

# **Programación Orientada a Objetos I**



# ÍNDICE

Presentación	5
Red de contenidos	6
<b>UNIDAD 1</b>	
<b>SEMANA 1</b> : Introducción a la arquitectura .NET Framework	7
<b>SEMANA 2</b> : Desarrollo de aplicaciones Windows y POO	25
<b>UNIDAD 2</b>	
<b>SEMANA 3</b> : Desarrollo de aplicaciones con Tipos de datos y Colecciones I	53
<b>SEMANA 4</b> : Desarrollo de aplicaciones con Colecciones II	75
<b>UNIDAD 3</b>	
<b>SEMANA 5</b> : Administración de servicios Windows	91
<b>SEMANA 6</b> : Construir aplicaciones Windows que utilicen Threads y Configuraciones	105
<b>SEMANA 7</b> : Semana de exámenes parciales teoría	
<b>SEMANA 8</b> : Semana de exámenes parciales laboratorio	
<b>SEMANA 9</b> : Implementación de Diagnosticos en .NET	121
<b>UNIDAD 4</b>	
<b>SEMANA 10</b> : Implementación de Serialización en .NET	135
<b>SEMANA 11</b> : Administración de Sistemas de Archivos	151
<b>UNIDAD 5</b>	
<b>SEMANA 12</b> : Implementación de Seguridad en Aplicaciones .NET	169
<b>SEMANA 13</b> : Interoperabilidad y Reflection	189
<b>SEMANA 14</b> : Implementación de Globalización	199
<b>SEMANA 15</b> : Examen final de laboratorio	
<b>SEMANA 16</b> : Sustentación de proyectos	
<b>SEMANA 17</b> : Examen final de teoría	



# PRESENTACIÓN

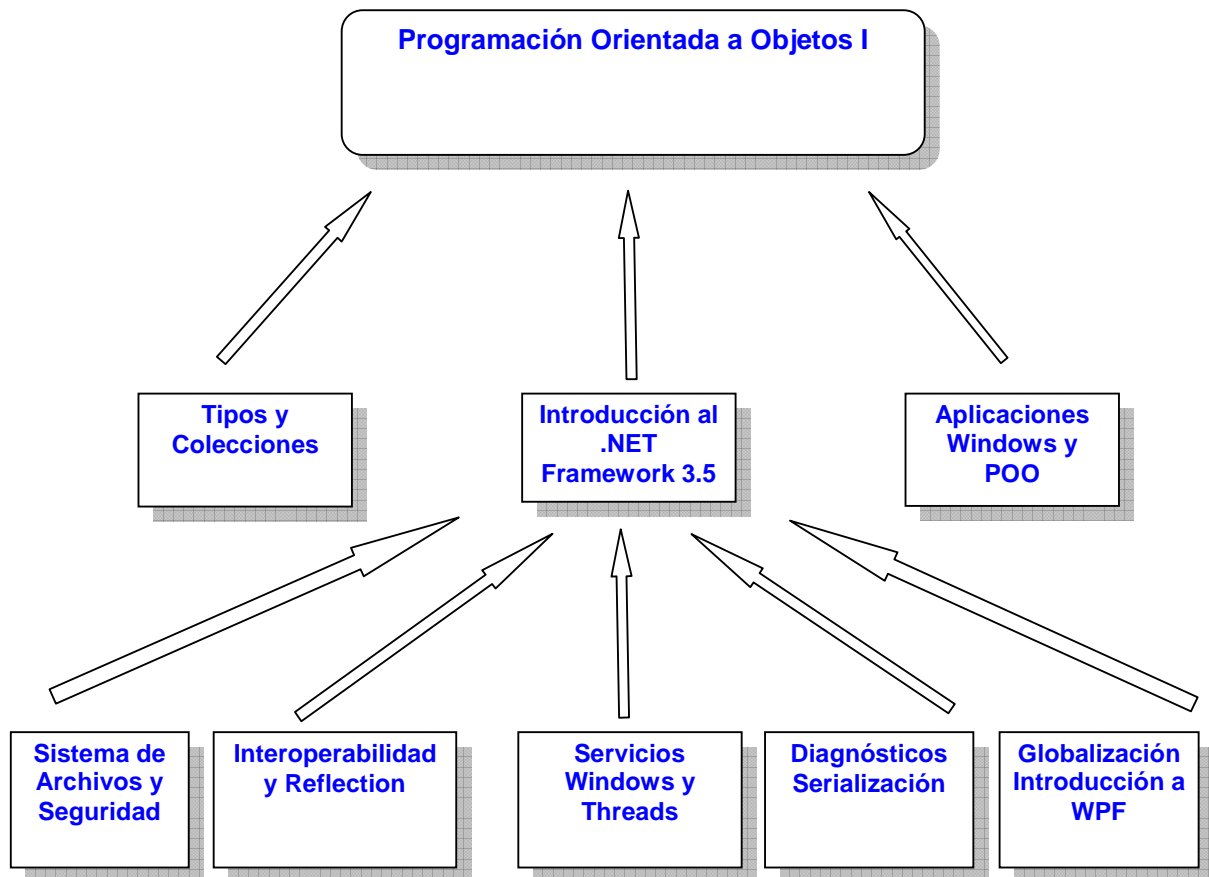
.NET es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones. Basado en ella, la empresa intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el sistema operativo hasta las herramientas de mercado.

El objetivo de .NET es obtener un entorno específicamente diseñado para el desarrollo y ejecución del software en forma de servicios que puedan ser tanto publicados como accedidos a través de Internet de forma independiente del lenguaje de programación, modelo de objetos, sistema operativo y hardware utilizados tanto para desarrollarlos como para publicarlos. Éste entorno es lo que se denomina la plataforma.NET, y los servicios antes mencionados son a los que se denomina servicios web.

El manual para el curso de POOI ha sido elaborado bajo unidades de aprendizaje, las que desarrollamos durante semanas determinadas. Así mismo, los temas que presenta este manual son parte del temario para rendir el examen de Microsoft 70 - 536 que es el requisito para obtener una serie de certificaciones como MCTS Windows, MCTS ADO.NET, MCTS Web.

La distribución de temas para este manual empieza por una introducción a la plataforma .NET. Luego, se va desarrollando los temas como Tipos de Datos y Colecciones, Servicios de Windows, Diagnosticos, Procesos, Manejo de archivos, Serialización, Seguridad, Interoperabilidad, Reflection y por último Globalización.

## RED DE CONTENIDOS



**UNIDAD DE  
APRENDIZAJE****1****SEMANA****1**

## **INTRODUCCIÓN A LA ARQUITECTURA .NET FRAMEWORK 3.5**

---

### **LOGRO DE LA UNIDAD DE APRENDIZAJE**

Al término de la unidad, los alumnos describen el funcionamiento de una aplicación .NET bajo la arquitectura del Framework .NET 2.0 explicando los elementos necesarios que se utilizan en la elaboración de esta aplicación.

### **TEMARIO**

- Arquitectura del .Net Framework, definición
- Manejo de clases y sus miembros
- Creación de proyectos Windows Application

### **ACTIVIDADES PROPUESTAS**

- Los alumnos reconocen el entorno de Visual Studio 2008
- Los alumnos crean aplicaciones Windows con Visual Studio 2008
- Los alumnos ejecutan los formularios de las aplicaciones creadas.

## 1. Microsoft .NET

Es el conjunto de nuevas tecnologías en las que Microsoft ha estado trabajando durante los últimos años con el objetivo de obtener una plataforma sencilla y potente para distribuir el software en forma de servicios que puedan ser suministrados remotamente y que puedan comunicarse y combinarse unos con otros de manera totalmente independiente de la plataforma, lenguaje de programación y modelo de componentes con los que hayan sido desarrollados. Ésta es la llamada **plataforma .NET**, y a los servicios antes comentados se les denomina **servicios Web**.

Para crear aplicaciones para la plataforma .NET, tanto servicios Web como aplicaciones tradicionales (aplicaciones de consola, aplicaciones de ventanas, servicios de Windows NT, etc.), Microsoft ha publicado el denominado kit de desarrollo de software conocido como **.NET Framework SDK**, que incluye las herramientas necesarias tanto para su desarrollo como para su distribución y ejecución y **Visual Studio.NET**, que permite hacer todo lo anterior desde una interfaz visual basada en ventanas.

## 2. .NET Framework

.NET Framework es un componente integral de Windows que admite la creación y la ejecución de la siguiente generación de aplicaciones y servicios Web XML. El diseño de .NET Framework está enfocado a cumplir los objetivos siguientes:

- Proporcionar un entorno coherente de programación orientada a objetos, en el que el código de los objetos se pueda almacenar y ejecutar de forma local, ejecutar de forma local pero distribuida en Internet o ejecutar de forma remota.
- Proporcionar un entorno de ejecución de código que reduzca lo máximo posible la implementación de software y los conflictos de versiones.
- Ofrecer un entorno de ejecución de código que fomente la ejecución segura del mismo, incluso del creado por terceras personas desconocidas o que no son de plena confianza.
- Proporcionar un entorno de ejecución de código que elimine los problemas de rendimiento de los entornos en los que se utilizan secuencias de comandos o intérpretes de comandos.
- Ofrecer al programador una experiencia coherente entre tipos de aplicaciones muy diferentes, como las basadas en Windows o en el Web.
- Basar toda la comunicación en estándares del sector para asegurar que el código de .NET Framework se puede integrar con otros tipos de código.

.NET Framework contiene dos componentes principales: Common Language Runtime y la biblioteca de clases de .NET Framework.

### a. Common Language Runtime CLR

Es el fundamento de .NET Framework. El motor en tiempo de ejecución se puede considerar como un agente que administra el código en tiempo de ejecución y proporciona servicios centrales, como la administración de



memoria, la administración de subprocesos y la interacción remota, al tiempo que aplica una seguridad estricta a los tipos y otras formas de especificación del código que fomentan su seguridad y solidez. De hecho, el concepto de administración de código es un principio básico del motor en tiempo de ejecución. El código destinado al motor en tiempo de ejecución se denomina código administrado, a diferencia del resto de código, que se conoce como código no administrado

### **Características de Common Language Runtime**

Common Language Runtime administra la memoria, ejecución de subprocesos, ejecución de código, comprobación de la seguridad del código, compilación y demás servicios del sistema. Estas características son intrínsecas del código administrado que se ejecuta en Common Language Runtime.

Con respecto a la seguridad, los componentes administrados reciben grados de confianza diferentes, en función de una serie de factores entre los que se incluye su origen (como Internet, red empresarial o equipo local). Esto significa que un componente administrado puede ser capaz o no de realizar operaciones de acceso a archivos, operaciones de acceso al Registro y otras funciones delicadas, incluso si se está utilizando en la misma aplicación activa. Por ejemplo, los usuarios pueden confiar en que un archivo ejecutable incrustado en una página Web puede reproducir una animación en la pantalla o entonar una canción, pero no puede tener acceso a sus datos personales, sistema de archivos o red.

Además, ofrece la solidez del código mediante la implementación de una infraestructura estricta de comprobación de tipos y código denominada CTS (Common Type System, Sistema de tipos común). CTS garantiza que todo el código administrado es autodescriptivo. Los diferentes compiladores de lenguajes de Microsoft y de terceros generan código administrado que se ajusta a CTS. Esto significa que el código administrado puede usar otros tipos e instancias administrados, al tiempo que se aplica inflexiblemente la fidelidad y seguridad de los tipos.

Así mismo, el entorno administrado del motor en tiempo de ejecución elimina muchos problemas de software comunes. Por ejemplo, el motor en tiempo de ejecución controla automáticamente la disposición de los objetos, administra las referencias a éstos y los libera cuando ya no se utilizan. Esta administración automática de la memoria soluciona los dos errores más comunes de las aplicaciones: la pérdida de memoria y las referencias no válidas a la memoria.

Finalmente, el motor en tiempo de ejecución está diseñado para mejorar el rendimiento. Aunque Common Language Runtime proporciona muchos servicios estándar de motor en tiempo de ejecución, el código administrado nunca se interpreta. Una característica denominada compilación JIT (Just-In-Time) permite ejecutar todo el código administrado en el lenguaje máquina nativo del sistema en el que se ejecuta. Mientras tanto, el administrador de memoria evita que la memoria se pueda fragmentar y aumenta la zona de referencia de la memoria para mejorar aún más el rendimiento.

## **b. Biblioteca de clases de .NET Framework Class Library**

La biblioteca de clases de .NET Framework es una colección de tipos reutilizables que se integran estrechamente con Common Language Runtime. La biblioteca de clases está orientada a objetos, lo que proporciona tipos de los que su propio código administrado puede derivar funciones. Esto ocasiona que los tipos de .NET Framework sean sencillos de utilizar y reduce el tiempo asociado con el aprendizaje de las nuevas características de .NET Framework. Además, los componentes de terceros se pueden integrar sin dificultades con las clases de .NET Framework.

Como en cualquier biblioteca de clases orientada a objetos, los tipos de .NET Framework permiten realizar diversas tareas de programación comunes, como son la administración de cadenas, recolección de datos, conectividad de bases de datos y acceso a archivos. Además de estas tareas habituales, la biblioteca de clases incluye tipos adecuados para diversos escenarios de desarrollo especializados. Por ejemplo, puede utilizar .NET Framework para desarrollar los siguientes tipos de aplicaciones y servicios:

- Aplicaciones de consola
- Aplicaciones GUI de Windows (Windows Forms).
- Aplicaciones de Windows Presentation Foundation (WPF).
- Aplicaciones de ASP.NET
- Servicios web.
- Servicios de Windows
- Aplicaciones orientadas a servicios utilizando Windows Communication Foundation (WCF).
- Aplicaciones habilitadas para el flujo de trabajo utilizando Windows Workflow Foundation (WF).


## **3. Arquitectura del .NET Framework**

.NET Framework versión 3.5 se basa en las versiones 2.0 y 3.0 y sus Service Pack correspondientes. .NET Framework versión 3.5 Service Pack 1 actualiza los ensamblados de la versión 3.5 e incluye nuevos Service Pack para las versiones 2.0 y 3.0. En este tema se resume brevemente la relación de las versiones 2.0, 3.0 y 3.5 de .NET Framework y sus Service Packs.


Los componentes que se enumeran a continuación se consideran parte de .NET Framework 3.5 SP1:

- .NET Framework 2.0
- Service Pack 1 y 2 de .NET Framework 2.0, que actualizan los ensamblados incluidos en .NET Framework 2.0.

- .NET Framework 3.0, que utiliza los ensamblados de .NET Framework 2.0 y sus Service Pack, e incluye los ensamblados necesarios para las tecnologías introducidas en .NET Framework 3.0. Por ejemplo, PresentationFramework.dll y PresentationCore.dll, que son necesarios para Windows Presentation Foundation (WPF), se instalan con .NET Framework 3.0.
- Service Pack 1 y 2 de .NET Framework 3.0, que actualizan los ensamblados que se introducen en .NET Framework 3.0.
- .NET Framework 3.5, que incluye nuevos ensamblados que proporcionan una funcionalidad adicional a .NET Framework 2.0 y 3.0.
- .NET Framework 3.5 Service Pack 1, que actualiza los ensamblados que se incluyen en .NET Framework 3.5.

 **Nota:** Windows Vista no admite la instalación independiente de las versiones 2.0 SP2 y 3.0 SP2 de .NET Framework. Windows 2000 no admite la versión 3.5 SP1 de .NET Framework ni la instalación independiente de .NET Framework versión 3.0 SP 2.

Una aplicación utiliza los mismos ensamblados sin tener en cuenta si tiene como destino la versión 2.0, 3.0 o 3.5 de .NET Framework o Client Profile, y sin tener en cuenta si los ensamblados se han actualizado en el equipo del usuario. Por ejemplo, una aplicación que utiliza WPF y tiene como destino .NET Framework 3.0 utiliza la misma instancia del ensamblado mscorlib que una aplicación que utiliza formularios Windows Forms y tiene como destino .NET Framework 2.0. Si un usuario ha instalado una versión posterior de .NET Framework o un Service Pack que actualiza su copia de mscorlib.dll, las dos aplicaciones utilizarán la versión actualizada del ensamblado.

 **Nota:** La relación entre las versiones 2.0, 3.0 y 3.5 de .NET Framework es diferente a la relación que existe entre las versiones 1.0, 1.1 y 2.0 de .NET Framework, que son totalmente independientes unas de otras, por lo que una versión puede estar en un equipo con independencia de si las otras versiones se encuentran o no en dicho equipo. Cuando las versiones 1.0, 1.1 y 2.0 están en el mismo equipo, cada versión tiene su propio Common Language Runtime, sus propias bibliotecas de clases, su propio compilador, etc. Los programadores de las aplicaciones pueden elegir qué versión quieren utilizar como destino. Para obtener más información, vea Ejecución simultánea, Especificar una versión concreta de .NET Framework e Especificar versiones concretas de .NET Framework con MSBuild.

### Características incluidas en .NET Framework 3.5 SP1

En esta sección se resumen las tecnologías de .NET Framework versión 3.5 SP1 y cada una de las versiones de .NET Framework que contiene. Esta lista no es exhaustiva, solo incluye algunas de las principales tecnologías que se incluyen en .NET Framework.

## **.NET Framework 3.5**

.NET Framework 3.5 introduce nuevas características para las tecnologías de las versiones 2.0 y 3.0 e incorpora tecnologías adicionales en forma de nuevos ensamblados. Las tecnologías siguientes se introducen en .NET Framework 3.5:

- Language Integrated Query (LINQ).
- Nuevos compiladores para C#, Visual Basic y C++.
- ASP.NET AJAX.

## **.NET Framework 3.5 SP1**

.NET Framework 3.5 Service Pack 1 actualiza varios ensamblados que se suministran con .NET Framework 3.5. Las actualizaciones incorporan cambios intrascendentes, nuevos elementos de API y una funcionalidad adicional para las tecnologías que se incluyen en .NET Framework 3.5. Las tecnologías siguientes se suministran con .NET Framework 3.5 SP1:

- Datos dinámicos de ASP.NET.
- ADO.NET Entity Framework.
- Compatibilidad con el proveedor de datos de SQL Server 2008.
- Compatibilidad con .NET Framework Client Profile.

## **Proceso de ejecución administrada**

El proceso de ejecución administrada incluye los pasos siguientes:

1. Elegir un compilador.

Para obtener los beneficios que proporciona Common Language Runtime, se deben utilizar uno o más compiladores de lenguaje orientados al tiempo de ejecución.

2. Compilar el código a Lenguaje intermedio de Microsoft (MSIL).

La compilación convierte el código fuente en MSIL y genera los metadatos requeridos.

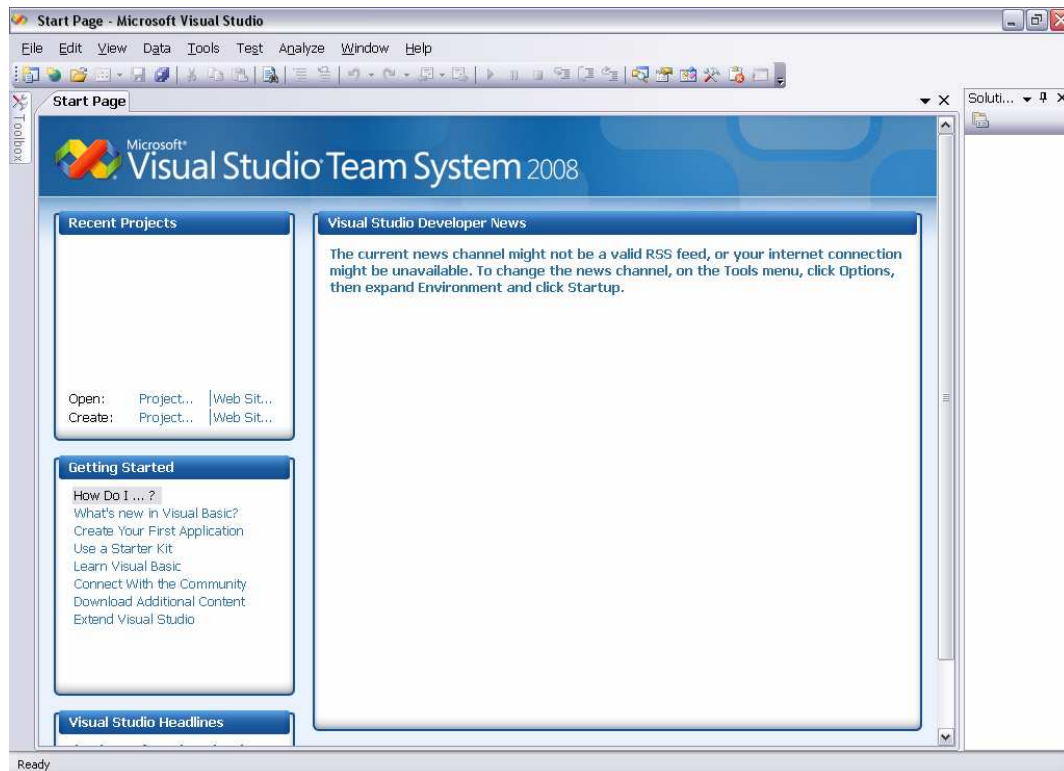
3. Compilar MSIL a código nativo.

En tiempo de ejecución, un compilador Just-In-Time (JIT) convierte MSIL en código nativo. Durante esta compilación, el código debe pasar un proceso de comprobación que examina el MSIL y los metadatos para averiguar si el código garantiza la seguridad de tipos.

4. Ejecutar código.

Common Language Runtime proporciona la infraestructura que permite que la ejecución tenga lugar, así como una amplia gama de servicios que se pueden utilizar durante la ejecución.

## 4. Entorno de Desarrollo Visual Studio 2008



## 5. Introducción a soluciones, proyectos y elementos

Visual Studio dispone de dos contenedores que le ayudan a administrar eficazmente los elementos necesarios para el desarrollo, como referencias, conexiones de datos, carpetas y archivos. Estos contenedores se denominan soluciones y proyectos. Asimismo, Visual Studio proporciona carpetas de soluciones para organizar proyectos relacionados en grupos y, a continuación, llevar a cabo acciones en esos grupos de proyectos. El Explorador de soluciones, una interfaz para ver y administrar estos contenedores y sus elementos asociados, forma parte del entorno de desarrollo integrado (IDE).

Contenedores: soluciones y proyectos

Las soluciones y los proyectos contienen elementos en forma de referencias, conexiones de datos, carpetas y archivos necesarios para crear la aplicación. Una solución puede contener varios proyectos y un proyecto normalmente contiene varios

elementos. Estos contenedores permiten sacar partido del IDE mediante las siguientes tareas:

- Administrar la configuración de la solución en su totalidad o dividida en proyectos individuales
- Utilizar el Explorador de soluciones para controlar los detalles de la administración de archivos y centrarse al mismo tiempo en los elementos que constituyen la labor de desarrollo.
- Agregar elementos útiles a varios proyectos de la solución o a la solución sin tener que hacer referencia a dichos elementos en cada proyecto.
- Trabajar en diversos archivos, independientes de soluciones o proyectos

Elementos: archivos, referencias y conexiones de datos

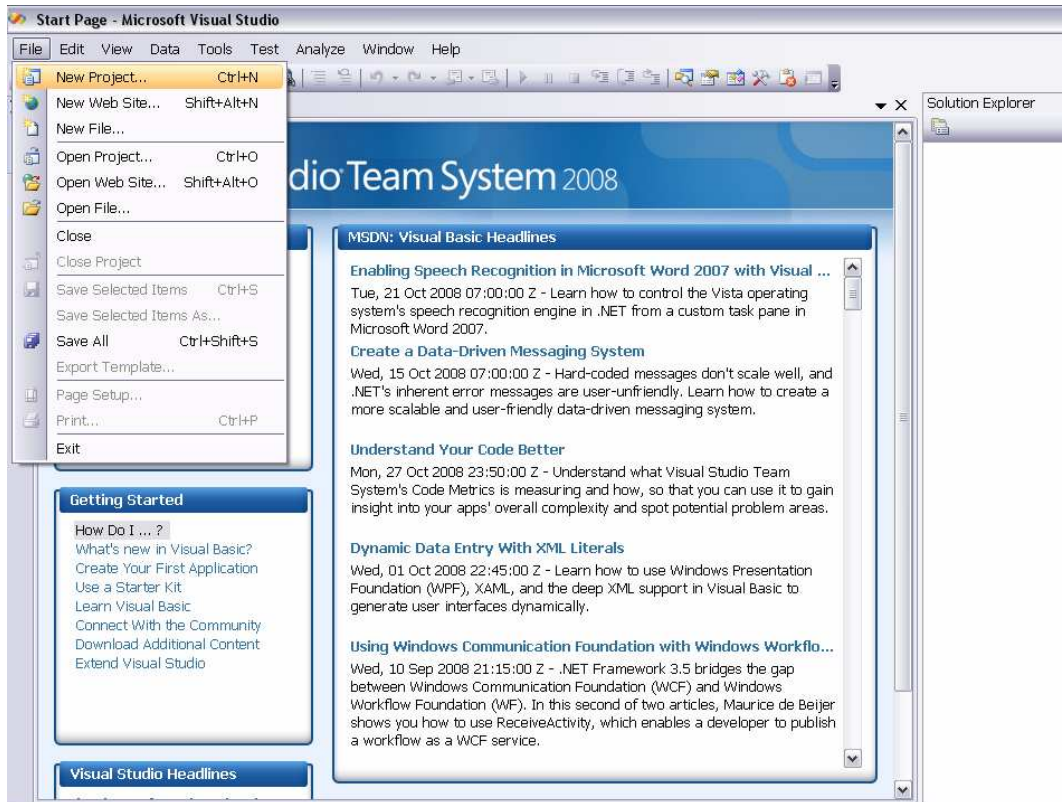
Los elementos pueden ser archivos y otras partes del proyecto como referencias, conexiones de datos o carpetas. En el Explorador de soluciones los elementos pueden organizarse de varias formas:

- En forma de elementos del proyecto, es decir, elementos que componen el proyecto, tales como formularios, archivos de código fuente y clases de un proyecto del Explorador de soluciones. La organización y la presentación dependerán de la plantilla de proyecto que se seleccione, así como de cualquier modificación que se realice.
- En forma de elementos de la solución para archivos que se aplican a la solución en su totalidad en la carpeta Elementos de la solución del Explorador de soluciones.
- En forma de varios archivos que no están asociados a ningún proyecto ni a ninguna solución y que pueden mostrarse en la carpeta Archivos varios.

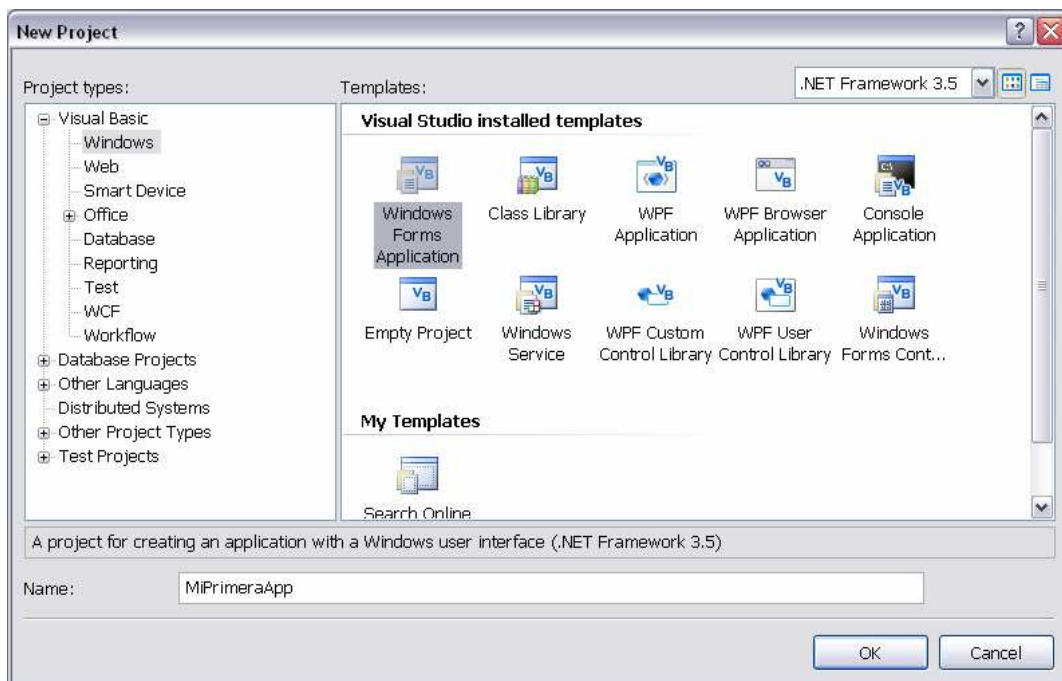
Iniciando un Proyecto y Solución en VStudio 2008

Después de iniciar el IDE de Visual Studio 2008, presenta una ventana que nos permite crear un proyecto de solución.

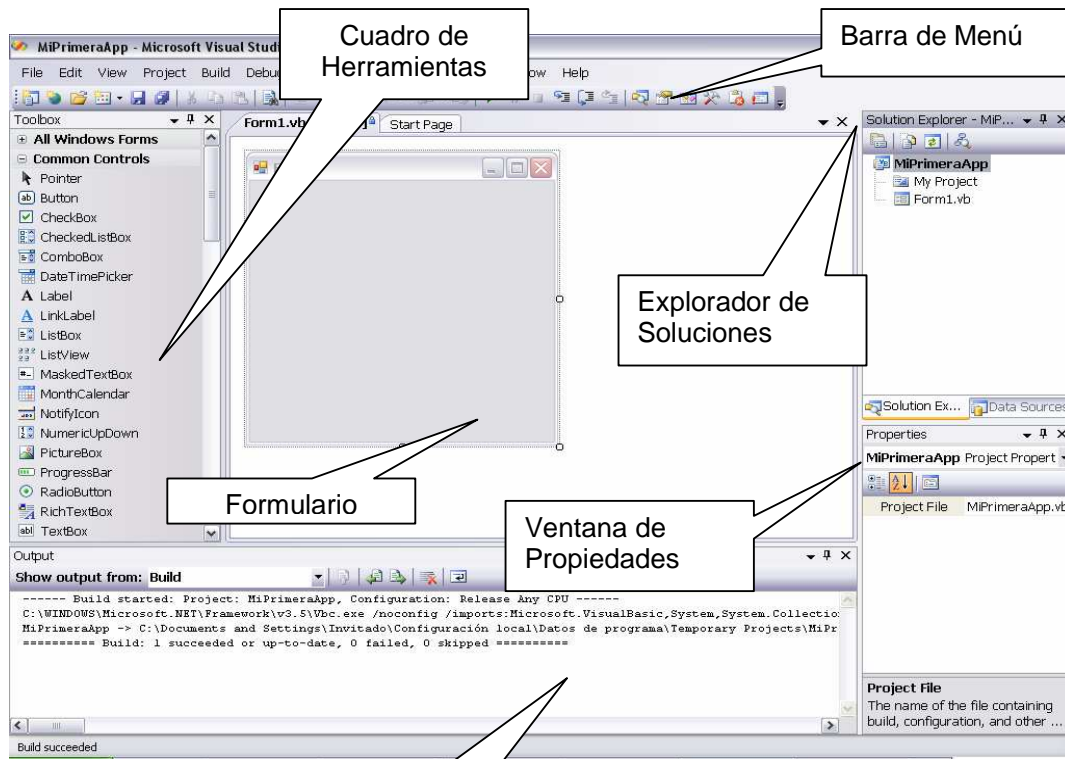
1. Ingresar al Menu File y Seleccionar la opción New Project



2. En la siguiente pantalla se debe elegir el lenguaje a utilizar Visual Basic y la plantilla de proyecto Windows Application



3. Seleccionamos OK. y se muestra la pantalla para empezar a trabajar.



### Detalle de la entorno

Ventana de Salida

### Cuadro de Herramientas

Contiene los objetos y controles que se pueden añadir a los formularios para crear una aplicación.

### Barra de Menús

Contiene comandos que son usados por Visual Basic. Como es habitual Microsoft siempre muestra menús habituales como Archivo, Edición, Ver, Insertar, etc. Sin embargo deberá prestar atención a los menús y comandos propios del entorno de desarrollo a los cuales permiten tener acceso a funciones específicas de programación como los menús de Proyecto, Construir y Depuración.

### Formulario

El formulario es el principal espacio o repositorio en el cual se basan todas las aplicaciones Windows Aplicación. Los formularios son las interfases de usuario de la aplicación sobre las cuáles se colocarán los objetos y controles necesarios para construir la GUI.

### Ventana de Salida

Muestra información relevante del proceso de compilación o de Debug de una aplicación

### Ventana de Propiedades

Lista de valores de las propiedades del formulario o control seleccionado los cuáles podrán ser personalizados.



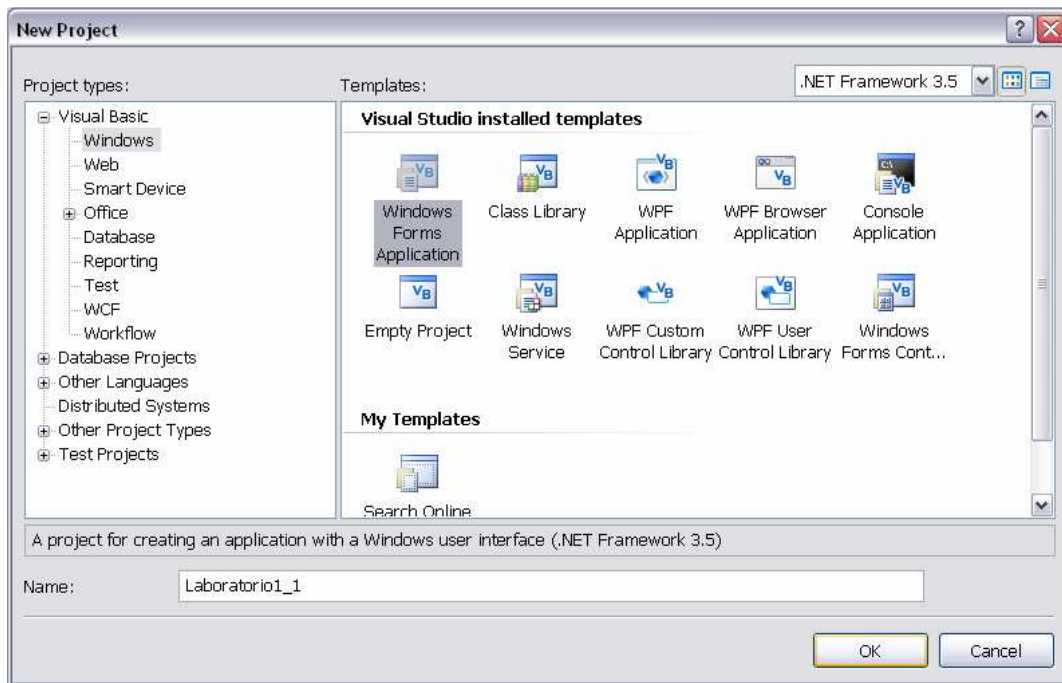
## Explorador de Soluciones

Dentro de este espacio se lista los proyectos que la solución contiene y asu vez los objetos que contiene cada proyecto.

## LABORATORIO 1.1

Cree una aplicación Windows que permita ingresar el Nombre de un usuario y muestre una ventana emergente de bienvenida con el nombre de usuario.

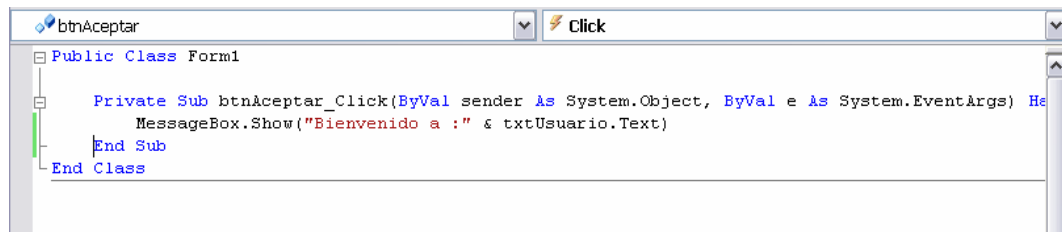
1. Ingrese a Visual Studio 2008
2. Seleccione Menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Application" y en Name escriba "Laboratorio1\_1"



4. Para la GUI considerar los siguientes objetos con sus propiedades tal como se indica en la siguiente tabla:

Objeto	Propiedad	Valor
Label	lblUsuario	Usuario
Textbox	txtUsuario	Ninguno
Button	btnAceptar	Aceptar

5. Dar doble click al botón e ingresar el siguiente código



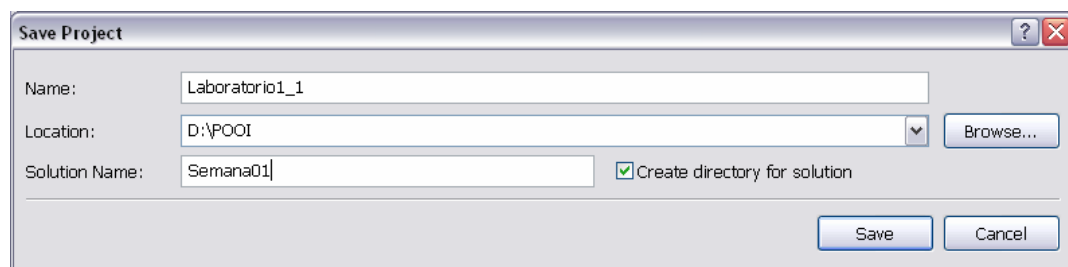
6. Ejecutar la aplicación presionando F5 o el icono de Start 



7. Evaluar la aplicación ingresando el nombre del usuario y presionando Aceptar.



8. Cerrar Formulario.  
9. Grabar proyecto en una ruta.

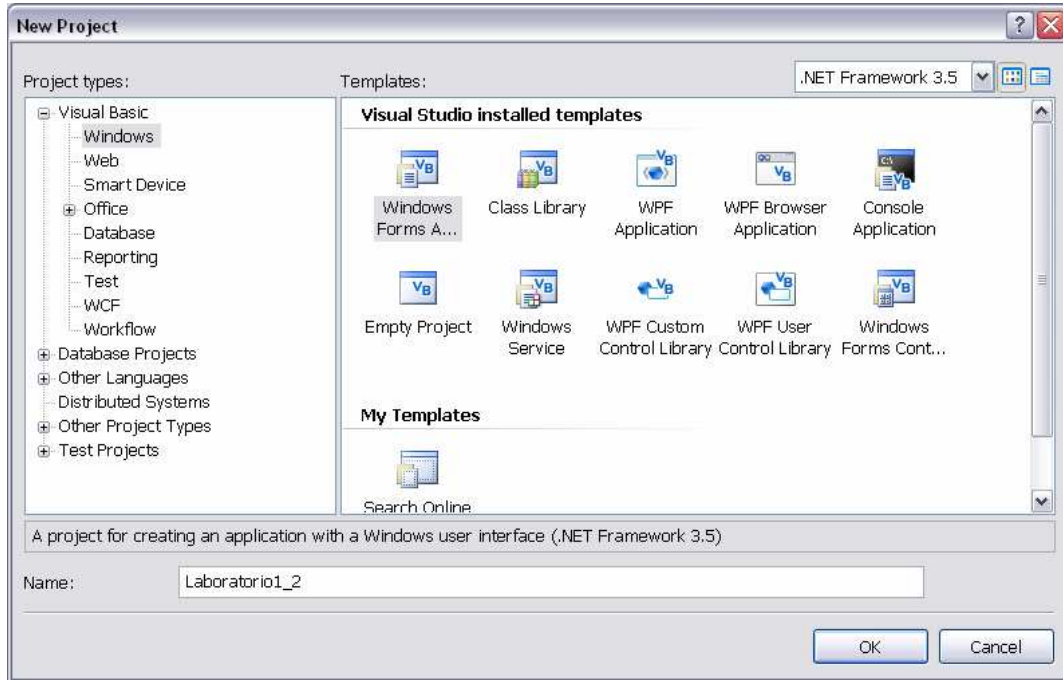


10. Seleccionar Save.

## LABORATORIO 1.2

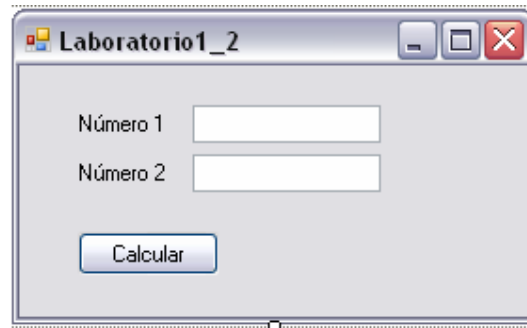
Cree una aplicación Windows que permita ingresar dos números enteros y muestre en una ventana la suma de los dos.

1. Ingrese a Visual Studio 2008
2. Seleccione Menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Application" y en Name escriba "Laboratorio1\_2"



4. Crear la GUI con los siguientes objetos según lo indica la tabla adjunta.

Objeto	Propiedad	Valor
Label1	Text	Numero 1
Label1	Text	Numero 2
Textbox1	Name	txtNumero1
Textbox2	Name	txtNumero2
Button1	Text	Calcular
Button2	Name	btnCalcular




5. Dar doble click al botón e ingresar el siguiente código.

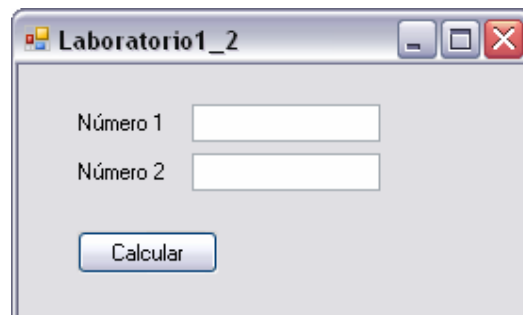
```

Public Class Form1
    Private Sub btnCalcular_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnCalcular.Click
        'Crear 2 variables de tipo enteros
        Dim num1 As Integer
        Dim num2 As Integer
        Dim suma As Integer
        'Asignar datos
        num1 = Convert.ToInt16(txtNumero1.Text.Trim)
        num2 = Convert.ToInt16(txtNumero2.Text.Trim)
        'validar que sean números enteros positivos
        If num1 < 0 Then
            MessageBox.Show("Número 1 debe ser mayor a cero")
            Exit Sub
        End If

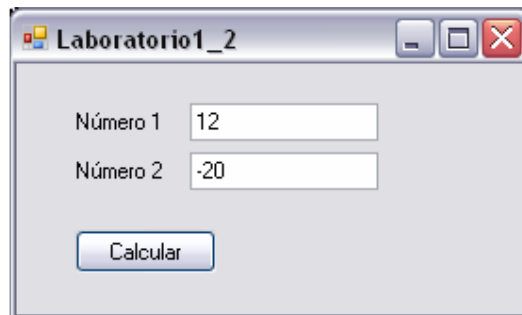
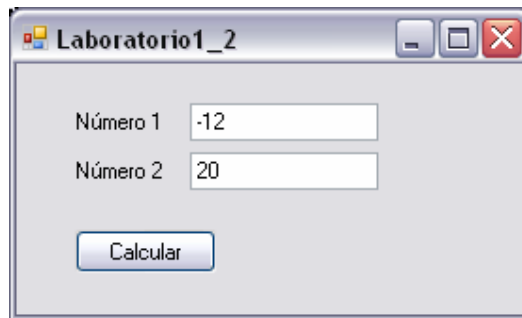
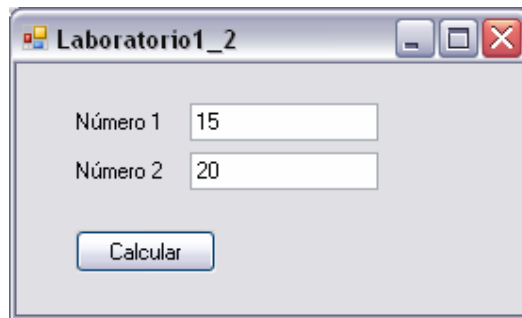
        If num2 < 0 Then
            MessageBox.Show("Número 2 debe ser mayor a cero")
            Exit Sub
        End If
        'si los dos son positivos, entonces sumarlos
        suma = num1 + num2
        MessageBox.Show("La suma es : " & suma)
    End Sub
End Class

```

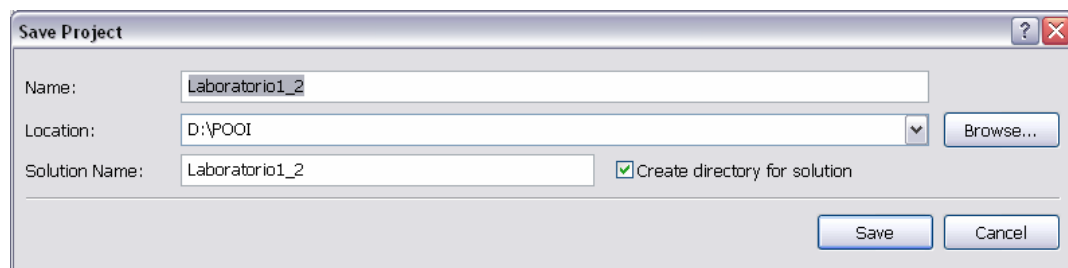
6. Ejecutar la aplicación presionando F5 o el icono de Start 



7. Evaluar la aplicación ingresando primera los dos números enteros positivos, luego un número negativo y el otro positivo, finalmente un número positivo y el otro negativo. Revisa los mensajes mostrados en cada caso.



8. Cerrar Formulario.
9. Grabar proyecto en una ruta.



10. Seleccionar

Save.

# Autoevaluación

1. ¿Qué es el CLR?

---

---

2. ¿Qué es .NET?

---

---

3. ¿Qué es un Assembly?

---

---

4. ¿Qué es una solución?

---

---

5. ¿Qué es un proyecto?

---

---

6. ¿Qué diferencias hay entre las propiedades Name y Text de los objetos?

---

---

7. ¿Qué es codificación basada en eventos?

---

---

## Para Recordar

Microsoft .NET es una plataforma de desarrollo y ejecución de aplicaciones Windows Forms, Aplicaciones de consola y Web.

Componentes principales de la plataforma:

- Un entorno de ejecución de aplicaciones, llamado Runtime
- Un conjunto de bibliotecas funcionales y controles reutilizables
- Un conjunto de lenguajes de programación de alto nivel, junto con sus compiladores.
- Un conjunto de utilitarios y herramientas de desarrollo para simplificar las tareas más comunes del proceso de desarrollo de aplicaciones.
- Documentación y guías de arquitectura, que describen las mejores prácticas de diseño, organización, desarrollo, prueba e instalación de aplicaciones de aplicaciones .NET
- Los Assemblies son reutilizables, versionables y auto-descriptivos, ya que no sólo contienen el código MSIL que representa la lógica de la aplicación, sino que también incluyen información sobre sí mismos y sobre todos los recursos externos de los que dependen para funcionar correctamente.
- Los principales namespaces de la biblioteca de clases .NET
  - **System:** Raíz de todos los otros namespaces, y dentro del cual se puede encontrar la mayoría de los namespaces correspondientes al Class Library.
  - **System.Data y System.Xml:** En conjunto, estos dos namespaces constituyen la tecnología conocida como ADO.NET.
  - **System.Web:** Dentro de este namespace se encuentran todos los tipos necesarios para programar aplicaciones y servicios Web ASP.NET.
  - **System.Windows.Forms:** Dentro de este namespace se encuentran todos los tipos necesarios para programar aplicaciones de escritorio basadas en formularios y ventanas Windows.

### Fuente

Texto adaptado de la página web:

<http://msdn.microsoft.com/es-es/library>





<b>UNIDAD DE APRENDIZAJE</b> <b>1</b>
<b>SEMANA</b> <b>2</b>

## **MANEJO DE CONTROLES BASICOS PARA FORMULARIOS Y APLICACION DE POO**

---

### **LOGRO DE LA UNIDAD DE APRENDIZAJE**

Al término de la unidad, los alumnos elaboran aplicaciones Windows .NET que se conectan a un origen de datos utilizando los objetos de ADO.NET para realizar operaciones de consulta y actualización de datos.

### **TEMARIO**

- Manejo de controles básicos de los proyectos Windows Application
- Manejo de Herencia e Interface

### **ACTIVIDADES PROPUESTAS**

- Los alumnos desarrollan los laboratorios de esta semana.
- Los alumnos desarrollan aplicaciones con los controles básicos.
- Los alumnos desarrollan clases y objetos.

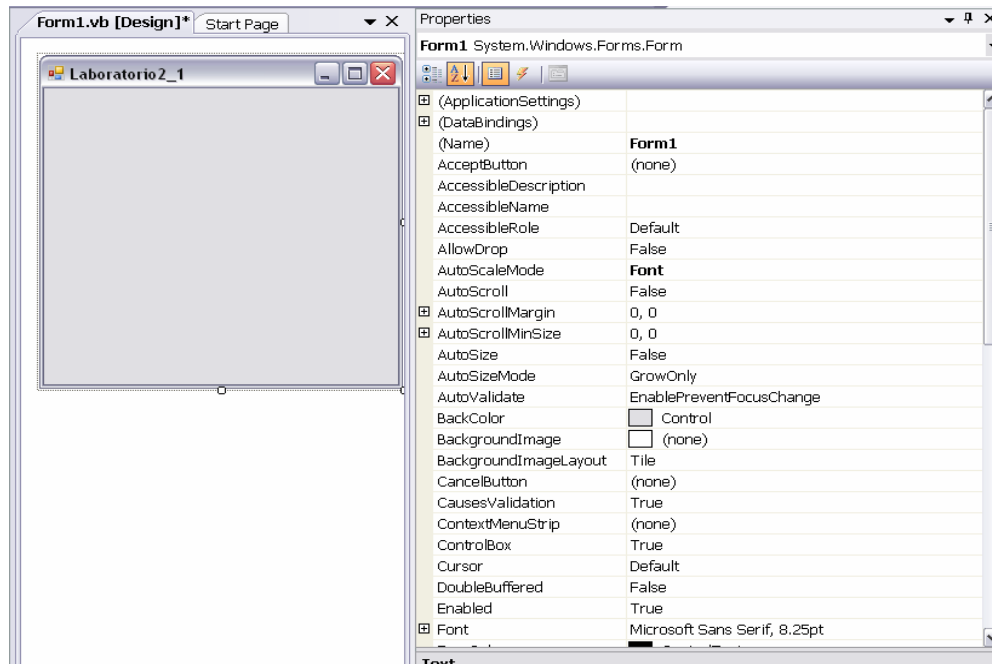
## 1. Formularios Windows y Controles Básicos

### 1.1. System.Windows.Forms (Espacio de nombres)

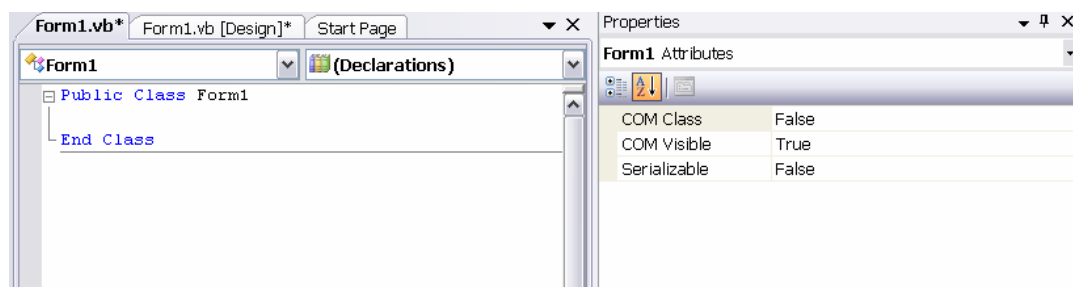
El espacio de nombres **System.Windows.Forms** contiene clases para crear aplicaciones para Windows que aprovechan todas las ventajas de las características de la interfaz de usuario disponibles en el sistema operativo Microsoft Windows.

La mayoría de las clases del espacio de nombres **System.Windows.Forms** derivan de la clase Control. La clase Control proporciona la funcionalidad base de todos los controles que se muestran en un objeto Form. La clase [Form](#) representa una ventana dentro de una aplicación.

#### Vista Diseño



#### Vista código



### 1.1.1. La clase Form

Un formulario **Form** es una representación de cualquier ventana mostrada en su aplicación. La clase **Form** se puede utilizar para crear ventanas estándar, de herramientas, sin bordes y flotantes. También puede utilizar la clase **Form** para crear las ventanas modales como un cuadro de diálogo. Un tipo especial de formulario, el formulario de interfaz de múltiples documentos (MDI), puede contener otros formularios denominados formularios MDI secundarios. Los formularios MDI se crean estableciendo la propiedad `IsMdiContainer` en `true`. Los formularios MDI secundarios se crean estableciendo la propiedad `MdiParent` en el formulario MDI principal que contendrá el formulario secundario.

#### Propiedades más comunes

BackColor	Permite asignar color de fondo.
BackgroundImage	Obtiene o establece la imagen de fondo que se muestra en el control.
ControlBox	Obtiene o establece un valor que indica si se muestra un cuadro de control en la barra de título del formulario.
Dock	Obtiene o establece que los bordes del control se acoplarán a su control principal
FormBorderStyle	Obtiene o establece el estilo del borde del formulario.
IsMdiContainer	Obtiene o establece un valor que indica si el formulario es un contenedor para formularios MDI (interfaz de múltiples documentos) secundarios.
KeyPreview	Obtiene o establece un valor que indica si el formulario recibe los eventos clave antes de que pasen al control que tiene el foco.
MdiParent	Obtiene o establece el formulario MDI (interfaz de múltiples documentos) principal actual de este formulario.
Name	Obtiene o establece el nombre del control. (Se hereda de Control).
Opacity	Obtiene o establece el nivel de opacidad del formulario.
StartPosition	Obtiene o establece la posición inicial del formulario en tiempo de ejecución.
Text	Muestra texto en barra de título.

## Eventos más comunes

Closed	Tiene lugar cuando el formulario está cerrado.
Closing	Tiene lugar cuando se cierra el formulario.
Disposed	Se produce cuando el componente se elimina mediante una llamada al método Dispose. (Se hereda de Component).
FormClosed	Se produce después de haberse cerrado el formulario.
FormClosing	Se produce antes de cerrar el formulario.
Load	Se produce antes de que se muestre un formulario por primera vez.

### 1.2. Manejo de controles básicos que se utilizan en formularios Windows Forms

A continuación se ofrece una lista alfabética de los controles y componentes que se pueden utilizar en formularios Windows Forms. Además de los controles de formularios Windows Forms que se tratan en esta sección, puede agregar controles ActiveX y controles personalizados a los formularios Windows Forms. Si no encuentra en esta lista el control que necesita, puede también crear uno propio

#### Nota:

En las tablas siguientes no se muestran todos los controles o componentes que puede utilizar en formularios Windows Forms; para una lista más completa, vea Controles que se utilizan en formularios Windows Forms

#### 1.2.1. Label (Control, formularios Windows Forms)

Los controles Label de formularios Windows Forms se utilizan para mostrar texto o imágenes que el usuario no puede editar. Se utilizan para identificar objetos en un formulario; por ejemplo, para proporcionar una descripción de lo que hará cierto control si se hace clic en él o para mostrar información en respuesta a un evento o proceso en tiempo de ejecución de la aplicación. Por ejemplo, puede utilizar etiquetas para agregar títulos descriptivos a cuadros de texto, cuadros de lista, cuadros combinados, etc. También puede escribir código que cambie el texto que muestra una etiqueta en respuesta a eventos en tiempo de ejecución. Por ejemplo, si la aplicación tarda varios minutos en procesar un cambio, puede mostrar en una etiqueta un mensaje que notifique el estado de procesamiento del cambio.

Aquí detallamos algunas de sus propiedades más comunes.

Nombre	Descripción
AutoSize	Obtiene o establece un valor que indica si el control cambia automáticamente de tamaño para mostrar todo su contenido.
BackColor	Obtiene o establece el color de fondo del control.
BackgroundImage	Infraestructura. Obtiene o establece la imagen representada en el fondo del control.
BorderStyle	Obtiene o establece el estilo de borde del control.
Dock	Obtiene o establece que los bordes del control se acoplarán a su control principal y determina cómo se cambia el tamaño de un control con su elemento primario.
Enabled	Obtiene o establece un valor que indica si el control puede responder a la interacción del usuario.
Font	Obtiene o establece la fuente del texto que muestra el control.
ForeColor	Obtiene o establece el color de primer plano del control.
Image	Obtiene o establece la imagen que se muestra en un control Label.
Name	Obtiene o establece el nombre del control.
Text	Establece el texto a mostrar en la etiqueta
Visible	Obtiene o establece un valor que indica si se muestran el control y todos sus controles primarios.

### **1.2.2. 1.2.2. TextBox (Control, formularios Windows Forms)**

Los cuadros de texto de formularios Windows Forms se utilizan para obtener entradas del usuario o para mostrar texto. El control TextBox se utiliza generalmente para el texto que se puede editar, aunque también puede configurarse como control de sólo lectura. Los cuadros de texto pueden mostrar varias líneas, ajustar el texto al tamaño del control y agregar formato básico. El control TextBox permite un único formato para el texto que se muestra o escribe en el control.

Aquí detallamos algunas de sus propiedades más comunes.

Nombre	Descripción
BackColor	Sobrecargado.
Font	Obtiene o establece la fuente del texto que muestra el control.
Multiline	Sobrecargado.
PasswordChar	Obtiene o establece los caracteres utilizados para enmascarar caracteres de una contraseña en un control TextBox de una sola línea.
ReadOnly	Obtiene o establece un valor que indica si el texto del cuadro de texto es de sólo lectura.
SelectedText	Obtiene o establece un valor que indica el texto seleccionado actualmente en el control.
Text	Sobrecargado.
TextAlign	Obtiene o establece cómo se alinea el texto en un control TextBox.
TextLength	Obtiene la longitud del texto en el control.

### 1.2.3. Button (formularios Windows Forms)

El control Button de los formularios Windows Forms permite al usuario hacer clic en él para ejecutar una acción. Cuando se hace clic en el botón, da la sensación de que se ha presionado y soltado. Cada vez que el usuario hace clic en un botón, se invoca al controlador del evento Click. El código se ubica en el controlador del evento Click para ejecutar la acción deseada.

El texto que se muestra en el botón se almacena en la propiedad Text. Si este texto supera el ancho del botón, se ajustará en la línea siguiente. No obstante, si el control no dispone del alto suficiente, el texto aparecerá cortado. El control Button también muestra imágenes mediante las propiedades Image y ImageList.

Aquí detallamos algunas de sus propiedades más comunes.

Nombre	Descripción
BackColor	Sobrecargado.
BackgroundImage	Obtiene o establece la imagen de fondo que se muestra en el control. (Se hereda de Control).
FlatAppearance	Obtiene el aspecto del borde y los colores utilizados para indicar el estado de comprobación y el estado del mouse. (Se hereda de ButtonBase).
Font	Obtiene o establece la fuente del texto que muestra el control. (Se hereda de Control).
ForeColor	Obtiene o establece el color de primer plano del control. (Se hereda de Control).
Text	Sobrecargado.
TextAlign	Obtiene o establece la alineación del texto en el control de botón. (Se hereda de ButtonBase).

#### 1.2.4. ComboBox (formularios Windows Forms)

Un **ComboBox** muestra un campo de edición de cuadro de texto combinado con un **ListBox** y permite al usuario seleccionar elementos de la lista o escribir texto nuevo. El comportamiento predeterminado de **ComboBox** es mostrar un campo de edición con una lista desplegable oculta.

Para agregar objetos a la lista en tiempo de ejecución, asigne una matriz de referencias a objetos con el método **AddRange**. De este modo la lista muestra el valor de cadena predeterminado para cada objeto. Puede agregar objetos individuales con el método **Add**.

##### **Nota:**

Si tiene un **ListBox**, **ComboBox** o **CheckedListBox** en un formulario de Windows de base y desea modificar las colecciones de cadenas de esos controles en un formulario de Windows derivado, las colecciones de cadenas de estos controles del formulario de Windows de base deben estar vacías. Si las colecciones de cadenas no están vacías, se establecen como de sólo lectura cuando se deriva otro formulario de Windows.

Aquí detallamos algunas de sus propiedades más comunes.

Nombre	Descripción
DropDownStyle	Obtiene o establece un valor que especifica el estilo del cuadro combinado.
Items	Obtiene un objeto que representa la colección de los elementos que contiene el ComboBox.
SelectedIndex	Obtiene o establece el índice que especifica el elemento seleccionado actualmente.
SelectedItem	Obtiene o establece el elemento seleccionado actualmente en el ComboBox.
SelectedText	Obtiene o establece el texto que se selecciona en la parte de un ComboBox que se puede editar.
SelectedValue	Obtiene o establece el valor de la propiedad miembro especificada por la propiedad ValueMember.
Sorted	Obtiene o establece un valor que indica si los elementos del cuadro combinado están ordenados.
ValueMember	Obtiene o establece la propiedad que se utilizará como valor real para los elementos del ListControl.

### 1.2.5. ListBox (formularios Windows Forms)

El control **ListBox** permite mostrar una lista de elementos para que el usuario los seleccione haciendo clic en ellos. Un control **ListBox** puede proporcionar una o varias selecciones mediante la propiedad **SelectionMode**. **ListBox** también proporciona la propiedad **MultiColumn** para poder mostrar los elementos en columnas en lugar de mostrarlos en una lista vertical. Con esto, el control puede mostrar más elementos visibles y el usuario ya no necesita desplazarse a un elemento.

Las propiedades **Items**, **SelectedItems** y **SelectedIndices** proporcionan acceso a las tres colecciones que **ListBox** utiliza. En la tabla siguiente se presentan las tres colecciones que **ListBox** utiliza y se indica su uso dentro del control.



Aquí detallamos algunas de sus propiedades más comunes.

Nombre	Descripción
Items	Obtiene los elementos del control ListBox.
MultiColumn	Obtiene o establece un valor que indica si el control ListBox admite varias columnas.
Name	Obtiene o establece el nombre del control. (Se hereda de Control).
SelectedIndex	Obtiene o establece el índice de base cero del elemento actualmente seleccionado en ListBox.
SelectedIndices	Obtiene una colección que contiene los índices de base cero de todos los elementos actualmente seleccionados en el control ListBox.
SelectedItem	Obtiene o establece el elemento actualmente seleccionado en el control ListBox.
SelectedItems	Obtiene una colección que contiene los elementos actualmente seleccionados en el control ListBox.
SelectedValue	Obtiene o establece el valor de la propiedad miembro especificada por la propiedad ValueMember.
ValueMember	Obtiene o establece la propiedad que se utilizará como valor real para los elementos del ListControl.

#### 1.2.6. CheckBox (formularios Windows Forms)

Utilice un control **CheckBox** para dar al usuario una opción del tipo verdadero/falso o sí/no. El control **CheckBox** de casilla de verificación puede mostrar una imagen o texto, o ambos.

Los controles **CheckBox** y **RadioButton** tienen una función similar: ambos permiten al usuario elegir de una lista de opciones. Los controles **CheckBox** permiten al usuario elegir una combinación de opciones.

La propiedad **Appearance** determina si la casilla de verificación **CheckBox** como un control **CheckBox** típico o como un botón.

Utilice la propiedad **Checked** para obtener o establecer el valor de un control **CheckBox** de casilla de verificación de dos estados .

**Nota:**

Si se establece la propiedad ThreeState en true, la propiedad Checked devuelve true para los estados activado o indeterminado.

Aquí detallamos algunas de sus propiedades más comunes.

Nombre	Descripción
Auto check	Obtiene o establece un valor que indica si los valores de Checked o CheckState y la apariencia de CheckBox cambian automáticamente al hacer clic en dicha casilla de verificación CheckBox.
Checked	Obtiene o establece un valor que indica si CheckBox está en el estado activado.
Text	Muestra un texto del control
TextAlign	Obtiene o establece la alineación del texto en el control CheckBox. (Invalida a ButtonBase.....TextAlign).

### 1.2.7. RadioButton (formularios Windows Forms)

El control **RadioButton** puede mostrar texto, Image o ambos.

Cuando el usuario selecciona un botón de opción dentro de un grupo, los otros se borran automáticamente. Todos los controles **RadioButton** de un contenedor determinado, como Form, constituyen un grupo. Para crear varios grupos en un formulario, coloque cada grupo en su propio contenedor, como un control GroupBox o Panel.

Los controles **RadioButton** y CheckBox tienen una función similar; es decir, ofrecen opciones que el usuario puede activar o desactivar. La diferencia consiste en que se pueden seleccionar múltiples controles CheckBox al mismo tiempo, mientras que los botones de opción se excluyen mutuamente.

Utilice la propiedad Checked para obtener o establecer el estado de un **RadioButton**. El aspecto de un botón de opción se puede modificar para que aparezca como botón de alternar o como botón de opción estándar al establecer la propiedad Appearance.

Nombre	Descripción
Checked	Obtiene o establece un valor que indica si el control está activado.
CheckAlign	Obtiene o establece la ubicación de la parte de casilla del control RadioButton.
TextAlign	Infraestructura. Obtiene o establece la alineación del texto en el control RadioButton.
AutoCheck	Obtiene o establece un valor que indica si el valor de Checked y el aspecto del control cambian automáticamente al hacer clic en el control.
Text	Muestra El texto del RadioButton

## 2. Introducción a POO en Visual Basic .NET

### 2.1 Clases y objetos

Las palabras "clase" y "objeto" se usan con tanta frecuencia en la programación orientada a objetos que es fácil confundir los términos. En general, una *class* es una representación abstracta de algo, mientras que un objeto es un ejemplo utilizable de lo que representa la clase. La única excepción a esta regla la constituyen los miembros de clases compartidas, que pueden utilizarse en instancias de una clase y en variables de objeto declaradas como tipo de la clase.

#### 2.1.1 Campos, propiedades, métodos y eventos

Las clases se componen de campos, propiedades, métodos y eventos. Los campos y propiedades representan información que contiene un objeto. Los campos se parecen a las variables ya que se pueden leer y establecer directamente. Por ejemplo, si tiene un objeto denominado "Car", podría almacenar su color en un campo denominado "Color".

Las propiedades se recuperan y establecen como los campos, pero se implementan mediante los procedimientos propiedad Get y Set, que proporcionan más control sobre la forma en que los valores se establecen o se devuelven.

Los métodos representan acciones que un objeto puede realizar. Por ejemplo, un objeto "Car" podría tener los métodos "StartEngine", "Drive" y "Stop". Los métodos se definen agregando procedimientos, ya sean rutinas o funciones Sub, a la clase.

Los eventos son notificaciones que un objeto recibe de, o transmite a, otros objetos o aplicaciones. Los eventos permiten a los objetos realizar acciones siempre que se produce un acontecimiento específico. Un ejemplo de evento para la clase "Car" sería un evento "Check\_Engine". Puesto que Microsoft Windows es un sistema controlado

por eventos, éstos pueden proceder de otros objetos, aplicaciones o entradas de usuario realizadas al hacer clic con el mouse (ratón) o al presionar teclas.

### **2.1.2. Encapsulación, herencia y polimorfismo**

La encapsulación significa que un grupo de propiedades, métodos y otros miembros relacionados se tratan como si de una sola unidad u objeto se tratase. Los objetos pueden controlar cómo se cambian propiedades o se ejecutan métodos.

Herencia describe la posibilidad de crear nuevas clases basadas en una clase existente. La nueva clase hereda todas las propiedades, métodos y eventos de la clase base, y puede personalizarse con propiedades y métodos adicionales.

Polimorfismo significa que puede tener múltiples clases que se pueden utilizar de forma intercambiable, si bien cada clase implementa las mismas propiedades o los mismos métodos de maneras diferentes. El polimorfismo es importante en la programación orientada a objetos puesto que permite usar elementos que tienen el mismo nombre, independientemente del tipo de objeto que se esté utilizando en ese momento

### **2.1.3. Sobrecarga, reemplazo y sombreado**

- Los miembros sobrecargados se utilizan para proporcionar diferentes versiones de una propiedad o método que tienen el mismo nombre, pero que aceptan un número diferente de parámetros, o parámetros con diferentes tipos de datos.
- Las propiedades y métodos reemplazados se utilizan para reemplazar una propiedad o método heredados que no son apropiados en una clase derivada. Los miembros reemplazados deben aceptar el mismo tipo de datos y número de argumentos. Las clases derivadas heredan los miembros reemplazados.
- Los miembros sombreados se utilizan para reemplazar localmente un miembro que tiene un ámbito más amplio. Cualquier tipo puede sombrear cualquier otro tipo. Por ejemplo, puede declarar una propiedad que sombree un método heredado con el mismo nombre. Los miembros sombreados no se pueden heredar.

## **3. Cómo definir clases en Visual Basic .NET**

### **Para definir una clase**

1. Cree un proyecto haciendo clic en Nuevo proyecto en el menú Archivo. Aparecerá el cuadro de diálogo Nuevo proyecto.
2. Seleccione Aplicación para Windows de la lista de plantillas del proyecto de Visual Basic para mostrar el nuevo proyecto.
3. Agregue una clase nueva al proyecto haciendo clic en Agregar clase en el menú Proyecto. Aparecerá el cuadro de diálogo Agregar nuevo elemento.

4. Seleccione la plantilla Clase.
5. Asigne a la nueva clase el nombre `UserNameInfo.vb` y, a continuación, haga clic en Agregar para mostrar el código de la nueva clase.

```
Public Class UserNameInfo  
  
End Class
```

6. Defina un campo privado para la clase; para ello, agregue el siguiente código entre las instrucciones Class y End Class:

```
Private _nombreUsuario As String
```

7. Declarar el campo como Private quiere decir que sólo se puede utilizar dentro de la clase. Se pueden utilizar modificadores de acceso más amplio, por ejemplo Public, para hacer que los campos estén disponibles desde fuera de la clase.
8. Defina una propiedad para la clase agregando el código siguiente:

```
Public Property nombreUsuario () As String  
    Get  
        ' Gets the property value.  
        Return _nombreUsuario  
    End Get  
    Set(ByVal Value As String)  
        ' Sets the property value.  
        _nombreUsuario = Value  
    End Set  
End Property
```

9. Defina un método para la clase agregando el código siguiente:

```
Public Sub AMayusculas()  
    ' Convertir a mayúsculas el valor de la propiedad  
    _nombreUsuario = UCase(userNameValue)  
End Sub
```

10. Defina un constructor parametrizado para la clase nueva agregando un procedimiento denominado `Sub New`:

```
Public Sub New(ByVal nomUsuario As String)
    ' Set the property value.
    Me._ nombreUsuario = nomUsuario
End Sub
```

Cuando se crea un objeto basado en esta clase, se llama al constructor **Sub New** automáticamente. Este constructor establece el valor del campo que contiene el nombre de usuario.

## Laboratorio 2.1

### Manejo de controles básicos en formularios Windows

1. Ingrese a Visual Studio 2008
2. Seleccione menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Application" y en Name escriba "Laboratorio2\_1".
4. Diseñe la GUI del formulario teniendo en cuenta los siguientes controles que se colocan en la tabla adjunta.

Objeto	Propiedad	Valor
Label1	Text	Nombre de Alumno
Label2	Text	Turno
Label3	Text	Día
Label4	Text	Curso
Textbox1	Name	txtAlumno
ComboBox1	Name	cboTurno
ComboBox2	Name	Clodia
ComboBox3	Name	cboCurso
RadioButton1	Name	rbContado
RadioButton1	Text	Contado
RadioButton2	Name	rbCredito

RadioButton2	Text	Crédito
RadioButton3	Name	rbCheque
RadioButton3	Text	Cheque
ListbBox1	Name	IstAlumnos
Button1	Name	btnAceptar
Button1	Text	Aceptar
Button2	Name	btnCancelar
Button2	Text	Cancelar
Button3	Name	btnCerrar
Button3	Text	Cerrar
CheckBox1	Name	chkMayorDeEdad
CheckBox1	Text	Mayor de Edad

4. Declarar el método cargarCombos con el siguiente código en la Vista Código del formulario.

```
Public Class Form1
    'método sin retorno que carga los combos
    Sub cargarCombos()
        cboTurno.Items.Add("Mañana")
        cboTurno.Items.Add("Tarde")
        cboTurno.Items.Add("Noche")
        cboTurno.SelectedIndex = 0

        cboDia.Items.Add("LUNES")
        cboDia.Items.Add("MARTES")
        cboDia.Items.Add("MIERCOLES")
        cboDia.Items.Add("JUEVES")
        cboDia.Items.Add("VIERNES")
        cboDia.Items.Add("SABADO")
        cboDia.Items.Add("DOMINGO")
        cboDia.SelectedIndex = 0

        CboCurso.Items.Add("Office Proefficient")
        CboCurso.Items.Add("Diseño FLASH")
        CboCurso.Items.Add(".NET 2008")
        CboCurso.Items.Add("SQL SERVER 2008")
        CboCurso.Items.Add("ASP .NET")
        CboCurso.SelectedIndex = 0

        rbContado.Checked = True
    End Sub
End Class
```

5. En el evento **Load** del formulario llamar a este método

```
Public Class Form1
    'método sin retorno que carga los combos
    Sub cargarCombos()
    End Sub

    Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
        cargarCombos()
    End Sub
End Class
```

6. Agregar las siguientes líneas de código a nivel local en el método de evento del botón Aceptar.

```
Private Sub btnAceptar_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnAceptar.Click
    'variables locales
    Dim strNombre, strTurno, strCurso, _
    strDia, strPago, strMayorEdad As String
    'asignar valores
    strNombre = txtAlumno.Text.Trim
    strCurso = cboCurso.SelectedItem.ToString
    strDia = cboDia.SelectedItem.ToString
    strTurno = cboTurno.SelectedItem.ToString
    strDia = cboDia.SelectedItem.ToString
    'verificar el tipo de pago elegido
    If rbContado.Checked = True Then
        strPago = "Contado"
    End If
    If rbCheque.Checked = True Then
        strPago = "Cheque"
    End If
    If rbCredito.Checked = True Then
        strPago = "Crédito"
    End If

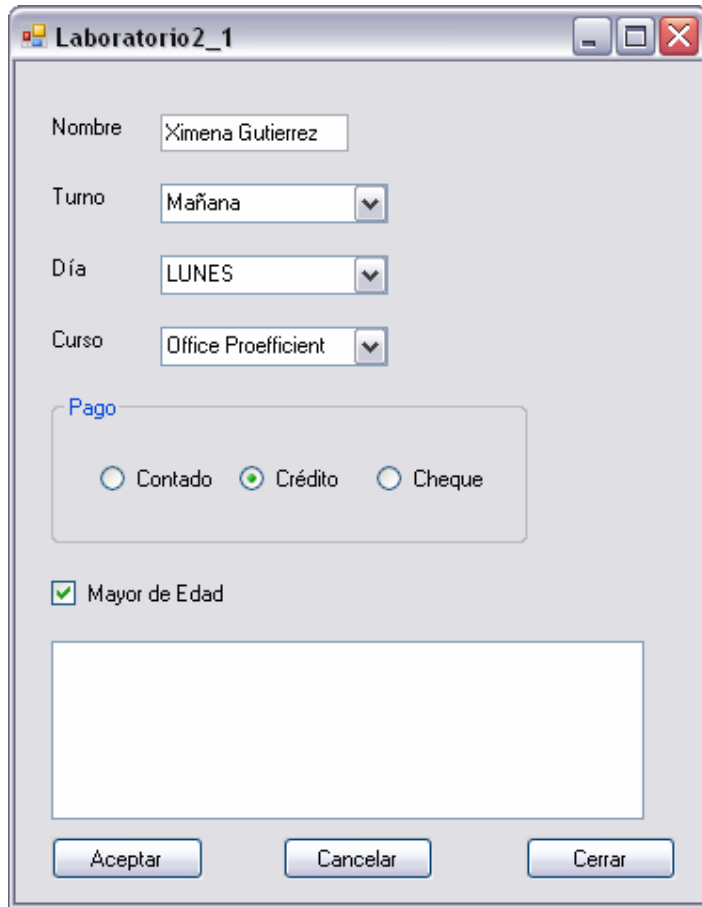
    If chkMayorDeEdad.Checked = True Then
        strMayorEdad = InputBox("Ingrese número de DNI")
    Else
        strMayorEdad = "No tiene"
    End If
    MessageBox.Show(" Confirmació de Registro " & vbCrLf & _
        "Alumno:" & strNombre & vbCrLf & _
        "Curso:" & strCurso & vbCrLf & _
        "Turno:" & strCurso & vbCrLf & _
        "Tipo de Pago:" & strPago & vbCrLf & _
        "DNI:" & strMayorEdad, "Información del Sistema", _
        MessageBoxButtons.OK, MessageBoxIcon.Information)
    'Agregar registro a la lista
    lstAlumnos.Items.Add(strNombre & vbTab & strCurso & vbTab & strTurno)
    limpiar()

End Sub
Sub limpiar()
    cboCurso.Text = "Seleccione Curso"
    cboDia.Text = "Seleccione Curso"
    cboTurno.Text = "Seleccione Curso"
    txtAlumno.Text = String.Empty
    rbCheque.Checked = False
    rbContado.Checked = False
    rbCredito.Checked = False
    chkMayorDeEdad.Checked = False
    txtAlumno.Focus()
End Sub

End Class
```

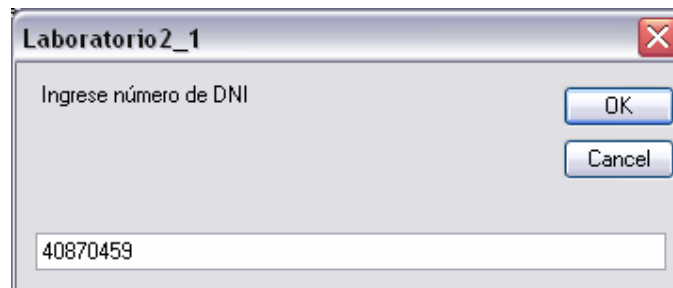


7. Ejecutar el formulario con F5 o el icono de iniciar.



The screenshot shows a Windows-style window titled "Laboratorio2\_1". Inside, there are several input fields: "Nombre" with the text "Ximena Gutierrez", "Turno" with a dropdown menu showing "Mañana", "Día" with a dropdown menu showing "LUNES", and "Curso" with a dropdown menu showing "Office Proefficient". Below these is a section titled "Pago" containing three radio buttons: "Contado", "Crédito" (which is selected), and "Cheque". Underneath the "Pago" section is a checked checkbox labeled "Mayor de Edad". At the bottom of the form is a large empty rectangular box. At the very bottom of the window are three buttons: "Aceptar", "Cancelar", and "Cerrar".

8. Ingresar nombres del Alumno, selecciona botón Aceptar y mostrará la siguiente ventana de entrada de datos



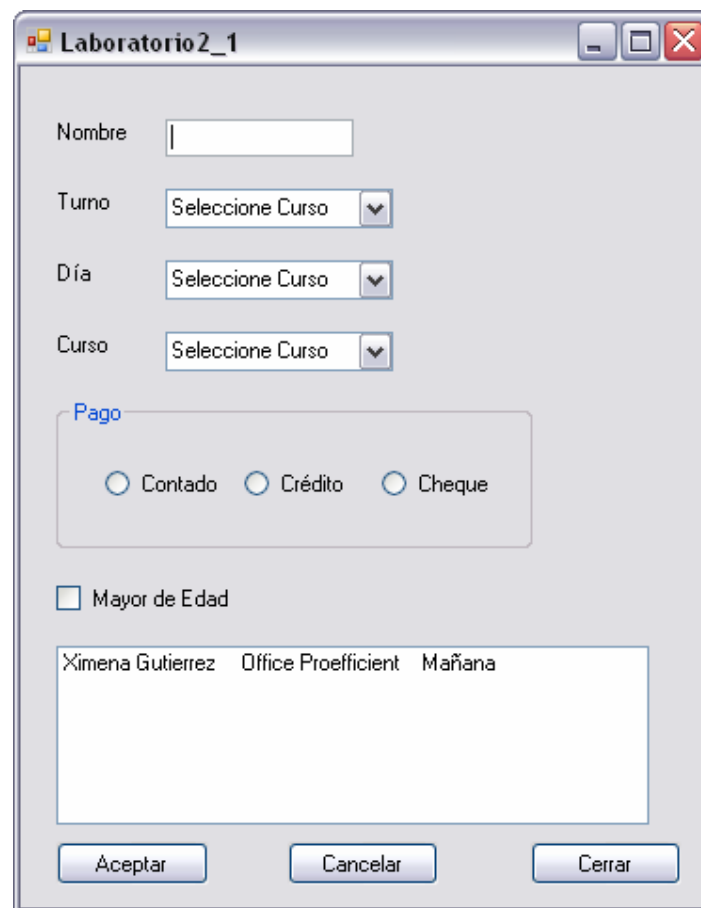
The screenshot shows a smaller dialog box titled "Laboratorio2\_1". It contains the text "Ingrese número de DNI" and two buttons, "OK" and "Cancel", on the right side. At the bottom, there is a text input field containing the number "40870459".

9. Ingresar el DNI solicitado y OK.

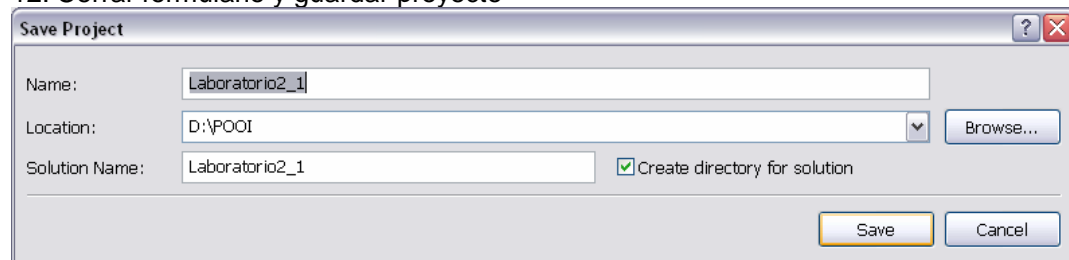
10. Mostrará la siguiente pantalla.



11. Una vez ingresado el alumno sus datos quedan registrados y se visualizan en el listbox.



12. Cerrar formulario y guardar proyecto



## Laboratorio 2.2

### Manejo de clases y objetos

En este laboratorio se resolverá el siguiente caso aplicando los conceptos de POO. Se debe crear una aplicación que permita el registro de los datos del docente o del administrativo.

En una empresa Educativa se desea controlar el registro de sus empleados, los cuáles han sido clasificados como Docentes o Administrativos.

Para todo empleado se desea conocer su código, nombres, apellidos, DNI, dirección, fecha de nacimiento y fecha de ingreso. Además, se debería obtener su edad y tiempo de servicio.

El Docente es un empleado, pero también tiene un curso y horas trabajadas. El sueldo de un docente es el producto de horas por tarifa, la cuál depende del curso que se dicta.

Curso	Tarifa(\$)
VB.NET 2008	45
ORACLE	65
WINDOWS 2003 SERVER	70
SQL SERVER 2008	40

El Administrativo es un empleado, pero también tiene un área de trabajo, sueldo bruto y AFP.

AFP	%
INTEGRA	12.3
PRIMA	12.6
PROFUTURO	11.9
HORIZONTE	12.4

El sueldo de un administrativo se calcula por la diferencia del sueldo bruto menos el monto de AFP.

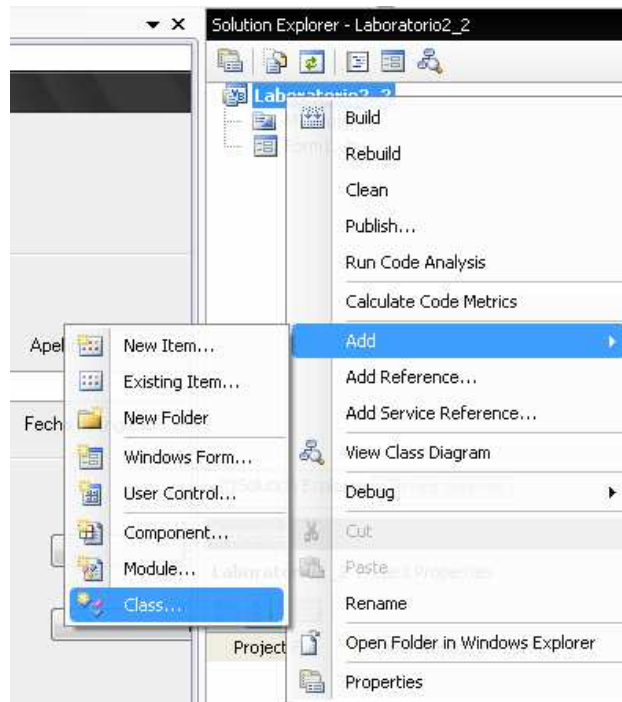
1. Ingrese a Visual Studio 2008
2. Seleccione menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Application" y en Name escriba "Laboratorio3\_2".

4. Diseñe la GUI del formulario teniendo en cuenta los siguientes controles que se colocan en la tabla adjunta.

Objeto	Propiedad	Valor
Label1	Text	Tipo de empleado
Label2	Text	Código
Label3	Text	Nombres
Label4	Text	Apellidos
Label5	Text	Dirección
Label6	Text	Fecha de Nacimiento
Label7	Text	Fecha de Ingreso
Label8	Text	Curso
Label9	Text	Horas
Label10	Text	Area
Label11	Text	Sueldo
Label12	Text	AFP
ComboBox1	Name	cboEmpleado
ComboBox2	Name	cboCurso
ComboBox3	Name	cboArea
ComboBox4	Name	cboAFP
TextBox1	Name	txtCodigo
TextBox1	Text	Vacio
TextBox2	Name	txtNombres
TextBox2	Text	Vacio
TextBox3	Name	txtApellidos
TextBox3	Text	Vacio
TextBox4	Name	txtDireccion
TextBox4	Text	Vacio

TextBox5	Name	txtHoras
TextBox5	Text	Vacío
TextBox6	Name	txtSueldoBruto
TextBox6	Text	Vacío
GroupBox1	Name	grpDatosGenerales
GroupBox2	Name	grpDocente
GroupBox3	Name	grpEmpleado
Button1	Name	btnRegistrar
Button1	Text	Registrar
Button2	Name	btnLimpiar
Button2	Text	Limpiar
DateTimePicker1	Name	DtpFecNac
DateTimePicker1	Format	Short
DateTimePicker2	Name	DtpFecIng
DateTimePicker2	Format	Short

- Primero se deben crear las clases necesarias para el desarrollo del caso. Click derecho a nivel del proyecto → Add → Class . Repetir este paso tres veces para crear las clases Empleado, Docente y Administrativo.



- Luego se debe codificar la clase base o padre, es decir, **Empleado**. Aquí se crearán los atributos, propiedades, constructor y métodos.

```
Public Class Empleado
    'atributos
    Protected strCodigo As String
    Protected strNombres As String
    Protected strApellidos As String
    Protected strDireccion As String
    Protected strDNI As String
    Protected dtFechaNac As Date
    Protected dtFechaIng As Date
```

```

'constructor
Sub New()
    strCodigo = ""
    strNombres = ""
    strApellidos = ""
    strDireccion = ""
    strDNI = ""
    dtFechaNac = Now
    dtFechaIng = Now
End Sub
'propiedades públicas
Public Property Codigo() As String
    Get
        Return strCodigo
    End Get
    Set(ByVal value As String)
        strCodigo = value
    End Set
End Property
Public Property Nombres() As String
    Get
        Return strNombres
    End Get
    Set(ByVal value As String)
        strNombres = value
    End Set
End Property
Public Property Apellidos() As String
    Get
        Return strApellidos
    End Get
    Set(ByVal value As String)
        strApellidos = value
    End Set
End Property
Public Property DNI() As String
    Get
        Return strDNI
    End Get
    Set(ByVal value As String)
        strDNI = value
    End Set
End Property
Public Property Direccion() As String
    Get
        Return strDireccion
    End Get
    Set(ByVal value As String)
        strDireccion = value
    End Set
End Property
Public Property FechaNacimiento() As Date
    Get
        Return dtFechaNac
    End Get
    Set(ByVal value As Date)
        dtFechaNac = value
    End Set
End Property
Public Property FechaIngreso() As Date

```

```
Get
    Return dtFechaIng
End Get
Set(ByVal value As Date)
    dtFechaIng = value
End Set
End Property
'método públicos
Public Function obtenerEdad() As Int16
    Return DateDiff(DateInterval.Year, dtFechaNac.Date, Now.Date)
End Function
Public Function obtenerTiempoServicio() As Int16
    Return DateDiff(DateInterval.Year, dtFechaIng.Date, Now.Date)
End Function
End Class
```

7. En la clase Docente se tendrá este otro código. Aquí aplicamos herencia a través de la palabra **inherits**.

```
Public Class Docente
    Inherits Empleado
    'atributo
    Private strCurso As String
    Private dbHoras As Double
    'constructor
    Sub New()
        strCurso = ""
        dbHoras = 0
    End Sub
    'propiedades públicas
    Public Property Curso() As String
        Get
            Return strCurso
        End Get
        Set(ByVal value As String)
            strCurso = value
        End Set
    End Property
    Public Property Horas() As Double
        Get
            Return dbHoras
        End Get
        Set(ByVal value As Double)
            dbHoras = value
        End Set
    End Property
    'métodos públicos
    Public Function obtenerSueldo() As Double
        Select Case strCurso
            Case "VB.NET 2008"
                Return 45 * dbHoras
            Case "ORACLE"
                Return 65 * dbHoras
            Case "WINDOWS 2003 SERVER"
                Return 70 * dbHoras
            Case "SQL SERVER 2008"
                Return 40 * dbHoras
        End Select
    End Function
End Class
```

End Class

8. En la clase Administrativo colocar el siguiente código.

```
Public Class Administrativo
    Inherits Empleado
    'atributos
    Private strArea As String
    Private dbSueldo As Double
    Private strAFP As String
    'constructor
    Sub New()
        strArea = ""
        dbSueldo = 0
        strAFP = ""
    End Sub
    'propiedades públicas
    Public Property Area() As String
        Get
            Return strArea
        End Get
        Set(ByVal value As String)
            strArea = value
        End Set
    End Property
    Public Property Sueldo() As Double
        Get
            Return dbSueldo
        End Get
        Set(ByVal value As Double)
            dbSueldo = value
        End Set
    End Property
    'métodos públicos
    Public Function obtenerSueldo() As Double
        Select Case strAFP
            Case "INTEGRA"
                Return dbSueldo - 0.123 * dbSueldo
            Case "PRIMA"
                Return dbSueldo - 0.126 * dbSueldo
            Case "PROFUTURO"
                Return dbSueldo - 0.119 * dbSueldo
            Case "HORIZONTE"
                Return dbSueldo - 0.124 * dbSueldo
        End Select
    End Function
End Class
```

9. En la vista código del formulario Form1 Declarar el método cargarCombo e invocarlo desde el evento Load del formulario.

```
Public Class Form1
    Sub cargarCombos()
        'Cursos
        cboCurso.Items.Add("VB.NET 2008")
        cboCurso.Items.Add("ORACLE")
        cboCurso.Items.Add("WINDOWS 2003 SERVER")
        cboCurso.Items.Add("SQL SERVER 2008")
        cboCurso.SelectedIndex = 0
        'Áreas
```



```
        cboArea.Items.Add("SISTEMAS")
        cboArea.Items.Add("RRHH")
        cboArea.Items.Add("VENTAS")
        cboArea.Items.Add("LOGISTICA")
        cboArea.Items.Add("FINANZAS")
        cboArea.SelectedIndex = 0
        'AFPs
        cboAFP.Items.Add("INTEGRA")
        cboAFP.Items.Add("PRIMA")
        cboAFP.Items.Add("PROFUTURO")
        cboAFP.Items.Add("HORIZONTE")
        cboAFP.SelectedIndex = 0
    End Sub
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        cargarCombos()
    End Sub
End Class
```

10. A nivel del botón Registrar, declare un objeto Docente o Administrativo según sea necesario.

```
Private Sub btnRegistrar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnRegistrar.Click
    'verificar que empleado se eligió
    If cboEmpleado.SelectedIndex = 0 Then
        Dim oDocente As New Docente
        With oDocente
            .Codigo = txtCodigo.Text.Trim
            .Nombres = txtNombres.Text.Trim
            .Apellidos = txtApellidos.Text.Trim
            .Direccion = txtDireccion.Text.Trim
            .DNI = txtDNI.Text.Trim
            .FechaNacimiento = dtpFecNac.Value
            .FechaIngreso = dtpFecIng.Value
            .Curso = cboCurso.SelectedItem.ToString
            .Horas = CDbI(txtHoras.Text.Trim)
        End With
        MessageBox.Show("Registro realizado correctamente" & vbCrLf & _
            "Sueldo docente:" & oDocente.obtenerSueldo.ToString & _
            vbCrLf & "Tiempo de servicio:" & oDocente.obtenerTiempoServicio.ToString)
    Else

        Dim oAdmin As New Administrativo
        With oAdmin
            .Codigo = txtCodigo.Text.Trim
            .Nombres = txtNombres.Text.Trim
            .Apellidos = txtApellidos.Text.Trim
            .Direccion = txtDireccion.Text.Trim
            .DNI = txtDNI.Text.Trim
            .FechaNacimiento = dtpFecNac.Value
            .FechaIngreso = dtpFecIng.Value
            .Area = cboArea.SelectedItem.ToString
            .Sueldo = CDbI(txtSueldo.Text.Trim)
        End With
        MessageBox.Show("Registro realizado correctamente" & vbCrLf & _
            "Sueldo administrativo:" & oAdmin.obtenerSueldo.ToString & _
            vbCrLf & "Tiempo de servicio:" & oAdmin.obtenerTiempoServicio.ToString)
```

```
End If
End Sub
```

11. A nivel del botón Limpiar, coloque este código para borrar los datos del registro.

```
Private Sub btnLimpiar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnLimpiar.Click
    'limpiar todas las cajas de texto de un groupbox
    For Each c As Control In Me.grpDatosGenerales.Controls
        If TypeOf c Is TextBox Then
            CType(c, TextBox).Clear()
        End If
    Next
    txtHoras.Clear()
    txtSueldo.Clear()
    cboEmpleado.SelectedIndex = 0
    cboAFP.SelectedIndex = 0
    cboArea.SelectedIndex = 0
    cboCurso.SelectedIndex = 0
    cboEmpleado.Focus()
    grpDocente.Enabled = False
    grpAdministrativo.Enabled = True
End Sub
```

12. A nivel del combo cboEmpleado en su evento SelectedIndexChanged, coloque este código para habilitar el groupBox que corresponde por el tipo de empleado.

```
Private Sub cboEmpleado_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles cboEmpleado.SelectedIndexChanged
    If cboEmpleado.SelectedIndex = 0 Then
        grpDocente.Enabled = True
        grpAdministrativo.Enabled = False
    Else
        grpAdministrativo.Enabled = True
        grpDocente.Enabled = False
    End If
End Sub
```

## Autoevaluación

1. ¿Qué es una clase?

---

---

---

2. ¿Qué es un objeto?

---

---

---

3. ¿Qué es un formulario?

---

---

---

4. ¿Qué es un control y para que sirve?

---

---

---

## Para Recordar

- Cuando se crea un proyecto de tipo Windows Application, lo primero que se presenta dentro del diseñador es un formulario, el cual es la base para la base de aplicaciones Windows.
- Todo control u objeto presenta propiedades, eventos y métodos los que se permiten manipular su comportamiento en la aplicación.
- Una clase es una agrupación de objetos que poseen características comunes y además posee atributos y operaciones propias, las cuales pueden ser heredadas a sus miembros.

**Fuente:**

**Texto adaptado de la página web:**<http://msdn.microsoft.com/es-es/library/zztsbwsx.aspx>

<b>UNIDAD DE APRENDIZAJE</b>
<b>2</b>
<b>SEMANA</b>
<b>3</b>

## **TIPOS DE DATOS Y COLECCIONES I**

---

### **LOGRO DE LA UNIDAD DE APRENDIZAJE**

Al término de la unidad, los alumnos elaborarán operaciones eficientes de almacenamiento, búsqueda y consultas mediante el empleo de colecciones del namespace System.Collections apropiadas del .NET Framework 2.0.

### **TEMARIO**

- Conociendo los elementos de código fuente de VB .NET: tipos de datos, variables, constantes, arreglos, estructuras, enumeraciones y los operadores aritéticos, lógicos, matemáticos y ternario.
- Colecciones en .NET Framework, definición, métodos y propiedades
- Trabajar con ArrayList, HashTable, Stack y Queue

### **ACTIVIDADES PROPUESTAS**

- Los alumnos conocen las colecciones del .NET Framework
- Los alumnos desarrollan aplicaciones Windows utilizando colecciones para almacenar registros.
- Los alumnos desarrollan los laboratorios de esta semana

## 1. Tipos de Datos

Un tipo de dato determina la naturaleza de los valores que puede tomar una variable., las operaciones en las que puede participar y el espacio de memoria que necesita.

Tipo de Visual Basic	Estructura de tipo Common Language Runtime	Asignación de almacenamiento nominal	Intervalo de valores
Boolean	Boolean	En función de la plataforma de implementación	True o False
Byte	Byte	1 byte	0 a 255 (sin signo)
Char (carácter individual)	Char	2 bytes	0 a 65535 (sin signo)
Date	DateTime	8 bytes	0:00:00 (medianoche) del 1 de enero de 0001 a 11:59:59 p.m. del 31 de diciembre de 9999.
Decimal	Decimal	16 bytes	0 a +/- 79.228.162.514.264.337.593.543.950.335 (+/-7,9... E+28) <sup>†</sup> sin separador decimal; 0 a +/- 7,9228162514264337593543 950335 con 28 posiciones a la derecha del decimal; el número distinto de cero más pequeño es +/- 0,0000000000000000000000 000001 (+/-1E-28) <sup>†</sup>
Double (punto flotante de precisión doble)	Double	8 bytes	-1,79769313486231570E+308 a -4,94065645841246544E-324 <sup>†</sup> para los valores negativos; 4,94065645841246544E-324 a 1,79769313486231570E+308 <sup>†</sup> para los valores positivos
Integer	Int32	4 bytes	-2.147.483.648 a 2.147.483.647 (con signo)
Long (entero largo)	Int64	8 bytes	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807 (9,2...E+18 <sup>†</sup> ) (con signo)
Object	Object (clase)	4 bytes en plataforma de 32 bits	Cualquier tipo puede almacenarse en una variable de tipo Object

		8 bytes en plataforma de 64 bits	
SByte	SByte	1 byte	-128 a 127 (con signo)
Short (entero corto)	Int16	2 bytes	-32.768 a 32.767 (con signo)
Single (punto flotante de precisión simple)	Single	4 bytes	-3,4028235E+38 a -1,401298E-45 <sup>†</sup> para los valores negativos; 1,401298E-45 a 3,4028235E+38 <sup>†</sup> para los valores positivos
String (longitud variable)	String (clase)	En función de la plataforma de implementación	0 a 2.000 millones de caracteres Unicode aprox.

<sup>†</sup> En la notación científica, "E" hace referencia a una potencia de 10. Por lo tanto, 3,56E+2 significa 3.56 x 10<sup>2</sup> o 356, y

## 2. Uso eficiente de tipos de datos

A las variables no declaradas y a las variables declaradas sin un tipo de datos se les asigna el tipo de datos Object. Esto facilita la creación más rápida de programas, pero puede provocar que se ejecuten de una forma más lenta.

### Establecimiento inflexible de tipos

La especificación de tipos de datos para todas las variables recibe el nombre de establecimiento inflexible de tipos. La utilización del establecimiento inflexible de tipos tiene diversas ventajas:

- Habilita la compatibilidad con IntelliSense<sup>®</sup> para las variables. Esto le permite ver las propiedades de las variables y otros miembros a medida que escribe el código.
- Saca partido de la comprobación de tipos del compilador. Permite detectar las instrucciones que pueden fallar en tiempo de ejecución debido a errores tales como el desbordamiento. También detecta llamadas a métodos en objetos que no las admiten.
- Tiene como consecuencia una ejecución más rápida del código.

### Tipos de datos más eficaces

Para las variables que no contienen nunca valores decimales, los tipos de datos integrales son más eficientes que los tipos no integrales. En Visual Basic, Integer y UInteger son los tipos numéricos más eficaces.

Para los números fraccionarios, Double es el más eficaz de los tipos de datos, porque los procesadores de las plataformas actuales realizan las operaciones de punto flotante en precisión doble. Sin embargo, las operaciones con Double no son tan rápidas como con los tipos integrales como Integer.

## Variables

Una variable tiene un nombre (la palabra que se usa para referirse al valor que contiene la variable). Una variable también tiene un tipo de datos, que determina el tipo de datos que puede almacenar la variable.

Utilice Instrucción *Dim* (*Visual Basic*) para declarar una variable de un tipo específico. Puede especificar su nivel de acceso simultáneamente utilizando la palabra clave **Public** (*Visual Basic*), **Protected** (*Visual Basic*), **Friend** (*Visual Basic*) o **Private** (*Visual Basic*).

**Dim** nombre\_variable **as** Tipo

**Dim** nombre\_variable **as** Tipo = valor

Visual Basic 2008 proporciona inferencia de tipo local, que le permite declarar las variables sin tener que indicar de forma explícita un tipo de datos. En lugar de ello, el compilador deduce el tipo de la variable a partir del tipo de la expresión de inicialización.

**Public Sub** inferenceExample()

```
' Usar el tipo explícitamente
Dim num1 As Integer = 3

' Usar un tipo local inferido
Dim num2 = 3
```

**End Sub**

### Nota:

La inferencia de tipo de variable local no se puede utilizar para declarar los campos de clase. Si `num2`, en el ejemplo anterior, fuese un campo en vez de una variable local, la declaración provocaría un error con la instrucción `Option Strict` activada y clasificaría `num2` como `Object` con la instrucción `Option Strict` desactivada.

## Option Infer

Una nueva opción, `Option Infer`, permite especificar si la inferencia de tipo de variable local se permite en un archivo determinado. Para habilitar o bloquear la opción, escriba una de las instrucciones siguientes al principio del archivo.

`Option Infer On`

`Option Infer Off`

Si no especifica ningún valor para `Option Infer` en el código, el valor predeterminado del compilador es `Option Infer On` para los proyectos creados en Visual Basic 2008 y `Option Infer Off` para los proyectos actualizados a partir de versiones anteriores. Para obtener más información, consulte `Option Infer` (Instrucción) y `/optioninfer`.



## Matrices O Arreglos

Una dimensión es una dirección en la que puede variar la especificación de los elementos de una matriz. Una matriz que contiene el total de ventas de todos los días del mes tiene una dimensión (el día del mes). Una matriz que contiene el total de ventas por departamento de todos los días del mes tiene dos dimensiones (el número del departamento y el día del mes). El número de dimensiones que tiene una matriz se denomina rango.

### Trabajar con dimensiones

Para especificar un elemento de una matriz, proporcione un índice o un subíndice para cada una de sus dimensiones. Los elementos son contiguos a lo largo de cada dimensión del índice 0 al índice más alto para esa dimensión.

Las ilustraciones siguientes muestran la estructura conceptual de matrices con rangos diferentes. Cada elemento de las ilustraciones muestra los valores de índice que tienen acceso a él. Por ejemplo, puede tener acceso al primer elemento de la segunda fila de la matriz bidimensional especificando los índices (1, 0).

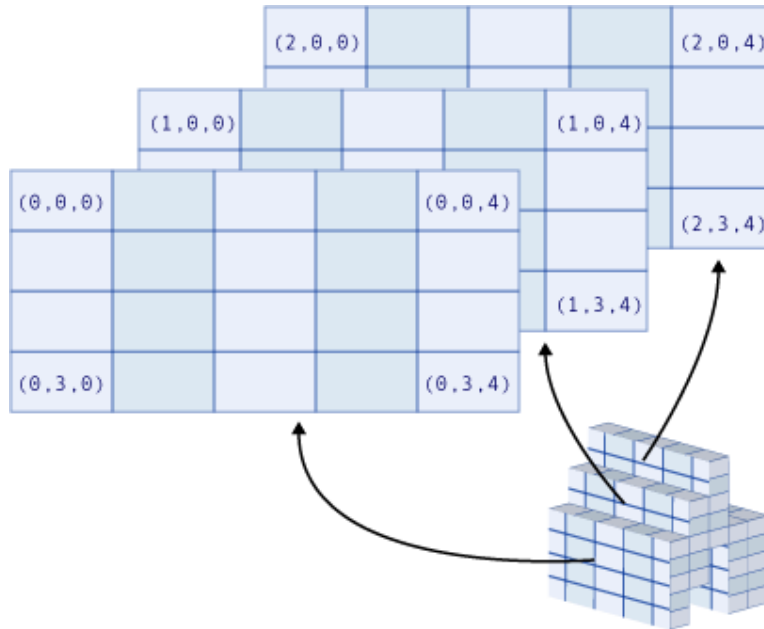
#### Matriz unidimensional

(0)	(1)	(2)	(3)	(4)
-----	-----	-----	-----	-----

#### Matriz bidimensional

(0,0)				(0,4)
(1,0)				
(3,0)				(3,4)

## Matriz tridimensional



### Declaración de una matriz unidimensional

**Dim** nombre\_matriz () **as** Tipo = **new** Tipo () {}

Esta declaración nos muestra una matriz de longitud cero.

Utilice Instrucción ReDim (Visual Basic) no sólo para crear una matriz sino para inicializar su longitud.

**ReDim** nombre\_matriz (Índice máximo)

#### Nota:

La cláusula New debe especificar el nombre de tipo, seguido de paréntesis y luego llaves, {}. Los paréntesis no representan una llamada a un constructor de matriz. Indican que el tipo de objeto es un tipo de matriz. Las llaves proporcionan los valores de inicialización. El compilador requiere las llaves aunque no proporcionen ningún valor. Por consiguiente, la cláusula New debe incluir paréntesis y llaves, aunque estén vacíos.

Ejemplos de declaraciones de matrices

**Dim** edades(10) **As** Int32

**Dim** contadores(20, 5) **As** Byte

**Dim** temperaturaDeAire(10,10, 8) **As** Single

## Constantes de Visual Basic

Las constantes almacenan valores que, como su nombre indica, permanecen constantes durante la ejecución de una aplicación. Utilice las constantes de su código para valores que se utilizan repetidamente o para facilitar la lectura del código.

**Public Const** DaysInYear = 365

**Private Const** WorkDays = 250

**Public Const** MyInteger **As Integer** = 42

**Private Const** DaysInWeek **As Short** = 7

**Protected Friend Const** Funday **As String** = "Sunday"

'Declaración de varias constantes

**Public Const** Four **As Integer** = 4, Five **As Integer** = 5, Six **As Integer** = 44

## Enumeraciones de Visual Basic

Una enumeración se crea con la instrucción Enum en la sección de declaraciones de una clase o módulo. No puede declarar una enumeración dentro de un método. Para especificar el nivel adecuado de acceso, utilice Private, Protected, Friend o Public.

Un tipo Enum tiene un nombre, un tipo subyacente y un conjunto de campos, cada uno de los cuales representa una constante. El nombre debe ser un calificador de Visual Basic 2005 válido. El tipo subyacente debe ser uno de los tipos de enteros —Byte, Short, Long o Integer. Integer es el valor predeterminado. Las enumeraciones tienen siempre establecimiento inflexible de tipos y no son intercambiables con los tipos de números enteros.

Las enumeraciones no pueden tener valores de punto flotante. Si se asigna un valor de punto flotante a una enumeración con Option Strict On, se producirá un error del compilador. Si Option Strict es Off, el valor se convierte automáticamente en el tipo Enum.

[Private] o [Public] o [Friend] o [Protected] Enum Enumeracion1

Miembros

End Enum

### Ejemplo

**Public Enum** Dias

Sunday 'valor 0

Monday 'valor 1

Tuesday 'valor 2

Wednesday 'valor 3

Thursday 'valor 4

Friday 'valor 5

Saturday 'valor 6

**End Enum**

'recorrer elementos de la enumeración

Dim items As Array

items = System.Enum.GetValues(GetType(Dias))

Dim item As String

For Each item In items

    MessageBox.show(item)

Next

## Operadores de Visual Basic

Visual Basic proporciona los tipos de operadores siguientes:

- **Operadores aritméticos**, realizan los cálculos familiares en valores numéricos, incluido el desplazamiento de sus modelos de bits.

Operador	Operación
^	Potencia
*	Multiplicación
/	División real
Mod	Residuo
+	Suma
-	Resta
\	División entera

- **Operadores de comparación**, comparan dos expresiones y devuelven un valor Boolean que representa el resultado de la comparación.

Operador	Operación
=	Igual
<>	Diferente de
<	Menor que
>	Mayor que
>=, <=	Mayor igual o Menor igual
like	Compara dos cadenas. * → Cero o más caracteres ? → Cualquier carácter #-> Cualquier dígito [lista] → cualquier carácter en lista [!lista] → cualquier

	carácter que no este en lista
--	-------------------------------

- **Operadores de concatenación**, combinan varias cadenas en una sola.

Operador	Operación
&	Concatena cadenas

- **Operadores lógicos en Visual Basic**, combinan valores Boolean o numéricos y devuelven un resultado del mismo tipo de datos que los valores.

Operador	
And	"Y" lógico
Or	"O" lógico
Xor	"O" exclusivo
Not	Negación

Los elementos de valor que se combinan con un operador se denominan operandos de ese operador. Los operadores combinados con los elementos de valor forman expresiones, salvo el operador de asignación, que forma una instrucción.

### Operador ternario

En inglés: Ternary operator. Esto es algo que Visual Basic nunca ha tenido y muchos "pedíamos", en C/C# es el operador "?" : que se usa de esta forma:

```
String s = 10>5? "10 es mayor que 5":"10 no es mayor que 5";
```

Es decir, se evalúa una expresión y si se cumple, se usa lo que hay después de la interrogación?, si no se cumple, se usa lo que haya después de los dos puntos :

En Visual Basic 9.0 se usa con el IF normal, pero en "modo función", y sería el equivalente a IIF, con la ventaja de que el tipo de datos devuelto no hay que "convertirlo", ya que si lo que se devuelve es una cadena, pues el tipo devuelto (o valor) es una cadena, que es un tipo Double, pues... es decir, está "mejorado" con respecto a lo que teníamos con el actual IIF.

La forma de usar el operador ternario en Visual Basic 2008 es esta:

```
Dim s As String = Iff(10 > 5, "10 es mayor que 5", "10 no es mayor que 5")
```

## 1. System.Collections

El espacio de nombres **System.Collections** contiene interfaces y clases que definen varias colecciones de objetos, como listas, colas, matrices de bits, tablas hash y diccionarios.

Clase	Descripción
<a href="#">ArrayList</a>	Implementa la interfaz <a href="#">IList</a> mediante una matriz cuyo tamaño aumenta dinámicamente según se requiera.
<a href="#">CollectionBase</a>	Proporciona la clase base abstract para colecciones con establecimiento inflexible de tipos.
<a href="#">Comparer</a>	Compara dos objetos para ver si son iguales teniendo en cuenta la distinción entre mayúsculas y minúsculas de las cadenas.
<a href="#">Hashtable</a>	Representa una colección de pares de clave y valor organizados en función del código hash de la clave.
<a href="#">Queue</a>	Representa una colección de objetos de tipo "primero en entrar, primero en salir".
<a href="#">SortedList</a>	Representa una colección de pares de clave y valor ordenados por claves a los que se puede tener acceso por clave y por índice.
<a href="#">Stack</a>	Representa una colección sencilla de objetos no genéricos LIFO ("último en entrar, primero en salir").

### 1.1. ArrayList (Clase)

Implementa la interfaz [IList](#) mediante una matriz cuyo tamaño aumenta dinámicamente según se requiera.

No se garantiza que la matriz **ArrayList** esté ordenada. Debe ordenar la matriz **ArrayList** antes de realizar operaciones (como [BinarySearch](#)) que requieren que la matriz **ArrayList** esté ordenada.

La capacidad de un objeto **ArrayList** es el número de elementos que puede contener el objeto **ArrayList**. A medida que se agregan elementos a **ArrayList**, la capacidad aumenta automáticamente según lo requiera la reasignación. Se puede disminuir la capacidad llamando al método [TrimToSize](#) o estableciendo explícitamente la propiedad [Capacity](#).

Se puede obtener acceso a los elementos de esta colección utilizando un índice entero. Los índices de esta colección están basados en cero.

El objeto **ArrayList** acepta `Nothing` como valor nulo y permite elementos duplicados.

Public Class EjemplosArrayList

```
Public Sub MostrarValues(ByVal myList As ArrayList)
    Dim obj As Object
    For Each obj In myList
        ListBox1.Items.Add(obj)
    Next
```

End Sub

```
Private Sub EjemplosArrayList_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
```

```
    ' Crear e inicializar un ArrayList.
```

```
    Dim myAL As New ArrayList()
```

```
    myAL.Add("Hello")
```

```
    myAL.Add("World")
```

```
    myAL.Add("!")
```

```
    ' Mostrar datos
```

```
    ListBox1.Items.Add("ArrayList: myAL")
```

```
    ListBox1.Items.Add("Cantidad " & vbTab & myAL.Count)
```

```
    ListBox1.Items.Add("Capacidad" & vbTab & myAL.Capacity)
```

```
    ListBox1.Items.Add("Valores:")
```

```
    MostrarValues(myAL)
```

End Sub

End Class

## 1.2. Hashtable (Clase)

Representa una colección de pares de clave y valor organizados en función del código hash de la clave.

Cada elemento es un par de clave y valor almacenado en un objeto DictionaryEntry. Una clave no puede ser nullNothingnullptrreferencia null (Nothing en Visual Basic), pero un valor sí puede serlo.

Los objetos de claves deben permanecer inmutables mientras se utilicen como claves en **Hashtable**.

Cuando se agrega un elemento a **Hashtable**, el elemento se coloca en un sector de almacenamiento en función del código hash de la clave. Las búsquedas posteriores de la clave utilizarán su código hash para buscar en un sector de almacenamiento determinado solamente; de este modo, se reducirá considerablemente el número de comparaciones de clave necesarias para encontrar un elemento.

```
'crear elementos al HashTable
```

```
Dim openWith As New Hashtable()
```

```
'agregar elementos al HashTable
```

```
openWith.Add("txt", "notepad.exe")
```

```
openWith.Add("bmp", "paint.exe")
```

```
openWith.Add("dib", "paint.exe")
```

```
openWith.Add("rtf", "wordpad.exe")
```

```
'se puede omitir el tipo de la variable
```

```
Dim cadena = ""
```

```

'recorrer elementos del HashTable
For Each de As DictionaryEntry In openWith
    cadena &= de.Key & "-" & de.Value & vbCrLf
Next
MessageBox.Show(cadena)
If Not openWith.ContainsKey("doc") Then
    MessageBox.Show("La clave ""doc"" no se encontró")
End If

```

### 1.3. Queue (Clase)

Representa una colección de objetos de tipo "primero en entrar, primero en salir".

Las colas son útiles para almacenar mensajes en el orden en el que fueron recibidos para el procesamiento secuencial. Esta clase implementa una cola como una matriz circular. Los objetos almacenados en **Queue** se insertan en un extremo y se quitan del otro.

La capacidad de **Queue** es el número de elementos que **Queue** puede contener. A medida que se agregan elementos a **Queue**, la capacidad aumenta automáticamente según lo requiera la reasignación. La capacidad se puede disminuir si se llama al método **TrimToSize**.

El factor de crecimiento es el número por el cual se multiplica la capacidad actual cuando se requiere una capacidad mayor. El factor de crecimiento se determina al construir la clase **Queue**. El factor de crecimiento predeterminado es 2,0. La capacidad de **Queue** siempre aumentará su capacidad en cuatro, como mínimo, con independencia del factor de crecimiento. Por ejemplo, un objeto **Queue** con un factor de crecimiento igual a 1,0 siempre aumentará su capacidad en cuatro cuando se requiera una capacidad mayor.

```

Dim cola As New Queue
Dim num1 = 10
Dim num2 = 20
Dim num3 = 35
'agrega objeto a la cola
cola.Enqueue(num1)
cola.Enqueue(num2)
cola.Enqueue(num3)
'mostrar contenido eliminando elementos de cola
While Not cola.Count > 0
    MessageBox.Show("Número:" & cola.Dequeue)
End While
MessageBox.Show("Cantidad de elementos:" & cola.Count)

```

### 1.4. Stack (Clase)

Representa una colección sencilla de objetos no genéricos LIFO ("último en entrar, primero en salir").

**Stack** se implementa como un búfer circular.

La capacidad de **Stack** es el número de elementos que **Stack** puede contener. A medida que se agregan elementos a **Stack**, la capacidad aumenta automáticamente según lo requiera la reasignación.



Si **Count** es menor que la capacidad de la pila, **Push** es una operación  $O(1)$ . Si es necesario aumentar la capacidad para alojar el nuevo elemento, **Push** se convierte en una operación  $O(n)$ , en la que  $n$  es **Count**. **Pop** es una operación  $O(1)$ .

```
Public class FrmEjemplo1
```

```
    Dim myStack As New Stack
```

```
    'agregar elementos a la pila
```

```
    myStack.Push("Hello")
```

```
    myStack.Push("World")
```

```
    myStack.Push("!")
```

```
    ' muestra cantidad de elementos
```

```
    MessageBox.Show("Stack : myStack" & vbCrLf & _
```

```
    "Cantidad:" & myStack.Count)
```

```
    'recorre los elementos de la pila
```

```
    MostrarValues(myStack)
```

```
Public Sub MostrarValues(ByVal myCollection As Stack)
```

```
    While myCollection.Count > 0
```

```
        MessageBox.Show(myCollection.Pop())
```

```
    End While
```

```
End Sub
```

```
End class
```

## Laboratorio 3.1

### Manejo de Colecciones desde la propiedad Items de los objetos Listbox y Combobox

1. Ingrese a Visual Studio 2008
2. Seleccione menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Application" y en Name escriba "Laboratorio3\_1".
4. Diseñe la siguiente GUI del formulario



5. El control Listbox debe tener en su propiedad SelectionMode el valor MultiExtended.
6. En el evento click del btnAgregar coloque este código:

```
Private Sub btnAgregarG_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnAgregarG.Click
```

```
    'Agregar un elemento a la colección Items del Listbox
    'validar que no esté vacía la caja y que no se repita
    If lstGaseosas.FindStringExact(txtGaseosa.Text.Trim) Then
        If txtGaseosa.Text.Trim <> String.Empty Then
            lstGaseosas.Items.Add(txtGaseosa.Text.Trim.ToUpper)
            txtGaseosa.Clear()
            txtGaseosa.Focus()
        End If
    End If
End Sub
```

7. En el evento click del btnEliminar coloque este código:

```
Private Sub btnEliminarG_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEliminarG.Click
    'validar si hay un elemento seleccionado
    If lstGaseosas.SelectedIndex <> -1 Then
        'eliminar elemento seleccionado
        lstGaseosas.Items.RemoveAt(lstGaseosas.SelectedIndex)
    End If
End Sub
```

8. En el evento click del btnEliminarxIndiceG coloque este código

```
Private Sub btnEliminarxIndiceG_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnEliminarxIndiceG.Click

    Dim strIndice = InputBox("Ingrese índice")
    If IsNumeric(strIndice) And Val(strIndice) < lstGaseosas.Items.Count Then
        lstGaseosas.Items.RemoveAt(CInt(strIndice))
    End If

End Sub
```

9. En el evento click del botón InsertarxIndice coloque este código

```
Private Sub btnInsertar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnInsertar.Click
    Dim strIndice = InputBox("Ingrese índice")
    If IsNumeric(strIndice) And Val(strIndice) <= lstGaseosas.Items.Count Then
        lstGaseosas.Items.Insert(CInt(strIndice), txtGaseosa.Text.Trim)
    End If
End Sub
```

10. En el evento click del botón Limpiar Todo coloque este código

```
Private Sub btnLimpiar2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnLimpiar2.Click
    lstGaseosas.Items.Clear()
End Sub
```

11. En el evento click del botón Agregar en la sección del ComboBox. Coloque este código

```
Private Sub btnAgregar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnAgregar.Click
    'Agregar un elemento a la colección Items del ComboBox
    If cboDeportes.FindStringExact(txtDeporte.Text.Trim) = -1 Then
        If txtDeporte.Text.Trim <> String.Empty Then
            cboDeportes.Items.Add(txtDeporte.Text.Trim.ToUpper)
        End If
    End If
    txtDeporte.Clear()
    txtDeporte.Focus()
End Sub
```

12. En el evento click del botón Eliminar en la sección del ComboBox. Coloque este código

```
Private Sub btnEliminarD_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEliminarD.Click
    'validar si hay un elemento seleccionado
    If cboDeportes.SelectedIndex <> -1 Then
        'eliminar elemento seleccionado
        cboDeportes.Items.RemoveAt(cboDeportes.SelectedIndex)
    End If
End Sub
```

13. En el evento click del botón Limpiar en la sección del ComboBox. Coloque este código

```
Private Sub btnLimpiar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnLimpiar.Click
    cboDeportes.Items.Clear()
End Sub
```

14. Ejecutar la aplicación con F5, evaluar los resultados y luego cierre el formulario.

## Laboratorio 3.2

### Manejo de Colecciones del namespace System.Collections

1. Ingrese a Visual Studio 2008
2. Seleccione menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Application" y en Name escriba "Laboratorio3\_2".
4. Diseñe la siguiente GUI del formulario

5. Desde la vista código del formulario Form1, digite el siguiente código

Public Class Form1

'declaración de colecciones

Dim miLista As New ArrayList

Dim miTabla As New Hashtable

Dim cola As New Queue

Dim pila As New Stack

Private Sub btnAgregarA\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnAgregarA.Click

'verificar si elemento existe

If miLista.Contains(txtGaseosa.Text.Trim) = False Then

'agregar elemento al ArrayList

miLista.Add(txtGaseosa.Text)

Else

MessageBox.Show("Elemento repetido")

End If

```

txtGaseosa.Clear()
txtGaseosa.Focus()
End Sub

Private Sub btnEliminarA_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEliminarA.Click
    'buscar elemento para eliminar
    Dim eleBuscado = InputBox("Ingrese gaseosa a eliminar")
    For Each ele In miLista
        If ele = eleBuscado Then
            miLista.Remove(ele) 'elimina elemento encontrado
            Exit For 'abandona for
        End If
    Next
End Sub

Private Sub btnListarA_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnListarA.Click
    'limpia listbox antes de listar
    lstElementos.Items.Clear()
    'recorre elementos del ArrayList
    For Each ele In miLista
        lstElementos.Items.Add(ele)
    Next
End Sub

Private Sub btnAgregarH_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnAgregarH.Click
    'declara un componente que trabaja con clave y valor
    Dim eleH As DictionaryEntry
    eleH.Key = txtClave.Text
    eleH.Value = txtGaseosa2.Text
    'verifica si el elemento existe en Hashtable
    If miTabla.Contains(eleH) = False Then
        'agrega clave y valor como elemento del Hashtable
        miTabla.Add(txtClave.Text.Trim, txtGaseosa2.Text.Trim)
    Else
        MessageBox.Show("Elemento repetido")
    End If
    txtGaseosa2.Clear()
    txtClave.Clear()
    txtClave.Focus()
End Sub

Private Sub btnEliminarH_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEliminarH.Click
    'buscar elemento para eliminar
    Dim claveBuscada = InputBox("Ingrese clave de gaseosa a eliminar")
    'verifica si clave existe
    If miTabla.ContainsKey(claveBuscada) = True Then
        miTabla.Remove(claveBuscada) 'elimina elemento
    Else
        MessageBox.Show("Clave no registrada")
    End If
End Sub

```

```
Private Sub btnListarH_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnListarH.Click
    'borra elementos del listbox
    lstElementosH.Items.Clear()
    'muestra el contenido actual del hashtable
    For Each ele As DictionaryEntry In miTabla
        lstElementosH.Items.Add(ele.Key & vbTab & ele.Value)
    Next
End Sub

Private Sub btnAgregarQ_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnAgregarQ.Click
    'verifica si elemento existe en cola
    If cola.Contains(txtCurso.Text) = False Then
        'agrega elemento a la cola
        cola.Enqueue(txtCurso.Text.Trim)
        txtCurso.Clear()
        txtCurso.Focus()
    Else
        MessageBox.Show("Elemento repetido")
    End If
End Sub

Private Sub btnListarQ_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnListarQ.Click
    'borra elementos del listbox
    lstElementosQ.Items.Clear()
    'recorre los elementos de la cola y los elimina luego de mostrarlos
    While cola.Count > 0
        lstElementosQ.Items.Add(col.Dequeue)
    End While
End Sub

Private Sub btnAgregarS_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnAgregarS.Click
    'verifica si elemnto existe en pila
    If pila.Contains(txtDeporte.Text) = False Then
        'agrega elemento a la pila
        pila.Push(txtDeporte.Text.Trim)
        txtDeporte.Clear()
        txtDeporte.Focus()
    Else
        MessageBox.Show("Elemento repetido")
    End If
End Sub

Private Sub btnListarS_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnListarS.Click
    'borra elementos del listbox
    lstElementosS.Items.Clear()
    'recorre los elementos de la pila y los elimina luego de mostrarlos
    While pila.Count > 0
```

```
        IstElementosS.Items.Add(pila.Pop)
    End While
End Sub

End Class
```

6. Ejecute la aplicación con F5 y compruebe los resultados.



## Autoevaluación

1. ¿Qué propiedad del Listbox o combobox representa una colección?

---

---

---

2. ¿Qué colecciones pertenecen al namespace System.Collections?

---

---

---

3. ¿Qué tipo de dato reciben estas colecciones?

---

---

---

4. ¿Qué diferencia existe entre una colección Stack y una colección de tipo Queue?

---

---

---

## Para Recordar

- Visual Basic también proporciona una clase **Collection**, con la que puede definir y crear sus propias colecciones.
- Las instancias de la clase **Collection** de Visual Basic permiten tener acceso a un elemento utilizando un índice numérico o una clave **String**. Puede agregar elementos a los objetos **Collection** de Visual Basic especificando una clave o sin especificarla. Si agrega un elemento sin una clave, debe utilizar su índice numérico para tener acceso a él.

Por contraste, las colecciones como **System.Collections...::ArrayList** sólo permiten un índice numérico. No se pueden asociar claves con los elementos de estas colecciones, a menos que construya las suyas propias basadas en asignaciones, por ejemplo, en una matriz de **String** que contiene las claves.

- Aunque el objeto **Collection** de Visual Basic tiene una funcionalidad idéntica al objeto **Collection** de Visual Basic 6.0, estos objetos no pueden interoperar en un entorno COM.
- El espacio de nombres **System.Collections** contiene interfaces y clases que definen varias colecciones de objetos, como listas, colas, matrices de bits, tablas hash y diccionarios.

### Fuente:

#### Texto adaptado de la página web:

<http://msdn.microsoft.com/es-es/library/a1y8b3b3.aspx>

**UNIDAD DE  
APRENDIZAJE**

**2**

**SEMANA**

**4**

## **Desarrollo de Aplicaciones con Colecciones II**

---

### **LOGRO DE LA UNIDAD DE APRENDIZAJE**

Al término de la unidad, los alumnos elaborarán operaciones eficientes de almacenamiento, búsqueda y consultas mediante el empleo de colecciones genéricas apropiadas del .NET Framework 2.0.

### **TEMARIO**

- Colecciones Genéricas: List y Dictionary
- Manejo de errores: Try - Catch

### **ACTIVIDADES PROPUESTAS**

- Los alumnos crean aplicaciones que almacenan información en colecciones genéricas.
- Los alumnos crean aplicaciones que manejan errores con el bloque Try - catch.
- Los alumnos desarrollan los laboratorios dirigidos de esta semana.

## 1. Colecciones Genéricas

El espacio de nombres **System.Collections.Generic** contiene interfaces y clases que definen colecciones genéricas, permitiendo que los usuarios creen colecciones con establecimiento inflexible de tipos para proporcionar una mayor seguridad de tipos y un rendimiento mejor que los de las colecciones no genéricas con establecimiento inflexible de tipos.

### 1.1. List(Of T) (Clase)

Representa una lista de objetos con establecimiento inflexible de tipos a la que se puede obtener acceso por índice. Proporciona métodos para buscar, ordenar y manipular listas.

La clase **List(Of T)** es el equivalente genérico de la clase [ArrayList](#). Implementa la interfaz genérica [IList \(Of T\)](#) mediante una matriz cuyo tamaño aumenta dinámicamente según se requiera.

La clase **List(Of T)** utiliza un comparador de igualdad y un comparador de orden.

- Los métodos como [Contains](#), [IndexOf](#), [LastIndexOf](#) y [Remove](#) utilizan un comparador de igualdad para los elementos de lista.
- Los métodos como [BinarySearch](#) y [Sort](#) utilizan un comparador de orden para los elementos de lista.

No se garantiza que el objeto **List(Of T)** esté ordenado. Debe ordenar **List(Of T)** antes de realizar operaciones (como [BinarySearch](#)) que requieran que dicho objeto **List(Of T)** esté ordenado.

Se puede obtener acceso a los elementos de esta colección utilizando un índice entero. Los índices de esta colección están basados en cero.

**Public Sub** Colecciones()

**Dim** dinosaurs **As New** List (Of String)

MessageBox.show("Capacity:" & dinosaurs.Capacity)

dinosaurs.Add("Tyrannosaurus")  
dinosaurs.Add("Amargasaurus")  
dinosaurs.Add("Mamenchisaurus")  
dinosaurs.Add("Deinonychus")  
dinosaurs.Add("Compsognathus")

**For Each** dinosaur **As String In** dinosaurs  
    MessageBox.show((dinosaur))

**Next**

**End Sub**

### Métodos

Nombre	Descripción
<a href="#">Add</a>	Agrega un objeto al final de <a href="#">List&lt; (Of &lt; T)&gt;</a> .
<a href="#">BinarySearch</a>	Sobrecargado. Utiliza un algoritmo de búsqueda binaria para localizar un elemento concreto en la <a href="#">List&lt; (Of &lt; T)&gt;</a> ordenada o en una parte de ella.

Clear	Quita todos los elementos de <code>List&lt; Of &lt; (T)&gt;&gt;</code> .
Contains	Determina si un elemento se encuentra en <code>List&lt;(Of &lt;(T)&gt;&gt;</code> .
Equals	Determina si el objeto <code>Object</code> especificado es igual al objeto <code>Object</code> actual. (Se hereda de <code>Object</code> ).
Exists	Determina si <code>List&lt; (Of &lt; (T)&gt;&gt;</code> contiene elementos que cumplen las condiciones definidas por el predicado especificado.
Find	Busca un elemento que cumpla las condiciones definidas por el predicado especificado y devuelve la primera aparición en todo el objeto <code>List&lt; (Of &lt; (T)&gt;&gt;</code> .
FindAll	Recupera todos los elementos que coinciden con las condiciones definidas por el predicado especificado.
FindIndex	Sobrecargado. Busca un elemento que cumpla las condiciones definidas por un predicado especificado y devuelve el índice de base cero de la primera aparición en el objeto <code>List&lt;(Of &lt;(T)&gt;&gt;</code> o en una parte de él.
FindLast	Busca un elemento que coincida con las condiciones definidas por el predicado especificado y devuelve la última aparición en todo el objeto <code>List&lt; (Of &lt;(T)&gt;&gt;</code> .
FindLastIndex	Sobrecargado. Busca un elemento que cumpla las condiciones definidas por un predicado especificado y devuelve el índice de base cero de la última aparición en el objeto <code>List&lt;(Of &lt;(T)&gt;&gt;</code> o en una parte de él.
ForEach	Realiza la acción especificada en cada elemento de <code>List&lt; (Of &lt;(T)&gt;&gt;</code> .
GetEnumerator	Devuelve un enumerador que recorre en iteración la colección <code>List&lt;(Of &lt;(T)&gt;&gt;</code> .
IndexOf	Sobrecargado. Devuelve el índice de base cero de la primera aparición de un valor en la <code>List&lt;(Of &lt;(T)&gt;&gt;</code> o en una parte de ella.
Insert	Inserta un elemento en <code>List&lt;(Of &lt;(T)&gt;&gt;</code> , en el índice especificado.
Remove	Quita la primera aparición de un objeto específico de <code>List&lt;(Of &lt;(T)&gt;&gt;</code> .
RemoveAll	Quita todos los elementos que cumplen las condiciones definidas por el predicado especificado.
RemoveAt	Quita el elemento en el índice especificado de <code>List&lt;(Of &lt;(T)&gt;&gt;</code> .
Sort	Sobrecargado. Ordena los elementos en la <code>List&lt; (Of &lt; (T)&gt;&gt;</code> o en una parte de ella.
ToString	Devuelve una clase <code>String</code> que representa la clase <code>Object</code> actual. (Se hereda de <code>Object</code> ).

## 1.2. Dictionary(Of TKey, TValue) (Clase)

La clase genérica **Dictionary (Of (TKey, TValue))** proporciona una asignación de un conjunto de claves a un conjunto de valores. Cada elemento que se agrega al diccionario está compuesto de un valor y su clave asociada. Recuperar un valor utilizando su clave es muy rápido, cerca de  $O(1)$ , porque la clase **Dictionary (Of (TKey, TValue))** se implementa como una tabla hash.

### Nota:

La velocidad de recuperación depende de la calidad del algoritmo hash del tipo especificado para *TKey*.

Mientras se utilice un objeto como clave en el diccionario **Dictionary (Of (TKey, TValue))**, dicho objeto no se debe modificar en modo alguno que afecte a su valor hash. Cada clave de un diccionario **Dictionary(Of (TKey, TValue))** debe ser única conforme al comparador de igualdad del diccionario. Una clave no puede ser nullNothingnullptrreferencia null (Nothing en Visual Basic), aunque un valor sí puede serlo si el tipo de valor *TValue* es un tipo de referencia.

Public Sub Colecciones2()

Crear un objeto Diccionario que maneja clave y valor

Dim openWith As New Dictionary(Of String, String)

'Agregar elementos al objeto Dictionary

openWith.Add("txt", "notepad.exe")

openWith.Add("bmp", "paint.exe")

openWith.Add("dib", "paint.exe")

openWith.Add("rtf", "wordpad.exe")

'validar que no se repita clave

Try

openWith.Add("txt", "winword.exe")

Catch

MessageBox.show ("An element with Key = "txt" already exists.")

End Try

'Mostrar elemento por clave

MessageBox.show(openWith("rtf"))

Try

'Mostrar el elemento cuya clave es "tif"

MessageBox.show(openWith("tif"))

Catch

MessageBox.show( ("Clave no fue encontrada")

End Try

End Sub

## Métodos

Nombre	Descripción
All	Determina si todos los elementos de una secuencia satisfacen una condición. (Definido por Enumerable).
Concat	Concatena dos secuencias. (Definido por Enumerable).
Contains	Sobrecargado.
Count	Sobrecargado.
Distinct	Sobrecargado.
ElementAt	Devuelve el elemento situado en un índice especificado de una secuencia. (Definido por Enumerable).
First	Sobrecargado.
FirstOrDefault	Sobrecargado.
Single	Sobrecargado.
SingleOrDefault	Sobrecargado.
ToArray	Crea una matriz a partir de un objeto IEnumerable<Of <(T)>>. (Definido por Enumerable).
ToList	Crea un objeto List< Of < (T)>> a partir de un objeto IEnumerable<Of <(T)>>. (Definido por Enumerable).

## Laboratorio 4.1

### Manejo de Colecciones Genéricas

#### Se va desarrollar dos clases propias Registro y Comision

1. Ingrese a Visual Studio 2008
2. Seleccione menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Application" y en Name escriba "Laboratorio4\_1".
4. Diseñe la siguiente GUI del formulario .

5. Debemos declarar dos clases propias : clase Comision

Public Class Comision

'atributos privados

Private factura As String

Private vendedor As String

Private monto As Single '(Real)

'constructor

Sub New()

factura = ""

vendedor = ""

monto = 0

End Sub

'propiedades de la clase



```
Public Property p_factura() As String
    Get
        Return factura
    End Get
    Set(ByVal value As String)
        factura = value
    End Set
End Property

Public Property p_vendedor() As String
    Get
        Return vendedor
    End Get
    Set(ByVal value As String)
        vendedor = value
    End Set
End Property

Public Property p_monto() As Single
    Get
        Return monto
    End Get
    Set(ByVal value As Single)
        monto = value
    End Set
End Property
'método con retorno
Public Function obtenerComision() As Single
    Return monto * 0.05
End Function
End Class
```

6. Luego , declarar la clase Registro

```
Public Class ClaseRegistro
    'atributos privados
    Private codigo As String
    Private nombre As String
    Private he As Date

    Sub New() 'definir un constructor
        codigo = ""
        nombre = ""
        he = TimeOfDay 'hora actual
    End Sub

    Public Property p_codigo() As String
        Get
            Return codigo
        End Get
        Set(ByVal value As String)
            codigo = value
        End Set
    End Property
```

```

Public Property p_nombre() As String
    Get
        Return nombre
    End Get
    Set(ByVal value As String)
        nombre = value
    End Set
End Property

Public ReadOnly Property p_hora() As Date
    Get
        Return he
    End Get
End Property
End Class

```

7. Finalmente, codificar el bloque para la colección List y luego la de la colección ArrayList

```

Public Class frmColecciones
    'DEFINES UN GENERICS LIST DE TIPO CLASECOMISION
    Private LCOMISION As New List(Of ClaseComision)
    Private Sub BTNAGREGAR_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles BTNAGREGAR.Click
        'CREAR OBJETO DE TIPO CLASECOMISION
        Dim REG As New ClaseComision
        'ASIGNAR VALORES A LAS PROPIEDADES DEL OBJETO
        REG.p_factura = TXTFACTURA.Text
        REG.p_vendedor = TXTVENDEDOR.Text
        REG.p_monto = Val(TXTMONTO.Text)
        'AGREGAR OBJETO A LA COLECCION LIST
        LCOMISION.Add(REG)
        'AGREGAR VALORES DEL OBJETO EN EL CONTROL LISTBOX
        ListBox1.Items.Add(REG.p_factura + vbTab + _
            REG.p_vendedor + vbTab + REG.obtenerComision.ToString)
    End Sub
    Private Sub BTNQUITAR_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles BTNQUITAR.Click
        'SI HAS SELECCIONADO: SELECTEDINDEX > -1
        If ListBox1.SelectedIndex > -1 Then
            'ELIMINAR DEL ARRAYLIST
            LCOMISION.RemoveAt(ListBox1.SelectedIndex)
            'ELIMINA DEL CONTROL LISTBOX
            ListBox1.Items.RemoveAt(ListBox1.SelectedIndex)
        End If
    End Sub
    Private Sub BTNCALCULAR_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles BTNCALCULAR.Click
        'PARA CALCULAR SE VA A LEER LOS ELEMENTOS
        Dim REG As New ClaseComision 'VARIABLE PARA RECORRER COLECCION
        LCOMISION
        Dim COMISION As Single = 0 'ACUMULADOR
        'POR CADA REG CONTENIDO EN EL GENERICS LCOMISION
        For Each REG In LCOMISION
            COMISION += REG.obtenerComision
        Next REG
    End Sub
End Class

```

```
Next
Me.TXTCOMISION.Text = COMISION
End Sub

'DEFINIR AL DICTIONARY REGISTRO
'Dictionary(Of TipoDato de clave, TipoDato de elemento)
Private DREGISTRO As New Dictionary(Of String, String)
'Private nombreObjeto As New
'Dictionary(Of Tipo de dato para clave, Tipo de dato para elemento

Private Sub BTNAGREGAR2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BTNAGREGAR2.Click
    'PREGUNTAR SI EXISTE EL CODIGO(clave) A INGRESAR
    'ContainsKey(clave)-->Busca si existe la clave en el objeto Dictionary
    If DREGISTRO.ContainsKey(TXTCODIGO.Text) Then
        MessageBox.Show("YA EXISTE EL CODIGO")
    Else
        'AGREGAR LOS ELEMENTOS AL objeto DICTIONARY
        'objetoDictionary.Add(clave,elemento)
        DREGISTRO.Add(TXTCODIGO.Text, TXTNOMBRE.Text)
        MessageBox.Show("REGISTRADO")
        LISTADO()
    End If
End Sub

'método sin retorno
Sub LISTADO()
    ListBox1.Items.Clear() 'LIMPIAR LA LISTA
    'DEFINO EL KEYVALUEPAIR LLAMADO R
    ' Dim R As KeyValuePair(of TipoDato Clave,TipoDato de elemento)
    Dim R As KeyValuePair(Of String, String)
    'POR CADA R DENTRO DE DREGISTRO(Dictionary)
    For Each R In DREGISTRO
        'R.Key-->obtiene clave
        'R.Value-->obtiene valor de elemento
        ListBox1.Items.Add(R.Key + vbTab + R.Value)
    Next
End Sub

Private Sub BTNQUITAR2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BTNQUITAR2.Click
    'DEFINO UN INPUTBOX PARA INGRESAR EL CODIGO
    Dim COD As String = InputBox("INGRESE EL CODIGO")
    If DREGISTRO.ContainsKey(COD) = True Then
        'ELIMINAR elemento por código de dictionary
        DREGISTRO.Remove(COD)
        'LISTAR LOS ELEMENTOS
        LISTADO()
    Else
        MessageBox.Show("CODIGO NO EXISTE")
    End If
End Sub
End Class
```

## Laboratorio 4.2

### Manejo de Excepciones

#### Se va a desarrollar dos clases propias Registro y Comision

1. Ingrese a Visual Studio 2008
2. Seleccione menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Application" y en Name escriba "Laboratorio4\_2".
4. Diseñe la siguiente GUI del formulario .

5. Coloca el código fuente que le corresponde a cada botón

Public Class Form1

'DECLARAR COLECCION DICTIONARY

Dim MisAlumnos = New Dictionary(Of String, String)

Private Sub btnProcesar\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnProcesar.Click

Try

'Aqui comienza el control de errores

Dim dividendo As Int16 = Convert.ToInt16(txtNumero1.Text)

'Si hemos introducido cero como número2

Dim divisor As Int16 = Convert.ToInt16(txtNumero2.Text)

Dim división As Int16

división = dividendo / divisor

MessageBox.Show("División:" & división.ToString)

Catch ex As OverflowException

'Si se produce un error por división entre 0, se generará una excepción

```
'que se captura en este bloque de código
MessageBox.Show("Mensaje error:División por cero")
Catch ex As Exception
'Si se produce un error e otro tipo, se generará una excepción
'que se captura en este bloque de código
MessageBox.Show(ex.Message)
MessageBox.Show("Origen error:" & ex.Source)
Finally
'CODIGO QUE SIEMPRE SE EJECUTA
End Try
End Sub

Private Sub btnIngresar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnIngresar.Click
Try
'SI AGREGAMOS UNA CLAVE QUE EXISTA
'SE PRODUCE UN ERROR
MisAlumnos.Add(txtCodigo.Text, txtAlumno.Text)
lstAlumnos.Items.Add(txtCodigo.Text & txtAlumno.Text)
txtAlumno.Clear()
txtCodigo.Clear()
txtCodigo.Focus()
Catch ex As Exception
'SE CONTROLA ERROR INDICANDO EL MENSAJE
MessageBox.Show("La clave es duplicada")
End Try

End Sub

Private Sub btnBuscar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnBuscar.Click
Try
'SI BUSCAMOS UNA CLAVE QUE NO EXISTE
'SE PRODUCE UN ERROR
Dim clave = InputBox("Ingrese clave de alumno")
MessageBox.Show("Alumno:" & MisAlumnos(clave))
Catch ex As Exception
'SE CONTROLA ERROR INDICANDO EL MENSAJE
MessageBox.Show("Clave no existe")
End Try

End Sub
End Class
```

Las excepciones se manejan con el uso de la palabra reservada **Catch**.

La manera en la que .NET utiliza **Catch** es similar al uso de las sentencias condicionales **If**. Si dentro de un conjunto anidado de sentencias **Catch** se maneja una excepción, el resto de instrucciones **Catch** no se ejecutan.

Por lo tanto, el código general del manejo de excepciones en un pequeño ejemplo de C# sería el siCuando se produce una excepción, se preguntará por el primer **Catch**, y

si se cumple, ya no se mirará otro **Catch**. Lo que se debe hacer, tal y como se indica en los ejemplos, es indicar las excepciones de manera que primero se gestione la excepción más concreta y precisa, y finalmente la excepción más general.

La particularidad viene cuando modificamos en el entorno de desarrollo de Visual Studio 2005 la gestión de las excepciones, posicionando el manejo de excepciones general, antes o por delante del manejo de la excepción concreta.

La parte de código escrito en C# correspondiente a la modificación en la gestión de excepciones quedaría de la siguiente manera:

```
...
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
catch (DivideByZeroException)
{
    MessageBox.Show("División por cero");
}
...
```

El mismo código equivalente pero en VB 2005, sería el siguiente:

```
...
Catch ex As Exception
    MessageBox.Show(ex.Message)
Catch ex As OverflowException
    MessageBox.Show("División por cero")
End Try
...
```

Ahora bien, si utilizamos la modificación de código indicada anteriormente y acudimos al entorno de desarrollo, Visual Studio 2005 nos indicará un aviso para VB 2005 y un error para C#.

```
Catch ex As Exception
    MessageBox.Show(ex.Message)
Catch otra As DivideByZeroException
    Nunca se alcanzó el bloque 'Catch' porque 'System.DivideByZeroException' hereda de 'System.Exception'.
End Try
End Sub
```

Class

---

```
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
catch (DivideByZeroException)
{
    Una cláusula catch previa ya detecta todas las excepciones de este tipo o de tipo superior ("System.Exception")
    MessageBox.Show("División por cero");
}
```

La consecuencia es clara, mientras que VB 2005 nos permite ejecutar el código, C# no.

Esto nos lleva a una segunda consecuencia, y es que en el caso de VB 2005, el código se ejecuta (el aviso no nos lo impide mientras que el error sí), y por lo tanto, la gestión de excepciones no se realiza de forma satisfactoria, así que si trabajáis a menudo con excepciones... cuidado con el uso de las mismas.

## Autoevaluación

1. ¿En que caso se requiere usar una colección Genérica?

---

---

---

2. ¿Qué ventajas tiene usar una colección Genérica de una que no lo es?

---

---

---

3. ¿Qué controla el bloque Try y el bloque Catch respectivamente?

---

---

---



## Para Recordar

- El espacio de nombres **System.Collections.Generic** contiene interfaces y clases que definen colecciones genéricas, permitiendo que los usuarios creen colecciones con establecimiento inflexible de tipos para proporcionar una mayor seguridad de tipos y un rendimiento mejor que los de las colecciones no genéricas con establecimiento inflexible de tipos.
- **Dictionary<Of <(TKey, TValue)>>** puede admitir varios sistemas de lectura a la vez, siempre y cuando no se modifique la colección
- La interfaz **IDictionary<Of <(TKey, TValue)>>** es la interfaz base para las colecciones genéricas de pares de clave y valor.
- Cada elemento es un par de clave y valor almacenado en un objeto [KeyValuePair<Of <\(TKey, TValue\)>>](#).
- La interfaz genérica **IList<Of <(T)>>** es una descendiente de la interfaz genérica **ICollection<Of <(T)>>** y es la interfaz base de todas las listas genéricas.

### Fuente

Texto adaptado de las páginas web:

<http://msdn.microsoft.com/es-es/library/system.collections.generic.aspx>

[http://geeks.ms/blogs/jorge/pages/Visual-Studio-2005-\\_3A003A00\\_-Diferencias-en-tiempo-de-dise\\_F100\\_o-de-C\\_2300\\_-y-VB-2005-en-el-uso-de-excepciones.aspx](http://geeks.ms/blogs/jorge/pages/Visual-Studio-2005-_3A003A00_-Diferencias-en-tiempo-de-dise_F100_o-de-C_2300_-y-VB-2005-en-el-uso-de-excepciones.aspx)



<b>UNIDAD DE APRENDIZAJE</b>
<b>3</b>
<b>SEMANA</b>
<b>5</b>

## **Administración de Servicios Windows**

---

### **LOGRO DE LA UNIDAD DE APRENDIZAJE**

Al término de la unidad, los alumnos crean aplicaciones personalizadas para la Administración del Sistema Operativo utilizando algoritmos y librerías del Framework .NET 2.0 para el manejo de archivos, carpetas, servicios, procesos y diagnósticos.

### **TEMARIO**

- Implementando un servicio de Windows desde vb .NET
- Configurar las propiedades de un servicio, manejo del Instalador de servicio
- Realizar consultas sobre los servicios de Windows

### **ACTIVIDADES PROPUESTAS**

- Los alumnos desarrollan un servicio de Windows, lo instalan y lo ejecutan.
- Los alumnos desarrollan aplicaciones que consulten los servicios de Windows.

## 1. Servicios Windows

Los Servicios de Windows son aplicaciones que funcionan sin interactuar directamente con el usuario y por regla general se inician junto con el sistema, sin que ningún usuario tenga que iniciarlo.

### 1. ServiceController (Clase)

Representa a un servicio de Windows y permite conectarse a un servicio en ejecución o detenido, manipularlo u obtener información acerca del mismo.

Se puede utilizar la clase **ServiceController** para conectarse y controlar el comportamiento de los servicios existentes. Al crear una instancia de la clase **ServiceController**, se pueden establecer las propiedades correspondientes de forma que interactúen con un servicio de Windows específico. De esta manera, se puede utilizar la clase para iniciar, detener o manipular el servicio.

Probablemente, se utilizará el componente **ServiceController** para tareas administrativas. Por ejemplo, se puede crear una aplicación para Windows o una aplicación Web que envíe los comandos personalizados a un servicio a través de la instancia de **ServiceController**. Esto podría resultar de utilidad, porque el complemento Microsoft Management Console del Administrador de control de servicios (SCM, Service Control Manager) no admite comandos personalizados.

Después de crear una instancia de **ServiceController**, se deben establecer dos propiedades en ella para identificar el servicio con el que interactúa: el nombre del equipo y el nombre del servicio que se desea controlar.

### Definir el comportamiento del servicio

En la clase de servicio, reemplace las funciones de clase base que determinan lo que ocurre cuando se cambia el estado del servicio en el Administrador de control de servicios. La clase **ServiceBase** expone los siguientes métodos, que puede reemplazar para agregar comportamientos personalizados.

Método	Reemplázelo para
OnStart	Indicar qué acciones deben ejecutarse cuando empieza a funcionar el servicio. Para que el servicio ejecute un trabajo útil, deberá escribir código en este procedimiento.
OnPause	Indicar qué debe ocurrir cuando se pausa el servicio.
OnStop	Indicar qué debe ocurrir cuando se detenga la ejecución del servicio.
OnContinue	Indicar qué debe ocurrir cuando el servicio reanuda su funcionamiento normal tras una pausa.
OnShutdown	Indicar qué debe ocurrir justo antes de que el sistema se cierre, en caso de que se esté ejecutando el servicio en ese momento.
OnCustomCommand	Indicar qué debe ocurrir cuando el servicio reciba un

	comando personalizado. Para obtener más información acerca de los comandos personalizados, consulte MSDN Online.
<a href="#">OnPowerEvent</a>	Indicar cómo debe responder el servicio cuando se reciba un evento de administración de energía, por ejemplo, una batería agotada o una operación suspendida.

[Imports](#) System

[Imports](#) System.ServiceProcess

[Imports](#) System.Diagnostics

[Imports](#) System.Threading

[Class](#) Program

[Public Enum](#) SimpleServiceCustomCommands

StopWorker = 128

RestartWorker

CheckWorker

[End Enum](#) 'SimpleServiceCustomCommands

[Shared Sub](#) Main([ByVal](#) args() [As String](#))

[Dim](#) scServices() [As](#) ServiceController

scServices = ServiceController.GetServices()

[Dim](#) scTemp [As](#) ServiceController

[For Each](#) scTemp [In](#) scServices

[If](#) scTemp.ServiceName = "Simple Service" [Then](#)

' Display properties for the Simple Service sample

' from the ServiceBase example

[Dim](#) sc [As New](#) ServiceController("Simple Service")

Console.WriteLine("Status = " + sc.Status.ToString())

Console.WriteLine("Can Pause and Continue = " + \_  
sc.CanPauseAndContinue.ToString())

Console.WriteLine("Can ShutDown = " + sc.CanShutdown.ToString())

Console.WriteLine("Can Stop = " + sc.CanStop.ToString())

[If](#) sc.Status = ServiceControllerStatus.Stopped [Then](#)

sc.Start()

[While](#) sc.Status = ServiceControllerStatus.Stopped  
Thread.Sleep(1000)

```

        sc.Refresh()
    End While
End If
' Issue custom commands to the service
' enum SimpleServiceCustomCommands
' { StopWorker = 128, RestartWorker, CheckWorker };
sc.ExecuteCommand(Fix(SimpleServiceCustomCommands.StopWorker))
sc.ExecuteCommand(Fix(SimpleServiceCustomCommands.RestartWorker))
sc.Pause()
While sc.Status <> ServiceControllerStatus.Paused
    Thread.Sleep(1000)
    sc.Refresh()
End While
Console.WriteLine("Status = " + sc.Status.ToString())
sc.Continue()
While sc.Status = ServiceControllerStatus.Paused
    Thread.Sleep(1000)
    sc.Refresh()
End While
Console.WriteLine("Status = " + sc.Status.ToString())
sc.Stop()
While sc.Status <> ServiceControllerStatus.Stopped
    Thread.Sleep(1000)
    sc.Refresh ()
End While
Console.WriteLine("Status = " + sc.Status.ToString())
Dim argArray() As String = _
{"ServiceController arg1", "ServiceController arg2"}
sc.Start(argArray)
While sc.Status = ServiceControllerStatus.Stopped
    Thread.Sleep(1000)
    sc.Refresh()
End While
Console.WriteLine ("Status = " + sc.Status.ToString())
' Display the event log entries for the custom commands
' and the start arguments.
Dim el As New EventLog("Application")
Dim elec As EventLogEntryCollection = el.Entries
Dim ele As EventLogEntry

```

```
For Each ele in elec
    If ele.Source.IndexOf("SimpleService.OnCustomCommand") >= 0 Or
       ele.Source.IndexOf("SimpleService.Arguments") >= 0 Then
        Console.WriteLine (ele.Message)
    End If
Next ele
End If
Next scTemp
End Sub 'Main
End Class 'Program

'This sample displays the following output if the Simple Service
'sample is running:
'Status = Running
'Can Pause and Continue = True
'Can ShutDown = True
'Can Stop = True
'Status = Paused
'Status = Running
'Status = Stopped
'Status = Running
'4:14:49 PM - Custom command received: 128
'4:14:49 PM - Custom command received: 129
'ServiceController arg1
'ServiceController arg2
```

## 2. ServiceBase (Clase)

Proporciona una clase base para un servicio que existirá como parte de una aplicación de servicio. Deberá derivarse de **ServiceBase** cuando se cree una nueva clase de servicio.

Derive de **ServiceBase** al definir la clase de servicio de una aplicación de servicio. Cualquier servicio útil reemplaza los métodos **OnStart** y **OnStop**. Para obtener funcionalidad adicional, puede reemplazar **OnPause** y **OnContinue** por un comportamiento específico en respuesta a cambios en el estado del servicio.

Un servicio es un ejecutable de ejecución larga que no admite una interfaz de usuario y que puede no ejecutarse en la cuenta de usuario que ha iniciado la sesión. El servicio puede ejecutarse sin que ningún usuario haya iniciado una sesión en el equipo.

De forma predeterminada, los servicios se ejecutan en la cuenta de sistema, que no es igual que la cuenta de administrador. No puede cambiar los derechos de la cuenta de sistema. De modo alternativo, puede utilizar **ServiceProcessInstaller** para especificar una cuenta de usuario en la que se ejecutará el servicio.

Un ejecutable puede contener más de un servicio, pero tiene que contener un componente [ServiceInstaller](#) independiente para cada uno de los servicios. La instancia de [ServiceInstaller](#) registra el servicio en el sistema. Asimismo, el instalador asocia cada servicio a un registro de eventos que puede utilizar para registrar comandos de servicio. La función [main\(\)](#) del ejecutable define qué servicios deben ejecutarse. El directorio de trabajo actual del servicio es el directorio del sistema, no el directorio en el que está situado el ejecutable.

Cuando se inicia un servicio, el sistema busca el ejecutable y ejecuta el método [OnStart](#) de ese servicio, contenido dentro del ejecutable. Sin embargo, ejecutar el servicio no equivale a ejecutar el ejecutable. El ejecutable solamente carga el servicio. Se obtiene acceso al servicio (por ejemplo, lo inicia y lo detiene) mediante el Administrador de control de servicios.

El ejecutable llama al constructor de la clase derivada de **ServiceBase** la primera vez que se llama a Start para el servicio. Se llama al método de control de comandos [OnStart](#) inmediatamente después de que se ejecute el constructor. El constructor no se vuelve a ejecutar después de la primera vez que se ha cargado el servicio, por lo que es necesario separar el procesamiento realizado por el constructor del realizado por [OnStart](#). Cualquier recurso que [OnStop](#) pueda liberar tiene que crearse en [OnStart](#). La creación de recursos en el constructor impide que estos se creen adecuadamente si se inicia de nuevo el servicio una vez que [OnStop](#) ha liberado los recursos.

El Administrador de control de servicios (SCM, Service Control Manager) permite la interacción con el servicio. Puede utilizar el SCM para pasar comandos Iniciar, Detener, Pausar, Continuar o comandos personalizados al servicio. El SCM utiliza los valores de las propiedades [CanStop](#) y [CanPauseAndContinue](#) para determinar si el servicio acepta los comandos Detener, Pausar o Continuar. Detener, Pausar y Continuar sólo estarán habilitados en los menús contextuales del SCM si la propiedad [CanStop](#) o [CanPauseAndContinue](#) correspondiente es true en la clase de servicio. Si está habilitado, el comando se pasa al servicio y se llama a [OnStop](#), [OnPause](#) o [OnContinue](#). Si la propiedad [CanStop](#), [CanShutdown](#) o [CanPauseAndContinue](#) es false, no se procesará el método de control de comandos correspondiente (como [OnStop](#)), aunque se haya implementado el método.

Puede utilizar la clase [ServiceController](#) para realizar mediante programación lo que el SCM realiza mediante una interfaz de usuario. Es posible automatizar las tareas disponibles en la consola. Si la propiedad [CanStop](#), [CanShutdown](#) o [CanPauseAndContinue](#) es true, pero no implementó el método de control de comandos correspondiente (como [OnStop](#)), el sistema produce una excepción y omite el comando.

No tiene que implementar el método [OnStart](#), [OnStop](#) ni ningún otro método en **ServiceBase**. Sin embargo, el comportamiento del servicio se describe en [OnStart](#), por lo que, como mínimo, tendrá que reemplazarse este miembro. La función [main\(\)](#) del ejecutable registra el servicio en el ejecutable con el Administrador de control de servicios; para ello, llama al método [Run](#). La propiedad [ServiceName](#) del objeto **ServiceBase** pasada al método [Run](#) debe coincidir con la propiedad [ServiceName](#) del instalador de ese servicio.

Puede utilizar InstallUtil.exe para instalar servicios en el sistema.



## LABORATORIO 5\_1

Cree una aplicación Windows que permita ingresar el estado de un servicio y luego a través del botón Listar se liste todos los servicios que tienen el estado seleccionado

1. Ingrese a Visual Studio 2008
2. Seleccione Menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Application" y en Name escriba "Laboratorio5\_1"
4. Desarrollar la siguiente GUI para la consulta de servicios por estado

Nombre Servicio	Descripci...	Estado	

5. Colocar el siguiente código desde la vista código del formulario
6. En la parte superior del código fuente importar el namespace para el uso de servicios.

**Imports** System.Serviceprocess

**Public Class** frmServicios

```
Private Sub btnListar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnListar.Click
    'Crear un arreglo de tipo ServiceController
    Dim arrServicios() As ServiceController
    ' ServiceController.GetServices()-->obtiene todos los ervicios locales
    arrServicios = ServiceController.GetServices()
    'mostrar cantidad de elementos del arreglo
    MessageBox.Show("Elementos del arreglo:" & arrServicios.Length)
    'limpiar el listview
    lvServicios.Items.Clear()
    'crear un objeto que representa a un elemento del listview
    Dim ele As ListViewItem
```

```

'recorrer arreglo de servicios
For i As Integer = 0 To arrServicios.Length - 1
    'agregar primera columna al Listview
    ele = lvServicios.Items.Add(arrServicios(i).ServiceName)
    'agregar segunda columna al listview
    ele.SubItems.Add(arrServicios(i).DisplayName)
    'agregar tercera columna del listview
    ele.SubItems.Add(arrServicios(i).Status.ToString)
    'agregar cuarta columna
    ele.SubItems.Add(arrServicios(i).ServiceType.ToString)

Next
End Sub

Private Sub btnVerEstado_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnVerEstado.Click
    'verificar que se haya seleccionado elemento en listview
    If lvServicios.SelectedItems.Count > 0 Then
        'mostrar nombre del servicio seleccionado
        Dim strServicio As String
        strServicio = lvServicios.SelectedItems(0).Text
        MessageBox.Show(strServicio)
        'Crear un objeto de tipo ServiceController
        Dim miServicio As New ServiceController(strServicio)
        'consultar estado del servicio elegido
        If miServicio.Status = ServiceControllerStatus.Running Then
            MessageBox.Show("Iniciado")
        ElseIf miServicio.Status = ServiceControllerStatus.Stopped Then
            MessageBox.Show("Detenido")
        End If
    Else
        MessageBox.Show("Debe seleccionar un registro")
    End If
End Sub
End Class

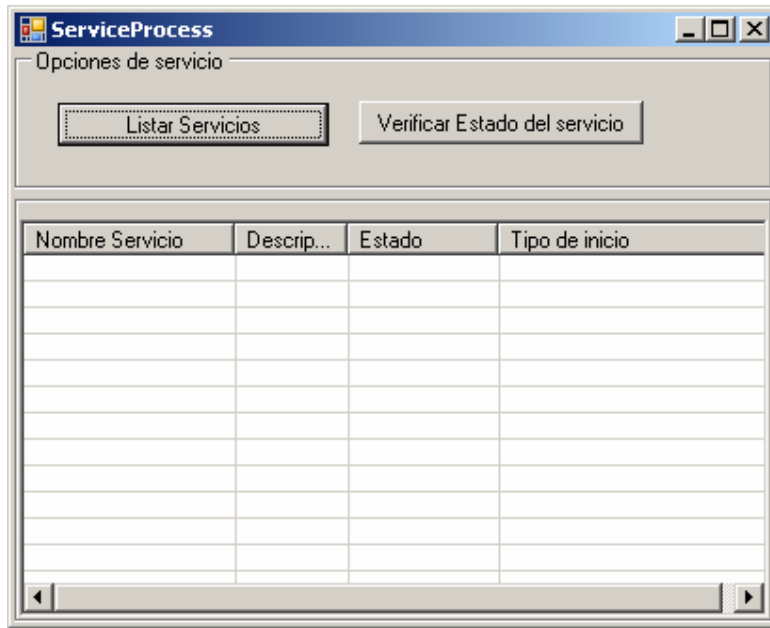
```

7. Ejecutar el formulario y comprobar consulta desarrollada.

## LABORATORIO 5.2

Cree una aplicación Windows que permita consultar el estado de un servicio , el cual será elegido desde un listview.

1. Ingrese a Visual Studio 2008
2. Seleccione Menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Application" y en Name escriba "Laboratorio5\_2"
4. Diseñar la siguiente GUI para la consulta de servicios



5. Colocar este código en la vista diseño del formulario
6. Importar el namespace de Servicios en la parte superior del código fuente.

```
Imports System.ServiceProcess
Public Class frmServiciosporEstado
    'crear método que retorne un arreglo de tipo ServiceController
    Function listarServicios _
        (ByVal tipo As ServiceControllerStatus) As ServiceController()
        'obtener cantidad de servicios de la pc
        'Dim cantidad As Int32 = ServiceController.GetServices.Length
        'crear un arreglo de solo una casilla
        Dim arrServicios(0) As ServiceController
        Dim i As Int16 = 0
        For Each serv As ServiceController In ServiceController.GetServices
            'verificar estado del servicio
            If serv.Status = tipo Then
                arrServicios(i) = New ServiceController
                arrServicios(i) = serv
                'redimensionar arreglo
                i += 1
                'aumentar una casilla al arreglo
                ReDim Preserve arrServicios(i)
            End If
        Next
        Return arrServicios
    End Function
    'botón listar
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
        System.EventArgs) Handles Button1.Click
        'arreglo
        Dim misServicios() As ServiceController
        'verificar que estado fue elegido
        If RadioButton1.Checked = True Then
            misServicios = listarServicios(ServiceControllerStatus.Running)
        End If
    End Sub
End Class
```

```

Elseif RadioButton2.Checked = True Then
    misServicios = listarServicios(ServiceControllerStatus.Stopped)
Else
    misServicios = listarServicios(ServiceControllerStatus.Paused)
End If
'mostrar cantidad de elementos del arreglo
Label1.Text = "Q de Servicios por estado:"
If misServicios.Length = 1 Then
    If misServicios(0) Is Nothing Then
        Label2.Text = 0
    End If
Else
    Label2.Text = misServicios.Length.ToString
End If

'mostrar elementos del arreglo
lvServicios.Items.Clear()
Dim ele As ListViewItem
For i As Integer = 0 To misServicios.Length - 1
    ele = lvServicios.Items.Add(misServicios(i).ServiceName)
    ele.SubItems.Add(misServicios(i).DisplayName)
    ele.SubItems.Add(misServicios(i).Status.ToString)
Next
End Sub
End Class

```

## LABORATORIO 5.3

Cree una aplicación Windows que permita consultar el estado de un servicio , el cual será elegido desde un listview.

1. Ingrese a Visual Studio 2008
2. Seleccione Menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Service" y en Name escriba "Laboratorio5\_3"
4. Nos aparecerá directamente la clase **Service1.vb** que también podemos renombrar a algo más intuitivo como **ServicioAlerta.vb**.
5. En la vista diseño de la clase que se nos ha creado hacemos clic con el botón derecho del ratón y seleccionamos la opción **Agregar instalador**. Con esto se nos creará la clase **ProjectInstaller.vb** que contiene dos controles: **ServiceInstaller1** y **ServiceProcessInstaller1**.
6. En el control **ServiceInstaller** modificamos las propiedades **DisplayName** y **ServiceName** para dejarlas ambas en **ServicioAlertas**. Modificamos también la propiedad **StartType** para dejarla en **Automatic**. Esto último sirve para que después cuando instalemos el servicio, éste quede puesto para que arranque automáticamente al iniciarse el sistema operativo.

DisplayName: ServicioAlertas  
 ServiceName: ServicioAlertas  
 StartType: Automatic

7. En el control **ServiceProcessInstaller** modificamos la propiedad **Account** para dejarla en **LocalSystem**. Esto indica qué tipo de cuenta se utilizará para ejecutar el servicio. Podemos dejarla en **User**, pero después al instalar el

servicio pedirá cuenta de usuario y contraseña, así que se recomienda cambiarla a **LocalSystem**.

8. Accedemos al código de la clase **ServicioAlerta.vb** y observamos que por defecto ya se nos han generado dos métodos: **OnStart** y **OnStop**.
9. Se codifica en el método **OnStart** y se usará un temporizador para que cada 5 segundos me escriba en un log la hora actual y en el método **OnStop** se colocará un mensaje en el que se detuvo el servicio.

Public Class **ServicioAlerta**

Private DBTimer As System.Timers.Timer

' Iniciar service.

'-----

Protected Overrides Sub OnStart(ByVal args() As String)

    'Creando timer para cada 5 segundos

    DBTimer = New System.Timers.Timer(5000)

    DBTimer.Enabled = True

    AddHandler DBTimer.Elapsed, AddressOf Me.ShowAlert

End Sub

'-----

' Detener service.

'-----

Protected Overrides Sub OnStop()

    Dim LogFile As New System.IO.StreamWriter("C:\log.txt", True)

    LogFile.WriteLine("Servicio Detenido - " & Date.Now)

    LogFile.Close()

End Sub

Private Sub ShowAlert(ByVal source As Object, \_

ByVal e As System.Timers.ElapsedEventArgs)

    DBTimer.Enabled = False

    Dim LogFile As New System.IO.StreamWriter("C:\log.txt", True)

    LogFile.WriteLine("Alerta .NET - " & Date.Now)

    LogFile.Close()

    DBTimer.Enabled = True

End Sub

End Class

10. El archivo **.exe** (que encontramos en la carpeta **bin/Release** de nuestro proyecto **ServicioAlertas**) lo copiaremos en una ruta más corta (**C:\**). Y los dos archivos **.bat** que necesitamos son los siguientes:

**Instalador.bat**

```
C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\InstallUtil
"C:\ServicioAlertas.exe"
```

pause

**Desinstalador.bat**

```
C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\InstallUtil /U
"C:\ServicioAlertas.exe"
```

pause

11. ejecutamos el archivo **Instalador.bat** y se nos abrirá una ventana de línea de comando con el proceso de la instalación terminando exitosamente. Ahora nos vamos a **Panel de control - Herramientas administrativas - Servicios** y encontraremos nuestro servicio **ServicioAlertas**. Lo iniciamos y observaremos al cabo de pocos segundos que en la ruta **C:\** se nos ha creado un archivo **log.txt**. Si dejamos el servicio un rato funcionando y después lo detenemos veremos que este archivo ha quedado más o menos así:

Alerta .NET - 13/12/2008 11:12:18

Alerta .NET - 13/12/2008 11:12:18

Alerta .NET - 13/12/2008 11:12:18

Alerta .NET - 13/12/2008 11:12:18

Alerta .NET - 13/12/2008 11:12:18

12. Si en algún momento queremos desinstalar el servicio (si es tan poco útil como **ServicioAlertas** seguro que querremos) solo debemos ejecutar **Desinstalador.bat** y el servicio quedará desinstalado (tendremos que refrescar la lista de servicios para ver que efectivamente así es).

## Autoevaluación

1. ¿Qué es un servicio windows?

---

---

---

2. ¿Qué tipo de consultas se pueden realizar con la clase ServiceController ?

---

---

---

3. ¿Qué clase se utiliza para crear un nuevo servicio windows con Visual Basic .NET?

---

---

---

## Para Recordar

- Al crear una instancia de la clase **ServiceController**, se pueden establecer las propiedades correspondientes de forma que interactúen con un servicio de Windows específico. De esta manera, se puede utilizar la clase para iniciar, detener o manipular el servicio.
- Se deberá derivarse de **ServiceBase** cuando se cree una nueva clase de servicio. Cualquier servicio útil reemplaza los métodos OnStart y OnStop. Para obtener funcionalidad adicional, puede reemplazar OnPause y OnContinue por un comportamiento específico en respuesta a cambios en el estado del servicio.
- Es fundamental que la propiedad ServiceName sea idéntica a la propiedad ServiceBase.ServiceName de la clase que se derivó de ServiceBase. Normalmente, el valor de la propiedad ServiceBase.ServiceName del servicio está establecido en la función Main() del ejecutable de la aplicación de servicio.

**Fuente:****Texto adaptado de las páginas web:**

<http://msdn.microsoft.com/es-es/library/system.serviceprocess.aspx>

<http://www.albertmata.net/>



<b>UNIDAD DE APRENDIZAJE</b>
<b>3</b>
<b>SEMANA</b>
<b>6</b>

## **Construir aplicaciones Windows que utilicen Threads y Configuraciones**

---

### **LOGRO DE LA UNIDAD DE APRENDIZAJE**

Al término de la unidad, los alumnos crean aplicaciones personalizadas para la Administración del Sistema Operativo utilizando algoritmos y librerías del Framework .NET 2.0 para el manejo de archivos, carpetas, servicios, procesos y diagnósticos.

### **TEMARIO**

- Definición de un Thread, propiedades y métodos
- Componente BackgroundWorker
- Manejo de Temporizadores
- Manejo de funciones de cadenas

### **ACTIVIDADES PROPUESTAS**

- Los alumnos crean aplicaciones Windows que utilizan temporizadores
- Los alumnos crean aplicaciones que utilizan BackgroundWorker
- Los alumnos crean aplicaciones que utilizan Threads para ejecutar procesos en paralelos los cuales utilizan funciones de cadenas.

## 1. Thread

Un Thread se define como un camino posible de ejecución a lo largo de un proceso. Cada proceso (programa) tiene uno o mas threads de ejecución. Cuando se inicia una aplicación se crea el proceso de la misma y consigo se ejecuta el thread principal a partir del entry point del mismo.

La creación y gestión de threads en .Net se halla enmascarada bajo el namespace **Threading**. En particular para crear un nuevo Thread en nuestra aplicación el framework de .Net nos provee una clase que gestiona todos los pequeños detalles del mismo, la clase **Threading.Thread**.

### 1.1 Apartments States

Cada thread de .Net tiene un apartment state, un Apartment es un conjunto de reglas compartidas por un determinado grupo de objetos. El concepto de Apartment State viene de la transición de los primeros windows que no soportaban multithreading y define el nivel de seguridad del thread (es decir que consideraciones tendrá el sistema para acceder a los datos de los objetos que viven en dicho thread), a saber hay tres tipos de Apartment States:

- **MTA Multi Thread Apartment:** Los Objetos se diseñan para correr en ambientes completamente asincronicos (las variables del objeto pueden cambiar en cualquier momento de la ejecución del programa por lo que debe sincronizarse el acceso a las mismas)
- **STA Single Thread Apartment:** Estos objetos están diseñados para ejecutarse en threads unicos por lo que el acceso a sus datos debe de ser secuencial. En cada STA existirá uno y solo un Thread. Las llamadas a los metodos de los objetos de un STA son sincronizados por una cola de Mensajes
- **NA Neutral Apartments:** Tiene la Particularidad de que los objetos que se ejecutan en este apartment pueden ser llamados **Directamente** desde cualquier otro apartment. Este apartment a la diferencia de MTA y STA no tienen ningún thread sino que los objetos son ejecutados en los threads desde los que son llamados

#### **Nota:**

El concepto de tareas concurrentes en idioma inglés es denominado **Threading** y a cada una de las tareas se las denomina simplemente **Thread**. En español las traducciones que he visto son **hilo**, **hebray subproceso**.

## ¿Cómo determinar si un puerto está abierto?

Para detectar si un puerto está abierto, basta con intentar conectar un Socket a dicho puerto.

Imports System.Net.Sockets

```
Public Function IsPortOpen(ByVal Host As String, ByVal Port As Integer) As Boolean
    Dim m_sck As New Socket(AddressFamily.InterNetwork, SocketType.Stream,
    ProtocolType.Tcp)
    Try
        m_sck.Connect(Host, Port)
        Return True
    Catch ex As SocketException
        'Código para manejar error del socket (cerrado, conexión rechazada)
    Catch ex As Exception
        'Código para manejar otra excepción
    End Try
    Return False
End Function
```

Como se puede apreciar, solamente se crea el objeto de tipo Socket y se intenta conectarlo mediante la función sincrónica Connect. Esta función bloqueará la ejecución hasta obtener una respuesta (favorable o no).

La llamada a la función *Connect* está dentro de un bloque *Try-Catch* puesto que esta función generará excepciones dependiendo del motivo por el cual no pudo abrir el socket hacia el puerto indicado. En este caso sólo nos interesa saber si se pudo o no.

## 1.2 Otras opciones (BackgroundWorker y ThreadPool)

La clase Thread (System.Threading) brinda la mayor flexibilidad en manejo de hilos, pero, cuando se trata de arreglos o colas, la codificación se torna complicada y confusa. Existen algunas clases que pueden ser útiles en alguno casos simplificando un poco el manejo de los hilos, como por ejemplo la clase **BackgroundWorker** (System.ComponentModel) y la clase **ThreadPool** (System.Threading).

La clase **BackgroundWorker** ofrece una práctica solución para los casos de operaciones que se demoren mucho tiempo y se necesite una interfaz rápida. Permite ejecutar en segundo plano una operación, y permite la generación sencilla de eventos para crear informes del progreso y para que señalen su finalización. La clase está pensada para la ejecución de una operacion larga (más que para muchas operaciones pequeñas).

La clase **ThreadPool** permite crear un grupo de subprocesos (ó *una cola de hilos* ó *una lista elementos de trabajo*) para su ejecución en forma concurrente. Para encolar un subproceso se llama al método *QueueUserWorkItem* que toma como parámetro una referencia al método (o delegado) al que llamará el subproceso.

El inconveniente del ThreadPool es que no permite cancelar un subproceso después de haber sido encolado. Stephen Toub de Microsoft escribió la clase

*AbortableThreadPool* que brinda las mismas características de un *ThreadPool* pero además permite la interrupción de subprocesos. Pero está codificado en C#.

Para los que no sean amigos de C# también adjunto la versión en VB.NET de esta clase (la clase *AbortableThreadPool\_VB*) que un día, aburrido, la traduje y comenté.

A continuación se muestra una forma de explotar la clase *TCPPort* con un *ThreadPool* normal (no abortable), en este caso no tendremos control sobre los hilos y **no** será necesaria la clase *PortScanner*.

Simplemente encolamos las tareas y éstas se ejecutan:

```
Imports System.Threading
```

```
Module Main_ThreadPool
```

```
    Private WithEvents TCP As New TCPPort
```

```
Sub main()
```

```
    Dim datos As TCPPort.HostPortData, i As Integer
```

```
    datos.Host = "localhost"
```

```
    For i = 1 To 100
```

```
        datos.Port = i
```

```
        'Encolar la tarea
```

```
        ThreadPool.QueueUserWorkItem(AddressOf TCP.PortOpen, datos)
```

```
    Next
```

```
    Console.ReadKey()
```

```
End Sub
```

```
    'Interceptar los eventos generados e informar el estatus
```

```
Private Sub TCP_PuertoProcesado(ByVal Host As String, ByVal Port As Integer, ByVal
```

```
bAbierto As Boolean, _
```

```
    ByVal sMensaje As String) Handles TCP.PuertoProcesado
```

```
    Console.WriteLine(Port & " : " & bAbierto.ToString)
```

```
End Sub
```

```
End Module
```

## 2. Hilos en VB.NET

La forma más fácil de visualizar la implementación de hilos es con un ejemplo. A continuación se muestra un ejemplo básico de la utilización de dos hilos (*Thread*) para ejecutar un método (*sub*) con parámetros:

```
Imports System.Threading
```

```
Module PruebaHilo
```

```
    'Sub que ejecutarán los hilos
```

```
Public Sub MiSub(ByVal Parametro As Object)
```

```
    Try
```

```
        Randomize()
```

```
    Do
```

```
        Dim iDormir As Integer = CInt(3000 * Rnd()) 'Valor random entre 0 y 3000
```

```
        Console.WriteLine("{0} sleep({1})", Parametro, iDormir)
```

```

        Thread.Sleep(iDormir) 'Me bloqueo entre 0 y 3 segundos
    Loop
Catch ex As ThreadAbortException
    Console.WriteLine("{0} Abortado", Parametro)
End Try
End Sub
'Sub principal
Sub Main()
    Dim hilo1 As New Thread(AddressOf MiSub) 'Crear el hilo 1
    Dim hilo2 As New Thread(AddressOf MiSub) 'Crear el hilo 2

    Console.WriteLine("Ejecutando hilos a abortar en 6 segundos...")
    hilo1.Start("hilo 1 ") 'Comenzar ejecución de hilo 1
    Thread.Sleep(500) 'Me Bloqueo 500 ms
    hilo2.Start(" hilo 2") 'Comenzar ejecución de hilo 2
    Thread.Sleep(6000) 'Me bloqueo 6 segundos

    hilo1.Abort() 'Abortar al hilo 1
    hilo2.Abort() 'Abortar al hilo 2

    'Esperar a que realmente mueran los hilos
    While hilo1.IsAlive Or hilo2.IsAlive
    End While

    Console.WriteLine("Abortados")
    Console.ReadKey()
End Sub
End Module

```

### 3. Hilos y Windows Forms

Si intentaste utilizar la clase *PortScanner* dentro de un form (forma o formulario) de Windows Forms, seguramente te encontraste con la excepción **InvalidOperationException** al intentar modificar el valor de un textbox o cualquier otro control dentro del evento disparado por los hilos.

Esto es porque *no se puede tener acceso a un control de windows forms desde un proceso distinto al que lo creó*. Al utilizar una aplicación de consola no se tiene este inconveniente, pero si es necesario utilizar Windows Forms, la forma de solucionar este problema es mediante un **delegado** y el método **Invoke** y se describe en el siguiente código:

```

Public Class Form1
    Private WithEvents PS As PortScanner 'Instancia de la clase PortScanner

    'Delegado para poder acceder a los miembros del form desde el sub ImprimirEstatus
    Delegate Sub ImprimirEstatusCB(ByVal Status As String)

    'Comenzar el proceso al hacer click en un botón
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
        System.EventArgs) _
        Handles Button1.Click
        PS = New PortScanner("localhost", 1, 100)
        PS.Start()
    End Sub
End Class

```

```

End Sub

'Manejar al evento generado por los hilos
Private Sub PS_PuertoScaneado(ByVal Port As Integer, ByVal bAbierto As Boolean,
-
                                ByVal sMensaje As String) Handles PS.PuertoScaneado
    ImprimirEstatus(Port & " " & bAbierto.ToString)
End Sub

'Imprimir el estatus del proceso en el caption del form y el ultimo estatus en un
TextBox
Private Sub ImprimirEstatus(ByVal Status As String)
    If Me.InvokeRequired Then
        'Si es necesario utilizar Invoke, llamo al delegado
        Me.Invoke(New ImprimirEstatusCB(AddressOf ImprimirEstatus), New Object()
{Status})
    Else
        'Aquí puedo modificar los controles de esta forma
        TextBox1.Text = Status
        Me.Text = TCPPort.m_CountThreads
    End If
End Sub
End Class

```

Lo interesante está en el sub *ImprimirEstatus* que al principio evalúa la propiedad *InvokeRequired* que indicará si es necesario utilizar el método *Invoke (then)* o si es posible modificar los controles de Windows Forms directamente (*else*). En el caso de ser necesario el *Invoke*, se pasará a éste (al *invoke*) un nuevo delegado del sub *ImprimirEstatusCB* y un arreglo de *Objects* conteniendo los parámetros que el sub requiera.

El delegado debe tener la misma *firma* que el sub al que invocará (y será invocado). En este caso el delegado es llamado *ImprimirEstatusCB* y el sub es llamado *ImprimirEstatus*.

### Nota

El proyecto **ScanWinForms** (versión C) es otra versión del escaneador pero en este caso se utiliza una forma de Windows Forms y se utiliza la clase *PortScanner* para llevar a cabo el proceso.

## Laboratorio 6.1

1. Inicie Microsoft Visual Studio .NET o Microsoft Visual Studio 2005.
2. Crear un nuevo proyecto aplicación de Windows de Visual Basic denominado ThreadWinApp.
3. Agregue un control **botón** al formulario. El botón se denominado **Button1** de forma predeterminada.
4. Agregar un componente de **ProgressBar** al formulario. La barra de progreso se denomina **ProgressBar1** de forma predeterminada.
5. Haga clic con el botón secundario del mouse en el formulario y, a continuación, haga clic en **Ver código**.
6. Agregar la siguiente instrucción al principio del archivo:

```
Imports System.Threading
```

7. Agregar el siguiente controlador de eventos Click para Button1 :

```
Private Sub Button1_Click( _  
    ByVal sender As System.Object, ByVal e As System.EventArgs) _  
    Handles Button1.Click  
  
    MessageBox.Show("This is the main thread")  
End Sub
```

8. Agregar la siguiente variable a la clase de Form1 :

```
Private trd As Thread
```

9. Agregar el método siguiente a la clase Form1 :

```
Private Sub ThreadTask()  
    Dim stp As Integer  
    Dim newval As Integer  
    Dim rnd As New Random()  
  
    Do  
        stp = ProgressBar1.Step * rnd.Next(-1, 2)  
        newval = ProgressBar1.Value + stp  
        If newval > ProgressBar1.Maximum Then  
            newval = ProgressBar1.Maximum  
        ElseIf newval < ProgressBar1.Minimum Then  
            newval = ProgressBar1.Minimum  
        End If  
  
        ProgressBar1.Value = newval  
  
        Thread.Sleep(100)  
    Loop  
End Sub
```

**Nota :** este es el código subyacente del subproceso. Este código está un infinito un bucle que aleatoriamente aumenta o disminuye el valor de **ProgressBar1** y, a continuación, espera 100 milisegundos antes de continúe.

10. Agregar el siguiente controlador de evento Load para Form1 . Este código crea un nuevo subproceso, hace que el subproceso de un subproceso en segundo plano y, a continuación, inicia el subproceso.

```
Private Sub Form1_Load( _  
    ByVal sender As System.Object, ByVal e As System.EventArgs) _  
    Handles MyBase.Load  
  
    trd = New Thread(AddressOf ThreadTask)  
    trd.IsBackground = True  
    trd.Start()  
  
End Sub
```



## Compruebe que funciona

1. Genere y ejecute la aplicación. Observe que el valor de la barra de progreso cambia aleatoriamente. Trata el nuevo subproceso en operación.
2. Para demostrar que el subproceso principal es independiente del subproceso que cambia el valor de **ProgressBar**, haga clic en el formulario el botón. Un cuadro de mensaje muestra el siguiente mensaje:

This is the main thread

Espere de entrada. Observe que el valor en la barra de progreso se mantiene cambiar.

### Laboratorio 6.2

1. Inicie Microsoft Visual Studio .NET o Microsoft Visual Studio 2005.
2. Crear un nuevo proyecto ClassLibrary de Visual Basic denominado ThreadBuscarPalabras.
3. Agregar el siguiente código:

```
Option Strict On

Imports System.IO
Imports System.Threading
'
Namespace POOI.Clases
    Public Class cBuscarPalabrasEnFicheros
        ' La colección de palabras a manipular
        Private lasPalabras As PooI.Clases.cPalabras
        ' Necesitamos que este objeto produzca eventos
        Private mProcesarFic() As
PooI.Clases.cProcesarFicheroThread
        Private WithEvents mProcesarFic2 As
PooI.Clases.cProcesarFicheroThread
        ' Array para cada uno de los threads
        Private mThreadFic() As Thread
        ' Variable para controlar el número de Threads
        creados
        Private nThread As Integer = 0
        ' Los eventos que producirá esta clase
        Public Event ProcesandoFichero(ByVal sMsg As
String)
        Public Event FicherosProcesados(ByVal sMsg As
String)
    End Class
End Namespace
```

```

        ' El constructor de la clase necesita la
        colección de palabras
        ' que se van a manipular por esta clase.
        Public Sub New(ByVal quePalabras As
PooI.Clases.cPalabras)

            MyBase.New()

            '

            lasPalabras = quePalabras
            mProcesarFic2 = New
PooI.Clases.cProcesarFicheroThread(lasPalabras)
        End Sub
    '

    Private Sub
mProcesarFic2_ProcesandoFichero(ByVal sMsg As String)

        ' Este evento se produce cuando se está
        procesando un fichero por uno de los threads.
        ' Desde aquí producimos nuestro evento a la
        aplicación

        RaiseEvent ProcesandoFichero(sMsg)
    End Sub
    '

    Private Sub ProcesarSubDir(ByVal sDir As String,
-
                                Optional ByVal sExt
As String = " *.*")
        '-----
        ' Este procedimiento será llamado de forma
        recursiva para procesar
        ' cada uno de los directorios.
        '-----
        '-----

        Dim tDir As Directory
        Dim tSubDirs() As Directory
        '

        ' Asigna a tSubDirs los subdirectorios
        incluidos en sDir

        tSubDirs =
tDir.GetDirectoriesInDirectory(sDir)

        ' Crear un thread para cada directorio a
        procesar

        nThread += 1

        ReDim Preserve mThreadFic(nThread)
        ReDim Preserve mProcesarFic(nThread)

```

```
        mProcesarFic(nThread - 1) = New
PooI.Clases.cProcesarFicheroThread(lasPalabras)
        mProcesarFic(nThread - 1).sDir = sDir
        mProcesarFic(nThread - 1).sExt = sExt
        ' Los procedimientos a usar no deben aceptar
parámetros
        mThreadFic(nThread - 1) = New Thread(New
ThreadStart(AddressOf mProcesarFic(nThread -
1).ProcesarDir))
        '
        ' Iniciar el thread
        mThreadFic(nThread - 1).Start()
        ' Llamada recursiva a este procedimiento
para procesar los subdirectorios
        ' de cada directorio
        For Each tDir In tSubDirs
            ' Procesar todos los directorios de cada
subdirectorio
            ProcesarSubDir(tDir.FullName, sExt)
        Next
    End Sub
    '
    Public Function Procesar(ByVal sDir As String, _
Optional ByVal sExt As
String = "*.*", _
Optional ByVal conSubDir
As Boolean = False _
) As String
        ' Procesar los directorios del path indicado en sDir,
        ' buscando ficheros con la extensión sExt,
        ' y si se deben procesar los subdirectorios.
        '-----
        '
        Dim totFiles As Integer = 0
        Dim s As String
        '
        ' Comprobar si se van a procesar los
subdirectorios
        If conSubDir Then
            ProcesarSubDir(sDir, sExt)
        Else
            nThread += 1
```

```

        ReDim Preserve mThreadFic(nThread)
        ReDim Preserve mProcesarFic(nThread)
        mProcesarFic(nThread - 1) = New
PooI.Clases.cProcesarFicheroThread(lasPalabras)
        mProcesarFic(nThread - 1).sDir = sDir
        mProcesarFic(nThread - 1).sExt = sExt
        mThreadFic(nThread - 1) = New Thread(New
ThreadStart(AddressOf mProcesarFic(nThread -
1).ProcesarDir))
        '
        mThreadFic(nThread - 1).Start()
    End If
    ' Aquí llegará incluso antes de terminar todos los
    threads
    'Debug.WriteLine("Fin de Procesar todos los ficheros")
    ' Comprobar si han terminado los threads
    Dim i, j As Integer
    Do
        j = 0
        For i = 0 To nThread - 1
            ' Comprobar si alguno de los Threads está "vivo"
            ' si es así, indicarlo para que continúe el bucle
            If mThreadFic(i).IsAlive() Then
                j = 1
            End If
        Next
        ' Esto es necesario, para que todo siga funcionando
        System.Windows.Forms.Application.DoEvents()
        Loop While j = 1
        'Debug.WriteLine("Han finalizado los threads")
        ' Ahora podemos asignar el número de ficheros procesados
        totFiles = mProcesarFic2.TotalFicheros
        If totFiles = 1 Then
            s = " Procesado 1 fichero."
        Else
            s = " Procesados " & CStr(totFiles) & "
ficheros."
        End If
        RaiseEvent FicherosProcesados(s)
        ' Asignamos a cero el valor de total ficheros
        ' por si se vuelve a usar, para que no siga acumulando.

```

```
mProcesarFic2.TotalFicheros = 0  
Return s  
End Function  
End Class  
End Namespace
```

**Comprobar la búsqueda de palabras desde una una aplicación Windows**

## Autoevaluación

1. ¿Qué es un Thread?

---

---

---

2. ¿Qué namespace es el que utilizamos para acceder a un temporizador?

---

---

---

3. ¿Qué proporciona la clase ThreadPool?

---

---

---

## Para Recordar

- Todas las aplicaciones se ejecutan en un Thread (o hilo de ejecución). Pero cada aplicación puede tener más de un Thread al mismo tiempo, es decir se pueden estar haciendo varias cosas a un mismo tiempo. En Visual Basic.Net, a diferencia de las versiones anteriores, se pueden crear Threads para que podamos realizar diferentes tareas a un mismo tiempo.
- Cuando un Thread se inicia, con Start, se continúa ejecutando el código que sigue y así podremos seguir usando nuestra aplicación de forma paralela al Thread que está en ejecución. Así mismo, se pueden crear y ejecutar varios Threads a un mismo tiempo y cada uno de ellos hará su trabajo de forma independiente.

Puede ser que en algunas ocasiones necesitemos saber si un Thread aún se está ejecutando, para ello se puede usar la propiedad **IsAlive** del objeto Thread:

```
If mThreadFic.IsAlive() Then
```

### Fuente

Texto Adaptado de las páginas web:

<http://www.elguille.info/NET/VB/threads.htm#usarShared>

<http://support.microsoft.com/kb/315577>





**UNIDAD DE  
APRENDIZAJE**

**3**

**SEMANA**

**9**

## **Caracteristics de Diagnosticos En .Net**

---

### **LOGRO DE LA UNIDAD DE APRENDIZAJE**

Al término de la unidad, los alumnos crean aplicaciones personalizadas para la Administración del Sistema Operativo utilizando algoritmos y librerías del Framework .NET 2.0 para el manejo de archivos, carpetas, servicios, procesos y diagnósticos.

### **TEMARIO**

- Manejo del Log de eventos.
- Procesos del sistema, y manejo de clases y métodos
- Información Administrativa (WMI)

### **ACTIVIDADES PROPUESTAS**

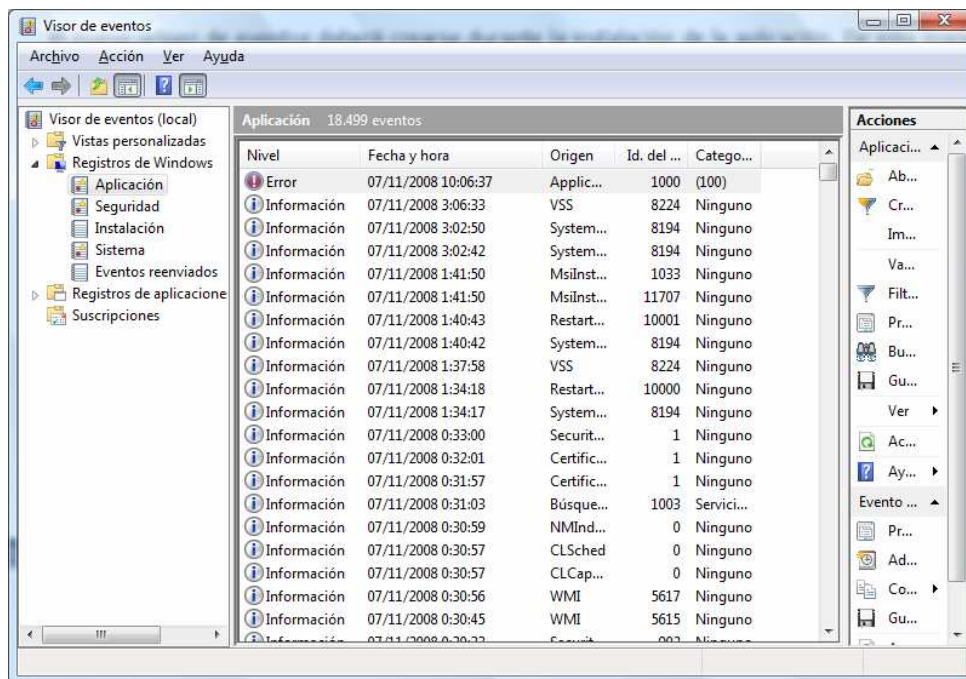
- Los alumnos se conectan al visor de eventos
- Los alumnos consultan las notificaciones de los logs de eventos
- Los alumnos consultan los procesos activos en la pc.
- Los alumnos realizan consultas sobre las herramientas del Sistema operativo Windows a través de WMI.

## Diagnosticos

### 1. EventLog : System.Diagnostics

Permite obtener acceso a registros de eventos de Windows o personalizarlos; estos registros graban información sobre eventos de software o hardware importantes. Mediante **EventLog**, se puede leer a partir de registros existentes, escribir entradas en registros, crear o eliminar orígenes de eventos, eliminar registros y responder a entradas de registros. También se pueden crear nuevos registros al crear un origen de eventos.

Para escribir eventos en un registro de eventos, deberán utilizarse [WriteEvent](#) y [WriteEntry](#). Para escribir eventos, es necesario especificar un origen de eventos; asimismo, antes de escribir la primera entrada con el origen, es necesario crearlo y configurarlo.



El nuevo origen de eventos deberá crearse durante la instalación de la aplicación. De esta manera, el sistema operativo dispondrá de tiempo para actualizar la lista de orígenes de eventos registrados y sus configuraciones. Si el sistema operativo aún no ha actualizado la lista de orígenes de eventos y se intenta escribir un evento con el nuevo origen, se producirá un error de la operación de escritura. Para configurar un nuevo origen, puede utilizarse [EventLogInstaller](#) o el método [CreateEventSource](#). Es necesario contar con derechos administrativos en el equipo para crear un nuevo origen de eventos.

Cuando una aplicación escriba entradas usando tanto identificadores de recursos como valores de cadena, deberán registrarse dos orígenes diferentes. Por ejemplo, puede configurarse un origen con archivos de recursos y usarlo en el método [WriteEvent](#) para escribir entradas en el registro de eventos mediante identificadores de recursos. A continuación, puede crearse otro origen sin archivos de recursos y usarlo en el método [WriteEntry](#) para escribir cadenas directamente en el registro de eventos.

[Option Explicit](#)  
[Option Strict](#)

```
Imports System
Imports System.Diagnostics
Imports System.Threading
```

```
Class MySample
    Public Shared Sub Main()
```

```
        If Not EventLog.SourceExists("MySource") Then
            ' Crear origen si no existe
            EventLog.CreateEventSource("MySource", "MyNewLog")
            Console.WriteLine("CreatingEventSource")
        End If
```

```
        ' Crea una instancia de EventLog y le asigna el origen MySource
        Dim myLog As New EventLog()
        myLog.Source = "MySource"
```

```
        'escribe la entrada en el EventLog instanciado
        myLog.WriteEntry("Escritura de un log")
```

```
    End Sub
End Class
```

### Propiedades

Nombre	Descripción
Entries	Obtiene el contenido del registro de eventos.
Events	Obtiene la lista de controladores de eventos asociados a Component. (Se hereda de Component).
Log	Obtiene o establece el nombre del registro del que se lee o en el que se escribe.
LogDisplayName	Obtiene el nombre descriptivo del registro de eventos.
MachineName	Obtiene o establece el nombre del equipo en el que se van a leer o en el que se van a escribir los eventos.
MaximumKilobytes	Obtiene o establece el tamaño máximo del registro de eventos, en kilobytes.
Source	Obtiene o establece el nombre de origen que se va a registrar y utilizar al escribir en el registro de eventos.

### Métodos

Nombre	Descripción
Clear	Quita todas las entradas del registro de eventos.
CreateEventSource	Sobrecargado. Establece una aplicación como capaz de escribir información de eventos en un determinado registro de actividad del sistema.

Delete	Sobrecargado. Quita un recurso de registro.
DeleteEventSource	Sobrecargado. Quita del registro de eventos un registro del origen de eventos de una aplicación.
Exists	Sobrecargado. Determina si existe el registro especificado.
SourceExists	Sobrecargado. Busca en un origen de eventos dado el Registro de un equipo.
WriteEntry	Sobrecargado. Escribe una entrada en el registro de eventos.
WriteEvent	Sobrecargado. Escribe una entrada de evento adaptada en el registro de eventos.

## 2. **Process** : System.Diagnostics

Un componente **Process** proporciona acceso a un proceso que se está ejecutando en un equipo. Un proceso, dicho de un modo sencillo, es una aplicación en ejecución. Un subproceso es la unidad básica a la que el sistema operativo asigna tiempo de procesador. Un subproceso puede ejecutar cualquier parte del código del proceso, incluidas las partes que otro subproceso esté ejecutando en ese momento.

El componente **Process** es una herramienta útil para iniciar, detener, controlar y supervisar aplicaciones. Mediante el componente **Process** se puede obtener una lista de los procesos en ejecución o se puede iniciar un nuevo proceso. Un componente **Process** se utiliza para obtener acceso a los procesos del sistema. Un componente **Process**, una vez inicializado, puede utilizarse para obtener información sobre el proceso en ejecución. Entre la información proporcionada se incluye el conjunto de subprocesos, los módulos cargados (archivos .dll y .exe) y la información de rendimiento, por ejemplo, la cantidad de memoria que está utilizando el proceso.

**Class** MyProcess

' These are the Win32 error code for file not found or access denied.

**Private** ERROR\_FILE\_NOT\_FOUND **As Integer** = 2

**Private** ERROR\_ACCESS\_DENIED **As Integer** = 5

/' <summary>

/' Prints a file with a .doc extension.

/' </summary>

**Sub** PrintDoc()

**Dim** myProcess **As New** Process()

**Try**

' Get the path that stores user documents.

**Dim** myDocumentsPath **As String** = \_

Environment.GetFolderPath(Environment.SpecialFolder.Personal)

myProcess.StartInfo.FileName = myDocumentsPath + "MyFile.doc"

myProcess.StartInfo.Verb = "Print"

myProcess.StartInfo.CreateNoWindow = **True**

myProcess.Start()

```

Catch e As Win32Exception
    If e.NativeErrorCode = ERROR_FILE_NOT_FOUND Then
        Console.WriteLine((e.Message + ". Check the path.))

    Else
        If e.NativeErrorCode = ERROR_ACCESS_DENIED Then
            ' Note that if your word processor might generate exceptions
            ' such as this, which are handled first.
            Console.WriteLine((e.Message + ". You do not have permission to print
            this file.))
        End If
    End If
End Try
End Sub 'PrintDoc

Public Shared Sub Main()
    Dim myProcess As New MyProcess()
    myProcess.PrintDoc()
End Sub 'Main
End Class 'MyProcess

```

### 3. WMI: System.Management

Proporciona acceso a un amplio conjunto de información y eventos de administración relacionados con el sistema, dispositivos y aplicaciones instrumentadas para la infraestructura Windows Management Instrumentation (WMI). Las aplicaciones y los servicios pueden consultar información de interés sobre administración (como por ejemplo cuánto espacio libre queda en el disco, cuál es el nivel actual de utilización de la CPU, a qué base de datos está conectada cierta aplicación y mucho más), por medio de clases derivadas de ManagementObjectSearcher y ManagementQuery, o suscribirse a diversos eventos de administración por medio de la clase ManagementEventWatcher. El acceso a los datos es posible a través de componentes administrados y no administrados en el entorno distribuido.

```

Public Class Sample
    Public Overloads Shared Function _
        Main(ByVal args() As String) As Integer

        Dim myScope As New ManagementScope("root\CIMV2")
        Dim q As New SelectQuery("Win32_LogicalDisk")
        Dim s As New ManagementObjectSearcher(myScope, q)

        For Each disk As ManagementObject In s.Get()
            'show the disk instance
            Console.WriteLine(disk.ToString())
        Next

    End Function 'Main
End Class 'Sample

```

Imports System

**Imports** System.Management

**Public Class** Sample

**Public Overloads Shared Function** \_

Main(**ByVal** args() **As String**) **As Integer**

```
Dim s As New ManagementObjectSearcher( _
    "root\MyApp", _
    "SELECT * FROM Win32_Service", _
    New EnumerationOptions( _
        Nothing, System.TimeSpan.MaxValue, 1, _
        True, False, True, True, False, _
        True, True))
```

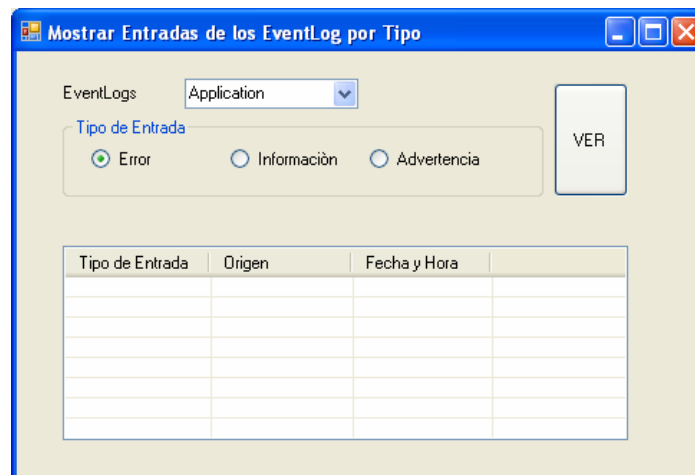
```
For Each service As ManagementObject In s.Get()
    'show the instance
    Console.WriteLine(service.ToString())
Next
```

**End Function** 'Main  
**End Class**

## LABORATORIO 9.1

**Consulta de los sucesos y notificaciones utilizando EventLog:**  
System.Diagnostics

1. Ingrese a Visual Studio 2008
2. Seleccione Menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Application" y en Name escriba "Laboratorio9\_1"
4. Desarrolla la siguiente GUI.



'Namespace  
**Imports** System.Diagnostics

```
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'Lista de Eventos Log de Application
        Dim ev As New EventLog("Application")
        For Each el As EventLogEntry In ev.Entries
            ListBox1.Items.Add(el.Index.ToString + vbTab + el.Source)
        Next
    End Sub

    Private Sub ListBox1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles ListBox1.Click
        'Al Seleccionar un Entry visualizamos sus detalles
        'Capturamos un Entry, utilizando el SelectedIndex
        Dim ev As New EventLog("Application")
        Dim el As EventLogEntry = ev.Entries(ListBox1.SelectedIndex)

        'con una Cadena almaceno su contenido
        Dim str As String = "Origen: " + el.Source + vbCrLf
        str += "Tiempo: " + el.TimeGenerated.ToString + vbCrLf
        str += "Tipo de Cuenta: " + el.EntryType.ToString + vbCrLf
        str += "Mensaje: " + el.Message
        Me.txtcontenido.Text = str
        ev.Close()
    End Sub

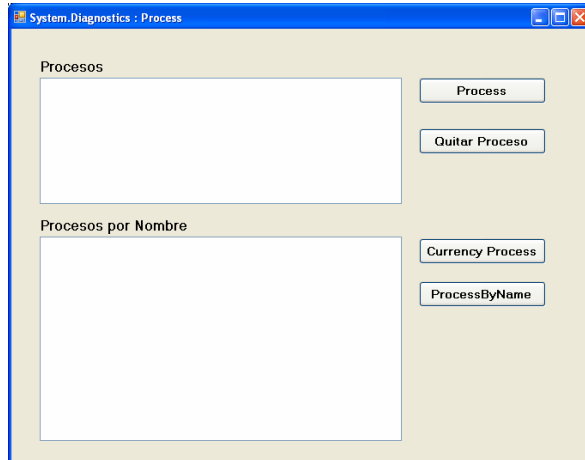
    Private Sub txtcontenido_TextChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles txtcontenido.TextChanged

    End Sub
End Class
```

## LABORATORIO 9.2

Cree una aplicación Windows que permita visualizar la lista de procesos activos para luego eliminarlos una vez que se haya seleccionado un proceso.

1. Ingrese a Visual Studio 2008
2. Seleccione Menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Application" y en Name escriba "Laboratorio1\_1"
4. Diseñar la siguiente GUI del formulario



5. Colocar el siguiente código en la vista código del formulario.
6. Importar el namespace para utilizar la clase Process.

```
Imports System.Diagnostics
Public Class Form2
```

```
Private Sub BtnCurrencyProcess_Click(ByVal sender As Object, ByVal e As
EventArgs) Handles BtnCurrencyProcess.Click
```

```
    'Visualice el Proceso Actual
```

```
    Dim pr As Process = Process.GetCurrentProcess
```

```
    ListBox2.Items.Clear()
```

```
    ListBox2.Items.Add("Proceso Actual")
```

```
    ListBox2.Items.Add("Id: " + pr.Id.ToString)
```

```
    ListBox2.Items.Add("Nombre: " + pr.ProcessName)
```

```
    ListBox2.Items.Add("Memoria: " + pr.VirtualMemorySize64.ToString)
```

```
End Sub
```

```
Private Sub BtnProcess_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BtnProcess.Click
```

```
    'Listado de Procesos
```

```
    ListBox1.Items.Clear()
```

```
    For Each p As Process In Process.GetProcesses
```

```
        'Procesos por su ID, Memoria Virtual y el Nombre del Proceso
```

```
        ListBox1.Items.Add(p.Id.ToString + vbTab + _
```

```
        p.VirtualMemorySize64.ToString _
```

```
        + vbTab + p.ProcessName)
```

```
    Next
```

```
End Sub
```



```
Private Sub BtnByName_Click(ByVal sender As Object, ByVal e As EventArgs)
Handles BtnByName.Click
    'Proceso por su Name, Id
    'Capturo el ID del Proceso seleccionado en el Listbox1
    Dim id As Integer
    id = Val(ListBox1.Text.Substring(0, ListBox1.Text.IndexOf(vbTab)))
    'pr recibe el Proceso segun el id seleccionado
    Dim pr As Process = Process.GetProcessById(id)
    ListBox2.Items.Clear()
    'Visualizo los procesos por su nombre, que sean comunes
    For Each p As Process In Process.GetProcessesByName(pr.ProcessName)
        ListBox2.Items.Add(p.ProcessName + vbTab + p.Id.ToString)
    Next
End Sub

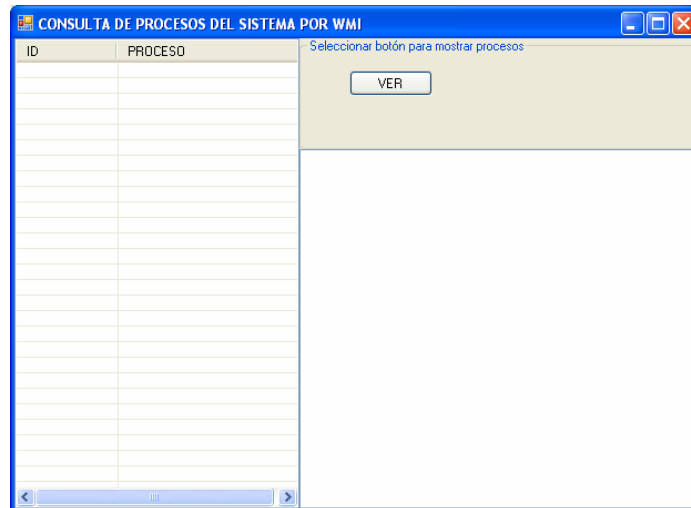
Private Sub BtnQuitar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BtnQuitar.Click
    Dim id As Integer
    id = Val(ListBox1.Text.Substring(0, ListBox1.Text.IndexOf(vbTab)))
    'pr recibe el Proceso segun el id seleccionado
    Dim p As Process = Process.GetProcessById(id)
    Dim rp As Integer
    rp = MessageBox.Show("Quitar", "KiLL", MessageBoxButtons.OKCancel)
    If rp = vbOK Then
        p.Kill() 'Cerrar un Proceso
    End If

End Sub
End Class
```

## LABORATORIO 9.3

Cree una aplicación Windows que permita realizar consulta de los procesos. Utilizando para ello la clase WMI.

1. Ingrese a Visual Studio 2008
2. Seleccione Menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Application" y en Name escriba "Laboratorio9\_3"
4. Diseñar la siguiente GUI.



5. Colocar el siguiente código dentro del código fuente del formulario.

```
Imports System.Management
```

```
Public Class Form1
```

```
Private Sub BTNVER_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTNVER.Click
```

```
    ListView1.Items.Clear()
```

```
    Dim ITEM As New ListViewItem
```

```
    Dim sprocesso As New ManagementObjectSearcher( "Select * From Win32_Process")
```

```
    For Each p As ManagementObject In sprocesso.Get
```

```
        ITEM = ListView1.Items.Add(p("ProcessId"))
```

```
        ITEM.SubItems.Add(p("Name").ToString)
```

```
    Next
```

```
End Sub
```

```
Private Sub ListView1_SelectedIndexChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles ListView1.SelectedIndexChanged
```

```
    If ListView1.SelectedItems.Count > 0 Then
```

```
        Try
```

```
            Dim id As String = ListView1.SelectedItems(0).SubItems(1).Text
```

```
            Dim sprocesso As New ManagementObjectSearcher ( _  
                "Select * From Win32_Process Where Name=" + id + "" ) _
```

```
            Me.TextBox1.Text = Nothing
```

```
            For Each p As ManagementObject In sprocesso.Get
```

```
                Me.TextBox1.Text = "Titulo: " + p("caption") + vbCrLf
```

```
                Me.TextBox1.Text += "Ruta: " + p("ExecutablePath") + vbCrLf
```

```
                Me.TextBox1.Text += "Name: " + p("Name") + vbCrLf
```

```
                Me.TextBox1.Text += "Status: " + p("Status") + vbCrLf
```

```
                Me.TextBox1.Text += "WindowsVersion: " + p("WindowsVersion") +  
                vbCrLf
```

```
            Next
```

```
Catch ex As ManagementException
```

```
        MessageBox.show(ex.Message)
    End Try
End If
End Sub
End Class
```

## Autoevaluación

1. ¿Qué es un EventLog?

---

---

---

2. ¿Qué tipos de EventLogs tiene Windows por defecto?

---

---

---

3. ¿Qué es un proceso?

---

---

---

4. ¿Qué es WMI?

---

---

---

## Para Recordar

- El espacio de nombres **System.Diagnostics** proporciona también clases que permiten depurar la aplicación y hacer un seguimiento de la ejecución del código. Para obtener más información, vea las clases Trace y Debug.
- La clase Process proporciona funcionalidad para supervisar los procesos de sistema en toda la red y para iniciar y detener procesos del sistema local.
- El componente EventLog proporciona la funcionalidad para escribir en registros de eventos, leer las entradas de los registros de eventos y crear y eliminar registros de eventos y orígenes de eventos en la red. EntryWrittenEventHandler proporciona una forma para interactuar con los registros de eventos de manera asincrónica.
- El System.Management proporciona acceso a un amplio conjunto de información y eventos de administración relacionados con el sistema, dispositivos y aplicaciones instrumentadas para la infraestructura Windows Management Instrumentation (WMI).

**Nota:** Texto adaptado de las páginas web:

**<http://msdn.microsoft.com/es-es/library/system.diagnostics.aspx>**

**<http://msdn.microsoft.com/es-es/library/system.management.aspx>**



**UNIDAD DE  
APRENDIZAJE**

**4**

**SEMANA**

**10**

## **Implementación de Serialización en .NET**

---

### **LOGRO DE LA UNIDAD DE APRENDIZAJE**

Al término de la unidad, los alumnos utilizando algoritmos y librerías del Framework .NET 2.0, construyen aplicaciones Windows .NET para optimizar el manejo, control, almacenamiento y seguridad de los datos.

### **TEMARIO**

- Administración Serialización y deserialización de un objeto
- Serialización con XML
- Serializacion Binaria
- Serialización SOAP

### **ACTIVIDADES PROPUESTAS**

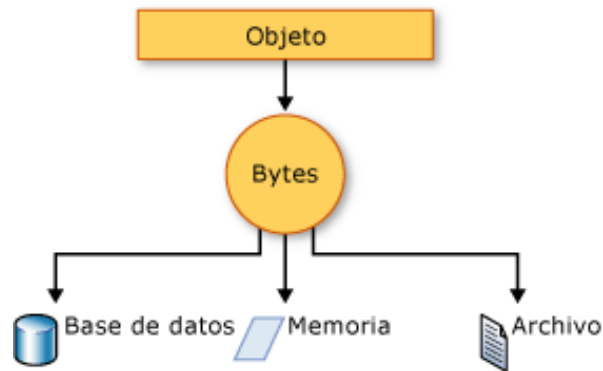
- Los alumnos crean aplicaciones que serializan un tipo de objeto.
- Los alumnos utilizan para sus aplicaciones los formatos XML y binario

## 1. Serializacion y Deserializacion

La serialización es el proceso de convertir un objeto en una secuencia de bytes para conservarlo en memoria, una base de datos o un archivo. Su propósito principal es guardar el estado de un objeto para poder crearlo de nuevo cuando se necesita. El proceso inverso se denomina deserialización.

### ¿Cómo funciona la serialización?

Este ejemplo muestra el proceso total de serialización.



El objeto se serializa en una secuencia que, además de los datos, contiene información sobre el tipo de objeto, como la versión, referencia cultural y nombre de ensamblado. Esa secuencia se puede almacenar en una base de datos, un archivo o en memoria.

### Usos de la serialización

La serialización permite al desarrollador guardar el estado de un objeto y volver a crearlo cuando es necesario, y proporcionar almacenamiento de objetos e intercambio de datos. A través de la serialización, un desarrollador puede realizar acciones como enviar un objeto a una aplicación remota por medio de un servicio Web, pasar un objeto de un dominio a otro, pasar un objeto a través de un firewall como una cadena XML o mantener la seguridad o información específica del usuario entre aplicaciones.

### Crear un objeto serializable

Para serializar un objeto, se necesita que el objeto esté serializado, una secuencia que contenga el objeto serializado y un objeto [Formatter](#). La enumeración [System.Runtime.Serialization](#) contiene las clases necesarias para serializar y deserializar objetos.

Aplique el atributo [SerializableAttribute](#) a un tipo para indicar que se pueden serializar las instancias de ese tipo. Se produce una excepción [SerializationException](#) si se intenta serializar pero el tipo no tiene el atributo [SerializableAttribute](#).

Si no desea que un campo de su clase se pueda serializar, aplique el atributo [NonSerializedAttribute](#). Si un campo de un tipo serializable contiene un puntero, controlador u otra estructura de datos específica de un entorno determinado, y el significado del campo no se puede reconstruir en un entorno diferente, es conveniente que no sea serializable.

Si una clase serializada contiene referencias a objetos de otras clases marcadas con el atributo [SerializableAttribute](#), esos objetos también se serializarán.



## 2. Serialización binaria y XML

Se puede utilizar serialización binaria o XML. En la serialización binaria, se serializan todos los miembros, incluso aquellos que son de sólo lectura, y se mejora el rendimiento. La serialización XML proporciona código más legible, así como mayor flexibilidad para compartir objetos y utilizarlos para fines de interoperabilidad.

### 2.1. Serialización binaria

La serialización binaria utiliza la codificación binaria a fin de generar una serialización compacta para usos como almacenamiento o secuencias de red basadas en sockets. No es conveniente pasar los datos a través de un firewall, pero proporciona mejor rendimiento al almacenarlos.

#### BinaryFormatter (Clase)

Serializa o deserializa un objeto o todo un gráfico de objetos conectados, en formato binario.

**Espacio de nombres:** [System.Runtime.Serialization.Formatters.Binary](#)

[Imports](#) System.IO

[Imports](#) System.Collections

[Imports](#) System.Runtime.Serialization.Formatters.Binary

[Imports](#) System.Runtime.Serialization

[Module](#) App

[Sub](#) Main()

    Serialize()

    Deserialize()

[End Sub](#)

[Sub](#) Serialize()

    ' Create a hashtable of values that will eventually be serialized.

    Dim addresses As New Hashtable

    addresses.Add("Jeff", "123 Main Street, Redmond, WA 98052")

    addresses.Add("Fred", "987 Pine Road, Phila., PA 19116")

    addresses.Add("Mary", "PO Box 112233, Palo Alto, CA 94301")

    ' To serialize the hashtable (and its key/value pairs),

    ' you must first open a stream for writing.

    ' In this case, use a file stream.

    Dim fs As New FileStream("DataFile.dat", FileMode.Create)

    ' Construct a BinaryFormatter and use it to serialize the data to the stream.

    Dim formatter As New BinaryFormatter

    Try

        formatter.Serialize(fs, addresses)

    Catch e As SerializationException

        Console.WriteLine("Failed to serialize. Reason: " & e.Message)

        Throw

    Finally

        fs.Close()

    End Try

```

End Sub

Sub Deserialize()
    ' Declare the hashtable reference.
    Dim addresses As Hashtable = Nothing

    ' Open the file containing the data that you want to deserialize.
    Dim fs As New FileStream("DataFile.dat", FileMode.Open)
    Try
        Dim formatter As New BinaryFormatter

        ' Deserialize the hashtable from the file and
        ' assign the reference to the local variable.
        addresses = DirectCast(formatter.Deserialize(fs), Hashtable)
    Catch e As SerializationException
        Console.WriteLine("Failed to deserialize. Reason: " & e.Message)
        Throw
    Finally
        fs.Close()
    End Try

    ' To prove that the table deserialized correctly,
    ' display the key/value pairs.
    Dim de As DictionaryEntry
    For Each de In addresses
        Console.WriteLine("{0} lives at {1}.", de.Key, de.Value)
    Next
End Sub
End Module

```

## 2.2. Serialización XML

La serialización XML serializa las propiedades y campos públicos de un objeto o los parámetros y valores devueltos de los métodos en una secuencia XML que se ajusta a un documento específico del lenguaje de definición de esquemas XML (XSD). La serialización XML produce clases con establecimiento inflexible de tipos cuyas propiedades y campos se convierten a XML. La enumeración [System.Xml.Serialization](#) contiene las clases necesarias para serializar y deserializar XML.

Se pueden aplicar atributos a clases y miembros de clase para controlar la manera en que [XmlSerializer](#) serializa o deserializa una instancia de la clase. Para obtener más información, vea [Controlar la serialización XML mediante atributos](#) y [Atributos que controlan la serialización XML](#).

### System.Xml.Serialization (Espacio de nombres)

El espacio de nombres **System.Xml.Serialization** contiene clases que se utilizan para serializar objetos en secuencias o documentos con formato XML.

La clase central en el espacio de nombres es la clase [XmlSerializer](#). Para utilizar esta clase, use el constructor [XmlSerializer](#) con el fin de crear una instancia de la clase utilizando el tipo de objeto que se va a serializar. Tras crear [XmlSerializer](#), cree una instancia del objeto que se va a serializar. También hay que crear un objeto para escribir el archivo en un documento o una secuencia, como [Stream](#), [TextWriter](#) o [XmlWriter](#). A continuación, hay que llamar al método `Serialize()` para convertir el objeto en un documento XML.

## Serializar un objeto en XML

1. Cree el objeto y establezca sus campos públicos y propiedades.
2. Construya un `XmlSerializer` utilizando el tipo de objeto. Para obtener más información, vea los constructores de clase **`XmlSerializer`**.
3. Llame al método `Serialize` para generar o una secuencia XML o una representación del archivo de las propiedades públicas del objeto y campos. En el ejemplo siguiente se crea un archivo.

## Cómo: Deserializar un objeto

Al deserializar un objeto, el formato de transporte determina si creará una secuencia u objeto de archivo. Una vez determinado el formato de transporte, puede llamar [Serialize](#) o los métodos [Deserialize](#), como se requiera.

## Para deserializar un objeto en XML

1. Construya un [XmlSerializer](#) utilizando el tipo del objeto para deserializar.
2. Llame al método **`Deserialize`** para generar una réplica del objeto. Al deserializar, debe convertir el objeto devuelto al tipo del original, como se muestra en el ejemplo siguiente, que deserializa el objeto en un archivo (aunque también se pudo deserializar en una secuencia).

## Para crear una secuencia XML con un nombre de elemento alternativo

1. Cree una instancia de la clase `XmlElementAttribute`.
2. Establece el `ElementName` de `XmlElementAttribute` a "BookID".
3. Cree una instancia de la clase `XmlAttributes`.
4. Agregue el objeto **`XmlElementAttribute`** a la colección a la que ha accedido mediante la propiedad `XmlElements` de `XmlAttributes`.
5. Cree una instancia de la clase `XmlAttributeOverrides`.
6. Agregue **`XmlAttributes`** a `XmlAttributeOverrides`, pasando el tipo del objeto para invalidarlo y el nombre de miembro invalidado.
7. Cree una instancia de la clase **`XmlSerializer`** con **`XmlAttributeOverrides`**.
8. Cree una instancia de la clase `Book` y serialice o deserialícela.

## 2.3. Serialización SOAP

La serialización XML también se puede usar para serializar objetos en secuencias XML que se ajustan a la especificación SOAP. SOAP es un protocolo basado en XML, diseñado específicamente para transportar llamadas a procedimiento utilizando XML. Al igual que la serialización XML normal, estos atributos también pueden utilizarse para controlar los mensajes SOAP de estilo literal que genera un servicio Web XML. Para obtener más información, vea [Serialización XML con servicios web XML](#) y [Atributos que controlan la serialización SOAP codificada](#).

### Para serializar un objeto como secuencia XML con codificación SOAP

1. Cree la clase utilizando Herramienta de definición de esquema XML (Xsd.exe).
2. Aplique uno o más de los atributos especiales situados en **System.Xml.Serialization**. Vea la lista en "Atributos que controlan la serialización SOAP codificada".
3. Cree **XmlTypeMapping** creando un nuevo **SoapReflectionImporter** invocando el método **ImportTypeMapping** con el tipo de la clase serializada.

En el ejemplo de código siguiente llama el método **ImportTypeMapping** de la clase **SoapReflectionImporter** para crear un **XmlTypeMapping**.

## 3. System.IO.Compression (Espacio de nombres)

El espacio de nombres **System.IO.Compression** contiene clases que proporcionan servicios de compresión y descompresión básica para las secuencias.

### Clases

Clase	Descripción
<a href="#">DeflateStream</a>	Proporciona métodos y propiedades para comprimir y descomprimir secuencias mediante el algoritmo Deflate.
<a href="#">GZipStream</a>	Proporciona los métodos y propiedades que permiten comprimir y descomprimir secuencias

La clase **DeflateStream** no proporciona de forma inherente la funcionalidad para agregar o extraer archivos en archivos .zip. Para obtener un ejemplo de manipulación de archivos comprimidos, vea [Ejemplo de aplicación de compresión](#).

La clase **GZipStream** utiliza el formato de datos gzip, que incluye un valor de prueba cíclica de redundancia para detectar daños en los datos. El formato de datos gzip utiliza el mismo algoritmo de compresión que la clase **DeflateStream**.

La funcionalidad de compresión en **DeflateStream** y **GZipStream** se expone como una secuencia. Los datos se leen byte a byte, por lo que no es posible realizar varios pases a fin de determinar el método más adecuado para comprimir archivos enteros o grandes bloques de datos. Las clases **DeflateStream** y **GZipStream** se utilizan de forma óptima en orígenes de datos sin comprimir. Si los datos de origen ya están comprimidos, el uso de estas clases puede aumentar el tamaño de la secuencia.

## LABORATORIO 10.1

Cree una aplicación Windows que permita ingresar el Nombre de un usuario y muestre una ventana emergente de bienvenida con el nombre de usuario.

1. Ingrese a Visual Studio 2008
2. Seleccione Menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Application" y en Name escriba "Laboratorio10\_1"
4. Diseñar la GUI del formulario.



### Clase Persona

**Imports** System.Drawing

**Imports** System.Runtime.Serialization

'<Serializable(> : Este atributo indica que la clase

'serà serializada

<Serializable(> Public Class Persona

Implements IDeserializationCallback

```
'Atributos privados
Private mName As String
Private mApellido As String

'Atributo público que no será serializado
<NonSerialized(> Public Edad As Int32

'Atributo público que guardará una imagen
Public Foto As Bitmap

'constructor de la clase
Sub New(ByVal Nombre As String, ByVal Apellido As String)
    'Asignar valores a los atributos privados de la clase
    mName = Nombre
    mApellido = Apellido
End Sub

'propiedades de sólo lectura
'WriteOnly-->Atributo de sólo escritura
Public ReadOnly Property Nombre() As String
    Get
        Return mName
    End Get
End Property

Public ReadOnly Property Apellido() As String
    Get
        Return mApellido
    End Get
End Property

Public Sub OnDeserialization(ByVal sender As Object) Implements
System.Runtime.Serialization.IDeserializationCallback.OnDeserialization
    MessageBox.Show("La edad no se serializó")
End Sub
End Class

Formulario
```

'Namespace para usar clase FileStream

Imports System.IO

'Namespace para usar la clase BinaryFormatter

Imports System.Runtime.Serialization.Formatters.Binary

Public Class frmSerializacionBinaria

Private Sub btnCargar\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnCargar.Click

    'Utilizo una ventan OpenFileDialog

    Dim opDialog As New OpenFileDialog

    'Establecer 2 filtros para esta ventana

    'opDialog.Filter=Nombre a Mostrar|Filtro1;Filtro2

    opDialog.Filter = \_

    "Picture files|.jpg;\*.gif;\*.bmp|All files (\*.\*)|\*.\*"

    opDialog.FilterIndex = 1

    If opDialog.ShowDialog() = Windows.Forms.DialogResult.OK Then

        picFoto.Image = Image.FromFile(opDialog.FileName)

    End If

End Sub

Private Sub btnLimpiar\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnLimpiar.Click

    txtApellido.Text = ""

    txtNombre.Text = ""

    txtEdad.Text = ""

    picFoto.Image = Nothing

End Sub

Private Sub btnGrabar\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnGrabar.Click

    'Creo una instancia de la clase Persona

```

Dim oPersona As New Persona(txtNombre.Text, txtApellido.Text)

'establecer valores para los atributos públicos
oPersona.Edad = Convert.ToInt32(txtEdad.Text)

'Función de conversión:CType(Expresión,Tipo de dato a Convertir)
oPersona.Foto = CType(picFoto.Image, Bitmap)

'Uso una ventana SaveFileDialog para definir el archivo a crear
Dim svDialog As New SaveFileDialog()

'establecer filtro de archivos a crear
svDialog.Filter = "Archivo Data(*.bin)|*.bin|All files (*.*)|*.*"

'Mostrar ventana SaveFileDialog y verificar
'que se dio click al botón Guardar
If svDialog.ShowDialog() = Windows.Forms.DialogResult.OK Then

    'Crear el archivo con el nombre elegido
    'en la ventana SaveFileDialog

    Dim fs As New FileStream _
        (svDialog.FileName, FileMode.Create, FileAccess.Write)

    'Serializo el objeto

    Dim objBF As New BinaryFormatter

    'se ejecuta la serialización y se guarda los datos
    'serializados en un archivo fs
    objBF.Serialize(fs, oPersona)

    'cerrar archivo
    fs.Close()

    MessageBox.Show("Se grabó con éxito.")
End If

```

**End Sub**

```

Private Sub btnRecuperar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnRecuperar.Click

```

```

    'Uso un Common Dialog para localizar el archivo

```



```
Dim oDialog As New OpenFileDialog()
oDialog.Filter = "Archivo data(*.bin)|*.bin|All files (*.*)|*.*"
'Mostrar Ventana y verificar si presionò botòn Abrir
If oDialog.ShowDialog = Windows.Forms.DialogResult.OK Then
    'crear objeto FileStream para leer archivo que contiene la data serializada
    Dim fs As New FileStream(oDialog.FileName, FileMode.Open,
    FileAccess.Read)
    'Crear objeto BinaryFormatter para deserializar
    Dim objBF As New BinaryFormatter
    'Deserializar el objeto convirtiéndolo al tipo Persona
    Dim oPersona As Persona = CType(objBF.Deserialize(fs), Persona)
    'Muestro los valores deserializados
    txtNombre.Text = oPersona.Nombre
    txtApellido.Text = oPersona.Apellido
    picFoto.Image = oPersona.Foto
    'cerrar archivo
    fs.Close()
End If
End Sub
End Class
```

## 4. LABORATORIO 10.2

Cree una aplicación Windows que permita ingresar datos de un objeto Persona y luego se serialice esta información en formato XML.

1. Ingrese a Visual Studio 2008
2. Seleccione Menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Application" y en Name escriba "Laboratorio10\_2"
4. Diseñe la siguiente GUI.

'Namespace para usar la clase FileStream

Imports System.IO

'Namespace para usar la clase XMLSerializer

Imports System.Xml.Serialization

Public Class frmSerializacionXML

Private Sub btnSerializacion\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnSerializar.Click

Dim svDialog As New SaveFileDialog

svDialog.Filter = "Archivos XML|\*.xml"

If svDialog.ShowDialog() = Windows.Forms.DialogResult.OK Then

'crear archivo con el nombre indicado en la ventana

'SaveFileDialog

Dim fs As New FileStream \_

(svDialog.FileName, FileMode.Create, FileAccess.Write)

'Preparar los datos a serializar

'Crear un instancia de la clase profesor

Dim objProfesor As New Profesor

'asignar valores a los atributos públicos

objProfesor.strCodigo = txtCodigo.Text

objProfesor.strNombre = txtNombre.Text

objProfesor.strApellidos = txtApellidos.Text

objProfesor.fecFechaIngreso = dtFechaIngreso.Value.Date

```
'Dim NombreArreglo(IndiceMaximo) as tipo
Dim arrAsignaturas(IstAsignaturas.Items.Count - 1) As String
'Asignar valores al arreglo creado
For i As Int16 = 0 To IstAsignaturas.Items.Count - 1
    arrAsignaturas(i) = IstAsignaturas.Items.Item(i)
Next
'Pasar el arreglo al atributo asignaturas del objeto profesor
objProfesor.asignaturas = arrAsignaturas
'Crear variable para serializar en formato XML
'Dim xs As New XmlSerializer(Tipo de dato a serializar)
Dim xs As New XmlSerializer(GetType(Profesor))
'ejecutar la serialización en formato XML
xs.Serialize(fs, objProfesor)
fs.Close()

End If

End Sub

Private Sub cboAsignatura_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cboAsignatura.SelectedIndexChanged

    'IndexOf(Elemento):Busca el elemento elegido en el combo
    ' dentro de la lista.Si lo encuentra retorna su indice sino retorna -1
    If IstAsignaturas.Items.IndexOf(cboAsignatura.SelectedItem) = -1 Then
        'Agregar a la lista lo seleccionado en el combo
        IstAsignaturas.Items.Add(cboAsignatura.SelectedItem)
    End If

End Sub

Private Sub btnDeserializacion_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnDeserializar.Click

    Dim opDialog As New OpenFileDialog
    opDialog.Filter = "Archivos XML |*.xml"

    If opDialog.ShowDialog() = Windows.Forms.DialogResult.OK Then
```

```
Dim xs As XmlSerializer = New XmlSerializer(GetType(Profesor))

Dim fs As FileStream = New FileStream(opDialog.FileName, FileMode.Open,
FileAccess.Read)

Dim objProfesor As New Profesor
objProfesor = CType(xs.Deserialize(fs), Profesor)

Dim strReporte As String = ""
strReporte &= "Código:" & objProfesor.strCodigo & vbCrLf
strReporte &= "Nombres:" & objProfesor.strNombre & vbCrLf
strReporte &= "Apellidos:" & objProfesor.strApellidos & vbCrLf
strReporte &= "Fecha:" & objProfesor.fecFechaIngreso & vbCrLf
strReporte &= "Asignaturas:"

For Each asig As String In objProfesor.asignaturas
    strReporte &= asig & vbCrLf
Next

MessageBox.Show(strReporte)

fs.Close()

End If

End Sub

End Class
```

## Autoevaluación

1. ¿Qué es la serialización?

---

---

---

2. ¿Qué formatos proporciona el Framework .NET para serializar objetos?

---

---

---

3. ¿Qué namespace se requiere para serializar un objeto con el formato XML?

---

---

---

## Para Recordar

- El espacio de nombres **System.Runtime.Serialization.Formatters.Binary** contiene la clase `BinaryFormatter`, que se puede utilizar para serializar y deserializar los objetos en formato binario.
- `BinaryFormatter` serializa o deserializa un objeto o todo un gráfico de objetos conectados, en formato binario.
- El espacio de nombres **System.Xml.Serialization** contiene clases que se utilizan para serializar objetos en secuencias o documentos con formato XML.
- **XmlSerializer** serializa y deserializa objetos en y desde documentos XML. Permite controlar el modo en que se codifican los objetos en XML.

### Fuente:

[http://msdn.microsoft.com/es-](http://msdn.microsoft.com/es-es/library/system.runtime.serialization.formatters.binary.binaryformatter.aspx)

[es/library/system.runtime.serialization.formatters.binary.binaryformatter.aspx](http://msdn.microsoft.com/es-es/library/system.runtime.serialization.formatters.binary.binaryformatter.aspx)

<http://msdn.microsoft.com/es-es/library/system.xml.serialization.aspx>

**UNIDAD DE  
APRENDIZAJE**

**4**

**SEMANA**

**11**

## **Administración de Sistemas de Archivos**

---

### **LOGRO DE LA UNIDAD DE APRENDIZAJE**

Al Al término de la unidad, los alumnos utilizando algoritmos y librerías del Framework .NET 3.5, construyen aplicaciones Windows .NET para optimizar el manejo, control, almacenamiento y seguridad de los datos.

### **TEMARIO**

- Acceso a los archivos y directorios
- Manejo de datos utilizando los objetos Stream
- Compresión de datos

### **ACTIVIDADES PROPUESTAS**

- Los alumnos acceden a los archivos y carpetas usando la clase File System.
- Los alumnos manejan los datos de una aplicación usando las clases Reader, Writer y Streams.
- Los alumnos comprenden la diferencia entre las clases DeflateStream y ZipStream para comprimir archivos

## 1. Administracion De Sistemas De Archivos

### 1.1. Acceso a archivos con Visual Basic

El objeto **My.Computer.FileSystem** proporciona herramientas para trabajar con archivos y carpetas. Sus propiedades, métodos y eventos permiten crear, copiar, mover, investigar y eliminar archivos y carpetas. **My.Computer.FileSystem** ofrece un mayor rendimiento que las funciones heredadas (**FileOpen**, **FileClose**, **Input**, **InputString**, **LineInput**, etc.) proporcionadas por Visual Basic para ofrecer compatibilidad con versiones anteriores.

El método **ReadAllText** del objeto **My.Computer.FileSystem** permite leer de un archivo de texto. Se puede especificar la codificación del archivo si el contenido del mismo utiliza una codificación como ASCII o UTF-8.

Si está leyendo de un archivo que incluye caracteres extendidos, deberá especificar la codificación del archivo.

#### Para leer de un archivo de texto

Utilice el método **ReadAllText** del objeto **My.Computer.FileSystem** para leer el contenido de un archivo de texto en una cadena, proporcionando la ruta de acceso. El ejemplo siguiente lee el contenido del archivo test.txt, lo coloca en una cadena y, a continuación, lo muestra en un cuadro de mensaje.

```
Dim fileReader As String  
fileReader = My.Computer.FileSystem.ReadAllText("C:\test.txt")  
MessageBox.show(fileReader)
```

#### Para leer de un archivo de texto que está codificado

Utilice el método **ReadAllText** del objeto **My.Computer.FileSystem** para leer el contenido de un archivo de texto en una cadena, proporcionando la ruta de acceso y el tipo de codificación del archivo. El ejemplo siguiente lee el contenido del archivo UTF32 test.txt, lo coloca en una cadena y, a continuación, lo muestra en un cuadro de mensaje.

Las condiciones siguientes pueden producir una excepción:

- La ruta de acceso no es válida por una de las razones siguientes: es una cadena de longitud cero, sólo contiene un espacio en blanco, contiene caracteres no válidos o es una ruta de acceso de dispositivo ([ArgumentException](#)).
- La ruta de acceso no es válida porque es **Nothing** ([ArgumentNullException](#)).
- El archivo no existe ([FileNotFoundException](#)).
- El archivo está en uso por otro proceso o hay un error de E/S ([IOException](#)).
- La ruta supera la longitud máxima definida por el sistema ([PathTooLongException](#)).



- Un nombre de archivo o de directorio de la ruta de acceso contiene un signo de dos puntos (:) o tiene un formato no válido ([NotSupportedException](#)).
- No hay suficiente memoria para escribir la cadena en el búfer ([OutOfMemoryException](#)).
- El usuario no tiene los permisos necesarios para ver la ruta de acceso ([SecurityException](#)).

### Leer archivos de texto delimitado por comas en Visual Basic

El objeto **TextFieldParser** proporciona un método para analizar de forma sencilla y eficaz archivos de texto estructurados, como registros. La propiedad **TextFieldType** define si se trata de un archivo delimitado o uno con campos de ancho fijo de texto.

#### Para analizar un archivo de texto delimitado por comas

1. Cree un nuevo objeto **TextFieldParser**. El código siguiente crea el objeto **TextFieldParser** denominado **MyReader** y abre el archivo **test.txt**.

```
Using MyReader As New Microsoft.VisualBasic.FileIO.TextFieldParser _  
("C:\TestFolder\test.txt")  
End Using
```

2. Defina el tipo de **TextField** y el delimitador. El código siguiente define la propiedad **TextFieldType** como **Delimited** y el delimitador como **","**.

```
MyReader.TextFieldType = FileIO.FieldType.Delimited  
MyReader.SetDelimiters (",")
```

3. Recorra los campos del archivo. Si alguna línea está dañada, cree un informe de error y continúe el análisis. El código siguiente recorre el archivo para mostrar cada campo a la vez e indica los campos con formato incorrecto.

```
Dim currentRow As String()  
While Not MyReader.EndOfData  
Try  
currentRow = MyReader.ReadFields()  
Dim currentField As String  
For Each currentField In currentRow  
MsgBox(currentField)  
Next  
Catch ex As Microsoft.VisualBasic.FileIO.MalformedLineException  
MsgBox("Line " & ex.Message & _  
"is not valid and will be skipped.")  
End Try  
End while
```

4. Cierre los bloques **While** y **Using** con **End While** y **End Using**.

Ejemplo

En este ejemplo se lee el archivo **test.txt**.

```

Using MyReader As New _
Microsoft.VisualBasic.FileIO.TextFieldParser("C:\testfile.txt")
MyReader.TextFieldType = FileIO.FieldType.Delimited
MyReader.SetDelimiters(",")
Dim currentRow As String()
While Not MyReader.EndOfData
    Try
        currentRow = MyReader.ReadFields()
        Dim currentField As String
        For Each currentField In currentRow
            MsgBox(currentField)
        Next
    Catch ex As Microsoft.VisualBasic.FileIO.MalformedLineException
        MsgBox("Line " & ex.Message & "is not valid and will be skipped.")
    End Try
End While
End Using

```

```

Microsoft.VisualBasic.FileIO.MalformedLineException
MsgBox("Line " & ex.Message & _
"is not valid and will be skipped.")
End Try
End While
End Using

```

### Para eliminar archivos y carpetas

Option Explicit On

Public Class Form1

```

Private Sub Form1_Load( ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load
    Try
        ' Eliminar el archivo, mostrando el cuadro
        'de diálogo de eliminar de windows para confirmar
        Dim sdir As String = "c:\Nueva carpeta"
        Dim Spath As String = "c:\archivo.txt"
        ' Archivo
        My.Computer.FileSystem.DeleteFile(Spath, _
FileIO.UIOption.AllDialogs, FileIO.RecycleOption.SendToRecycleBin, _
FileIO.UICancelOption.DoNothing)
        ' carpeta
    End Try
End Sub

```

```
My.Computer.FileSystem.DeleteDirectory( _
    sdir, _
    FileIO.UIOption.AllDialogs, _
    FileIO.RecycleOption.SendToRecycleBin, _
    FileIO.UICancelOption.DoNothing)
' errores
```

Catch ex As Exception

```
MsgBox(ex.Message.ToString, MsgBoxStyle.Critical)
```

End Try

End Sub

End Class

## 1.2. System.IO (Espacio de nombres)

El espacio de nombres System.IO contiene tipos que permiten leer y escribir en los archivos y secuencias de datos, así como tipos que proporcionan compatibilidad básica con los archivos y directorios.

Directory	Expone métodos estáticos para crear, mover y enumerar archivos en directorios y subdirectorios. Esta clase no se puede heredar.
DirectoryInfo	Expone métodos de instancia para crear, mover y enumerar archivos en directorios y subdirectorios. Esta clase no se puede heredar.
DriveInfo	Proporciona acceso a información sobre una unidad.
DriveNotFoundException	La excepción que se produce al intentar obtener acceso a una unidad o un recurso compartido que no está disponible.
File	Proporciona métodos estáticos para crear, copiar, eliminar, mover y abrir archivos y contribuye a la creación de objetos FileStream.
FileInfo	Proporciona métodos de instancia para crear, copiar, eliminar, mover y abrir archivos y contribuye a la creación de objetos FileStream. Esta clase no se puede heredar.
FileNotFoundException	Excepción que se produce cuando se produce un error al intentar tener acceso a un archivo que no existe en el disco.
FileStream	Expone un objeto Stream alrededor de un archivo; se admiten operaciones de lectura y escritura sincrónica y asincrónica.
FileSystemInfo	Proporciona la clase base para los objetos FileInfo y DirectoryInfo.
FileSystemWatcher	Escucha las notificaciones de cambio del sistema de archivos y provoca eventos cuando cambia un directorio o un archivo de un directorio.
MemoryStream	Crea una secuencia cuyo almacén de respaldo es la memoria.
Stream	Proporciona una vista genérica de una secuencia de bytes.

<a href="#">StreamReader</a>	Implementa un TextReader que lee los caracteres de una secuencia de bytes en una codificación determinada.
<a href="#">StreamWriter</a>	Implementa TextWriter para escribir los caracteres de una secuencia en una codificación determinada.
<a href="#">StringReader</a>	Implementa TextReader que lee en una cadena.

## LABORATORIO 11.1

### Manipular archivos y directorios en Visual Basic

1. En el menú Archivo, haga clic en Nuevoproyecto.  
Aparecerá el cuadro de diálogo Nuevo proyecto.
2. En el panel Tipos de proyecto, haga clic en Proyectos de Visual Basic y, a continuación, elija Aplicación para Windows en el panel Plantillas.
3. En el cuadro Nombre, escriba [FileExplorer](#) como nombre del proyecto.  
Visual Studio agregará el proyecto al Explorador de soluciones y se abrirá el Diseñador de Windows Forms.
4. Agregue los controles de la siguiente tabla al formulario y establezca los correspondientes valores para sus propiedades.

Objeto	Propiedades	Valor
TextBox	Name	txtDirectory
	Text	Directorio
Button	Name	btnSubmit
	Text	Enviar
Buton	Name	btnExamine
	Text	Exminar
ComboBox	Name	IstFilePick
	Text	Selecione Archivo
CheckBox	Name	chkFileLength
	Text	Longitud de archivo
	Checked	True
CheckBox	Name	chkLastAccess
	Text	Hora de último acceso
	Checked	True
CheckBox	Name	chkSave
	Text	Guardar resultados
	Checked	False

### Mostrar el directorio actual

La aplicación FileExplorer necesita un punto de inicio. En consecuencia, el control `txtDirectoryTextBox` utiliza la función `My.Computer.FileSystem.CurrentDirectory` para devolver y mostrar una cadena que representa la ruta de acceso actual.

#### Para devolver el directorio actual

1. Haga doble clic en el formulario para crear un controlador de eventos para `Form1_Load`.  
Se abrirá el Editor de código.
2. Agregue el código siguiente para que el control `txtDirectoryTextBox` muestre la ubicación actual.

```
txtDirectory.Text = My.Computer.FileSystem.CurrentDirectory
```

3. Ejecute el programa para comprobar que se devuelve la ruta de acceso correcta.

El control `txtDirectoryTextBox` muestra el directorio actual.

### Cambiar directorios

Dado que es posible que un usuario desee seleccionar archivos de un directorio diferente, la aplicación utiliza la misma propiedad para cambiar de directorio. Para cambiar a un directorio diferente, el usuario escribe una nueva ruta de acceso en el control `txtDirectoryTextBox`.

#### Para cambiar de directorio

1. Haga doble clic en el control del formulario para crear un controlador de eventos clic para `btnSubmit`.  
Se abrirá el Editor de código.
2. Agregue el código siguiente al controlador de eventos clic.

```
Dim NewPath As String  
' NewPath holds the path the user has entered.  
NewPath = txtDirectory.Text  
' Change the location to NewPath.  
My.Computer.FileSystem.CurrentDirectory = NewPath
```

### Comprobar si se escribió una ruta de acceso válida

Utilice una instrucción `Try...Catch` para detectar excepciones que surgen de escribir una ruta de acceso en blanco o no válida.

#### Para garantizar rutas de acceso válidas

1. En el evento `btnSubmit_Click`, después de la línea de código `Dim NewPath As String`, agregue `Dim ErrorMessage As String` en una nueva línea.
2. Antes de la línea de código `My.Computer.FileSystem.CurrentDirectory = NewPath`, agregue una instrucción `Try` en su propia línea, como se indica a

continuación. Si presiona la tecla de retorno, el Editor de código insertará automáticamente las instrucciones `Catch ex As Exception` y `End Try`. Quítelas, agregará las suyas propias en el paso siguiente.

- Después de la línea de código `My.Computer.FileSystem.CurrentDirectory = NewPath`, agregue lo siguiente:

```
Try
    ' This checks to make sure the path is not blank.
Catch ex As Exception When NewPath = ""
    ErrorMessage = "You must enter a path."
    ' This catches errors caused by a path that is not valid.
Catch
    ErrorMessage = "You must enter a
valid path. If trying " & _
    "to access a different drive, remember to include the drive " & _
    "letter."
Finally
    ' Display the error message only if one exists.
    If ErrorMessage <> Nothing Then
        MsgBox(ErrorMessage)
    End If
End Try
```

### Mostrar el contenido del directorio en un control ComboBox

Para permitir que la aplicación muestre el contenido del directorio actual, puede utilizar el método `My.Computer.FileSystem.GetFiles`, que devuelve una colección de cadenas que representan los nombres de los archivos en el directorio. Puede utilizar comodines con `GetFiles` para seleccionar sólo archivos de un modelo determinado. En este ejemplo, sólo se devuelven los archivos que tienen la extensión `.txt`.

#### Para mostrar el contenido del directorio

- Al principio del evento `btnSubmit_Click`, inserte lo siguiente.  
`Dim fileList As System.Collections.ObjectModel.ReadOnlyCollection(Of String).`
- Después de que la línea `End Try`, inserte lo siguiente.  

```
fileList = My.Computer.FileSystem.GetFiles( _
    My.Computer.FileSystem.CurrentDirectory, _
    FileIO.SearchOption.SearchTopLevelOnly, "*.txt")
For Each foundFile As String In fileList
    lstFilePick.Items.Add(foundFile)
Next
```
- La información recopilada aparece en el control `lstFilePick` **ComboBox**, en el que podrá seleccionar un archivo específico para examinarlo.

Para probar la aplicación, ejecútela primero en un directorio que no contenga archivos `.txt` y, a continuación, en uno que contenga más de un archivo `.txt`. En el primer caso, la aplicación muestra el mensaje de error correspondiente. En el segundo, la

aplicación crea, en el control [ComboBox](#), una lista de todos los archivos .txt del directorio especificado en el control [txtDirectoryTextBox](#).

Permitir que el usuario seleccione un archivo para examinarlo  
Aunque el control [lstFilePickComboBox](#) muestra todos los archivos de un directorio, es probable que el usuario desee seleccionar y examinar un archivo específico.

### Para habilitar la selección de un archivo concreto

Cree un controlador de evento clic para [btnExamine\\_Click](#) y agregue el código siguiente para confirmar la selección de un archivo.

### Permitir que el usuario determine la información que desea recopilar

Una vez que se muestran los archivos en el control [lstFilePickComboBox](#), el código adicional permite que el usuario especifique la información de la que se informa. Por ejemplo, es posible que un usuario sólo desee saber la fecha en la que se tuvo acceso al archivo por última vez. Es posible que otro usuario desee conocer también el tamaño de un archivo. Los usuarios pueden activar o desactivar las casillas de verificación ([chkLastAccess](#), [chkFileLength](#)) para personalizar los resultados.

### Para mostrar información específica

1. Declare estas variables al principio del evento [btnExamine\\_Click](#) después de ([lstFilePick.SelectedItem](#)):

```
Dim stringlength As String
stringLength = "The file's length, in bytes, is: "
Dim stringLastAccess As String
stringLastAccess = "The file was last accessed on: "
Dim LastAccess As Date
Dim Length As Long
Dim FirstLine As String = ""
Dim FinalString As String = ""
Dim NewName As String
NewName = CType(lstFilePick.SelectedItem, String)

If NewName = Nothing Then
    MsgBox("You must select a file to examine.")
    Exit Sub
End If
```

2. El método **My.Computer.FileSystem.GetFileInfo** devuelve un objeto [FileInfo](#) que se puede consultar para obtener información sobre un archivo.
3. Agregue el código siguiente al final del evento [btnExamine\\_Click](#).  
' Check last access time.  
If [chkLastAccess.Checked](#) = True Then  
    LastAccess = thisFile.LastAccessTime  
End If

La propiedad [LastAccessTime](#) determina la hora del último acceso al archivo. El valor [Date](#) devuelto indica la fecha y la hora en que se creó un archivo o en que fue modificado por última vez.

1. Agregue el código siguiente al final del evento [btnExamine\\_Click](#).

```

' Check Length.
If chkFileLength.Checked = True Then
    Length = thisFile.Length
End If

```

La propiedad `Length`, que determina la longitud del archivo, devuelve un valor de tipo **Long** que especifica la longitud del archivo en bytes.

### Mostrar los resultados

Para completar la funcionalidad de la aplicación, un **MsgBox** informa de la información recopilada.

### Para mostrar los resultados

1. Al final de la instrucción `If` que determina si se ha activado o no el control `chkLastAccessCheckBox`, agregue lo siguiente antes de la instrucción `End If` final.

```

' Add to the messagebox.
FinalString = FinalString & stringLastAccess & LastAccess & "." _
& vbCrLf

```

2. Al final de la instrucción `If` que determina si se ha activado o no el control `chkFileLengthCheckBox`, agregue lo siguiente antes de la instrucción `End If` final.

```

' Add to the messagebox.
FinalString = FinalString & stringlength & CStr(Length) & "." & vbCrLf

```

3. Al final de la instrucción `If` que determina si se ha activado o no el control `chkFirstLineCheckBox`, agregue lo siguiente antes de la instrucción `End If` final.

```

' Add to the messagebox.
FinalString &= FirstLine & vbCrLf

```

### Guardar los resultados

Es posible que el usuario desee guardar los resultados de examinar un archivo. En consecuencia, debe agregar código que compruebe si existe un archivo de registro, cree uno si es necesario y, a continuación, escriba los resultados en el archivo de registro.

### Para crear un archivo de registro:

Agregue lo siguiente al final del evento `btnExamine_Click`.

```

' Check to see if results should be saved.
If chkSave.Checked = True And FinalString <> "" Then
    My.Computer.FileSystem.WriteAllText("log.txt", FinalString, True)
End If

```



### Para probar la aplicación

1. En el directorio de su elección, cree un archivo de texto denominado `test.txt` con la primera línea siguiente:  
"Ésta es la primera línea del primer archivo. La aplicación FileExplorer sólo examina archivos de texto."
2. En el mismo directorio, cree un segundo archivo de texto denominado `test2.txt` con la primera línea siguiente:  
"Ésta es la primera línea del segundo archivo. La aplicación FileExplorer sólo examina archivos de texto."
3. Inicie la aplicación.
4. Escriba una ruta de acceso no válida y haga clic en Enviar.  
Aparecerá el siguiente mensaje: "You must enter a valid path. If trying to access a different drive, remember to include the drive letter."
5. Escriba la ruta de acceso al directorio que almacena `test.txt` y haga clic en Enviar.  
El control `IstFilePickComboBox` mostrará los archivos de texto.
6. Seleccione `test.txt` en el control `IstFilePickComboBox`. Asegúrese de que están activadas todas las casillas de verificación y, a continuación, haga clic en Examinar.  
El formulario de resultados incluye la fecha del último acceso y la longitud.
7. Seleccione `test2.txt` en el control `IstFilePickComboBox`, desactive todas las casillas de verificación y, a continuación, haga clic en Examinar.  
Aparecerá el siguiente mensaje de error : "No file attribute checkboxes selected."
8. Seleccione Hora del último acceso y Guardar resultados, y haga clic en Examinar.  
El formulario de resultados muestra sólo la hora del último acceso.
9. Cierre FileExplorer.  
Dado que activó la opción Guardar resultados, FileExplorer generó un archivo de registro denominado `log.txt` en el mismo directorio que los archivos de texto.

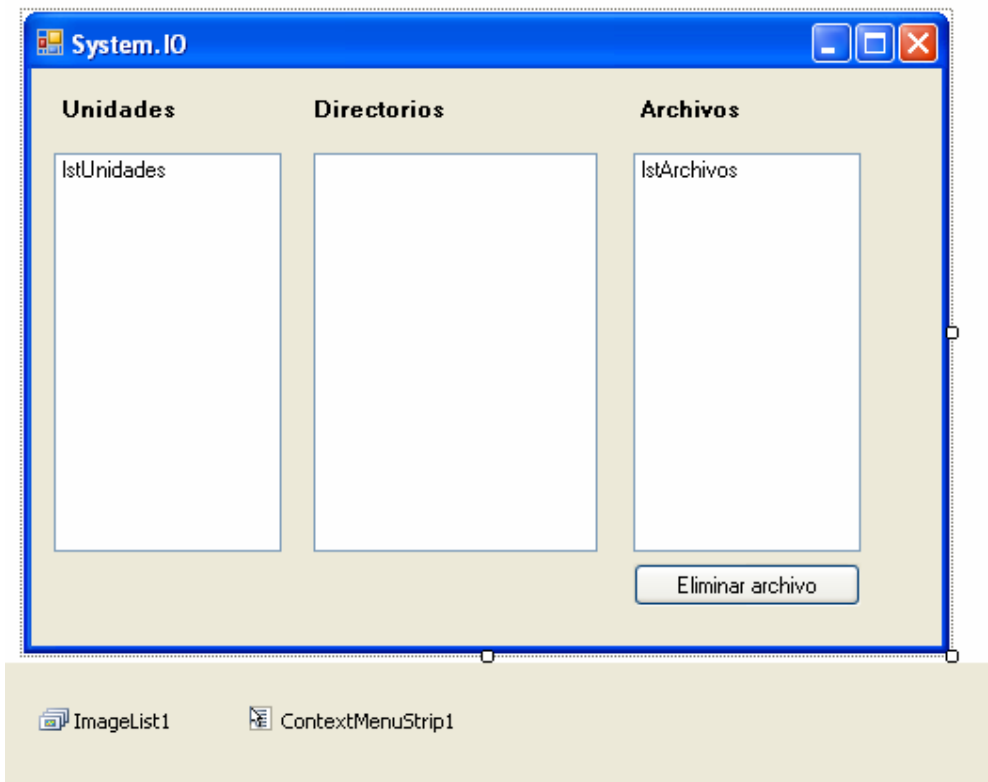
### Para comprobar el registro

En el directorio actual, abra `log.txt` y confirme que FileExplorer registró la información correcta.

## LABORATORIO 11.2

Cree una aplicación Windows que permita ingresar y visualizar un miniexplorador para archivos y directorios.

1. Ingrese a Visual Studio 2008
2. Seleccione Menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Application" y en Name escriba "Laboratorio11\_2"
4. Diseñe la siguiente GUI.



5. Complete el siguiente código fuente.

```
'Namespace para el manejo de archivos y directorios
Imports System.IO
Public Class frm_Ejemplo1
    Sub mostrarUnidades()
        'Crear un arreglo de unidades de la PC
        'DriveInfo.GetDrives-->método q obtiene una colección de todas
        'las unidades de la PC
        Dim drives() As DriveInfo = DriveInfo.GetDrives
        'recorrer el arreglo y agregar c/unidad a la lista
        For Each drive As DriveInfo In drives
            lstUnidades.Items.Add(drive.Name)
        Next
    End Sub
```

```
Private Sub frm_Ejemplo1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    mostrarUnidades()
End Sub
Sub MostrarDirectios()
    'recuperar la unidad seleccionada
    Dim unidadSeleccionada As String = IstUnidades.SelectedItem
    'Obtener los directorios referentes a una unidad
    Dim misDirectorios As DirectoryInfo = _
    New DirectoryInfo(unidadSeleccionada)
    Try
        lvDirectorios.Items.Clear()
        'variable q representa a un elemento del Listview
        Dim ele As ListViewItem
        'recorrer la colección de directorios
        For Each Dir As DirectoryInfo In misDirectorios.GetDirectories
            ele = lvDirectorios.Items.Add(Dir.Name)
            ele.ImageKey = "Cube.ico"
        Next
    Catch ex As IOException
        MessageBox.Show(ex.Message)
    End Try
End Sub
Sub MostrarArchivos()
    'recuperar la unidad seleccionada
    Dim unidadSelect As String = IstUnidades.SelectedItem
    'recuperar directorio seleccionado
    Dim directorioSelect As String = lvDirectorios.SelectedItems.Item(0).Text
    'Obtener los archivos referentes al directorio
    Dim misArchivos As DirectoryInfo = _
    New DirectoryInfo(unidadSelect & directorioSelect)
    Try
        IstArchivos.Items.Clear()
        'recorrer la colección de archivos
        For Each archivo As FileInfo In misArchivos.GetFiles
            IstArchivos.Items.Add(archivo)
        Next
    Catch ex As IOException
        MessageBox.Show(ex.Message)
    Catch ex As Exception
        MessageBox.Show(ex.Message)
    End Try
End Sub

Private Sub IstUnidades_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
IstUnidades.SelectedIndexChanged
    MostrarDirectios()
End Sub

Private Sub PropiedadesToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
PropiedadesToolStripMenuItem.Click
    Dim unidad As String = IstUnidades.SelectedItem
```

```

Dim u As New DriveInfo (unidad)
MessageBox.Show ("Tipo Unidad:" & u.DriveType.ToString & vbCrLf _
& "Espacio Libre:" & Math.Round(u.TotalFreeSpace / 1024 / 1024 / 1024, _
2) & "GB" & vbCrLf & "Capacidad Total:" & Math.Round(u.TotalSize / 1024 /
1024 / 1024, 2) & "GB" , "Información de Unidad" , _
MessageBoxButtons.OK, MessageBoxIcon.Information)
End Sub

Private Sub lvDirectorios_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
lvDirectorios.SelectedIndexChanged
If lvDirectorios.SelectedItems.Count > 0 Then
MostrarArchivos()
End If
End Sub
End Class

```

## LABORATORIO 11.3

1. Cree una nueva aplicación de consola llamada ComprimiendoArchivos
2. Agregue una clase llamada CompresionArchivos y declare dos métodos “**ComprimirArchivo**” y “**DescomprimirArchivo**” los cuales recibirán dos parámetros “**archivoEntrada**” y “**archivoSalida**” de tipo string.
3. Implemente el método “**ComprimirArchivos**”, en donde utilizará dos objetos de tipo FileStream que servirán para leer el archivo origen y guardar el archivo comprimido.

La clase **GZipStream**, la cual en su constructor recibe como argumentos el Stream de salida y un modo de Compresion para indicar si vamos a comprimir o des-comprimir datos. Utilice el método “**Write**” para escribir la información al stream de salida.

4. Implemente el método “**DescomprimirArchivo**”, a continuación la forma de implementación es la misma, con la única diferencia en la forma en que inicializamos el objeto tipo GZipStream, también aquí no se puede hacer uso de la instrucción compStream.Lenght por que nos genera una excepción indicando que esta funcionalidad no está implementada.

Para leer los datos utilice un ciclo while para ir leyendo la información. Veamos como queda este método:

Como vemos los únicos cambios realizados fueron los siguientes: El constructor de la clase GZipStream ahora utiliza el modo “**Decompress**” para indicar que vamos a descomprimir la información de un archivo. Luego leemos toda la información del archivo haciendo uso de la instrucción compStream.ReadByte().

5. Para hacer uso de la clase solo invoque a los dos metodos desde Sub Main().

Si desea usar el algoritmo **DEFLATE**, lo unico que debe modificar en la clase que se ha desarrollado es el uso de la Clase GzipStream por DeflateStream, incluso el constructor de la clase **DeflateStream** utiliza los mismos argumentos que la clase GzipStream.

¿Cuándo utilizar GZipStream y cuándo utilizar DeflateStream ?, ambas clases utilizan el mismo algoritmo de compresión, la única diferencia es que el formato GZip agrega información en el header para que pueda ser interpretada por programas como winzip. Mientras que Deflate no agrega esta información, por lo que si no van a utilizar los archivos con aplicaciones comerciales, es más recomendable utilizar deflate.

## Autoevaluación

1. ¿Qué namespace y clases son requeridos para leer archivos de texto?

---

---

---

2. ¿ Si se desea ver los archivos que existen en un directorio que clase se debe utilizar del System.IO?

---

---

---

3. ¿ En qué se diferencian las clases DeflateStream y Gzipstream ?.  
¿Cuáles son los usos que se les da?

---

---

---

## Para Recordar

- El espacio de nombres System.IO contiene tipos que permiten leer y escribir en los archivos y secuencias de datos, así como tipos que proporcionan compatibilidad básica con los archivos y directorios.
- El objeto **My.Computer.FileSystem** proporciona herramientas para trabajar con archivos y carpetas.
- Para abrir y leer el archivo, se utilizan también objetos del nombre de espacio System.IO, específicamente la clase System.IO.StreamReader.
- En el .net Framework 2.0 se utilizan dos nuevas clases incluidas en el NameSpace System.IO.Compression se trata de: GZipStream y DeflateStream.
- La clase **MemoryStream** crea secuencias que utilizan como almacén de respaldo la memoria en lugar de un disco o una conexión de red. MemoryStream encapsula los datos almacenados como una matriz de bytes sin signo que se inicializa al crear un objeto MemoryStream; también se puede crear una matriz vacía. Es posible obtener acceso directamente a los datos encapsulados en la memoria. Las secuencias de memoria pueden reducir la necesidad de archivos y búferes temporales en una aplicación.

### Fuente

Texto Adpatado en las páginas web:

<http://carloslone.wordpress.com/2007/08/27/comprimiendo-archivos-en-net-20/>

[http://msdn.microsoft.com/es-es/library/system.io\(VS.80\).aspx](http://msdn.microsoft.com/es-es/library/system.io(VS.80).aspx)





**UNIDAD DE  
APRENDIZAJE**

**5**

**SEMANA**

**12**

## **Seguridad en Aplicaciones Windows**

---

### **LOGRO DE LA UNIDAD DE APRENDIZAJE**

Al término de la unidad, los alumnos elaborarán aplicaciones de mejor rendimiento y acceso asignando recursos, permisos, componentes, cultura y una adecuada distribución de la misma para su implementación.

### **TEMARIO**

- Control, permisos y privilegios a recursos
- Acceso y modificación de la información de usuarios
- Creación de unidades de aislamiento para el CLR

### **ACTIVIDADES PROPUESTAS**

- Los alumnos se crearán aplicaciones para encriptar información, utilizando algunos de los algoritmos simétricos o asimétricos de encriptación.

## 1. Seguridad en .NET Framework

Common Language Runtime y .NET Framework proporcionan un gran número de clases y servicios útiles que permiten a los programadores escribir de manera sencilla código de seguridad. Estas clases y servicios permiten también que los administradores del sistema personalicen el acceso del código a los recursos protegidos. Asimismo, el motor de tiempo de ejecución y .NET Framework proporcionan clases y servicios útiles que facilitan el uso de criptografía y la seguridad basada en funciones.

## 2. Seguridad basada en funciones

Las aplicaciones empresariales suelen proporcionar acceso a datos o recursos basándose en credenciales proporcionadas por el usuario. Normalmente, dichas aplicaciones comprueban la función de un usuario y proporcionan acceso a recursos basándose en dicha función. Common Language Runtime proporciona compatibilidad para la autorización basada en funciones tomando como base una cuenta de Windows o una identidad personalizada.

### 4. Objetos Principal e Identity

El código administrado puede descubrir la identidad o la función de un principal a través de un objeto [Principal](#), que contiene una referencia a un objeto [Identity](#). Puede ser útil la comparación de objetos de identidad y principales con conceptos familiares, como las cuentas de grupo y usuario. En la mayoría de los entornos de red, las cuentas de usuario representan a personas o programas, mientras que las cuentas de grupo representan a ciertas categorías de usuarios y los derechos que poseen. De forma similar, los objetos de identidad de .NET Framework representan a usuarios, mientras que las funciones representan pertenencias y contextos de seguridad. En .NET Framework, el objeto principal encapsula un objeto de identidad y una función. Las aplicaciones .NET Framework conceden derechos al principal basándose en su identidad o, lo que es más normal, en su pertenencia a una función.

### Objetos Identity

El objeto de identidad encapsula información relativa al usuario o la entidad que se van a validar. En su nivel más básico, los objetos de identidad contienen un nombre y un tipo de autenticación. El nombre puede ser un nombre de usuario o el nombre de una cuenta de Windows, mientras que el tipo de autenticación puede ser un protocolo de inicio de admisión admitido, como Kerberos V5, o un valor personalizado. .NET Framework define un objeto [GenericIdentity](#) que se puede utilizar para la mayoría de los escenarios de inicio de sesión personalizados y un objeto [WindowsIdentity](#) más especializado que se puede utilizar cuando se desea que la aplicación se base en una autenticación de Windows. Además, se puede definir la clase de identidad propia que encapsula la información personalizada del usuario.

La interfaz [IIdentity](#) define propiedades para el acceso a un nombre y un tipo de autenticación, como Kerberos V5 o NTLM. Todas las clases **Identity** implementan la interfaz **IIdentity**. No hay una relación necesaria entre un objeto **Identity** y el símbolo (token) de proceso de Windows NT bajo el cual se está ejecutando actualmente un subproceso. No obstante, si el objeto **Identity** es un objeto **WindowsIdentity**, se supone que la identidad representa un símbolo de seguridad de Windows NT.

## Objetos Principal

El objeto principal representa el contexto de seguridad bajo el cual se ejecuta código. Las aplicaciones que implementan seguridad basada en funciones conceden derechos tomando como base la función asociada a un objeto principal. De forma similar a los objetos de identidad, .NET Framework proporciona un objeto [GenericPrincipal](#) y un objeto [WindowsPrincipal](#). También se pueden definir clases principales personalizadas propias.

La interfaz [IPrincipal](#) define una propiedad para el acceso a un objeto **Identity** asociado, así como un método para determinar si el usuario identificado por el objeto **Principal** es miembro de una función dada. Todas las clases **Principal** implementan la interfaz **IPrincipal**, así como las propiedades y métodos adicionales necesarios. Por ejemplo, Common Language Runtime proporciona la clase **WindowsPrincipal**, que implementa funcionalidad adicional para asignar a funciones la pertenencia a un grupo de Windows NT o Windows 2000.

Un objeto **Principal** se enlaza a un objeto de contexto de llamada ([CallContext](#)) dentro de un dominio de aplicación ([AppDomain](#)). Un contexto de llamada predeterminado se crea siempre con cada dominio **AppDomain** nuevo, por lo que siempre hay un contexto de llamada disponible para aceptar el objeto **Principal**. Cuando se crea un subproceso nuevo, también se crea un objeto **CallContext** para el subproceso. La referencia del objeto **Principal** se copia automáticamente desde el subproceso creador en el objeto **CallContext** del subproceso nuevo. Si el motor en tiempo de ejecución no puede determinar qué objeto **Principal** pertenece al creador del subproceso, sigue la directiva predeterminada para la creación de objetos **Principal** e **Identity**.

Una directiva específica de dominio de aplicación configurable define las reglas para decidir qué tipo de objeto **Principal** se ha de asociar a un dominio de aplicación nuevo. Cuando la directiva de seguridad lo permite, el motor en tiempo de ejecución puede crear objetos **Principal** e **Identity** que reflejan el símbolo (token) de sistema operativo asociado al subproceso actual de ejecución. De forma predeterminada, el motor en tiempo de ejecución utiliza objetos **Principal** e **Identity** que representan a usuarios no autenticados, pero no los crea hasta que el código intenta el acceso a ellos.

El código de confianza que crea un dominio de aplicación puede establecer la directiva de dominio de aplicación que controla la construcción de los objetos **Principal** e **Identity** predeterminados. Esta directiva específica de dominio de aplicación se aplica a todos los subprocesos de ejecución de dicho dominio. Un host de confianza no administrado tiene la capacidad inherente de establecer esta directiva, pero el código administrado que la establece debe tener el permiso `System.Security.Permissions` para controlar la directiva de dominio.

Cuando se transmite un objeto **Principal** a través de dominios de aplicación, pero dentro del mismo proceso (y, por tanto, en el mismo equipo), la infraestructura de interacción remota copia una referencia en el objeto **Principal** asociado al contexto de quien efectúa una llamada en el contexto de quien la recibe.

## Comprobaciones de seguridad basada en funciones

Una vez definidos los objetos de identidad y principal, se pueden realizar comprobaciones de seguridad frente a ellos mediante uno de los métodos siguientes:

- Utilización de comprobaciones de seguridad imperativa.
- Utilización de comprobaciones de seguridad declarativa.
- Acceso directo al objeto [Principal](#).

El código administrado puede utilizar comprobaciones de seguridad imperativa o declarativa para determinar si un objeto principal concreto es miembro de una función conocida, tiene una identidad conocida o representa una identidad conocida que actúa en una función. Para que la comprobación de seguridad se realice utilizando seguridad imperativa o declarativa, se debe efectuar una demanda de seguridad para un objeto [PrincipalPermission](#) correctamente creado. Durante la comprobación de seguridad, Common Language Runtime examina el objeto principal de quien efectúa una llamada para determinar si su identidad y su función coinciden con las representadas por el objeto **PrincipalPermission** que se demanda. Si el objeto de entidad de seguridad no coincide, se produce una excepción [SecurityException](#). Sólo se examina el objeto principal del subproceso actual; la clase **PrincipalPermission** no produce un recorrido de pila con permiso de acceso a código.

Además, se puede tener acceso directo a los valores del objeto principal y realizar comprobaciones sin un objeto **PrincipalPermission**. En este caso, basta con leer los valores del principal del subproceso actual o utilizar la autorización de ejecución del método **IsInRole**.

### Realizar comprobaciones de seguridad imperativas

En el caso de una demanda imperativa, se puede llamar al método **Demand** del objeto **PrincipalPermission** para determinar si el objeto de entidad de seguridad actual representa la identidad especificada, la función especificada o ambas. Suponiendo que se ha creado apropiadamente un objeto **PrincipalPermission** denominado [MyPrincipalPermission](#), se puede llamar a una demanda imperativa con el código siguiente.

En el ejemplo de código siguiente se utiliza una comprobación imperativa para garantizar que un objeto [GenericPrincipal](#) coincide con el objeto **PrincipalPermission**. Una comprobación imperativa es útil cuando muchos métodos u otros ensamblados del dominio de aplicación deben tomar determinaciones basadas en funciones. Aunque este ejemplo es sumamente simple, ilustra el comportamiento asociado a una petición basada en funciones.

```
Imports System.Security.Permissions
Imports System.Security.Principal
Imports System.Security
Imports System.Threading
Imports System.Security.Cryptography
```

```
Public Class MainClass
```

```
    Public Overloads Shared Function Main() As Integer
```

```
        Console.WriteLine("Enter '1' to use the proper identity or " _
            & "any other character to use the improper identity.")
```

```
        If Console.ReadLine() = "1" Then
```

```
            ' Create a generic identity.
```

```
            Dim MyIdentity As New GenericIdentity("MyUser")
```

```
            ' Create a generic principal.
```

```
            Dim MyString As [String]() = {"Administrator", "User"}
```

```
            Dim MyPrincipal As New GenericPrincipal( _
                MyIdentity, MyString)
```

```
        Thread.CurrentPrincipal = MyPrincipal
    End If

    PrivateInfo()

    Return 0
End Function

Public Shared Sub PrivateInfo()
    Try
        ' Create a PrincipalPermission object.
        Dim MyPermission As New PrincipalPermission( _
            "MyUser", "Administrator")

        ' Demand this permission.
        MyPermission.Demand()

        ' Print secret data.
        Console.WriteLine(ControlChars.Cr & ControlChars.Cr & _
            "You have access to the private data!")
    Catch e As SecurityException
        Console.WriteLine(e.Message)
    End Try
End Sub
End Class
```

### Realizar comprobaciones de seguridad declarativa

Las demandas declarativas de permisos [PrincipalPermission](#) funcionan de la misma manera que las demandas declarativas de permisos de acceso a código. Las demandas se pueden colocar en el nivel de clase así como sobre eventos, propiedades y métodos individuales. Si una demanda declarativa se coloca en el nivel de clase y de miembro, la demanda declarativa sobre el miembro reemplaza la demanda en el nivel de clase.

El ejemplo de código siguiente muestra una versión modificada del método [PrivateInfo](#) a partir del ejemplo de la sección anterior. Esta versión utiliza seguridad declarativa. [PrincipalPermissionAttribute](#) define la entidad de seguridad que el subprocesso actual debe tener para invocar el método. Basta con pasar la clase [SecurityAction.Demand](#) con el nombre y la función que se requieren.

```
Public Shared Sub _
    <PrincipalPermissionAttribute(SecurityAction.Demand, Name := "MyUser", Role :=
    "User")> _
    PrivateInfo()

    'Print secret data.
    Console.WriteLine(ControlChars.CrLf + "You have access to the private data!")
End Sub
```

## 5. Crear objetos GenericPrincipal y GenericIdentity

Se puede utilizar la clase [GenericIdentity](#) conjuntamente con la clase [GenericPrincipal](#) para crear un esquema de autorización que sea independiente del dominio de Windows NT o Windows 2000.

### Para crear un objeto GenericPrincipal

1. Cree una instancia nueva de la clase de identidad e inicialícela con el nombre que desee que contenga. El código siguiente crea un objeto **GenericIdentity** nuevo y lo inicializa con el nombre [MyUser](#).

```
Dim MyIdentity As New GenericIdentity("MyUser")
```

2. Cree una instancia nueva de la clase **GenericPrincipal** e inicialícela con el objeto **GenericIdentity** creado anteriormente y una matriz de cadenas que representen las funciones que desee asociar a este principal. En el siguiente ejemplo de código se especifica una matriz de cadenas que representan una función de administrador y una función de usuario. A continuación, la clase **GenericPrincipal** se inicializa con el objeto **GenericIdentity** anterior y la matriz de cadenas.

```
Dim MyStringArray As String() = {"Manager", "Teller"}
Dim MyPrincipal As New GenericPrincipal(MyIdentity, MyStringArray)
```

3. Utilice el código siguiente para asociar el principal al subproceso actual. Este método es muy útil en situaciones en que la entidad de seguridad debe validarse varias veces, debe validarse en otro código que se ejecuta en la aplicación o debe validarse mediante un objeto [PrincipalPermission](#). Se puede seguir realizando la validación basada en funciones en el objeto principal sin asociarlo al subproceso.

```
Thread.CurrentPrincipal = MyPrincipal
```

```
Imports System
Imports System.Security.Principal
Imports System.Threading
```

```
Public Class Class1
```

```
Public Shared Sub Main()
    ' Crear generic identity.
    Dim MyIdentity As New GenericIdentity("MyIdentity")

    ' Crear generic principal.
    Dim MyStringArray As String() = {"Manager", "Teller"}
    Dim MyPrincipal As New GenericPrincipal(MyIdentity, MyStringArray)

    'Enlazar principal al thread actual
    'Esto no es requerido a menos que deba haber una validación repetida
    ' otro código debería validarlo o el objeto PrincipalPermission object sea usado.
    Thread.CurrentPrincipal = MyPrincipal

    ' Mostrar valores en la consola
    Dim Name As String = MyPrincipal.Identity.Name
```

```
Dim Auth As Boolean = MyPrincipal.Identity.IsAuthenticated  
Dim IsInRole As Boolean = MyPrincipal.IsInRole("Manager")
```

```
Console.WriteLine("The Name is: {0}", Name)  
Console.WriteLine("The IsAuthenticated is: {0}", Auth)  
Console.WriteLine("Is this a Manager? {0}", IsInRole)
```

End Sub

End Class

### 3. Servicios criptográficos

Las redes públicas como Internet no proporcionan un medio de comunicación segura entre entidades. La comunicación en esas redes es susceptible de que terceras personas, sin autorización, tengan acceso a ella o la modifiquen. Además de permitir el cifrado de los archivos de los discos locales, la criptografía ayuda a crear medios de comunicación seguros sobre canales que, de otro modo, serían inseguros, proporcionando a su vez integridad de datos y autenticación.

La criptografía ayuda a proteger los datos para que no puedan ser vistos, proporciona mecanismos para la detección de datos modificados y facilita un medio de comunicación seguro en canales que, de otra forma, no serían seguros. Por ejemplo, los datos pueden cifrarse con un algoritmo criptográfico y transmitirse en un estado cifrado a una tercera persona, que posteriormente los descifrará. Si un tercero intercepta los datos cifrados, le resultará difícil descifrarlos.

Para alcanzar estos objetivos, se puede usar una combinación de algoritmos y prácticas conocidas como primitivas criptográficas para crear un esquema criptográfico. En la tabla siguiente se enumeran las primitivas criptográficas y su uso.

Primitiva criptográfica	Uso
Cifrado de clave secreta (criptografía simétrica)	Realiza la transformación de los datos para impedir que terceros los lean. Este tipo de cifrado utiliza una clave secreta compartida para cifrar y descifrar los datos.
Cifrado de clave pública (criptografía asimétrica)	Realiza la transformación de los datos para impedir que terceros los lean. Este tipo de cifrado utiliza un par de claves pública y privada para cifrar y descifrar los datos.
Firmas criptográficas	Ayuda a comprobar que los datos se originan en una parte específica mediante la creación de una firma digital única para esa parte. En este proceso también se usan funciones hash.
Valores hash criptográficos	Asigna datos de cualquier longitud a una secuencia de bytes de longitud fija. Los valores hash son únicos estadísticamente; el valor hash de una secuencia de dos bytes distinta no será el mismo.

.NET Framework proporciona las siguientes clases que implementan algoritmos de cifrado de clave pública:

- [DSACryptoServiceProvider](#)
- [RSACryptoServiceProvider](#)
- [ECDiffieHellman](#) (clase base)
- [ECDiffieHellmanCng](#)
- [ECDiffieHellmanCngPublicKey](#) (clase base)
- [ECDiffieHellmanKeyDerivationFunction](#) (clase base))
- [ECDsaCng](#)

.NET Framework proporciona las siguientes clases que implementan algoritmos de firma digital:

- [DSACryptoServiceProvider](#)
- [RSACryptoServiceProvider](#)
- [ECDsa](#) (clase base)
- [ECDsaCng](#)

.NET Framework proporciona las siguientes clases que implementan algoritmos de firma digital:

- [HMACSHA1](#).
- [MACTripleDES](#).
- [MD5CryptoServiceProvider](#).
- [RIPEMD160](#).
- [SHA1Managed](#).
- [SHA256Managed](#).
- [SHA384Managed](#).
- [SHA512Managed](#).

.NET Framework proporciona las siguientes clases que implementan algoritmos de firma digital:

- [HMACSHA1](#).
- [MACTripleDES](#).
- [MD5CryptoServiceProvider](#).
- [RIPEMD160](#).
- [SHA1Managed](#).
- [SHA256Managed](#).
- [SHA384Managed](#).
- [SHA512Managed](#).
- Variaciones HMAC de todos los algoritmos SHA (algoritmo hash seguro),



## Claves simétricas

Las clases de cifrado simétrico que proporciona .NET Framework requieren una clave y un nuevo vector de inicialización (IV) para cifrar y descifrar datos.

En el ejemplo siguiente se muestra la creación de una nueva instancia de la clase [TripleDESCryptoServiceProvider](#) que implementa el algoritmo TripleDES.

```
Dim TDES as TripleDESCryptoServiceProvider = new _  
TripleDESCryptoServiceProvider ()
```

Cuando se ejecuta el código anterior, se genera una nueva clave y un IV, y se colocan en las propiedades **Key** e **IV**, respectivamente.

En ocasiones tendrá que generar varias claves. En este caso, puede crear una nueva instancia de una clase que implemente un algoritmo simétrico y después crear una nueva clave e IV mediante una llamada a los métodos **GenerateKey** y **GenerateIV**. En el ejemplo de código siguiente se ilustra cómo crear nuevas claves y vectores de inicialización una vez creada una nueva instancia de la clase criptográfica asimétrica.

```
Dim TDES As TripleDESCryptoServiceProvider = new _  
TripleDESCryptoServiceProvider()  
TDES.GenerateIV ()  
TDES.GenerateKey ()
```

Cuando se ejecuta el código anterior, se generan una clave y un IV al crear la nueva instancia de **TripleDESCryptoServiceProvider**. Se crean otra clave e IV cuando se llama a los métodos **GenerateKey** y **GenerateIV**.

## Claves asimétricas

.NET Framework proporciona las clases [RSACryptoServiceProvider](#) y [DSACryptoServiceProvider](#) para el cifrado asimétrico. Estas clases crean un par de claves pública y privada cuando utiliza el constructor predeterminado para crear una nueva instancia. Las claves asimétricas pueden almacenarse para utilizarse en sesiones múltiples o bien generarse para una sesión únicamente. Aunque la clave pública puede ponerse a disposición general, la clave privada debe guardarse bien.

En el ejemplo de código siguiente se crea una nueva instancia de la clase **RSACryptoServiceProvider**, se crea un par de claves pública y privada, y se guarda la información de la clave pública en una estructura **RSAParameters**.

'Generar un par de claves public/private

```
Dim RSA as RSACryptoServiceProvider = new RSACryptoServiceProvider()
```

'Guardar la clave public key de información en una estructura RSAParameters

```
Dim RSAKeyInfo As RSAParameters = RSA.ExportParameters(false)
```

## 4. AppDomain (Clase)

Representa un dominio de aplicación, que es un entorno aislado donde se ejecutan las aplicaciones. Esta clase no se puede heredar.

Los dominios de aplicación, representados por los objetos **AppDomain**, ayudan a proporcionar aislamiento, descarga y límites de seguridad para ejecutar el código administrado.

- Utilice los dominios de aplicación para aislar tareas que podrían derrumbar un proceso. Si el estado del objeto **AppDomain** que está ejecutando una tarea se vuelve inestable, el objeto **AppDomain** se puede descargar sin afectar al proceso. Es importante cuando un proceso debe ejecutarse durante largos períodos sin reiniciar. También puede utilizar los dominios de aplicación para aislar tareas que no deberían compartir datos.
- Si se carga un ensamblado en el dominio de aplicación predeterminado, no se podrá descargar de la memoria mientras el proceso se esté ejecutando. Sin embargo, si se abre un segundo dominio de aplicación para cargar y ejecutar el ensamblado, éste se descarga cuando se descarga ese dominio de aplicación. Utilice esta técnica para minimizar el espacio de trabajo de los procesos de ejecución larga que en ocasiones utilizan grandes archivos DLL.

Los dominios de aplicación se crean mediante el método [CreateDomain](#). Las instancias de **AppDomain** se utilizan para cargar y ejecutar ensamblados ([Assembly](#)). Cuando un **AppDomain** deja de utilizarse, puede descargarse.

La clase **AppDomain** implementa un conjunto de eventos que permiten a las aplicaciones responder cuando se cargue un ensamblado, se descargue un dominio de aplicación o se produce una excepción no controlada.

En el siguiente código, se carga un ensamblado denominado "example.exe" o "example.dll" en el dominio de aplicación actual, se obtiene un tipo denominado `Example` del ensamblado, se obtiene un método sin parámetros denominado `MethodA` para ese tipo y se ejecuta el método. Si desea una descripción completa de cómo obtener información de un ensamblado cargado, vea [Cargar y utilizar tipos dinámicamente](#).

## LABORATORIO 12.1

Cree una aplicación Windows que permita consultar los permisos y roles del usuario actual.

1. Ingrese a Visual Studio 2008
2. Seleccione Menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Application" y en Name escriba "Laboratorio12\_1"
4. Diseñe la siguiente GUI.



5. Agregue el código fuente que corresponde a cada botón.

Imports System.Security.Principal

Public Class Form1

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

'windowsIdentity: representa a un usuario de windows. Se obtiene el usuario actual

Dim objidentidadActual As WindowsIdentity

objidentidadActual = WindowsIdentity.GetCurrent()

'se muestra informacion del ususario

MessageBox.Show("Nombre de usuario: " + objidentidadActual.Name)

MessageBox.Show("ID Token del Usuario: " + objidentidadActual.Token.ToString)

MessageBox.Show("Tipo de Autenticacion: " + objidentidadActual.AuthenticationType)

'Se muestra informacion basada en valores booleanos

If objidentidadActual.IsAnonymous = True Then

MessageBox.Show("Es un usuario anónimo")

End If

If objidentidadActual.IsAuthenticated = True Then

MessageBox.Show("Es un usuario autenticado")

```

End If

If objidentidadActual.IsSystem = True Then
    MessageBox.Show("Es parte del sistema")
End If

If objidentidadActual.IsGuest = True Then
    MessageBox.Show("Es un invitado Guest")
End If

End Sub

Private Sub btnRoles_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnRoles.Click

    'Se obtiene el usuario actual

    Dim objUserActual As WindowsIdentity = WindowsIdentity.GetCurrent ()

    'WindowsPrincipal: Permite que se compruebe la condicion
    'de pertenencia de un usuario de windows a un grupo de windows

    Dim currentPrincipal As WindowsPrincipal = New
    WindowsPrincipal(objUserActual)

    MessageBox.Show("El usuario actual es un miembro de los siguientes grupos: " )

    'se verifica los 3 grupos a los que pertenece

    If currentPrincipal.IsInRole(WindowsBuiltInRole.Administrator) Then
        MessageBox.Show(WindowsBuiltInRole.Administrator.ToString)
    End If

    If currentPrincipal.IsInRole(WindowsBuiltInRole.PowerUser) Then
        MessageBox.Show(WindowsBuiltInRole.PowerUser.ToString)
    End If

    If currentPrincipal.IsInRole(WindowsBuiltInRole.User) Then
        MessageBox.Show(WindowsBuiltInRole.User.ToString)
    End If

End Sub

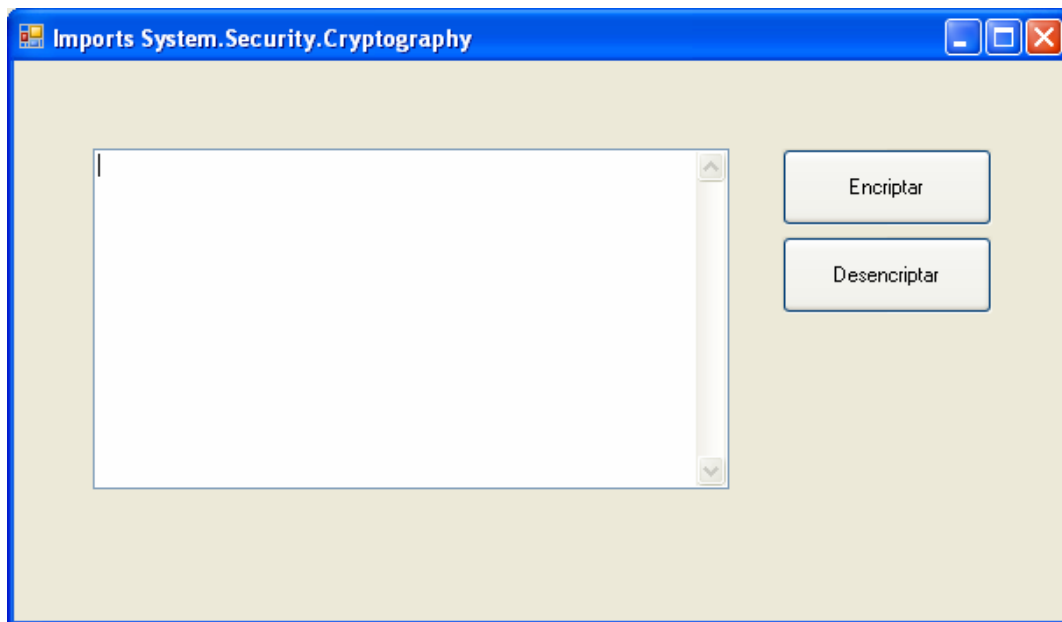
End Class

```

## LABORATORIO 12.2

Cree una aplicación windows para evaluar el resultado de encriptar un archivo de texto. El Algoritmo a utilizar será ***SymmetricAlgorithm***

1. Ingrese a Visual Studio 2008
2. Seleccione Menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Application" y en Name escriba "Laboratorio12\_2"
4. Diseñe la siguiente GUI.



5. Complete el código fuente en el botón que le corresponde

```
Imports System.Security.Cryptography
```

```
Imports System.IO
```

```
Public Class Form1
```

```
Private Sub btnEncriptar_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles btnEncriptar.Click
```

```
    Dim ingreso As String = "C:\Persona.txt"
```

```
    Dim salida As String = "C:\Persona.enc"
```

```
    'Paso 1: se crea el Objeto Stream
```

```
    Dim inFile As FileStream = New FileStream (ingreso, FileMode.Open,  
FileAccess.Read)
```

```

Dim outFile As FileStream = New FileStream(salida, FileMode.OpenOrCreate,
FileAccess.Write)

'se crea un password
Dim password As String = "clave"

'Paso 2: crea el objeto SymmetricAlgorithm y llamas al administrador del algoritmo
'de encriptacion
' crea un objeto cifrador rhijndael simetrico
Dim myAlg As SymmetricAlgorithm = New RijndaelManaged()

'Paso 3: Especificamos la llave
Dim salt As Byte() = System.Text.Encoding.ASCII.GetBytes("This is my sa1t")
Dim key As Rfc2898DeriveBytes = New Rfc2898DeriveBytes(password, salt)
myAlg.Key = key.GetBytes(myAlg.KeySize / 8)
myAlg.IV = key.GetBytes(myAlg.BlockSize / 8)

'Lea el archivo no encriptado dentro de fileData
Dim fileData(inFile.Length - 1) As Byte
inFile.Read(fileData, 0, CType(inFile.Length, Integer))

'Paso 4: crea el objeto ICryptoTransform
'define las operaciones basicas de las transformaiones cryptograficas
Dim encryptor As ICryptoTransform = myAlg.CreateEncryptor

'Paso 5: crea el objeto CryptoStream
Dim encryptStream As CryptoStream = _
New CryptoStream(outFile, encryptor, CryptoStreamMode.Write)

'Paso 6: Escribe el contenido dentro del CryptoStream
encryptStream.Write(fileData, 0, fileData.Length)

'cierra el archivo
encryptStream.Close()
inFile.Close()
outFile.Close()

MessageBox.Show("Mensaje Terminado")

End Sub

```

**Private Sub** btnDesencriptar\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnDesencriptar.Click

Dim lectura As String = "C:\Persona.enc"

Dim nuevo As String = "C:\NuevaPersona.txt"

Dim password As String = "clave"

Dim saltValueBytes As Byte() = System.Text.Encoding.ASCII.GetBytes( "This is my sa1t" )

Dim passwordKey As Rfc2898DeriveBytes = New Rfc2898DeriveBytes(password, saltValueBytes)

' Crear el algoritmo y especificar la clave y el IV

Dim myAlg As RijndaelManaged = New RijndaelManaged

myAlg.Key = passwordKey.GetBytes(myAlg.KeySize / 8)

myAlg.IV = passwordKey.GetBytes(myAlg.BlockSize / 8)

' Leer el archive encriptado dentro de fileData

Dim decryptor As ICryptoTransform = myAlg.CreateDecryptor

Dim inFile As FileStream = New FileStream(lectura, FileMode.Open, FileAccess.Read)

Dim decryptStream As CryptoStream = New CryptoStream(inFile, decryptor, CryptoStreamMode.Read)

Dim fileData(inFile.Length) As Byte

decryptStream.Read(fileData, 0, inFile.Length)

' Escribir el contenido del archive desencriptado

Dim outFile As FileStream = New FileStream(nuevo, FileMode.OpenOrCreate, FileAccess.Write)

outFile.Write(fileData, 0, fileData.Length)

' Cerrar objetos que intervienen en la desencriptación

decryptStream.Close()

inFile.Close()

outFile.Close()

MessageBox.Show("Mensaje Decodificado")

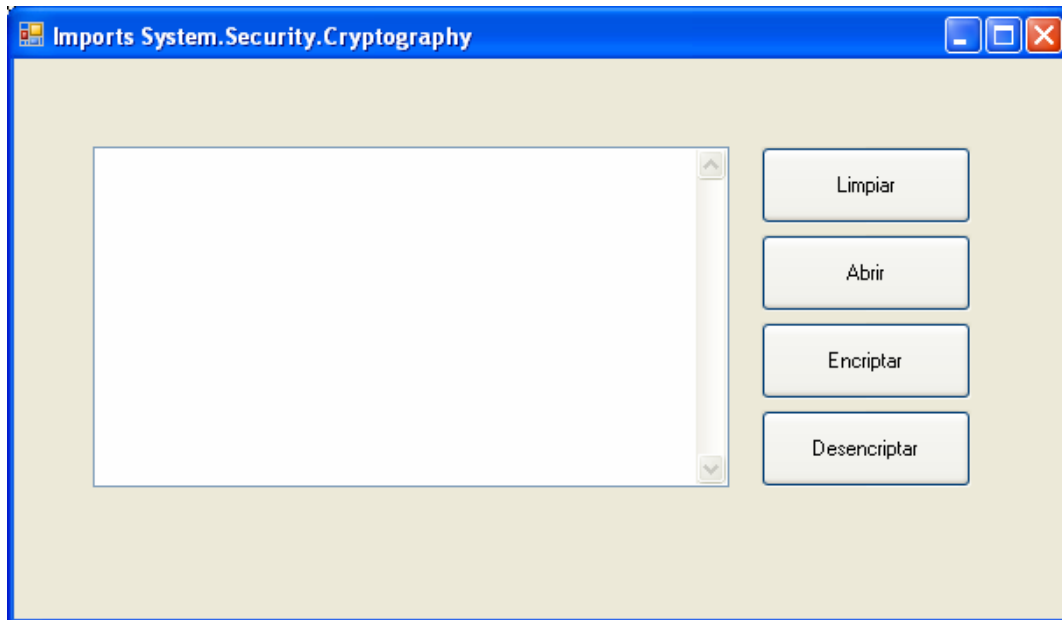
**End Sub**

**End Class**

## LABORATORIO 12.3

Cree una aplicación Windows que permita encriptar datos de un archivo para luego desencriptarlo. El Algoritmo a utilizar será **DESCryptoServiceProvider**

1. Ingrese a Visual Studio 2008
2. Seleccione Menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Application" y en Name escriba "Laboratorio12\_3"
4. Diseñe la siguiente GUI.



5. Complete el código fuente para cada botón y defina las variables globales correspondientes.

`Imports System.IO`

`Imports System.Security.Cryptography`

`Public Class Form2`

`'Proveedor para encriptar, su llave y vector inicial`

`Dim DES As New DESCryptoServiceProvider`

`Dim key() As Byte = DES.Key`

`Dim v() As Byte = DES.IV`

`Private Sub btnLimpiar_Click( ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnLimpiar.Click`

`Me.TextBox1.Text = ""`



End Sub

```
Private Sub btnAbrir_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles btnAbrir.Click
```

```
    'abrir un archivo y leer su contenido
```

```
    Dim op As New OpenFileDialog
```

```
    op.Filter = "Archivo de Texto|*.txt"
```

```
    If op.ShowDialog = Windows.Forms.DialogResult.OK Then
```

```
        'definir un Lector
```

```
        Dim SR As New StreamReader(op.FileName)
```

```
        'leer su contenido
```

```
        Me.TextBox1.Text = SR.ReadToEnd
```

```
        'cerrar el Reader
```

```
        SR.Close()
```

```
    End If
```

End Sub

```
Private Sub btnCodificar_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles btnCodificar.Click
```

```
    Dim sv As New SaveFileDialog
```

```
    sv.Filter = "Archivo de Texto|*.txt"
```

```
    If sv.ShowDialog = Windows.Forms.DialogResult.OK Then
```

```
        '1.convertir el texto en una secuencia de bytes
```

```
        Dim Ms As New MemoryStream
```

```
        Dim SW As New StreamWriter(Ms)
```

```
        SW.Write(TextBox1.Text) 'escribir
```

```
        SW.Flush() 'liberar el escritor, el Memory leera contenido
```

```
        'convertir los datos almacenados en secuencia de bytes
```

```
        Dim data() As Byte = Ms.ToArray
```

```
        '2.Codificar y almacenar
```

```
        Dim F As New FileStream(sv.FileName, FileMode.Create)
```

```
        'definir el CryptoStream para codificar
```

```

        Dim CS As New CryptoStream(F, DES.CreateEncryptor(key, v),
CryptoStreamMode.Write)

        'codificar

        CS.Write(data, 0, data.Length)

        F.Close() 'cerrar el archivo

    End If

End Sub

```

```

Private Sub btnDecodificar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnDecodificar.Click

```

```

    'Abrir el archivo encriptado

    Dim op As New OpenFileDialog
    op.Filter = "Archivo de Texto|*.txt"
    If op.ShowDialog = Windows.Forms.DialogResult.OK Then

        '1.Abrir el Archivo y convertirlo en secuencia de bytes
        Dim F As New FileStream(op.FileName, FileMode.Open)
        Dim data(F.Length) As Byte
        F.Read(data, 0, data.Length)
        F.Close()

        '2.Decodificar almacenando en un MemoryStream
        Dim Ms As New MemoryStream
        'definir el crypto para decodificar
        Dim CS As New CryptoStream(Ms, _
DES.CreateDecryptor(key, v), CryptoStreamMode.Write)
        'escribir para decodificar data
        CS.Write(data, 0, data.Length)

        'Mostrar el contenido decodificado
        Ms.Position = 0 'ubicar en la posicion inicial
        Dim SR As New StreamReader(Ms) 'lector al Ms
        'mostrar el contenido
        MessageBox.Show (SR.ReadToEnd)
        SR.Close() 'cerrar

    End If

End Sub

End Class

```

## Autoevaluación

1. ¿Qué tipos de seguridad se pueden utilizar para aplicaciones windows?

---

---

2. ¿Qué namespace define un objeto principal que representa el contexto de seguridad?

---

---

---

3. ¿Qué representa un objeto Principal?

---

---

---

4. ¿Qué representa un objeto Identity?

---

---

---

## Resumen

- La seguridad basada en funciones de .NET Framework admite la autorización poniendo a disposición del subproceso actual información relativa al principal, que se crea a partir de una identidad asociada. La identidad (y el principal que ayuda a definir) puede estar basada en una cuenta de Windows o puede ser una identidad personalizada no relacionada con una cuenta de Windows.
- El objeto de identidad encapsula información relativa al usuario o la entidad que se van a validar.
- El objeto principal representa el contexto de seguridad bajo el cual se ejecuta código. Las aplicaciones que implementan seguridad basada en funciones conceden derechos tomando como base la función asociada a un objeto principal.
- La criptografía ayuda a proteger los datos para que no puedan ser vistos, proporciona mecanismos para la detección de datos modificados y facilita un medio de comunicación seguro en canales que, de otra forma, no serían seguros.
- .NET Framework proporciona implementaciones de numerosos algoritmos criptográficos estándar. Estos algoritmos son fáciles de utilizar y disponen de las propiedades predeterminadas más seguras. Además, el modelo de criptografía de .NET Framework de la herencia de objetos, el diseño de las secuencias y la configuración son muy extensibles.

**Fuente:**

**Texto adaptado de la página web:**

**<http://msdn.microsoft.com/es-es/library/52kd59t0.aspx>**

**UNIDAD DE  
APRENDIZAJE**

**5**

**SEMANA**

**13**

## **Interoperabilidad y Reflection**

---

### **LOGRO DE LA UNIDAD DE APRENDIZAJE**

Al término de la unidad, los alumnos elaborarán aplicaciones de mejor rendimiento y acceso asignando recursos, permisos, componentes, cultura y una adecuada distribución de la misma para su implementación.

### **TEMARIO**

- Exposición de componentes COM al .NET Framework
- Invocación de funciones DLLs no manejadas
- Manejo de Reflection
- Manejo del lenguaje intermedio

### **ACTIVIDADES PROPUESTAS**

- Los alumnos crean clases que pueden interoperar en aplicaciones COM
- Los alumnos crear aplicaciones que usan reflection.

## 1. Interoperabilidad con los Componentes COM

### 1.1. Interoperar con código no administrado

.NET Framework promueve la interacción con los componentes COM, los Servicios COM+, las bibliotecas de tipos externas y muchos servicios del sistema operativo. Los tipos de datos, los prototipos de los métodos y los mecanismos de control de errores no son iguales en los modelos de objetos administrados y no administrados. Para simplificar la interoperación entre los componentes de .NET Framework y el código no administrado y facilitar la ruta de acceso de migración, Common Language Runtime oculta las diferencias que existen entre estos modelos de objetos a los clientes y a los servidores.

El código que se ejecuta bajo el control del motor en tiempo de ejecución se denomina código administrado. Por el contrario, el código que se ejecuta fuera del motor en tiempo de ejecución se denomina código no administrado. Los componentes COM, las interfaces ActiveX y las funciones de la API Win32 son ejemplos de código no administrado.

### 1.2. Aplicar atributos de interoperabilidad

El espacio de nombres [System.Runtime.InteropServices](#) proporciona tres categorías de atributos específicos de la interoperabilidad: los que el usuario aplica en tiempo de diseño, los que las API y las herramientas de interoperabilidad COM aplican durante el proceso de conversión y los que aplica el usuario o la interoperabilidad COM.

Si no está familiarizado con la tarea de aplicar atributos al código administrado, vea [Extender metadatos mediante atributos](#). Del mismo modo que se aplican otros atributos personalizados, los atributos específicos de la interoperabilidad pueden aplicarse a tipos, métodos, propiedades, parámetros, campos y otros miembros.

Atributos en tiempo de diseño

Se puede ajustar el resultado del proceso de conversión realizado por las API y las herramientas de interoperabilidad COM utilizando atributos en tiempo de diseño. En la siguiente tabla se describen los atributos que se pueden aplicar al código fuente administrado. En alguna ocasión, las herramientas de interoperabilidad COM podrían aplicar también los atributos descritos en esta tabla.

Atributo	Descripción
<a href="#">AutomationProxyAttribute</a>	Especifica si las referencias del tipo deben calcularse utilizando el contador de referencias de automatización o un proxy y un código auxiliar personalizados.
<a href="#">ClassInterfaceAttribute</a> <a href="#">ComClassAttribute</a>	Controla el tipo de interfaz generada para una clase. Identifica el CLSID de la coclase original importada de una biblioteca de tipos. Las herramientas de interoperabilidad COM aplican normalmente este atributo.
<a href="#">ComImportAttribute</a>	Indica que una definición de interfaz o coclase se importó desde una biblioteca de tipos COM. El motor en tiempo de ejecución utiliza este indicador para conocer la forma de activar el tipo y de calcular sus referencias. Este atributo impide que el tipo vuelva a exportarse a la biblioteca de tipos. Las herramientas de interoperabilidad COM aplican normalmente este atributo.

ComRegisterFunctionAttribute	Indica que debe llamarse a un método cuando el ensamblado se registra para su uso desde COM, de modo que el código escrito por el usuario pueda ejecutarse durante el proceso de registro.
ComSourceInterfacesAttribute	Identifica las interfaces que son orígenes de eventos para la clase. Las herramientas de interoperabilidad COM pueden aplicar este atributo.
ComUnregisterFunctionAttribute	Indica que debe llamarse a un método cuando se anula el registro del ensamblado desde COM, de modo que el código escrito por el usuario pueda ejecutarse durante el proceso.
ComVisibleAttribute	Hace que los tipos no sean visibles para COM cuando el valor del atributo es <b>false</b> . Este atributo puede aplicarse a un tipo individual o a todo un ensamblado para controlar la visibilidad de COM. De forma predeterminada, todos los tipos públicos administrados son visibles; este atributo no es necesario para hacerlos visibles.
InAttribute	Indica que el cálculo de referencias de los datos debe realizarse en el llamador. Puede utilizarse en parámetros de atributo.
InterfaceTypeAttribute	Controla la forma en que una interfaz administrada se expone en los clientes COM (dual, derivada de IUnknown o sólo IDispatch). Las herramientas de interoperabilidad COM pueden aplicar este atributo.
OptionalAttribute	Indica que un parámetro es opcional. Las herramientas de interoperabilidad COM pueden aplicar este atributo.

### 1.3. System.Runtime.InteropServices.ComTypes (Espacio de nombres)

El espacio de nombres System.Runtime.InteropServices.ComTypes contiene métodos que son definiciones de funciones COM para el código administrado. Estas funciones reemplazan a los métodos UCOM\*, ahora obsoletos, del espacio de nombres

## 2. System.Reflection (Espacio de nombres)

El espacio de nombres System.Reflection contiene clases e interfaces que proporcionan una vista administrada de los campos, los métodos y los tipos cargados, con la posibilidad de crear e invocar tipos dinámicamente.

Además contiene tipos que recuperan información sobre los ensamblados, módulos, miembros, parámetros y otras entidades del código administrado examinando sus metadatos. Estos tipos también se pueden utilizar para manipular instancias de tipos cargados, por ejemplo, para enlazar eventos o llamar a métodos. Para crear tipos dinámicamente, utilice el espacio de nombres System.Reflection.Emit.

## 2.1. Assembly (Clase)

Representa un ensamblado, que es un bloque de creación reutilizable, versionable y autodescriptivo de una aplicación de Common Language Runtime.

Utilice la clase **Assembly** para cargar los ensamblados, explorar los metadatos y las partes constituyentes de los ensamblados, detectar los tipos incluidos en los ensamblados y crear instancias de esos tipos.

Para obtener una matriz de objetos **Assembly** que represente los ensamblados que actualmente están cargados en un dominio de aplicación (por ejemplo, el dominio de aplicación predeterminado de un proyecto simple), utilice el método [AppDomain.GetAssemblies](#).

Para cargar ensamblados dinámicamente, la clase **Assembly** proporciona los métodos estáticos siguientes (métodos Shared en Visual Basic). Los ensamblados se cargan en el dominio de aplicación en que se produce la operación de carga.

- Se recomienda cargar los ensamblados utilizando el método [Load](#), que identifica el ensamblado que se va a cargar mediante su nombre para mostrar (por ejemplo, "System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"). La búsqueda del ensamblado se rige por las normas descritas en [Cómo el motor en tiempo de ejecución ubica ensamblados](#).
- Los métodos [ReflectionOnlyLoad](#) y [ReflectionOnlyLoadFrom](#) permiten cargar un ensamblado para su reflexión, pero no para su ejecución. Por ejemplo, un ensamblado que tiene como destino una plataforma de 64 bits puede ser examinado por el código que se ejecuta en una plataforma de 32 bits.
- Los métodos [LoadFile](#) y [LoadFrom](#) se proporcionan para aquellas situaciones poco habituales en que un ensamblado debe identificarse mediante una ruta de acceso.

Si desea obtener un objeto **Assembly** para el ensamblado que se encuentra actualmente en ejecución, utilice el método [GetExecutingAssembly](#).

Muchos miembros de la clase **Assembly** proporcionan información sobre un ensamblado. Por ejemplo:

- El método [GetName](#) devuelve un objeto [AssemblyName](#) que proporciona acceso a las partes del nombre para mostrar del ensamblado.
- El método [GetCustomAttributes](#) muestra los atributos que se aplican al ensamblado.
- El método [GetFiles](#) proporciona acceso a los archivos del manifiesto del ensamblado.
- El método [GetManifestResourceNames](#) proporciona los nombres de los recursos del manifiesto del ensamblado.

El método [GetTypes](#) muestra todos los tipos del ensamblado. El método [GetExportedTypes](#) indica los tipos que los llamadores pueden ver desde fuera del ensamblado. El método [GetType](#) se puede utilizar para buscar un tipo determinado en el ensamblado. El método [CreateInstance](#) se puede utilizar para buscar y crear instancias de tipos en el ensamblado.



Imports System

Imports System.Reflection

Imports System.Security.Permissions

<assembly: AssemblyVersionAttribute("1.0.2000.0")>

Public Class Example

Private factor As Integer

Public Sub New(ByVal f As Integer)

factor = f

End Sub

Public Function SampleMethod(ByVal x As Integer) As Integer

Console.WriteLine(vbCrLf & "Example.SampleMethod({0}) executes.", x)

Return x \* factor

End Function

Public Shared Sub Main()

Dim assem As [Assembly] = [Assembly].GetExecutingAssembly()

Console.WriteLine("Assembly Full Name:")

Console.WriteLine(assem.FullName)

' The AssemblyName type can be used to parse the full name.

Dim assemName As AssemblyName = assem.GetName()

Console.WriteLine(vbLf + "Name: {0}", assemName.Name)

Console.WriteLine("Version: {0}.{1}", assemName.Version.Major, \_  
assemName.Version.Minor)

Console.WriteLine(vbLf + "Assembly CodeBase:")

Console.WriteLine(assem.CodeBase)

' Create an object from the assembly, passing in the correct number

' and type of arguments for the constructor.

Dim o As Object = assem.CreateInstance("Example", False, \_

BindingFlags.ExactBinding, Nothing, \_

New Object() { 2 }, Nothing, Nothing)

' Make a late-bound call to an instance method of the object.

Dim m As MethodInfo = assem.GetType("Example").GetMethod("SampleMethod")

Dim ret As Object = m.Invoke(o, New Object() { 42 })

Console.WriteLine("SampleMethod returned {0}.", ret)

Console.WriteLine(vbCrLf & "Assembly entry point:")

Console.WriteLine(assem.EntryPoint)

End Sub

End Class

```
' Este código produce una salida similar a lo siguiente:
'
'Assembly Full Name:
'source, Version=1.0.2000.0, Culture=neutral, PublicKeyToken=null
'
'Name: source
'Version: 1.0
'
'Assembly CodeBase:
'file: C:/sdtree/AssemblyClass/vb/source.exe
'
'Example.SampleMethod (42) executes.
'SampleMethod returned 84.
'
'Assembly entry point:
'Void Main()
```

## LABORATORIO 13.1

Cree una aplicación Windows que permita ingresar datos de un objeto Persona y luego se serialice esta información en formato XML.

1. Ingrese a Visual Studio 2008
2. Seleccione Menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Class Library" y en Name escriba "InteropFunciones"
4. Agregar el siguiente código en el archivo de la clase Operación.

'Namespace para manejar la interoperabilidad de la clase Operacion

Imports System.Runtime.InteropServices

'crear una interfase

Public Interface IOperacion

Function Sumar(ByVal a As Integer, ByVal b As Integer) As Integer

Function Restar(ByVal a As Integer, ByVal b As Integer) As Integer

Function CalcularUltimoDiaMes(ByVal mes As String, ByVal anho As String) As Integer

<ComVisible(False)> \_

Function Multiplicar(ByVal a As Integer, ByVal b As Integer) As Integer

End Interface

<ClassInterface(ClassInterfaceType.None)> \_

Public Class Operacion

Implements IOperacion

Public Function Multiplicar(ByVal a As Integer, ByVal b As Integer) As Integer

Implements IOperacion.Multiplicar

Return a \* b

End Function

Public Function Restar(ByVal a As Integer, ByVal b As Integer) As Integer Implements

IOperacion.Restar

Return a - b

End Function

Public Function Sumar(ByVal a As Integer, ByVal b As Integer) As Integer Implements

IOperacion.Sumar

Return a + b

End Function

Public Function CalcularUltimoDiaMes(ByVal mes As String, ByVal anho As String) As

Integer Implements IOperacion.CalcularUltimoDiaMes

Dim strFecha As String

Dim dtFecha As DateTime

strFecha = "01/" + mes + "/" + anho

dtFecha = CDate(strFecha)

dtFecha = DateAdd(DateInterval.Month, 1, dtFecha)

dtFecha = DateAdd(DateInterval.Day, -1, dtFecha)

Return dtFecha.Day

End Function

End Class

## Autoevaluación

1. ¿Qué es Interoperabilidad COM?

---

---

---

---

2. ¿Qué es Reflection?

---

---

---

---

3. ¿Qué representa la clase Assembly?

---

---

---

---

## Para Recordar

- Assembly Representa un ensamblado, que es un bloque de creación reutilizable, versionable y autodescriptivo de una aplicación de Common Language Runtime.
- El espacio de nombres **System.Runtime.InteropServices** proporciona una gran variedad de miembros que admiten la interoperabilidad COM y los servicios de invocación de plataforma.
- La clase ComVisibleAttribute controla la accesibilidad a COM de un tipo o miembro administrado individual o de todos los tipos de un ensamblado.

### Nota:

<http://msdn.microsoft.com/es-es/library/system.runtime.interopservices.comvisibleattribute.aspx>

<http://msdn.microsoft.com/es-es/library/system.reflection.aspx>



**UNIDAD DE  
APRENDIZAJE**

**5**

**SEMANA**

**14**

## **Globalización e Introducción a WPF**

---

### **LOGRO DE LA UNIDAD DE APRENDIZAJE**

Al término de la unidad, los alumnos elaboran aplicaciones Windows .NET que se conectan a un origen de datos utilizando los objetos de ADO.NET para realizar operaciones de consulta y actualización de datos.

### **TEMARIO**

- Manejo de globalización y cultura
- Foramas de efectuar la globalización.
- Introducción a WPF

### **ACTIVIDADES PROPUESTAS**

- Los alumnos crean aplicaciones con etiquetas según la cultura que utiliza la pc.
- Los alumnos crean aplicacones WPF para mejorar las interfaces de usuario.

## 1. Globalizar y localizar aplicaciones

Si piensa distribuir su aplicación a un público internacional, deberá tener en cuenta una serie de puntos durante las fases de diseño y desarrollo. Incluso si no tiene esos planes, un pequeño esfuerzo previo puede facilitar las cosas si cambia de idea en futuras versiones de la aplicación. Los servicios integrados en .NET Framework facilitan el desarrollo de aplicaciones que se adaptan a distintas configuraciones regionales mediante el desarrollo administrado con Visual Studio.

Globalización es el proceso mediante el cual se diseñan y desarrollan productos de software que pueden funcionar para varias referencias culturales. Esta sección se aplica a las páginas de formularios tanto Windows Forms como Web Forms.

### Para establecer opciones de formato apropiadas para una referencia cultural determinada

1. Si desea reemplazar la configuración del usuario o del sistema operativo, establezca las propiedades `CurrentCulture` y `CurrentUICulture`.

Normalmente, desea especificar una referencia cultural para que cada parte de la interfaz de usuario de la aplicación sea adecuada a esa referencia cultural. Por tanto, deberá establecerla antes de que se llame al método **InitializeComponent**.

```
' Put the Imports statements at the beginning of the code module
Imports System.Threading
Imports System.Globalization
' Put the following code before InitializeComponent()
' Sets the culture to French (France)
Thread.CurrentThread.CurrentCulture = new CultureInfo("fr-FR")
' Sets the UI culture to French (France)
Thread.CurrentThread.CurrentUICulture = new CultureInfo("fr-FR")
```

### Nota

El valor de referencia cultural deberá ser siempre específico (como "fr-FR"), no una referencia cultural neutra (como "fr"). La razón es que una referencia cultural como "fr" puede aplicarse a todas las referencias culturales francohablantes, y a las distintas monedas usadas en Francia, Bélgica y Quebec.

2. Invoque métodos de formato con la referencia cultural de todos los idiomas (invariant) para las cadenas que deban aparecer sin cambios, sin importar el valor de la propiedad **CurrentCulture**.

```
Dim MyInt As Integer = 100
Dim MyString As String = MyInt.ToString("C", CultureInfo.InvariantCulture)
MessageBox.Show(MyString)
```



### Thread.CurrentCulture (Propiedad)

Obtiene o establece la referencia cultural del subproceso actual.

### CultureInfo (Clase)

Proporciona información de una referencia cultural concreta, como los nombres de la referencia cultural, el sistema de escritura, el calendario utilizado y cómo se da formato a las fechas y se ordenan las cadenas.

### CultureInfo (Propiedades)

Nombre	Descripción
<b>Calendar</b>	Obtiene el calendario predeterminado utilizado por la referencia cultural.
<b>CompareInfo</b>	Obtiene el <a href="#">CompareInfo</a> que define el modo en que se comparan las cadenas para la referencia cultural.
<b>CultureTypes</b>	Obtiene los tipos de referencia cultural que pertenecen al objeto CultureInfo actual.
<b>CurrentCulture</b>	Obtiene el CultureInfo que representa la referencia cultural utilizada por el subproceso actual.
<b>CurrentUICulture</b>	Obtiene el CultureInfo que representa la referencia cultural actual utilizada por el administrador de recursos que busca recursos específicos de la referencia cultural en tiempo de ejecución.
<b>DateTimeFormat</b>	Obtiene o establece un DateTimeFormatInfo que define el formato de presentación de fechas y horas culturalmente apropiado.
<b>DisplayName</b>	Obtiene el nombre de la referencia cultural en el formato "<idiomacompleto> (<país/regióncompletos>)" en el idioma de la versión traducida de .NET Framework.
<b>EnglishName</b>	Obtiene el nombre de la referencia cultural en el formato "<idiomacompleto> (<país/regióncompletos>)" en inglés.
<b>ietfLanguageTag</b>	Obtiene la identificación del estándar RFC 3066 (bis) para un idioma.
<b>InstalledUICulture</b>	Obtiene el CultureInfo que representa la referencia cultural instalada con el sistema operativo.
<b>InvariantCulture</b>	Obtiene el CultureInfo que es independiente de la referencia cultural (invariable).
<b>IsNeutralCulture</b>	Obtiene un valor que indica si el CultureInfo actual representa una referencia cultural neutra.
<b>IsReadOnly</b>	Obtiene un valor que indica si el CultureInfo actual es de sólo lectura.
<b>KeyboardLayoutId</b>	Obtiene el identificador de configuración regional de entrada activo.
<b>LCID</b>	Obtiene el identificador de referencia cultural del CultureInfo actual.
<b>Name</b>	Obtiene el nombre de la referencia cultural en el formato "<códigoidioma2>-<códigopaís/región2>".
<b>NativeName</b>	Obtiene el nombre de la referencia cultural en el formato "<idiomacompleto> (<país/regióncompletos>)" en el

	idioma de presentación en el que está establecida la referencia cultural.
<b>NumberFormat</b>	Obtiene o establece un NumberFormatInfo que define el formato de presentación de números, moneda y porcentaje culturalmente apropiado.
<b>TextInfo</b>	Obtiene el TextInfo que define el sistema de escritura asociado a la referencia cultural.

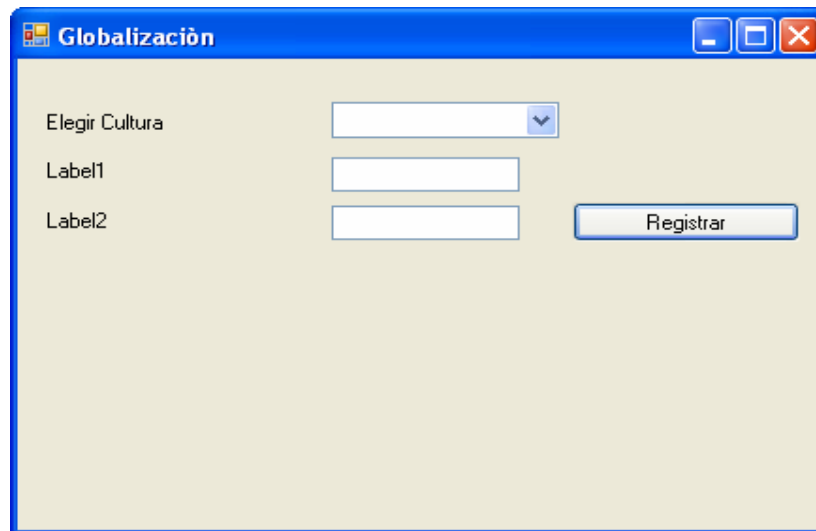
A diferencia de [CultureInfo](#), **RegionInfo** no representa las preferencias del usuario y no depende del idioma ni de la referencia cultural del usuario.

El nombre de **RegionInfo** es uno de los códigos de dos letras definidos en ISO 3166 para el país o la región. La distinción entre mayúsculas y minúsculas no es relevante; sin embargo, las propiedades [Name](#), [TwoLetterISORegionName](#) y [ThreeLetterISORegionName](#) devuelven el código correspondiente en mayúsculas.

## LABORATORIO 14.1

Cree una aplicación Windows que permita ingresar datos en un formulario que muestra sus etiquetas de acuerdo al lenguaje elegido desde un combobox.

1. Ingrese a Visual Studio 2008
2. Seleccione Menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Application" y en Name escriba "Laboratorio10\_2"
4. Diseñe la siguiente GUI.



5. Agregar el siguiente código fuente para el formulario desde la vista código.

'Namespace para el manejo de cultura o region

Imports System.Globalization

'Namespace para usar archivos de recursos

Imports System.Resources

'Namespace para usar threads

Imports System.Threading

Public Class Form1

'Declarar variable de tipo ResourceManager para usar archivo de recursos

Dim rm As ResourceManager

'Declarar variable CultureInfo para trabajar con una cultura especifica

Dim cult As CultureInfo

Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles MyBase.Load

'Cargar los archivos de recursos

rm = ResourceManager.CreateFileBasedResourceManager \_  
("Strings", "C:\", Nothing)

End Sub

Private Sub btnRegistrar\_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles btnRegistrar.Click

lblTitulo.Visible = True

lblNombres.Visible = True

lblApellidos.Visible = True

lblTitulo.Text = rm.GetString("Sal", cult)

lblNombres.Text = rm.GetString("SalNom", cult) + " " + txtNombre.Text.Trim

lblApellidos.Text = rm.GetString("SalApe", cult) + " " + txtApellido.Text

End Sub

Private Sub cboCulturas\_SelectedIndexChanged(ByVal sender As Object, ByVal e As  
System.EventArgs) Handles cboCulturas.SelectedIndexChanged

'Instanciar una nueva cultura de acuerdo a lo

'seleccionado en el combo

cult = New CultureInfo(cboCulturas.Text.Trim)

'cambiar la cultura actual por la seleccionada en el Combobox

```
Thread.CurrentThread.CurrentCulture = cult
```

```
Thread.CurrentThread.CurrentUICulture = cult
```

```
'Mostrar el atributo Nom que se encuentra en el archivo
```

```
'de recursos de la cultura elegida
```

```
lblPregunta1.Text = rm.GetString("Nom", cult)
```

```
lblPregunta2.Text = rm.GetString("Ape", cult)
```

```
lblTitulo.Visible = False
```

```
lblNombres.Visible = False
```

```
lblApellidos.Visible = False
```

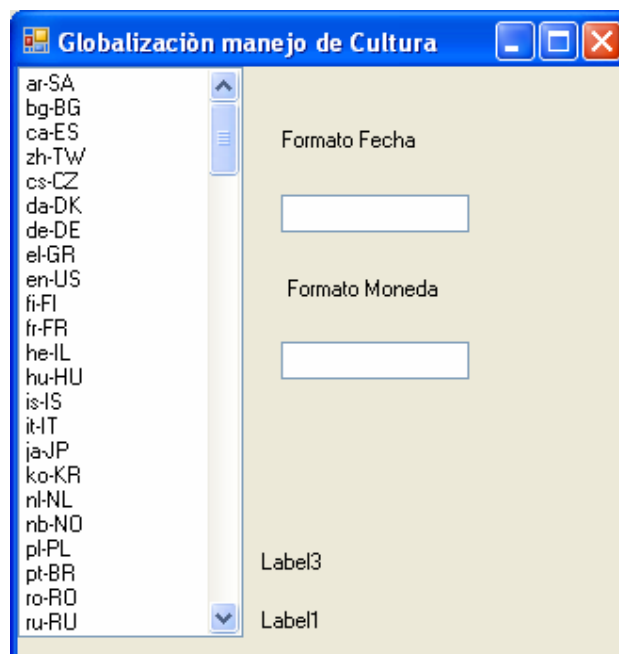
```
End Sub
```

```
End Class
```

## LABORATORIO 14.2

Cree una aplicación Windows que permita ingresar datos como fecha y moneda, de esta manera seleccionando una cultura específica se cambiarán estos datos al formato fecha y moneda de la cultura seleccionada.

1. Ingrese a Visual Studio 2008
2. Seleccione Menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "Windows Application" y en Name escriba "Laboratorio14\_2"
4. Diseñe la siguiente GUI.



5. Agregue el siguiente código fuente

'Namespace para trabajar con la clase CulterelInfo

Imports System.Globalization

Public Class Form2

'crear método q cargue las culturas actuales

Sub MostrarCulturas()

For Each cultura As CulterelInfo In \_

CulterelInfo.GetCultures(CultureTypes.SpecificCultures)

'listar cada nombre de cultura en el listBox

ListBox1.Items.Add(cultura.Name)

Next

End Sub

Private Sub Form2\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

MostrarCulturas()

End Sub

Private Sub ListBox1\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ListBox1.SelectedIndexChanged

'recuperar la cultura seleccionada

Dim oCulture As String = ListBox1.SelectedItem.ToString

'crear un objeto CulterelInfo con la cultura seleccionada

Dim cu As New CulterelInfo(oCulture)

Dim nValor As Double = 7814.86

'aplicar el formato correspondiente en base a la cultura

'para la fecha

txtFormatoFecha.Text = Now.ToString("d", cu)

'y para la moneda

txtFormatoMoneda.Text = nValor.ToString("c", cu)

'mostramos el nombre en español de la cultura.

lblCultura.Text = cu.DisplayName

'mostrar pais de la cultura

```
Dim region As New RegionInfo(oCulture)
```

```
lblPais.text = region.DisplayName
```

```
End Sub
```

```
End Class
```

## 2 Introducción a WPF

Windows Presentation Foundation (WPF) proporciona a los programadores un modelo de programación unificado con el que generar experiencias de cliente inteligentes de Windows, en las que se incorpora la interfaz de usuario, multimedia y documentos.

La plataforma de desarrollo WPF se ha generado sobre un sistema de programación básico, que se extiende para admitir un amplio conjunto de características de desarrollo de aplicaciones que incluyen el propio modelo de aplicación, recursos, controles, gráficos, diseño, enlace de datos, documentos y seguridad.

Windows Presentation Foundation (WPF) es un sistema de presentación de la próxima generación, para crear aplicaciones cliente de Windows que proporcionen una experiencia impactante para el usuario desde el punto de vista visual. Con WPF, puede crear una amplia gama de aplicaciones independientes y hospedadas en explorador. Algunos ejemplos de ello son Yahoo! Messenger y New York Times Reader, así como la aplicación de ejemplo para atención sanitaria, Contoso Healthcare Sample Application que se muestra en la ilustración siguiente.



El núcleo de WPF es un motor de representación independiente de la resolución y basado en vectores construido para aprovechar al máximo el hardware de gráficos moderno. WPF amplía el núcleo con un completo conjunto de características de

programación de aplicaciones, entre las que se incluyen Lenguaje de marcado de aplicaciones extensible (XAML), controles, enlace de datos, diseño, gráficos 2D y 3D, animación, estilos, plantillas, documentos, multimedia, texto y tipografía. WPF se incluye en Microsoft .NET Framework, lo que permite crear aplicaciones que incorporen otros elementos de la biblioteca de clases de .NET Framework.

Una mejora evidente de WPF es la capacidad para programar una aplicación mediante código de lenguaje marcado y subyacente, una experiencia con la que resultará familiar a los programadores de ASP.NET. En general, se utiliza el lenguaje marcado Lenguaje de marcado de aplicaciones extensible (XAML) para implementar la apariencia de una aplicación, y los lenguajes de programación administrados (subyacentes) para implementar su comportamiento. Esta separación entre la apariencia y el comportamiento aporta las ventajas siguientes:

- Se reducen los costos de programación y mantenimiento, al no estar el marcado específico de la apariencia estrechamente relacionado con el código específico del comportamiento.
- La programación es más eficaz porque los diseñadores pueden implementar la apariencia de una aplicación al mismo tiempo que los programadores implementan su comportamiento.
- Se pueden utilizar varias herramientas de diseño para implementar y compartir el lenguaje marcado XAML.
- La globalización y localización de las aplicaciones WPF se ha simplificado en gran medida

### **Marcado**

XAML es un lenguaje de marcado basado en XML que se utiliza para implementar la apariencia de una aplicación mediante declaración. Se suele utilizar para crear ventanas, cuadros de diálogo, páginas y controles de usuario, así como para rellenarlos con controles, formas y gráficos.

En el ejemplo siguiente se utiliza XAML para implementar la apariencia de una ventana que contiene un solo botón.

```
<Window
```

```
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
  x:Class="SDKSample.AWindow"
```

```
Title="Window with Button"

Width="250" Height="100">

<!-- Add button to window -->

<Button Name="button" Click="button_Click">Click Me!</Button>

</Window>
```

El código en visual Basic.

Namespace SDKSample

Partial Public Class AWindow

Inherits System.Windows.Window

Public Sub New()

' InitializeComponent call is required to merge the UI

' that is defined in markup with this class, including

' setting properties and registering event handlers

InitializeComponent()

End Sub

Private Sub button\_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)

' Muestra mensaje

MessageBox.Show("Hello, Windows Presentation Foundation!")

End Sub

End Class

End Namespace

En este ejemplo, el código subyacente implementa una clase que se deriva de la clase Window.

El atributo x:Class se utiliza para asociar el marcado a la clase de código subyacente. Se llama a InitializeComponent desde el constructor de la clase de código subyacente para combinar la interfaz de usuario definida en el marcado con la clase de código subyacente.



La combinación de `x: Class` e `InitializeComponent` garantiza que la implementación se inicializa correctamente cada vez que se crea.

En la ilustración siguiente se muestra el resultado de hacer clic en el botón.



### Gráficos

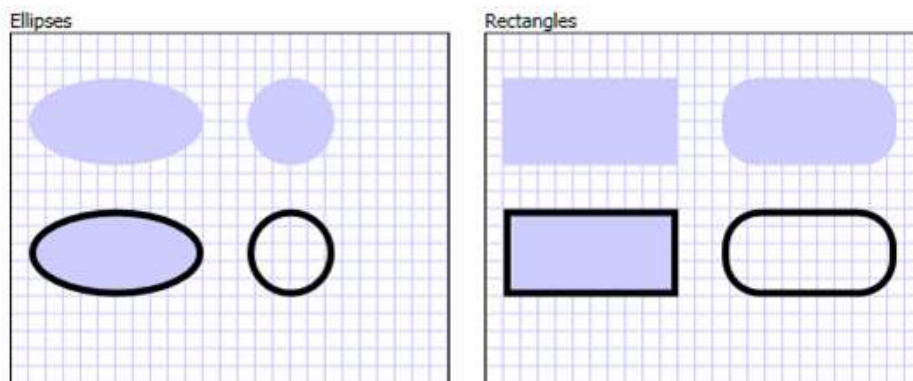
WPF presenta un conjunto extenso, escalable y flexible de características de gráficos que aportan las ventajas siguientes:

- **Gráficos independientes de la resolución e independientes del dispositivo.** La unidad de medida básica del sistema de gráficos de WPF es el píxel independiente del dispositivo, que es 1/96 de pulgada, independientemente de la resolución de pantalla real, y que proporciona la base para la representación independiente de la resolución y del dispositivo. Cada píxel independiente del dispositivo se escala automáticamente para coincidir con el valor de puntos por pulgada (ppp) del sistema en que se representa.
- **Precisión mejorada.** El sistema de coordenadas de WPF se mide con números de punto flotante de precisión doble, en lugar de precisión simple. Las transformaciones y los valores de opacidad también se expresan como doble precisión. WPF admite también una amplia gama de colores (sRGB) y proporciona compatibilidad integrada para administrar las entradas desde espacios de color diferentes.
- **Compatibilidad con gráficos avanzados y animación.** WPF simplifica la programación de gráficos administrando automáticamente las escenas de animación; no tendrá que preocuparse por el procesamiento de escenas, los bucles de representación ni la interpolación bilineal. Además, WPF admite la comprobación de clics y proporciona compatibilidad plena con la composición alfa.

- **Aceleración de hardware.** El sistema de gráficos de WPF saca partido del hardware de gráficos para minimizar el uso de la CPU.

## Formas 2D

WPF proporciona una biblioteca de formas 2D comunes dibujadas mediante vectores, como los rectángulos y las elipses que se muestran en la ilustración siguiente.

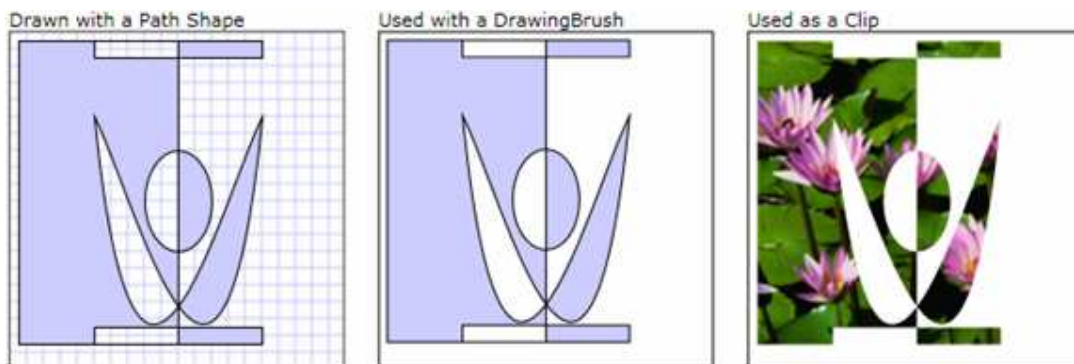


## Geometrías 2D

Las formas 2D proporcionadas por WPF abarcan el conjunto estándar de formas básicas. Sin embargo, puede que sea preciso crear formas personalizadas para facilitar el diseño de una interfaz de usuario personalizada. Para este fin, WPF proporciona las geometrías. En la ilustración siguiente se muestra el uso de geometrías para crear una forma personalizada que se puede dibujar directamente, utilizar como un pincel o utilizar para recortar otras formas y controles.

Los objetos [Path](#) se pueden utilizar para dibujar formas cerradas o abiertas, varias formas o incluso formas curvas.

Los objetos [Geometry](#) se pueden utilizar para el recorte, la comprobación de visitas y la representación de datos de gráficos 2D.



## Lista controles de WPF

A continuación se muestra la lista de controles de WPF integrados.

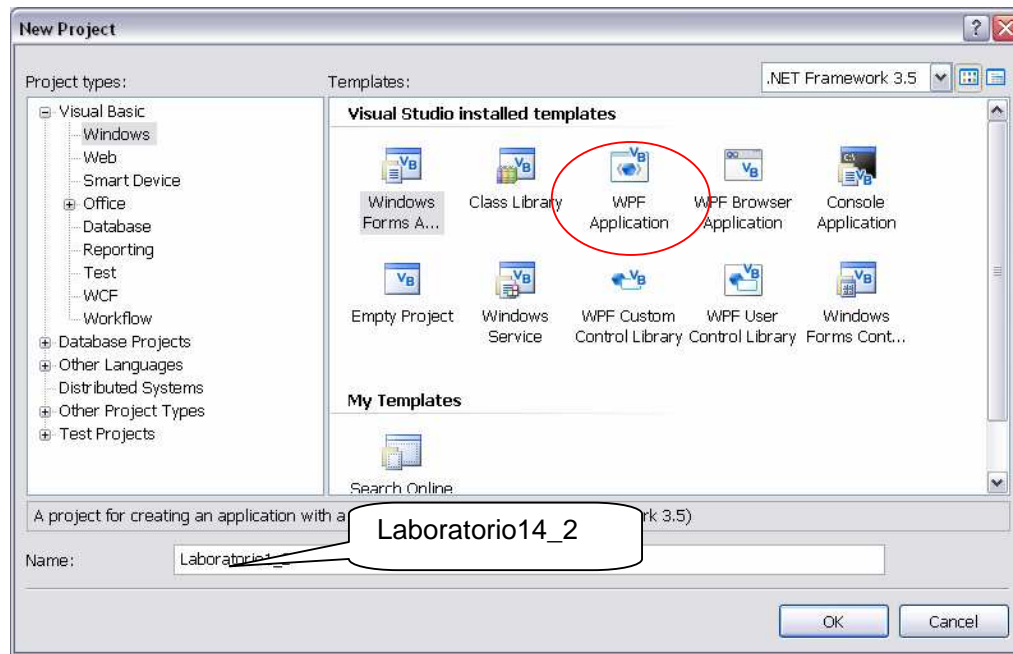
- **Botones:** Button y RepeatButton.
- **Cuadros de diálogo:** OpenFileDialog, PrintDialog y SaveFileDialog.
- **Entradas manuscritas digitales:** InkCanvas y InkPresenter.
- **Documentos:** DocumentViewer, FlowDocumentPageViewer, FlowDocumentReader, FlowDocumentScrollViewer y StickyNoteControl.
- **Entrada:** TextBox, RichTextBox y PasswordBox.
- **Diseño:** Border, BulletDecorator, Canvas, DockPanel, Expander, Grid, GridView, GridSplitter, GroupBox, Panel, ResizeGrip, Separator, ScrollBar, ScrollViewer, StackPanel, Thumb, Viewbox, VirtualizingStackPanel, Window y WrapPanel.
- **Multimedia:** Image, MediaElement y SoundPlayerAction.
- **Menús:** ContextMenu, Menu y ToolBar.
- **Navegación:** Frame, Hyperlink, Page, NavigationWindow y TabControl.
- **Selección:** CheckBox, ComboBox, ListBox, TreeView y RadioButton, Slider.
- **Información para el usuario:** AccessText, Label, Popup, ProgressBar, StatusBar, TextBlock y ToolTip.

## Laboratorio14

### TIPOS DE CONTENEDORES EN WPF

Cree una aplicación Windows que permita ingresar dos números enteros y muestre en una ventana la suma de los dos.

1. Ingrese a Visual Studio 2008
2. Seleccione Menú File -> New Project
3. En "Projects Types" elegir "Visual Basic" y en Templates elija "WPF Application" y en Name escriba "Laboratorio14\_2"



4. Existen 5 tipos principales de contenedores en WPF:

- StackPanel
- DockPanel
- Grid
- WrapPanel
- Canvas

A continuación se trabajarán con los cinco tipos de canvas en una misma ventana.

Agregar primero un panel de tipo StackPanel



Agregar el siguiente código en código XAML

**<StackPanel Orientation="Horizontal">**

**<Button Width="100" Height="50"**

**Background ="AliceBlue" FontFamily="Arial"**

**FontSize="20" FontWeight="Bold">**

**Aceptar**

**</Button>**

**<Button Width="100" Height="50"**

**Background ="AliceBlue" FontFamily="Arial"**

**FontSize="20" FontWeight="Bold">**

**Cancelar**

**</Button>**

**<Button Width="100" Height="50"**

**Background ="AliceBlue" FontFamily="Arial"**

**FontSize="20" FontWeight="Bold">**

**Omitir**

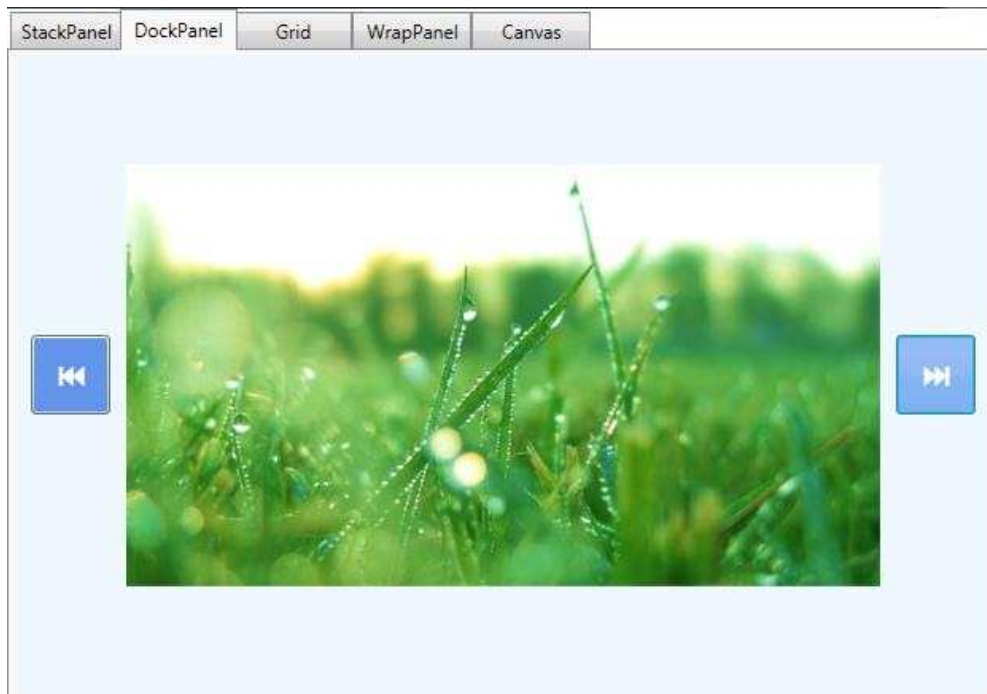
**</Button>**

**</StackPanel>**

El contenedor agrupa varios objetos, y los ubica de forma horizontal porque así se ha especificado con la propiedad **Orientation="Horizontal"**, cambiando el atributo a "Vertical", los elementos automáticamente se distribuirían verticalmente.

Agregar un Panel DockPanel

Llamado "Panel de acoplamiento", y es que este panel lo que hace es ubicar a los elementos hacia donde queremos que se mantenga, por ejemplo si decimos a la derecha, el elemento especificado se mantendrá a la derecha.



Agregar el siguiente código XAML

```
<DockPanel LastChildFill="True" Background="AliceBlue">
```

```
    <Button DockPanel.Dock="Left" Width="50" Height="50"
```

```
    Background="CornflowerBlue" FontFamily="Webdings"
```

```
    FontSize="20" FontWeight="Bold"
```

```
    Foreground="White" Margin="10,0,0,0"> 9 </Button>
```

```
    <Button DockPanel.Dock="Right" Width="50" Height="50"
```

```
    Background="CornflowerBlue" FontFamily="Webdings"
```

```
    FontSize="20" FontWeight="Bold"
```

```
    Foreground="White" Margin="0,0,10,0"> : </Button>
```

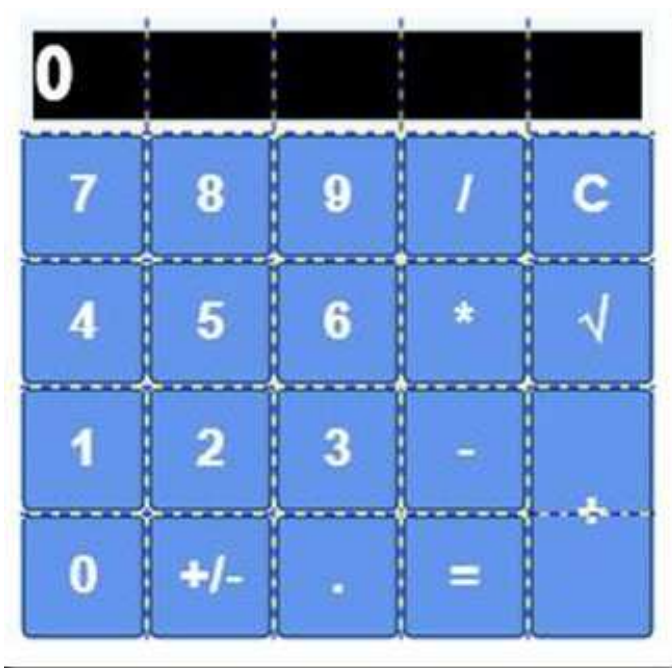
```
    <Image Source="Foto.jpg" Margin="10"></Image>
```

## </DockPanel>

Se pueden observar varias ubicaciones en este panel, la primera es que estamos aplicando lo que mencionamos anteriormente, estamos ubicando los elementos hacia donde queremos que se mantengan constantes, en el primer caso del botón, estamos aplicando `<Button DockPanel.Dock="Left">`; lo cual indica que el primer botón va a ser ubicado a la izquierda, note que estamos dando un ancho y un alto, con lo cual mantendrá esos valores constantes. De igual forma pasa con el botón `<Button DockPanel.Dock="Right">`.

Algo distinto sucede con la imagen del final: `<Image Source="Foto.jpg" Margin="10"></Image>`; note como no se le está asignando ni un anchor, ni características de ancho y alto, entonces, esta imagen pasa a ocupar todo el espacio que queda libre. Pero esto sucede gracias al atributo `LastChildFill="True"`.

Agregando un panel grid para diseñar una calculadora:



Esta calculadora ha sido desarrollada a partir de las definiciones de filas y columnas de un panel GRID y cada botón ha sido agregado en cada una de las celdas.

A continuación agregar el siguiente código XAML:

```
<Grid HorizontalAlignment="Center"
VerticalAlignment="Center"
Background="AliceBlue">
<Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto"/>
    <ColumnDefinition Width="Auto"/>
    <ColumnDefinition Width="Auto"/>
    <ColumnDefinition Width="Auto"/>
    <ColumnDefinition Width="Auto"/>
</Grid.ColumnDefinitions>
<Grid.RowDefinitions>
    <RowDefinition Height="Auto"/>
    <RowDefinition Height="Auto"/>
    <RowDefinition Height="Auto"/>
    <RowDefinition Height="Auto"/>
    <RowDefinition Height="Auto"/>
</Grid.RowDefinitions>

    <TextBlock Grid.Column="0" Grid.Row="0"
Grid.ColumnSpan="5"
Background="Black" FontFamily="Arial" FontSize="30"
FontWeight="Bold" Foreground="White" Margin="5">
0
</TextBlock>

    <Button Grid.Column="0" Grid.Row="1" Height="50"
Width="50"
Background="CornflowerBlue" FontFamily="Arial"
FontSize="20" FontWeight="Bold" Foreground="White">
7
</Button>

    <Button Grid.Column="1" Grid.Row="1" Height="50"
Width="50"
Background="CornflowerBlue" FontFamily="Arial"
FontSize="20" FontWeight="Bold" Foreground="White">
8
</Button>
```



```
<Button Grid.Column="2" Grid.Row="1" Height="50"
Width="50"
```

```
Background = "CornflowerBlue" FontFamily="Arial"
```

```
FontSize="20" FontWeight="Bold" Foreground="White">
```

```
9
```

```
</Button>
```

```
<Button Grid.Column="0" Grid.Row="2" Height="50"
Width="50"
```

```
Background = "CornflowerBlue" FontFamily="Arial"
```

```
FontSize="20" FontWeight="Bold" Foreground="White">
```

```
4
```

```
</Button>
```

```
<Button Grid.Column="1" Grid.Row="2" Height="50"
Width="50"
```

```
Background = "CornflowerBlue" FontFamily="Arial"
```

```
FontSize="20" FontWeight="Bold" Foreground="White">
```

```
5
```

```
</Button>
```

```
<Button Grid.Column="2" Grid.Row="2" Height="50"
Width="50"
```

```
Background = "CornflowerBlue" FontFamily="Arial"
```

```
FontSize="20" FontWeight="Bold" Foreground="White">
```

```
6
```

```
</Button>
```

```
<Button Grid.Column="0" Grid.Row="3" Height="50"
Width="50"
```

```
Background = "CornflowerBlue" FontFamily="Arial"
```

```
FontSize="20" FontWeight="Bold" Foreground="White">
```

```
1
```

```
</Button>
```

```
<Button Grid.Column="1" Grid.Row="3" Height="50"
Width="50"
```

```
Background = "CornflowerBlue" FontFamily="Arial"
```

```
FontSize="20" FontWeight="Bold" Foreground="White">
```

```
2
```

```
</Button>
```

```
<Button Grid.Column="2" Grid.Row="3" Height="50"
Width="50"
```

```
Background ="CornflowerBlue" FontFamily="Arial"
```

```
FontSize="20" FontWeight="Bold" Foreground="White">
```

```
3
```

```
</Button>
```

```
<Button Grid.Column="0" Grid.Row="4" Height="50"
Width="50"
```

```
Background ="CornflowerBlue" FontFamily="Arial"
```

```
FontSize="20" FontWeight="Bold" Foreground="White">
```

```
0
```

```
</Button>
```

```
</Button>
```

```
<Button Grid.Column="1" Grid.Row="4" Height="50"
Width="50"
```

```
Background ="CornflowerBlue" FontFamily="Arial"
```

```
FontSize="20" FontWeight="Bold" Foreground="White">
```

```
+/-
```

```
</Button>
```

```
<Button Grid.Column="2" Grid.Row="4" Height="50"
Width="50"
```

```
Background ="CornflowerBlue" FontFamily="Arial"
```

```
FontSize="20" FontWeight="Bold" Foreground="White">
```

```
.
```

```
</Button>
```

```
<Button Grid.Column="3" Grid.Row="1" Height="50"
Width="50"
```

```
Background ="CornflowerBlue" FontFamily="Arial"
```

```
FontSize="20" FontWeight="Bold" Foreground="White">
```

```
/
```

```
</Button>
```

```
<Button Grid.Column="3" Grid.Row="2" Height="50"  
Width="50"
```

```
Background ="CornflowerBlue" FontFamily="Arial"
```

```
FontSize="20" FontWeight="Bold" Foreground="White">
```

```
*
```

```
</Button>
```

```
<Button Grid.Column="3" Grid.Row="3" Height="50"  
Width="50"
```

```
Background ="CornflowerBlue" FontFamily="Arial"
```

```
FontSize="20" FontWeight="Bold" Foreground="White">
```

```
-
```

```
</Button>
```

```
<Button Grid.Column="3" Grid.Row="4" Height="50"  
Width="50"
```

```
Background ="CornflowerBlue" FontFamily="Arial"
```

```
FontSize="20" FontWeight="Bold" Foreground="White">
```

```
=
```

```
</Button>
```

```

<Button Grid.Column="4" Grid.Row="3"
Grid.RowSpan="2" Width="50"

Background ="CornflowerBlue" FontFamily="Arial"

FontSize="20" FontWeight="Bold" Foreground="White">

+

</Button>

<Button Grid.Column="4" Grid.Row="1" Height="50"
Width="50"

Background ="CornflowerBlue" FontFamily="Arial"

FontSize="20" FontWeight="Bold" Foreground="White">

C

</Button>

<Button Grid.Column="4" Grid.Row="2" Height="50"
Width="50"

Background ="CornflowerBlue" FontFamily="Symbol"

FontSize="20" FontWeight="Bold" Foreground="White">

Ö

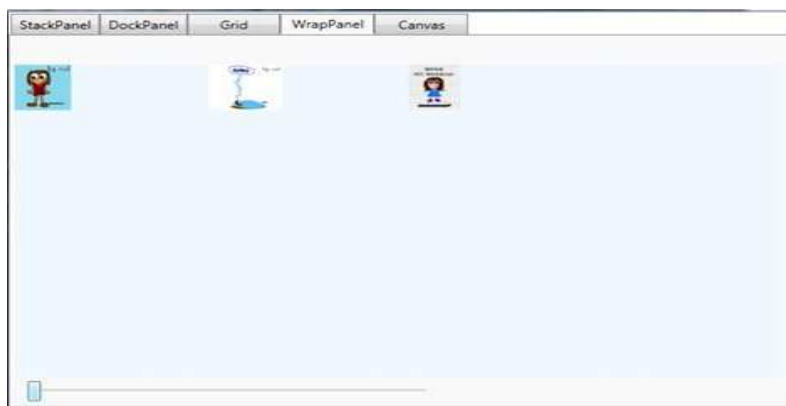
</Button>

</Grid>

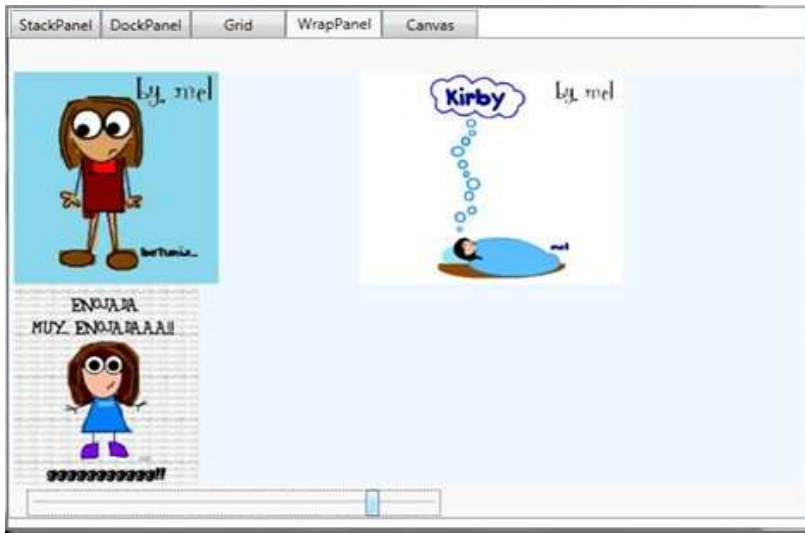
```

Agregar el panel Wrappanel que permite agrupar los elementos de forma ordenada y que sean siempre visible. Es decir, reacomodan los objetos si es necesario.

Slider con valor 50



Slider con valor 200 se mostraría así:



Finalmente el panel CANVAS que permite ubicar los elementos según la ubicación que especifiquemos.



El código para tener este efecto en el canvas es:

```
<Button Grid.Column="3" Grid.Row="2" Height="50"
Width="50"

Background ="CornflowerBlue" FontFamily="Arial"

FontSize="20" FontWeight="Bold" Foreground="White">
*

</Button>

<Button Grid.Column="3" Grid.Row="3" Height="50"
Width="50"

Background ="CornflowerBlue" FontFamily="Arial"

FontSize="20" FontWeight="Bold" Foreground="White">
-

</Button>

<Button Grid.Column="3" Grid.Row="4" Height="50"
Width="50"

Background ="CornflowerBlue" FontFamily="Arial"

FontSize="20" FontWeight="Bold" Foreground="White">
=

</Button>
```

# Autoevaluación

1. ¿Qué es globalización?

---

---

---

2. ¿Que representa la clase CultureInfo y RegionInfo?

---

---

---

3. ¿Cuál es el objetivo de WPF?

---

---

---

4. ¿Los controles en las aplicaciones WPF, no pueden tener eventos?

---

---

---

## Para Recordar

- El espacio de nombres **System.Globalization** contiene clases que definen información relativa a la referencia cultural, incluido el idioma, el país o región, los calendarios utilizados, los modelos de formato para las fechas, la moneda y los números y el criterio de ordenación de las cadenas.
- La clase **CultureInfo** proporciona información sobre una referencia cultural concreta (lo que se denomina "configuración regional" en desarrollo de código no administrado).
- La clase **CultureInfo** especifica un nombre único para cada referencia cultural, basándose en RFC 4646 (Windows Vista y posterior). Este nombre está formado por un código ISO 639 para la referencia cultural que está asociado a un idioma y se compone de dos letras minúsculas y por un código ISO 3166 para la referencia cultural secundaria que está asociado a un país o región y está compuesto de dos letras.
- WPF permite suministrar interfaces de usuario mejoradas muy interactivas y estéticamente atractivas. Los elementos visuales ofrecen transparencia, brillo, efectos 3d, animaciones, transformaciones, etc.

### Fuente

<http://msdn.microsoft.com/es-es/library/system.globalization.aspx>

<http://msdn.microsoft.com/es-es/library/system.globalization.cultureinfo.aspx>

<http://hectorperez.wordpress.com/2008/05/24/introduccion-a-wpf-y-creacion-de-una-aplicacion-wpf-3-d-parte-3-de-n/>