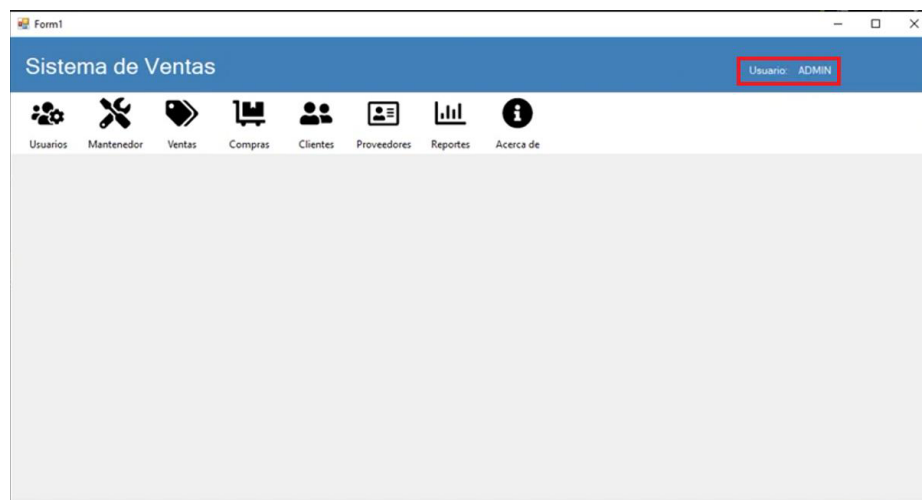


	UT 2 – <i>Práctica Sistema de Ventas</i> Parte III – Filtro Menú según Rol	CFGS Desarrollo de Aplicaciones Multiplataforma  Módulo: DDI
--	---	---

**Este diseño se corresponde con el video de apoyo número 3.**

En el formulario de inicio vamos a visualizar el nombre del usuario que se ha logado. Para ello vamos a agregar dos etiquetas en la parte superior derecha. La primera una etiqueta cuyo texto Text=Usuario, ForeColor=blanco, BackColor=SteelBlue y tamaño fuente Size=9. La segunda que llamaremos lblusuario con las mismas propiedades que utilizaremos para visualizar el nombre de usuario.



En el evento de click del botón de inicio de sesión del formulario login obteníamos los datos del usuario que inicia sesión y lo almacenábamos en ousuario. Vamos a enviar este objeto a nuestro formulario de inicio.

```
private void btIngresar_Click(object sender, EventArgs e)
{
    List<Usuario> TEST = new CN_Usuario().Listar();

    Usuario ousuario = new CN_Usuario().Listar().Where(u => u.Documento == txtdocumento.Text && u.Clave == txtclave.Text).FirstOrDefault();

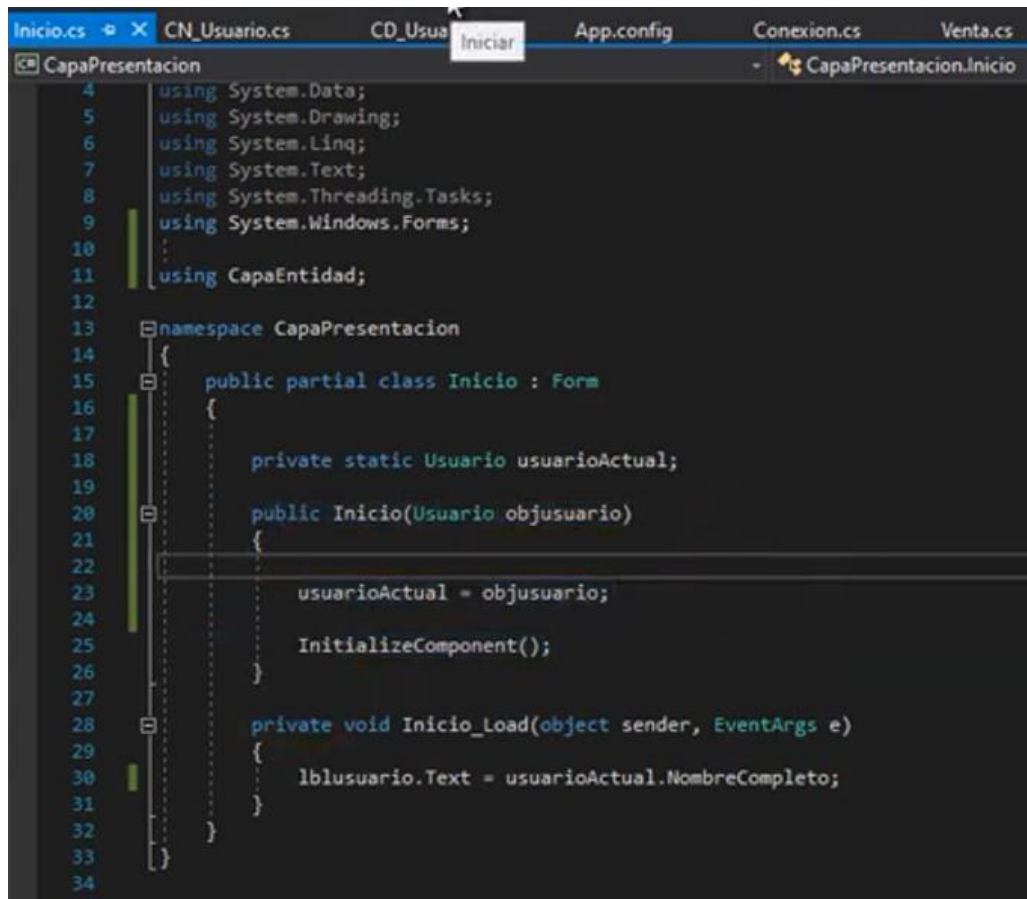
    if (ousuario != null)
    {
        Inicio form = new Inicio(ousuario);
        form.Show();
        this.Hide();

        form.FormClosing += frm_closing;
    }
    else {
        MessageBox.Show("no se encontro el usuario", "Mensaje", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
    }
}
```

Para visualizar el usuario en nuestro formulario tenemos que indicar en el **constructor** del formulario **que va a recibir un objeto usuario** que almacenaremos en una variable. Para que nos reconozcan la clase usuario tenemos que indicar la referencia a la CapaEntidad.

	<b>UT 2 – <i>Práctica Sistema de Ventas</i></b> <b>Parte III – Filtro Menú según Rol</b>	CFGS Desarrollo de Aplicaciones Multiplataforma Módulo: DDI
--	---	---

En el evento de Load del formulario inicio vamos a indicarle que muestre el nombre del usuario.

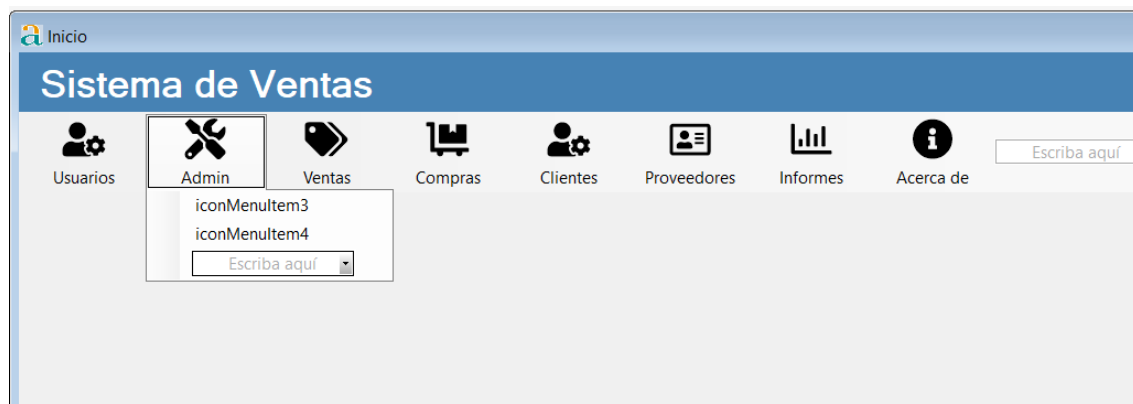


```

4      using System.Data;
5      using System.Drawing;
6      using System.Linq;
7      using System.Text;
8      using System.Threading.Tasks;
9      using System.Windows.Forms;
10     ...
11     using CapaEntidad;
12
13     namespace CapaPresentacion
14     {
15         public partial class Inicio : Form
16         {
17
18             private static Usuario usuarioActual;
19
20             public Inicio(Usuario objusuario)
21             {
22
23                 usuarioActual = objusuario;
24
25                 InitializeComponent();
26             }
27
28             private void Inicio_Load(object sender, EventArgs e)
29             {
30                 lblusuario.Text = usuarioActual.NombreCompleto;
31             }
32         }
33     }
34

```

Trabajaremos ahora con nuestro menú que tiene que tener estas funcionalidades:



-El menú de Admin (o mantenimiento). va a tener dos submenús añadidos dos IconMenuItems con el Text Categoría y Producto que se llamaran submenucategoria y submenuproducto. Los añadimos pinchando sobre la flecha que aparece al lado

-El menú de Ventas también va a tender dos submenús con Text Nueva Venta y Ver detalle y

	<b>UT 2 – <i>Práctica Sistema de Ventas</i></b> <b>Parte III – Filtro Menú según Rol</b>	CFGS Desarrollo de Aplicaciones Multiplataforma Módulo: DDI
--	---	---

nombres submenuregistraventa y submenuverdetalleventa

-El menú de Ventas también va a tender dos submenús con Text Nueva Compra y Ver detalle y nombres submenuregistracompra y submenuverdetallecompra

-Al pulsar sobre un IconMenuitem éste tiene que resaltarse cambiando el color, de este modo sabremos que opción hemos seleccionado. El resto de iconos deben aparecer con su fondo blanco.

-Al pulsar sobre un IconMenuitem tiene que abrirse el formulario correspondiente y ocultarse el que estuviese abierto si lo había.

Añadimos un panel que llamaremos Contenedor que ocupará la parte del formulario de inicio que queda libre dónde mostraremos el formulario ligado al IconMenuitem seleccionado (o vacío en caso de que se haya seleccionado aún ninguno).

Dentro de nuestra CapaPresentacion agregamos un nuevo formulario frmUsuarios al que añadimos un Label con un Text que nos sirva para identificar el formulario cuando testemos la aplicación. Repetimos el proceso con los formularios frmCategoria, frmProducto, frmVentas, frmDetalleVenta, frmCompras, frmDetalleCompra, frmClientes, frmProveedores y frmInformes,

## ENLACES DEL MENÚ

Comenzamos con el IconMenuitem menuusuario. Dando doble clic sobre él vamos a enlazar con su formulario definiendo un nuevo método AbrirFormulario al que llamaremos desde el evento click del IconMenuitem menuusuario,

El método abrir formulario recibirá dos parámetros, uno de tipo IconMenuitem sobre el que hemos clicado (si no lo reconoce añadir la referencia a FontAwesome) y el formulario que va a ser visualizar.

Tenemos que controlar si había un menú ya seleccionado y un formulario visible para ocultarlo antes de mostrar el nuevo seleccionado.

Si el menú activo no es nulo existirá uno ya seleccionado por lo que cambiaremos su color de fondo dejándolo a su valor blanco y el menú que actualmente estamos seleccionando lo cambiaremos a color silver. Nuestro menú activo pasará a ser el IconMenuitem seleccionado que se corresponde con el parámetro menu que hemos pasado al método.

Si el formulario activo no es nulo habrá un formulario ocupando el espacio de nuestro panel contenedor por lo que tendremos que cerrarlo.

El formulario activo pasa a ser el formulario que vamos a abrir y que lo hemos pasado como parámetro al método.

Un formulario de nivel superior es una ventana que no tiene un formulario principal o cuyo formulario principal es la ventana del escritorio. Las ventanas de nivel superior se utilizan normalmente como formulario principal en una aplicación. En este caso el nuevo formulario que abrimos no es el principal por lo que establecemos la propiedad `TopLevel` a falso. También indicamos que no va a tener bordes, en la propiedad `Dock` que nos alinea el formulario tendrá el valor `fill` para que rellene todo el espacio del contenedor disponible, le ponemos el fondo por defecto, lo agregamos al panel llamado contenedor y lo visualizamos.

```

Inicio.cs* x lto.cs [Diseño]*
CapaPresentacion
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10
11 using CapaEntidad;
12 using FontAwesome.Sharp;
13
14
15 namespace CapaPresentacion
16 {
17     public partial class Inicio : Form
18     {
19
20         private static Usuario usuarioActual;
21         private static IconMenuItem MenuActivo = null;
22         private static Form FormularioActivo = null;
23
24         public Inicio(Usuario objusuario)
25         {
26             usuarioActual = objusuario;
27             InitializeComponent();
28         }
29
30         private void Inicio_Load(object sender, EventArgs e)
31         {
32             lblusuario.Text = usuarioActual.NombreCompleto;
33         }
34
35         private void AbrirFormulario(IconMenuItem menu, Form formulario) {
36
37             if (MenuActivo != null) {
38                 MenuActivo.BackColor = Color.White;
39             }
40             menu.BackColor = Color.Silver;
41             MenuActivo = menu;
42
43             if (FormularioActivo != null) {
44                 FormularioActivo.Close();
45             }
46
47             FormularioActivo = formulario;
48             formulario.TopLevel = false;
49             formulario.FormBorderStyle = FormBorderStyle.None;
50             formulario.Dock = DockStyle.Fill;
51             formulario.BackColor = Color.SteelBlue;
52
53             contenedor.Controls.Add(formulario);
54             formulario.Show();
55         }
56
57         private void menuusuarios_Click(object sender, EventArgs e)
58         {
59             AbrirFormulario((IconMenuItem)sender, new frmUsuarios());
60         }
61     }
62 }
63

```

	UT 2 – <i>Práctica Sistema de Ventas</i> Parte III – Filtro Menú según Rol	CFGS Desarrollo de Aplicaciones Multiplataforma Módulo: DDI
--	---	---

Agregaremos el evento click de cada menú y submenú haciendo doble clic sobre el ítem correspondiente teniendo en cuenta una diferencia entre `IconMenuItem` del menú principal o de un submenú. Al hacer click un submenú el parámetro `sender` se corresponde con el valor `IconMenuItem` del menú por lo que tenemos que hacer referencia directamente a su nombre

```

63
64 private void menuusuarios_Click(object sender, EventArgs e)
65 {
66     AbrirFormulario((IconMenuItem)sender, new frmUsuarios());
67 }
68
69 private void submenucategoria_Click(object sender, EventArgs e)
70 {
71     AbrirFormulario(menumantenedor, new frmCategoria());
72 }
73
74 private void submenuproducto_Click(object sender, EventArgs e)
75 {
76     AbrirFormulario(menumantenedor, new frmProducto());
77 }
78
79 private void submenuregistrarventa_Click(object sender, EventArgs e)
80 {
81     AbrirFormulario(menuventas, new frmVentas());
82 }
83
84 private void submenuverdetalleventa_Click(object sender, EventArgs e)
85 {
86     AbrirFormulario(menuventas, new frmDetalleVenta());
87 }
88
89 private void submenuregistrarcompra_Click(object sender, EventArgs e)
90 {
91     AbrirFormulario(menucompras, new frmCompras());
92 }
93
94 private void submenuverdetallecompra_Click(object sender, EventArgs e)
95 {
96     AbrirFormulario(menucompras, new frmDetalleCompra());
97 }
98
99 private void menuclientes_Click(object sender, EventArgs e)
100 {
101     AbrirFormulario((IconMenuItem)sender, new frmClientes());
102 }
103
104 private void menuproveedores_Click(object sender, EventArgs e)
105 {
106     AbrirFormulario((IconMenuItem)sender, new frmProveedores());
107 }
108
109 private void menureportes_Click(object sender, EventArgs e)
110 {
111     AbrirFormulario((IconMenuItem)sender, new frmReportes());
112 }
113 }
114
115

```

	<b>UT 2 – <i>Práctica Sistema de Ventas</i></b> <b>Parte III – Filtro Menú según Rol</b>	CFGS Desarrollo de Aplicaciones Multiplataforma Módulo: DDI
--	---	---

## FILTRO MENÚ POR ROL

Los roles de usuario nos van a servir para restringir o permitir el acceso a las distintas opciones del menú. En nuestra BBDD tenemos una tabla permiso (actualmente vacía) con columnas, id rol y nombremenu en la que vamos a indicar a qué opciones del menú (ItemsIcon) va a tener acceso un rol.

En nuestra base de datos introducimos los permisos para el rol Administrador y el rol Empleado. La descripción tiene que coincidir con el nombre del ItemsIcon al que queremos dar acceso. Si un rol no debe tener acceso a una opción del menú no daremos de alta la relación idrol-nombre item en la tabla. De este modo el usuario que se logea en la aplicación solo tendrá acceso a las opciones de menú que su rol tiene identificadas en la tabla permiso.



```
USE BDSISTEMAVENTAS;
```

```
SELECT * FROM ROL;
SELECT * FROM PERMISO;
```

```
INSERT INTO PERMISO (IdRol,Descripcion) VALUES (
('1','menuusuarios'),
('1','menumantenedor'),
('1','menuventas'),
('1','menucompras'),
('1','menuclientes'),
('1','menuproveedores'),
('1','menureportes'),
('1','menuacercade'));
```

```
INSERT INTO ROL (Descripcion) VALUES ('EMPLEADO');
```

```
SELECT * FROM ROL;
```

```
INSERT INTO PERMISO (IdRol,Descripcion) VALUES (
('2','menuventas'),
('2','menucompras'),
('2','menuclientes'),
('2','menuproveedores'),
('2','menuacercade'));
```

```
SELECT * FROM PERMISO;
```

	<b>UT 2 – <i>Práctica Sistema de Ventas</i></b> <b>Parte III – Filtro Menú según Rol</b>	CFGS Desarrollo de Aplicaciones Multiplataforma Módulo: DDI
--	---	---

Para realizar la consulta a nuestra tabla de permisos tenemos que crear una nueva clase en nuestra **capa de datos** llamada **CD\_PERMISO**.

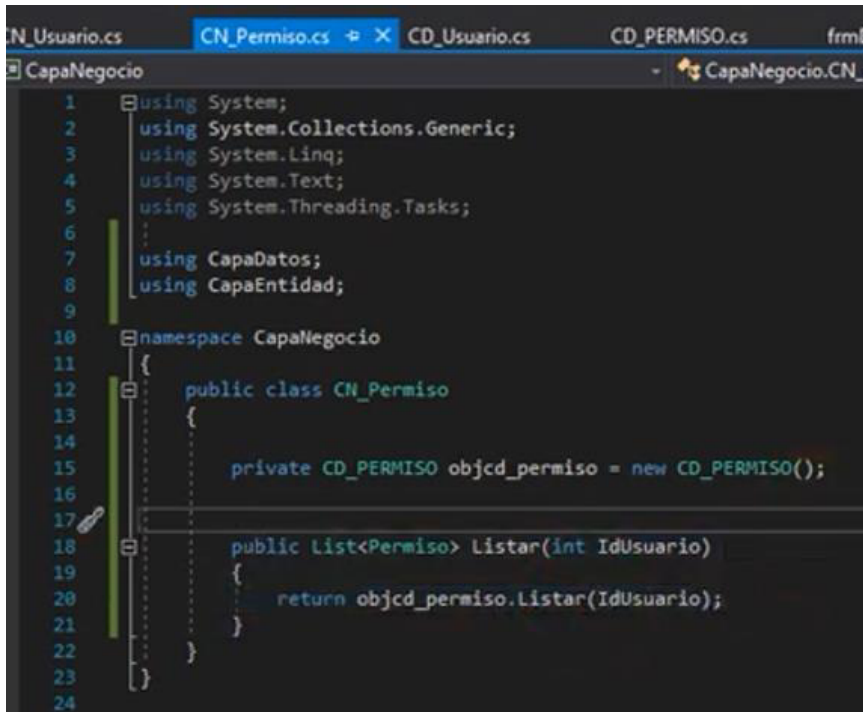
```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  using System.Data;
8  using System.Data.SqlClient;
9  using CapaEntidad;
10
11 namespace CapaDatos
12 {
13     public class CD_PERMISO
14     {
15
16         public List<Permiso> listar(int idusuario)
17         {
18             List<Permiso> lista = new List<Permiso>();
19
20             using (SqlConnection oconexion = new SqlConnection(Conexion.cadena))
21             {
22                 try
23                 {
24                     StringBuilder query = new StringBuilder();
25                     query.AppendLine("select p.IdRol,p.NombreMenu from PERMISO p");
26                     query.AppendLine("inner join ROL r on r.IdRol = p.IdRol");
27                     query.AppendLine("inner join USUARIO u on u.IdRol = r.IdRol");
28                     query.AppendLine("where u.IdUsuario = @idusuario");
29
30                     SqlCommand cmd = new SqlCommand(query.ToString(), oconexion);
31                     cmd.Parameters.AddWithValue("@idusuario", idusuario);
32                     cmd.CommandType = CommandType.Text;
33
34                     oconexion.Open();
35
36                     using (SqlDataReader dr = cmd.ExecuteReader())
37                     {
38                         while (dr.Read())
39                         {
40                             lista.Add(new Permiso()
41                             {
42                                 oRol = new Rol() { IdRol = Convert.ToInt32(dr["IdRol"]) },
43                                 NombreMenu = dr["NombreMenu"].ToString(),
44                             });
45                         }
46                     }
47                 }
48                 catch (Exception ex)
49                 {
50
51                 }
52             }
53
54             lista = new List<Permiso>();
55
56             return lista;
57         }
58     }
59 }

```

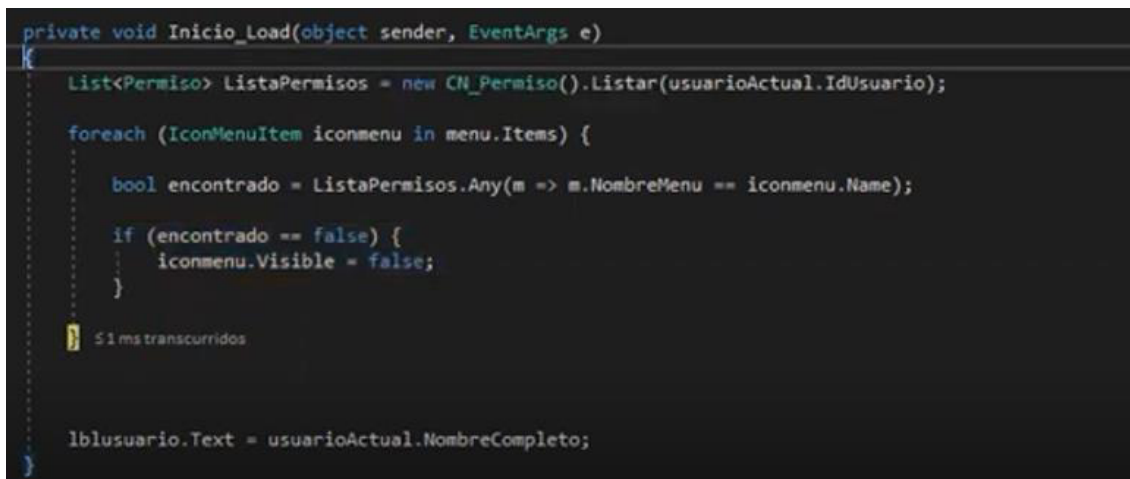
Y de nuevo para que nuestra capa presentación tenga acceso tenemos que crear el enlace en la capa de negocio. Creamos una clase en la capa de negocio llamada CN\_Permiso





```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  ...
7  using CapaDatos;
8  using CapaEntidad;
9
10 namespace CapaNegocio
11 {
12     public class CN_Permiso
13     {
14
15         private CD_PERMISO objcd_permiso = new CD_PERMISO();
16
17
18         public List<Permiso> Listar(int IdUsuario)
19         {
20             return objcd_permiso.Listar(IdUsuario);
21         }
22     }
23 }
24
```

Y ya podemos ejecutar el método desde nuestra capa presentación. Vamos al formulario de inicio y en el momento en que se carga la página solo tenemos que visualizar los permisos que tenga el usuario que se haya logado, para ello comparamos si cada elemento del iconmenuitems esta dentro de la lista de permisos de ese usuario logado que hemos obtenido de nuestra base de datos (hacemos **referencia a capa de negocio** CN\_Permiso para poder obtener nuestra lista de permisos por lo que incluiremos using CapaNegocio)



```
private void Inicio_Load(object sender, EventArgs e)
{
    List<Permiso> ListaPermisos = new CN_Permiso().Listar(usuarioActual.IdUsuario);

    foreach (IconMenuItem iconmenu in menu.Items) {

        bool encontrado = ListaPermisos.Any(m => m.NombreMenu == iconmenu.Name);

        if (encontrado == false) {
            iconmenu.Visible = false;
        }

        // 51 ms transcurridos

        lblusuario.Text = usuarioActual.NombreCompleto;
    }
}
```

Con esto ya tendríamos nuestro filtro menús por rol.