GUI: Formularios

6.1

Programación II y Laboratorio de Computación II

Edición 2018

Windows Forms

- Windows Forms es la plataforma de desarrollo de aplicaciones para Windows basadas en el marco .NET.
- Este marco de trabajo proporciona un conjunto de clases que permite desarrollar complejas aplicaciones para Windows.
- Las clases se exponen en el NameSpace System.Windows.Forms y NameSpaces asociados.
- Es posible crear clases propias que hereden de algunas de las anteriores.

Formularios

- Un formulario Windows Forms actúa como interfaz del usuario local de Windows.
- Los formularios pueden ser ventanas estándar, interfaces de múltiples documentos (MDI), cuadros de diálogo, etc.
- Los formularios son objetos que exponen propiedades, métodos que definen su comportamiento y eventos que definen la interacción con el usuario.

Formularios

- Al momento de diseñar un formulario, el diseñador de Visual Studio escribe de forma automática el código que describe a cada uno de los controles y al propio formulario.
- El concepto de Partial Class, que se incorpora en .NET 2.0, permite separar el código de una clase en dos archivos fuentes diferentes.
- El diseñador de formularios utiliza esta técnica para escribir en un archivo aparte todo el código que él mismo genera.
- Esto permite organizar más claramente el código, manteniendo separada la lógica de la aplicación en un archivo diferente.

Objeto Form

- El objeto Form es el principal componente de una aplicación Windows.
- Algunas de sus propiedades admiten valores de alguno de los tipos nativos (por valor) de .NET.

```
miForm.ShowInTaskBar = false; // Mostrar en barra de tareas
miForm.Text = "Un Formulario"; // Título
miForm.Opacity = 0.85; // Opacidad para transparencia
```

- Otras propiedades requieren la asignación de objetos.

```
miForm.Size = new Size(100, 100); // Tamaño en ancho y alto
miForm.Location = new Point(0, 0); // Ubicación en X e Y
miForm.Font = new Font("Arial", 10); // Fuente y tamaño
```

Objeto Form

- Show()
  - Visualiza el formulario. Puede especificarse su formulario Owner (dueño o propietario).
- ShowDialog()
  - Puede utilizar este método para mostrar un cuadro de diálogo modal en la aplicación.
- Close()
  - Cierra el formulario.
- Hide()
  - Oculta el formulario del usuario.

Ciclo de Vida

- **Muchos de los eventos a los que responde el objeto *Form* pertenecen al ciclo de vida del formulario.**

New	**Se crea la instancia del formulario**
Load	El formulario está en memoria, pero invisible
Paint	Se dibuja el formulario y sus controles
Activated	El formulario recibe foco
FormClosing	Permite cancelar el cierre
FormClosed	**El formulario ya es invisible**
Disposed	El objeto está siendo destruido

MessageBox

- Para mostrar información o pedir intervención del usuario, es posible utilizar la clase MessageBox.
- Esta clase contiene métodos estáticos que permiten mostrar un cuadro de mensaje para interactuar con el usuario de la aplicación.
- Los parámetros se especifican a través de enumerados que facilitan la legibilidad del código, por ejemplo:
  - MessageBoxButtons.AbortRetryIgnore
  - MessageBoxIcon.Error
  - MessageBoxDefaultButton.Button1

Propiedades

- Name
  - Indica el nombre utilizado en el código para identificar el objeto.
- BackColor
  - Indica el color de fondo del componente.
- Cursor
  - Cursor que aparece al pasar el puntero por el control.
- Enabled
  - Indica si el control esta habilitado.
- Font
  - Fuente utilizada para mostrar el texto en el control.
- ForeColor
  - Color utilizado para mostrar texto.

Propiedades

- Locked
  - Determina si se puede mover o cambiar de tamaño el control.
- Modifiers
  - Indica el nivel de visibilidad del objeto.
- TabIndex
  - Determina el índice del orden de tabulación que ocupará este control.
- Text
  - Texto asociado al control.
- Visible
  - Determina si el control esta visible u oculto.

Controles: CheckBox

- CheckBox
  - Este control muestra una casilla de verificación, que podemos marcar para establecer un estado.
  - Generalmente el estado de un CheckBox es marcado (verdadero) o desmarcado (falso), sin embargo, podemos configurar el control para que sea detectado un tercer estado, que se denomina indeterminado, en el cual, el control se muestra con la marca en la casilla pero en un color de tono gris.

Controles: RadioButton y GroupBox

- RadioButton
  - Los controles RadioButton nos permiten definir conjuntos de opciones auto excluyentes, de modo que situando varios controles de este tipo en un formulario, sólo podremos tener seleccionado uno en cada ocasión.
- GroupBox
  - Permite agrupar controles en su interior, tanto RadioButton como de otro tipo, ya que se trata de un control contenedor.

Controles: ListBox

- ListBox
  - Un control ListBox contiene una lista de valores, de los cuales, el usuario puede seleccionar uno o varios simultáneamente.

  - **Items.** Contiene la lista de valores que visualiza el control. Se trata de un tipo ListBox.ObjectCollection, de manera que el contenido de la lista puede ser tanto tipo cadena, numéricos u objetos de distintas clases.
  - **SelectionMode**. Establece el modo en el que se pueden seleccionar los elementos de la lista.

Controles: ListBox

- SelectionMode
  - **None**, no se realizará selección.
  - **One**, permite seleccionar los valores uno a uno.
  - **MultiSimple** permite seleccionar múltiples valores de la lista pero debemos seleccionarlos independientemente.
  - **MultiExtended** nos posibilita la selección múltiple, con la ventaja de que podemos hacer clic en un valor, y arrastrar, seleccionando en la misma operación varios elementos de la lista.

Controles: ComboBox

- El ComboBox es un control basado en la combinación (de ahí su nombre) de dos controles: TextBox y ListBox.
- Un control ComboBox dispone de una zona de edición de texto y una lista de valores, que se pueden desplegar desde el cuadro de edición.
- El estilo de visualización por defecto de este control, muestra el cuadro de texto y la lista oculta, aunque mediante la propiedad **DropDownStyle** se puede cambiar dicho estilo.

Controles: ComboBox

- La propiedad **DropDownStyle** también influye en una diferencia importante de comportamiento entre el estilo **DropDownList** y los demás, dado que cuando se crea un ComboBox con el mencionado estilo, el cuadro de texto sólo podrá mostrar información, no permitiendo que esta sea modificada.

- En el caso de que la lista desplegable sea muy grande, mediante la propiedad **MaxDropDownItems**, se puede asignar el número de elementos máximo que mostrará la lista del control.