Herramientas de Programación 1

Tema Nº5:

Formularios MDI.

Indicador de logro Nº5:Realiza aplicaciones comerciales básicas utilizando formularios MDI y sus controles asociados

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº5:**

Formularios MDI

Los programas de ordenador gráficos de interfaz de múltiples documentos (MDI) son aquellos cuyas ventanas se encuentran dentro de una ventana padre. Las aplicaciones de escritorio comerciales generalmente cuentan con un formulario MDI, que es el principal, y varios formularios hijos, cada uno con un propósito funcional específico.

La propiedad que convierte a un formulario simple a un formulario MDI es la propiedad:

|  |  |
| --- | --- |
| Propiedad | Descripción |
| IdMdiContainer | Determina si un formulario es un contenedor MDI. |

Esta propiedad debe tener el valor true.

**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 5:**

5.1 Formularios MDI y controles asociados: MenuStrip, ToolStrip y StatusStrip

**Objetivos**

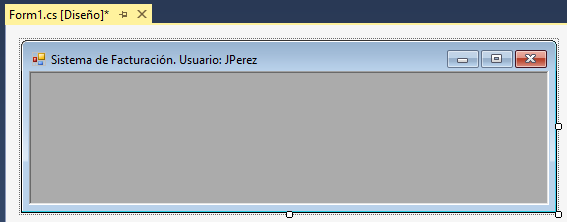
* Desarrollar aplicaciones utilizando formularios Padre (MDI) e Hijos.
* Agregar al formulario MDI un menú superior, una barra de herramientas y una barra de estados.

**Introducción**

Durante esta actividad, vamos a desarrollar una aplicación base para la implementación de sistemas comerciales, al menos para la capa de presentación (GUI).

Paso 1. Formularios MDI.

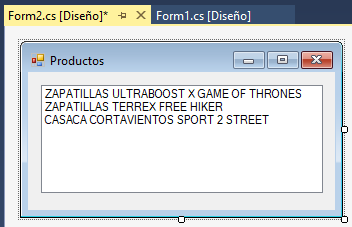
1. Creamos el proyecto Semana05 agregamos un nuevo formulario con el diseño siguiente:



|  |  |  |  |
| --- | --- | --- | --- |
| Control | Propiedad | Valor | |
| Form1 | (Name) | | frmPrincipal |
|  | IsMdiContainer | | True |
|  | Text | | Sistema de Facturación. Usuario: JPerez |
|  | WindowsState | | Maximized |

Con esto ya tenemos el formulario principal que es de tipo MDI.

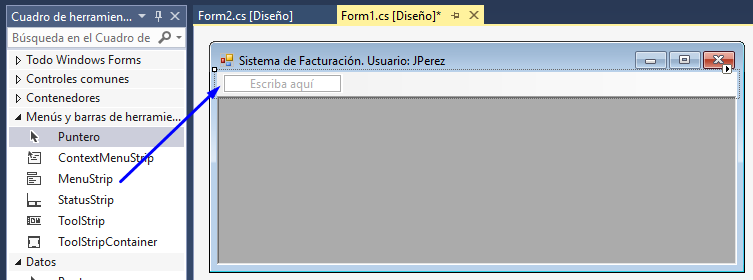
1. Vamos a crear dos formularios hijos, el primero lo hacemos con este diseño:



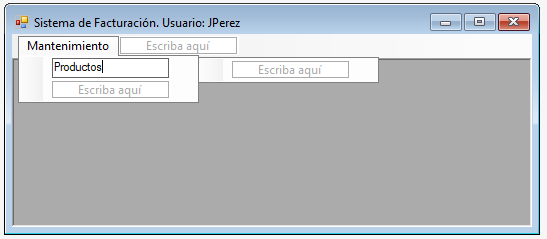
|  |  |  |  |
| --- | --- | --- | --- |
| Control | Propiedad | Valor | |
| Form1 | (Name) | | frmProductos |
|  | Text | | Productos |
| ListBox1 | (Name) | | lsbProductos |
|  | Items | | [A criterio del alumno] |

Paso 2. MenuStrip.

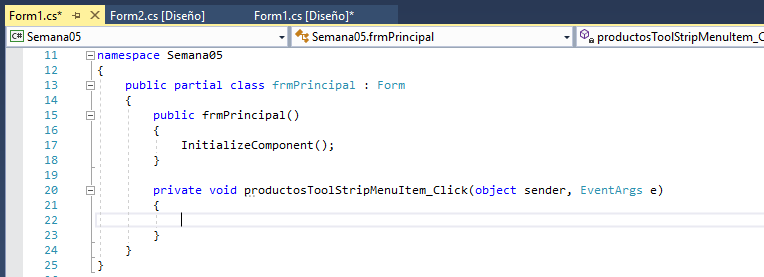
1. En una aplicación MDI los formularios hijos inicialmente se abren desde alguna opción del menú, por ello ahora vamos agregar un menú al formulario principal.



1. Para agregar opciones de menú basta con ubicarse donde crearlas y empezar a digitar el nombre de la opción, vamos a poner dos opciones, de las cuales la opción Productos permitirá abrir el formulario hijo frmProductos.



1. Para agregar código al evento Click de la opción de menú Productos hacemos doble click en la opción del menú.



Agregamos código de acuerdo a:

public partial class frmPrincipal : Form

{

private frmProductos miFormularioDeProductos;

public frmPrincipal()

{

InitializeComponent();

}

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

{

miFormularioDeProductos = new frmProductos();

miFormularioDeProductos.MdiParent = this;

miFormularioDeProductos.Show();

}

}

Para crear y mostrar un formulario hijo debemos hacer varias cosas.

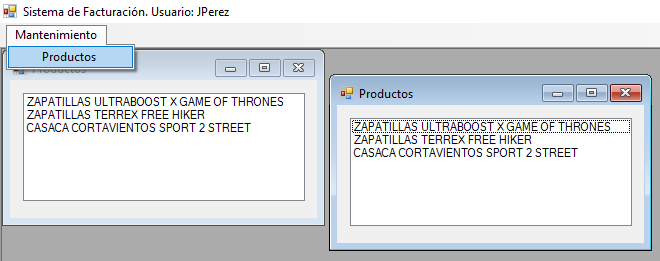
Para este caso lo primero es agregar a la clase frmPrincipal el atributo miFormularioDeProductos del tipo frmProductos.

Luego, dentro del código del evento Click:

* Creamos una instancia del formulario hijo.
* Le indicamos cual es el padre del formulario hijo, en este ejemplo la palabra this identifica al formulario frmPrincipal.
* Finalmente lo mostramos con el método Show().

Debemos resaltar que cada vez que presionamos la opción Productos se va abriri un nuevo formulario Producto.

1. Ejecutamos la aplicación.



1. Para mantener un control de los formularios hijos abiertos debemos mostrarlos una sola vez, esto es, crear solo una instancia. Para ver cómo hacerlo modificamos el evento Click de la misma opción de menú y agregamos un método:

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

{

if(miFormularioDeProductos == null)

{

miFormularioDeProductos = new frmProductos();

miFormularioDeProductos.MdiParent = this;

miFormularioDeProductos.FormClosed += new FormClosedEventHandler(CerrarFormularioDeProductos);

miFormularioDeProductos.Show();

}

else

{

miFormularioDeProductos.Activate();

}

}

void CerrarFormularioDeProductos(object sender, FormClosedEventArgs e)

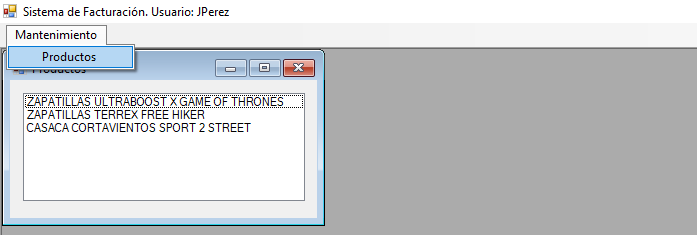
{

miFormularioDeProductos = null;

}

El método CerrarFormularioDeProductos() se encargará eliminar la instancia del formulario miFormularioDeProductos, ya que cuando se cierra este formulario solo deja de mostrarse pero la instancia sigue creada.

1. En el código del evento Click lo primero que hacemos es verificar si ya hay una instancia creada del formulario miFormularioDeProductos, si no existe lo creamos y mostramos, si ya existe (así esté cerrado o no) simplemente lo volvemos a mostrar con el método Activate().



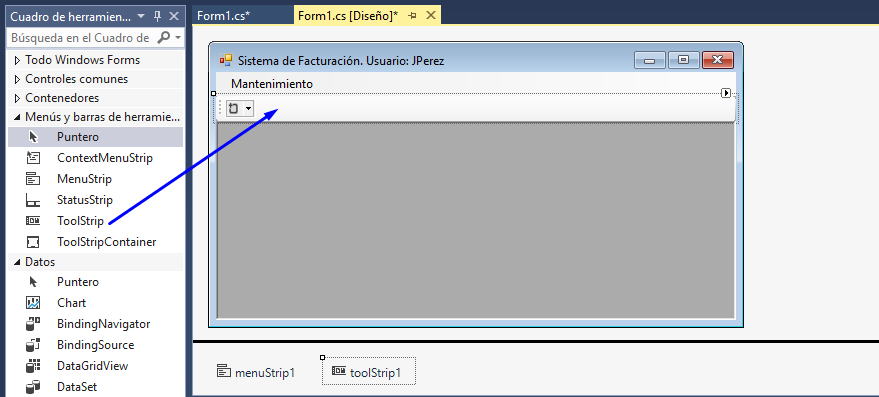
En cuanto a formularios MDI debemos tener en cuenta lo siguiente:

* Cuando tengamos abierto el formulario MDI y alguno(s) de sus formularios hijo, en la barra de tareas de Windows solo se mostrara un icono: el del MDI.
* Para cerrar todos los formularios de la aplicación bastara con cerrar el formulario MDI.

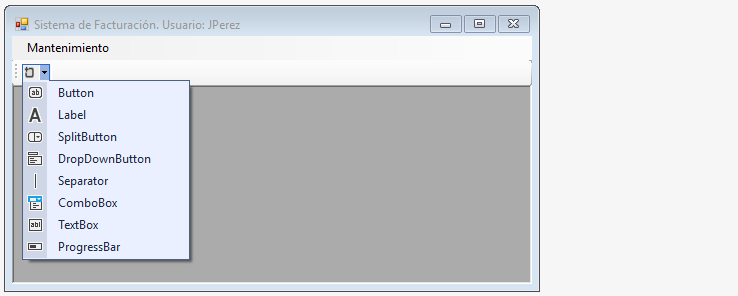
Paso 3. ToolStrip.

Es bastante útil poner en una barra de herramientas (ToolStrip) accesos directos (iconos) a formularios hijos que se usan frecuentemente, esta es la ventaja de usar un ToolStrip.

1. Agregamos a nuestro formulario MDI un ToolStrip.

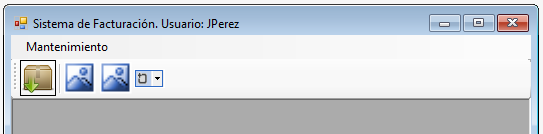


1. Son distintos los controles que podemos agregar en una barra de herramientas, los más comunes son los botones y los separadores.



1. Agregamos tres botones y un separador, cambiamos las propiedades del ToolStrip:

|  |  |  |
| --- | --- | --- |
| Control | Propiedad | Valor |
| ToolStrip1 | (Name) | ToolStrip1 |
|  | ImageWcalingSize > Width | 32 |
|  | ImageWcalingSize > Height | 32 |
| ToolStripButton1 | (Name) | tsbProductos |
|  | Image | [Seleccionar una imagen de 32x32] |
|  | Text | Productos |



1. Hacemos doble click en el primer botón para agregar código a su evento Click. Agregamos el mismo código que la opción de menú Productos.

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

{

if(miFormularioDeProductos == null)

{

miFormularioDeProductos = new frmProductos();

miFormularioDeProductos.MdiParent = this;

miFormularioDeProductos.FormClosed += new FormClosedEventHandler(CerrarFormularioDeProductos);

miFormularioDeProductos.Show();

}

else

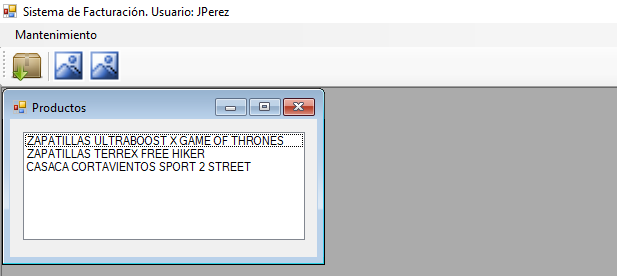
{

miFormularioDeProductos.Activate();

}

}

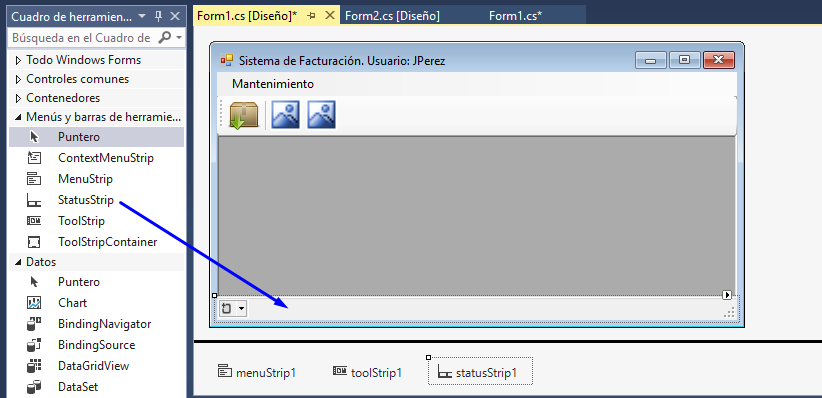
1. Ejecutamos la aplicación y abrimos el formulario de Productos con la opción del menú y el botón de la barra de herramientas.



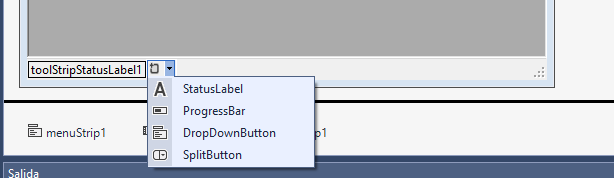
Paso 4. StatusStrip.

Es bastante útil también poner en una barra de estado dentro de un formulario MDI, esto con el objeto de mostrar mensajes al usuario de forma centralizada.

1. Agregamos a nuestro formulario MDI un StatusStrip.



1. Son distintos los controles que podemos agregar en una barra de herramientas, los más comunes son las etiquetas (StatusLabel) y las barras de progreso (ProgressBar).



1. Agregamos un StatusLabel, cambiamos sus propiedades:

|  |  |  |
| --- | --- | --- |
| Control | Propiedad | Valor |
| StatusStrip1 | (Name) | StatusStrip1 |
| ToolStripStatusLabel1 | (Name) | tslMensajes |
|  | Text | … |

1. El primer mensaje que mostraremos será cuando el formulario principal (MDI) se muestre:

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

{

tslMensajes.Text = "Bienvenido";

}

1. También cuando se cierra el formulario miFormularioDeProductos:

void CerrarFormularioDeProductos(object sender, FormClosedEventArgs e)

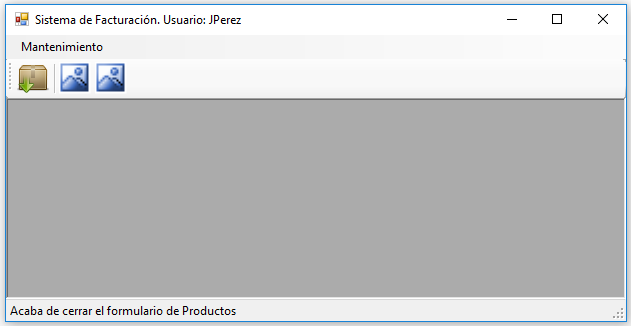
{

miFormularioDeProductos = null;

tslMensajes.Text = "Acaba de cerrar el formulario de Productos";

}

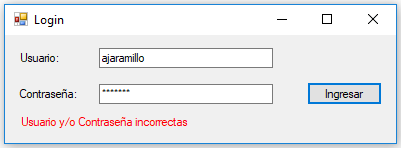
1. Ejecutamos la aplicación y verificamos su funcionamiento.



**Actividad:**

* Ingresa a la plataforma virtual, revisa los siguientes enlaces luego realiza las actividades propuestas
* Formularios MDI.
* <https://docs.microsoft.com/es-es/dotnet/framework/winforms/controls/walkthrough-creating-an-mdi-form-with-menu-merging-and-toolstrip-controls>
* <https://www.youtube.com/watch?v=zngH8voq_5U&list=PLMp96nOrGcZezSAPW08ulofVcEZRpYfH&index=53>
* StatusStrip.
* https://www.youtube.com/watch?v=erpK3UtIn3U&list=PLMp96nOrGcZezSAPW08ulofVcEZRpYfH&index=18

1. Agregar al proyecto el formulario frmLogin con el diseño que se muestra a continuación. Este formulario será el primero que se muestre. Al presionar el botón aceptar:
   1. Si las credenciales (usuario y contraseña) son correctas deberá abrirse el formulario MDI y cerrar el formulario frmLogin.
   2. Si no son correctas deberá mostrar el mensaje “Usuario y/o Contraseña incorrectas”



1. Agregar dos formularios adicionales que se puedan abrir (en forma No Modal) desde el menú y desde los botones de la barra de herramientas.

.