo, MessageBoxButtons.OK);

}

if (rbtOkCancel.Checked)

{

drBoton = MessageBox.Show(strMensaje, strTitulo, MessageBoxButtons.OKCancel);

}

if (rbtYesNo.Checked)

{

drBoton = MessageBox.Show(strMensaje, strTitulo, MessageBoxButtons.YesNo);

}

if (rbtYesNoCancel.Checked)

{

drBoton = MessageBox.Show(strMensaje, strTitulo, MessageBoxButtons.YesNoCancel);

}

if (rbtRetryCancel.Checked)

{

drBoton = MessageBox.Show(strMensaje, strTitulo, MessageBoxButtons.RetryCancel);

}

if (rbtAbortRetryIgnore.Checked)

{

drBoton = MessageBox.Show(strMensaje, strTitulo, MessageBoxButtons.AbortRetryIgnore);

}

//Evaluamos el botón que presionó el usuario

if (drBoton == DialogResult.OK)

{

lblResultado.Text = "Presionó el botón 'Aceptar'";

}

if (drBoton == DialogResult.Cancel)

{

lblResultado.Text = "Presionó el botón 'Cancelar'";

}

if (drBoton == DialogResult.Yes)

{

lblResultado.Text = "Presionó el botón 'Sí'";

}

if (drBoton == DialogResult.No)

{

lblResultado.Text = "Presionó el botón 'No'";

}

if (drBoton == DialogResult.Retry)

{

lblResultado.Text = "Presionó el botón 'Reintentar'";

}

if (drBoton == DialogResult.Abort)

{

lblResultado.Text = "Presionó el botón 'Anular'";

}

if (drBoton == DialogResult.Ignore)

{

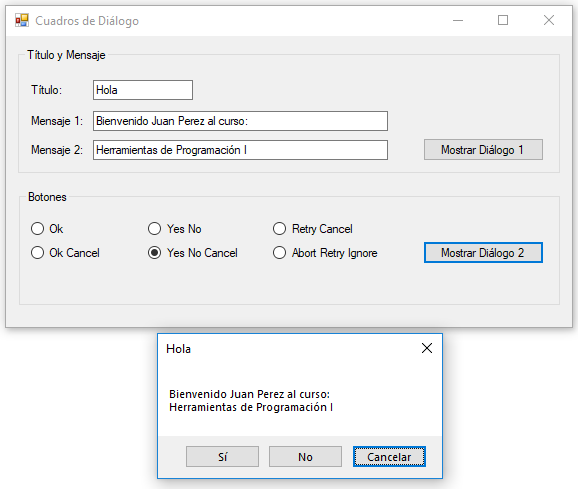
lblResultado.Text = "Presionó el botón 'Omitir'";

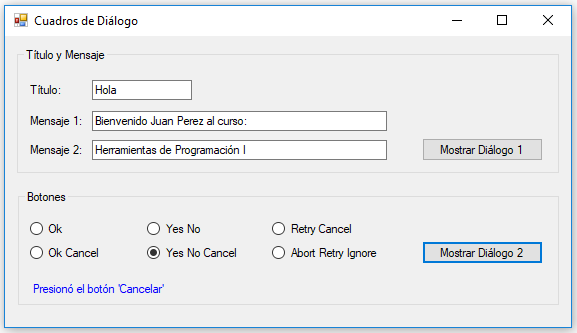
}

}

Inicialmente declaramos la variable drBoton del tipo DialogResult, el resultado de la ejecución de la función MessageBox.Show( ) indicará que botón ha presionado el usuario. Una vez identificado el botón presionado pasamos a ejecutar el código asociado a cada botón.

1. Ejecutamos la aplicación.





Paso 3: Iconos

Es posible también mostrar iconos en los diálogos, para ello vamos agregar en el cuarto parámetro de la función MesasgeBox.Show() el icono que queremos mostrar.

1. Modificamos el código del evento Click del botón btnMostrarDialogo2 de acuerdo a:

private void btnMostrarDialogo2\_Click(object sender, EventArgs e)

{

String strTitulo = txbTitulo.Text;

String strMensaje = txbMensaje1.Text + "\n" + txbMensaje2.Text;

DialogResult drBoton = DialogResult.OK;

lblResultado.Text = "";

if (rbtOk.Checked)

{

drBoton = MessageBox.Show(strMensaje, strTitulo, MessageBoxButtons.OK, MessageBoxIcon.Information);

}

if (rbtOkCancel.Checked)

{

drBoton = MessageBox.Show(strMensaje, strTitulo, MessageBoxButtons.OKCancel, MessageBoxIcon.Question);

}

if (rbtYesNo.Checked)

{

drBoton = MessageBox.Show(strMensaje, strTitulo, MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation);

}

if (rbtYesNoCancel.Checked)

{

drBoton = MessageBox.Show(strMensaje, strTitulo, MessageBoxButtons.YesNoCancel, MessageBoxIcon.Error);

}

if (rbtRetryCancel.Checked)

{

drBoton = MessageBox.Show(strMensaje, strTitulo, MessageBoxButtons.RetryCancel, MessageBoxIcon.Stop);

}

if (rbtAbortRetryIgnore.Checked)

{

drBoton = MessageBox.Show(strMensaje, strTitulo, MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Warning);

}

//Evaluamos el botón que presionó el usuario

if (drBoton == DialogResult.OK)

{

lblResultado.Text = "Presionó el botón 'Aceptar'";

}

if (drBoton == DialogResult.Cancel)

{

lblResultado.Text = "Presionó el botón 'Cancelar'";

}

if (drBoton == DialogResult.Yes)

{

lblResultado.Text = "Presionó el botón 'Sí'";

}

if (drBoton == DialogResult.No)

{

lblResultado.Text = "Presionó el botón 'No'";

}

if (drBoton == DialogResult.Retry)

{

lblResultado.Text = "Presionó el botón 'Reintentar'";

}

if (drBoton == DialogResult.Abort)

{

lblResultado.Text = "Presionó el botón 'Anular'";

}

if (drBoton == DialogResult.Ignore)

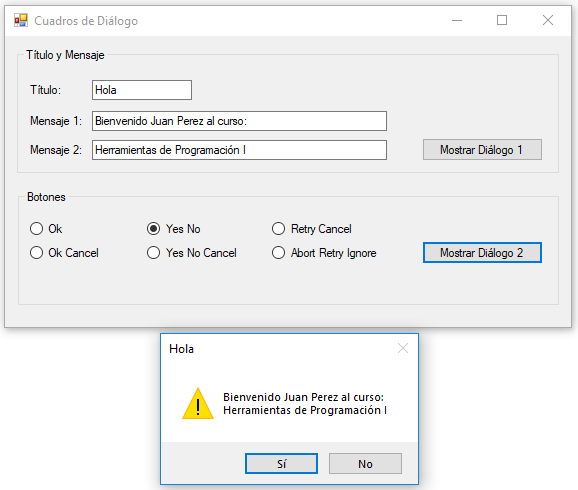
{

lblResultado.Text = "Presionó el botón 'Omitir'";

}

}

1. Ejecutamos la aplicación.



4.2 Formularios Modales y No Modales.

Objetivos

* Desarrollar aplicaciones mostrando formularios en forma Modal y No Modal.

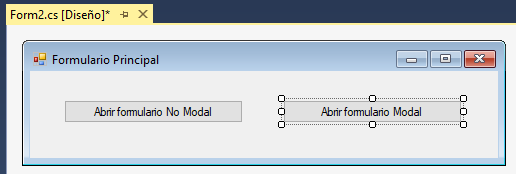
Introducción

Durante esta actividad, podrá mostrar formularios Modales y No Modales, además de comprender la diferencia en el uso de cada uno de ellos.

En el proyecto Semana04 vamos a crear tres formularios, el primero será un formulario a partir del cual abriremos los otros dos formularios, uno de forma Modal y otro de forma No Modal.

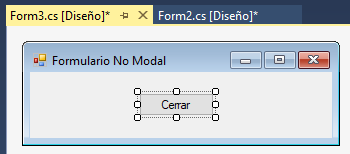
Paso 1. Formulario No Modal.

1. Creamos el formulario frmPrincipal con el siguiente diseño:



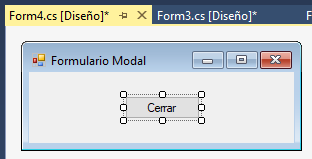
|  |  |  |  |
| --- | --- | --- | --- |
| Control | Propiedad | Valor | |
| Form1 | (Name) | | frmPrincipal |
|  | Text | | Formulario Principal |
|  | StartPosition | | CenterScreen |
| Button1 | (Name) | | btnAbrirNoModal |
|  | Text | | Abrir formulario No Modal |
| Button2 | (Name) | | btnAbrirModal |
|  | Text | | Abrir formulario Modal |

1. Creamos el formulario frmNoModal con el siguiente diseño:



|  |  |  |  |
| --- | --- | --- | --- |
| Control | Propiedad | Valor | |
| Form1 | (Name) | | frmNoModal |
|  | Text | | Formulario No Modal |
|  | StartPosition | | CenterScreen |
| Button1 | (Name) | | btnCerrar |
|  | Text | | Cerrar |

1. Creamos el formulario frmModal con el siguiente diseño:



|  |  |  |  |
| --- | --- | --- | --- |
| Control | Propiedad | Valor | |
| Form1 | (Name) | | frmModal |
|  | Text | | Formulario Modal |
|  | StartPosition | | CenterScreen |
| Button1 | (Name) | | btnCerrar |
|  | Text | | Cerrar |

1. Agregamos el mismo código al evento Click de los dos últimos formularios de acuerdo a:

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

{

this.Close();

}

1. En el formulario frmPrincipal agregamos código al evento Click del botón btnAbrirNoModal

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

{

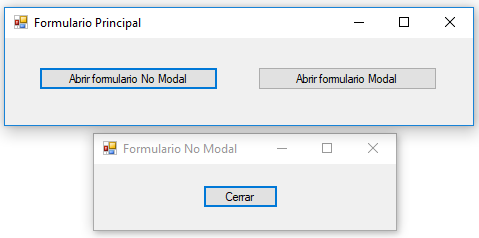
frmNoModal miFormularioNoModal = new frmNoModal();

miFormularioNoModal.Show();

}

Antes de abrir un formulario debemos crearlo y luego mostrarlo, cuando abrimos un formulario a través de la función Show() estamos abriendo el formulario de forma No Modal, esto indica que podemos acceder a cualquiera de los dos formularios en todo momento y trabajar con ellos.

1. Modificamos el archivo Program.cs para que muestre inicialmente el formulario frmPrincipal y ejecutamos la aplicación.



Podemos comprobar que una vez abierto el formulario No Modal podemos acceder al formulario Principal

Paso 2. Formulario Modal.

1. En el formulario frmPrincipal agregamos código al evento Click del botón btnAbrirModal

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

{

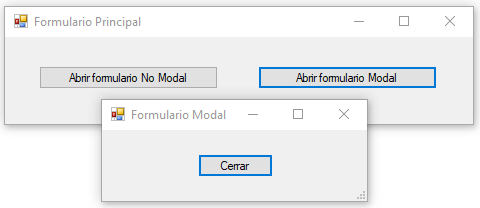
frmModal miFormularioModal = new frmModal();

miFormularioModal.ShowDialog();

}

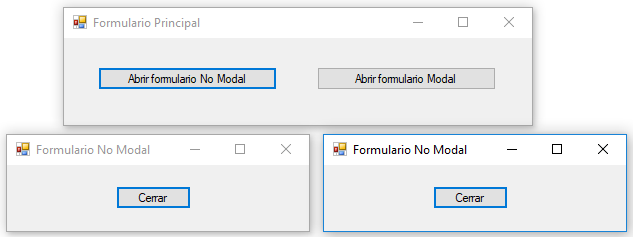
Cuando abrimos un formulario a través de la función ShowDialog() estamos abriendo el formulario de forma Modal, esto indica que no podemos acceder a ningún formulario de la aplicación hasta cerrar el formulario Modal abierto.

1. Ejecutamos la aplicación y abrimos el formulario Modal.



Una vez abierto el formulario Modal ya no podemos acceder al formulario principal hasta que primero cerremos el formulario Modal.

Ahora vamos a ver un caso especial con el formulario No Modal, una vez cerrado el formulario Modal vamos a presionar dos veces el botón que abre el formulario No Modal.



Notamos que podemos abrir la cantidad de formularios que queramos, ya que el código del botón No Modal hace eso, esto es, crear una instancia del formulario y mostrarla cada vez que se presiona el botón. En la siguiente semana vamos a trabajar con formularios MDI y es donde vamos a ver como mostrar solo una instancia de cada formulario.

4.3 Eventos del teclado.

Objetivos

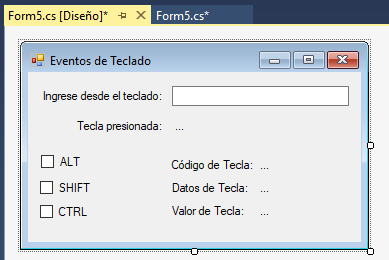
* Desarrollar aplicaciones que gestionen los eventos del teclado.

Introducción

Durante esta actividad, podrá identificar los eventos y los datos del teclado que se ha presionado.

Paso 1. Eventos del teclado.

1. Creamos el formulario frmEventosTeclado con el siguiente diseño:



|  |  |  |  |
| --- | --- | --- | --- |
| Control | Propiedad | Valor | |
| Form5 | (Name) | | frmEventosTeclado |
|  | Text | | Eventos de Teclado |
|  | StartPosition | | CenterScreen |
| TextBox1 | (Name) | | txtTeclas |
| Label1 | (Name) | | Label1 |
|  | Text | | Ingrese desde el teclado: |
| Label2 | (Name) | | Label2 |
|  | Text | | Tecla presionada: |
| Label3 | (Name) | | Label3 |
|  | Text | | Código de Tecla: |
| Label4 | (Name) | | Label4 |
|  | Text | | Datos de Tecla: |
| Label5 | (Name) | | Label5 |
|  | Text | | Valor de Tecla: |
| Label6 | (Name) | | lblTecla |
|  | Text | | … |
| Label7 | (Name) | | lblCodigo |
|  | Text | | … |
| Label8 | (Name) | | lblDatos |
|  | Text | | … |
| Label9 | (Name) | | lblValor |
|  | Text | | … |
| CheckBox1 | (Name) | | chkAlt |
|  | Text | | ALT |
| CheckBox2 | (Name) | | chkShift |
|  | Text | | SHIFT |
| CheckBox3 | (Name) | | chkCtrl |
|  | Text | | CTRL |

1. Vamos a capturar los eventos del teclado desde el control txtTeclas, agregamos código al evento KeyPress de acuerdo a:

private void txtTeclas\_KeyPress(object sender, KeyPressEventArgs e)

{

lblTecla.Text = e.KeyChar.ToString();

}

El segundo parámetro contiene información de la tecla presionada, su propiedad KeyChar nos indica cuál de las teclas ha sido presionada y esta información lo mostraremos en el conrol lblTecla.

1. En ocasiones presionamos dos o más teclados para escribir un carácter especial, por ejemplo el @ (ALT+6+4), para identificar las teclas especiales que se presionan vamos a utilizar el evento KeyDown, vamos agregar código a este evento:

private void txtTeclas\_KeyDown(object sender, KeyEventArgs e)

{

chkAlt.Checked = false;

chkShift.Checked = false;

chkCtrl.Checked = false;

if (e.Alt)

{

chkAlt.Checked = true;

}

if (e.Shift)

{

chkShift.Checked = true;

}

if (e.Control)

{

chkCtrl.Checked = true;

}

lblCodigo.Text = e.KeyCode.ToString();

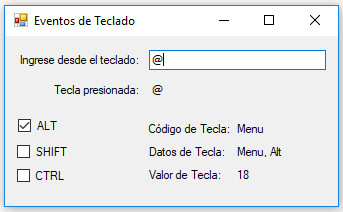
lblDatos.Text = e.KeyData.ToString();

lblValor.Text = e.KeyValue.ToString();

}

De la misma forma, el segundo argumento tiene información de las teclas especiales que se han presionado. Por ejemplo el atributo Alt nos indicará si se ha presionado la tecla ALT y el atributo KeyValue nos devuelve un número entero que identifica el código de la tecla presionada.

1. Ejecutamos la aplicación.



A medida que vamos ingresando texto al cuadro de texto podemos validar cuál de ellas ha sido presionada.

**Actividad:**

* Ingresa a la plataforma virtual, revisa los siguientes enlaces luego realiza las actividades propuestas
* Cuadros de diálogo personalizados.

https://www.youtube.com/watch?v=TMBN8IavL7I&list=PLMp96nOrGcZezSAPW08ulofVcEZRpYfH&index=8

* Formularios Modales y No Modales.

https://www.youtube.com/watch?v=boBmvGWTWJ8&list=PLMp96nOrGcZezSAPW08ulofVcEZRpYfH&index=9

1. Mostrar los cuadros de dialogo idóneos para los siguientes mensajes (agregar los botones, titulo e icono correctos):
   1. El campo ‘Código’ es obligatorio.
   2. ¿Desea incluir la propina en la Factura?
   3. Se va a proceder con el cierre contable correspondiente al periodo Junio-2019
2. Con ejemplos explicar en qué casos es conveniente usar formularios Modales y en qué casos los No Modales.