UNIVERSIDAD ANDINA DEL CUSCO

FACULTAD DE INGENIERIA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERIA AMBIENTAL

TEMA:

APLIACIONES WINDOWS FORMS C#

ASIGNATURA: LENGUAJE DE PROGRAMACION

DOCENTE: Ing. DEL CARPIO CUENTAS, Luis Enrique

ALUMNAS: ESCALANTE AYRAMPO, Rogers

NAVA AGUILAR, Anthony

TACUSI NINAYA, Nilo

CUSCO – PERU

2018

Contenido

INTRODUCCION

Windows Forms es una tecnología de smart client para .NET Framework, un conjunto de bibliotecas administradas que simplifican las tareas comunes de las aplicaciones, como leer y escribir en el sistema de archivos. Cuando se usa un entorno de desarrollo como Visual Studio, puede crear aplicaciones smart client de Windows Forms que muestran información, pedir la opinión de los usuarios y se comunican con equipos remotos a través de una red.

En Windows Forms, un formulario es una superficie visual en la que se muestra información al usuario. Normalmente, las aplicaciones de Windows Forms se compilan mediante la adición de controles a los formularios y el desarrollo de respuestas a las acciones del usuario, como clics del mouse o presiones de teclas. Un control es un elemento de interfaz de usuario (UI) discreto que muestra datos o acepta la entrada de datos.

Cuando un usuario realiza una acción en un formulario o en uno de sus controles, la acción genera un evento. La aplicación reacciona a estos eventos mediante código y procesa los eventos cuando se producen. .

Windows Forms contiene diversos controles que puede agregar a los formularios: controles que muestran cuadros de texto, botones, cuadros desplegables, botones de radio e incluso páginas web. Para obtener una lista de todos los controles que puede usar en un formulario, consulte el artículo sobre controles que se utilizan en formularios Windows Forms. Si un control existente no satisface sus necesidades, Windows Forms también permite crear controles personalizados mediante la clase User Control.

Windows Forms tiene controles de interfaz de usuario enriquecidos que emulan las características de las aplicaciones de tecnología avanzada como Microsoft Office. Los controles ToolStrip y MenuStrip le permiten crear barras de herramientas y menús que contienen texto e imágenes, muestran submenús y hospedan otros controles como cuadros de texto y cuadros combinados.

Con el Diseñador de formularios de Windows de arrastrar y colocar de Visual Studio, puede crear fácilmente aplicaciones de Windows Forms. Simplemente seleccione los controles con el cursor y agréguelos donde desee en el formulario. El diseñador proporciona herramientas como líneas de cuadrícula y líneas de ajuste para minimizar la molestia de alinear los controles. Y si usa Visual Studio o compilar en la línea de comandos, se puede utilizar el Flow Layout Panel, Table Layout Panel y Split Container diseños de formularios de controles para crear avanzados en menos tiempo.

Por último, si debe crear sus propios elementos de interfaz de usuario personalizados, el espacio de nombres System. Drawing contiene una gran selección de clases para representar lineal, círculos y otras formas directamente en un formulario.

1. WINDOWS FORMS PARA C#

1.1 DEFINICION

Windows Forms es una de las dos tecnologías que se utiliza en Visual C# para crear aplicaciones cliente inteligentes basadas en Windows que se ejecutan en .NET Framework. Windows Forms es especialmente adecuado para escenarios de desarrollo rápido de aplicaciones donde la prioridad principal no es una interfaz gráfica de usuario compleja. El Diseñador de Windows Forms se utiliza para crear la interfaz de usuario, y permite obtener acceso a otras características de diseño y ejecución, tales como las siguientes:

- Implementación ClickOnce.
- Uso avanzado de bases de datos mediante DataGridView (Control, formularios Windows Forms).
- Barras de herramientas y otros elementos de interfaz de usuario que pueden tener el aspecto y el comportamiento de Microsoft Windows XP, Microsoft Office o Microsoft Internet Explorer.

En Visual C#, podemos utilizar el Diseñador de Windows Forms o el Diseñador de Windows Presentation Foundation (WPF) para crear interfaces de usuario de forma rápida y cómoda.

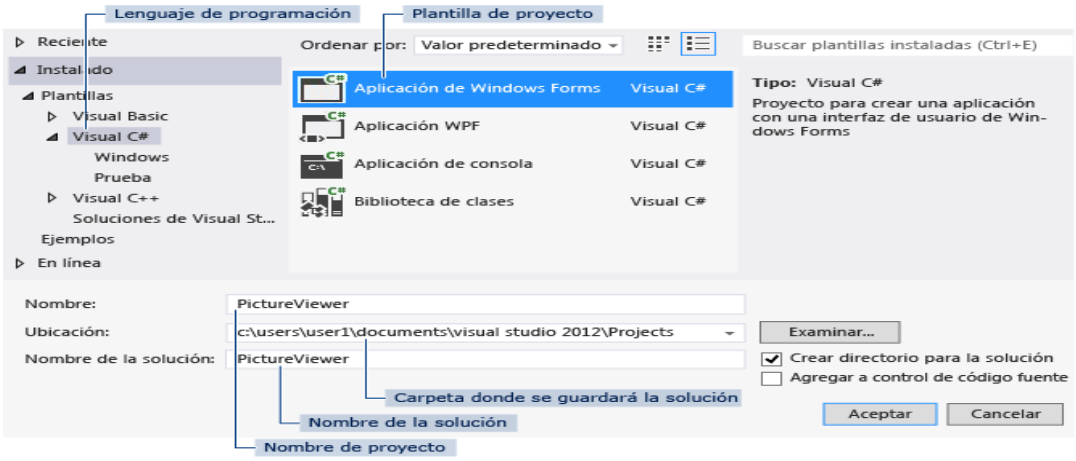
Los tres pasos básicos para crear interfaces de usuario son:

- Agregar los controles a la superficie de diseño.
- Establecer las propiedades iniciales de los controles.
- Escribir los controladores para los eventos especificados.

1.2 ACCEDE PARA CREAR UN FORMS EN C#

Para acceder a crear un forms en C# se sigue los siguientes pasos:

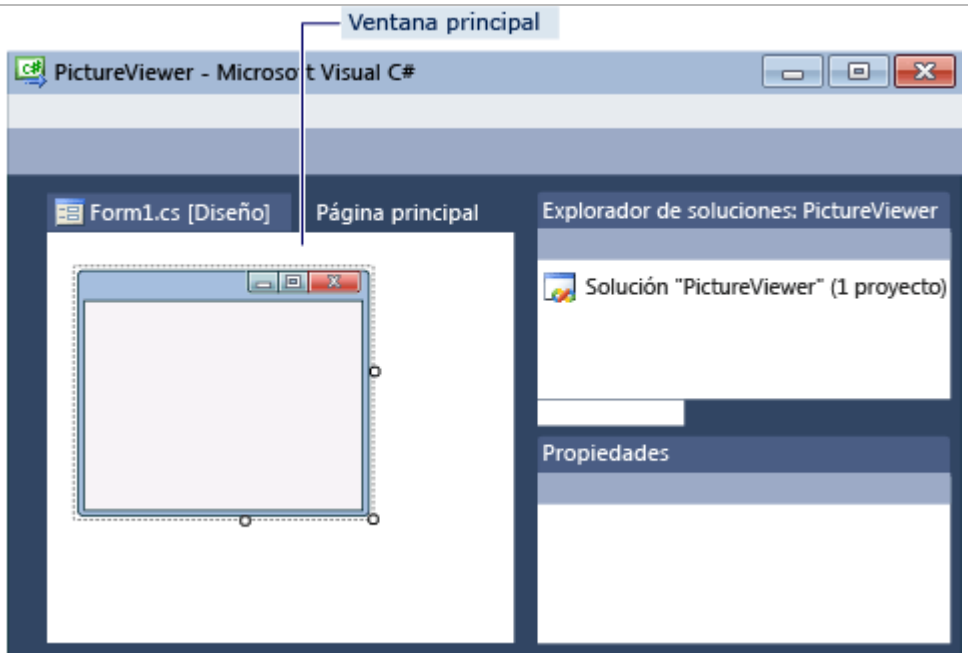
1. En la barra de menús, elija **Archivo**, **Nuevo**, **Proyecto**. El cuadro de diálogo debe tener un aspecto similar al que se muestra a continuación.



Cuadro de diálogo Nuevo proyecto

2. Elija **Visual C#** o **Visual Basic** en la lista **Plantillas instaladas**.
3. En la lista de plantillas, elija el icono **Aplicación de Windows Forms**. Asigne un nombre al nuevo formulario PictureViewer y, a continuación, elija el botón **Aceptar**.Visual Studio crea una solución para el programa. Una solución actúa como un contenedor de todos los proyectos y archivos necesarios para el programa. Estos términos se explicarán con más detalle en secciones posteriores de este tutorial.
4. En la ilustración siguiente se muestra lo que debería aparecer ahora en la interfaz de Visual Studio.

> **Nota**
>
> El diseño de la ventana puede no ser exactamente igual que el de la ilustración. El diseño exacto de la ventana depende de la versión de Visual Studio, el lenguaje de programación utilizado y otros factores. No obstante, debe comprobar que aparezcan las tres ventanas.



Ventana del IDE

5. La interfaz contiene tres ventanas: una ventana principal, el **Explorador de soluciones** y la ventana **Propiedades**.
6. Si falta alguna de estas ventanas, restaure el diseño de ventana predeterminado; para ello, en la barra de menús, elija **Ventana**, **Restablecer diseño de la ventana**. También puede mostrar ventanas mediante comandos de menú. En la barra de menús, elija **Ver**, **Ventana**

Propiedadeso **Explorador de soluciones**. Si hay otras ventanas abiertas, ciérrelas eligiendo el botón **Cerrar** (x) de la esquina superior derecha.

7. La ilustración se muestran las ventanas siguientes (en el sentido de las agujas del reloj desde la esquina superior izquierda):
   - **Ventana principal** En esta ventana, realizará la mayor parte del trabajo, como ejecutar formularios y editar código. En la ilustración, la ventana muestra un formulario en el Editor de formularios. En la parte superior de la ventana, aparecerán las pestañas **Página principal**y **Form1.cs [Diseño]**. (En Visual Basic, el nombre de la pestaña termina con .vb en lugar de .cs).
   - **Ventana Explorador de soluciones** En esta ventana, puede ver todos los elementos de la solución y navegar por ellos. Al elegir un archivo, cambia el contenido de la ventana **Propiedades**. Si abre un archivo de código (que finaliza en .cs para Visual C# y en .vb para Visual Basic), aparece el archivo de código o un diseñador para él. Un diseñador es una superficie visual en la que se pueden agregar controles, como botones y listas. En los formularios de Visual Studio, el diseñador se llama Diseñador de Windows Forms.
   - **Ventana Propiedades** En esta ventana, puede cambiar las propiedades de los elementos elegidos en las otras ventanas. Por ejemplo, si elige Form1, puede cambiar el título estableciendo la propiedad **Text** y el color de fondo estableciendo la propiedad **Backcolor**.

> **Nota**
>
> En la línea superior del **Explorador de soluciones** se muestra **Solución "PictureViewer" (1 proyecto)**, lo que significa que Visual Studio ha creado una solución automáticamente. Una solución puede contener varios proyectos, pero, por ahora, trabajará con soluciones que contengan un solo proyecto.

8. En la barra de menús, elija **Archivo**, **Guardar todo**.Si lo desea, también puede elegir el botón **Guardar todo** de la barra de herramientas que se muestra en la ilustración siguiente.


Botón de la barra de herramientas Guardar todo

Visual Studio rellena automáticamente el nombre de la carpeta y el nombre del proyecto, y después guarda el proyecto en la carpeta de proyectos.

1.3 QUE Y CUALES SON LOS CONTROLES EN FORMS

Los controles comunes de Visual Basic se encuentran dentro del archivo ejecutable de Visual Basic. Los controles intrínsecos están siempre incluidos en el cuadro de herramientas, no como los

controles ActiveX y los objetos insertables, que se pueden quitar o agregar al cuadro de herramientas.

- **Label**

Un control Label es un control gráfico que puede usar para mostrar texto que el usuario no podrá cambiar directamente.

Puede escribir código que cambie el texto mostrado por un control Label como respuesta a eventos en tiempo de ejecución. Por ejemplo, si la aplicación tarda unos minutos en realizar un cambio, puede mostrar un mensaje de estado del proceso en un control Label. También puede usar Label para identificar un control, como un control TextBox, que no tenga su propia propiedad Caption.

Establezca las propiedades AutoSize y WordWrap si desea que Label muestre líneas de longitud variable o números variables de líneas.

- **TextBox (Caja de texto)**

Un control TextBox, también denominado control de campo de edición o control de edición, muestra información introducida en tiempo de diseño, introducida por el usuario o asignada al control en código en tiempo de ejecución.

Para mostrar múltiples líneas de texto en un control TextBox, establezca la propiedad MultiLine a True. Si un TextBox de múltiples líneas no tiene una barra de desplazamiento horizontal, el texto se ajustará automáticamente aunque se ajuste el tamaño de TextBox. Para personalizar la combinación de barra de desplazamiento en un TextBox, establezca la propiedad ScrollBars.

Las barras de desplazamiento siempre aparecerán en el TextBox cuando su propiedad MultiLine se haya establecido a True y su propiedad ScrollBars a cualquier cosa excepto None (0). Si establece la propiedad MultiLine a True, podrá emplear la propiedad Alignment para establecer la alineación del texto dentro del TextBox. El texto está justificado a la izquierda de forma predeterminada. Si la propiedad MultiLine es False, el valor de la propiedad Alignment no tiene ningún efecto.

- **CommandButton (Botón de comando)**

Utilice un control CommandButton para comenzar, interrumpir o terminar un proceso. Cuando está activado, un CommandButton aparece presionado y a veces se denomina botón de comando.

Para mostrar texto en un control CommandButton , establezca su propiedad Caption . Un usuario siempre puede elegir un CommandButton si hace clic en él. Para permitir que el usuario lo elija presionando la tecla ENTRAR, establezca la propiedad Default a True.

- **Picture Box (imagen)**

Un control PictureBox puede mostrar un gráfico a partir de un mapa de bits, un icono o un metarchivo, así como un metarchivo mejorado, un archivo JPEG o un archivo GIF. Recorta el gráfico si el control no es lo bastante grande para mostrar la imagen completa.

También puede utilizar un control PictureBox para agrupar controles OptionButton y mostrar el resultado de métodos gráficos y texto escrito con el método Print.

Para hacer que un control PictureBox ajuste automáticamente su tamaño para mostrar un gráfico completo, establezca su propiedad AutoSize a True.

- **Listas (ListBox)**

Una lista es un control que pone a disposición del usuario un conjunto de elementos, de los cuales elegirá uno. Si la cantidad de elementos rebasa el número de los que pueden ser visualizados simultáneamente en el espacio disponible en la lista, aparecerá automáticamente una barra de desplazamiento vertical para que el usuario pueda desplazar los elementos de la lista hacia arriba o hacia abajo. Generalmente, una lista es apropiada cuando se quiere limitar la entrada a una serie de elementos determinados.

Por omisión, los elementos de una lista son visualizados verticalmente en una a columna, aunque usted puede, si quiere, establecer múltiples columnas (sólo a en listas simples no en listas desplegables, propiedad Columnas). Si el ancho total las columnas supera el ancho de la lista, aparecerá automáticamente una barra de desplazamiento horizontal para que el usuario pueda desplazar las columnas hacia la izquierda o hacia la derecha.

Una lista desplegable es un control que combina las características de una caja de texto y de una lista. Esto permite al usuario elegir un elemento de varios, escribiéndolo directamente en la caja de texto o seleccionándolo de la lista. Generalmente, un control combinado es apropiado cuando hay una lista de elemento sugeridos; además, el usuario puede introducir otros que no estén en la lista.

1.4 QUE SON LAS PROPIEDADES DE LOS CONTROLES

Después de agregar un control a la superficie de diseño, podemos utilizar la ventana Propiedades para establecer sus propiedades, como son el color de fondo y el texto predeterminado.

En el Diseñador de Windows Forms, los valores que especifique en la ventana Propiedades son los valores iniciales que se asignarán a la propiedad cuando se cree el control en tiempo de ejecución. En el Diseñador de WPF, los valores que especifique en la ventana Propiedades se almacenan como atributos en el archivo XAML de la ventana.

En muchos casos, se puede tener acceso a estos valores o modificarlos mediante programación en tiempo de ejecución; para ello, basta con obtener o establecer desde la aplicación la propiedad para la instancia de la clase del control. La ventana Propiedades resulta útil en tiempo de diseño porque permite examinar todas las propiedades, eventos y métodos que admite un control.

1.5 EVENTOS Y MANEJO CON LOS CONTROLES

Cuando ocurre algo interesante, los eventos habilitan una clase u objeto para notificarlo a otras clases u objetos. La clase que envía (o genera) el evento recibe el nombre de publicador y las clases que reciben (o controlan) el evento se denominan suscriptores.

En una aplicación web o una aplicación de Windows Forms en C# típica, se puede suscribir a eventos generados por controles, como botones y cuadros de lista. Puede usar el entorno de desarrollo integrado (IDE) de Visual C# para examinar los eventos que publica un control y seleccionar los que quiera administrar. El IDE agrega automáticamente un método de controlador de eventos vacío y el código para suscribirse al evento. Para obtener más información, vea How to: Subscribe to and Unsubscribe from Events (Cómo: Suscribir y cancelar la suscripción a eventos [Guía de programación de C#]).

Los eventos tienen las siguientes propiedades:

- El publicador determina el momento en el que se genera un evento; los suscriptores determinan la acción que se lleva a cabo en respuesta al evento.
- Un evento puede tener varios suscriptores. Un suscriptor puede controlar varios eventos de varios publicadores.
- Nunca se generan eventos que no tienen suscriptores.
- Los eventos se suelen usar para indicar acciones del usuario, como los clics de los botones o las selecciones de menú en las interfaces gráficas de usuario.
- Cuando un evento tiene varios suscriptores, los controladores de eventos se invocan sincrónicamente cuando se genera un evento. Para invocar eventos de forma asincrónica, consulte Calling Synchronous Methods Asynchronously.
- En la biblioteca de clases .NET Framework , los eventos se basan en el delegado EventHandler y en la clase base EventArgs .

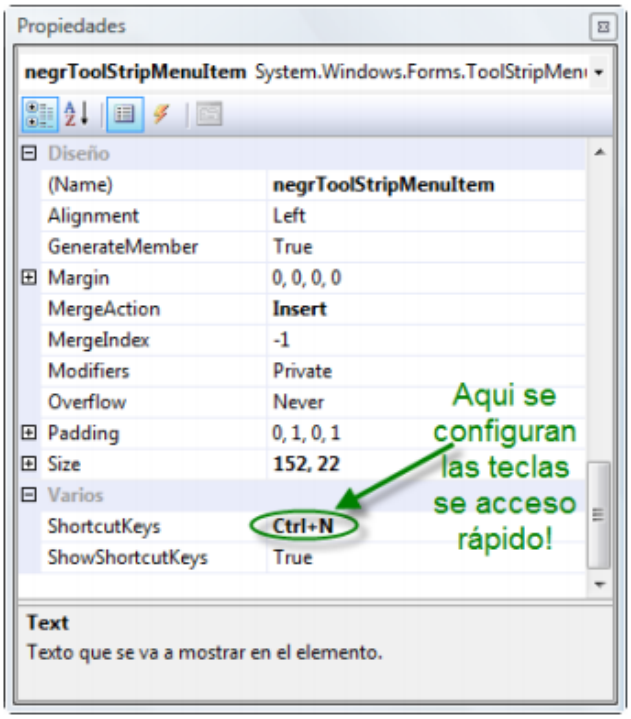
1.6 EJEMPLO EN WINDOWS FORMS C#

Creación de Menús

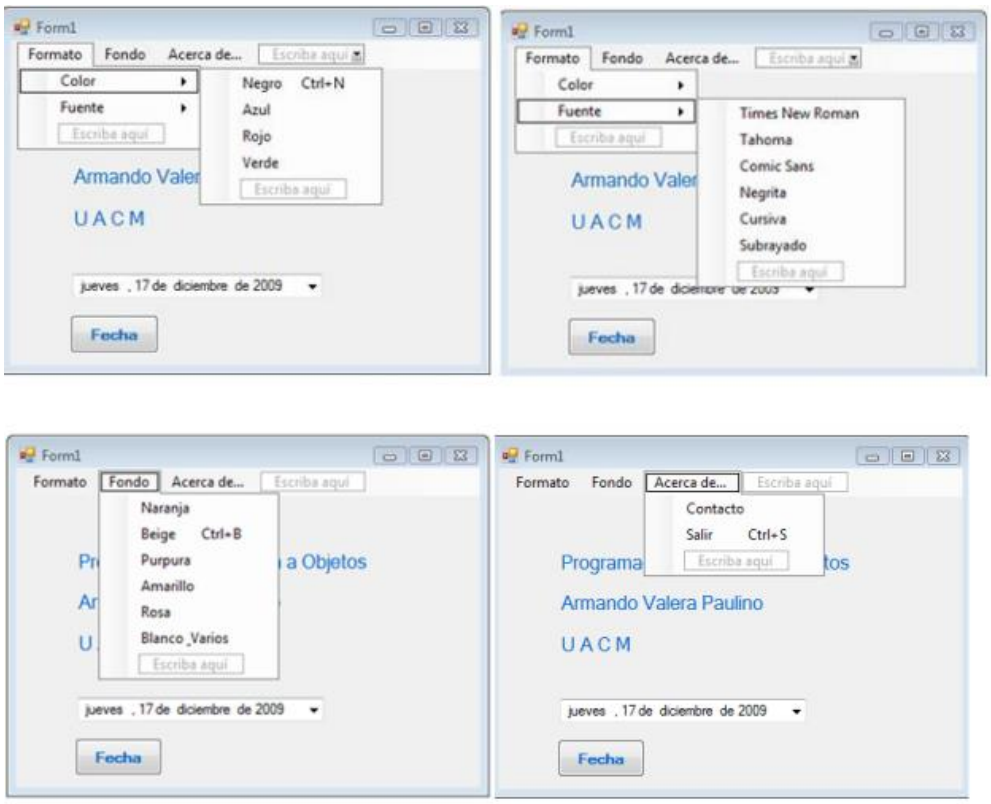
El siguiente proyecto muestra la creación de menús, utilizamos tres etiquetas y mediante las opciones del menú Formato y Fondo, cambiamos el color de fondo, el tipo y color de la fuente o letra, mediante el menú Acerca de… mostramos información de contacto del autor del proyecto y la opción Salir. Como parte complementaria y para no incluirla en un solo proyecto agregamos el control dateTimePicker y el botón Fecha, esto simplemente muestra un calendario seleccionamos una fecha y al dar clic en el botón nos muestra la fecha elegida y del día actual, esto con ayuda de un MessageBox.Show.

La estructura general del proyecto es la siguiente:

Con ingresar un solo menuStrip se pueden derivar todos los menús que necesitemos, en las siguientes imágenes se muestran cada una de las opciones del menú, es importante aclarar que la opciones que aparecen como Ctrl+N, Ctrl+B, Ctrl+S, no se escriben si no son accesos rápidos para trabajar con el proyecto por ejemplo con Ctrl+S, sale uno del proyecto, en la siguiente pantalla se muestra como se configura esta propiedad.



En las siguientes figuras se muestra a detalle cada una de las opciones del menú:

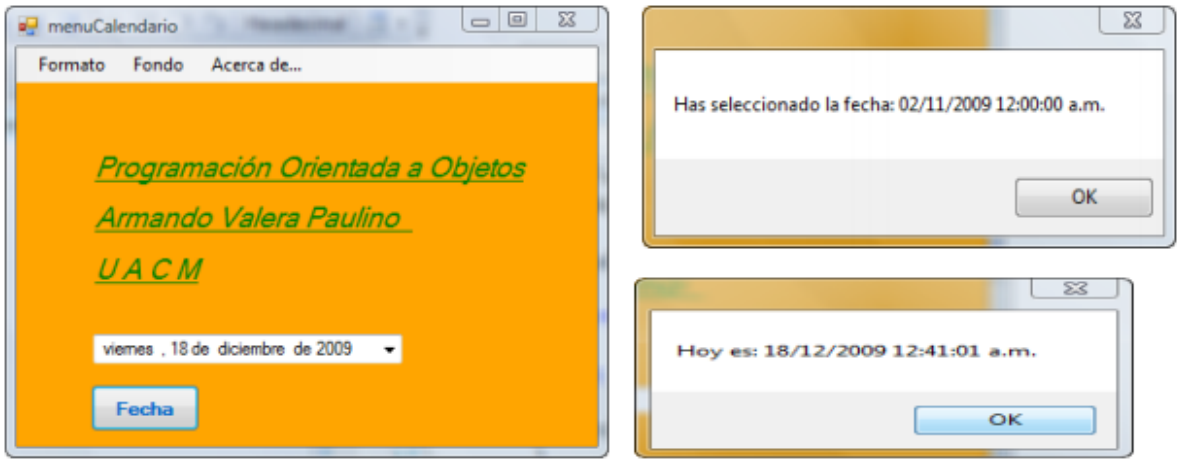


El código del botón fecha nos da la información de la fecha seleccionada y el día actual junto con la hora, su código es: private void button1_Click (object sender, EventArgs e)

```
{
// Muestra la fecha seleccionada:
MessageBox.Show("Has seleccionado la fecha: " +
dateTimePicker1.Value.Date);
// Muestra la fecha de hoy:
MessageBox.Show("Hoy es: " + DateTime.Now);
}
```

Por razones de espacio el código de cada menú puede consultarse en el apéndice 5: menuCalendario -- Form1.cs

Ejemplo de la ejecución del proyecto:



BIBLIOGRAFIA

- Capítulo 5: Windows Forms (s.f.) Recuperado de: file:///C:/Users/chelo/Downloads/windows-150922023330-lva1-app6891.pdf
- Hernandez, M.C.(s.f.) Introducción a Windows Forms en C#.NET. México. Recuperado de:http://www.itnuevolaredo.edu.mx/takeyas/Apuntes/POO/Apuntes/GuiaRapidaWindowsForms.pdf
- Microsoft (2016) Paso 1: Crear un proyecto de aplicación de Windows Forms. Recuperado de: https://msdn.microsoft.com/es-es/library/dd492132.aspx