Tema: VALIDACIONES DE CAMPOS - LENGUAJE C#

Objetivo:

- ✓ Realizar validaciones de campos vacíos y de tipos de datos en interfaces graficas de usuario (IGU) de tipo Windows Form utilizando el lenguaje de programación C#

I. VALIDACIÓN DE CAMPOS VACIOS

La validación de campos vacíos es un recurso de programación que se hace necesario cuando se tiene campos obligatorios en la base de datos y/o cuando se quiere garantizar que los campos no se almacenen en blanco, por ejemplo en el caso de tener una llave primaria que no es autonumérica.

La validación de campos vacíos se puede realizar de muchas formas, una de ellas es aplicando una toma de decisión sobre el contenido del textbox a validar, verificando si este se encuentra vació o no. Una función en C# que efectuaría esta tarea seria por ejemplo:

```
public bool ValidarCampoVacio(TextBox CuadroTexto, string nombrecampo )
     {
        string valoringresado = CuadroTexto.Text.Trim();
        if (valoringresado == "")
        {
            MessageBox.Show("El Campo " + nombrecampo + " No puede estar
            vacio", "Atención", MessageBoxButtons.OK,
            MessageBoxIcon.Information);

            CuadroTexto.BackColor = Color.Tomato;
            return false;
        }
        else
        {
            CuadroTexto.BackColor = Color.White;
            return true;
        }
     }
```

Nota: *La función Trim() de C# sirve para quitar todos los espacios es blanco que podrían estar presentes en el textbox*

Esta función recibe como parámetros de entrada el textbox a validar y el nombre del campo que representa, y retorna un valor booleano (true o false) dependiendo si el campo está vacío o no. Adicionalmente cambia el color del textbox cuando este efectivamente se detecta como vacío. Para utilizarla se hace su llamado por cada campo a validar en el botón correspondiente. Por ejemplo:

```
if (ValidarCampoVacio(textBox1, "Identificación") &&
    ValidarCampoVacio(textBox2, "Nombres") &&
    ValidarCampoVacio(textBox3, "Apellidos"))
        {
           MessageBox.Show("Muy Bien, Todos los datos ingresados son
           validos!!!", "Validación De Campos Vacios",
           MessageBoxButtons.OK, MessageBoxIcon.Information);
         }
```

II. VALIDACION DE TIPOS DE DATOS

Este tipo de validación es importante para garantizar que la información que se envía hacia la base de datos corresponda con el tipo de dato definido para cada campo. Para realizar este tipo de validaciones existen diferentes métodos, uno de ellos es haciendo uso de la denominada ***Tabla ASCII***, la cual corresponde a un listado predefinido que le da una representación numérica a cada carácter del teclado de un computador.

La Tabla ASCII está distribuida tal y como se muestra a continuación :

ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo
0	0	NUL	16	10	DLE	32	20	(espacio)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?

ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	\	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	

Teniendo en cuenta la distribución de los códigos ASCII asignados, los caracteres a tener en cuenta para realizar validaciones de tipos de datos serian:

a.) **Solo Números:** Códigos ASCII del 48 al 57.

b.) **Solo Letras:** Códigos ASCII del 65 al 90 para mayúsculas
Códigos ASCII del 97 al 122 para minúsculas

c.) **Enter:** Código ASCII 13

d.) **Retroceso:** Código ASCII 8

e.) **Espacio:** Código ASCII 32

Un método muy usado para efectuar la validación de tipos de datos es usando el evento "KeyPress" del TextBox, el cual se ejecuta inmediatamente cuando el usuario ingresa un carácter dentro de él. Dentro de este evento se debe programar una toma de decisiones con los códigos ASCII que se requieren validar. El código ASCII de un carácter se obtiene en el lenguaje C# con la expresión "`e.KeyChar`"

III. VALIDACION DE CORREO ELECTRONICO (E-MAIL)

Un correo electrónico es una cadena de caracteres compuesta de una cierta estructura que ya está definida y que es posible validar mediante programación, y habrá ocasiones en las que se requiere validar que un correo electrónico se encuentre escrito en el formato correcto. Para validar este formato se puede crear un método se encargará de recibir una cadena de texto y verificar que posee un formato correcto, si es así regresara como resultado un "true" aceptando que la cadena introducida está escrita correctamente y en caso contrario regresara un "false" indicando que la cadena no posee el formato correcto.

Para hacer la comprobación de correo electrónico en el lenguaje C# se utiliza una clase llamada "Regex", la cual viene está disponible desde el Framework de .Net y permite mediante un patrón verificar si una cadena cumple o no con dicho patrón. Para poder utilizar la clase "Regex" se debe añadir el espacio de nombre "System.Text.RegularExpressions"

IV. VALIDACIÓN DE FECHAS

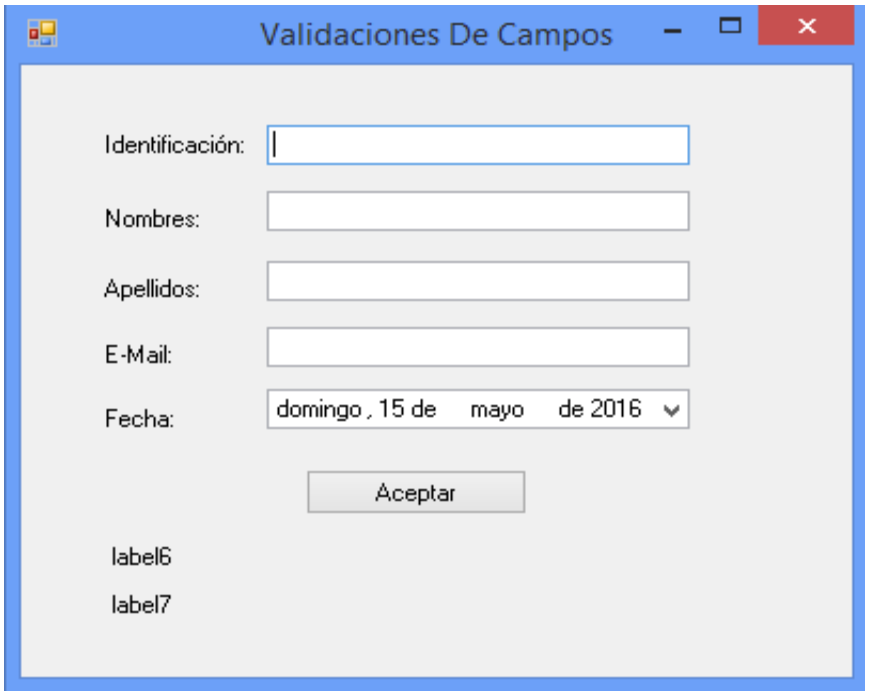
Para validar fechas se puede hacer uso del control llamado DateTimePicker1, el cual despliega un calendario para que el usuario escoja una fecha. Para leer el valor de fecha seleccionado se usa la instrucción: "DateTimePicker1.Value.Date.ToShortDateString()"

V. VALIDACIÓN DE NUMERO MAXIMO DE CARACTERES

Para validar el número máximo de caracteres que puede tener un campo de texto basta con reasignarle el valor a la propiedad "MaxLength" en el cuadro de propiedades del control. Se recomienda ajustar este parámetro para que tenga la misma longitud del campo que representa en la base de datos en caso de que se esté desarrollando una aplicación de 3 capas.

EJERCICIO EJEMPLO

Paso 1: En tiempo de diseño desarrolle el siguiente formulario:



Paso 2: En la propiedad **"MaxLength"** de los textBox correspondientes coloque los siguientes valores:

- ✓ Máximo 15 números para la Identificación
- ✓ Máximo 30 Caracteres para los Nombres
- ✓ Máximo 30 Caracteres para los Apellidos
- ✓ Máximo 50 Caracteres para el E-mail

Paso 3: Digite el código que se muestra a continuación

```
using System.Text.RegularExpressions;

public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

  public bool ValidarCampoVacio(TextBox CuadroTexto, string nombrecampo)
    {
        string valoringresado = CuadroTexto.Text.Trim();

        if (valoringresado == "")
         {
            MessageBox.Show("El Campo " + nombrecampo + " No puede estar
            vacio", "Cuidado", MessageBoxButtons.OK,
            MessageBoxIcon.Information);

            CuadroTexto.BackColor = Color.Tomato;
            return false;
         }
        else
         {
            CuadroTexto.BackColor = Color.White;
            return true;
         }
    }

public bool validarEmail(TextBox CuadroTexto)
        {
            string email = CuadroTexto.Text;

            bool emailvalido = false;
            String expresion = "\\w+([-+.']\\w+)*@\\w+([-
            .]\\w+)*\\.\\w+([-.]\\w+)*";

            if (Regex.IsMatch(email, expresion))
            {
                  CuadroTexto.BackColor = Color.White;
                  emailvalido = true;
            }
            else
            {
                  MessageBox.Show("Debe Ingresar un valor de E-Mail
                  Valido(Ejemplo: correo@servidor.com)", "Atención",
                  MessageBoxButtons.OK, MessageBoxIcon.Information);

                  CuadroTexto.BackColor = Color.Tomato;
                  emailvalido = false;
            }
            return emailvalido;
        }
```

```
// Importante: Para activar correctamente los eventos textBox_KeyPress,
en el cuadro de propiedades del textBox dar clic en el icono de eventos
(forma de rayo) y luego dar doble clic en el evento KeyPress
correspondiente

private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
    {
       if (!char.IsNumber(e.KeyChar) && e.KeyChar != 8 && e.KeyChar != 13)
          {
             MessageBox.Show("Debe Ingresar sólo Numeros en este campo",
             "Atención", MessageBoxButtons.OK,MessageBoxIcon.Information);

             e.Handled = true;
          }

        if (e.KeyChar == 13)
           {
             textBox2.Focus();
           }
     }

  private void textBox2_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!char.IsLetter(e.KeyChar) && e.KeyChar != 8 &&
            e.KeyChar != 13 && e.KeyChar != 32)
              {
                  MessageBox.Show("Debe Ingresar sólo Letras en este
                  campo", "Atención", MessageBoxButtons.OK,
                  MessageBoxIcon.Information);

                  e.Handled = true;
              }

            if (e.KeyChar == 13)
            {
                textBox3.Focus();
            }
        }

   private void textBox3_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!char.IsLetter(e.KeyChar) && e.KeyChar != 8 &&
            e.KeyChar != 13 && e.KeyChar != 32)
              {
                  MessageBox.Show("Debe Ingresar sólo Letras en este
                  campo", "Atención", MessageBoxButtons.OK,
                  MessageBoxIcon.Information);

                  e.Handled = true;
              }
```

```
            if (e.KeyChar == 13)
            {
                textBox4.Focus();
            }
        }


private void button1_Click(object sender, EventArgs e)
        {
            if (ValidarCampoVacio(textBox1, "Identificación") &&
                ValidarCampoVacio(textBox2, "Nombres") &&
                ValidarCampoVacio(textBox3, "Apellidos") &&
                validarEmail(textBox4) )
            {

                // Aquí podria ir el codigo correspondoente a la tarea
                CRUD asociada a este Boton (Es decir, aquí deberia ir el
                codigo para Guardar, Consultar, Actualizar o eliminiar)

                MessageBox.Show("Muy Bien, Todos los datos ingresados son
                validos!!!", "Validación De Campos Vacios",
                MessageBoxButtons.OK, MessageBoxIcon.Information);

                label6.Text = "Los datos Ingresaron fueron " +
                textBox1.Text + " - " + textBox2.Text + " - " +
                textBox3.Text + " - " + textBox4.Text;

                label7.Text = "La Fecha Seleccionada Fue " +
                dateTimePicker1.Value.Date.ToShortDateString();

            }
        }
```

Paso 4: Ejecute el proyecto, deje campos vacíos, intente ingresar datos invalidados en los campos y luego presione clic en el botón Aceptar. Verifique el funcionamiento de las validaciones aplicadas en el formulario