Visual Basic .Net y ASP .NET

Manual Lenguaje de programacion avanzado I y II

Bucaramanga 2014

Contenido

PRÓLOGO

Visual Basic y ASP .NET a su alcance proporciona los elementos necesarios para conocer el entorno del lenguaje de Programación Visual Basic. Este libro ha sido pensado para todos aquellos que estén interesados en conocer este lenguaje de Programación desde sus aspectos básicos, el manejo de bases de datos y ASP.NET. Todo el material didáctico del libro se basa en el aprendizaje guiado por la conceptualización de cada tema, la realización de ejemplos prácticos explicados y con gráficos que visualizán el objetivo de los ejercicios.

Debido al enfoque practico, basado en ejercicios y al esfuerzo de síntesis resulta posible utilizarlo para adquirir con facilidad y rapidez un conocimiento para el desarrollo de aplicaciones en Visual Basic.NET.

Todos los ejemplos del libro han sido compilados y ejecutados con el programa Microsoft Visual Studio 2003 (la mayoría de los ejemplos también los puede compilar y ejecutar con las versiones Express Edition Visual Basic 2005 y Visual Web Developer 2005); cada ejemplo es explicado línea por línea, solamente se exceptúan aquellas instrucciones que se han explicado con anterioridad.

Capítulos del Libro

Visual Basic y ASP .NET a su alcance contiene 14 capítulos distribuidos de la siguiente forma:

- **Capítulo 1, Iniciando Visual Studio .NET**: Habla sobre el entorno de desarrollo de Visual Studio .NET: página de inicio, plantillas de aplicaciones, menú principal, barra de herramientas, cuadro de herramientas, explorador de soluciones, diseñador de formularios Windows Form y editor de código.
- **Capítulo 2, Un Primer Proyecto Visual Basic .NET**: Habla de cómo se crea un primer proyecto, creación de la interfaz de usuario, establecimiento de las propiedades en un objeto, programación, generación de archivo un ejecutable y ejecución del proyecto.
- **Capítulo 3, Fundamentos de Visual Basic .NET**: Maneja los conceptos básicos del lenguaje de programación Visual Basic .NET: variables, constantes, tipos de datos, operadores, expresiones y palabras reservadas, además encontrará ejemplos explicativos.
- **Capítulo 4, Estructuras de control**: Maneja los conceptos propios de las estructuras de decisión *if, else, select- case* y las estructuras repetitivas *while, for, do- loop while,* así como una serie de ejemplos que permiten reforzar los conceptos. Al final del capítulo encontrará ejercicios para resolver**.**
- **Capítulo 5, Módulos y Procedimientos**: Aquí se habla sobre creación de módulos y procedimientos Sub y Function. Además se conceptualiza sobre algunas funciones predeterminadas matemáticas y de manipulación de caracteres; complementado con ejemplos prácticos.
- **Capítulo 6, Matrices**: Se habla sobre la estructuración de datos en matrices unidimensionales y bidimensionales con elementos de un mismo tipo; también se presentan ejemplos aplicados al tema; así como ejercicios para practicar.

- **Capítulo 7, Interfaz de Usuario Avanzada**: Se habla sobre algunos controles que permiten mejorar la interfaz gráfica de usuario en un programa, y con cada control se realiza un ejemplo práctico.
- **Capítulo 8, Formularios de Interfaz Sencilla (SDI) y Múltiple (MDI)**: En este capítulo encontrará la forma de crear aplicaciones con interfaz de documento sencillo, múltiple, así como la creación de barras de herramientas; explicado a través de ejemplos.
- **Capítulo 9, Gráficos y animación**: Habla sobre los objetos gráficos comunes: ínea, curva, rectángulo, arco, elipse y polígono**. A**demás se trabajan las propiedades top, left, width, height, utlizadas para el movimiento, contracción y expansión de un control
- **Capítulo 10, La Programación orientada a Objetos con .NET**: Se manejan los conceptos de programación orientada a abjetos utilizando el lenguaje de programación Visual Basic .NET en lo referente a: constructores, sobrecarga de métodos, herencia, polimorfismo: Todo explicado a través de ejemplos.
- **Capítulo 11, Acceso a Bases de Datos con .NET**: Se conceptualiza sobre bases de datos, lenguaje de consulta estructurado (SQL), acceso a una base de datos. Complementada con ejemplos.
- **Capítulo 12, ASP.NET**: En este capítulo se habla de cómo crear, ejecutar una aplicación Web ASP.NET. Además se explica por medio de ejemplos la utilización de varios controles en aplicaciones Web ASP.NET: Label, TextBox, Button, CheckBox, Radiobutton, DropDownList, RequiredFieldValidator, RangeValidator, CompareValidator**,** CustomValidator, RegularExpressionValidator.
- **Capítulo 13, Acceso a Bases de Datos con ASP.NET**: Aplicando los conceptos de bases de datos vistos en el capítulo 11, se realiza una conexión a una base de datos para manipular información utilizando aplicaciones Web, apoyado con ejemplos.
- **Capítulo 14, Servicios Web**: En este capítulo se conceptualiza sobre los servicios Web, creándose ejemplos con aplicaciones Windows Forms y Web.

INTRODUCCIÓN

Visual Basic .NET (**VB.NET**) es una versión de Visual Basic enfocada al desarrollo de aplicaciones .NET. El lenguaje de programación es Visual Basic que apareció en el año 1991 como una evolución del QuickBasic que fabricaba Microsoft. Dicho lenguaje de programación es orientado a objetos, donde es posible la creación de clases que pueden derivarse de otras mediante herencia, sobrecarga de métodos, control estructurado de excepciones o creación de aplicaciones con múltiples hilos de ejecución, además de contar con la extensa librería de .NET, la que permite desarrollar tanto aplicaciones Windows Forms y formularios Web, así como el manejo de diversos proveedores de bases de datos, el envío de datos mediante documentos XML[1] y la generación de informes mediante Crystal Reports a partir de archivos de texto, bases de datos, etc.

Que es Microsoft .NET

Microsoft .NET es un entorno integrado de ejecución, compilación, depuración, y desarrollo de aplicaciones. Los diferentes lenguajes de programación de la plataforma, comparten el mismo entorno, normas, reglas y librerías de Microsoft .NET Framework.

La plataforma .NET proporciona software que permite conectar sistemas, información, dispositivos y usuarios distintos de un modo más unificado y personalizado. Incorpora servicios Web XML como medio para permitir la interoperabilidad entre tecnologías diferentes. También proporciona a los desarrolladores de software herramientas, tecnología para crear rápida y eficazmente soluciones de negocio que abarcan múltiples aplicaciones y múltiples dispositivos cliente entre diversas organizaciones, además permite a los usuarios controlar qué información, cómo y cuándo se les entrega.

.NET incluye una completa familia de productos creados para trabajar con los estándares de XML e Internet. Estos productos incluyen los siguientes componentes que trabajan con soluciones basadas en XML:

- Herramientas para desarrollar soluciones
- Servidores para gestionar, crear e implantar soluciones
- Servicios para integrar, conectar y reutilizar soluciones

Que es .NET Framework

El .NET Framework es un conjunto de servicios de programación diseñados para simplificar el desarrollo de aplicaciones sobre el entorno distribuido de Internet. 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.

[1] XML, es el estándar de Extensible Markup Language. XML no es más que un conjunto de reglas para definir etiquetas semánticas que nos organizan un documento en diferentes partes. XML es un metalenguaje que define la sintaxis utilizada para definir otros lenguajes de etiquetas estructurados.

- 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.

.NET Framework contiene dos componentes principales: Common Language Runtime y la biblioteca de clases de .NET Framework. Common Language Runtime 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. La biblioteca de clases, el otro componente principal de .NET Framework, es una completa colección orientada a objetos de tipos reutilizables que se pueden emplear para desarrollar aplicaciones que abarcan desde las tradicionales herramientas de interfaz gráfica de usuario (GUI) o de línea de comandos hasta las aplicaciones basadas en las innovaciones más recientes proporcionadas por ASP.NET, como los formularios Web Forms y los servicios Web XML.

En la ilustración siguiente se muestra la relación de Common Language Runtime y la biblioteca de clases con las aplicaciones y el sistema en su conjunto.

.NET Framework en contexto.

Cuando se crea una nueva aplicación Windows en Visual Basic .NET, se genera un código inicial que incluye el espacio de nombres **System.Windows.Forms** y la clase **Form**. Con esta clase, se pueden crear fácilmente ventanas, botones, menús, barras de herramientas y otros elementos de pantalla.

Al compilar la aplicación, el código se traduce al lenguaje común del entorno de ejecución, Microsoft Intermediate Language (MSIL). Una vez la aplicación se ha compilado, el entorno de ejecución gestiona su construcción.

El entorno de ejecución incluye una característica denominada compilación *just-in-time* (JIT), que traduce código MSIL al lenguaje máquina del sistema en el que la aplicación se ejecutará. Cuando un dispositivo cliente con la plataforma .NET ejecuta la aplicación en Visual Basic .NET, se ejecuta en el lenguaje máquina del sistema cliente y puede integrarse totalmente e interactuar con otras aplicaciones y servicios basados en .NET independientemente del lenguaje en el que hayan sido desarrollados. Para entender cómo funciona el .NET Framework, es necesario conocer la siguiente terminología:

- *Clase*: es una entidad de programación con nombre que consta de un conjunto común de métodos, propiedades y atributos. Por ejemplo, Form es una de las clases del espacio de nombres System.Windows.Forms que se utiliza para crear formularios Windows Forms.
- *Espacio de nombres*: identifica una colección de clases relacionadas y/u otros espacios de nombres del .NET Framework. Algunos ejemplos de espacios de nombres incluyen:
    - System
    - System.Windows.Forms

- *Biblioteca de clases*: es una colección completa orientada a objetos de clases reutilizables y organizadas en espacios de nombres jerárquicos con base a su funcionalidad. 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. Por ejemplo, las clases de colección de .NET Framework implementan un conjunto de interfaces que puede usar para desarrollar sus propias clases de colección. Éstas se combinarán fácilmente 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).

- *Common Language Runtime*: es la base del .NET Framework. 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. El motor en tiempo de ejecución impone seguridad en el acceso al código. 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. Por ello, las características de seguridad del motor en tiempo de ejecución permiten que el software legítimo implementado en Internet sea excepcionalmente variado.

1. INICIANDO VISUAL STUDIO .NET

Visual Studio .NET es un entorno de desarrollo integrado (IDE) que ayuda a diseñar, desarrollar, depurar e implantar con rapidez soluciones basadas en .NET Framework. Se puede acceder a un conjunto común de herramientas, diseñadores y editores desde cualquiera de los lenguajes de programación de Visual Studio .NET. También se pueden crear aplicaciones Windows Forms y Web Forms que integren datos y lógica de negocio.

En este capítulo las plantillas que se referencian son del entorno grafico del Visual Studio 2003. Si trabaja con el Visual Basic Express 2005, el entorno grafico es exactamente igual pero ahorra pasos para iniciar un nuevo proyecto.

Visual Studio .NET incluye las características de programación que se describen en la siguiente tabla:

Tabla 1.1 Caracteristicas de programación en Visual Basic .NET.

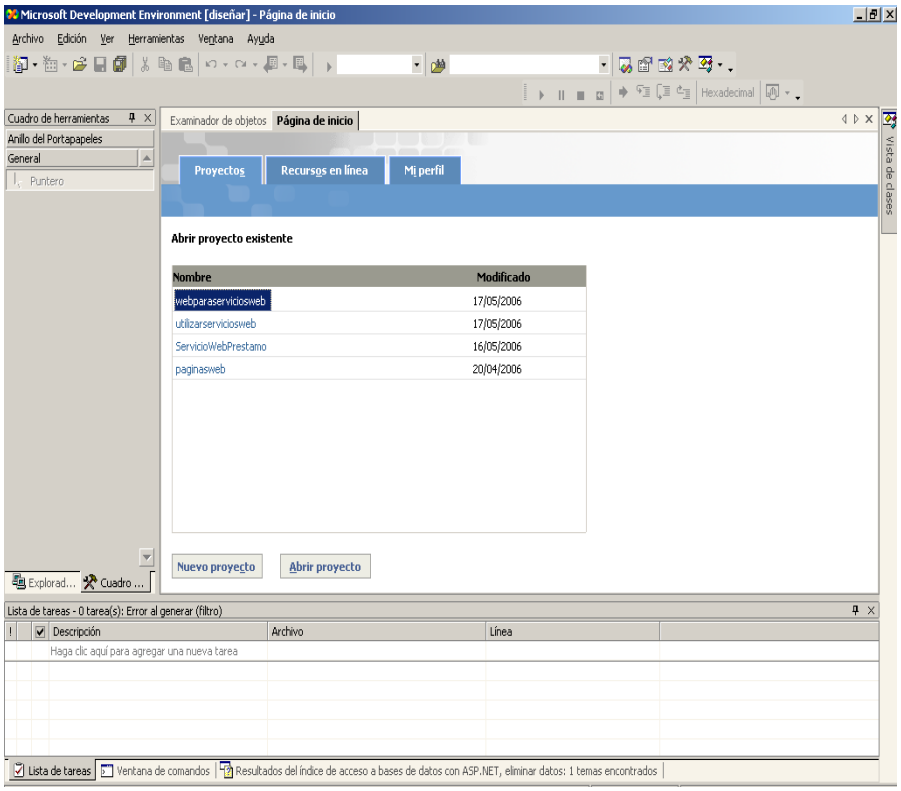
Característica	Descripción
Diseñador de Windows Forms	Una superficie de diseño gráfico que permite crear rápidamente la interfaz de usuario de una aplicación. Se puede arrastrar o dibujar controles sobre esta superficie.
Herramientas para Windows Forms	Se proporciona un Diseñador de Windows Forms, una plantilla *Aplicación Windows*, referencias de proyectos básicos y código de inicio como ayuda para crear aplicaciones Windows Forms estándares.
Herramientas para Web Forms	Se proporciona un Diseñador de Web Forms, una plantilla *Aplicación Web ASP.NET*, referencias de proyectos básicos y código de inicio como ayuda para crear aplicaciones Web Forms en las que la interfaz de usuario principal es un navegador.
Herramientas para servicios Web XML	Se proporciona una plantilla *Servicios Web ASP.NET*. Esta plantilla construye la estructura de un proyecto de aplicación Web en un servidor Web de desarrollo.
Soporte de múltiples lenguajes	Todos los lenguajes de programación de la plataforma .NET, incluyendo Visual Basic .NET y Visual C#, están integrados en el entorno de desarrollo.
Acceso a datos	Componentes para crear aplicaciones que comparten datos, herramientas de bases de datos visuales para acceder a los datos y un robusto conjunto de clases de Microsoft ADO.NET.
Gestión de errores	Las herramientas de depuración con soporte multilenguaje ayudan a encontrar y solucionar errores de código, y podemos utilizar clases de excepciones estructuradas para incluir la gestión de errores en nuestra aplicación.
Asistentes	Los asistentes ayudan a completar rápidamente tareas comunes. Cada página de un asistente ayuda a establecer opciones, configurar y personalizar proyectos.

Aunque el lenguaje de programación que se utilizará es Visual Basic .NET, el entorno de desarrollo que se usará para escribir programas recibe el nombre de entorno de desarrollo de Microsoft Visual Studio .NET, el cual contiene todas las herramientas necesarias para construir programas en el entorno Windows.

1.1 Página de Inicio Visual Studio .NET

Desde la página de inicio de Visual Studio .NET se puede: abrir proyectos, crear perfiles y establecer las preferencias del usuario. La figura 1.1, muestra la página de inicio:

Figura 1.1 Página de Inicio de Visual Studio .NET.



1.2 Crear o abrir proyectos Windows Forms

Para iniciar un nuevo proyecto o abrir un proyecto existente desde la página de inicio, se requiere unos pasos muy sencillos como son:

- Para crear un nuevo proyecto
  - Dar clic en la opción **Nuevo Proyecto**.

  - O -
  - Dar clic en la opción **Archivo** del menú, seleccionar **Nuevo,** luego la opción **Proyecto.**

- Abrir un proyecto existente

  - Haga clic en la opción **Abrir Proyecto.**

  - O -

  - Haga clic en la opción **Archivo** del menú, seleccionar **Abrir,** luego la opción **Proyecto.**

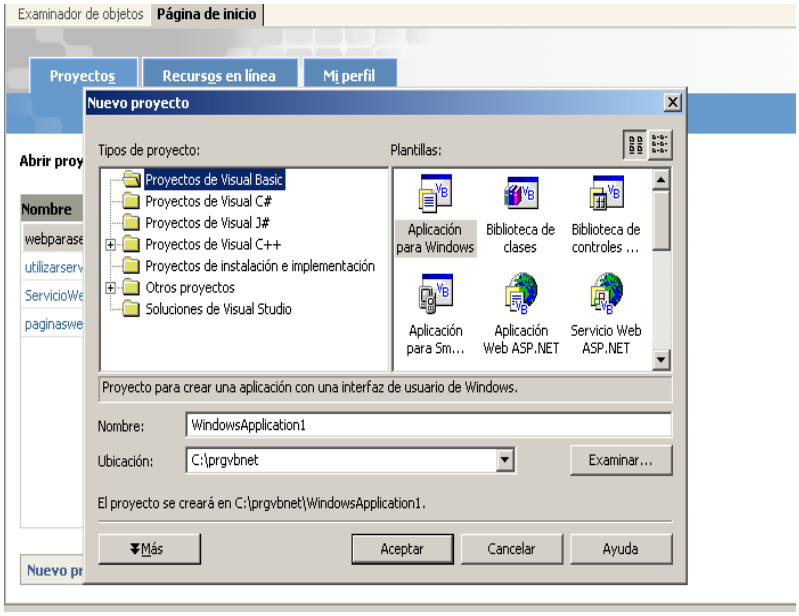
1.3 Plantillas de aplicaciones

Visual Studio .NET ofrece varias plantillas de aplicaciones para soportar el desarrollo de diferentes tipos de aplicaciones y componentes. Antes de iniciar un nuevo proyecto, se debe escoger el tipo de plantilla que se va a utilizar.

Una plantilla de aplicación proporciona archivos de inicio y una estructura de proyecto, además contiene los objetos básicos del proyecto y la configuración del entorno que se necesita para crear el tipo de aplicación que se desea.

Las plantillas que más se utilizan son: *Aplicación para Windows* y *Aplicación Web ASP.NET*. La plantilla *Aplicación para Windows* proporciona herramientas, estructuras y código de inicio para crear una aplicación estándar basada en Windows, añadiendo automáticamente las referencias básicas del proyecto y los archivos a utilizar como punto de partida para la aplicación. La plantilla *Aplicación Web ASP.NET* se utiliza para crear una aplicación Web ASP.NET en un equipo que tenga instalado Internet Information Services (IIS)[2] versión 5.0 o posterior. Para iniciar el diseño de la aplicación, la plantilla crea los archivos básicos necesarios en el servidor.

Figura 1.2 Plantilla de aplicaciones.



[2] Este servicio convierte a un computador en un servidor de Internet o Intranet es decir que en los computadores que tienen este servicio instalado se pueden publicar páginas Web tanto local como remotamente (servidor Web).

1.4 Elegir una plantilla

Cuando se inicia un nuevo proyecto en Visual Studio .NET uno de los primeros pasos es escoger una plantilla de aplicaciones. Para este caso se utilizará la plantilla **Aplicación para Windows**. Los pasos necesarios para crear un nuevo proyecto en Visual Basic .NET son los siguientes:

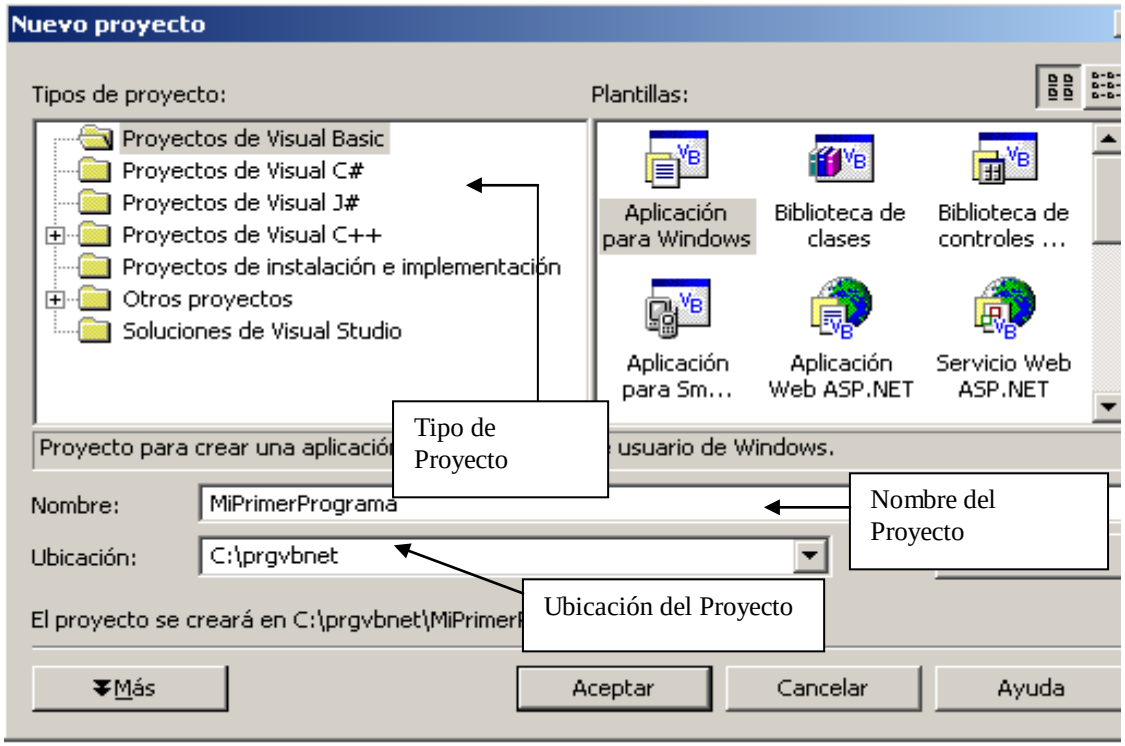
1. Abrir Visual Studio .NET.
2. En la ventana de **Inicio**, hacer clic en **Nuevo Proyecto**.

– O –

En el menú **Archivo**, seleccionar **Nuevo** y a continuación hacer clic en **Proyecto**.

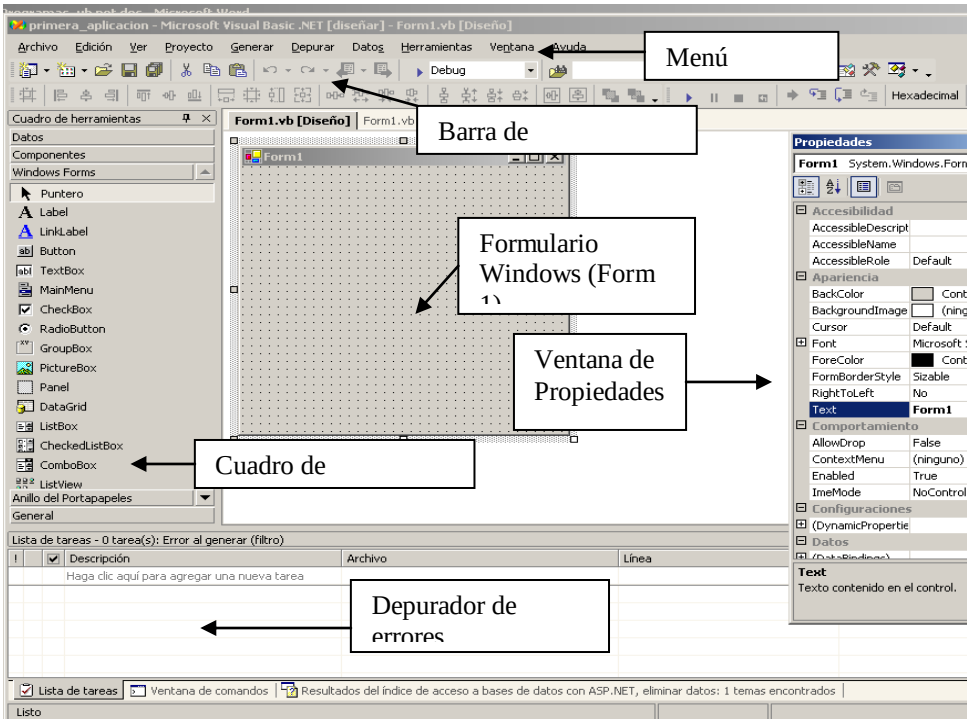
1. En la ventana **Tipos de proyecto**, dar clic en **Proyectos de Visual Basic**. En el panel **Plantillas**, hacer clic en **Aplicación para Windows**.
2. En el campo **Nombre**, escribir un nombre de proyecto exclusivo que indique el objetivo de la aplicación.
3. En el campo **Ubicación**, indicar la carpeta en la que desea guardar el proyecto, o haga clic en el botón **Examinar** para navegar hasta la carpeta destino.
4. Dar clic en el botón **Aceptar**.

Figura 1.3 Elección de una plantilla.



Al elegir la plantilla, se abrirá el diseñador de **Windows Forms**, visualizándose el formulario **Form1** del proyecto que se ha creado, la pantalla se representa en la figura 1.4:

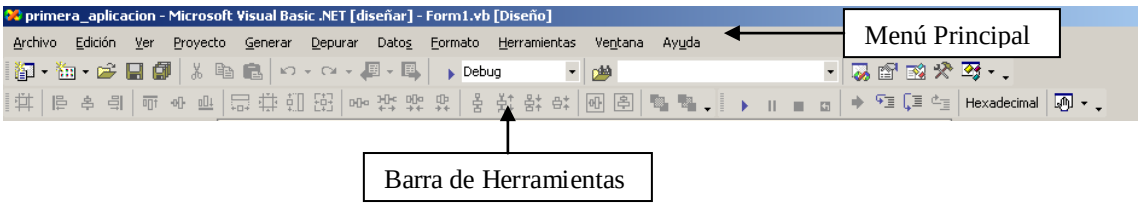
Figura 1.4 Pantalla inicial de un nuevo proyecto.



Cuando se selecciona una plantilla de aplicaciones aparece el entorno de desarrollo visual, donde encontrará:

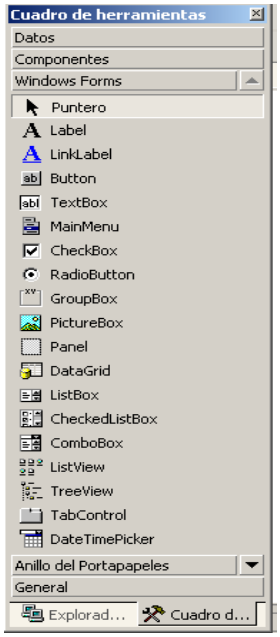
- **Menú principal y la barra de herramientas estándar**: Proporcionan acceso a la mayoría de comandos que controlan el entorno de programación de Visual Studio .NET. Los comandos del menú principal funcionan igual que todas las aplicaciones basadas en Windows. La barra de herramientas proporciona los iconos que sirven como acceso a la mayoría de comandos estándar de Windows. También contiene iconos que permiten abrir el explorador de soluciones, la ventana de propiedades, el cuadro de herramientas y otros elementos importantes del entorno de desarrollo.

Figura 1.5 Menu principal y barra de herramientas estándar.



- **Cuadro de Herramientas:** El cuadro de herramientas contiene diversos controles que se pueden utilizar para añadir ilustraciones, etiquetas, botones, cuadros de lista, barras de desplazamiento, menús y formas geométricas a una interfaz de usuario. Cada control que se añade a un formulario se convierte en un objeto de la interfaz de usuario programable en la aplicación. Estos objetos son visibles para los usuarios cuando la aplicación se ejecuta y funcionan como los objetos estándares de cualquier aplicación basada en Windows.

Figura 1.6 Cuadro de Herramientas.



Para cerrar y abrir el cuadro de herramientas, se deben realizar los siguientes pasos:

- Para cerrar el cuadro de herramientas, haga clic en el botón **Cerrar** (**x**) de la esquina superior derecha del cuadro de herramientas.
- Para abrir el cuadro de herramientas, en el menú **Ver** haga clic en **Cuadro de herramientas**.

Para ocultar o visualizar el cuadro de herramientas:

- Para ocultar el cuadro de herramientas, haga clic en el pin de la barra de títulos del cuadro de herramientas.
- Para visualizar el cuadro de herramientas cuando este oculto, en el menú **Ver** haga clic en **cuadro de herramientas**.

Para cambiar de lugar el cuadro de herramientas:

- Haga clic con el botón derecho en la barra de títulos del cuadro de herramientas y, a continuación, haga clic en **Flotante**.
- Arrastre el cuadro de herramientas a la posición deseada.

1.5 Ventana de propiedades

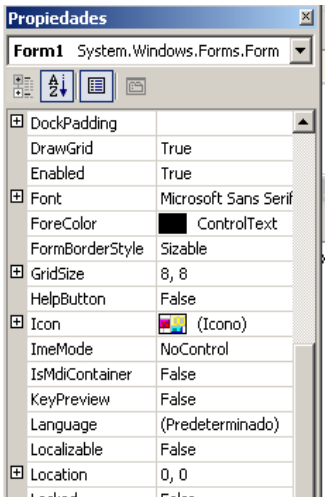
La ventana de propiedades muestra una lista de las propiedades de los controles lo que facilita la configuración de valores o propiedades del control seleccionado; las cuales se pueden modificar mientras se crea o se edita una aplicación. Una propiedad describe las características de un objeto, como tamaño, título, color, etc.

Para visualizar la ventana de propiedades se puede realizar una de las siguientes operaciones:

- Hacer clic en **Ventana Propiedades** en el menú **Ver** o pulse la tecla **F4**.

- Hacer clic en el objeto para seleccionarlo, pulsar el botón secundario del **mouse** y seleccionar **propiedades**.

Figura 1.7 Cuadro de propiedades.



Algunos controles, documentos y formularios muestran un gran número de propiedades en la ventana **Propiedades**., Esto puede dificultar la localización de la propiedad que se desea establecer. La ventana **Propiedades** permite seleccionar las propiedades de un formulario o control en una vista ordenada por categorías en lugar de una vista alfabética, según sea el caso se puede realizar lo siguiente:

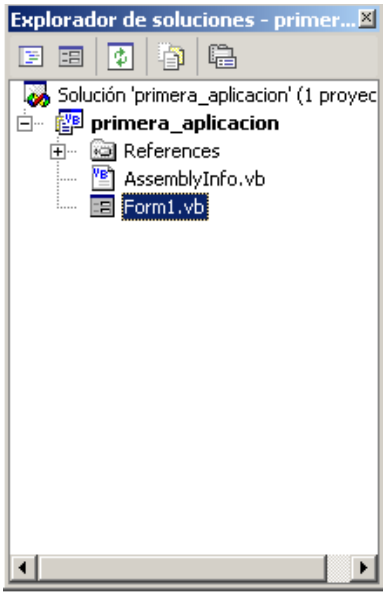
- **Para visualizar las propiedades por categorías:** Haga clic en el botón **Por categorías**  de la ventana **Propiedades**. Las propiedades para el formulario o control seleccionado se dividirán en categorías definidas por el control.
- **Para visualizar las propiedades de una categoría**: se expande el nodo  de la propiedad El signo más (+) permite expandir la propiedad y el signo menos (-) contraer la propiedad.
- Para visualizar las propiedades alfabéticamente, haga clic en el botón **Alfabético**  de la ventana de propiedades.

1.6 Explorador de soluciones

El explorador de soluciones  permite visualizar archivos, realizar tareas de administración de archivos en una solución o en un proyecto. Una única solución basada en Visual Basic .NET y sus proyectos aparecen en una lista jerárquica que proporciona información actualizada sobre el estado de la solución, los proyectos y los archivos. Una solución es un contenedor para proyectos y elementos de solución que pueden incluirse en una aplicación. Normalmente, una solución contiene uno o más proyectos relacionados. Un proyecto es un contenedor dentro de una solución que se utiliza para administrar, generar y depurar lógicamente los elementos del (os) proyecto (s) que constituyen la aplicación.

Para abrir el explorador de soluciones, en el menú **Ver** haga clic en **explorador de soluciones**. Aparecerá una ventana en la esquina superior derecha del área de desarrollo. Se puede mover y cambiar el tamaño de esta ventana mediante la funcionalidad estándar de arrastrar y soltar.

Figura 1.8 Ventana Explorador de soluciones.



Para mostrar todos los archivos, en la barra de herramientas del **explorador de soluciones** haga clic en **Mostrar todos los archivos**. , esta opción permite ver los archivos creados manualmente o durante la compilación.

Visual Studio .NET soporta numerosos tipos de archivos y sus extensiones de archivo asociadas. La siguiente tabla describe algunos tipos de archivos habituales específicos para las soluciones basadas en Visual Basic .NET.

Tabla 1.2 Tipos de Archivos en Visual Basic .NET.

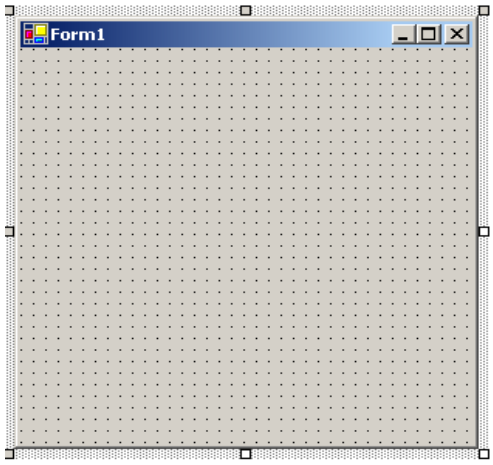
Extensión	Nombre	Descripción
.sln	Solución Visual Studio	Organiza proyectos, elementos de proyectos y elementos de soluciones en una solución proporcionando al entorno referencias a sus ubicaciones en disco.
.suo	Solution user options	Registra todas las opciones que se pueden asociar a una solución de forma que cada vez que abra la solución incluya las personalizaciones que se hallan realizado.
.vb	Proyecto Visual Basic	Representa los archivos de formularios, controles de usuario, clases y módulos que pertenecen a la solución de un solo proyecto. Los archivos que no estén basados en un lenguaje de programación tienen su propia extensión. Por ejemplo, un archivo Crystal Report tiene la extensión .rpt y un archivo de

		texto tiene la extensión .txt.
.vbproj	Proyectos Visual Basic	Representa los archivos de formularios, controles de usuario, clases y módulos que pertenecen a la solución con múltiples proyectos. Esta extensión permite diferenciar entre archivos escritos en Visual Basic .NET y otros lenguajes compatibles con .NET. (Visual C# utiliza .csproj.)
.aspx .asmx .asax	Elementos de proyecto Web	Los elementos de proyectos Web incluyen archivos Web específicos como: .aspx para Web Forms, .asmx para servicios Web XML y .asax para clases globales de aplicaciones. Los proyectos Web también utilizan la extensión .vb para clases y módulos.

1.7 Diseñador formulario Windows (Windows Forms)

Cuando se inicia un proyecto en Visual Basic .NET el diseñador de formulario de Windows (Windows Forms) se abre en vista diseño (un formulario tiene vista diseño cundo se inicia el proyecto y modo de ejecución cuando se ejecuta la aplicación), mostrándose el formulario **Form1** del proyecto. En dicho formulario se pueden ubicar los controles u objetos necesarios para la aplicación arrastrándolos desde el cuadro de herramientas para crear la interfaz de usuario. El formulario predeterminado contiene los elementos mínimos utilizados por la mayoría de formularios: barra de título, cuadro de control y los botones minimizar, maximizar y cerrar.

Figura 1.9 Formulario Windows.



Para visualizar un formulario en **vista diseño** se puede realizar siguiente:

- En el **explorador de soluciones**  , hacer doble clic en el formulario.

Ó

- En el **Explorador de soluciones**  , seleccionar el formulario, hacer clic en el botón **Ver Diseñador**  de la barra de herramientas.

Visual Studio .NET proporciona un **editor de código** que permite escribir y modificar el código del proyecto. Se puede asociar código directamente al formulario o a los controles del proyecto; también se puede ubicar el código en un módulo. Para visualizar el editor de código se puede realizar lo siguiente:

- Seleccionar el control, hacer clic con el botón derecho y seleccionar **Ver código**.
- Dar doble clic sobre el control seleccionado

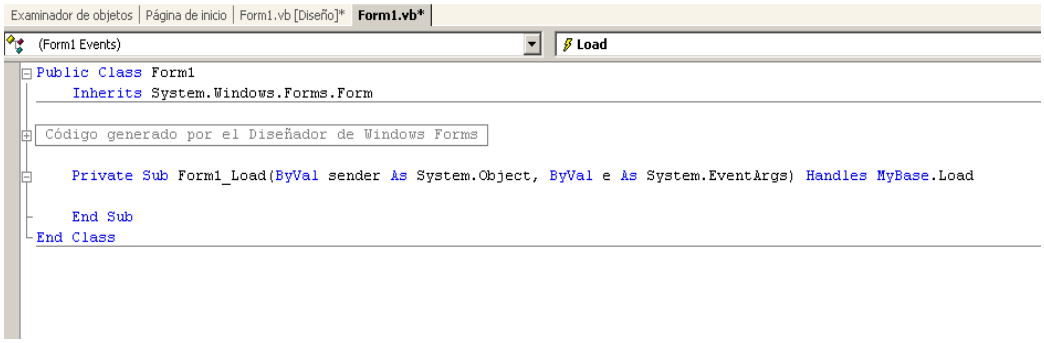
Si por ejemplo lo que se selecciona es un formulario se puede realizar lo siguiente:

1. En el **explorador de soluciones** haga clic en el formulario del que desea visualizar el código.
2. En la barra de herramientas del **explorador de soluciones** haga clic en el botón **Ver código**  .

– O –

1. Seleccione el formulario y en el menú **Ver**, haga clic en **Código.**
2. Seleccione el formulario y pulse la tecla **F7**.

Figura 1.10 Editor de código.



El editor de código contiene dos listas desplegables en la parte superior de la ventana: **Nombre de clase** a la izquierda y **Nombre de método** a la derecha. La lista **Nombre de clase** muestra todos los controles asociados al formulario. Si se dá clic en el nombre de un control, en la lista **Nombre de método se** muestra todos los eventos de dicho control (los eventos son acciones que se realizan sobre el control y que la aplicación puede interpretar). Utilizando las listas **Nombre de clase** y **Nombre de método** conjuntamente se puede localizar y editar rápidamente el código de la aplicación.

2. UN PRIMER PROYECTO VISUAL BASIC .NET

Crear una aplicación en Visual Basic .NET requiere de unos pasos muy sencillos como son: iniciar un nuevo proyecto Visual Basic .NET, crear la interfaz del usuario, establecer las propiedades de los objetos, escribir el código, guardar la aplicación y ejecutar el proyecto.

2.1 Iniciar un nuevo proyecto Visual Basic .NET

En la ventana inicial Visual Basic .NET proporciona herramientas para desarrollar una solución a un problema específico. Lo primero que se debe tener claro son las necesidades del usuario y el conjunto de características que se requieren en la aplicación (referirse al capítulo 1).

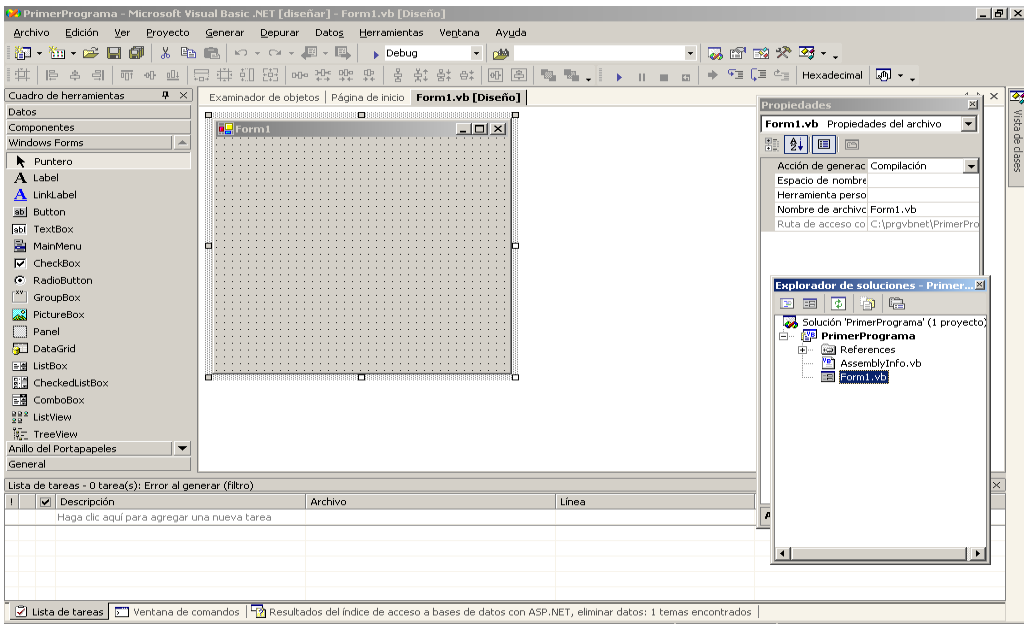
2.2 Ejemplo práctico

Como primer ejemplo se dará solución a una necesidad de un usuario. Este requiere un programa donde "*pueda capturar un nombre, la edad y que estos datos los pueda visualizar en una ventana de mensajes"*. Para esto se deben realizar los siguientes pasos:

1. Abrir Visual Studio .NET.
2. En el menú **Archivo** seleccionar **Nuevo**, a continuación hacer clic en **Proyecto**.
3. En el panel **Tipos de proyecto** hacer clic en **Proyectos de Visual Basic**. En el panel **Plantillas** hacer clic en **Aplicación para Windows**.
4. En el cuadro **Nombre** escriba **PrimerPrograma**.
5. Dar clic en **Examinar**, buscar la carpeta en donde se quiere crear el nuevo proyecto y a continuación haga clic en **Abrir,** luego dé clic en el botón **Aceptar**.

A continuación se visualizará la siguiente figura:

Figura 2.1 Pantalla proyecto PrimerPrograma.



2.3 Crear la interfaz de usuario

Para crear la interfaz de usuario de la aplicación en primer lugar se deben ubicar los controles necesarios en un formulario desde el cuadro de herramientas. Para este caso se diseñará la interfaz de usuario utilizando los siguientes controles: 2 **Label** (permite colocar texto descriptivo no editable), 2 **TextBox** (permite mostrar o aceptar como entrada una sola línea de texto) y 1 **Button** (representa un control botón de Windows).

Para añadir controles a un formulario se pueden seguir los siguientes pasos:

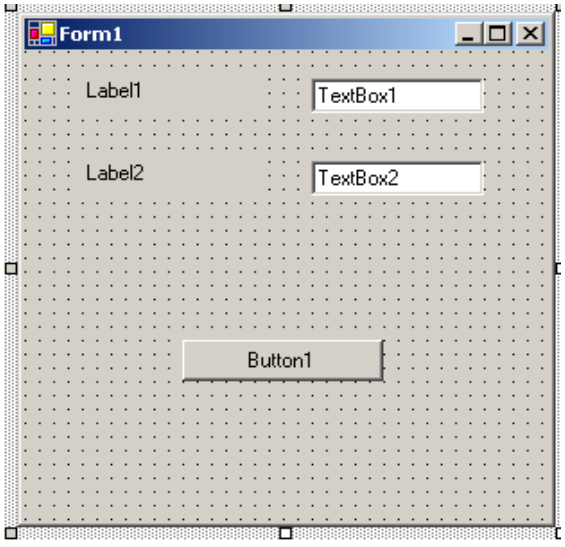
1. Si el cuadro de herramientas no está abierto, en el menú **Ver** haga clic en **Cuadro de herramientas**.
2. En el **cuadro de herramientas** haga clic en el control que desea añadir al formulario y arrástrelo a la ubicación deseada en el formulario.

Los siguientes pasos permiten:

1. **Cambiar la posición de un control:** Hacer clic en el mismo para seleccionarlo y arrastrarlo a la posición deseada en el formulario.
2. **Cambiar el tamaño de un control:** Hacer clic en el control para seleccionarlo, mantenerlo pulsado y arrastrar uno de los extremos de ajuste del tamaño hasta que el control tenga el tamaño deseado y soltar el botón del **mouse**.

Cuando existen varios controles del mismo tipo estos se enumeran en el mismo orden en que son colocados (nombre del control más numero del consecutivo). La interfaz de usuario del ejemplo se muestra en la siguiente figura:

Figura 2.2 Prepantalla de la interfaz de usuario.



2.4 Establecer las propiedades de los objetos de la interfaz de usuario

Después de colocar los controles u objetos a un formulario, se puede establecer sus propiedades en la ventana **Propiedades** o en el **Editor de código**.

Para modificar las propiedades se selecciona el control en el formulario y se

cambia su configuración en la ventana **Propiedades**, para lo cual se pueden realizar los siguientes pasos:

1. Si la ventana **Propiedades** no está abierta, en el menú **Ver** haga clic en **Ventana de propiedades**.
2. Haga clic en el control para el que desea establecer una propiedad.
3. En la ventana **Propiedades** seleccione la propiedad y establezca el valor deseado.

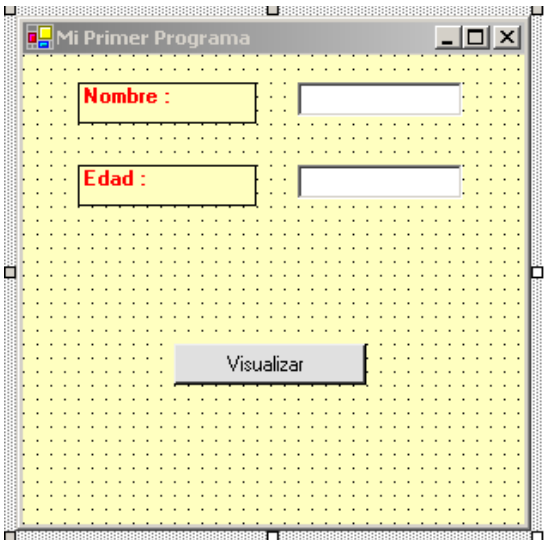
Para el caso del ejemplo establezca las siguientes modificaciones en la propiedad correspondiente a cada uno de los siguientes controles:

Tabla 2.1 Propiedades de los controles de la aplicación PrimerPrograma.

Control	Propiedad	Valor
Label1	Name	txtnombre
	Text	Nombre :
	BorderStyle	Fixedsimple
	Font	Negrita
	ForeColor	red (rojo)
Label2	Name	txtedad
	Text	Edad:
	BorderStyle	Fixedsimple
	Font	Negrita
	ForeColor	red (rojo)
Textbox1	Name	camponombre
	Text	En blanco
Textbox2	Name	campoedad
	Text	En blanco
Button1	Name	boton
	Text	Visualizar
Form1	Name	formulario
	Text	Mi primer programa
	BackColor	Amarrillo

La interfaz de usuario con las modificaciones respectivas en las propiedades de cada control queda como se muestra en la siguiente figura:

Figura 2.3 Pantalla final de la interfaz de usuario.



2.5 Escribir código

Una vez se hayan establecido las propiedades iniciales del formulario y sus objetos, se puede agregar el código que se ejecutará en respuesta a un evento específico de cada control. Los eventos ocurren cuando se realizan diferentes acciones sobre un control u objeto. Por ejemplo, el evento **Clic** de un botón tiene lugar cuando un usuario hace clic sobre él con el **mouse**. Por lo general en las aplicaciones es necesario escribir código para lograr el objetivo de dar solución al problema planteado por el usuario.

Para escribir código que se ejecute en un evento determinado de un control se debe realizar los siguientes pasos:

1. Seleccionar el control en el que se desea escribir código.
2. En el menú **Ver** haga clic en la opción **Código** para abrir el **Editor de código** o doble clic sobre el control.
3. En la lista **Nombre de método** haga clic en el evento deseado para abrir su gestor de eventos en el **Editor de código**. El Editor de código muestra las sentencias de programación que marcan el inicio (Sub) y el final (End Sub) del procedimiento del evento en particular.
4. Escriba el código entre los puntos inicial y final del cuerpo del procedimiento.

Para el caso del ejemplo seleccione el objeto **Button** llamado **boton** y abra el editor de código, la figura muestra el editor de código del control

Figura 2.4 Editor de código del objeto boton.

```
Public Class Form1
    Private Sub boton_Click(ByVal sender As System.Object, ByVal e As
                System.EventArgs) Handles boton.Click  { inicio del procedimiento
    ……..
    …….
    ,,,,,,,,,,,,,,,,
    End Sub  } final del procedimiento
End Class
```

Ahora escriba el siguiente código:

```
Msgbox("Hola" & camponombre.text & " Tú tienes:" & campoedad.text & "años de edad")
```

El editor de código quedaría de la siguiente forma:

Figura 2.5 Código Visual Basic escrito en el editor de código del objeto boton.

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

Msgbox ("Hola" & camponombre.text & " Tú tienes:" & campoedad.text & "años de edad")

End Sub
```

En el anterior código se utiliza la función predeterminada **MsgBox** que permite visualizar un cuadro de mensajes. La función muestra la etiqueta "Hola", el contenido de la propiedad **Text** del objeto **camponombre** unido por el operador de concatenación

de cadenas (&), además se muestra la etiqueta “ Tú tienes:“ y el valor de la propiedad **Text** del control **campoedad** concatenado con la etiqueta “ años de edad”.

2.6 Guardar la aplicación

Una vez finalizada la creación de la aplicación, se debe guarda la aplicación dentro del entorno de desarrollo para asegurar que todos los cambios realizados se almacenen, esto se puede realizar de la siguiente forma:

- En el menú **Archivo**, haga clic en **Guardar todo**.
- O dé clic en el icono  de la barra de herramientas

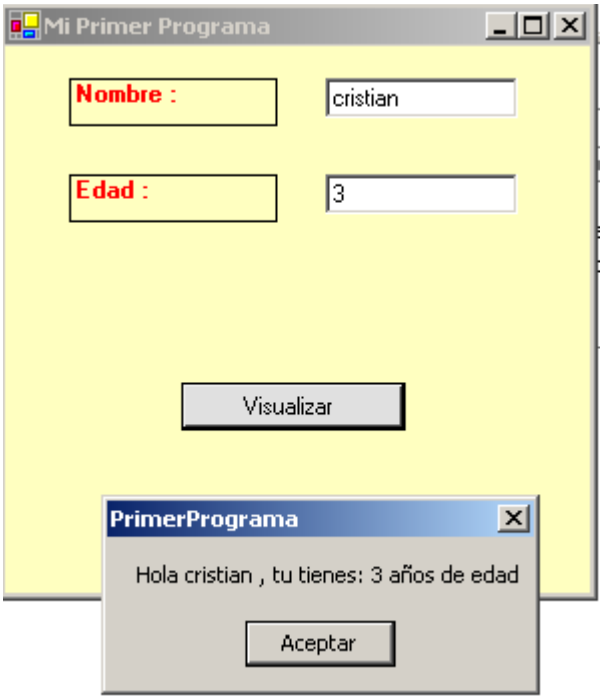
2.7 Ejecutar el proyecto

Para ejecutar el proyecto en el entorno de desarrollo de Visual Basic.NET, se debe realizar lo siguiente:

1. Haga clic en el botón Iniciar  de la barra de herramientas estándar. También puede presionar **F5** para ejecutar el proyecto. Si la aplicación se ejecuta sin errores, aparecerá una versión ejecutable del formulario.
2. Cuando termine de ejecutar la aplicación se debe cerrar el formulario para regresar al entorno de programación.

Al ejecutarse y digitar los textos respectivos y pulsar el boton **Visualizar**, se mostrará la siguiente figura:

Figura 2.6 Ejecución de la aplicación PrimerPrograma.



2.8 Generar un archivo ejecutable para el proyecto

Cuando se ejecuta un proyecto o solución, automáticamente se crea un archivo ejecutable (.exe) en la carpeta **nombre_carpeta_proyecto\bin** (en el caso del ejercicio el nombre de la carpeta es PrimerPrograma\bin)**.** En el menú **Generar** en la opción **Configuración de la solución**, existen dos opciones de configurar las soluciones activas: Si se selecciona en la lista **Debug**, se creará un segundo archivo .exe en la carpeta nombre_carpeta_proyecto\obj\Debug; Si se selecciona **Release**, el segundo archivo .exe se creará en la carpeta **nombre_carpeta_proyecto\obj\Release**. Otra forma de generar un archivo ejecutable es seleccionar en el menú **Generar** la opción **Generar solución.**

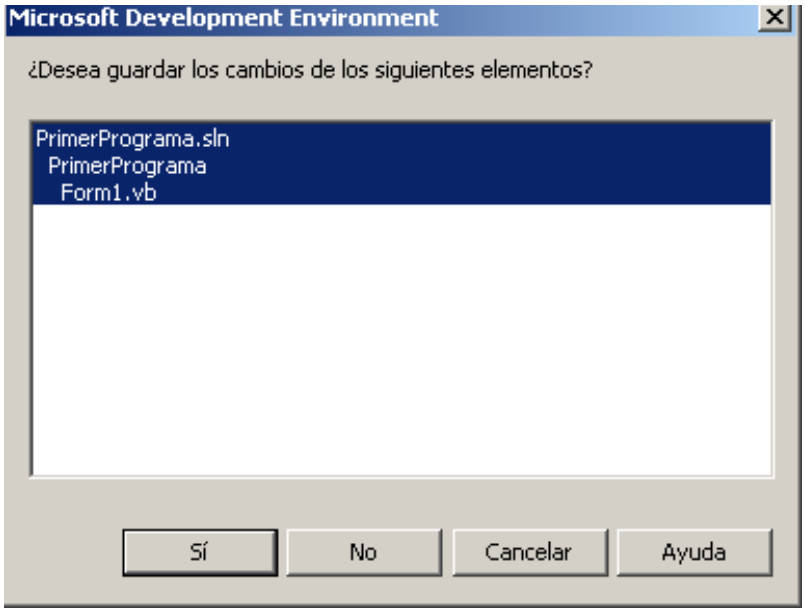
Para ejecutar la aplicación **PrimerPrograma** desde el sistema operativo Windows, existen dos formas de hacerlo: la primera es utilizando el comando **Ejecutar** del menú **Inicio** y buscar el archivo **PrimerPrograma.exe** que se encuentra en la carpeta **PrimerPrograma\bin.** La segunda forma es realizar la búsqueda de la carpeta **PrimerPrograma\bin** utilizando el explorador de Windows y dar doble clic en el nombre del archivo **PrimerPrograma.exe** para ejecutarlo.

2.9 Guardar, cerrar el proyecto PrimerPrograma y salir de Visual Studio .NET

Para guardar y cerrar el proyecto **PrimerPrograma** o cualquier otro proyecto y a la vez salir de Visual Studio. NET se pueden realizar los siguientes pasos:

1. En el menú **Archivo**, hacer clic en la opción **Salir**.
2. Si al realizar el primer paso se muestra la figura 2.7 se debe dar clic en **Sí** para guardar todos los cambios que se hayan realizado en la aplicación. Si por el contrario no desea guardar los cambios que se realizaron en la aplicación dé clic en **No**.

Figura 2.7 Guardar y cerrar la aplicación PrimerPrograma.



2.10 Abrir y ejecutar una aplicación existente en Visual Basic .NET

Para volver a abrir la aplicación **PrimerPrograma** o cualquier otra aplicación existente se deben realizar los siguientes pasos:

1. Abrir Visual Studio .NET. Si Visual Studio.NET ya está abierto, en el menú **Archivo** haga clic en **Cerrar solución** para cerrar todas las soluciones abiertas.
2. Si en la pantalla inicial de Visual Studio .NET se visualiza el nombre del proyecto que se desea abrir haga doble clic para abrir la aplicación. Si por el contrario no se visualiza la aplicación en el menú **Archivo**, seleccione **Abrir** y a continuación haga clic en **Proyecto**.
3. Busque la carpeta **PrimerPrograma** y seleccione el archivo **PrimerPorgrama.sln** y pulse la opción **Abrir,** para abrir la aplicación en el entorno de desarrollo de Visual Basic .NET.

3. FUNDAMENTOS DE VISUAL BASIC .NET

3.1 Variables

Una **variable** es un espacio de memoria para almacenar un valor de un determinado tipo de dato dentro de un programa. El valor de la variable puede ser modificado durante la ejecución del programa. El nombre de la **variable** puede contener letras, números, etc., los nombres de las **variables** deben aportar información que permita identificar el tipo de información que se va a almacenar. Para declarar una variable se utiliza la sentencia **Dim.** El formato de la declaración de variables es:

```
Dim nombre_variable As tipo_de_dato
```

Después de declarar el tipo de variable, se le puede asignar un valor inicial de la siguiente forma:

```
nombre_variable= valor_inicial
```

También es posible declarar una variable en el siguiente formato:

```
Dim nombre_variable As tipo_de_dato = valor_inicial
```

Las variables pueden ser locales o globales. Las variables locales son aquellas que se crean dentro de un bloque específico de programación y se destruirán al finalizarse el bloque de programación.

Ejemplo

```
Private Sub funcion ()
        Dim var_local As Integer (variable local)
        var_local=10
        ……
End Sub
```

Las variables globales son aquellas que pueden ser modificadas desde cualquier punto de un programa.

Ejemplo variable global

```
Public Class variables
  Public var_global As Integer (variable global)
    Private Sub funcion1 ()
    Dim var_local As Integer
    var_global=10 (valor modificado en la funcion1 ())
      ……
 End Sub
 Private Sub funcion2 ()
   Dim var_local As Integer
   var_global=20 (valor modificado en la funcion2 ())
    ……
    …….
  End Sub
  ::::::::::
End Class
```

3.2 Constantes

Son variables que permanecen constantes durante el desarrollo del programa. Existen constantes numéricas, de carácter, lógicas, etc. El formato de la declaración de una variable constante es:

Const nombre_variable= valor_constante

3.3 Tipos de datos

Un tipo de dato define todo el posible rango de valores que una variable puede tomar al momento de ejecución del programa y a lo largo de toda la vida útil del propio programa. Para seleccionar un tipo de dato en un programa se debe tener en cuenta el que mejor se adapte a los datos. Por ejemplo, si un programa necesita almacenar valores entre -20000 y 30000 los tipos de datos que se podrían utilizar serian **short**, **long**, **Integer**. Si utiliza el tipo de dato **short** la aplicación utilizaría menos memoria para almacenar el valor de la variable. A continuación se presenta la tabla con los tipos de datos que soporta Visual Basic .NET y su respectivo tamaño.

Tabla 3.1 Tipos de datos de Visual Basic .NET.

Tipo	Valores	Tamaño
Boolean	Representa un valor verdadero (true) o falso (false)	2 bytes
Byte	Representa un valor de 8 bits en un rango entre 0 y 255	1 byte
Char	Representa un carácter de 16 bits	2 bytes
DateTime	Representa un valor de fecha y hora	8 bytes
Decimal	Representa un valor de 28 dígitos significativos	12 bytes
Double	Representa un valor en coma flotante de 64 bits	8 bytes
Integer	Representa un valor entre un rango de +-2,147,483,698	4 Bytes
Long	Representa un valor entre un rango de +-9.223.372.036.854.775.807	8 Bytes
Short	Representa un valor entre un rango de +-32.677	2 Bytes
String	Cadena de caracteres	0 a 2 billones de caracteres

3.4 Ejemplo práctico tipos de datos

Realizar una aplicación que permita a un usuario visualizar en cajas de texto los diferentes tipos de datos cuando se pulse un botón llamado **Tipos de Datos**. Además poder salir de la aplicación utilizando un botón **Salir**.

- **Iniciar un nuevo proyecto Visual Basic .NET**

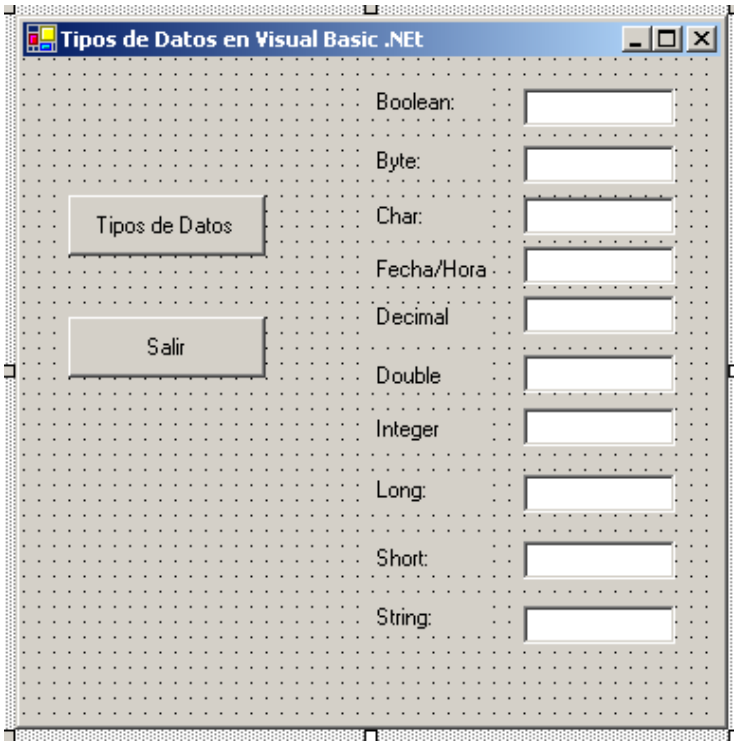
1. En el menú **Archivo**, seleccione **Nuevo** y a continuación haga clic en **Proyecto**.
2. En el panel **Tipos de proyecto**, haga clic en Proyectos de Visual Basic. En el panel **Plantillas** haga clic en **Aplicación para Windows**.

3. En el cuadro **Nombre**, escriba **TiposdeDatos**.
4. Haga clic en **Examinar**, busque la carpeta donde quiera crear el nuevo proyecto y a continuación haga clic en **Abrir**, luego haga clic en el botón **Aceptar**.

- **Crear la interfaz de usuario.**

Utilizando el cuadro de herramientas haga clic en el control específico y ubique los siguientes controles en el formulario en la posición deseada: 10 **Label**, 10 **TextBox**, 2 **Button**. La figura 3.1., muestra la interfaz de usuario para la aplicación **TiposdeDatos**.

Figura 3.1 Interfaz de usuario (tipos de datos).



- **Establecer las propiedades de los objetos del interfaz de usuario**

Después de colocar los controles u objetos a un formulario, se puede establecer o modificar sus propiedades en la ventana **Propiedades** o en el **Editor de código**.

Para modificar las propiedades se selecciona el control en el formulario y se cambia su configuración en la ventana **Propiedades**. Para el caso del ejemplo, establezca las modificaciones a los controles que se muestran en la tabla 3.2.:

Tabla 3.2 Propiedades de los controles de la aplicación TiposdeDatos.

Control	Propiedad	Valor
Label1	Name	lblboolean
	Text	Boolean:
Label2	Name	
	Text	Byte:
Label3	Name	lblchar
	Text	Char:
Label4	Name	lblfechora

	Text	Fecha/Hora:
Label5	Name	lbldecimal
	Text	Decimal:
Label6	Name	lbldouble
	Text	Double:
Label7	Name	lblinteger
	Text	Integer:
Label8	Name	lbllong
	Text	Long:
Label9	Name	lblshort
	Text	Short:
Label10	Name	lblstring
	Text	String:
TextBox1…TextBox10	Name	Txtcampo1…txtcampo10
	Text	En blanco
Button1	Name	boton
	Text	Tipos de Datos
Button2	Name	botonsalir
	Text	Salir
Form1	Name	formulario
	Text	Tipos de datos en Visual Basic .NET

- **Escribir código**

Una vez se hayan establecido las propiedades iniciales del formulario y sus objetos, se puede agregar el código que se ejecutará en respuesta a eventos.

Para escribir código que se ejecute en un evento determinado de un control, realice lo siguiente:

1. En el **explorador de soluciones**, seleccione el control para el que desea escribir código.
2. Escriba el código entre los puntos inicial y final del cuerpo del procedimiento.

Para el ejemplo, seleccione el objeto **boton** y abra el editor de código, la figura 3.4 muestra el editor de código del control.

Figura 3.2 Editor de código del control Button 1 de la aplicación TiposdeDatos.

```
Public Class formulario
    Private Sub boton_Click(ByVal sender As System.Object, ByVal e As  System.EventArgs)
                                Handles boton.Click

    ……..
    ……..
    ,,,,,,,,,,,,,,,,,
    End Sub
End Class
```

Entre **Sub** y **End Sub** escriba el siguiente código:

```
Dim boleana As Boolean = False
Dim bytes As Byte = 101
Dim caracter As Char = "a"
```

```
Dim fechas_hora As Date
Dim con_decimal As Decimal = 10.23
Dim entera As Integer = 32000
Dim doble As Double = 63528457
Dim larga As Long = 100258479
Dim corta As Short = 27000
Dim cadena As String = "hola mi gente"
txtcampo1.Text = boleana
txtcampo2.Text = bytes
txtcampo3.Text = caracter
txtcampo4.Text = fechas_hora
txtcampo5.Text = con_decimal
txtcampo6.Text = doble
txtcampo7.Text = entera
txtcampo8.Text = larga
txtcampo9.Text = corta
txtcampo10.Text = cadena
```

En el anterior código se definen las diferentes variables determinando el tipo de dato que almacenarán respectivamente. Además se inicializan las variables con los valores apropiados para cada tipo de dato. La variable **fechas_hora** es la única que no es inicializada, en dicha variable se almacenará los valores predeterminados de tipo **Date** del sistema operativo. Por último se le asigna a la propiedad **Text** de cada control **txtcampo1…10** la respectiva variable. Cuando el usuario pulse el botón **Tipo de Datos** en tiempo de ejecución se mostrará en cada cuadro de texto el valor respectivo de la variable que ha sido asignada.

Después seleccione el objeto **Salir**, abra el editor de código y escriba el siguiente código:

```
End
```

En el anterior código se utiliza la instrucción **End** que permite detener la ejecución de una aplicación. Visual Basic .NET dispone de varias palabras reservadas que no pueden ser utilizadas como variables, este tema que será tratado más adelante.

- **Guardar la aplicación**

Una vez finalizada la creación de la aplicación, se guarda dentro del entorno de desarrollo para asegurarnos de se almacenen los cambios realizados en los diferentes objetos que contiene la aplicación, esto lo puede realizar de la siguiente forma:

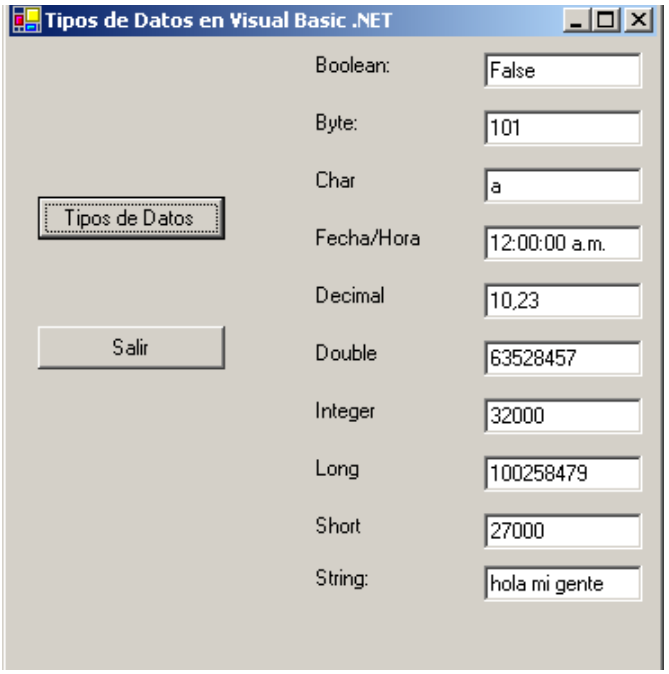
- En el menú **Archivo**, dé clic en **Guardar todo**.
- O haga clic en el icono  de la barra de herramientas

- **Ejecutar el proyecto**

Para ejecutar el proyecto en el entorno de desarrollo de Visual Basic.NET se debe realizar lo siguiente:

- Haga clic en el botón **Iniciar** de la barra de herramientas estándar. También puede presionar **F5** para ejecutar el proyecto. Si la aplicación se ejecuta sin errores aparecerá una versión ejecutable de la aplicación. Al pulsar el botón **Tipos de Datos** se visualizará la siguiente figura:

Figura 3.3 Ejecución aplicación TiposdeDatos.



Si desea salir de la aplicación pulse el botón **Salir** para regresar al entorno de programación.

3.5 Operadores y Expresiones

Los operadores son símbolos que indican cómo se operan los datos. Se clasifican en: aritméticos, relacionales, lógicos y de asignación. Las expresiones son combinaciones de constantes, variables, símbolos de operaciones, paréntesis.

3.5.1 Operadores aritméticos

Se utilizan para crear expresiones aritméticas, estas pueden resultar de la unión de variables o constantes con operadores.

Tabla 3.3 Operadores aritméticos.

Operadores de Visual Basic .NET	Operador	Expresión
Suma	+	numero1 +numero2
Resta	-	numero1 – numero2
Multiplicación	*	Numero1 * numero2
División	/	Numero1 / numero2
División entera	\	Numero1 \ numero2
Residuo	mod	numero1 mod numero2
Exponenciación	^	Numero1 ^ numero2

3.5.2 Operadores relacionales

Se utilizan para la toma de decisiones que se deban realizar en un programa.

Tabla 3.5 Operadores relacionales.

Operadores Relacionales	Operador	Expresión
Mayor que	>	variable1>variable2
Mayor o igual que	>=	variable1>=variable2
Menor que	<	variable1<variable2
Menor o igual que	<=	variable1<=variable2
Diferente	<>	variable1<>variable2
Igual a	=	variable1=variable2

3.5.3 Operadores lógicos

Al igual que los operadores de relación se utilizan para la toma de decisiones.

Tabla 3.6 Operadores lógicos.

Operadores lógicos	Operador	Expresión
Y	And	Es verdadero, si al evaluar cada uno de los operandos el resultado es verdadero, si uno de los operandos es falso el resultado será falso.
También Y	AndAlso	Es falso, si al evaluar el primer operando el resultado es falso, el segundo operando no es evaluado.
O	Or	Es falso, si al evaluar cada uno de los operandos el resultado es falso, si uno de los operandos es verdadero el resultado será verdadero.
También O	OrAlso	Es verdadero, si al evaluar el primer operando el resultado es verdadero, el segundo operando no es evaluado.
Negación	Not	El resultado de aplicar este operando es falso si al evaluar su operando el resultado es verdadero, y verdadero en caso contrario.
	Xor	Da como resultado verdadero, si al evaluar cada uno de los operando uno de ellos es verdadero y el otro falso, caso contrario es falso.

3.6 Ejemplo práctico operadores aritméticos

Realizar una aplicación que permita a un usuario visualizar en cajas de texto las operaciones aritméticas básicas cuando se pulse un botón llamado **Operadores Aritméticos**.

- **Iniciar un nuevo proyecto Visual Basic .NET**
  1. En el menú **Archivo**, seleccione **Nuevo** y a continuación haga clic en **Proyecto**.
  2. En el panel **Tipos de proyecto**, haga clic en Proyectos de Visual Basic. En el panel **Plantillas**, haga clic en **Aplicación para Windows**.
  3. En el cuadro **Nombre**, escriba **OperadoresAritmeticos**
  4. Haga clic en **Examinar**, busque la carpeta donde quiera crear el nuevo proyecto y a continuación haga clic en **Abrir**, luego haga clic en **Aceptar**.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control específico y ubique los siguientes controles en el formulario en la posición deseada: 7 **Label**, 7 **TextBox**, 2 **Button**.

- **Establecer las propiedades de los objetos de la interfaz de usuario**

Después de colocar los controles u objetos en el formulario, se establecen las propiedades en la ventana **Propiedades** o en el **Editor de código**.

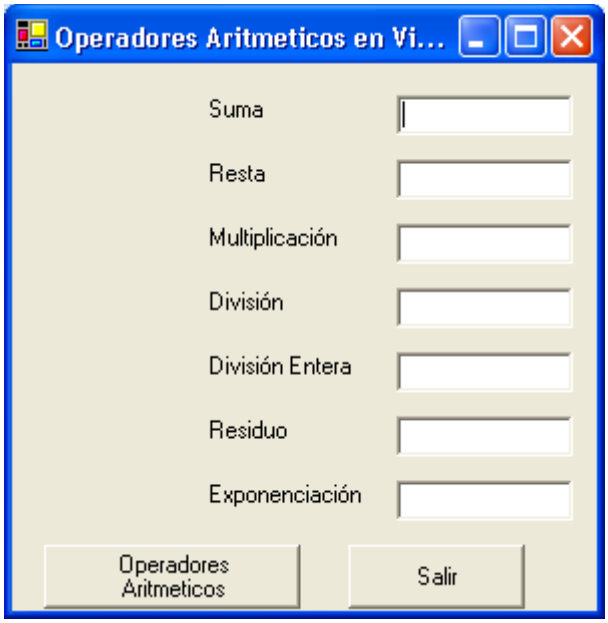
Para el caso del ejemplo establezca las siguientes modificaciones a los controles:

Tabla 3.4 Propiedades de los controles del proyecto Operadores Aritméticos.

Control	Propiedad	Valor
Label1	Name	lblsuma
	Text	Suma
Label2	Name	lblresta
	Text	Resta:
Label3	Name	lblmultiplicacion
	Text	Multiplicación
Label4	Name	lbldivision
	Text	División
Label5	Name	lbldiventera
	Text	División Entera
Label6	Name	lblresiduo
	Text	Residuo
Label7	Name	lblexponeniacion
	Text	Exponenciación
TextBox1…TextBox7	Name	txtcampo1…txtcampo7
	Text	En blanco
Button1	Name	boton
	Text	Operadores Aritméticos
Button2	Name	Botonsalir
	Text	Salir
Form1	Name	formulario
	Text	Operadores Aritméticos en Visual Basic .NET

La interfaz de usuario queda como se muestra en la siguiente figura:

Figura 3.4 Interfaz de usuario (operadores aritméticos).



- **Escribir código**

Una vez se hayan establecido las propiedades iniciales del formulario y sus objetos, se puede agregar código que se ejecute en respuesta a eventos.

Para escribir código que se ejecute en un evento determinado de un control, realice lo siguiente:

1. En el **explorador de soluciones**, seleccione el control para el que desea escribir código.
2. Escriba el código entre los puntos inicial y final del cuerpo del procedimiento.

Seleccione el objeto **botón,** abra el editor de código y escriba el siguiente código:

```
Dim valor_uno, valor_dos As Integer
valor_uno=8
valor_dos=5
txtcampo1.Text = valor_uno + valor_dos
txtcampo2.Text = valor_uno - valor_dos
txtcampo3.Text = valor_uno * valor_dos
txtcampo4.Text = valor_uno / valor_dos
txtcampo5.Text = valor_uno \ valor_dos
txtcampo6.Text = valor_uno mod valor_dos
txtcampo7.Text = valor_uno ^ valor_dos
```

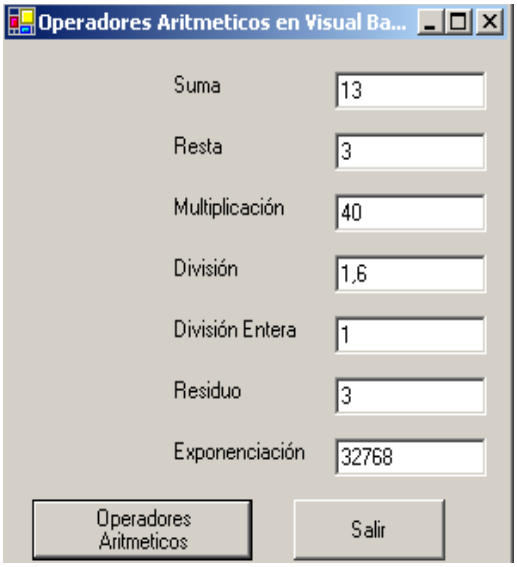
En el anterior código se definen las variables **valor_uno** y **valor_dos** de tipo **Integer**. Luego en dichas variables se almacenan los valores 8 y 5 respectivamente. También se le asigna una operación aritmética en la propiedad **Text** de cada caja de texto. Cuando el usuario pulse el botón en tiempo de ejecución mostrará en cada cuadro de texto el valor respectivo de la operación que le fue asignada.

- **Ejecutar el proyecto**

Para ejecutar el proyecto en el entorno de desarrollo de Visual Basic.NET se debe realizar lo siguiente:

- Haga clic en el botón Iniciar de la barra de herramientas estándar. También puede presionar F5 para ejecutar el proyecto. Si la aplicación se ejecuta sin errores, aparecerá una versión ejecutable del formulario y al pulsar el **botón Operadores Aritméticos** se visualizará la siguiente figura:

Figura 3.5 Ejecución aplicación Operadores Aritméticos.



3.7 Palabras clave

Las palabras clave son identificadores predefinidos que tiene significado especial para el compilador de Visual Basic .NET. Por lo tanto, una variable o constante definida por el usuario no puede tener el mismo nombre de una palabra clave. Las palabras clave de Visual Basic .NET se muestran en la siguiente tabla:

Tabla 3.7 Palabras Clave.

Palabras Clave				
AddHandler	AddressOf	ÁndAlso	Alias	And
Ansi	As	Assembly	Auto	Boolean
ByRef	Byte	Byval	Call	Case
Match	CBool	CByte	CChar	Cdate
CDec	CDbl	Char	CInt	Class
CLng	CObj	Const	CShort	CSng
CStr	CType	Date	Decimal	Declare
Default	Delegate	Dim	DirectCast	Do
Double	Each	Else	ElseIf	End
Enum	Erase	Error	Event	Exit
False	Finally	For	Friend	Function
Get	GetType	GoTo	Handles	If
Implements	Imports	In	Inherits	Integer
Interface	Is	Lib	Like	Long

Loop	Me	Mod	Module	MustInherit
MustOverride	MyBase	MyClass	Namespace	New
Next	Not	Nothing	NotInhertable	NotOverridable
Object	On	Option	Optional	Or
OrElse	Private	Property	Protected	Public
RaiseEvent	ReadOnly	ReDim	Rem	RemoveHandler
Resume	Return	Select	Set	Shadows
Shared	String	Structure	Static	Step
Stop	Short	Single	Sub	SyncLock
Then	Throw	To	True	Try
Typeof	Unicote	Until	When	While
UIT	WithEvents	WriteOnly	Xor	

4. ESTRUCTURAS DE CONTROL

Los programas vistos anteriormente son sencillos y lineales, donde se ejecutan las instrucciones una tras otra. Sin embargo, en los programas generalmente se necesita hacer cosas distintas dependiendo del estado de las variables o realizar un mismo proceso muchas veces sin escribir la misma línea de código una y otra vez.

Para realizar programas más complejos se utilizan las estructuras de control, como son la de toma de decisiones y los ciclos repetitivos.

4.1 Toma de decisiones

Al tener programas más complejos es necesario que se evalúen algunos resultados para determinar qué proceso u operación se debe ejecutar. Existen palabras clave de Visual Basic .NET que permiten tomar estas decisiones como son: **if** (si), **if-else** (si-sino), **select - case** (seleccionar un caso).

4.1.1 Sentencia If (Si)

Representa una toma de decisión sencilla, es decir si la condición que se evalúa es verdadera se realizan las instrucciones que pertenezcan al **If** y continuará con el resto del programa. Si la condición es falsa no entrará al **If** y por consiguiente todas las instrucciones que están en el **If** no se ejecutarán. Cada vez que se utilice un **If** se debe finalizar con **EndIf**. Su formato es:

```
1) If (condición) Then
       Instrucción(es) a ejecutarse si la condición es verdadera
   Endif
   Resto del programa
```

Si la condición del **If** es verdadera se realiza la instrucción y continúa con el resto del programa.

```
2) If (condición) Then      Instrucción a ejecutarse si la condición es verdadera
   Resto del programa
```

Si la condición del **If** es verdadera y se requiere realizar solo una instrucción no es necesaria la finalización del **If**.

.

```
3) If (condición) Then
       Instrucción(es) a ejecutarse si la condición es verdadera
     If (condición) Then
         Instrucción(es) a ejecutarse si la condición es verdadera
    Endif
  Endif
  Resto del programa
```

Se pueden anidar varios **If**, es decir, uno entre otro. Para ingresar cada vez al **If** más interno es necesario que la condición sea verdadera.

También se pueden utilizar los operadores lógicos (and, andalso, or, oralso, xor, not) para concatenar varias condiciones. Si se utiliza el operador lógico **And**, cada una de las condiciones debe ser verdadera para que ingrese al **If** y se ejecuten las instrucciones. En el caso del operador **Or**, si una de las condiciones es verdadera ingresará al **If** y ejecutará las instrucciones.

```
4) If (condición1 and condicion2…) Then
       Instrucción(es) a ejecutarse si las condiciones son verdaderas
   Endif
   Resto del programa
Ó
   If (condición1 or condicion2…) Then
       Instrucción(es) a ejecutarse si las condiciones son verdaderas
   Endif
   Resto del programa
```

Además de utilizar los operadores lógicos, también se pueden utilizar los operadores relaciones (<, >, >=, <=, <>), con el fin de evaluar una o más variables para determinar su valor verdadero para que ingrese al **If** y se ejecuten las instrucciones correspondientes.

```
5) If (condición1 > condicion2) Then
       Instrucción(es) a ejecutarse si las condiciones son verdaderas
   Endif
   Resto del programa
Ó
   If (condición1 > condicion2 and condicion1 > condicion3) Then
       Instrucción(es) a ejecutarse si las condiciones son verdaderas
   Endif
   Resto del programa
```

4.1.2 Sentencia If- Else (Si - Sino)

Es una estructura compuesta que evalúa una condición. Si esta es verdadera realizará las instrucciones contenidas en el **If**, en caso contrario ingresará por el **Else**. El programa solo tomará una de las dos alternativas y continuará con el resto del programa. Su formato es el siguiente:

```
If (condición) Then
    Instrucción(es) a ejecutarse si la condición es verdadera
Else
    Instrucción(es) a ejecutarse si la condición es falsa
Endif
Resto del programa
```

Como en el caso de la toma de decisión simple también se puede utilizar los operadores lógicos y relaciones, además se pueden anidar varios **If** – **Else**.

4.1.3 Select – case (Seleccionar caso)

Es una toma de decisión con varias opciones, esto es, según sea el valor (entero o caracter) de una variable escogerá un caso entre varias alternativas. Su formato es:

```
Select (variable)
        Case expresion1
            instrucciones1
        Case expresion2
            instrucciones2
        ...
         ...
        Case Else:
          instruccionesN
EndSelect
```

Cuando se ejecuta **Select**, se evalúa la **variable** y se busca el primer **Case** que incluya el valor evaluado. Si no existe un valor igual a la **variable** se ejecutá la (s) instrucción (es) a continuación del **Case Else**, si se ha especificado.

La *expresión1*, *expresion2……expresiónN* representan una lista de expresiones que pueden tener cada **Case** y se puede expresar de las siguientes formas:

```
Case Is <y        ‘ variable < y
Case 3            ‘ variable =3
Case y to 10      ‘ variable = y, y+1,…..,10
Case 3, x         ‘ variable = 3, x
Case -5, w To 5           ‘ variable = -1, w, w+1….,5
Case “dato”, “DATO”       ‘ variable =”dato”, “DATO”
Case Is >=200             ‘ variable >=200
```

4.1.4 Ejemplo práctico toma de decisiones

Realizar una aplicación que permita a un usuario capturar tres valores enteros en cajas de texto y visualizar en dos etiquetas el valor mayor y el valor menor digitado por teclado.

NOTA: a partir de este capítulo se omitirán pasos que se supone que el lector ya maneja como son: iniciar un nuevo proyecto, escoger los controles y los pasos para ejecutar el proyecto.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control específico y ubique los siguientes controles en el formulario en la posición deseada: 5 **Label**, 3 **TextBox**, 1 **Button**.

- **Establecer las propiedades de los objetos de la interfaz de usuario**

Para el caso del ejemplo establezca las siguientes modificaciones a los controles:

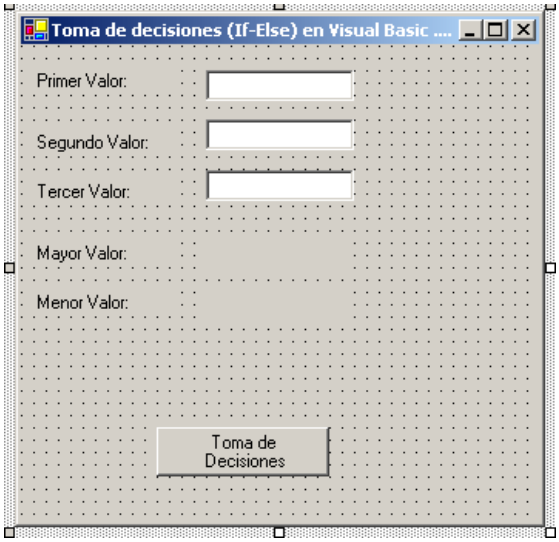
Tabla 4.1 Propiedades de los controles del proyecto TomadeDecisiones.

Nombre del proyecto: TomadeDecisiones		
Control	**Propiedad**	**Valor**
Label1	Name	lblprimer
	Text	Primer valor

Label2	Name	lblsegundo
	Text	Segundo valor
Label3	Name	lbltercero
	Text	Tercer valor
Label4	Name	lblmayor
	Text	Menor valor
Label5	Name	lblmenor
	Text	Mayor valor
Label6….Label7	Name	txtvalor1…txtvalor2
	Text	En blanco
	TextAlign	MiddleCenter
	Font	Sans Serif, 10pt, Negrilla
TextBox1…TextBox3	Name	txtcampo1….txtcampo3
	Text	En blanco
	TextAlign	Center
Button1	Name	boton
	Text	Toma de decisiones
Form1	Name	formulario
	Text	Toma de decisiones (If-Else) en Visual Basic .NET

La interfaz de usuario queda como se muestra en la siguiente figura:

Figura 4.1 Interfaz de usuario (toma de decisiones).



- **Escribir código**

Seleccione el objeto **botón,** abra el editor de código y escriba el siguiente código:

```
Dim valor_uno, valor_dos, valor_tres, mayor, menor As Integer
        valor_uno = txtcampo1.Text
        valor_dos = txtcampo2.Text
        valor_tres = txtcampo3.Text
        If (valor_uno > valor_dos And valor_uno > valor_tres) Then
           mayor = valor_uno
```

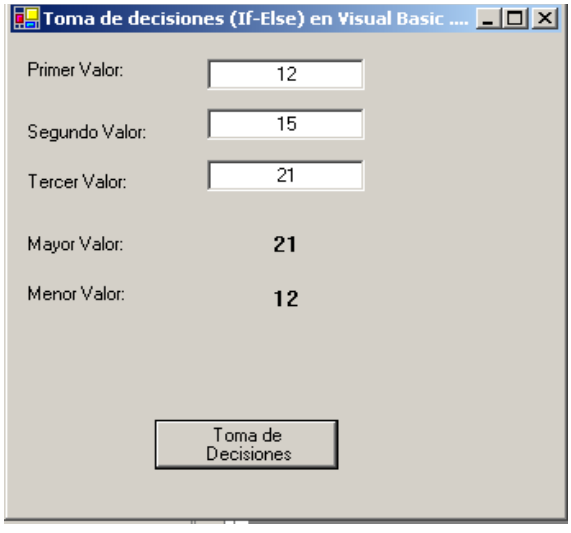
```
    If (valor_dos < valor_tres) Then
       menor = valor_dos
    Else
       menor = valor_tres
    End If
End If
If (valor_dos > valor_uno And valor_dos > valor_tres) Then
    mayor = valor_dos
    If (valor_uno < valor_tres) Then
       menor = valor_uno
    Else
       menor = valor_tres
    End If
End If
If (valor_tres > valor_uno And valor_tres > valor_dos) Then
    mayor = valor_tres
    If (valor_uno < valor_dos) Then
       menor = valor_uno
    Else
       menor = valor_dos
    End If
End If
Txtvalor1.Text = mayor
Txtvalor2.Text = menor
```

En el anterior código se definen las variables **valor_uno**, **valor_dos, valor_tres, mayor**, **menor** de tipo **Integer**. Luego en dichas variables se almacenan los valores de **txtcampo1**, **txtcampo2**, **txtcampo3** respectivamente. Utilizando la sentencia de control **If** se obtiene el mayor valor y se le asigna a la variable **mayor**. Con otra sentencia de control **If** se obtiene el menor valor entre las dos variables resultantes después de hallar el valor mayor y se le asigna a la variable **menor**. Por último a las etiquetas **txtvalor1** y **txtvalor2** en su propiedad **Text** se le asigna el valor de las variables **mayor** y **menor** respectivamente. Cuando el usuario pulse el botón en tiempo de ejecución mostrará en cada etiqueta el valor respectivo que le fue asignado.

- **Ejecutar el proyecto**

Al ejecutarse el proyecto en el entorno de desarrollo de visual Basic.NET, se visualizará la siguiente pantalla:

Figura 4.2 Ejecución aplicación Toma de Decisiones.



4.2 Ciclos (estructuras repetitivas)

Los ciclos permiten repetir varias veces una o más instrucciones, en Visual Basic .NET se usan las estructuras: While, Do…Loop While y For.

4.2.1 While

Sirve para repetir una secuencia de instrucciones siempre que la condición evaluada sea verdadera. Si al evaluarse la condición es falsa no ingresará al ciclo y continuara con el resto del programa. Su formato es:

```
While (condición)
   Instruccion(es)
End While
Resto del programa
```

4.2.2 Do…Loop While

Existen muchas situaciones en las que se desea que un ciclo se ejecute al menos una vez antes de comprobar la condición de repetición. En la estructura **While** si el valor de la expresión booleana es inicialmente falso, las instrucciones del ciclo no se ejecutarán; por ello, se necesitan otros tipos de estructuras repetitivas como **Do… Loop While**, que se ejecuta por lo menos una vez. Su formato es:

```
Do
   Instrucción(es)
Loop While (condición)
```

4.2.3 For

Sirve para repetir una o varias instrucciones, usando una variable que por lo general es llamada **contador**; esta estructura inicializa el contador y evalúa su valor por medio de una condición, si esta es verdadera se ejecutarán las instrucciones del ciclo y aumentará o disminuirá el contador automáticamente, de lo contrario se finalizará el ciclo. Su formato es:

```
For variable =expresion1 To expresión2 Step expresion3
    Instruccion(es)
Next
```

El valor de **variable** es inicializado con el valor que contenga **expresion1**. **Expresion2** representa el valor final que tomara la **variable**. La sentencia **Step** es opcional, por defecto sino existe dicha sentencia el incremento será de uno (1), si por el contrario se utiliza la sentencia **Step** se pueden realizar incrementos o decrementos diferentes de uno (1).

4.3 Ejemplo práctico ciclos

Hacer una aplicación que permita a un usuario capturar 10 valores enteros en un cuadro de diálogo y visualizar en dos etiquetas el mayor y el menor valor de los 10 valores capturados utilizando los ciclos **While, Do…Loop While, For**.

- **Crear la interfaz de usuario**

Seleccione del cuadro de herramientas los siguientes controles: 4 **Label**, 3 **Button**.

- **Establecer las propiedades de los objetos de la interfaz de usuario**

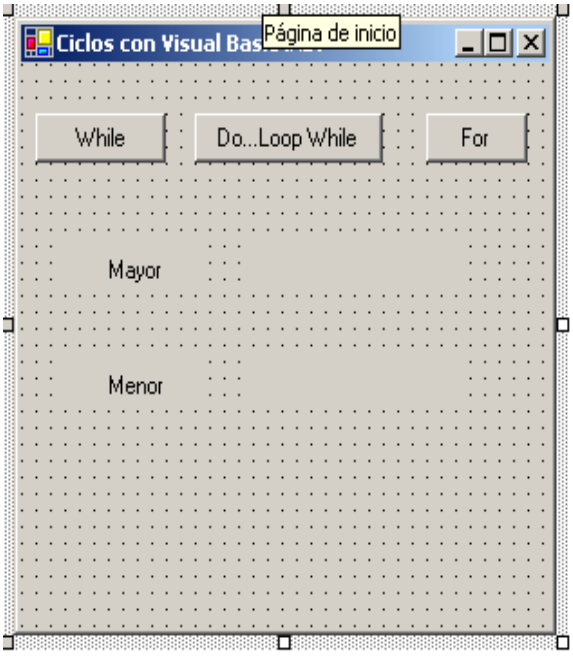
Para el caso del ejemplo establezca los siguientes valores a los controles:

Tabla 4.2 Propiedades de los controles del proyecto Ciclos.

Nombre del proyecto: Ciclos		
Control	**Propiedad**	**Valor**
Label1	Name	lblmayor
	Text	Mayor valor
Label2	Name	lblmenor
	Text	Menor valor
Label3…Label4	Name	txtmayor….txtmenor
	Text	En blanco
	TextAlign	MiddleCenter
	Font	Sans Serif, 10pt, Negrilla
Button1	Name	botonwhile
	Text	While
Button2	Name	botondoloop
	Text	Do..Loop While
Button3	Name	botonfor
	Text	For
Form1	Name	formulario
	Text	Ciclos (While) en Visual Basic .NET

La interfaz de usuario queda como se muestra en la siguiente figura:

Figura 4.3 Interfaz de usuario (Ciclos).



- **Escribir código**

Seleccione el objeto **botonwhile** y abra el editor de código y escriba el siguiente código:

```
Dim mayor, menor, numero, contador As Integer
contador = 1
While (contador < 11)
        numero = InputBox ("Digite numero:")
        If (contador = 1) Then
           mayor = numero
           menor = numero
        End If
         If (mayor < numero) Then
           mayor = numero
        End If
       If (menor > numero) Then
         menor = numero
       End If
       contador += 1
End While
txtmayor.Text = mayor
txtmenor.Text = menor
```

En el anterior código se definen las variables **mayor**, **menor**, **numero**, **contador** de tipo **Integer**. Se inicializa la variable **contador** con un valor de uno (1), crea un ciclo **while** que permitirá ejecutar una serie de instrucciones mientras **contador** sea menor que 11. Dentro del ciclo a la variable **numero** se le asigna el valor capturado en la función predeterminada **InputBox** (cuadro de captura). Esta función permite capturar un valor por teclado; en este caso solamente se utiliza un parámetro para establecer un titulo al cuadro de captura. Cuando se captura el primer valor como la variable **contador** tiene un valor de 1 se ingresa a la primera sentencia de control **If** donde se le asigna a **mayor** y **menor** el valor de la variable **numero**. A dicha instrucción solo se ingresará una vez pues más adelante se incrementa la variable **contador** y por consiguiente obtendría un valor mayor que 1. En las siguientes sentencias se evalúan las variables **mayor** y **menor** respectivamente con la variable **numero** para determinar si a dichas variables se les asigna un nuevo valor. Por último en el ciclo **While** se incrementa la variable **contador** en 1 para que al digitar los 10 valores se termine el ciclo y se le asigne a las etiquetas **txtmayor** y **txtmenor** en su propiedad **Text** los valores de las variables **mayor** y **menor**.

Seleccione el objeto **botondoloop** y abra el editor de código y escriba el siguiente código:

```
Dim mayor, menor, numero, contador As Integer
contador = 1
Do
   numero = InputBox("Digite numero:")
   If (contador = 1) Then
      mayor = numero
      menor = numero
   End If
   If (mayor < numero) Then
      mayor = numero
   End If
   If (menor > numero) Then
      menor = numero
```

```
    End If
    contador += 1
Loop While (contador < 11)
txtmayor.Text = mayor
txtmenor.Text = menor
```

En el anterior código utiliza el ciclo **Do … Loop While**, nótese que las condición va al final del ciclo, lo que permite capturar un valor, evaluarlo con cada una de las sentencias **If**, y por último se evalúa la condición del ciclo, como la condición es verdadera el ciclo sigue ejecutándose.

Seleccione el objeto **botonfor,** abra el editor de código y escriba el siguiente código:

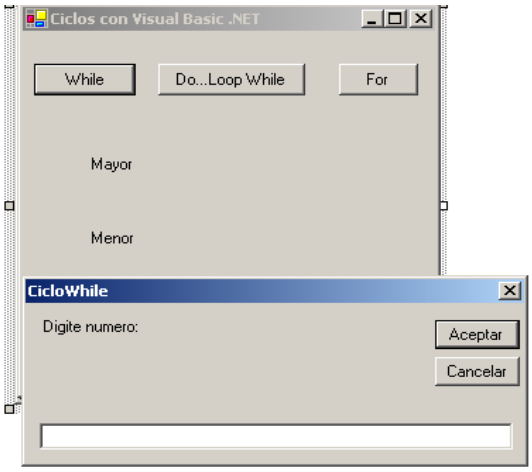
```
Dim mayor, menor, numero, contador As Integer
For contador = 1 To 10
        numero = InputBox("Digite numero:")
        If (contador = 1) Then
            mayor = numero
            menor = numero
        End If
        If (mayor < numero) Then
            mayor = numero
        End If
        If (menor > numero) Then
            menor = numero
       End If
 Next
 txtmayor.Text = mayor
 txtmenor.Text = menor
```

El anterior código utiliza el ciclo **For**, la variable **contador** se incrementa automáticamente y el ciclo solo se evalúa 10 veces.

- **Ejecutar el proyecto**

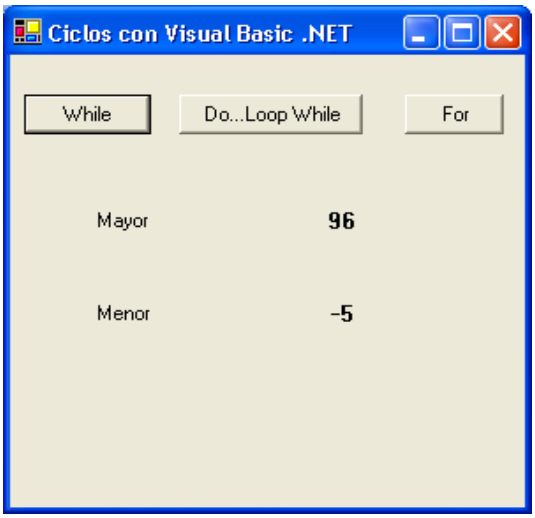
Al ejecutarse el proyecto en el entorno de desarrollo de visual Basic.NET, se visualizará la siguiente pantalla:

Figura 4.4 Ejecución aplicación Ciclos.



Si digita 10, 22, 33, 96, 45, 78, 54, -5, -2, 52, y pulsa uno de los tres botones, se debe visualizar la siguiente pantalla:

Figura 4.5 Resultados de la aplicación Ciclos.



4.3 Ejercicios de estructuras de control

1. Escribir un programa que pida al usuario digitar dos números, e imprima las operaciones con los operadores aritméticos, los operadores relacionales y los operadores lógicos.
2. Realizar un programa que pida al usuario digitar dos números enteros e imprima él numero mayor seguido del texto "Es mayor".
3. Diseñar un programa que capture tres números enteros, e imprima él número mayor, el del medio y el menor.
4. Hacer un programa que convierta una temperatura dada en grados Celsius a grados Fahrenheit. La formula de conversión es F=9/5c +32.
5. Realizar un programa que pida al usuario digitar la hora, los minutos y los segundos e imprima la hora, los minutos y los segundos un segundo después.
6. Capturar 10 números e imprimir el mayor y el menor.
7. Hacer un programa que permita sumar los números enteros de 1 a 100 usando las estructuras: for, while, do loop while.
8. Realizar un programa que imprima todos los números primos entre 2 y 100 inclusive (nota: utilice la propiedad Multiline del control TextBox para imprimir los resultados).
9. Una estación climática proporciona un par de temperaturas diarias (máxima, mínima) (no es posible que alguna o ambas temperaturas sea 0 grados). La pareja fin de temperaturas es 0,0. Se pide determinar el número de días, cuyas temperaturas se proporcionaron, las medias máximas y mínima, el número de errores – temperaturas de 0 grados – y el porcentaje que representan.
10. Los números Astromg o cubos perfectos, son aquellos que sumados los cubos de cada uno de sus dígitos nos dan el mismo número. Por ejemplo 153 es un cubo perfecto, pues (1) elevado a 3 + (5) elevado a 3 + (3) elevado a 3 es igual a 153. Escriba un programa que dado un número entero, diga si es o no es un cubo perfecto.

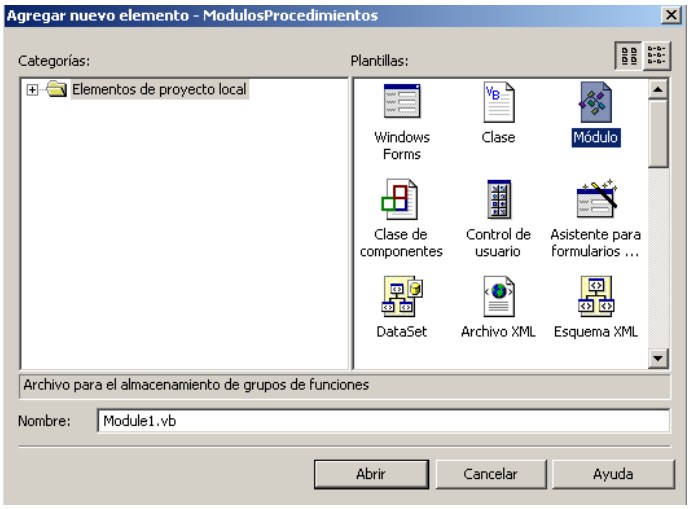
5. MÓDULOS Y PROCEDIMIENTOS

5.1 Módulos

A medida que se escriban programas más complejos, se utilizarán variables y procedimientos similares en varios formularios. Para compartir variables y procedimientos en varios formularios es necesario crear un módulo, este es un contenedor independiente que puede contener variables globales ó públicas, procedimientos **sub** y **function**. Los módulos se visualizarán de forma independiente en el explorador de soluciones y solamente podrán incluir código y no tendrán interfaz de usuario. Para crear un nuevo módulo en una aplicación, se selecciona **agregar nuevo elemento** del menú **Proyecto**, donde se visualizará un cuadro de diálogo que le permitirá seleccionar la **plantilla de modulo** y asignarle un nombre a éste. Por defecto el primer módulo de un programa tendrá como nombre **module1.vb.** Otra forma de realizar el mismo proceso es utilizar el botón **agregar nuevo elemento**  de la barra de herramientas estándar.

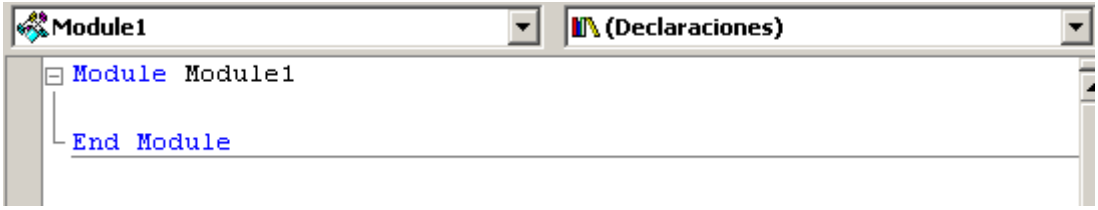
La figura que se visualizará al seleccionar cualquiera de los dos procesos es la siguiente:

Figura 5.1 Agregar un modulo al programa.



Al hacer clic en el botón **Abrir**, Visual Basic .NET agregará al proyecto el módulo llamado **Module1.vb** apareciendo el **Editor de Texto** como se muestra a continuación:

Figura 5.2 Editor Module1.vb.



Luego se podrá observar en el **explorador de soluciones**  , lo siguiente:

Figura 5.3 Módulo en el Explorador de soluciones.

Para declarar una variable pública (aquellas que están disponibles para todos los procedimientos de la aplicación) en un módulo solo se necesita que al inicio del módulo se escriba la palabra clave **public** seguido por el nombre de la variable y el tipo de dato. Una vez declarada esta variable podrá ser utilizada para leerla, modificarla o visualizarla por cualquier procedimiento del programa.

5.2 Procedimientos

En Visual Basic .NET existen dos tipos de procedimientos: **Function** y **Sub**. Los procedimientos **Function** son invocados por su nombre desde otros procedimientos, pueden recibir argumentos (son datos necesarios para que un procedimiento trabaje correctamente y deben ir dentro de paréntesis y separados por comas), y siempre devuelven un valor con el nombre de la función. Los procedimientos **Sub** son invocados por su nombre desde otros procedimientos, pueden recibir argumentos y devolver valores no asociados con el nombre.

Los procedimientos **Function** y **Sub** se pueden definir en el código de un formulario, pero lo más aconsejable es definirlos dentro de un módulo para que todos los elementos de un proyecto puedan utilizarlos. Con esto se ahorra tiempo y trabajo disminuyendo las posibilidades de errores y harán que los programas sean más cortos y sencillos.

Un procedimiento **Function** es un grupo de código localizado entre una instrucción **Function** y **EndFunction**. Se podrá llamar a un procedimiento **Function** desde un programa utilizando el nombre asociado al procedimiento junto con los argumentos necesarios. Su formato es:

```
Function nombre_funcion( argumentos) As tipo de dato
        Instrucciones
        nombre_function = valor a retornar
End Function
```

Ó

```
Function nombre_funcion( argumentos) As tipo de dato
        Instrucciones
        Return = valor a retornar
End Function
```

Para llamar al procedimiento **Function** desde otro procedimiento y suponiendo que se tiene una etiqueta **Label1**, seria de la siguiente forma:

```
Label1.Text = nombre_funcion(argumentos)
```

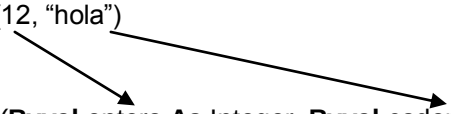
Un procedimiento **Sub** es similar aun procedimiento **Function** solo que **Sub** no retorna un valor asociado con su nombre pero pueden devolver uno o más valores al procedimiento que lo llamo. En la llamada al procedimiento el número y el tipo de argumentos enviados deben ser igual al número y tipo de argumentos del procedimiento. Su formato es:

```
Sub nombre_sub( argumentos)
        Instrucciones
End Sub
```

Para llamar al procedimiento **Sub** desde otro procedimiento se debe especificar el nombre del procedimiento y el número de argumentos. El ejemplo sería el siguiente:
Se envía:

```
miSub(12, “hola”)
```

Se recibe:



```
Mi Sub(Byval entero As Integer, Byval cadena As String)
```

5.2.1 Ejemplo práctico módulo y procedimientos Sub y Function

Realizar una aplicación que permita a un usuario capturar un número y verificar si dicho número es PRIMO ó NO, utilizando procedimientos *Sub* y *Function*.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control específico y ubique los siguientes controles en el formulario en la posición deseada: 1 **TextBox**, 2 **Button**.

- **Establecer las propiedades de los objetos del interfaz de usuario**

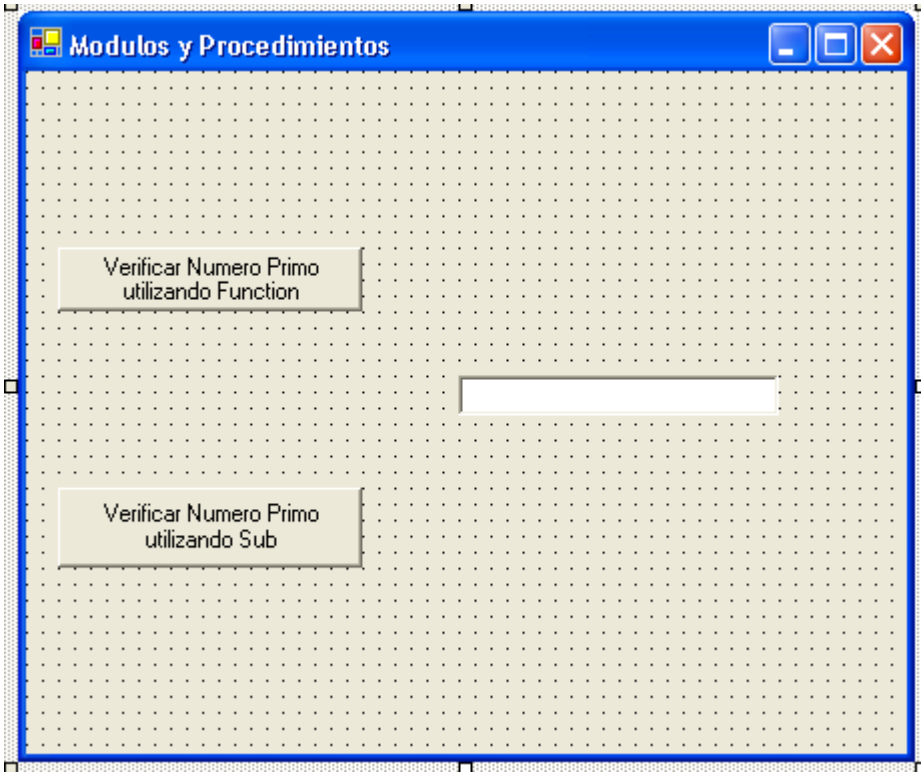
Para el caso del ejemplo establezca las siguientes modificaciones a los controles:

Tabla 4.1 Propiedades de los controles de la aplicación ModulosProcedimientos.

Nombre del proyecto : ModulosProcedimientos		
Control	**Propiedad**	**Valor**
TextBox1	Name	txtvalor
	Text	En blanco
Button1	Name	botonfunction
	Text	Verificar Numero Primo utilizando Function
Button2	Name	botonsub
	Text	Verificar Numero Primo utilizando Sub
Form1	Name	
	Text	módulos y procedimientos

La interfaz de usuario queda como se muestra en la siguiente figura:

Figura 5.4 Interfaz de usuario (módulos y procedimientos).



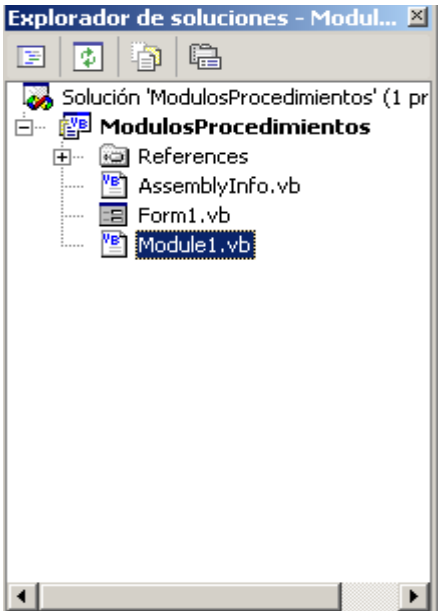
- **Agregar módulo**

Como se dijo anteriormente existen dos formas de agregar un modulo a la aplicación: una forma es seleccionar **agregar nuevo elemento** del menú **Proyecto**, donde se

visualizará un cuadro de diálogo que le permitirá seleccionar la plantilla de modulo y asignarle un nombre al módulo. Por defecto el primer módulo de un programa tendrá como nombre **module1.vb**. La segunda forma es utilizar el botón **agregar nuevo elemento** de la barra de herramientas estándar.

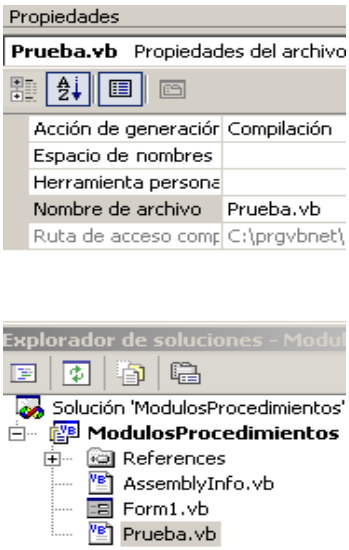
De cualquiera de las dos formas se obtendrá la siguiente figura en el explorador de soluciones

Figura 5.5 Módulo Module1.vb en la aplicación Módulos procedimientos.



Si se desea cambiar el nombre del módulo, se pueden seleccionar las propiedades del módulo dando clic sobre el nombre actual de éste. Como los módulos solo contienen código, las propiedades son mínimas. Una de las propiedades más importantes es **nombre de archivo**, que permite cambiar el nombre del módulo (tenga en cuenta que la extensión debe seguir igual "vb"). Por ejemplo, si se cambia el nombre actual por **prueba.vb** la ventana de propiedades quedaría de la siguiente forma:

Figura 5.6 Módulo con nuevo nombre.



- **Escribir código en el módulo**

En el módulo dé doble clic sobre el nombre de éste y entre las instrucciones **Module** y **End Module** escriba el siguiente código:

```
Module Module1
    Public incremento As Integer
    Public auxiliar As Integer
    Function primo_Function(ByVal numero As Integer) As String
        auxiliar = 0
        incremento = 2
        If (numero > 2) Then
            While (auxiliar = 0)
                If ((numero Mod incremento) = 0) Then
                    primo_Function = "El numero: " & numero & " no es PRIMO"
                    auxiliar = 1
                Else
                    incremento += 1
                End If
                If (numero = incremento) Then
                    primo_Function = "El numero: " & numero & " es PRIMO"
                    auxiliar = 1
                End If
            End While
        Else
            If (numero >= 1 And numero < 3) Then
                primo_Function = "El numero: " & numero & " es PRIMO"
            Else
                primo_Function = "El numero: " & numero & " no es positivo"
            End If
        End If
    End Function
    Sub primo_Sub(ByVal numero As Integer)
        auxiliar = 0
        incremento = 2
        If (numero > 2) Then
            While (auxiliar = 0)
                If ((numero Mod incremento) = 0) Then
                    MsgBox("El numero:" & numero & " no es PRIMO", MsgBoxStyle.Information,
"Utilizando procedimientos Sub")
                    auxiliar = 1
                Else
                    incremento += 1
                End If
                If (numero = incremento) Then
                    MsgBox("El numero: " & numero & " es PRIMO", MsgBoxStyle.Information,
"Utilizando procedimientos Sub")
                    auxiliar = 1
                End If
            End While
        Else
            If (numero >= 1 And numero < 3) Then
                MsgBox("El numero: " & numero & " es PRIMO", MsgBoxStyle.Information,
"Utilizando procedimientos Sub")
            Else
                MsgBox("El numero: " & numero & " no es POSITIVO", MsgBoxStyle.Information,
"Utilizando procedimientos Sub")
            End If
        End If
```

```
End Sub
End Module
```

En el módulo se crean dos variables públicas **auxiliar** e **incremento**, las cuales pueden ser utilizadas por cualquier procedimiento que este dentro del módulo. En este caso dichas variables son utilizadas por los procedimientos **primo_Funtion** y **primo_Sub**. Tanto en el procedimiento **primo_Funtion** como en el procedimiento **primo_Sub** se reciben un número el cual es evaluado para determinar si es primo ó no. En el procedimiento **primo_Funtion** se retorna un **String** al procedimiento que lo llamo, mientras que en el procedimiento **primo_Sub**, el resultado se visualiza por medio de la función predeterminada **MsgBox** (permite mostrar en un cuadro de diálogo mensajes).

- **Escribir código en los controles Button**

Seleccione el objeto **botonfunction** y abra el editor de código y escriba el siguiente código:

```
Dim valor As Integer
 valor = InputBox("Digite un numero:")
txtcampo.Text = primo_Function(valor)
```

En el anterior código se define la variable **valor** de tipo **Integer**. A dicha variable se le asigna el valor capturado en la función predeterminada **InputBox**. Esta función permite capturar un valor por teclado; en este caso solamente se utiliza un parámetro para definirle un titulo. Por último a la propiedad **Text** del control **txtcampo** se le asigna el valor que retorne el procedimiento **primo_Function**. Como se puede apreciar, a dicho procedimiento se le envía como argumento la variable **valor** que es exactamente el número de argumentos que recibe **primo_Function**.

Seleccione el objeto **botonsub** y abra el editor de código y escriba el siguiente código:

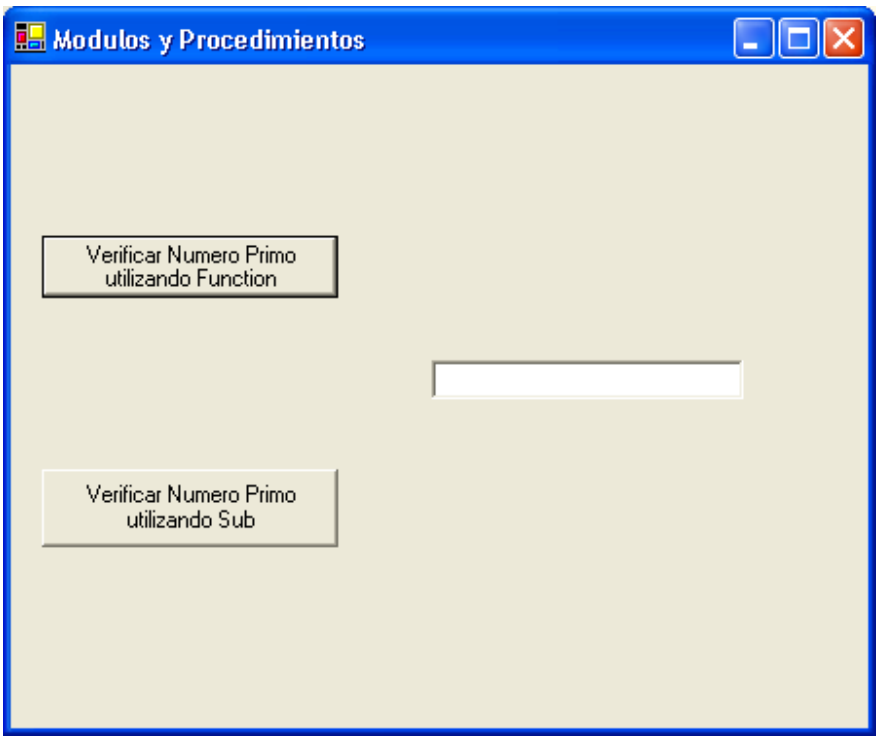
```
Dim valor As Integer
 valor = InputBox("Digite un numero:")
 primo_sub(valor)
```

En este código también se define la variable **valor** de tipo **Integer**. A dicha variable se le asigna el valor capturado en la función predeterminada **InputBox**. Se llama al procedimiento **primo_Sub** y se le envía como argumento la variable **valor** que es el número de argumentos que recibe el procedimiento **primo_Sub**.

- **Ejecutar el proyecto**

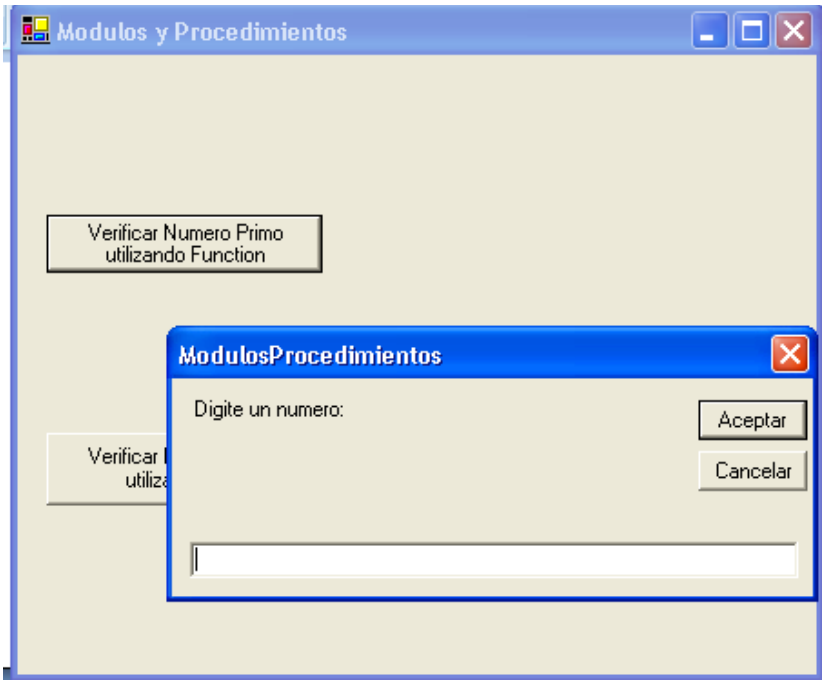
Al ejecutarse el proyecto en el entorno de desarrollo de Visual Basic.NET se visualizará la siguiente pantalla:

Figura 5.7 Ejecución aplicación ModulosProcedimientos.



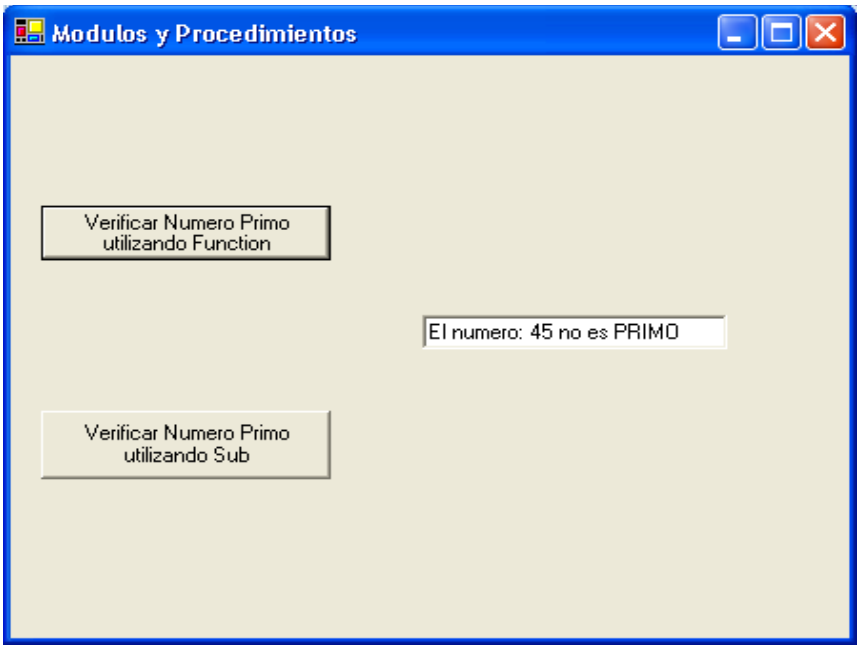
Al pulsar el botón **Verificar Numero Primo utilizando Function** o el botón **Verificar Numero Primo utilizando Sub**, se podrá apreciar en la siguiente figura

Figura 5.8 Opción del número primo.



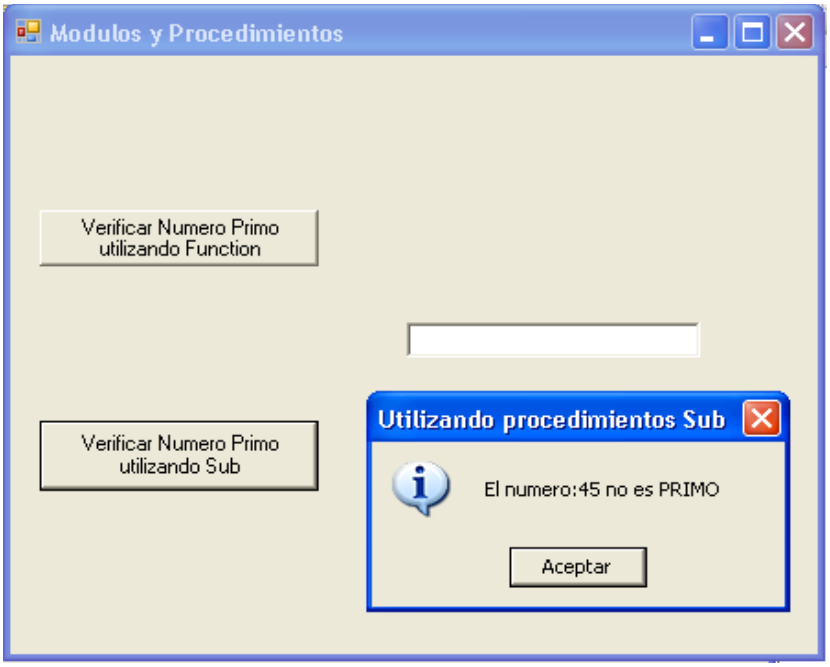
Si se digita el numero 45 y utiliza el botón **Verificar Numero Primo utilizando Function**, se visualizará la siguiente figura:

Figura 5.9 Utilizando el botón Verificar Numero Primo utilizando Function.



Si digita el numero 45 y utiliza el botón **Verificar Numero Primo utilizando Sub**, se visualizará la siguiente figura:

Figura 5.10 Utilizando el botón Verificar Numero Primo utilizando Sub.



5.3 Funciones predeterminadas

En Visual Basic .NET existen funciones predeterminadas o incorporadas que facilitan la realización de tareas específicas, entre ellas se pueden encontrar funciones: matemáticas, graficas, para manipulación de cadenas de caracteres, para tipos de letras, etc.

5.3.1 Funciones Matemáticas

La clase **Math** de Visual Basic .NET contiene una serie de funciones trigonométricas, logarítmicas y otras funciones matemáticas que sirven para realizar cálculos aritméticos, en la siguiente tabla se muestran las funciones más comunes:

Tabla 5.1 Funciones matemáticas de Visual Basic .NET.

Método de Visual Basic .NET	Descripción
Math.Ceiling	Devuelve el número entero más pequeño mayor o igual que el número especificado. Math.ceiling(1.6)=2 Math.ceiling(0.8)=1
Math.Floor	Devuelve el número entero más grande menor o igual que el número especificado. Math.Floor(1.6)=1 Math.Floor(0.8)=0
Math.sqrt	Devuelve la raíz cuadrada de un número.
Math.pow	Devuelve un número especificado elevado a la potencia especificada.
Math.Abs	Devuelve el valor absoluto de un número especificado.
Math.sin	Devuelve el seno del ángulo especificado.
Math.cos	Devuelve el coseno del ángulo especificado.
Math.tan	Devuelve la tangente del ángulo especificado.
Math.Max	Devuelve el mayor de dos números.
Math.min	Devuelve el menor de dos números.
Math.BigMul	Calcula el producto de dos números.
Math.round	Devuelve el número más próximo al valor especificado.

5.3.1.1 Ejemplo práctico funciones matemáticas

Realizar una aplicación que permita a un usuario visualizar en cajas de texto las funciones aritméticas básicas cuando se pulse un botón llamado **Verificar Funciones.**

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control específico y ubique los siguientes controles en el formulario en la posición deseada: 11 **Label**, 11 **TextBox**, 1 **Button**.

- **Establecer las propiedades de los objetos de la interfaz de usuario**

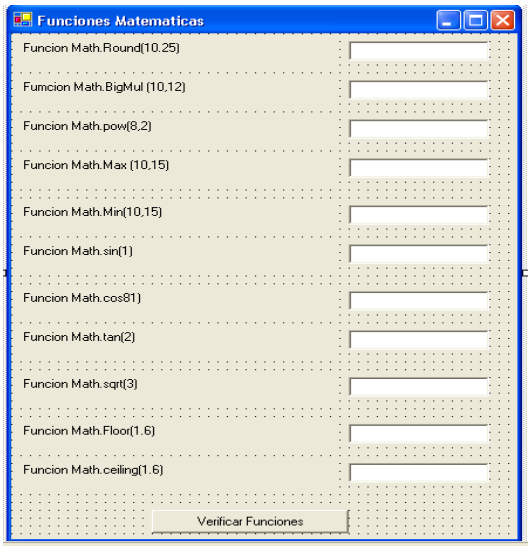
Para el caso del ejemplo establezca las siguientes modificaciones a los controles:

Tabla 5.2 Propiedades de los controles del proyecto Funciones_matematicas.

Nombre proyecto: Funciones_matematicas		
Control	**Propiedad**	**Valor**
Label1	Name	lblround
	Text	Funcion Math.Round(10.25)
Label2	Name	lblbigmul
	Text	Función Math.BigMul (10,12)
Label3	Name	lblpow
	Text	Funcion Math.pow(8,2)
Label4	Name	lblmax
	Text	Funcion Math.Max (10,15)
Label5	Name	lblmin
	Text	Funcion Math.Min(10,15)
Label6	Name	lblsin
	Text	Funcion Math.sin(1)
Label7	Name	lblcos
	Text	Funcion Math.cos81)
Label8	Name	lbltan
	Text	Funcion Math.tan(2)
Label9	Name	lblsqrt
	Text	Funcion Math.sqrt(3)
Label10	Name	lblfloor
	Text	Funcion Math.Floor(1.6)
Label11	Name	lblceiling
	Text	Funcion Math.ceiling(1.6)
TextBox1…TextBox11	Name	txtcampo1…txtcampo7
	Text	En blanco
Button1	Name	boton
	Text	Verificar Funciones
Form1	Name	formulario
	Text	Funciones matemáticas en Visual Basic .NET

La interfaz de usuario quedá como se muestra en la siguiente figura:

Figura 5.11 Interfaz de usuario (Funciones_matematicas).



- **Escribir código**

Seleccione el objeto **boton** y abra el editor de código y escriba el siguiente código:

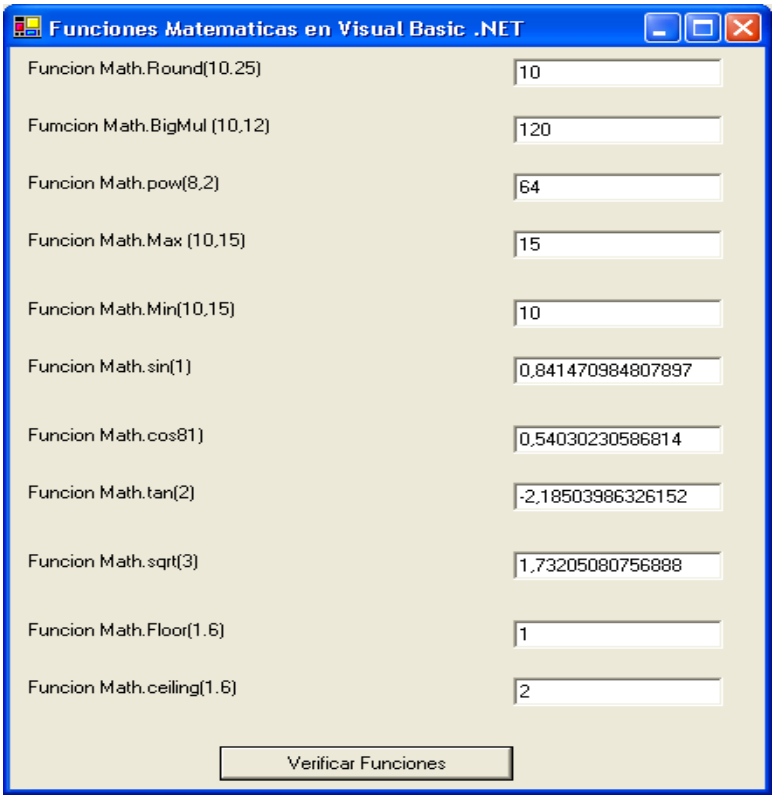
```
txtcampo1.Text = Math.Round(10.25)
txtcampo2.Text = Math.BigMul(10, 12)
txtcampo3.Text = Math.Pow(8, 2)
txtcampo4.Text = Math.Max(10, 15)
txtcampo5.Text = Math.Min(10, 15)
txtcampo6.Text = Math.Sin(1)
txtcampo7.Text = Math.Cos(1)
txtcampo8.Text = Math.Tan(2)
txtcampo9.Text = Math.Sqrt(3)
txtcampo10.Text = Math.Floor(1.6)
txtcampo11.Text = Math.Ceiling(1.6)
```

En el anterior código se asigna una función matemática a la propiedad **Text** de cada caja de texto. Cuando el usuario pulse el botón en tiempo de ejecución mostrará en cada cuadro de texto el valor respectivo de la operación que le fue asignada.

- **Ejecutar el proyecto**

Al ejecutarse el proyecto en el entorno de desarrollo de visual Basic.NET se debe visualizar:

Figura 5.12 Ejecución aplicación Funciones_matematicas.



5.3.2 Funciones de cadenas de caracteres

Visual Basic .NET contiene una serie de funciones para la manipulación de cadenas de caracteres que sirven para realizar diferentes operaciones entre caracteres, en la siguiente tabla se muestran las funciones más comunes:

Tabla 5.3 Funciones para manipulación de cadenas de caracteres.

Método de Visual Basic	Descripción
Chars(n)	Permite obtener un carácter específico de una cadena de caracteres. Dim micadena As String = "ABCDE" Dim micaracter As Char micaracter = micadena.Chars(3) micaracter = "D"
Length	Permite obtener la longitud de de una cadena de caracteres. Dim micadena As String = "ABCDE" Dim Entero As Integer entero=cadena.length entero=5
Concat(texto1, texto,....,texton)	Permite unir dos o más cadenas de caracteres. Dim micadena As String = "hola" Dim unircadena As String unircadena= String.Concat(micadena, " ", "pueblo") unircadena=" hola pueblo"
ToUpper	Convierte una cadena de caracteres de minúscula a mayúscula. Dim micadena As String = "hola" micadena= micadena.ToUpper() micadena=" HOLA"
ToLower	Convierte una cadena de caracteres de mayúscula a minúscula. Dim micadena As String = "HOLA" micadena= micadena.ToLower() micadena=" hola"
Remove(posición inicial, numero de caracteres)	Permite eliminar una cantidad determinada de caracteres en una posición específica de una cadena de caracteres. Dim micadena As String = "campoalegre" micadena= micadena.Remove(2,3) micadena=" caalegre"
Insert (posición inicial, "cadena de caracteres")	Permite insertar una cantidad determinada de caracteres en una posición específica de una cadena de caracteres. Dim micadena As String = "campoalegre" micadena= micadena.insert(4," más ") micadena=" campo más alegre"
SubString((posición inicial, numero de caracteres)	Permite obtener una subcadena de una cadena de caracteres.

	Dim micadena As String = "campoalegre" Dim subcadena As String subcadena= micadena.substring(5,4) subcadena=”aleg”
Replace(cadena de caracteres, carácter original, nuevo carácter)	Permite reemplazar una subcadena determinada por otra subcadena especificada. Dim micadena As String = "campoalegre" Dim Nuevacadena As String nuevacadena =replace**(**micadena, "e", "i"**)** nuevacadena=”campoaligri”
StrReverse(cadena de caracteres)	Devuelve una cadena de caracteres invertida según el orden de los caracteres de la cadena especificada. Dim micadena As String = "campoalegre" Dim Nuevacadena As String nuevacadena =StrReverse**(**micadena**)** nuevacadena=”ergelaopmac”
Mid (cadena de caracteres, posición inicial, numero de caracteres) Mid (cadena de caracteres, posición inicial)	Devuelve una subcadena que a su vez contiene un número especificado de caracteres de una cadena de caracteres. Dim micadena As String = "campoalegre" Dim Nuevacadena As String nuevacadena =Mid**(**micadena,1,5**)** nuevacadena=”campo” Dim micadena As String = "campoalegre" Dim Nuevacadena As String nuevacadena =Mid**(**micadena,5**)** nuevacadena=”alegre”
Len (cadena de caracteres)	Devuelve un entero que contiene el número de caracteres de una cadena de caracteres. Dim micadena As String = "ABCDE" Dim Entero As Integer entero=len(micadena) entero=5
TocharArray	Convierte una cadena de caracteres en un arreglo de caracteres. Dim micadena As String = "campeon" Dim arreglo() As char arreglo=mi cadena.ToCharArray() arreglo={‘c’,’a’,’m’,’p’,’e’,’o’,’n’}

5.3.2.1 Ejemplo práctico funciones de cadena de caracteres

Realizar una aplicación que permita a un usuario visualizar en una caja de texto las funciones de manipulación de cadena de caracteres básicas cuando se pulse un botón específico.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control específico y ubique los siguientes controles en el formulario en la posición deseada: 2 **Label**, 2 **TextBox**, 12 **Button**.

- **Establecer las propiedades de los objetos de la interfaz de usuario**

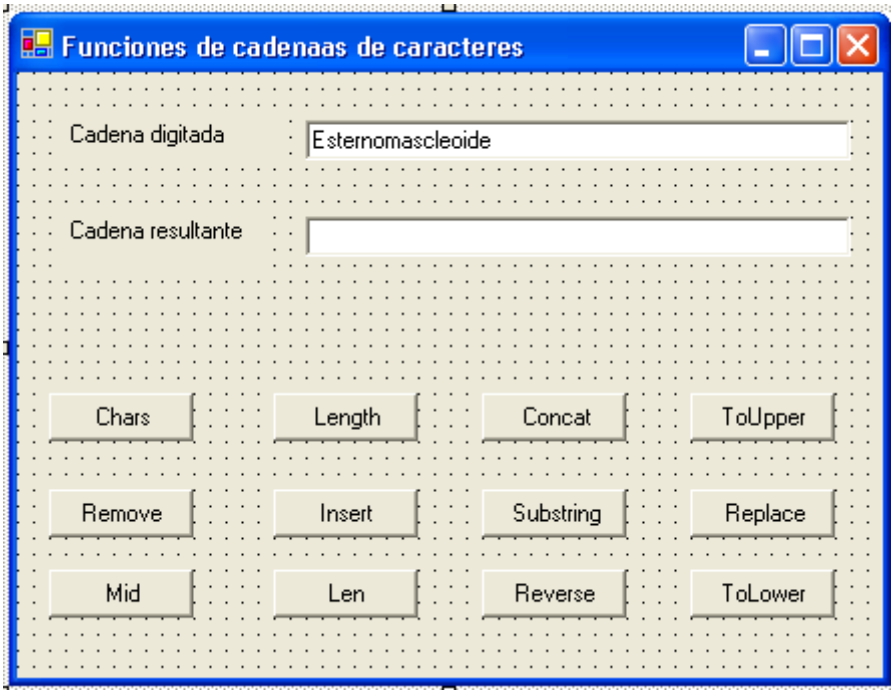
Para el caso del ejemplo establezca las siguientes modificaciones a los controles:

Tabla 5.4 Propiedades de los controles del proyecto cadena_caracteres.

Nombre proyecto: cadenas_caracteres		
Control	**Propiedad**	**Valor**
Label1	Name	lblcadena
	Text	Cadena digitada
Label2	Name	lblresultado
	Text	Cadena resultante
Button1	Name	boton1
	Text	Chars
Button2	Name	boton2
	Text	Length
Button3	Name	boton3
	Text	Concat
Button4	Name	boton4
	Text	toupper
Button5	Name	boton5
	Text	Remove
Button6	Name	boton6
	Text	Insert
Button7	Name	boton7
	Text	Substring
Button8	Name	boton8
	Text	Replace
Button9	Name	boton9
	Text	Mid
Button10	Name	boton10
	Text	Len
Button11	Name	boton11
	Text	Reverse
Button12	Name	boton12
	Text	Tolower
TextBox1	Name	txtcadena
	Text	Esternomascleoide
Textbox2	Name	txtresultante
	Text	En blanco
Form1	Name	formulario
	Text	Funciones de cadenas de caracteres.

La interfaz de usuario queda como se muestra en la siguiente figura:

Figura 5.13 Interfaz de usuario (cadenas_caracteres).



- **Escribir código**

Seleccione el objeto **boton**1 y abra el editor de código y escriba el siguiente código:

```
txtresultado.Text = txtcadena.Text.Chars(2)
```

Seleccione el objeto **boton2** y abra el editor de código y escriba el siguiente código:

```
txtresultado.Text = txtcadena.Text.Length
```

Seleccione el objeto **boton3** y abra el editor de código y escriba el siguiente código:

```
txtresultado.Text = String.Concat(txtcadena.Text, "  ", "que palabrita")
```

Seleccione el objeto **boton4** y abra el editor de código y escriba el siguiente código:

```
txtresultado.Text = txtcadena.Text.ToUpper()
```

Seleccione el objeto **boton5** y abra el editor de código y escriba el siguiente código:

```
txtresultado.Text = txtcadena.Text.Remove(2, 5)
```

Seleccione el objeto **boton6** y abra el editor de código y escriba el siguiente código:

```
txtresultado.Text = txtcadena.Text.Insert(2, "hola")
```

Seleccione el objeto **boton7** y abra el editor de código y escriba el siguiente código:

```
txtresultado.Text = txtcadena.Text.Substring(2, 6)
```

Seleccione el objeto **boton8** y abra el editor de código y escriba el siguiente código:

```
txtresultado.Text = txtcadena.Text.Replace("e", "x")
```

Seleccione el objeto **boton9** y abra el editor de código y escriba el siguiente código:

```
txtresultado.Text = Mid(txtcadena.Text, 2, 8)
```

Seleccione el objeto **boton10** y abra el editor de código y escriba el siguiente código:

```
txtresultado.Text = Len(txtcadena.Text)
```

Seleccione el objeto **boton11** y abra el editor de código y escriba el siguiente código:

```
txtresultado.Text = StrReverse(txtcadena.Text)
```

Seleccione el objeto **boton12** y abra el editor de código y escriba el siguiente código:

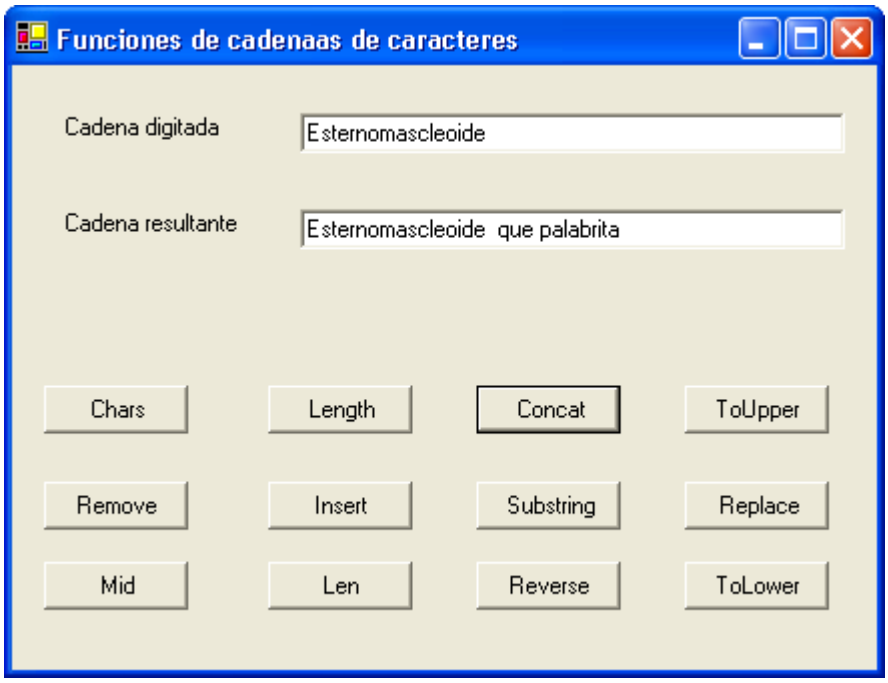
```
txtresultado.Text = txtcadena.Text.ToLower()
```

En el anterior código se le asigna una función de manipulación de cadenas a la propiedad **Text** de cada caja de texto. Cuando el usuario pulse un botón en tiempo de ejecución se mostrará en la caja de texto **txtresultado** el valor respectivo de la operación que le fue asignada.

- **Ejecutar el proyecto**

Al ejecutarse el proyecto en el entorno de desarrollo de visual Basic.NET, se debe visualizar la siguiente figura:

Figura 5.14 Ejecución aplicación cadenas_caracteres.



5.4 Ejercicios de procedimientos

1. Crear un programa que contenga un método llamado perfecto que reciba un parámetro entero e imprima si este es o no perfecto. Se dice que un número perfecto si la suma de sus factores, incluido el 1, da como resultado el número. Por ejemplo, 6 es un número perfecto porque 6=1+2+3**.**
2. Realizar un programa que contenga un método llamado Celsius que reciba un valor representado en grados Celsius y retorne el valor representado en grados Fahrenheit digitado desde el teclado.
3. Hacer un programa que contenga un método que reciba un número entero y retorne el número con sus dígitos invertidos. Ejemplo: dado el número 7631 el método deberá retornar 1367.
4. Escribir un programa que ayude a un estudiante de primaria a aprender a multiplicar. Utilice Math.random para producir dos enteros positivos de un solo digito. El programa deberá exhibir entonces una pregunta en un cuadro de texto como:

   ¿Cuánto es 6 por 7?

   El estudiante tecleará la respuesta en un campo de texto. El programa deberá verificar la respuesta e imprimir **"Muy Bien"** en una etiqueta si la respuesta es correcta y si es equivocada deberá imprimir **"No. Por favor inténtalo de nuevo",** permitiendo que el estudiante conteste una y otra vez hasta que se dé la respuesta correcta. Se usará un método para generar cada pregunta. Este método se debe invocar una vez sé inicializa el programa y cada vez que el usuario conteste la pregunta correctamente.

5. Realizar un programa con un método que calcule las raíces de una ecuación de segundo grado.El discriminante es (b^2- 4*a*c). Se deben tener en cuenta todas las posibles validaciones.
6. Diseñar un programa utilizando un método que determine cuantas cifras posee un número entero positivo introducido por teclado.
7. Leer una cadena de caracteres, digitar el carácter que se quiera eliminar e imprimir la cadena resultante.
8. Crear un programa para imprimir la suma de los números impares menores o iguales que n.
9. Escribir un programa que reciba una cadena de caracteres e imprima cuantas vocales tiene dicha cadena.
10. Hacer un programa utilizando métodos que lea un número no mayor que 1000 e imprima ese número en letras.

6. MATRICES

Una matriz es un grupo de posiciones de memoria contiguas que almacena el mismo nombre y tipo de dato. Para referirse a un elemento de la matriz se debe especificar el nombre de la matriz seguido de uno o más subíndices encerrados entre paréntesis.

6.1 Matrices de una dimensión o unidimensionales

Una matriz de una dimensión se declara de la siguiente forma:

```
Dim <nombre matriz> (<numero de elementos>) As Tipo de dato;
Dim      mercancía              (12)                  As Integer
```

También se puede inicializar una matriz, de la siguiente manera:

```
Dim  ventas ()  As Double = {100.0, 254.5, 478.5, 125.8, 458.66}
Dim   vendedores () As String = {"Julieta", "María", "Jenny", "Sara", "Lina"}
```

Para almacenar valores en una matriz se debe utilizar el nombre de la matriz y un subíndice que indica en qué posición se almacenará el valor especificado. En las siguientes líneas se creará una matriz de 5 posiciones, (el primer valor se almacenará en la posición 0) y se le asignarán valores a la matriz. Por lo general para almacenar o mostrar los valores de una matriz se utilizan los ciclos.

```
Dim ventas (5), i, z As Integer
For i= 0 To 4
   Ventas (i)= i+3
Next
```

Para acceder a los valores de una matriz se debe utilizar un valor de índice para especificar la posición a la que se desea acceder. También se pueden utilizar ciclos para recorrer toda la matriz y acceder a todos los valores.

```
z = ventas (2)   'a z se le asignará el valor de 5 de acuerdo al ejemplo anterior.
```

6.1.1 Ejemplo práctico matrices unidimensionales

Hacer una aplicación que permita a un usuario capturar 10 números enteros positivos o negativos e imprima los valores en el orden que se capturan como también los valores ordenados de menor a mayor.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control específico y ubique los siguientes controles en el formulario en la posición deseada: 2 **TextBox**, 2 **Label**, 1 **Button**.

- **Establecer las propiedades de los objetos del interfaz de usuario**

Después de colocar los controles u objetos en el formulario, establezca las siguientes

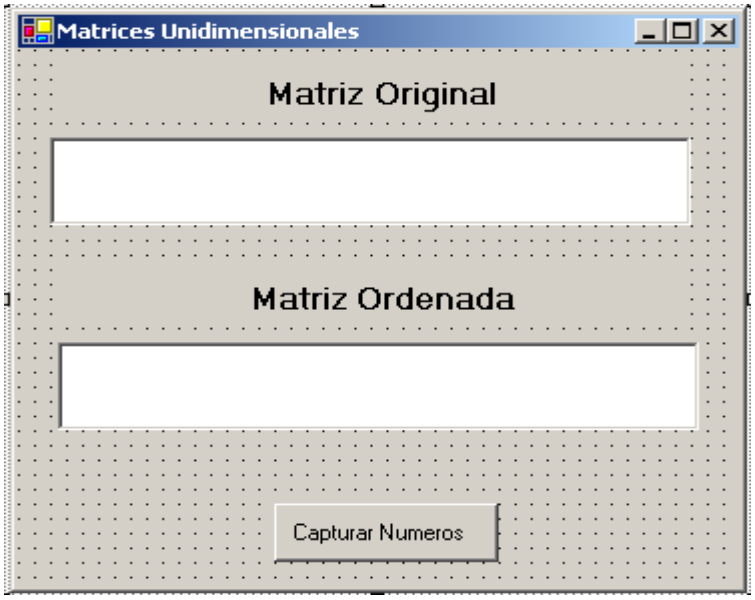
propiedades a los controles:

Tabla 6.1 Propiedades de los controles de la aplicación MatricesUnidimensionales.

Nombre proyecto: MatricesUnidimensionales		
Control	**Propiedad**	**Valor**
TextBox1	Name	txtoriginal
	Text	En blanco
	TextAlign	Center
	Font	Microsoft Sans Serif, 12pt, style=Bold
TextBox2	Name	txtordenado
	Text	En blanco
	TextAlign	Center
	Font	Microsoft Sans Serif, 12pt, style=Bold
Label1	Name	lbloriginal
	Text	Matriz Original
	TextAlign	MiddleCenter
	Font	Microsoft Sans Serif, 12pt, style=Bold
Label2	Name	lblordenado
	Text	Matriz Original
	TextAlign	MiddleCenter
	Font	Microsoft Sans Serif, 12pt, style=Bold
Button1	Name	boton
	Text	Capturar números
Form1	Name	formulario
	Text	Matrices unidimensionales

La interfaz de usuario queda como se muestra en la siguiente figura:

Figura 6.1 Interfaz de usuario (Matrices Unidimensionales).



- **Escribir código**

Seleccione el objeto **boton** y abra el editor de código y escriba el siguiente código:

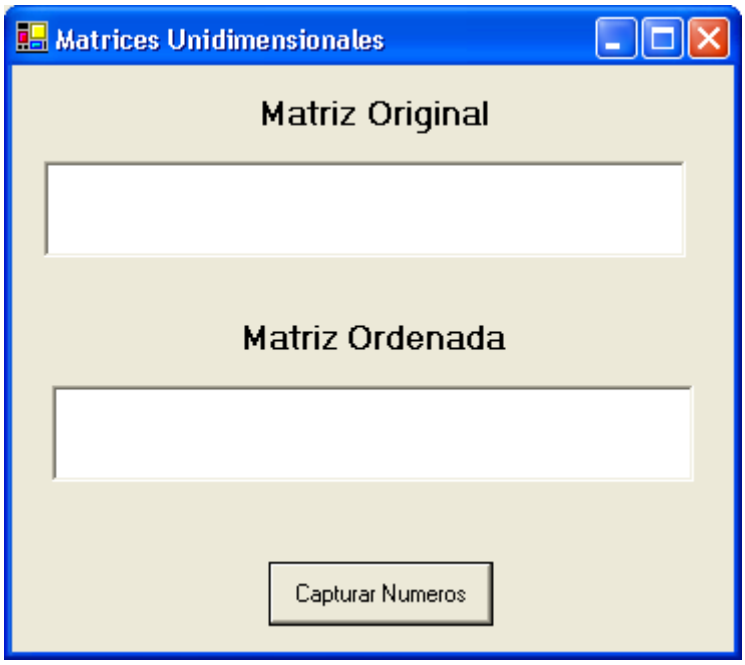
```
Dim datos (10) As Integer
Dim i, j, numero, temporal As Integer
txtoriginal.Text = " "
txtordenado.Text = " "
For i = 0 To 9
    numero = InputBox("Digite un Numero entero")
    datos (i) = numero
    txtoriginal.Text = txtoriginal.Text & datos (i) & " , "
Next
For i = 0 To 9
          For j = i To 9
              If (datos (i) > datos (j)) Then
                  temporal = datos (i)
                  datos (i) = datos (j)
                  datos (j) = temporal
              End If
          Next
Next
For i = 0 To 9
    txtordenado.Text = txtordenado.Text & datos (i) & " , "
Next
```

En el anterior código se define una matriz llamada **datos** la cual permitirá almacenar 10 valores enteros, como también se definen las variables **i, j, numero** de tipo entero. Los controles **txtoriginal** y **txtordenado** se inicializan en su propiedad **Text** en blanco. Se realiza la programación utilizando cuatro ciclos **For**: el primero permite la capturar de cada uno de los valores digitados por teclado y se le asigna a la variable **numero**, luego dicho valor es almacenado en la posición **i** de la matriz y por último al control **txtoriginal** en su propiedad **Text** se le asigna lo que contenía anteriormente ese control más el valor que ha sido almacenado en **datos (i)** agregándole una coma (,). El segundo y tercer ciclo se utilizan para comparar pares de elementos adyacentes e intercambiarlos cuando se cumpla la condición definida dentro del ciclo más interno; este método de comparación es conocido como el método de **búrbuja**. El cuarto ciclo sirve para recorrer la matriz e ir imprimiendo cada uno de los elementos de la matriz en el control **txtordenado** en su propiedad **Text**.

- **Ejecutar el proyecto**

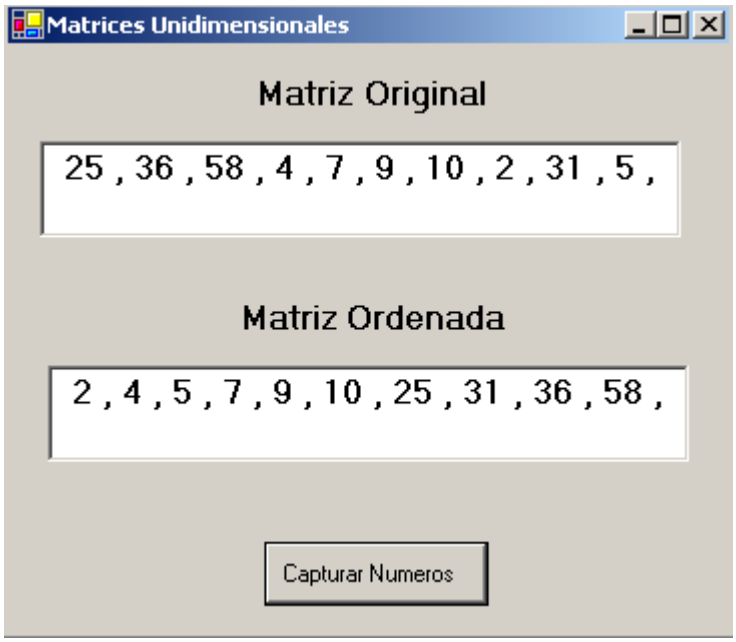
Al ejecutar el proyecto en el entorno de desarrollo de visual Basic.NET se debe visualiza la siguiente pantalla:

Figura 6.2 Ejecución aplicación Matrices Unidimensionales.



Al pulsar el botón **Capturar números** y digitar los números 25,36,58,4,7,9,10,2,31,5, se visualizará los valores como previamente se capturarón y los valores ordenados de menor a mayor. La figura es:

Figura 6.3 Matriz unidimensional al capturar 10 valores.



6.2 Matrices de dos dimensiones (Bidimensionales) o más.

Una matriz puede ser definida de dos, tres o más dimensiones, de cualquier tipo de dato. Para crear una matriz bidimensional en los paréntesis se separa cada dimensión por medio de comas (,). El formato para declarar una matriz bidimensional es:

```
Dim nombre_matriz (filas, columnas) As Integer
Dim notas (2, 3) As Integer
```

Para almacenar valores en la matriz bidimensional se utilizan dos subíndices, el primero indica las filas y el segundo las columnas donde se localiza el valor. El primer valor de las filas y de las columnas es cero (0). En las siguientes líneas se creará una matriz bidimensional de 2 filas y 3 columnas y se le asignarán valores a la matriz. Para almacenar los valores se utilizan dos ciclos.

```
Dim ventas (2, 3), i, j as Integer
For i= 0 to 1
    For j=0 to 2
        ventas (i,j)= i+3
    Next
Next
```

Para acceder a los valores de una matriz bidimensionales, se debe utilizar los dos subíndices, como también se utilizan dos ciclos para recorrer toda la matriz y acceder a todos los valores.

```
Dim ventas (2, 3), i, j, z as Integer
For i= 0 to 1
    For j=0 to 2
        z=ventas (i, j) 'A z se le asignará el valor de la posición (i,j).
    Next
Next
```

6.2.1 Ejemplo práctico matrices bidimensionales

En el siguiente ejercicio se realizará una aplicación que permita a un usuario capturar 6 valores enteros positivos o negativos en una matriz de 2 filas por 3 columnas e imprimir la suma de cada fila y de cada columna.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control específico y ubique los siguientes controles en el formulario en la posición deseada: 6 **Label**, 1 **Button**.

- **Establecer las propiedades de los objetos del interfaz de usuario**

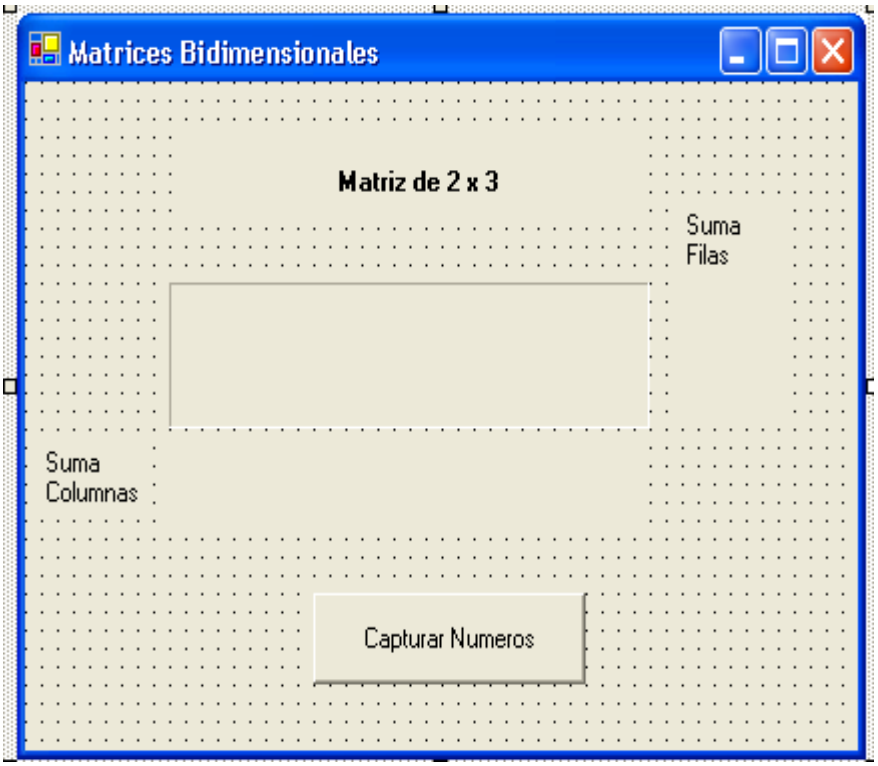
Después de colocar los controles u objetos en el formulario, establezca las siguientes propiedades a los controles:

Tabla 5.2 Propiedades de los controles de la aplicación MatricesBidimensionales.

Control	Propiedad	Valor
Label1	Name	lblnumfila
	Text	En blanco
Label2	Name	lblcolumnas
	Text	Suma Columnas
Label3	Name	lblfilas
	Text	Suma Filas
Label4	Name	lblnumcol
	Text	En Blanco
Label5	Name	lblnumcol
	Text	En Blanco
Label6	Name	lbltexto
	Text	Matriz de 2 x 3
	TextAlign	MiddleCenter
	Font	Microsoft Sans Serif, 8pt, style=Bold
Button1	Name	boton
	Text	Capturar números
Form1	Name	formulario
	Text	Matrices Bidimensionales

La interfaz de usuario queda como se muestra en la siguiente figura:

Figura 6.4 Interfaz de usuario (Matrices Bidimensionales).



- **Escribir código**

Seleccione el objeto **boton** y abra el editor de código y escriba el siguiente código:

```
Dim i, j, numero, sfilas, scolumnas As Integer
lblmatriz.Text = " "
Dim datos(2, 3) As Integer
For i = 0 To 1
    For j = 0 To 2
       datos(i, j) = InputBox("Digite numero")
       lblmatriz.Text = lblmatriz.Text & Space(15) & datos(i, j)
       sfilas = sfilas + datos(i, j)
    Next
    lblnumfila.Text = lblnumfila.Text & sfilas & vbCrLf & vbCrLf
    sfilas = 0
    lblmatriz.Text = lblmatriz.Text & vbCrLf & vbCrLf
Next
For i = 0 To 2
    For j = 0 To 1
        scolumnas = scolumnas + datos(j, i)
    Next
    lblnumcol.Text = lblnumcol.Text & Space(14) & scolumnas
    scolumnas = 0
Next
```

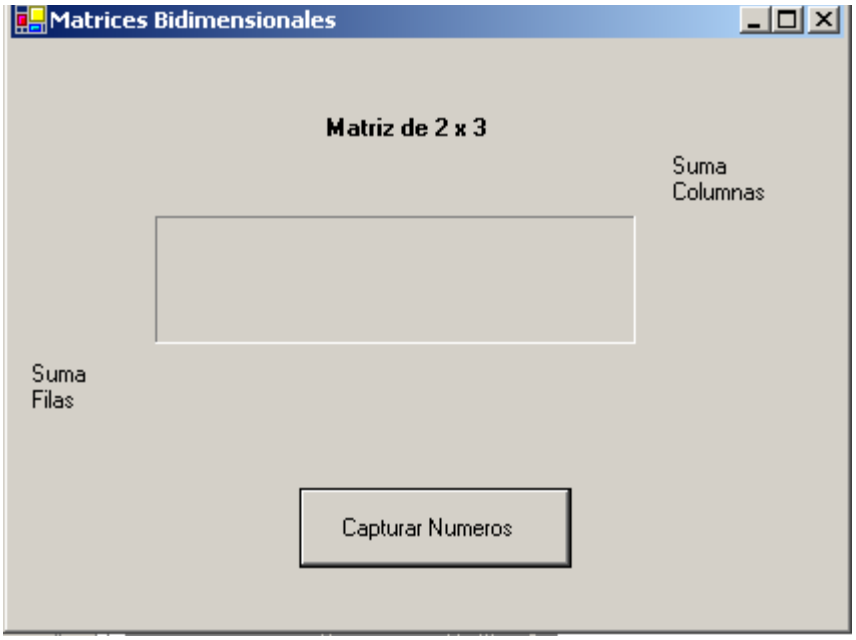
En el anterior código se define una matriz de 2 filas x 3 columnas llamada **datos**, la cual permitirá almacenar 6 valores enteros, también se definen las variables **i, j, numero**, **sfilas, scolumnas** de tipo **Integer**. Se realiza la programación utilizando cuatro ciclos **For**: el primero y el segundo permiten la captura de los valores digitados por teclado lo cual se realiza en el segundo **For**, el valor capturado es asignado a la matriz **datos**, en su posición **i, j.** Al objeto **txtmatriz** en su propiedad **Text** se le asigna el texto que contiene dicho control agregándole 15 espacios en blanco más el valor asignado a la matriz **datos** en su posición **i, j**. La variable **sfilas** se incrementa en cada proceso que realiza el ciclo con el valor de **datos(i,j)**. Al terminar de ejecutarse el segundo **For** se retorna al primer ciclo donde al objeto **txtnumfila** en su propiedad **Text** se le asigna el texto que contiene dicho control agregándole el valor de la variable **sfilas** y se utiliza la función **vbcrlf** (salto de línea) para que realice dos saltos de línea. Por último se reinicia la variable **sfilas** en cero para volver a ingresar al segundo ciclo. Este proceso se ejecutará hasta que la condición sea verdadera en el primer ciclo.

El tercer y cuarto ciclo permite hacer la sumatoria por columnas utilizando la variable **scolumnas**. Al terminar el cuarto ciclo regresa al tercer ciclo y al control **txtnumcol** en su propiedad **Text** se le asigna el texto que contiene dicho control agregándole 14 espacios en blanco más el valor que contiene la variable **scolumnas**. Por último se reinicia la variable **scolumnas** en cero para volver a ingresar al cuarto **For**. Este proceso se ejecutá hasta que la condición sea verdadera en el tercer ciclo.

- **Ejecutar el proyecto**

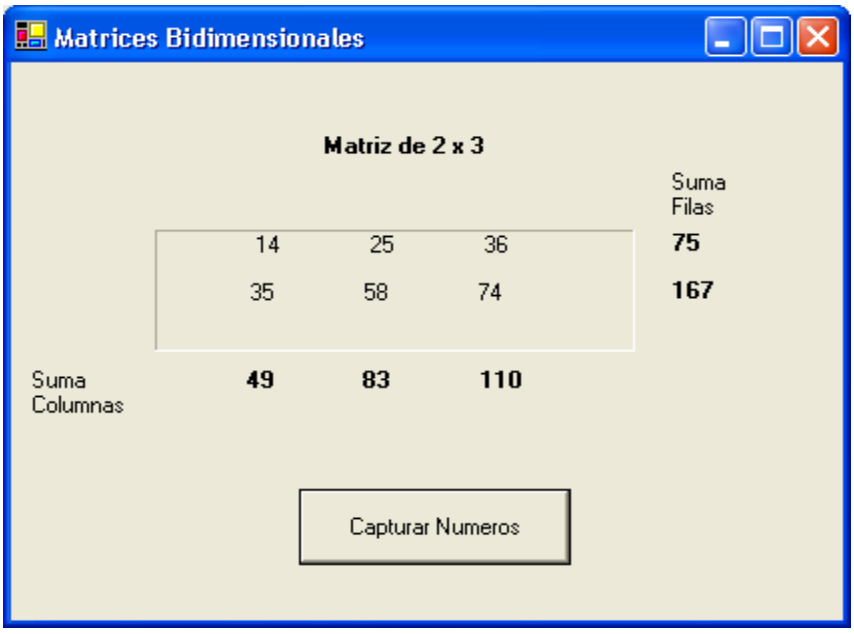
Al ejecutar el proyecto en el entorno de desarrollo de visual Basic.NET se puede ver la siguiente figura:

Figura 6.5 Ejecución aplicación Matrices Bidimensionales.



Al pulsar el botón **Capturar números** y digitar los números 14, 25, 36, 35, 58, 74, se visualizará los valores capturados, así como la respectiva suma por filas y columnas. La figura es:

Figura 6.6 Matriz bidimensional al capturar 6 valores.



6.3 Ejercicios matrices

1. Escribir un programa que permita obtener el número de elementos positivos de un vector de 10 elementos.
2. Calcular el número de elementos negativos, cero y positivos de un vector dado de 20 elementos.
3. Calcular la suma de los elementos de sus diagonales en una matriz cuadrada. Se debe capturar la dimensión de la matriz.
4. Se tiene las notas de 40 alumnos. Cada uno de ellos puede tener una o varias notas. Escribir un programa que permita obtener la media de cada alumno y la media de la clase a partir de la entrada de las notas desde el teclado.
5. Se dispone de una lista de 100 números enteros. Calcular su valor máximo y el orden que ocupan en el vector.
6. Hacer un programa que le permita insertar o eliminar elementos de un arreglo (los elementos deben mostrarse como se ingresaron).
7. Leer una matriz cuadrada A. Calcular si la matriz es simétrica. Se considera una matriz simétrica si A [i, j]=A [j, i] y esto se cumple para todos los elementos i, j de la matriz.
8. Sean A[m,n] y B[n] una matriz y un arreglo respectivamente, hacer un programa que asigne valores a A, a partir de B teniendo en cuenta los siguientes criterios:
   A[i,j]=[i] si i<=J
   A[i,j]=0 si i>j
9. Hacer un programa que intercambie las filas de una matriz. Los elementos de la fila 0 deben intercambiarse con los de la fila N, la fila 1 con los de la fila N-1 y así sucesivamente. Imprimir las dos matrices.
10. Se tiene un arreglo unidimensional C de N elementos, calcular:
    El numero de datos repetidos en el arreglo
    El numero de valores impares
    El numero de valores pares
    La cantidad de ceros.

7. INTERFAZ DE USUARIO AVANZADA

Hasta ahora se ha trabajo la interfaz de usuario con controles básicos del cuadro de herramientas como son los objetos **Label**, **TextBox** y **Button**. En este capítulo se abordara la programación de algunos de los controles que más comúnmente se utilizan en Visual Basic.NET. De forma transparente para el usuario Visual Basic .NET crea el código que interactúa con cada control.

7.1 Control LinkLabel

El control LinkLabel  permite que un texto contenido en un formulario de Visual Basic .NET se comporte como un vinculo de Internet o para ejecutar aplicaciones de Windows.

7.1.1 Ejemplo práctico control LinkLabel

Realizar una aplicación que permita a un usuario por medio de un control LinkLabel conectarse a una página Web y con otro control LinkLabel pueda abrir una aplicación de Windows.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control LinkLabel y ubique 2 controles en el formulario en la posición deseada. La figura 7.1 muestra la interfaz de usuario utilizando el control LinkLabel

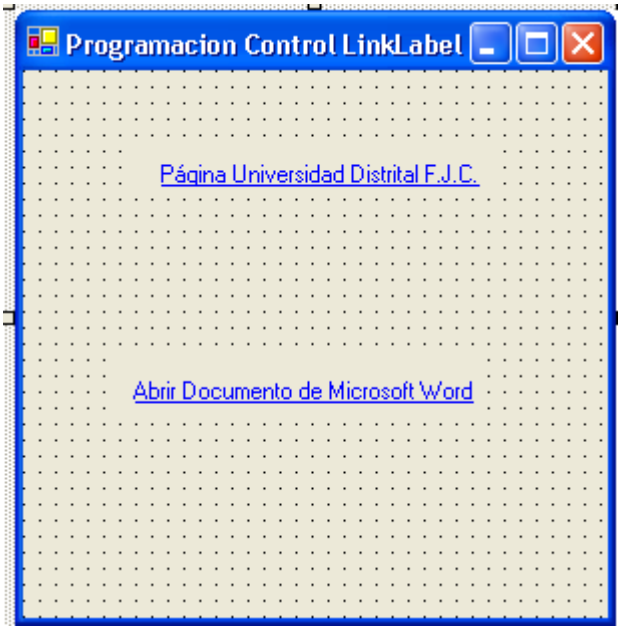
- **Establecer las propiedades de los objetos del interfaz de usuario**

Después de colocar los controles u objetos en el formulario, establezca las siguientes propiedades a los controles:

Tabla 7.1 Propiedades de los controles de la aplicación ControlLinkLabel.

Nombre del proyecto: ControlLinkLabel		
Control	**Propiedad**	**Valor**
LinkLabel1	Name	paginaweb
	Text	Página Universidad Distrital J.F.C.
	TextAlign	MiddleCenter
LinkLabel2	Name	programwindows
	Text	Abrir Documento de Microsoft Word
	TextAlign	MiddleCenter
Form1	Name	formulario
	Text	Programación Control LinkLabel

Figura 7.1 Interfaz de usuario (ControlLinkLabel).



- **Escribir código**

Seleccione el objeto **paginaweb**, de doble clic para abrir el editor del procedimiento *Paginaweb _LinkClicked* y escriba el siguiente código:

```
Paginaweb.LinkVisited = True
System.Diagnostics.Process.Start("http://www.udistrital.edu.co")
```

En la primera línea de código se asocia el valor **true** a la propiedad **Linkvisited** lo que permite visualizar de color morado el texto del vínculo para indicar que ya se ha visitado la página de éste vínculo. En la segunda línea se utiliza el método **start** de la clase **Process** que administra la ejecución de un programa utilizando el espacio de nombres **System.Diagnostics**. Al incluir la dirección del sitio de Internet o la URL[3] a continuación del método **start** se le está diciendo a Visual Basic .NET que se desea ver el sitio Web especificado.

Seleccione el objeto **programawindows**, de doble clic para abrir el editor del procedimiento *programawindows_LinkClicked* y escriba el siguiente código:

```
programawindows.LinkVisited = True
System.Diagnostics.Process.Start("Winword.exe",c:\Programas_vb.net.doc")
```

Una característica del método **start** de la clase **Process** es que se puede utilizar

---

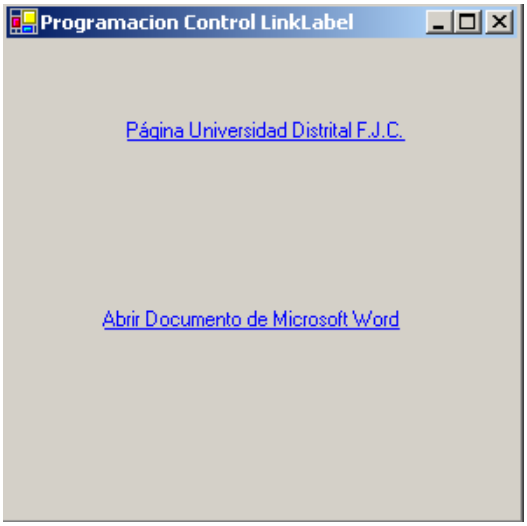
[3] **URL** significa ***Uniform Resource Locator***, es decir, localizador uniforme de recurso. Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.

para ejecutar diferentes aplicaciones. El método **start** recibe dos argumentos: el primer argumento se utiliza para ejecutar una aplicación específica sin necesidad de configurar la ubicación de la aplicación ya que Visual Basic. NET busca la ruta adecuada para ejecutar el programa. En el segundo argumento se define el documento que se quiere abrir con su respectiva ruta. En este caso se busca ejecutar el **WinWord** de **Microsoft** y abrir un archivo que existe en la raíz del disco de trabajo llamado **Programas_vb.net.doc.**

- **Ejecutar el proyecto**
-

Al realizar la ejecución del proyecto en el entorno de desarrollo de visual Basic.NET, se debe visualizar la siguiente pantalla:

Figura 7.2 Ejecución aplicación ControlLinkLabel.

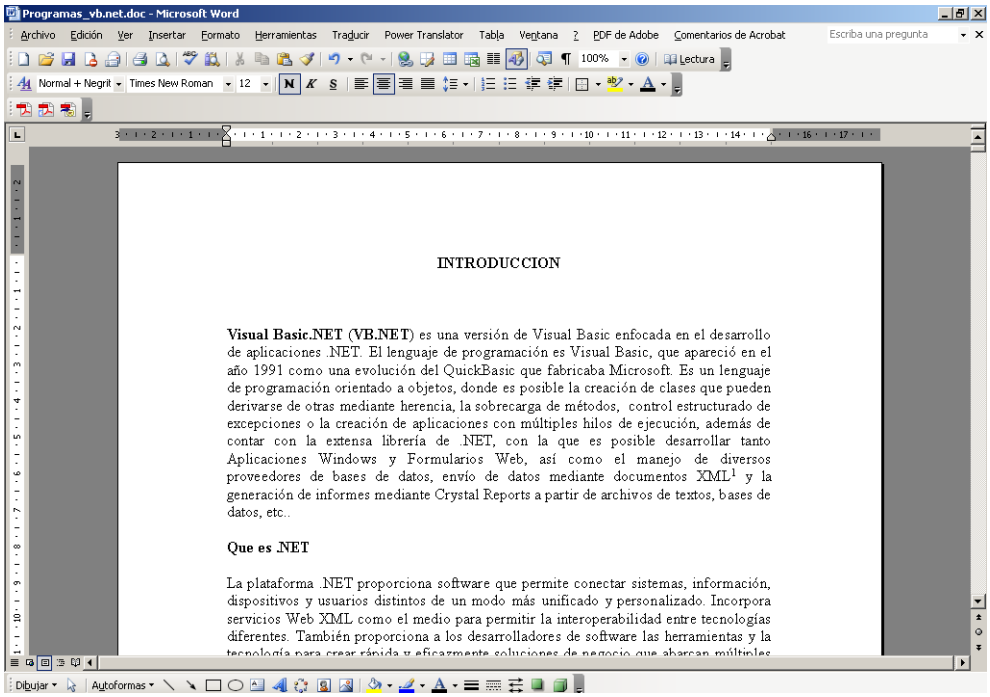


Al pulsar el control con texto **Página Universidad Distrital F.J.C.** se visualizará la siguiente figura:

Figura 7.3 Ejecución Control LinkLabel (abrir página Web).

Al pulsar el control con texto **Abrir Documento de Microsoft Word** se visualizará la siguiente figura:

Figura 7.4 Ejecución Control LinkLabel (abrir aplicación de Windows).



7.2 Controles ListBox y ComboBox

Un control **ListBox** muestra una lista de elementos de los cuales el usuario puede seleccionar uno o más. Si el número total de elementos supera el número que se puede mostrar dentro del control se agregará automáticamente una barra de desplazamiento al control**.**

Un control **ComboBox** al igual que el control ListBox permite definir una lista de elementos donde el usuario puede seleccionar un elemento o más. Al momento de la ejecución del programa, toda la lista de elementos estarán a la vista del usuario para que éste los seleccione. Para seleccionar un elemento deberá pulsar la flecha que está al lado derecho del encabezado.

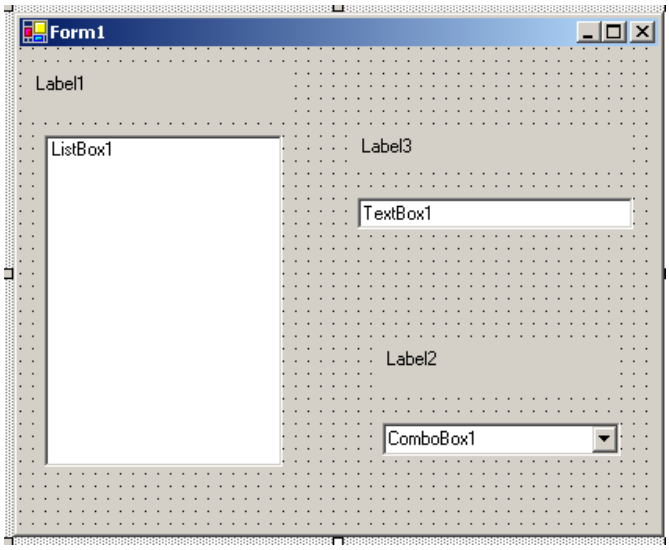
7.2.1 Ejemplo práctico controles ListBox y ComboBox

Hacer una aplicación que permita a un usuario por medio de un control ListBox seleccionar un país y ver su capital, y por medio de un control ComboBox seleccionar una capital y visualizar el país en un campo de texto.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control ListBox y ubique el control en el formulario en la posición deseada. También seleccione un Combo Box y ubique en el formulario. Además agregue 3 **Label** y 1 **TextBox**. La figura 7.5, muestra la interfaz de usuario utilizando los controles ListBox y ComboBox.

Figura 7.5 Interfaz de usuario (ListBox, ComboBox).



- **Establecer las propiedades de los objetos del interfaz de usuario**

Después de colocar los controles u objetos en el formulario, establezca las siguientes propiedades a los controles:

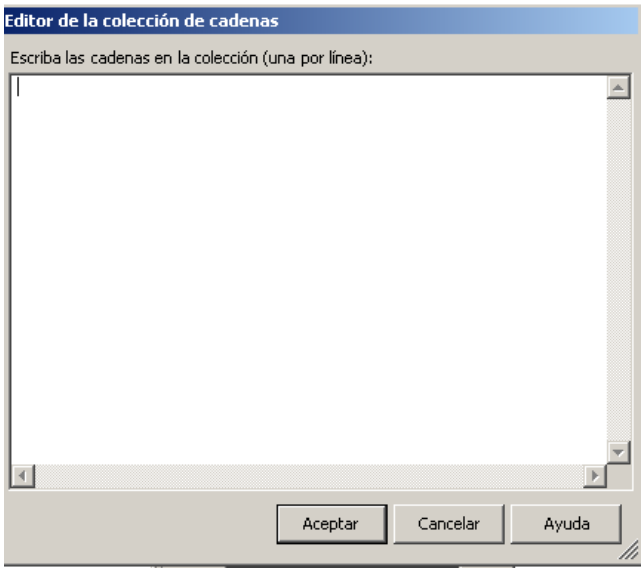
Tabla 7.2 Propiedades de los controles de la aplicación ListBoxComboBox.

Nombre del proyecto : ListboxComboBox		
Control	**Propiedad**	**Valor**
TextBox	Name	respuesta
	Text	En blanco
Label1	Name	txtpais
	Text	Escoja el País y vea su Capital
Label2	Name	txtcapital
	Text	Escoja la Capital y vea el País
Label3	Name	txtrespuesta
	Text	Se muestra la respuesta deseada
ComboBox1	Name	listacapitales
	Text	Capìtales
ListBox1	Name	listapaises
Form1	Name	formulario
	Text	Controles ListBox y ComboBox

A los controles ListBox y ComboBox se les puede agregar elementos de dos formas: la primera utilizando la propiedad **Ítems** y la segunda por medio de código desde el procedimiento **Load** del formulario. Para el ejemplo, se agregará elementos utilizando las dos formas, la primera desde este ítem y la segunda desde el ítem **Escribir Código.**

Entonces se van a agregar 4 elementos a cada control, en este ejemplo, solo se agregan al control **ListBox** ya que para el control **ComboBox** se hace de la misma forma. Lo primero es seleccionar el control **listapaises** y hacer clic sobre la propiedad **Ítems** la cual visualizará la siguiente figura:

Figura 7.6 Editor para agregar elementos a un ListBox o ComboBox.



En este caso se escribe el nombre de cuatro países, uno por cada línea:

Colombia

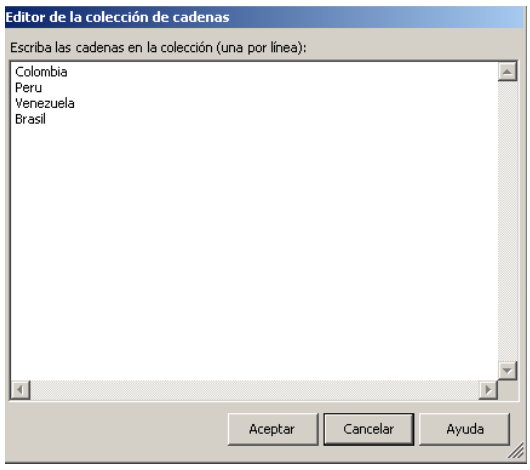
Perú

Venezuela

Brasil

Por último dé clic en el botón **Aceptar** para que los elementos se adicionen al control. Este mismo proceso se debe realizar para el control **listaciudades** para agregar los siguientes elementos: Lima, Buenos Aires, Montevideo, Quito. Los elementos del **ListBox** se muestran en la siguiente figura:

Figura 7.7 Editor con los elementos agregados al control ListBox.



- **Escribir código**

Seleccione el objeto **Formulario**, dé doble clic para abrir el editor de código **`Formulario_Load`** para escribir el código necesario para agregar los demás elementos en los controles **ListBox** y **ComboBox**.

```
listapaises.Items.Add("Argentina")
listapaises.Items.Add("Bolivia")
listapaises.Items.Add("Uruguay")
listapaises.Items.Add("Ecuador")
listacapitales.Items.Add("Bogotá")
listacapitales.Items.Add("La Paz")
listacapitales.Items.Add("Rio de Janeiro")
listacapitales.Items.Add("Caracas")
```

Para adicionar más elementos a los controles **ListBox** y **ComboBox** se utiliza el método **Add** de la propiedad **Ítems** y entre comillas dobles (“”) se escribe el elemento que se desea agregar.

Seleccione el objeto **listapaises**, dé doble clic para abrir el editor de código `listapaises_SelectedIndexChanged` y escriba el siguiente código:

```
respuesta.Text = ""
Select Case listapaises.SelectedIndex
   Case 0
      respuesta.Text = "Bogotá"
   Case 1
      respuesta.Text = "Lima"
   Case 2
      respuesta.Text = "Caracas"
   Case 3
      respuesta.Text = "Rio de Janeiro"
   Case 4
      respuesta.Text = "Buenos Aires"
   Case 5
      respuesta.Text = "La Paz"
   Case 6
      respuesta.Text = "Montevideo"
   Case 7
      respuesta.Text = "Quito"
End Select
```

Al adicionar elementos a un control **ListBox** ó **ComboBox**, el primer elemento tendrá como valor de índice cero (0), el segundo el valor de índice uno (1) y así sucesivamente hasta el elemento *n*. En el anterior código se inicializa la propiedad **Text** del control **respuesta** en blanco, luego por medio de la propiedad **SelectedIndex** (selecciona el índice) del control **listapaises** se obtiene el valor del índice que el usuario ha seleccionado para luego buscar el caso especifico y asignarle el valor del caso a la propiedad **Text** del control **respuesta**.

Luego se debe seleccionar el objeto **listacapitales**, dar doble clic para abrir el editor de código `listacapitales_SelectedIndexChanged` y escribir el siguiente código:

```
respuesta.Text = ""
Select Case listacapitales.SelectedIndex
   Case 0
     respuesta.Text = "Peru"
   Case 1
     respuesta.Text = "Argentina"
```

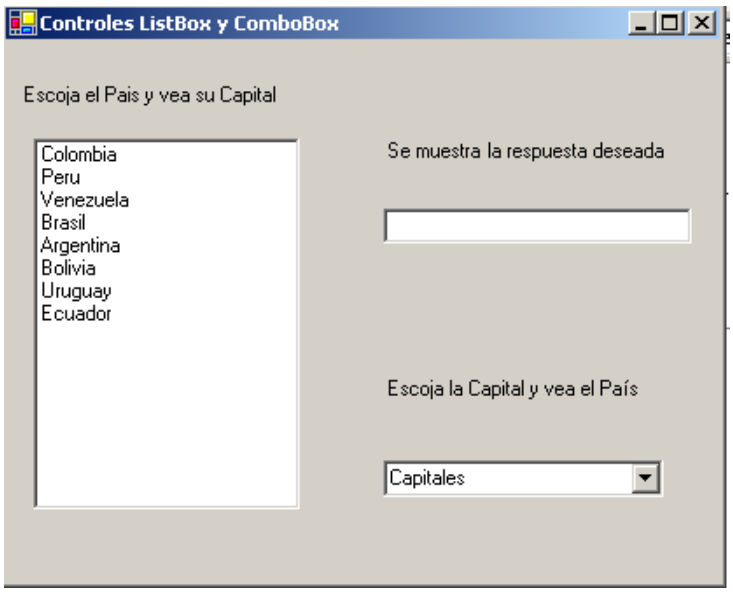
```
    Case 2
      respuesta.Text = "Uruguay"
    Case 3
      respuesta.Text = "Ecuador"
    Case 4
      respuesta.Text = "Colombia"
    Case 5
      respuesta.Text = "Bolivia"
    Case 6
      respuesta.Text = "Brasil"
    Case 7
      respuesta.Text = "Venezuela"
End Select
```

En este procedimiento se realiza la misma programación con el fin de retornar el valor deseado que el usuario selecciono.

- **Ejecutar el proyecto**

Cuando se ejecuta el proyecto se visualiza la siguiente figura:

Figura 7.8 Ejecución aplicación ListBoxComboBox.



Al dar clic en el **ListBox** en el país **Colombia**, en el cuadro de texto se debe mostrar como su capital a **Bogotá**, visualizándose la siguiente figura:

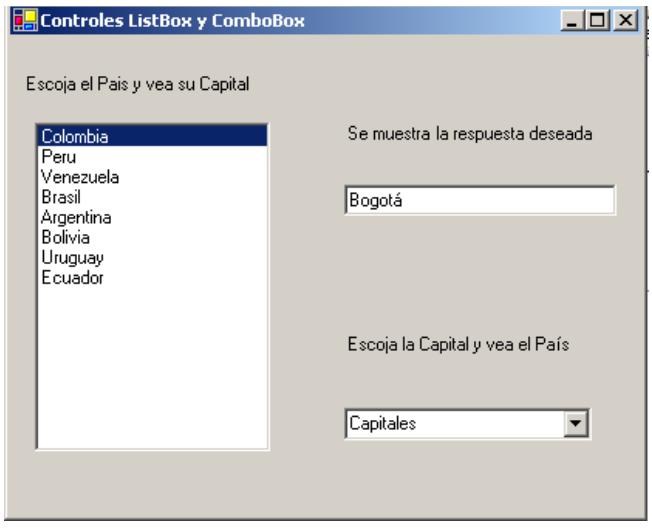
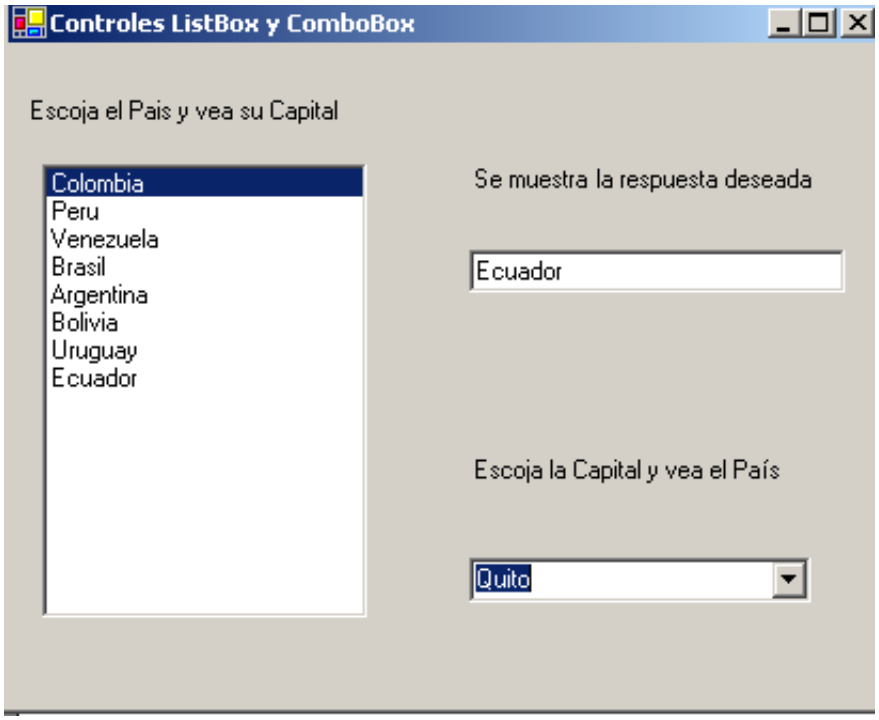


Figura 7.9 Selección de un país desde el ListBox.

Al dar clic en el **ComboBox** y seleccionar la capital **Quito**, en el cuadro de texto se debe mostrar como su país a **Ecuador**, visualizándose la siguiente figura:

Figura 7.10 Seleccionar una capital desde el ComboBox.



7.3 Controles VScrollBar, HScrollBar, TrackBar

Un control **VScrollBar**  es una barra de desplazamiento que permite manipular rangos de valores numéricos por medio de su propiedad **Value**, los cuales son establecidos por las propiedades **maximum** *(*máximo valor*)* y **minimum** (mínimo valor*)*. El control HScrollBar  realiza la misma función, sólo se diferencia en su forma de presentación. Estos controles son utilizados para implementar el

desplazamiento en los contenedores que no proporcionan sus propias barras de desplazamiento, como **PictureBox**.

El objeto **TrackBar**  es un control desplazable similar al control **ScrollBar**. Para configurar los intervalos entre los que se desplaza el valor de la propiedad **Value** de una barra de seguimiento, estableciendo la propiedad **Minimum** para especificar el extremo inferior del intervalo y la propiedad **Maximum** para especificar el extremo superior del intervalo.

7.3.1 Ejemplo práctico controles VScrollBar y TrackBar

Realizar una aplicación que permita a un usuario por medio de controles VScrollBar y TrackBar visualizar un color al realizar un desplazamiento en cualquiera de los controles.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control **VScrollBar** y ubique 3 controles en el formulario en la posición deseada, además, seleccione el control **TrackBar** y ubique 3 controles en el formulario. También seleccione el control **Label** y ubique 7 controles en el formulario. La figura 7.11., muestra la interfaz de usuario.

- **Establecer las propiedades de los objetos de la interfaz de usuario**

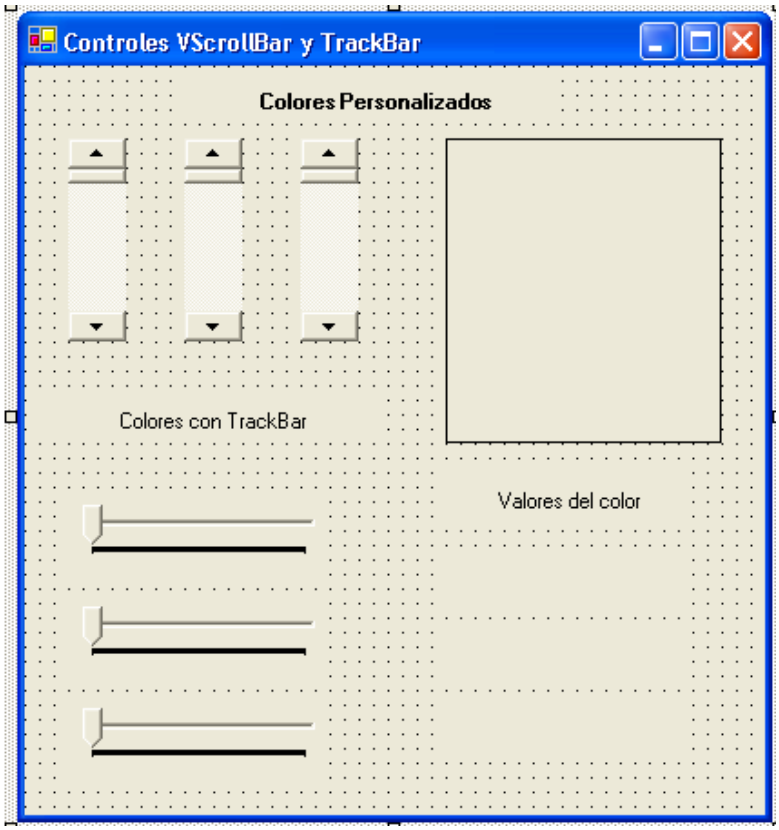
Después de colocar los controles u objetos en el formulario, establezca las siguientes propiedades a los controles:

Tabla 7.3 Propiedades de los controles de la aplicación BarrasdeDesplazamiento.

Nombre del proyecto :		
Control	**Propiedad**	**Valor**
VScrollBar1, VScrollBar2, VScrollBar3.	Name	barra1, barra2, barra3
	Maximum	255
	Minimum	0
TrackBar1, TrackBar2, TrackBar3	Name	desplazamiento1, desplazamiento2, desplazamiento3
	Maximum	255
	Minimum	0
Label1	Name	txtcolores
	Text	En blanco
	BorderStyle	FixedSingle
Label2	Name	txttitulo
	Text	Colores Personalizados
	TextAlign	MiddleCenter
	Font	Microsoft Sans Serif, 8pt, style=Bold
Label3	Name	txttrackbar
	Text	Colores con TrackBar
	TextAlign	MiddleCenter
Label4	Name	txtvalores
	Text	Valores del color
	TextAlign	MiddleCenter
Label5	Name	color1
	Text	En blanco
Label6	Name	color2

	Text	En blanco
Label7	Name	color3
	Text	En blanco
Form1	Name	formulario
	Text	Controles VScrollBar y TrackBar

Figura 7.11 Interfaz de usuario (VScrollBar, TrackBar).



- **Escribir código**

Seleccione el objeto **barra1**, dé doble clic para abrir el editor del procedimiento `barra1_Scroll` y escriba el siguiente código:

```
txtcolores.BackColor = Color.FromArgb(barra1.Value, barra2.Value, barra3.Value)
color1.Text = "Azul=" & barra3.Value
```

Seleccione el objeto **barra2**, dé doble clic para abrir el editor del procedimiento `barra2_Scroll` y escriba el siguiente código:

```
txtcolores.BackColor = Color.FromArgb(barra1.Value, barra2.Value, barra3.Value)
color2.Text = "Verde=" & barra2.Value
```

Seleccione el objeto **barra3**, dé doble clic para abrir el editor del procedimiento `barra3_Scroll` y escriba el siguiente código:

```
txtcolores.BackColor = Color.FromArgb(barra1.Value, barra2.Value, barra3.Value)
color3.Text = "Rojo=" & barra1.Value
```

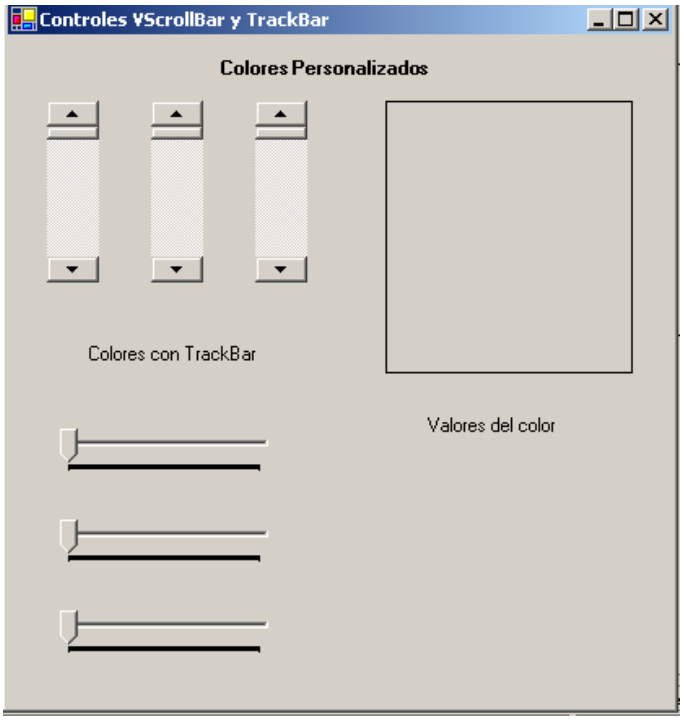
Para los controles **TrackBar** realice la misma programación, solamente es necesario cambiar el nombre del control es decir **barra?** por **desplazamiento?.**

En el anterior código al control **txtcolores** se le modifica la propiedad **Backcolor** asignándole el método **FromArgb** de la clase **Color** el cual define un color tomando como parámetros los valores para **rojo**, **verde** y **azul** y opcionalmente el nivel de transparencia. Al desplazarse sobre cada control se obtiene un valor el cual se asigna respectivamente a cada color. Después por cada barra de desplazamiento se le asigna a los objetos **Label** (color1, color2, color3) en su propiedad **Text** el nombre del color y el valor que toma el control en su propiedad **Value**. Esta operación se realiza también en el control **TrackBar**.

- **Ejecutar el proyecto**

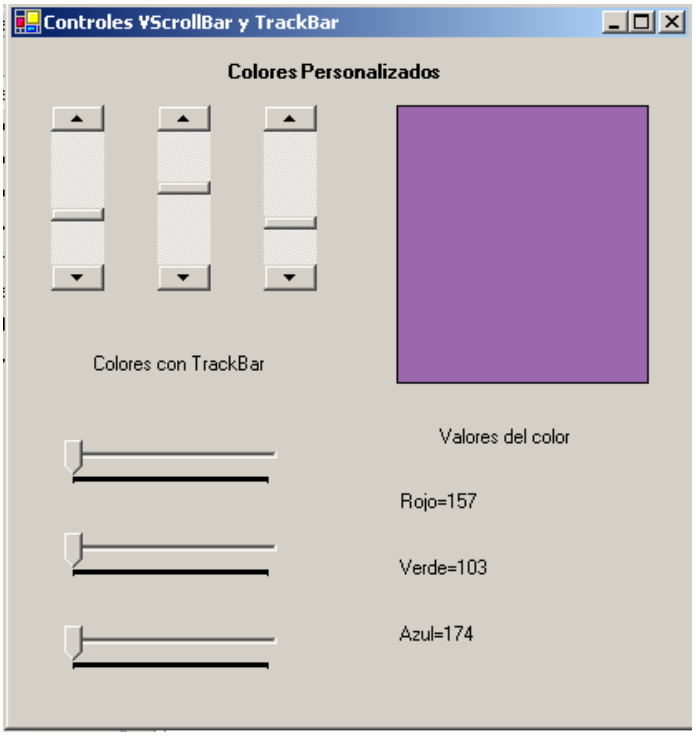
Al ejecutar el proyecto en el entorno de desarrollo de visual Basic.NET, se visualiza:

Figura 7.12 Ejecución aplicación Barras de Desplazamiento.



Al dar realizar el desplazamiento de la barra sobre cada uno de los controles **VScrollBar** el fondo del **Label** deberá ir cambiar de color. Al realizar esta operación se visualizará la siguiente figura:

Figura 7.13 Ejecución aplicación Barras de Desplazamiento.



Esto mismo se puede realizar cambiando de posición la barra de seguimiento (TrackBar) para obtener un color.

7.4 Controles CheckBox y RadioButton

Un control **CheckBox** es una casilla de verificación que permite obtener dos estados: verdadero si esta activada o falso si esta desactivada. Para obtener el valor de verdadero o falso se debe hacer a través de la propiedad **Checked**.

Un control **RadioButton** permite a un usuario escoger una alternativa entre varias alternativas. Al igual que el **CheckBox** por medio de la propiedad **Checked** se puede obtener sus dos estados: verdadero o falso.

La principal diferencia entre el **CheckBox** y el **RadioButton** es que si se tiene un conjunto de controles **CheckBox** y un conjunto de **RadioButton**, en los **CheckBox** se puede activar todos al mismo tiempo, mientras los **RadioButton** solo uno quedará activado en un momento dado.

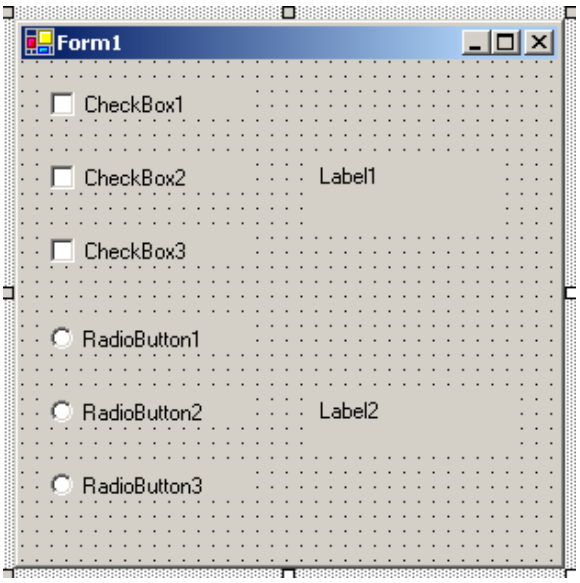
7.4.1 Ejemplo práctico controles CheckBox y RadioButton

Realizar una aplicación que permita a un usuario por medio de controles CheckBox y RadioButton cambiar el tipo de fuente, el color de fondo y el color del texto en un objeto Label.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control **CheckBox** y ubique 4 controles en el formulario en la posición deseada. También seleccione el control **RadioButton** y ubique 4 controles en el formulario, además, seleccione dos veces el control Label y ubíquelos en el formulario. La figura 7.14 muestra la interfaz de usuario

Figura 7.14 Interfaz de usuario (CheckBox, RadioButton).



- **Establecer las propiedades de los objetos de la interfaz de usuario**

Después de colocar los controles u objetos en el formulario, establezca las siguientes propiedades a los controles:

Tabla 7.4 Propiedades de los controles de la aplicación CheckBoxRadioButton.

Control	Propiedad	Valor
Label1	Name	txtcheckbox
	Text	Texto de prueba Checkbox
	TextAlign	MiddleCenter
	Font	Microsoft Sans Serif, 8,25pt, style=Italic
	Backcolor	White
Label2	Name	txtradiobutton
	Text	Texto de prueba RadioButtom
	TextAlign	MiddleCenter
	Font	Microsoft Sans Serif, 8,25pt, style=Italic
	Backcolor	White
CheckBox1	Name	chequeo1
	Text	Cambiar tipo de letra
CheckBox2	Name	chequeo2
	Text	Cambiar color de fondo

CheckBox3	Name	chequeo3
	Text	Cambiar color de texto
RadioButton1	Name	radio1
	Text	Cambiar tipo de letra
RadioButton2	Name	radio2
	Text	Cambiar Color de fondo
RadioButton3	Name	radio3
	Text	Cambiar color de texto
Form1	Name	formulario
	Text	Controles CheckBox y RadioButton

- **Escribir código**

Seleccione el objeto **Chequeo1**, dé doble clic para abrir el editor de código `Chequeo1_CheckedChanged` y escriba el siguiente código:

```
If (Chequeo1.Checked) Then
  txtcheckbox.Font=New Font ("Microsoft Sans Serif", 8.25!, FontStyle.Bold)
Else
  txtcheckbox.Font=New Font ("Microsoft Sans Serif",8.25!,FontStyle.Italic)
End If
```

La instrucción **Chequeo1.Checked** en la condición **if** se activará cuando la casilla de verificación del **CheckBox** este señalada. Por verdadero el tipo de fuente del **Label** cambiará al mismo tipo de fuente, el mismo tamaño y diferente estilo; en este caso a negrilla. Por falso retornará los valores predefinidos en el control, es este caso el tipo de fuente "Microsoft Sans Serif", el tamaño 8.25! y el estilo Itálico.

Seleccione el objeto **Chequeo2**, dé doble clic para abrir el editor del procedimiento Chequeo2_CheckedChanged y escriba el siguiente código:

```
If (Chequeo2.Checked) Then
  txtcheckbox.BackColor = System.Drawing.Color.Aquamarine
Else
  txtcheckbox.BackColor = System.Drawing.Color.White
End If
```

Por verdadero el **Label** cambiara de color de fondo utilizando el espacio de nombres System.Drawing.Color al cual se le asigna el color deseado. Por falso se retornara a los valores predefinidos en el control, es este caso el color de fondo blanco.

Seleccione el objeto **Chequeo3**, dé doble clic para abrir el editor del procedimiento Chequeo3_CheckedChanged y escriba el siguiente código:

```
If (Chequeo3.Checked) Then
  txtcheckbox.ForeColor = System.Drawing.Color.Red
Else
  txtcheckbox.ForeColor = System.Drawing.Color.Black
End If
```

Por verdadero el texto del **Label** cambiará de color utilizando también el espacio de nombres System.Drawing.Color al cual se le asigna el color deseado. Por falso se retornará los valores predefinidos en el control, es este caso el color del texto volverá a ser negro.

Seleccione el objeto **radio1**, dé doble clic para abrir el editor del procedimiento radio1_CheckedChanged y escriba el siguiente código:

```
If (radio1.Checked) Then
  txtradiobutton.Font= New Font("Microsoft Sans Serif", 8.25!,FontStyle.Bold)
Else
  txtradiobutton.Font=New Font("Microsoft Sans Serif",8.25!,FontStyle.Italic)
End If
```

Seleccione el objeto **radio2**, dé doble clic para abrir el editor del procedimiento radio2_CheckedChanged y escriba el siguiente código:

```
If (radio2.Checked) Then
  txtradiobutton.BackColor = System.Drawing.Color.Aquamarine
Else
  txtradiobutton.BackColor = System.Drawing.Color.White
End If
```

Seleccione el objeto **radio3**, dé doble clic para abrir el editor del procedimiento radio3_CheckedChanged y escriba el siguiente código:

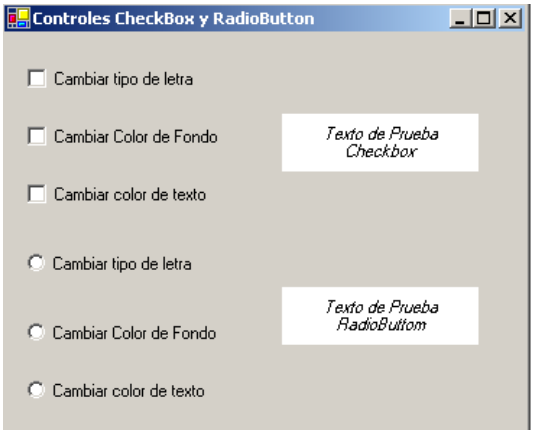
```
If (radio3.Checked) Then
   txtradiobutton.ForeColor = System.Drawing.Color.Red
Else
   txtradiobutton.ForeColor = System.Drawing.Color.Black
End If
```

Como se puede apreciar, la programación de los controles **RadioButton** es exactamente igual a la de los controles **CheckBox**.

- **Ejecutar el proyecto**

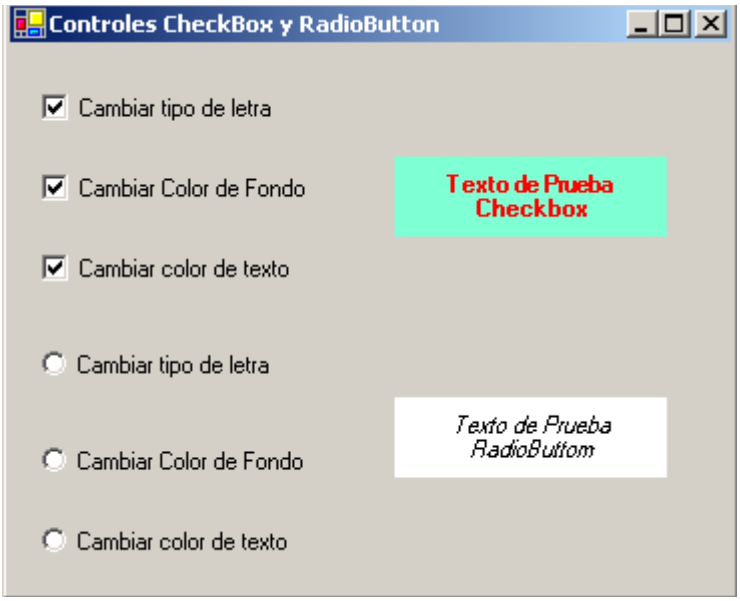
Al ejecutar el proyecto en el entorno de desarrollo de visual Basic.NET, se visualizará la siguiente figura:

Figura 7.15 Ejecución aplicación CheckBoxRadioButton.



Al dar clic en cada control **Checkbox** se cambiará el texto o el fondo del control **txtcheckbox**. Cada vez que se escoja un nuevo **CheckBox** si la opción escogida anteriormente sigue activada los cambios se mantendrán. Al escogerse todos los **Checkbox** se visualizará la siguiente figura:

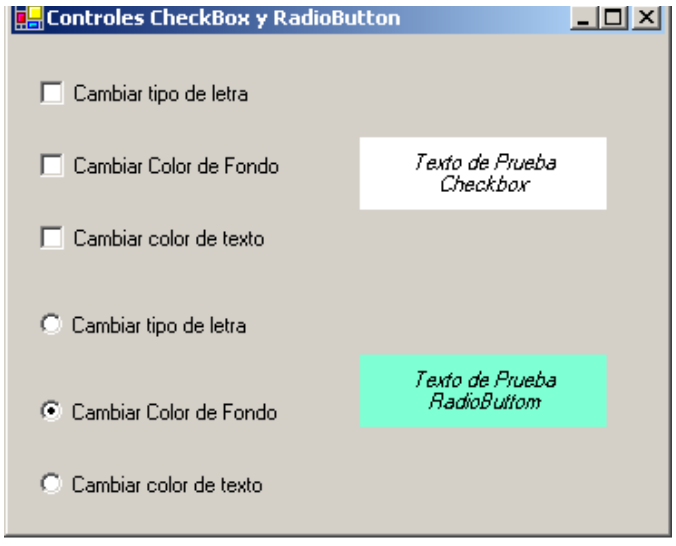
Figura 7.16 Ejecución de la aplicación con los controles CheckBox.



Como se puede apreciar, se realizaron todos los cambios que estaban especificados.

Al dar clic sobre un control **RadioButton** los demás controles **RadioButton** quedan desactivados; cuando existen varios controles **RadioButton** solo quedará activado aquel que el usuario haya especificado. Si se escoge la opción **Cambiar Color de Fondo**, los demás controles quedan desactivados como se puede apreciar, en la siguiente figura:

Figura 7.17 Ejecución aplicación con los controles RadioButton.



8. FORMULARIOS DE INTERFAZ SENCILLA (SDI) Y MÚLTIPLE (MDI)

Hasta ahora los ejemplos trabajados se han realizado en proyectos diferentes, es decir, por cada aplicación se ha creado un nuevo proyecto. Este tipo de aplicaciones son llamadas SDI (Interfaz de documento sencillo). Algunos programas que manejan este estilo de interfaz son: Wordpad, Bloc de notas, la calculadora de Microsoft Windows.
Una aplicación de Interfaz de documentos múltiples es aquella que permite visualizar varios documentos en una ventana, los ejemplos clásicos son el procesador de texto Word o la hoja de cálculo Excel de Microsoft. Las aplicaciones MDI (Interfaz de documentos múltiple) se reconocen por incluir menús con submenús para cambiar entre las distintas ventanas o documentos.

8.1 Creación de Menús con Interfaz de documento sencillo

Los menús y barras de herramientas proporcionan una forma estructurada y accesible para que los usuarios aprovechen los comandos y las herramientas contenidas en sus aplicaciones. La preparación y el diseño apropiados de menús y barras de herramientas garantizarán la funcionalidad en la aplicación.

A menudo, los usuarios examinan los menús antes de buscar información sobre la aplicación en cualquier otro lugar. Si los menús están bien diseñados los usuarios podrán comprender la aplicación y desarrollar un modelo mental basado únicamente en la organización de los menús y su contenido.

La creación de un sistema de menús implica varios pasos. Independientemente del tamaño de la aplicación y la complejidad de los menús que pretenda usar, deben:

- **Preparar y diseñar el sistema:** Decida los menús que necesita, dónde deben aparecer en la interfaz, cuáles requieren submenús, etc.
- **Crear los menús y submenús**: Defina los títulos, elementos de menú y submenús mediante el Diseñador de menús.
- **Asignar las tareas que desee al sistema:** Especifique las tareas que los menús deben realizar, como mostrar formularios y cuadros de diálogo.
- **Generar el programa de menú**: Ejecutar el programa para probar el sistema de menús.

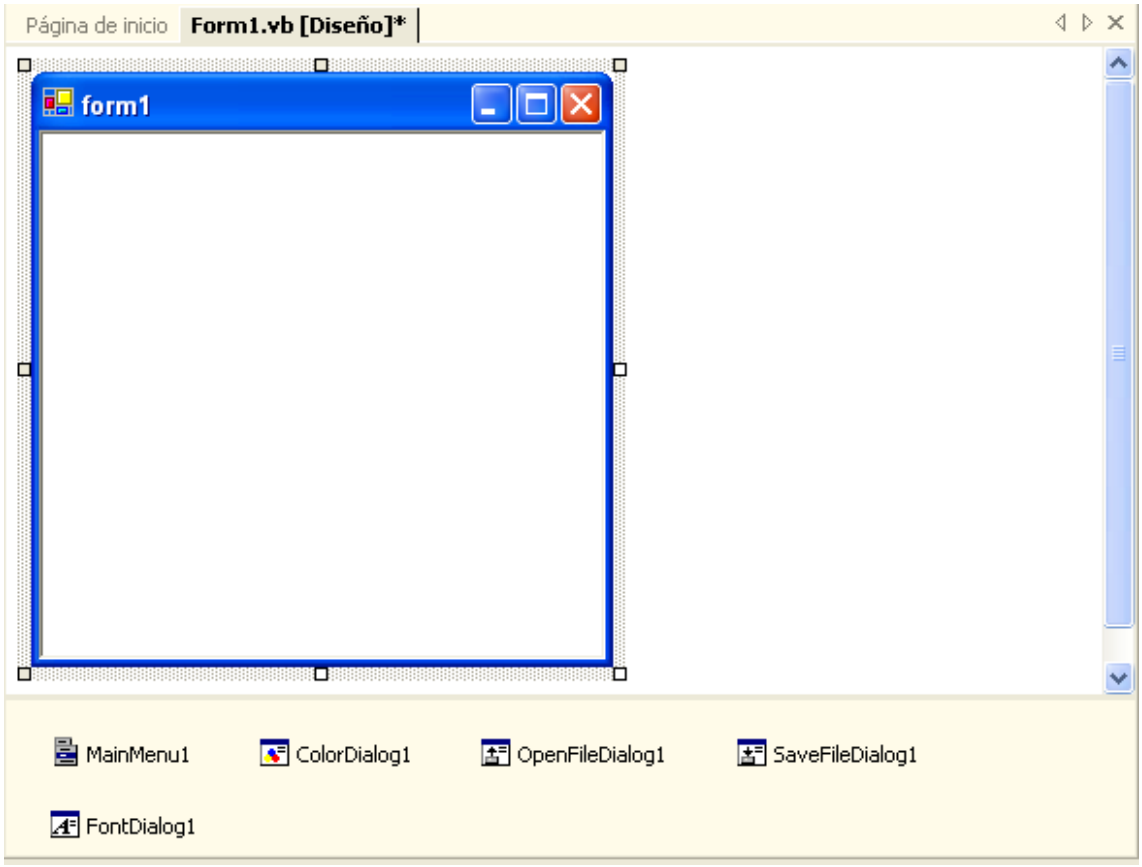
8.1.1 Ejemplo práctico menús con interfaz de documento sencillo

Diseñar una aplicación que contenga un formulario con menús y submenús, donde el usuario puede abrir y guardar un archivo de texto, salir de la aplicación, además, puede cambiar el color del texto y el tipo de fuente.

- **Crear la interfaz de usuario.**

Para crear la interfaz de documento sencillo se debe seleccionar el control **MainMenu**  del cuadro de herramientas y ubicarlo en el formulario, además deberá agregar un (1) **TextBox,** cambiándole el valor de la propiedad **Dock** a **Fill** y el valor de la propiedad **Multiline** a **True,** como también los controles **ColorDialog**, **FontDialog**, **OpenFileDialog** y **SaveFileDialog.** La figura que se debe visualizar es la siguiente:

Figura 8.1 Interfaz de usuario inicial de la aplicación MenuSencillo.



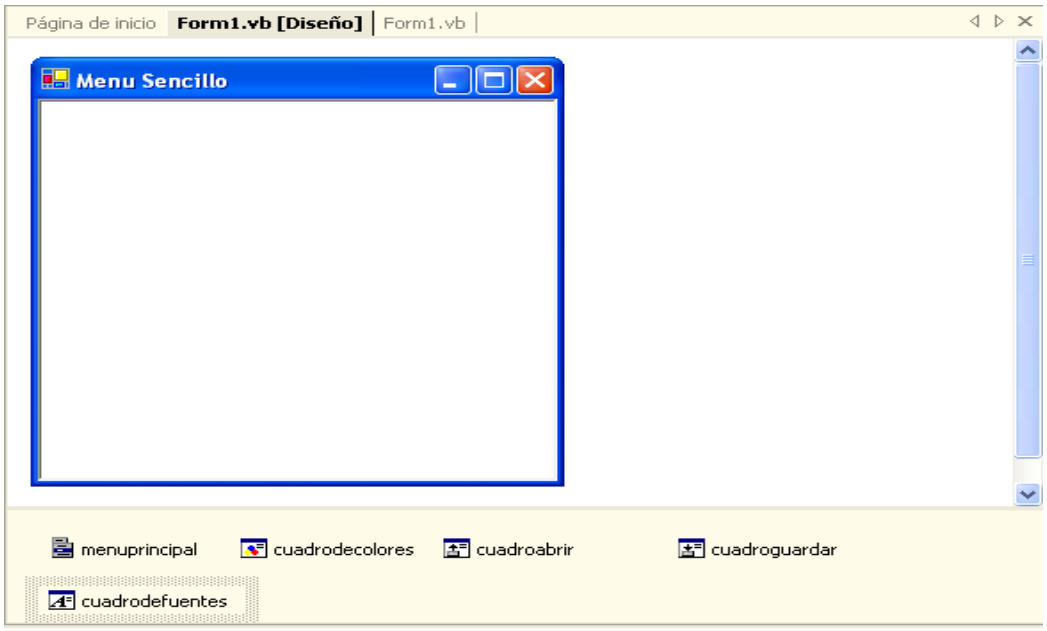
- **Establecer las propiedades de los objetos de la interfaz de usuario**

Después de colocar los controles u objetos en el formulario, establezca las siguientes propiedades a los controles:

Tabla 8.1 Propiedades de los controles de la aplicación MenuSencillo.

Control	Propiedad	Valor
Texbox1	Name	cuadro
	Text	En blanco
	Multiline	true
	dock	fill
Colordialog1	Name	cuadrodecolores
FontDialog1	Name	cuadrodefuentes
OpenFileDialog1	Name	cuadroabrir
SaveFileDialog1	Name	cuadroguardar
Mainmenu1	Name	menuprincipal
Form1	Name	formulario
	Menu	menuprincipal
	Text	Menu Sencillo

Figura 8.2 Interfaz de usuario con las propiedades modificadas.

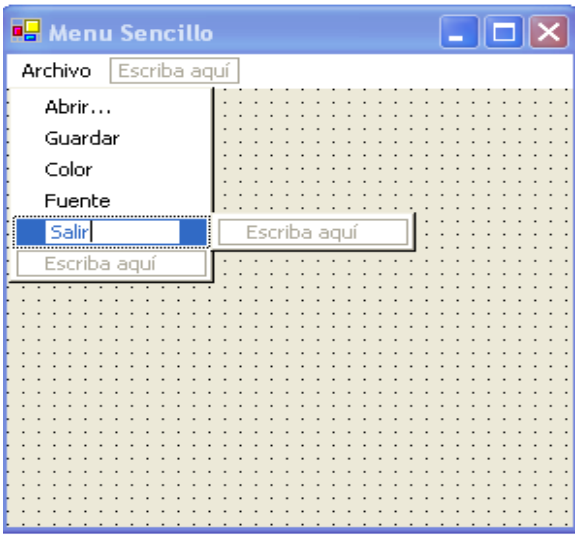


Pulse sobre el control **menuprincipal** para que aparezca sobre el formulario el texto **Escriba aquí**; allí escriba el nombre del menú **Archivo** y a continuación añada los siguientes elementos:

- Abrir
- Guardar
- Color
- Fuente
- Salir

La figura seria:

Figura 8.3 Menu de la aplicación MenuSencillo.



También es posible definir teclas de acceso o teclas de método abreviado a los diferentes menús utilizando el teclado. Por ejemplo si se quisiera acceder más rápidamente al menú **Archivo** solamente debe pulsar la tecla **Alt** y a continuación la letra **A**. una vez abierto la opción y si dentro de esta hubiese un submenú llamado **Guardar** con solo pulsar la tecla **G** se puede ejecutar el comando. Para asociar un método abreviado a las opciones del menú se debe anteponer a la letra que se quiere utilizar como tecla de acceso abreviado el carácter **ampersand** (&). Entonces siguiendo con el ejemplo anterior añada las teclas de acceso a las siguientes opciones: Archivo (letra A), Abrir (letra b), Guardar (letra G), Color (letra C), Fuente (letra F), Salir (letra S). La figura quedaría de la siguiente manera:

Figura 8.4 Menu con teclas de acceso abreviado.

Después de realizada esta tarea se debe seleccionar la opción a la cual se le va insertar el respectivo código y dár doble clic para visualizar el procedimiento. En el ejemplo, seleccione la opción **Color** y haga doble clic para visualizar el siguiente procedimiento:

```
Private Sub MenuItem4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

    Me.cuadrodecolores.ShowDialog()
     txtcuadro.ForeColor = Me.cuadrodecolores.Color

End Sub
```

En la primera línea se utiliza el método **ShowDialog**() que permite abrir el control **cuadrodecolores** en tiempo de ejecución. En la segunda línea se utiliza el método **Color** que permite obtener el color seleccionado por el usuario mediante el cuadro de diálogo **cuadrodecolores** para aplicarlo al texto mediante la propiedad **Forecolor** del control **txtcuadro**.

Ahora seleccione la opción **Fuente** y haga doble clic para visualizar el procedimiento y escriba el siguiente código:

```
Me.cuadrodefuentes.ShowDialog()
txtcuadro.Font = Me.cuadrodefuentes.Font
```

En la primera línea se utiliza el método **ShowDialog**() que permite abrir el control **cuadrodefuentes** en tiempo de ejecución. En la segunda línea se utiliza el

método **Font** que permite obtener la fuente seleccionada por el usuario mediante el cuadro de diálogo **cuadrodefuentes** para aplicarlo a la fuente del texto mediante la propiedad **Font** del control **txtcuadro**.

Después seleccione la opción **Abrir…** y haga doble clic para visualizar el procedimiento y escriba el siguiente código:

```
txtcuadro.text=" "
Me.cuadroabrir.Title = "Seleccione el archivo a abrir..."
Me.cuadroabrir.InitialDirectory = "C:\"
Me.cuadroabrir.Filter = "Archivo de Texto(*.txt)|*.txt"
Me.cuadroabrir.ShowDialog()
If Me.cuadroabrir.FileName.Length > 1 Then
    Dim verarchivo As New IO.StreamReader(Me.cuadroabrir.FileName)
    Me.txtcuadro.Text = verarchivo.ReadToEnd
Else
   MsgBox("Archivo no contiene información")
End If
```

En el anterior código lo primero que se hace es utilizar el método **Title** del control **cuadroabrir** el cual permite colocarle un titulo al cuadro de diálogo, luego por medio del método **InitialDirectory** se seleccioná la carpeta inicial que se visualizará en el cuadro de diálogo. El método **Filter** permite visualizar solamente los archivo que contenga la extensión .txt; luego se utiliza el método **ShowDialog**() que abre el control **cuadroabrir** en tiempo de ejecución. Por medio de la propiedad **Length** se determina si el archivo contiene información o no. Si contiene información, se crea una variable llamada **verarchivo** de tipo **IO.StreamReader** (clase que se utiliza para leer líneas de información desde un archivo de texto estándar) que guarda el nombre del archivo seleccionado y es asignado a la propiedad **Text** utilizando el método **ReadtoEnd** que lee el archivo hasta el final. Si por el contrario no contiene información se mostrará un mensaje informando que el archivo esta vació.

Ahora seleccione la opción **Guardar** y haga doble clic para visualizar el procedimiento y escriba el siguiente código:

```
Me.cuadroguardar.Title = "Cuadro de Diálogo para guardar un archivo"
Me.cuadroguardar.InitialDirectory = "C:\"
Me.cuadroguardar.Filter = "Archivo de Texto(*.txt)|*.txt"
Me.cuadroguardar.ValidateNames = True
Me.cuadroguardar.ShowDialog()
Dim guardarárchivo As IO.StreamWriter
guardarárchivo = New IO.StreamWriter(Me.cuadroguardar.FileName)
guardarárchivo.Write(Me.txtcuadro.Text)
guardarárchivo.Close()
```

En el anterior código se utiliza el método **Title** del control **cuadroguardar** el cual permite colocarle un titulo al cuadro de diálogo, luego por medio del método **InitialDirectory** se selecciona la carpeta inicial que se visualizará en el cuadro de diálogo. El método **Filter** permite visualizar solamente los archivo que contenga la extensión .txt; luego se utiliza el método **ValidateName** con valor **True** para validar que el nombre del archivo no contenga caracteres especiales, además se utiliza el método **ShowDialog**() para abrir el cuadro de diálogo del control **cuadroguardar** en tiempo de ejecución. Se crea una variable **guardarárchivo** de tipo **StreamWriter**

(clase que se utiliza para escribir información a un archivo de texto estándar) que guarda el nombre del archivo de texto y utilizando el método **Write** se guarda el contenido de **txtcuadro**. Por último se cierra el archivo por intermedio del método **close** ().

Ahora seleccione la opción **Salir** y haga doble clic para visualizar el procedimiento y escriba el siguiente código:

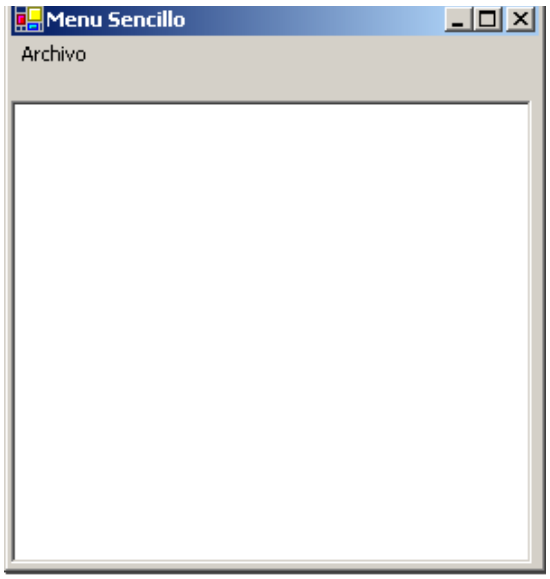
```
End
```

La instrucción **end** permitirá salir de la aplicación.

- **Ejecutar el formulario**

Cuando se ejecuta el proyecto en el entorno de desarrollo de visual Basic.NET, se presenta la siguiente figura:

Figura 8.5 Ejecución aplicación MenuSencillo.



Ahora puede escribir un texto y realizar los cambios que considere desde la opción del menú **Archivo**: cambiar de color el texto, cambiar el tipo de fuente, guardar el archivo y abrir el archivo.

8.2 Creación de Menús con interfaz múltiple

Una aplicación de interfaz múltiple se compone de un formulario principal denominado formulario MDI, que actuará como contenedor de otros formularios abiertos en el transcurso del programa denominado formularios hijos o secundarios MDI.

8.2.1 Ejemplo práctico formularios MDI

Diseñar una aplicación que contenga un formulario MDI con menús y submenús, donde se permita abrir tres formularios en ventanas independientes.

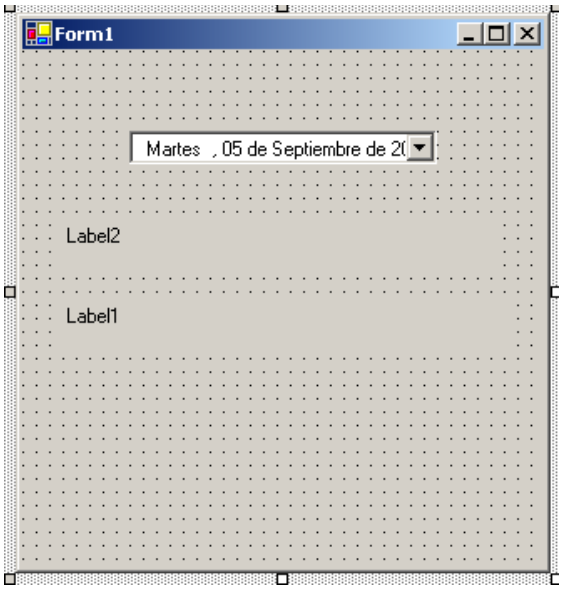
- **Crear la interfaz de usuario**

Para crear la interfaz de usuario se utilizarán cuatro formularios: el primero será el formulario principal y los demás formularios realizarán una labor específica cada uno.

- **Creación del primer Formulario**

En el primer formulario seleccione los siguientes controles: dos Label y un DateTimePicker (representa un control de Windows que permite al usuario seleccionar una fecha y una hora, y mostrarlas con un formato especificado) y sitúelos en el formulario de acuerdo a la siguiente figura:

Figura 8.6 Interfaz de usuario del primer formulario.



- **Establecer las propiedades de los objetos de la interfaz de usuario**

Tabla 8.2 Propiedades de los controles del primer formulario FormularioMDI.

Control	Propiedad	Valor
Label1	Name	lblfecha
	Text	En blanco
	TextAlign	MiddleCenter
	Font	Microsoft Sans Serif, 8,25pt, style=Italia
Label2	Name	lbltitulo
	Text	Esta es la nueva fecha escogida
	TextAlign	MiddleCenter
	Font	Microsoft Sans Serif, 8,25pt, style=Bold
DateTimePicket	Name	controlfecha
Form1	Name	formulariofecha
	Text	Control DateTimePicker

- **Escribir código**

Seleccione el objeto **controlfecha**, dé doble clic para abrir el editor del procedimiento `contrlfecha_ValueChanged` y escriba el siguiente código:

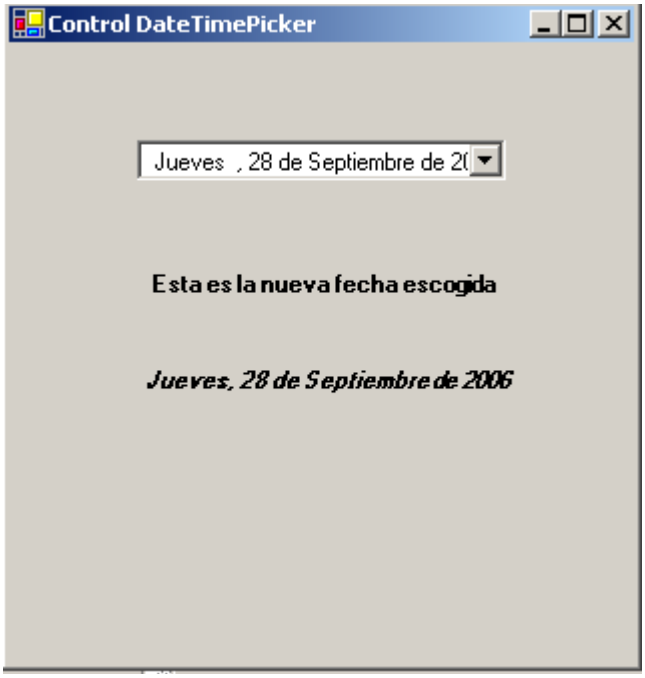
```
txtfecha.Text = controlfecha.Text
```

El código anterior permite visualizar en la propiedad **Text** del control **txtfecha** la fecha escogida por el usuario al dar clic en el control **controlfecha**.

- **Ejecutar el formulario**

Al ejecutarse el formulario se visualizará la siguiente pantalla:

Figura 8.7 Ejecución primer formulario.



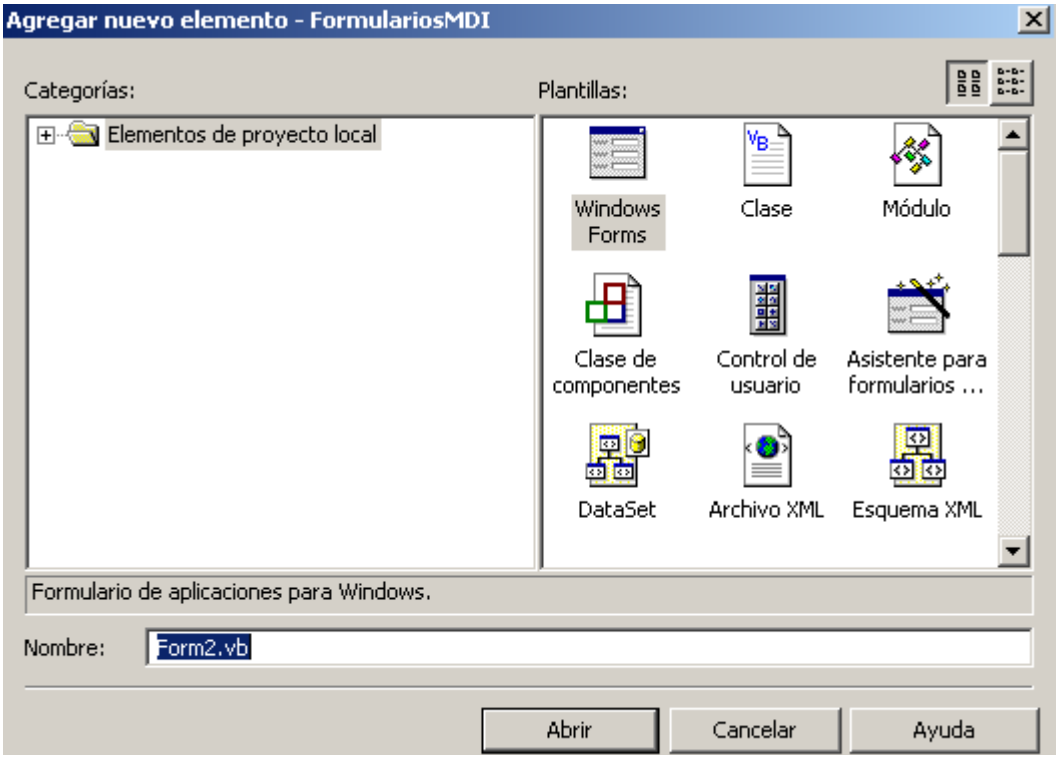
Seleccione del control **controlfecha** la fecha **Jueves, 28 de Septiembre de 2006** para visualizar la figura anterior.

Creación del segundo Formulario

Para crear el segundo formulario y que quede dentro de la aplicación **FormularioMDI,** se deben realizar los siguientes pasos:

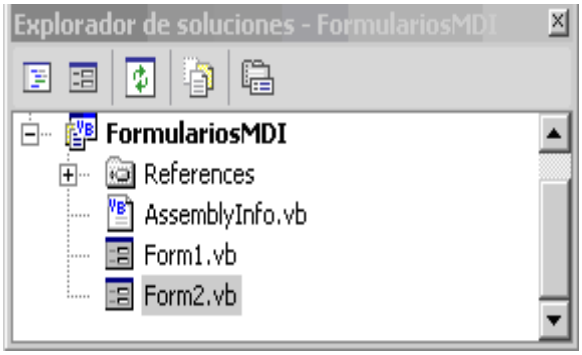
1. En el menú **Proyecto**, seleccione **Agregar Windows Forms**
2. En el cuadro **Nombre**, cambie el nombre del formulario por **formulariofiguras** y a continuación haga clic en **Abrir** para visualizar el segundo formulario

Figura 8.8 Agregar un nuevo formulario a la aplicación.



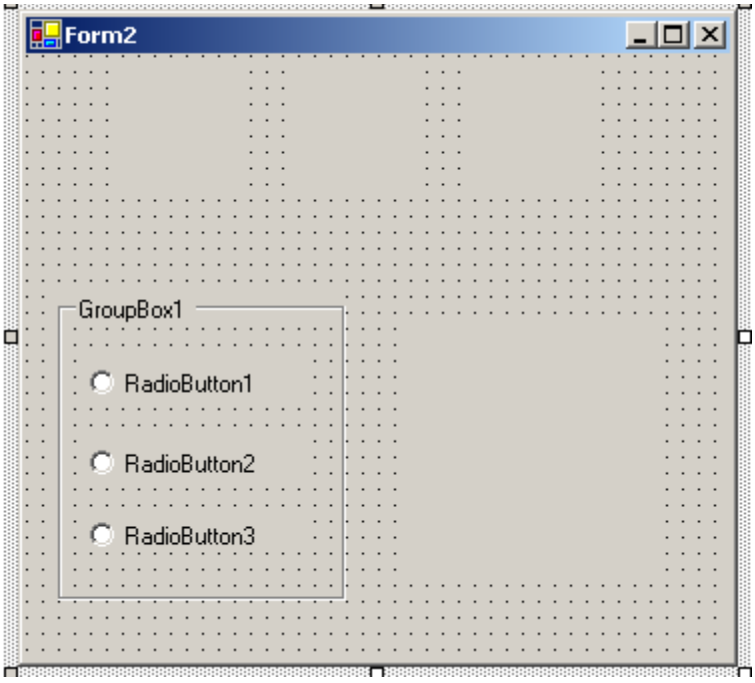
En este momento el explorador de soluciones tendría el siguiente aspecto:

Figura 8.9 Explorador de soluciones con dos formularios.



En este formulario seleccione los siguientes controles: 4 **PictureBox** (control que permite desplegar archivos gráficos de tipo, gif, bitmap, icon, jpeg.), 1 **GroupBox** (control de Windows que muestra un marco alrededor de un grupo de controles con un título opcional.) y 3 **RadioButton** (control que permite a un usuario escoger una alternativa entre varias opciones), la interfaz quedara como se muestra en la figura:

Figura 8.10 Interfaz de usuario segundo formulario.



- **Establecer las propiedades de los objetos de la interfaz de usuario**

Después de colocar los controles u objetos en el formulario, establezca las siguientes propiedades a los controles:

Tabla 8.3 Propiedades de los controles del segundo formulario.

Control	Propiedad	Valor
Picturebox1, PictureBox2, PictureBox3.	Name	imagen1….imagen3
	SizeMode	StretchImage
	Visible	False
	Image	La imagen que seleccione el usuario
Picture4	Name	imagenprincipal
	SizeMode	StretchImage
GroupBox1	Name	cuadro
	Text	Figuras
RadioButton1	Name	rbportatil
	Text	Portatil
RadioButton2	Name	rbsatelite
	Text	Satélite
RadioButton3	Name	rbfotocopiadora
	Text	Fotocopiadora
Form1	Name	formulariofiguras
	Text	Formulario figuras

- **Escribir código**

Seleccione el objeto **rbportatil**, dé doble clic para abrir el editor del procedimiento `rbportatil_CheckedChanged` y escriba el siguiente código:

```
If (rbportaltil.Checked) Then
    imagenprincipal.Image = imagen1.Image
End If
```

Seleccione el objeto **rbsatelite**, dé doble clic para abrir el editor del procedimiento rbsatelite`_CheckedChanged` y escriba el siguiente código:

```
If (rbsatelite.Checked) Then
    imagenprincipal.Image = imagen2.Image
End If
```

Seleccione el objeto **`rbfotocopiadora`**, dé doble clic para abrir el editor del procedimiento `rbfotocopiadora_CheckedChanged` y escriba el siguiente código:

```
If (rbfotocopiadora.Checked) Then
    imagenprincipal.Image = imagen3.Image
End If
```

En los códigos anteriores al seleccionar cualquiera de los controles **RadioButton** se asignará a la propiedad **Image** del control **imagenprincipal** la imagen que contenga el **PictureBox** que ha sido seleccionado.

- **Ejecutar el formulario**

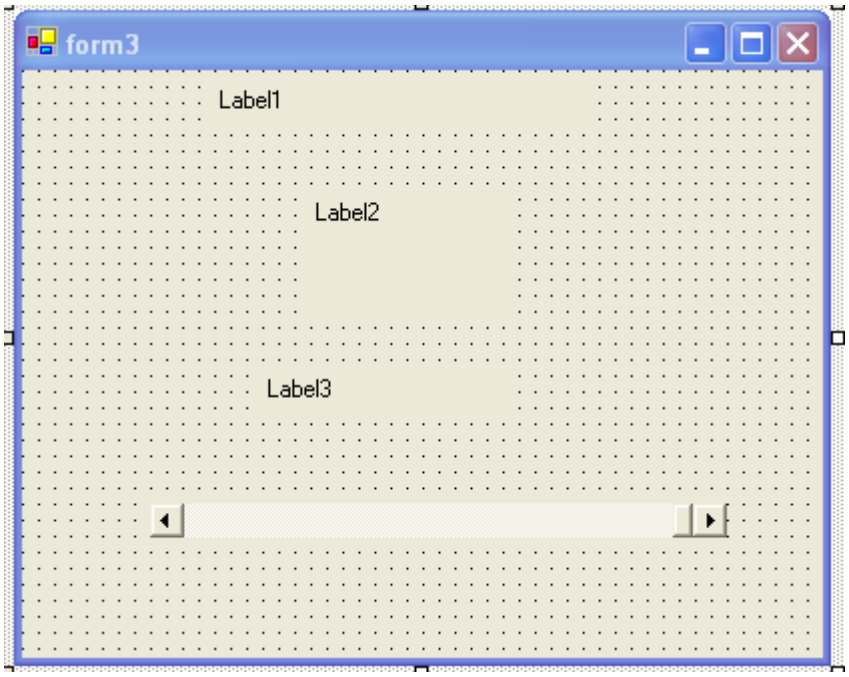
Al ejecutarse el proyecto se visualizará la siguiente figura:

Figura 8.11 Ejecución segundo formulario.

Creación del tercer Formulario

Para crear el tercer formulario y que quede dentro de la aplicación **FormularioMDI** se deben realizar los mismos pasos que se realizaron para agregar el formulario dos:

Figura 8.12 Interfaz de Usuario tercer Formulario.



- **Establecer las propiedades de los objetos de la interfaz de usuario**

Después de colocar los controles u objetos en el formulario, establezca las siguientes propiedades a los controles:

Tabla 8.4 Propiedades de los controles del tercer formulario.

Control	Propiedad	Valor
	Name	barra
HScrollBar1	Maximum	255
	Minimum	1
Label1	Name	lbltitulo
	Text	Código ASCII
	TextAlign	MiddleCenter
	Font	Microsoft Sans Serif, 8,25pt, style=Italia
Label2	Name	lblsimbolo
	Text	En blanco
	TextAlign	MiddleCenter
	Font	Microsoft Sans Serif, 26pt, style=Negrita
Label3	Name	lblnumero
	Text	En blanco
	TextAlign	MiddleCenter
	Font	Microsoft Sans Serif, 8.25pt, style=Negrita
Form1	Name	formularioascii
	Text	Código ASCII

- **Escribir código**

Seleccione el objeto **barra**, de doble clic para abrir el editor del procedimiento `barra_Scroll` y escriba el siguiente código:

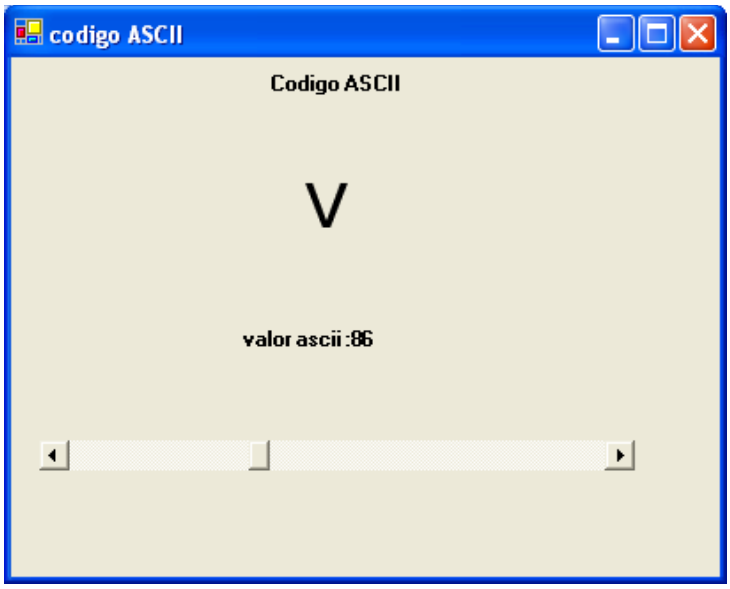
```
Private Sub Barra_Scroll(ByVal sender As System.Object, ByVal e As
                         System.Windows.Forms.ScrollEventArgs) Handles barra.Scroll

    lblsimbolo.Text = Chr(barra.Value)
    lblnumero.Text = "valor ascii :" & barra.Value

End Sub
```

En el código anterior al seleccionar la barra de desplazamiento del control **HScrollBar** se le asignará al objeto **lblsimbolo** por intermedio del método **Text** la representación grafica del código ASCII obtenido por el método **Value** de la barra de desplazamiento y convertido a texto por la función **CHR**. También al objeto **lblnumero** por intermedio del método **Text** se le asigna el texto "**valor ascii:"** más el valor de la barra de desplazamiento con el método **Value**.

- **Ejecutar el formulario**

Al ejecutar el proyecto se visualizará la siguiente figura:

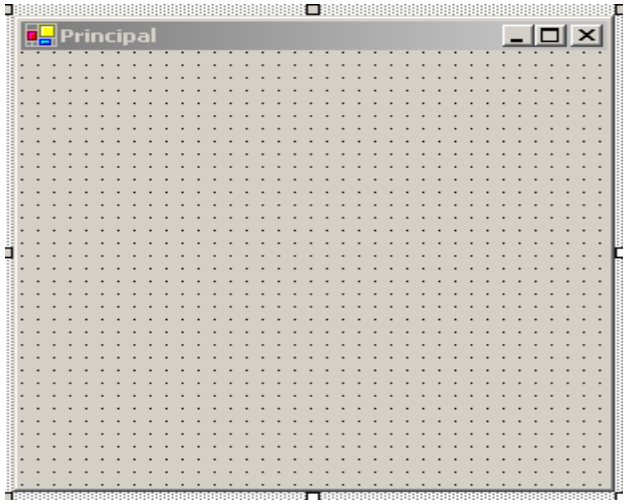
Figura 8.13 Ejecución tercer formulario.



Creación Formulario Principal (MDI)

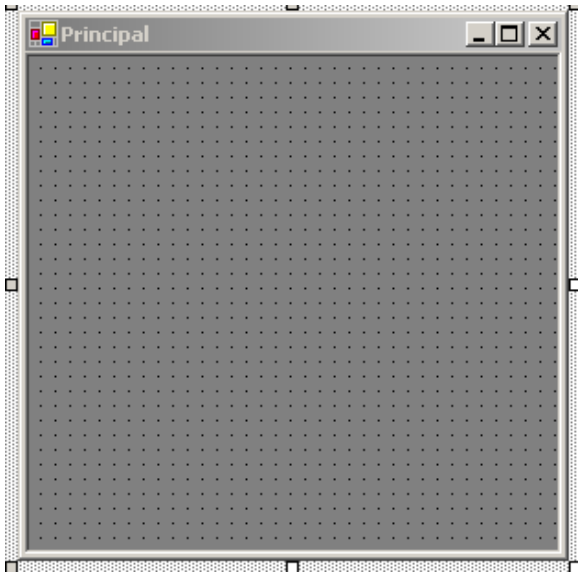
Para crear el formulario principal de la aplicación **FormularioMDI,** el cual contendrá el menú para ejecutar a los demás formularios como también otras opciones que se irán explicando**,** se deben hacer los mismos pasos que se realizaron para agregar el formulario dos y tres:

Figura 8.14 Formulario principal de FormularioMDI.



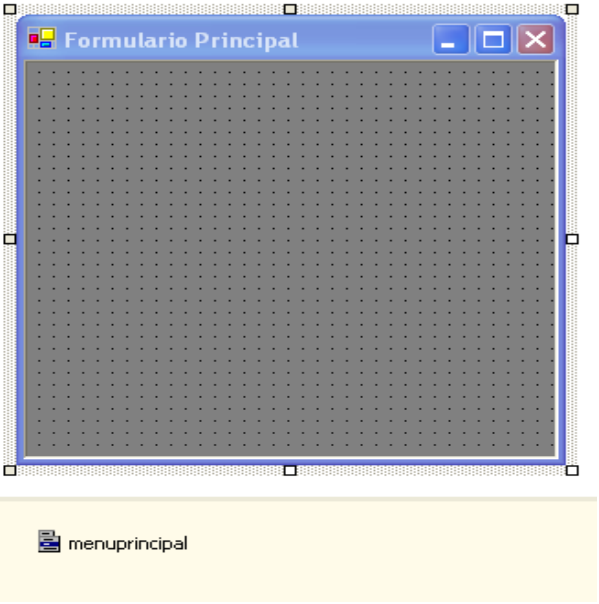
En éste formulario seleccione la propiedad **IsMdicontainer** y coloque el valor de **True** para que el formulario tenga un comportamiento de un contenedor **MDI**. Al cambiar el valor de la propiedad el color de fondo será de un gris más oscuro, como muestra la siguiente figura:

Figura 8.15 Formulario principal con comportamiento como contenedor.



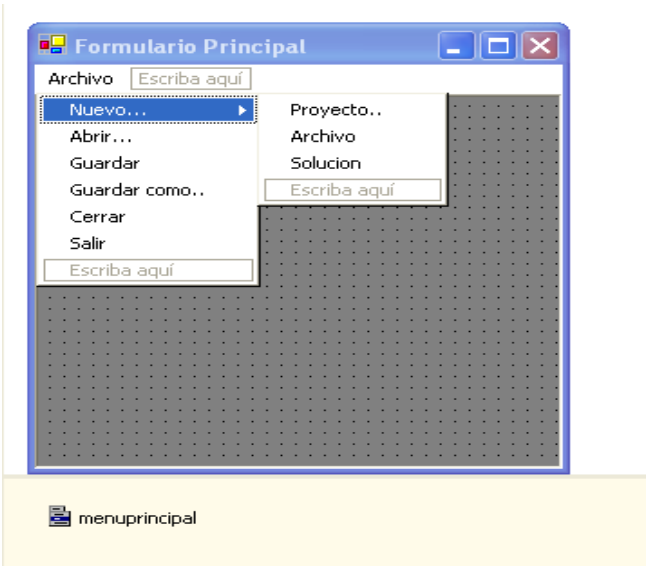
El siguiente paso para crear el menú es seleccionar el control **MainMenu**  del cuadro de herramientas (**MenuStrip** en Visual Basic 2005) y ubicarlo en el formulario. Cambie el valor de la propiedad **Name** del formulario por **principal** y en la misma propiedad del objeto **mainmenu1** por **menuprincipal**. Ahora cambie el valor de la propiedad **menú** del formulario por el objeto **menuprincipal** y el valor de la propiedad **Text** por **Formulario Principal.** También cambie el valor de la propiedad **Windows** del formulario por **Maximized** la cual al ejecutar el formulario principal se visualizará en pantalla completa.

Figura 8.16 formulario principal con el control MainMenu.



Ahora pulse el objeto **menuprincipal** para visualizar el texto **Escriba aquí.** Allí escriba el nombre del menú **Archivo** y vaya añadiendo los elementos que muestra la siguiente figura:

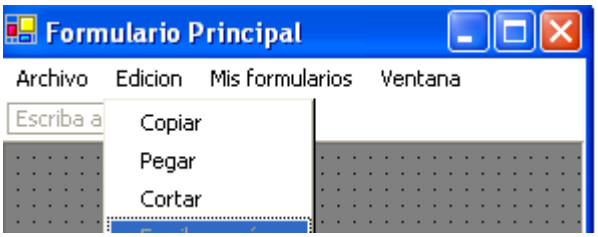
Figura 8.17 Formulario principal con el diseño del menú.



Modifique las opciones principales del menú de la siguiente forma:

- En la opción del menú **Edición** agréguele los siguientes submenús:
  - Copiar
  - Pegar
  - Cortar

Figura 8.18 Formulario principal y la opción Edición.

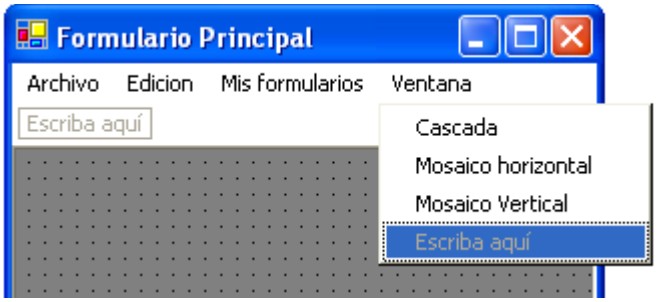


- En la opción del menú **Mis formularios** agréguele los siguientes submenús:
  - Fecha
  - Imágenes
  - Colores

Figura 8.19 Formulario principal y la opción Mis formularios.

- En la opción del menú **Ventana** agréguele los siguientes submenús:
  - Cascada
  - Mosaico horizontal
  - Mosaico Vertical
  - Organizar Iconos

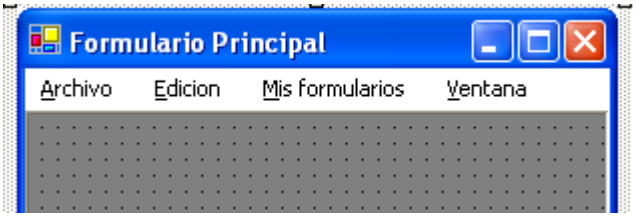
Figura 8.20 Formulario principal y la opción Ventana.



También es posible definir teclas de acceso o teclas de método abreviado a los diferentes menús utilizando el teclado. Por ejemplo, si se quisiera acceder más rápidamente al menú **Edición** solamente debe pulsar la tecla **Alt** y a continuación la letra **E**. Una vez abierto la opción y si dentro de esta hubiese un submenú llamado **Deshacer** con solo pulsar la tecla **D** se puede ejecutar el comando. Para asociar un

método abreviado a las opciones del menú se debe anteponer a la letra que se quiere utilizar como tecla de acceso abreviado el carácter ampersand (**&**). Entonces con el ejemplo anterior añada la tecla de acceso a las siguientes opciones: **Archivo**, **Edición**, **Mis formularios**, **Ventana**. La figura quedaría de la siguiente manera:

Figura 8.21 Formulario principal y el menú con teclas de acceso abreviado.



Bien, ahora se debe realizar la programación para poder abrir los formularios desde las diferentes opciones del menú. Lo primero que se debe realizar es seleccionar la opción deseada y dar doble clic para visualizar el procedimiento. En el ejemplo, seleccione la opción **Mis formularios** y allí seleccione la opción **Fecha**.

Figura 8.22 Submenú Fecha del formulario principal.

Haga doble clic sobre la opción **Fecha** para visualizar el procedimiento MenuItem20_Click (el numero del menuItem puede cambiar de acuerdo a las opciones que haya creado en el menú principal). Digite el siguiente código:

```
Dim formulario1 As New formulariofecha
formulario1.MdiParent = Me
formulario1.Show()
```

En la primera línea se crea un objeto llamado **formulario1** de tipo **formulariofecha**, en la segunda línea se asigna a la propiedad **MdiParent** la instancia actual del formulario en ejecución, es este caso **Me** contiene el formulario actual y por último, por medio del método **show** se muestra el formulario.
Haga doble clic sobre la opción **Imágenes** para visualizar el procedimiento. En este procedimiento digite el siguiente código:

```
Dim formulario2 As New formulariofiguras
formulario2.MdiParent = Me
formulario2.Show()
```

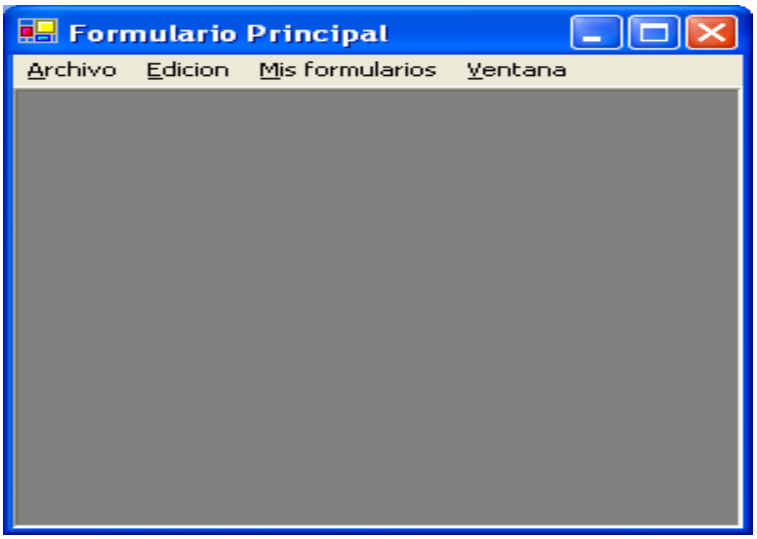
Haga doble clic sobre la opción **Código Ascii** para visualizar el procedimiento. En este procedimiento digite el siguiente código:

```
Dim formulario3 As New formularioascii
formulario3.MdiParent = Me
formulario3.Show()
```

- **Ejecutar el formulario**

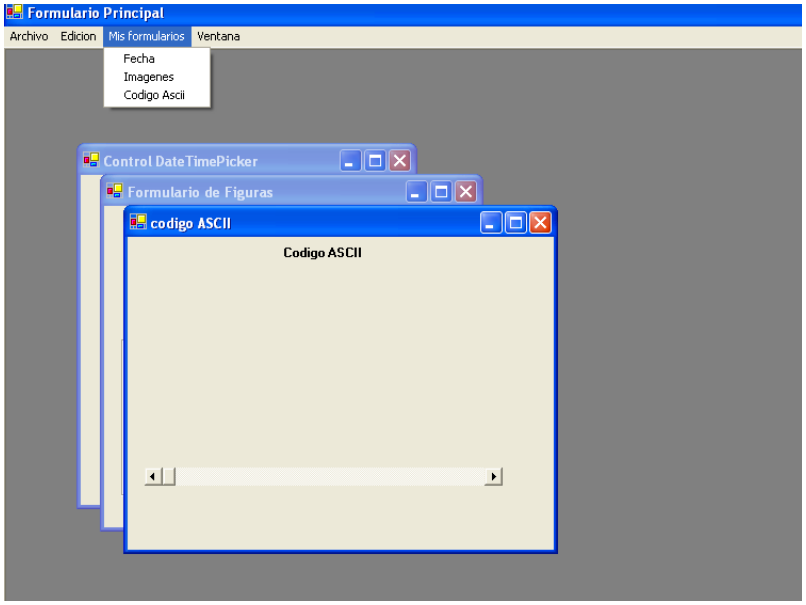
Al ejecutar el proyecto en el entorno de desarrollo de Visual Basic.NET, se visualizará:

Figura 8.23 Ejecución aplicación FormulariosMDI.



Al seleccionar cada una de las opciones del menú **Mis formularios**, se visualizarán cada uno de los formularios que se han diseñado. Podrá hacer clic en cualquier opción las veces que se quiera y se irá visualizándo el formulario escogido *n* veces.

Figura 8.24 Ejecución de formularios en el MDI.



8.3 Desactivar las opciones del menú en formularios MDI

En la aplicación que se ha trabajado se puede abrir tantas copias de los formularios como se necesiten. Para conseguir que de un determinado formulario se pueda abrir una sola instancia, en primer lugar se debe deshabilitar la opción del menú que abre dicho formulario, asignándole **False** a la propiedad **Enabled.** Entonces seleccione la opción **Fecha** haga doble clic y adicione el siguiente código:

```
Me.MenuItem20.Enabled = False
```

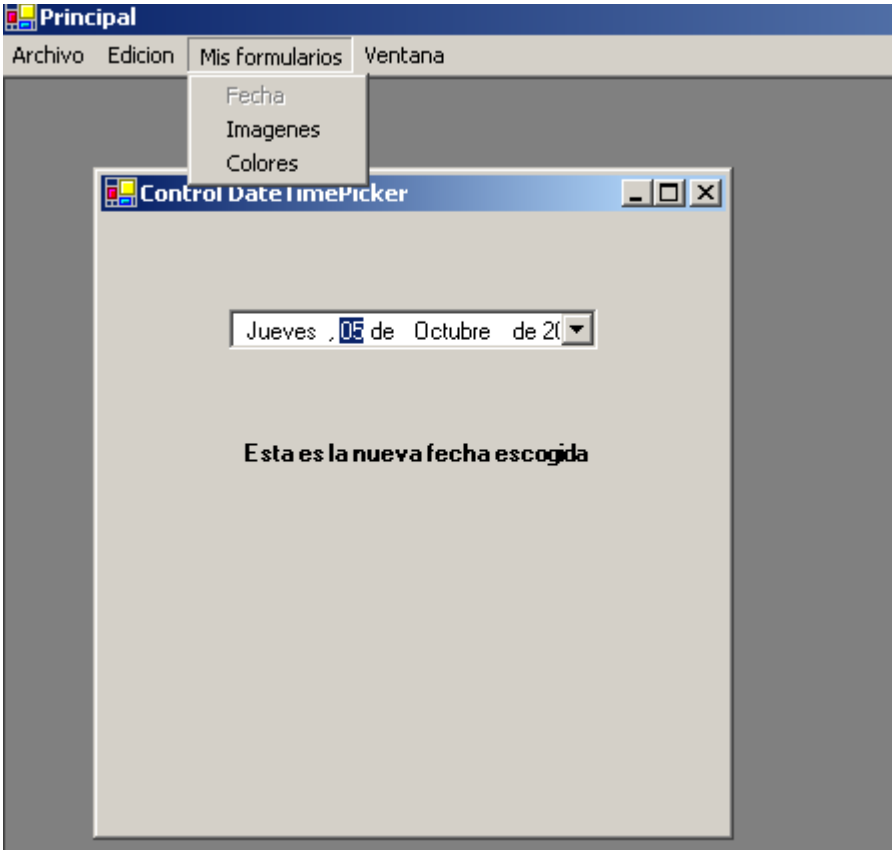
En segundo lugar, cada vez que se cierra un formulario se activa el evento **Closed**, por consiguiente se debe modificar dicho evento con el siguiente código:

```
CType(MdiParent, Principal).MenuItem20.Enabled = True
```

En el código anterior se utilizó la función **CType** que devuelve el resultado de convertir explícitamente una expresión a un tipo de datos, objeto, estructura, clase o interfaz. La expresión es la instancia actual del formulario en ejecución y el formulario es el tipo de dato, para así poder acceder a la opción **Fecha** del formulario **Principal** y volverla a habilitar.

Al ejecutar nuevamente el formulario **Principal** y seleccionar la opción **Fecha**, se podrá apreciar que esta queda deshabilitada. Se volverá a Habilitar al cerrar el formulario.

Figura 8.25 Inhabilitación de opciones del menú.



8.4 Manipulación del menú Ventana.

En el ejemplo que se ha venido trabajando, se creó una opción de menú llamada **Ventana**. Para dar un aspecto más profesional se añadió unas opciones que permiten la organización de los formularios en: cascada, mosaico horizontal y mosaico vertical. En este caso para organizar los archivos abiertos se debe ejecutar el método **LayoutMdi** () pasándole como argumento la opción deseada. Además se puede permitir visualizar los nombres de los formularios abiertos permitiendo cambiar de formulario activo al seleccionar una de esas opciones. Para esto en la opción **Ventana** se debe asignar a la propiedad **MdiList** el valor de **True**.

Ahora seleccione la opción Cascada, de doble clic y escriba el siguiente código:

```
Me.LayoutMdi (MdiLayout.Cascade)
```

Para la opción mosaico horizontal:

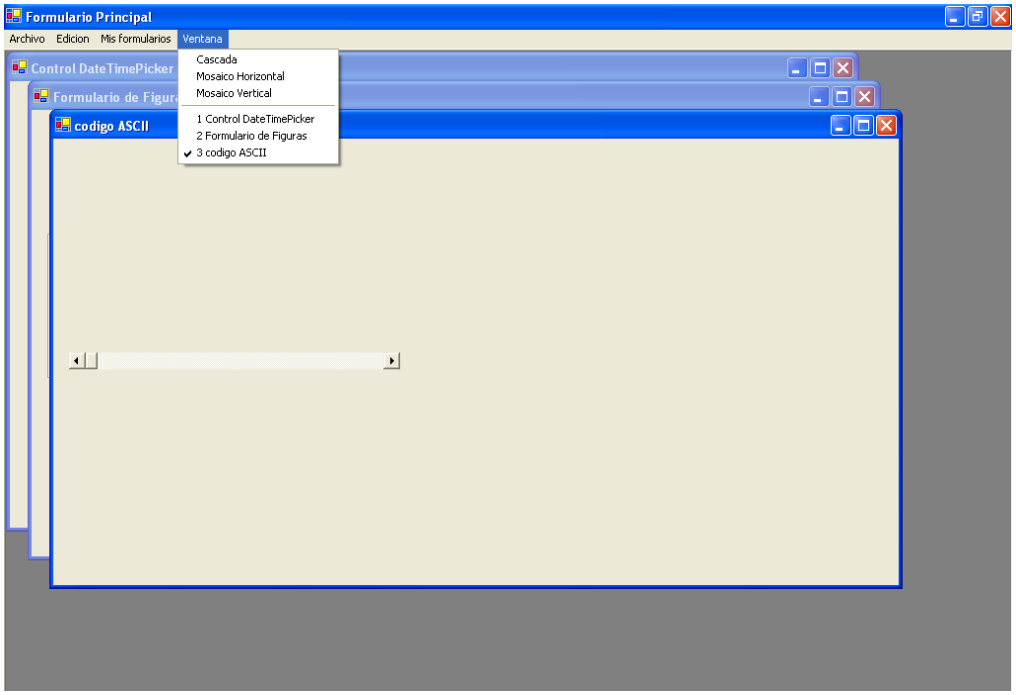
```
Me.LayoutMdi (MdiLayout.TileHorizontal)
```

Y para la opción mosaico vertical:

```
Me.LayoutMdi (MdiLayout.TileVertical)
```

Al ejecutar nuevamente la aplicación y seleccionando cada una de las opciones del menú **Mis formularios** y luego seleccionar la opción **Cascada** se visualizará la siguiente figura:

Figura 8.26 Formularios en Cascada.

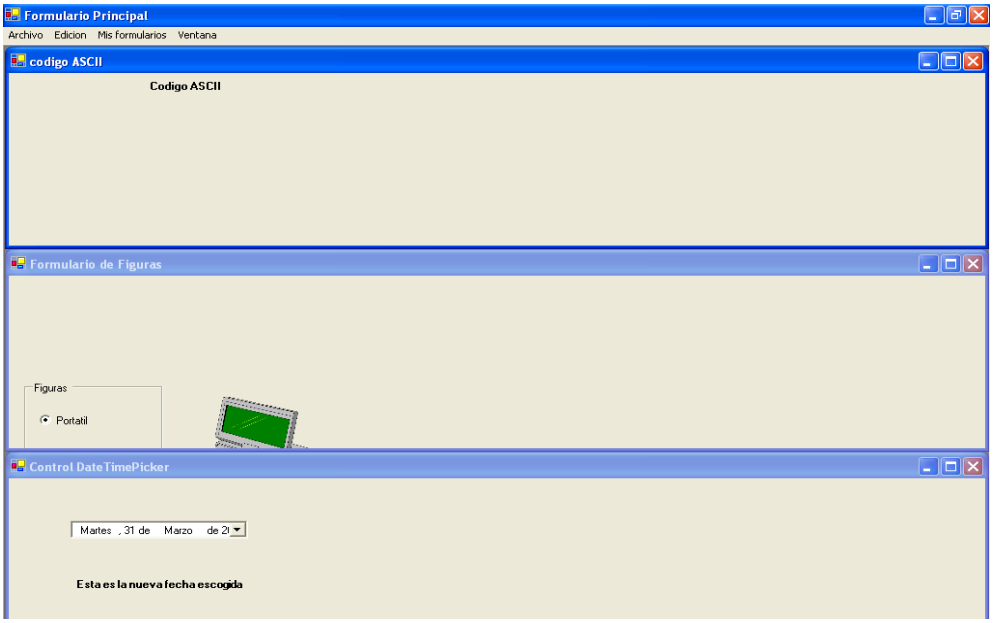


Como se puede apreciar, los formularios abiertos se organizan en cascada y

también se visualiza el nombre de los diferentes archivos abiertos y por medio de un cuadro de verificación se muestra el formulario activo.

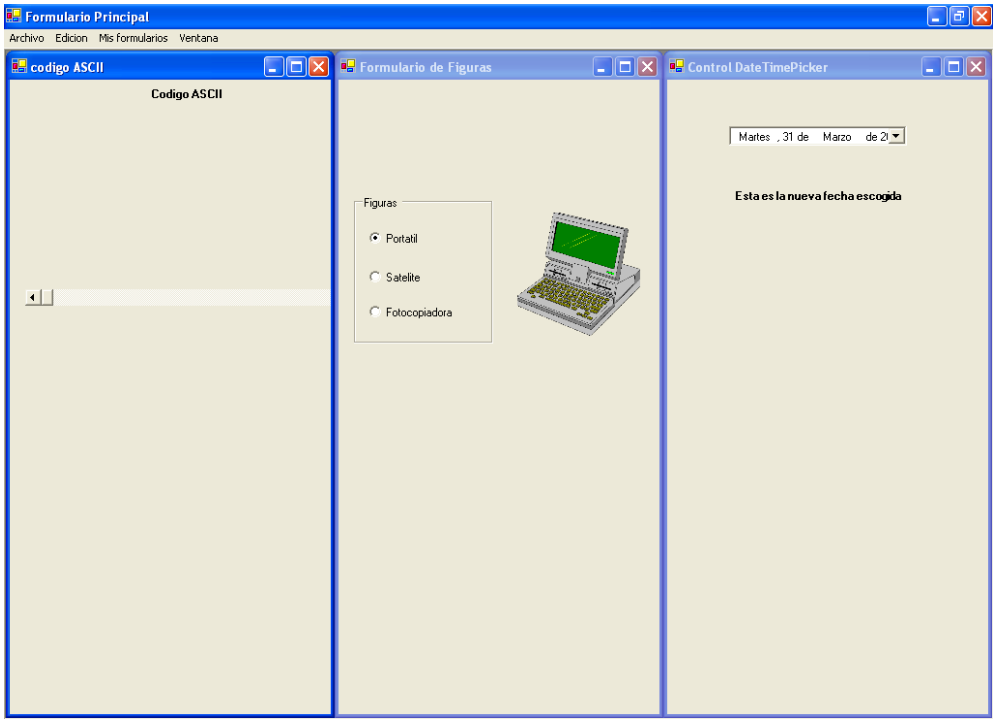
Si se selecciona la opción **Mosaico Horizontal**, se visualizará la siguiente figura:

Figura 8.27 Formularios en horizontal.



Al seleccionar la opción **Mosaico Vertical**, se visualizará la siguiente figura:

Figura 8.28 Formularios en Vertical.



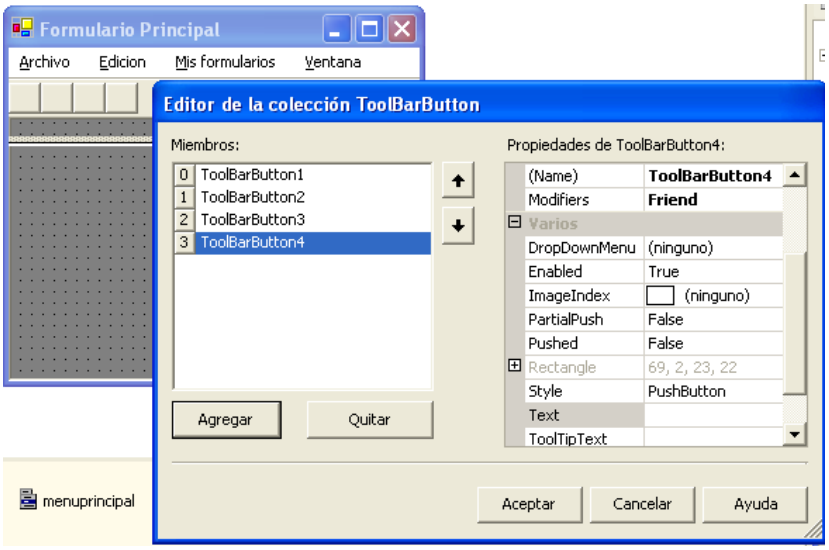
8.5 Creación de una barra de herramientas en un formulario MDI

Siguiendo con el ejemplo que se ha venido trabajando, también es posible añadirle una barra o más barras de herramientas como se maneja en el entorno Windows, con el fin de realizar de forma más rápida algunas tareas, como por ejemplo: copiar, pegar, cortar, guardar, etc. Para añadir una barra de herramientas se utiliza el control **ToolBar**  (**ToolStrip** en Visual Basic 2005). Entonces agregue un control **ToolBar** al ejemplo, para visualizar la siguiente figura:

Figura 8.29 Formulario con barra de herramientas.

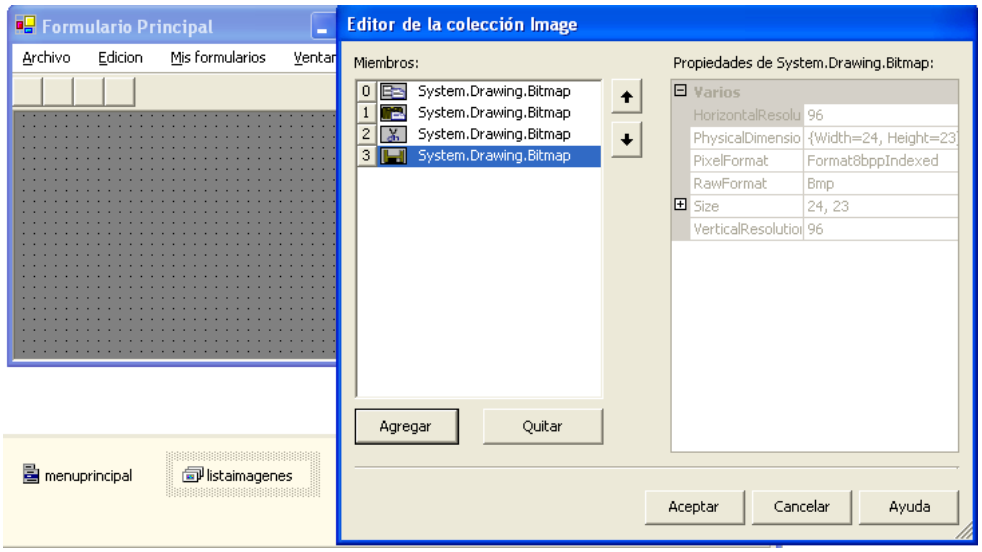
Ahora seleccione el control **ToolBar** cambie el valor de la propiedad **Name** por **barradeherramientas** y luego escoja la propiedad **Buttons** para agregarle botones a esta que permitirá realizar diversas tareas; en este caso agregue cuatro (4) botones. Al seleccionar la propiedad se muestra una ventana de diálogo llamada **Editor de la colección ToolBarButton** y con el botón **Agregar** se podrán agregar los botones que se desean. La figura visualiza la adición de los botones al ejemplo.

Figura 8.30 Editor de la colección ToolbarButton.



Ahora si se desea agregar imágenes a cada botón con el fin de hacerlo más grafico, entonces seleccione el control **ImageList** del cuadro de herramientas y agréguelo al formulario Principal, cambie el valor de la propiedad **Name** por **listaimagenes** y luego seleccione la propiedad **Images** del control **ImageList** para visualizar el **Editor de la colección Image** y con el botón **Agregar** seleccione las imágenes deseadas para incluirlas en los botones, la siguiente figura ilustra la selección de imágenes.

Figura 8.31 Editor de la colección Image.



El siguiente paso es adicionar cada imagen al respectivo botón. Lo primero que se debe hacer es seleccionar el control **barradeherramientas** y elegir la propiedad **ImageList** y agregarle el control **listaimagenes**. Ahora escoja la propiedad **Button** y seleccionando el botón respectivo modifique la propiedad **ImageIndex** y seleccione la imagen correspondiente. Modificando la propiedad **ToolTipText** puede adicionar un texto que indique la operación que realiza cada botón, esta información se visualizará cuando el usuario se acerque con el mouse al botón escogido. Se visualizaría la siguiente figura:

Figura 8.32 Formulario con barra de herramientas e imágenes.

Si se desea programar cada uno de los controles de la barra de herramientas, haga doble clic sobre el control **barradeherramientas** y en el procedimiento escriba el código correspondiente por cada control existente. En este caso se escribirá código para que se ejecute cuando se pulse el ToolbarButton1 de la siguiente manera:

```
If e.Button Is ToolBarButton1 Then
    MsgBox ("Tarea a realizar el control especificado")
End If
```

En el anterior código se pregunta que si el evento que se ha realizado es de tipo **Button** y si el objeto es el **ToolBarButton1** (corresponde al primer boton de la barra de herramientas) entonces se muestra el mensaje correspondiente. Al hacer clic sobre el control se visualiza la siguiente figura:

Figura 8.33 Ejecución de un botón de la barra de herramientas.

9. GRÁFICOS Y ANIMACIÓN

En Microsoft Visual Basic .NET crear efectos especiales de animación es una tarea sencilla. En este capítulo aprenderá a mover los controles en forma horizontal, vertical o diagonal utilizando las propiedades **Left** y **Top**, expandir o contraer los controles manipulando las propiedades **Height** y **Width**, como también crear efectos de animación sencillos utilizando los controles **PictureBox**, **Timer**, **ImageList**.

9.1 Gráficos utilizando el espacio de nombres System.Drawing

El espacio de nombres **System.Drawing** contiene numerosas clases con las que se pueden crear dibujos en un programa. Específicamente con la clase **System.Drawing.Graphics** la cual contiene métodos y propiedades para dibujar figuras en un formulario. El punto de partida del sistema de coordenadas de un formulario es la esquina superior izquierda con coordenada (0,0). El sistema de coordenadas está compuesto por filas y columnas que representan los puntos de una imagen llamados **píxeles**. Las figuras se pueden crear vacías (prefijo **draw**) o con relleno de color (prefijo **fill**).

Para utilizar un método grafico es necesario crear un objeto **System.Drawing.Graphics** y un objeto **Pen** (argumento cuando no se rellena con ningún color) o **Brush** (cuando se quiere tener un color de relleno).En la siguiente tabla se muestran algunas de las figuras geométricas de Visual Basic .NET:

Tabla 9.1 Figuras geométricas de .NET.

Figura	Método	Descripción
Punto (estructura)	Point	Representa un par ordenado de coordenadas **x** e **y** de enteros que define un punto en un plano bidimensional.
Rectangulo (estructura)	Rectangle	Almacena un conjunto de cuatro enteros que representan la posición y tamaño de un rectángulo.
Recta	DrawLine	Dibuja una línea que conecta los dos puntos especificados por los pares de coordenadas
Rectangulo	DrawRectangle, FillRectangle	Dibuja un rectángulo especificado por un par de coordenadas, un valor de ancho y un valor de alto.
Circulo / Elipse	DrawEllipse, FillEllipse	Dibuja una elipse definida por un rectángulo delimitador especificado por un par de coordenadas, un valor de alto y un valor de ancho.
Curva	DrawCurve	Dibuja una curva a través de una matriz especificada por la estructura Point.
Arco	DrawArc	Dibuja un arco especificado por un par de coordenadas, un valor de ancho y un valor de alto.
Poligono	DrawPolygon, FillPolygon	Dibuja un polígono definido por una matriz especificada por la estructura Point.
Pie	DrawPie, FillPie	Dibuja una forma circular definida por una elipse determinada por un par de coordenadas, unos valores de ancho y alto y dos líneas radiales.

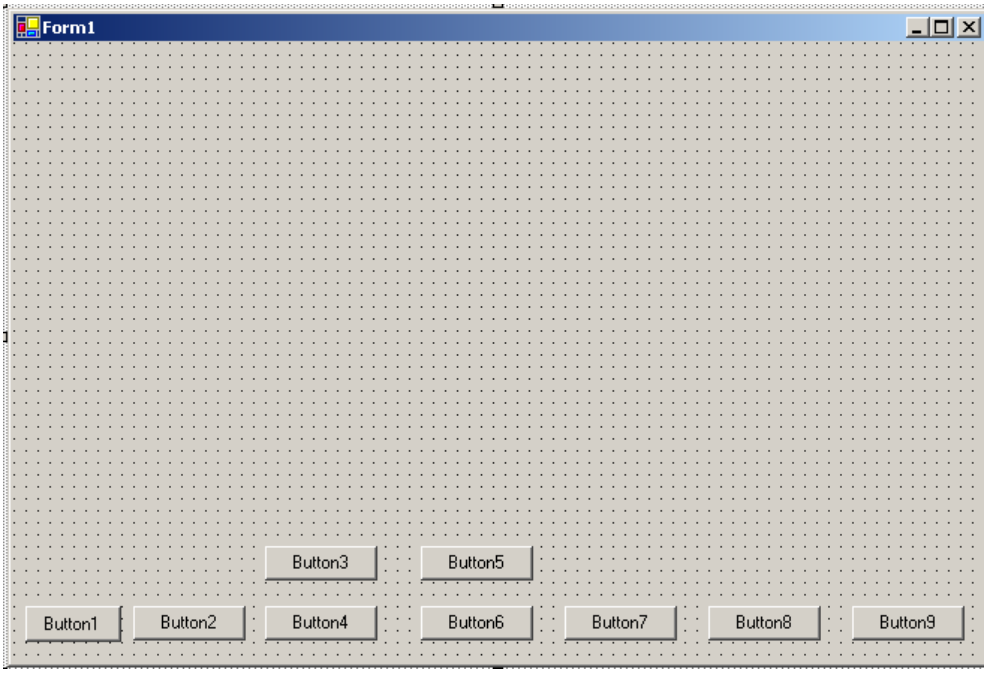
9.1.1 Ejemplo práctico gráficos con System.Drawing.Graphics

Diseñar una aplicación que permita a un usuario graficar: una línea, un rectángulo sin relleno, un rectángulo con relleno, una curva, una elipse sin relleno, una elipse con relleno, un arco , un polígono y una grafica de torta.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control **Button** y adicione nueve (9) al formulario. La figura muestra la interfaz de usuario:

Figura 9.1 Interfaz de usuario DibujosenFormularios.



- **Establecer las propiedades de los objetos de la interfaz de usuario**

Después de colocar los controles u objetos en el formulario, establezca las siguientes propiedades a los controles:

Tabla 9.2. Propiedades de los controles de la aplicación DibujosenFormularios

Control	Propiedad	Valor
Button1	Name	línea
	Text	línea
Button2	Name	curva
	Text	Curva
Button3	Name	rectangulosr
	Text	Rectangulo sin Relleno
Button4	Name	rectangulocr
	Text	Rectangulo con Relleno
Button5	Name	elipsesr
	Text	Elipse sin Relleno

Button6	Name	elipsecr
	Text	Elipse con Relleno
Button7	Name	arco
	Text	Arco
Button8	Name	polígono
	Text	Poligon
Button9	Name	pie
	Text	Torta
Form1	Name	Formulario
	Text	Gráficos Visual Basic .NET

- **Escribir código**

Seleccione el objeto **línea**, dé doble clic para abrir el editor del procedimiento *linea_Clicked* y escriba el siguiente código:

```
Dim graficar As System.Drawing.Graphics
Dim color As New System.Drawing.Pen (System.Drawing.Color.Red)
graficar = Me.CreateGraphics
graficar.DrawLine (color, 10, 20, 60, 80)
```

En el anterior código se define una variable **graficar** de tipo **System.Drawing.Graphics** a dicha variable se le asigna el método **CreateGraphics** para crear graficas, también se crea un objeto **color** de tipo **System.Drawing.Pen** para seleccionar el color de la línea. El método **DrawLine** permite dibujar una línea, dicho método contiene cinco parámetros: el primero es el color de la línea, el segundo es la coordenada **x** inicial, el tercero es la coordenada **y** inicial, el cuarto es la coordenada **x** final y el quinto es la coordenada **y** final.

Seleccione el objeto **curva**, dé doble clic para abrir el editor del procedimiento *curva_Clicked* y escriba el siguiente código:

```
Dim graficar As System.Drawing.Graphics
Dim color As New System.Drawing.Pen(System.Drawing.Color.Blue)
graficar = Me.CreateGraphics
Dim punto1 As New Point(80, 20)
Dim punto2 As New Point(100, 30)
Dim punto3 As New Point(110, 40)
Dim punto4 As New Point(70, 50)
Dim curva As Point () = {punto1, punto2, punto3, punto4}
graficar.DrawCurve (color, curva)
```

Se define una variable **graficar** de tipo **System.Drawing.Graphics** a dicha variable se le asigna el método **CreateGraphics** para crear graficas, también se crea un objeto **color** de tipo **System.Drawing.Pen** para seleccionar el color de la curva. Se definen por medio de la estructura **Point**, ocho puntos los cuales son almacenados es una variable **curva** de tipo **Point** (). El método **DrawCurve** permite dibujar una curva, dicho método contiene dos parámetros: el primero es el color de la curva y el segundo es un arreglo con los puntos definidos.

Seleccione el objeto **rectasr**, dé doble clic para abrir el editor del procedimiento *rectasr_Clicked* y escriba el siguiente código:

```
Dim graficar As System.Drawing.Graphics
Dim color As New System.Drawing.Pen(System.Drawing.Color.Yellow)
graficar = Me.CreateGraphics
graficar.DrawRectangle (color, 150, 10, 200, 100)
```

Se define una variable **graficar** de tipo **System.Drawing.Graphics** a dicha variable se le asigna el método **CreateGraphics** para crear graficas, también se crea un objeto **color** de tipo **System.Drawing.Pen** para seleccionar el color del rectángulo. El método **DrawRectangle** permite dibujar un rectángulo sin relleno, dicho método contiene cinco parámetros: el primero es el color del rectángulo, el segundo es la coordenada **x**, el tercero es la coordenada **y**, el cuarto es el ancho y el quinto es el alto.

Seleccione el objeto **arco**, dé doble clic para abrir el editor del procedimiento *arco_Clicked* y escriba el siguiente código:

```
Dim graficar As System.Drawing.Graphics
Dim color As New System.Drawing.Pen(System.Drawing.Color.Black)
graficar = Me.CreateGraphic
graficar.DrawArc (color, 140, 120, 200, 100, 100, -180)
```

Se define una variable **graficar** de tipo **System.Drawing.Graphics** a dicha variable se le asigna el método **CreateGraphics** para crear graficas, también se crea un objeto **color** de tipo **System.Drawing.Pen** para seleccionar el color del arco. El método **DrawArc** permite dibujar un arco, dicho método contiene siete parámetros: el primero es el color del arco, el segundo es la coordenada **x**, el tercero es la coordenada **y**, el cuarto es el ancho, el quinto es el alto, el sexto es el ángulo inicial y el séptimo es el ángulo final.

Seleccione el objeto **elipsesr**, dé doble clic para abrir el editor del procedimiento *elipsesr_Clicked* y escriba el siguiente código:

```
Dim graficar As System.Drawing.Graphics
Dim color As New System.Drawing.Pen(System.Drawing.Color.Green)
graficar = Me.CreateGraphics
graficar.DrawEllipse (color, 10, 140, 100, 60)
```

Se define una variable **graficar** de tipo **System.Drawing.Graphics** a dicha variable se le asigna el método **CreateGraphics** para crear graficas, también se crea un objeto **color** de tipo **System.Drawing.Pen** para seleccionar el color de la elipse. El método **DrawEllipse** permite dibujar una elipse sin relleno, dicho método contiene cinco parámetros: el primero es el color de la elipse, el segundo es la coordenada **x**, el tercero es la coordenada **y**, el cuarto es el ancho y el quinto es el alto.

Seleccione el objeto **rectacr**, dé doble clic para abrir el editor del procedimiento *rectacr_Clicked* y escriba el siguiente código:

```
Dim graficar As System.Drawing.Graphics
Dim relleno As New SolidBrush(Color.Salmon)
graficar = Me.CreateGraphics
graficar.FillRectangle (relleno, 400, 10, 430, 100)
```

Se define una variable **graficar** de tipo **System.Drawing.Graphics** a dicha variable se le asigna el método **CreateGraphics** para crear graficas, también se crea un objeto **relleno** de tipo **SolidBrush** para seleccionar el color de relleno del rectángulo. El

método **FillRectangle** permite dibujar un rectángulo con color de relleno, dicho método contiene cinco parámetros: el primero es el color de relleno, el segundo es la coordenada **x**, el tercero es la coordenada **y**, el cuarto es el ancho y el quinto es el alto.

Seleccione el objeto **elipsecr**, dé doble clic para abrir el editor del procedimiento *elipsecr_Clicked* y escriba el siguiente código:

```
Dim graficar As System.Drawing.Graphics
Dim relleno As New SolidBrush(Color.Gray)
graficar = Me.CreateGraphics
graficar.FillEllipse (relleno, 140, 120, 80, 80)
```

Se define una variable **graficar** de tipo **System.Drawing.Graphics** a dicha variable se le asigna el método **CreateGraphics** para crear graficas, también se crea un objeto **relleno** de tipo **SolidBrush** para seleccionar el color de relleno de la elipse. El método **FillEllipse** permite dibujar una elipse con relleno, dicho método contiene cinco parámetros: el primero es el color de relleno de la elipse, el segundo es la coordenada **x**, el tercero es la coordenada **y**, el cuarto es el ancho y el quinto es el alto.

Seleccione el objeto **polígono**, dé doble clic para abrir el editor del procedimiento *poligono_Clicked* y escriba el siguiente código:

```
Dim graficar As System.Drawing.Graphics
Dim relleno As New SolidBrush(Color.Chocolate)
graficar = Me.CreateGraphics
Dim punto1 As New Point(500, 140)
Dim punto2 As New Point(460, 180)
Dim punto3 As New Point(460, 220)
Dim punto4 As New Point(500, 260)
Dim punto5 As New Point(540, 260)
Dim punto6 As New Point(580, 220)
Dim punto7 As New Point(580, 180)
Dim punto8 As New Point(540, 140)
Dim curva As Point () = {punto1, punto2, punto3, punto4,
                         punto5, punto6, punto7, punto8}
graficar.FillPolygon (relleno, curva, Drawing2D.FillMode.Winding)
```

Se define una variable **graficar** de tipo **System.Drawing.Graphics** a dicha variable se le asigna el método **CreateGraphics** para crear graficas, también se crea un objeto **relleno** de tipo **SolidBrush** para seleccionar el color de relleno de un polígono. Se especifica por medio de la estructura **Point**, ocho puntos los cuales son almacenados es una variable **curva** de tipo **Point**. El método **FillPolygon** permite dibujar un polígono, dicho método contiene tres parámetros: el primero es el color de relleno, el segundo es el arreglo de puntos y el tercero es el tipo de relleno.

Seleccione el objeto ***pie***, dé doble clic para abrir el editor del procedimiento *pie_Clicked* y escriba el siguiente código:

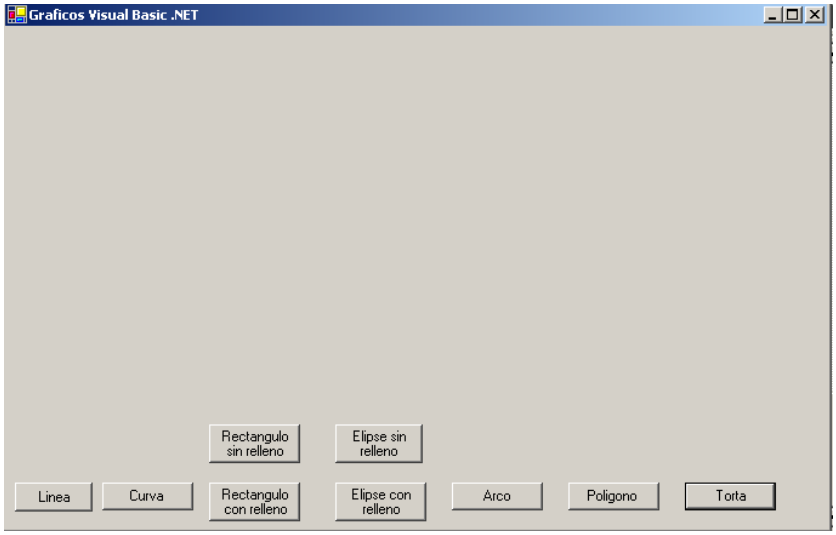
```
Dim graficar As System.Drawing.Graphics
Dim color As New System.Drawing.Pen(System.Drawing.Color.Red)
graficar = Me.CreateGraphics
graficar.DrawPie(color, New Rectangle(New Point(380, 240),
                 New Size (150, 85)), 0, 270)
```

Se define una variable **graficar** de tipo **System.Drawing.Graphics** a dicha variable se le asigna el método **CreateGraphics** para crear graficas, también se crea un objeto **color** de tipo **System.Drawing.Pen** para seleccionar el color del grafico. Se especifican por medio de la estructura **Rectangle**, dos puntos los cuales son almacenados es una variable **curva** de tipo **Point**. El método **DrawPie** permite dibujar una torta, dicho método contiene tres parámetros: el primero es una estructura **Rectangle** donde se especifican dos puntos, el segundo es el ancho y alto definido por la estructura **Size**, el tercero es el ángulo inicial y el cuarto es el ángulo final.

- **Ejecutar el proyecto**

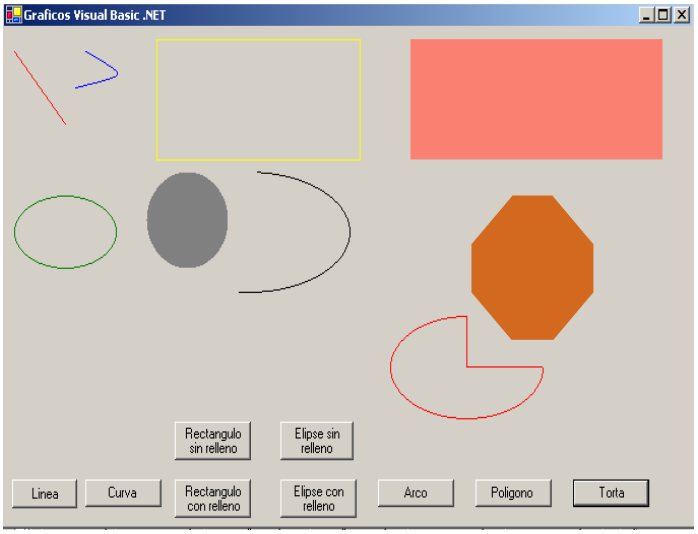
Al ejecutar el proyecto en el entorno de desarrollo de visual Basic.NET, se visualiza la siguiente figura:

Figura 9.2 Ejecución aplicación DibujosenFormularios.



Al pulsar cada uno de los botones de obtendrá la siguiente figura:

Figura 9.3 Formulario con figuras geométricas.



9.2 Movimiento de un control

Para que un control se mueva en forma horizontal, vertical o diagonal es necesario modificar las propiedades del control **Top** y **Left**. La propiedad **Left** permite mover un control de izquierda a derecha y viceversa. La propiedad **Top** permite mover un objeto de arriba abajo y viceversa. La combinación de **Left** y **Top** permite el movimiento de un control en forma diagonal. Existen dos formas de realizar esta operación: manualmente y automáticamente. De la forma manual se puede realizar haciendo un clic sobre un botón cada vez que el usuario desea mover la figura. Dicho control contendrá el código que permitirá realizar un movimiento específico. La forma automática se realiza por medio de un control **Timer** que permite manejar intervalos de tiempo y el cual se puede programar para que realice unas tareas especificas en un determinado intervalo de tiempo.

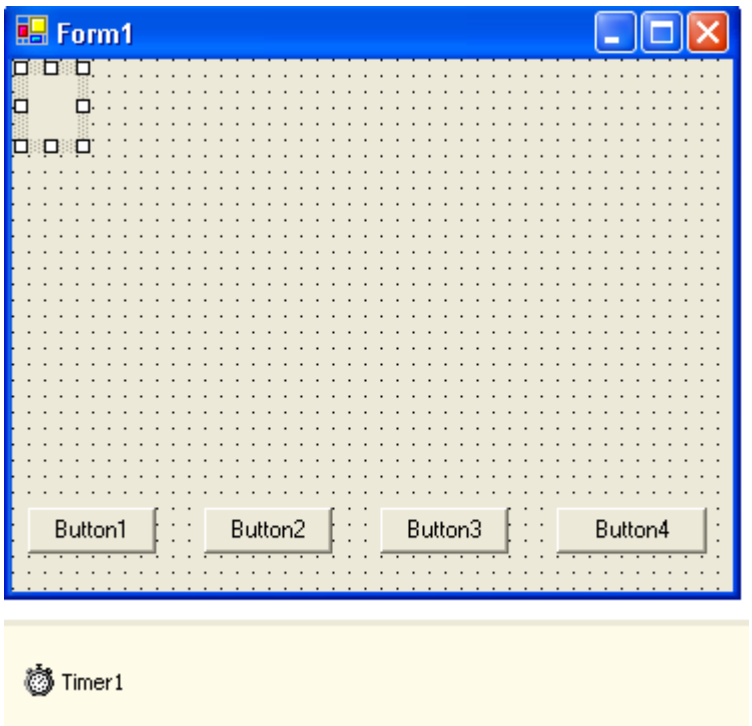
9.2.1 Ejemplo práctico movimiento de un control

Realizar una aplicación que permita a un usuario por medio de botones mover una imagen en forma horizontal y vertical en forma manual y automática.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en los controles PictureBox y Timer y adiciónelos al formulario. También adicione cuatro Button al formulario. La figura muestra la interfaz de usuario:

Figura 9.4. Interfaz de usuario



- **Establecer las propiedades de los objetos de la interfaz de usuario**

Después de colocar los controles u objetos en el formulario, establezca las siguientes propiedades a los controles:

Tabla 9.3 Propiedades de los controles de la aplicación MovimientoImagen.

Control	Propiedad	Valor
PictureBox1	Name	imagen
	SizeMode	StretchImage
	Image	Seleccione la imagen deseada
Button1	Name	Abajo
	Text	Mover a Abajo
Button2	Name	Derecho
	Text	Mover a la Derecha
Button3	Name	AutoAbajo
	Text	Mover Automáticamente a Abajo
Button4	Name	Autoderecho
	Text	Mover Automáticamente a la Derecha
Form1	Name	Formulario
	Text	Movimiento de una Imagen
Timer1	Name	reloj
	Interval	500

- **Escribir código**

Seleccione el objeto **abajo**, dé doble clic para abrir el editor del procedimiento *abajo*`_Clicked` y escriba el siguiente código:

```
imagen.top = imagen.top + 10
```

En este código se modifica el valor de la propiedad **Top** del objeto **imagen** en 10 píxeles, lo que hace que cuando el usuario de clic sobre el botón la imagen se desplazará 10 píxeles hacia abajo. Si se desea que el control **imagen** se devuelva los mismos 10 píxeles hacia arriba modifique la propiedad **Top** en -10.

Seleccione el objeto **derecha**, dé doble clic para abrir el editor del procedimiento y escriba el siguiente código:

```
imagen.left = imagen.left + 10
```

En este código se modifica el valor de la propiedad **Left** del objeto **imagen** en 10 píxeles, lo que hace que cuando el usuario de clic sobre el botón la imagen se desplazará hacia la derecha 10 píxeles. Si se desea que el control **imagen** se devuelva los mismos 10 píxeles hacia la izquierda modifique la propiedad **Left** en -10.

Seleccione el objeto **autoabajo**, de doble clic para abrir el editor del procedimiento y escriba el siguiente código:

```
reloj.enabled=true
```

En este código se modifica el valor de la propiedad **Enabled** del control **reloj** para que se active el reloj en el intervalo especificado. Además de realizar esta operación también se debe seleccionar el **Timer** y escribir el siguiente código:

```
imagen.top = imagen.top + 10
```

Esto permitirá que cuando se haga clic sobre el botón **Mover automáticamente hacia abajo**, la imagen se desplazará 10 píxeles indefinidamente hacia abajo. Si lo que se desea es hacer clic sobre el botón **Mover automáticamente a la derecha** en el procedimiento del objeto **autoderecho** escriba el siguiente código:

```
reloj.enabled=true
```

Y en el procedimiento del Timer escriba el código:

```
imagen.Left = imagen.Left + 10
```

Ahora si se quiere que la imagen se desplace diagonalmente hacia la derecha, escriba el siguiente código en el Timer:

```
imagen.top = imagen.top + 10
imagen.Left= imagen.Left + 10
```

Si lo que se desea es un desplazamiento diagonalmente hacia la izquierda a las propiedades **Top** y **Left** réstele 10 píxeles. También se podría agregar otro botón que le permitiera detener el reloj, escribiéndole el siguiente código:

```
reloj.enabled=false
```

- **Ejecutar el proyecto**

Al ejecutar el proyecto en el entorno de desarrollo de visual Basic.NET, se debe representa la siguiente figura:

Figura 9.5. Ejecución aplicación MovimientoImagen

Ahora haga clic sobre el botón deseado y visualice el movimiento de la imagen.

9.3 Expandir y contraer un control

Para que un control se pueda expandir o contraer es necesario modificar las propiedades del control **Width** y **Height**. La propiedad **Width** permite modificar el ancho del control y la propiedad **Height** permite modificar la altura de un objeto. También se puede realizar esta operación en forma manual y automática.

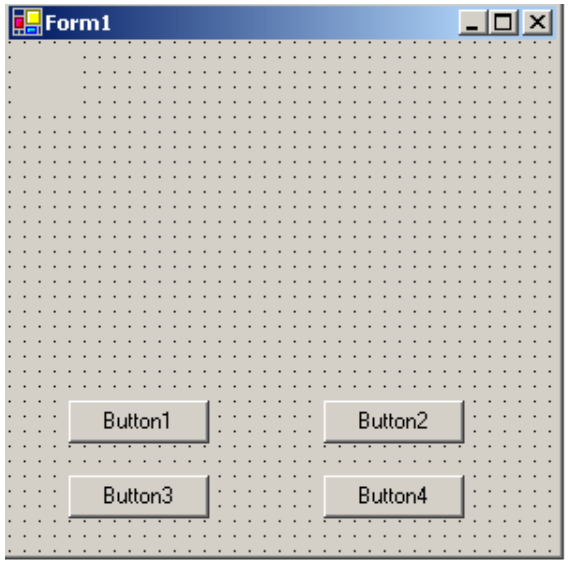
9.3.1 Ejemplo práctico expandir y contraer un control

Realizar una aplicación que permita a un usuario por medio de botones expandir y contraer una imagen en forma manual y automática.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control PictureBox y adiciónelo al formulario. También seleccione dos Timer y cuatro Button al formulario. La figura muestra la interfaz de usuario:

Figura 9.6 Interfaz de usuario ExpandirContraerImagen.



- **Establecer las propiedades de los objetos de la interfaz de usuario**

Después de colocar los controles u objetos en el formulario, establezca las siguientes propiedades a los controles:

Tabla 9.4 Propiedades de los controles de la aplicación ExpandirContraerImagen.

Control	Propiedad	Valor
PictureBox1	Name	imagen
	SizeMode	StretchImage
	Image	Seleccione la imagen deseada
Button1	Name	expandir
	Text	Expandir Imagen Manualmente
Button2	Name	Contraer
	Text	contraer Imagen Manualmente
Button3	Name	expandirAuto
	Text	Expandir Imagen Automáticamente
Button4	Name	ContraerAuto
	Text	contraer Imagen Automáticamente
Form1	Name	Formulario
	Text	Expandir y contraer una imagen
Timer1	Name	Reloj1
	Interval	500
Timer2	Name	Reloj2
	Interval	500

- **Escribir código**

Seleccione el objeto **expandir**, dé doble clic para abrir el editor del procedimiento y escriba el siguiente código:

```
imagen.Height = imagen.Height + 10
imagen.Width = imagen.Width + 10
```

En este código se modifica el valor de las propiedades **Width** y **Height** en 10 píxeles, lo que hace que cuando el usuario de clic sobre el botón la imagen se vuelva 10 píxeles más ancha y alta.

Seleccione el objeto **contraer**, dé doble clic para abrir el editor del procedimiento y escriba el siguiente código:

```
Imagen.Height = Imagen.Height - 10
Imagen.Width = Imagen.Width - 10
```

En este código se modifica el valor de las propiedades **Width** y **Height** en -10 píxeles, lo que hace que cuando el usuario de clic sobre el botón la imagen se vuelva 10 píxeles menos ancha y alta.

Seleccione el objeto **expandirAuto**, de doble clic para abrir el editor del procedimiento y escriba el siguiente código:

```
reloj1.Enabled = True
reloj2.Enabled = False
```

En este código se modifica el valor de la propiedad **Enabled** de los controles **Timer**, lo que permite que el usuario cuando haga clic sobre el botón se active el control **reloj1** y se desactive el control **reloj2**.

Seleccione el objeto **reloj2**, de doble clic para abrir el editor del procedimiento y escriba el siguiente código:

```
Imagen.Height = Imagen.Height - 10
Imagen.Width = Imagen.Width - 10
```

En este código se modifica el valor de las propiedades **Width** y **Height** en -10 píxeles, lo que hace que cuando el usuario de clic sobre el botón la imagen se vuelva 10 píxeles menos ancha y alta en forma automática.

Seleccione el objeto **reloj1**, de doble clic para abrir el editor del procedimiento y escriba el siguiente código:

```
Imagen.Height = Imagen.Height + 10
Imagen.Width = Imagen.Width + 10
```

En este código se modifica el valor de las propiedades **Width** y **Height** en 10 píxeles, lo que hace que cuando el usuario de clic sobre el botón la imagen se vuelva 10 píxeles más ancha y alta en forma automática.

Seleccione el objeto **contraerAuto**, de doble clic para abrir el editor del procedimiento y escriba el siguiente código:

```
reloj1.Enabled = False
reloj2.Enabled = True
```

En este código se modifica el valor de la propiedad **Enabled** de los controles **Timer**, lo que permite que el usuario cuando haga clic sobre el botón se active el control **reloj2** y se desactive el control **reloj1**.

- **Ejecutar el proyecto**

Al ejecutarse el proyecto en el entorno de desarrollo de visual Basic.NET, se visualiza la figura 9.7:

Figura 9.7 Ejecución aplicación ExpandirContraerImagen.

Ahora haga clic sobre el botón deseado y visualice la expansión o contracción de la imagen.

9.4 Ejemplos prácticos de animación

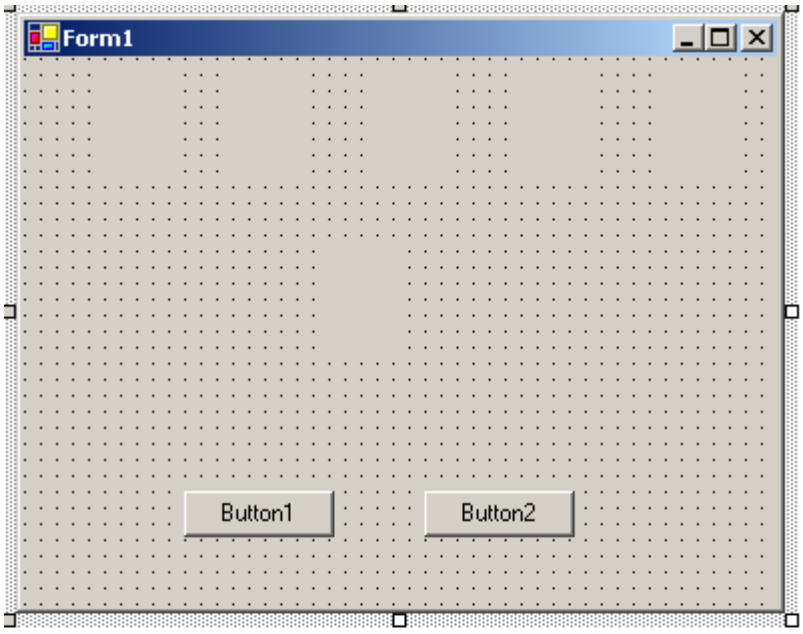
Ejemplo 1. Caritas

a) Realizar una aplicación donde realice la simulación de animación de varias imágenes de caritas.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control **PictureBox** y adicione seis controles y un control **Timer** al formulario. Además adicione 2 **Button** al formulario. La figura muestra la interfaz de usuario:

Figura 9.8 Interfaz de usuario Caritas.



- **Establecer las propiedades de los objetos de la interfaz de usuario**

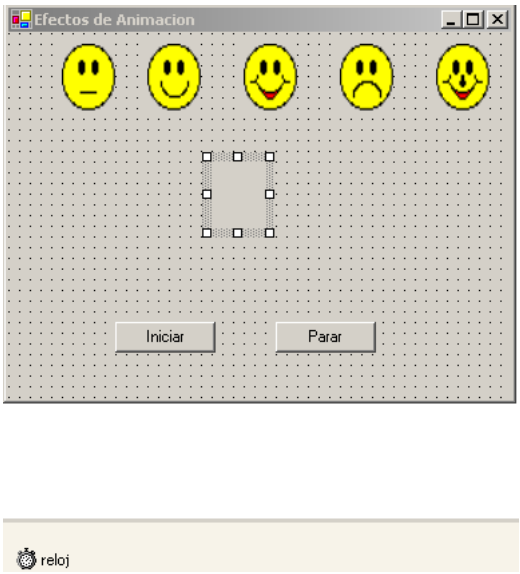
Después de colocar los controles u objetos en el formulario, establezca las siguientes propiedades a los controles:

Tabla 9.5 Propiedades de los controles de la aplicación Caritas.

Control	Propiedad	Valor
PictureBox1	Name	carita1
	SizeMode	StretchImage
	Image	Seleccione la imagen deseada (face01.ico)
	Visible	False
PictureBox2	Name	carita2
	SizeMode	StretchImage
	Image	Seleccione la imagen deseada (face02.ico)
	Visible	False
PictureBox3	Name	Carita3
	SizeMode	StretchImage
	Image	Seleccione la imagen deseada (face03.ico)
	Visible	False
PictureBox4	Name	Carita4
	SizeMode	StretchImage
	Image	Seleccione la imagen deseada (face04.ico)
	Visible	False
PictureBox5	Name	Carita5
	SizeMode	StretchImage
	Image	Seleccione la imagen deseada (face05.ico)
	Visible	False
PictureBox6	Name	animacion
	SizeMode	StretchImage
	Visible	True
Button1	Name	inicia
	Text	Iniciar
Button2	Name	Para
	Text	Parar
Form1	Name	Formulario
	Text	Efectos de Animación
Timer1	Name	reloj
	Interval	500

El formulario se visualizaría como muestra la siguiente figura:

Figura 9.9 Formulario con controles modificados de la aplicación Caritas.



- **Escribir código**

Cree una variable global llamada **mover**. Dicha variable se crea pulsando doble clic sobre el formulario y en el inicio del editor buscar la clase Formulario:

```
Public Class Formulario
   Inherits System.Windows.Forms.Form
  Dim mover As Integer
```

Seleccione el objeto **formulario**, dé doble clic para abrir el editor del procedimiento `Formulario_Load` y escriba el siguiente código:

```
mover=0
```

Seleccione el objeto **inicia**, dé doble clic para abrir el editor del procedimiento *inicia_`Clicked`* y escriba el siguiente código:

```
reloj.Enabled = True
```

En este código se modifica el valor lógico de la propiedad **Enabled** del control **reloj** para activar el control **Timer**.

Seleccione el objeto **para**, dé doble clic para abrir el editor del procedimiento *para_`Clicked`* y escriba el siguiente código:

```
reloj.Enabled = False
```

En este código se modifica el valor lógico de la propiedad **Enabled** del control **reloj** para desactivar el control **Timer**.

Seleccione el objeto **reloj**, de doble clic para abrir el editor del procedimiento y escriba el siguiente código:

```
If mover > 5 Then
   mover = 0
End If
Select Case mover
  Case 0
    animacion.Image = carita1.Image
  Case 1
    animacion.Image = carita2.Image
  Case 2
    animacion.Image = carita3.Image
  Case 3
    animacion.Image = carita4.Image
  Case 4
    animacion.Image = carita5.Image
End Select
mover = mover + 1
```

En este código se evalúa el valor de la variable **mover**. Si el valor es mayor que cinco, la variable se inicializa en cero. También se evalúa dicho valor para asignarle al control **animación** la imagen correspondiente y por último se va incrementando el valor de la variable **mover**.

- **Ejecutar el proyecto**

Cuando se ejecuta el proyecto en el entorno de desarrollo de visual Basic.NET se visualiza la siguiente pantalla:

Figura 9.10 Ejecución aplicación caritas.

Al hacer clic sobre el botón **Iniciar** se inicia el efecto de animación. Si se desea detener el efecto de animación se debe pulsar el botón **Parar**.

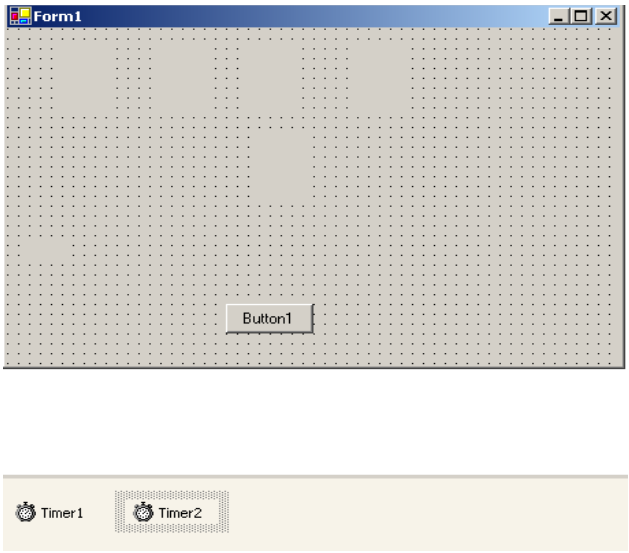
Ejemplo 2. Semaforo

b) Realizar una aplicación que simule la operación de un semáforo.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control **PictureBox** y adicione seis controles y dos controles **Timer** al formulario. Además, seleccione un control **Button** y adiciónelo al formulario. La figura muestra la interfaz de usuario:

Figura 9.11 Interfaz de usuario Semaforo.



- **Establecer las propiedades de los objetos de la interfaz de usuario**

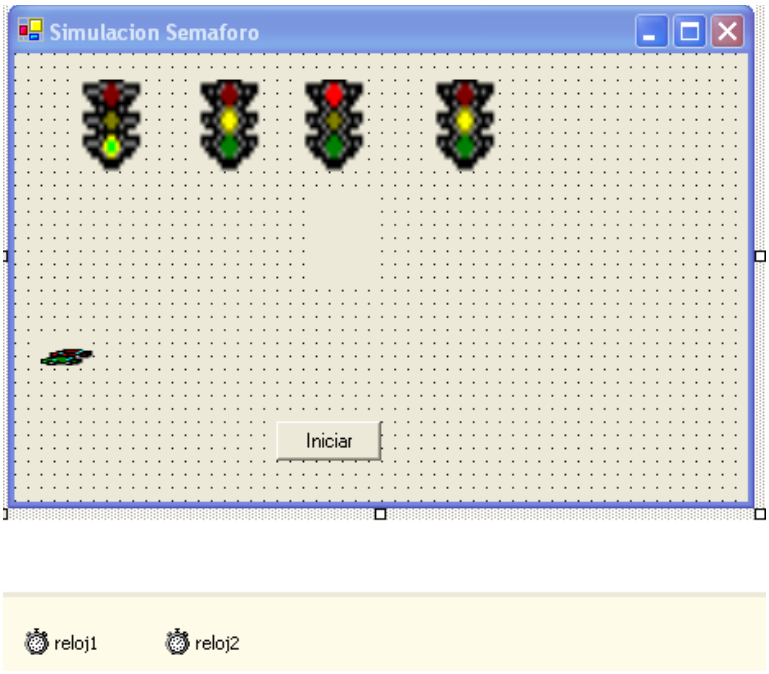
Después de colocar los controles u objetos en el formulario, establezca las siguientes propiedades a los controles:

Tabla 9.6 Propiedades de los controles de la aplicación MovimientoImagen.

Control	Propiedad	Valor
PictureBox1	Name	Semaforo1
	SizeMode	StretchImage
	Image	Seleccione la imagen deseada (trffc10A.ico)
	Visible	False
PictureBox2	Name	semaforo2
	SizeMode	StretchImage
	Image	Seleccione la imagen deseada (trffc10B.ico)
	Visible	False
PictureBox3	Name	Semaforo3
	SizeMode	StretchImage
	Image	Seleccione la imagen deseada (trffc10C.ico)
	Visible	False
PictureBox4	Name	Semafor4
	SizeMode	StretchImage
	Image	Seleccione la imagen deseada (trffc10B.ico)
	Visible	False
PictureBox5	Name	Animación
	SizeMode	StretchImage
	Visible	True
PictureBox6	Name	Carro
	SizeMode	StretchImage
	Visible	True
Button1	Name	botón
	Text	Iniciar
Form1	Name	Formulario
	Text	Simulación Semáforo
Timer1	Name	Reloj1
	Interval	500
	Enabled	False
Timer2	Name	Reloj2
	Interval	500
	Enabled	False

El formulario se visualizaría como muestra la siguiente figura:

Figura 9.12 Formulario con controles modificados de la aplicación Semaforo.



- **Escribir código**

Cree una variable global llamada **mover**. Dicha variable se crea pulsando doble clic sobre el formulario y en el inicio del editor buscar la clase Formulario:

```
Public Class Formulario
    Inherits System.Windows.Forms.Form
   Dim mover As Integer
```

Seleccione el objeto **formulario**, dé doble clic para abrir el editor del procedimiento `Formulario_Load` y escriba el siguiente código:

```
mover=0
```

Seleccione el objeto **botón**, dé doble clic para abrir el editor del procedimiento *inicia*`_Clicked` y escriba el siguiente código:

```
Reloj1.Enabled = True
```

En este código se modifica el valor lógico de la propiedad **Enabled** del control **reloj1** para activar el control **Timer**.

Seleccione el objeto **reloj2**, dé doble clic para abrir el editor del procedimiento y escriba el siguiente código:

```
carro.left=carro.left+10
```

En este código se modifica el valor de la propiedad **Left** del objeto **carro** en 10 píxeles.

Seleccione el objeto **reloj1**, dé doble clic para abrir el editor del procedimiento y escriba el siguiente código:

```
If mover = 4 Then
  mover = 1
Else
  mover = mover + 1
End If
Select Case mover
      Case 1
          reloj1.Interval = 800
          reloj2.Enabled = True
          animacion.Image = semaforo1.Image
      Case 2
          reloj1.Interval = 800
          reloj2.Enabled = False
          animacion.Image = semaforo2.Image
     Case 3
          reloj1.Interval = 2000
          reloj2.Enabled = False
          animacion.Image = semaforo3.Image
    Case 4
          reloj1.Interval = 800
          reloj2.Enabled = False
          animacion.Image = semaforo4.Image
End Select
```

En este código se evalúa el valor de la variable **mover**. Si el valor es igual a cuatro, la variable se inicializa en uno. También se evalúa dicho valor para asignarle al control **animación** la imagen correspondiente y modificar las propiedades **Interval** y **Enabled** de los controles **reloj1** y **reloj2** respectivamente.

- **Ejecutar el proyecto**

Al ejecutar el proyecto en el entorno de desarrollo de visual Basic.NET, se visualiza la siguiente figura:

Figura 9.13 Ejecución aplicación semáforo.

Al hacer clic sobre el botón **Iniciar** se inicia el efecto de animación.

9.5 Ejercicios de graficas y animación

1. Escribir un programa que dibuje diez círculos concéntricos. Los círculos deben estar separados por 5 píxeles utilizando DrawEllipse.
2. Realizar un programa que dibuje líneas de longitud y color aleatorio.
3. Hacer un programa que dibuje un espiral utilizando DrawArc.
4. Diseñar un programa que dibuje 10 palabras con fuente aleatoria de diferente tamaño.
5. Crear un programa que capture una palabra e imprima dicha palabra en forma aleatoria y de diferente color.
6. Escribir un programa que dibuje un tablero de ajedrez
7. Realizar un programa que dibuje un cubo
8. Hacer un programa que capture el número de triángulos que deben dibujarse. Dicho triángulos deben ser de diferente color.
9. Diseñar un programa que lea un par de coordenadas, el radio y dibuje el círculo, además imprimir el diámetro, la circunferencia y el área del círculo.
10. Crear un programa que simule un protector de pantalla. El programa deberá dibujar 50 líneas al azar y después limpiar la pantalla y viceversa.

10. LA PROGRAMACIÓN ORIENTADA A OBJETOS CON .NET

La Programación orientada a objetos (P.O.O) encapsula datos (atributos) y métodos (comportamientos) en objetos y trata de encontrar solución de problemas. En la P.O.O. se utilizan los siguientes conceptos:

- **Objetos**: Entidades provistas de datos (propiedades, atributos) y comportamientos (funcionalidad, programas, métodos). Corresponden a los objetos reales del mundo que nos rodea. Un objeto de una determinada clase se invoca utilizando el operador **New** para dicha clase. Por ejemplo, en la siguiente línea se crea un objeto de la clase ClaseCuenta y se asigna a la variable MiNuevaClase una instancia de la misma.

```
Dim miNuevaInstancia As ClaseCuenta=New ClaseCuenta ()
```

- **Clases**: Conjuntos de objetos que comparten propiedades y comportamientos.
- **Abstracción**: Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar cómo se implementan estas características.
- **Encapsulación**: También llamada **ocultación de la información**, esto asegura que los objetos no pueden cambiar el estado interno de otros objetos de manera inesperada; solamente los propios métodos internos del objeto pueden acceder a su estado. Cada tipo de objeto expone una interfaz a otros objetos que especifica cómo otros objetos pueden interactuar con él.
- **Polimorfismo**: Programas diferentes, asociados a objetos distintos, pueden compartir el mismo nombre, aunque el significado del programa varíe según el objeto al que se aplica.
- **Herencia**: Organiza y facilita el polimorfismo y la encapsulación permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementar su comportamiento.
- **Método**: Es un programa asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena mediante un "mensaje".
- **Mensaje**: Una comunicación dirigida a un objeto, que le ordena que ejecute uno de sus métodos con ciertos parámetros.
- **Propiedad, atributo o variable**: Datos asociados a un objeto o a una clase de objetos.

La P.O.O. expresa un programa como un conjunto de objetos, que se comunican entre ellos para realizar tareas; siendo un objeto una unidad que contiene datos y métodos que operan sobre esos datos. Los objetos tienen la propiedad de ocultamiento de la información, esto significa que los objetos saben comunicarse entre sí a través de interfaces (normalmente no se permite que un objeto sepa cómo están implementados otros objetos).

Los objetos pueden ser activados mediante la recepción de mensajes. Un mensaje es simplemente una petición para que un objeto se comporte de una determinada manera ejecutando un método. La técnica de enviar mensajes se conoce como paso de mensajes.

Ejemplo: Se tiene un objeto **Alumno** con los siguientes datos: **nombre_alumno, apeellido_alumno** y un objeto **curso** con los métodos: **leer_datos** e **imprimir_datos**.

Si el objeto **Alumno** recibe el mensaje **imprimir_datos**, esto se expresa:

```
Alumno.imprimir_datos ()
```

La sentencia anterior se lee **enviar mensaje imprimir_datos al objeto Alumno**. El objeto **Alumno** reacciona al mensaje ejecutando el método que ha sido llamado.

Una clase es un tipo definido por el usuario que determina las estructuras de datos y las operaciones asociadas con ese tipo. Cada vez que se construye un objeto de una clase, se crea una instancia de esa clase. En general, los términos objetos e instancias de una clase se pueden utilizar indiferentemente. Una clase es una colección de objetos similares y un objeto es una instancia de una definición de una clase. La comunicación con el objeto se realiza a través del paso de mensajes. El envío a una instancia de una clase produce la ejecución de un método. Para declarar una clase se utiliza la palabra reservada **Class** seguida del nombre de la clase y el cuerpo de la misma, terminándose con **End Class**. El cuerpo de la clase incluye los atributos y los métodos.

```
Class Alumno
   Cuerpo de clase (Atributos y métodos)
End Class
```

Toda clase debe contener una definición de variables o métodos precedida por un modificador de acceso a los miembros; los modificadores de acceso a miembros pueden aparecer varias veces y en cualquier orden en una definición de una clase.

Los datos o variables definidos en una clase se llaman variables de instancias. El código está contenido en los métodos. Los métodos y las variables de instancias definidas en una clase son los miembros de la clase.

```
Class Alumno
   Private  nombre_alumno As String
   Private asignatura As String
   Private curso As integer
   Public Function obtenerNombre(nombre As String) As String
        return nombre
   End Function
   Public Function obtenerCurso(curso As Integer) As Integer
        return curso
   End Function
   …..
End Class
```

Un programa orientado a objetos es una colección de clases. Se necesita un método principal que cree objetos y comience la ejecución mediante la invocación de sus métodos. En realidad, cuando se ejecuta un programa orientado a objetos ocurren tres acciones:

1. Se crean los objetos cuando se necesitan
2. Los mensajes se envían desde unos objetos y se reciben en otros
3. Se borran los objetos cuando ya no son necesarios y se recupera la memoria ocupada por ellos.

10.1 Modificadores de control de acceso

La encapsulación de datos proporciona el atributo de control de acceso. A través de la encapsulación se puede controlar que partes de un programa pueden acceder a los miembros de una clase para prevenir cambios no deseables en los datos. El acceso a un miembro de una clase se determina con el especificador de acceso que modifica su declaración. Los modificadores de acceso de Visual Basic.NET son: **public** (público), **private** (privado) y **protected** (protegido).

Cuando un miembro de una clase tiene el especificador **public**, ese miembro puede ser accedido por cualquier parte del programa. Cuando un miembro es **protected**, ese miembro puede ser utilizado para realizar la herencia. Cuando un miembro es **private**, ese miembro sólo puede ser accedido por otros miembros de esa clase.

Cuando no se utiliza ningún especificador de acceso, por defecto los miembros de una clase son públicos dentro de su propio paquete. Un especificador precede al resto de la especificación de tipo del miembro de la clase, es decir, debe iniciar una sentencia de declaración de un miembro.

```
Public numero As Integer
Private cuenta As double
Public Sub  mifuncion(x As integer, y As String)
Private function  mifuncion() As Integer
```

10.2 Constructores

Se llama constructor a un método que tiene el mismo nombre de la clase, el cual se inicializa automáticamente cuando se crea un objeto de la clase. Los constructores pueden recibir argumentos pero no pueden devolver valores.

10.2.1 Constructores sin parámetros

En algunos casos es necesario crear varios objetos que se inicialicen siempre con valores predeterminados, es decir, cada vez que se cree un nuevo objeto este se inicializaría con los mismos valores.

```
Class Alumno
    Private  nombre_alumno As String
    Private asignatura As String
    Private curso As Integer
    Public Sub Alumno ()
       Nombre_alumno=""
       asignatura=""
       curso=0
    End Sub
    Public Function obtenerNombre(nombre As String) As String
       return nombre
    End Function
    Public Function obtenerCurso(curso As Integer) As Integer
       return curso
    End Function
End Class
```

10.2.2 Constructores con parámetros

Existen casos en que es necesario crear varios objetos que se inicialicen con diferentes valores, como se puede apreciar, en el ejemplo anterior cada vez que se crea un nuevo objeto este se inicializaría con los mismos valores, la solución es crear un constructor que tenga parámetros.

```
Class Alumno
    Private  nombre_alumno As String
    Private asignatura As String
    Private curso As integer
    Public Sub Alumno(nombre As String, asig As String, numero As Integer)
        nombre_alumno=alumno
        asignatura=asig
        curso=numero
    End Sub
    Public Function obtenerNombre(nombre As String) As String
        return nombre
    End Function
    Public Function obtenerCurso(curso As Integer) As Integer
        return curso
    End Function
    …..
End Class
```

10.3 Sobrecarga de constructores

La sobrecarga de constructores se refiere a la creación de métodos constructores con el mismo nombre pero con diferente lista de parámetros en la misma clase. Cuando invoca un constructor sobrecargado, .NET utiliza como apoyo el número de parámetros como guía para determinar la versión del constructor sobrecargado que realmente debe llamar. Además, de la sobrecarga de constructores, también es posible realizar la sobrecarga con otros métodos. De hecho las clases que se implementan en la realidad, la sobrecarga de constructores es la norma y no la excepción.

```
Class Alumno
    Private  nombre_alumno As String
    Private asignatura As String
    Private curso As integer
    Public Sub Alumno(nombre As String, asig As String, numero As Integer)
        nombre_alumno=alumno
        asignatura=asig
        curso=numero
    End Sub
    Public Sub Alumno(nombre As String, asig As String)
        nombre_alumno=alumno
        asignatura=asig
        curso=0
    End Sub
    Public Function obtenerNombre(nombre As String) As String
        return nombre
    End Function
    Public Function obtenerCurso(curso As Integer) As Integer
        return curso
    End Function
    …..
End Class
```

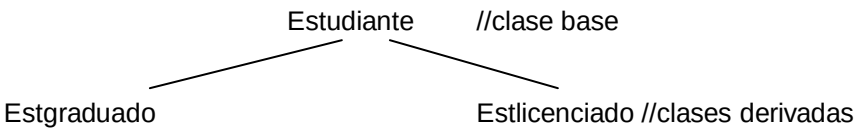
10.4 Herencia y polimorfismo

La herencia es la posibilidad de crear una nueva clase a partir de una clase existente, la clase nueva hereda todos los atributos y comportamientos de una clase existente. En la nueva clase se pueden agregar atributos y comportamientos o supeditar los comportamientos de la superclase a fin de adaptar la clase a las nuevas necesidades.

El polimorfismo permite escribir programas para manejar una amplia variedad de clases interrelacionadas existentes y por especificar.

Al crear una nueva clase (clase derivada), en lugar de escribir variables y métodos totalmente nuevos, el programador puede indicar que la nueva clase puede heredar las variables y los métodos de una superclase (clase base) previamente definida.

Para poder acceder a las variables de la superclase estas previamente deben estar con el modificador de acceso protected o public:



En Visual Basic .NET la palabra reservada **Inherits** permite realizar la herencia de una clase.

```
Class Notas
   Inherits Alumno
    Cuerpo…..
End class
```

En el anterior código se define la subclase **Notas** que hereda (**Inherits**) las propiedades y atributos de la clase **Alumno**.

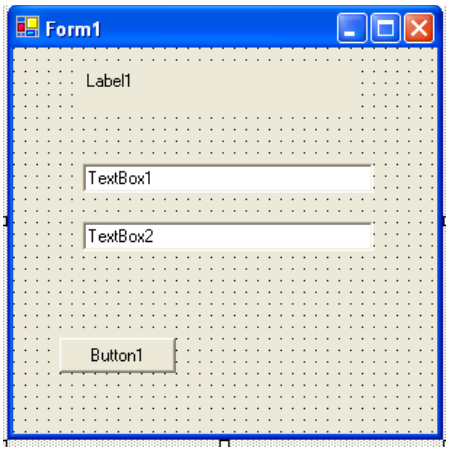
10.5 Ejemplo práctico creación de clases propias

Realizar una aplicación que permita a un usuario capturar sus nombres y apellidos e imprimir en una caja de mensaje lo capturado.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas agregue al formulario: un control Button, dos TextBox y un Label. La figura muestra la interfaz de usuario:

Figura 10.1 Interfaz de usuario ClasesPropias.



- **Establecer las propiedades de los objetos de la interfaz de usuario**

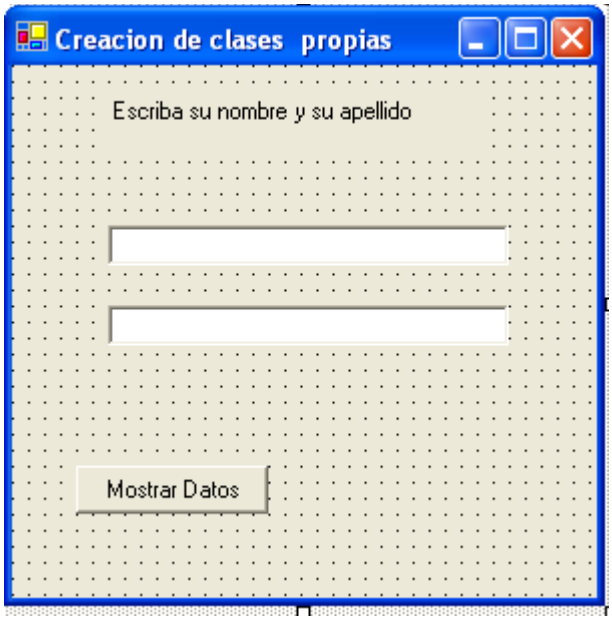
Después de colocar los controles u objetos en el formulario, establezca las siguientes propiedades a los controles:

Tabla 10.1 Propiedades de los controles de la aplicación ClasePropias.

Control	Propiedad	Valor
Button1	Name	botón
	Text	Mostrar Datos
Label1	Name	Texto
	Text	Escriba su nombre y su apellido
TextBox1	Name	txtnombre
	Text	En Blanco
Textbox2	Name	txtapellido
	Text	En Blanco
Form1	Name	formulario
	Text	Creación de clases propias

El formulario se visualizaría como muestra la siguiente figura:

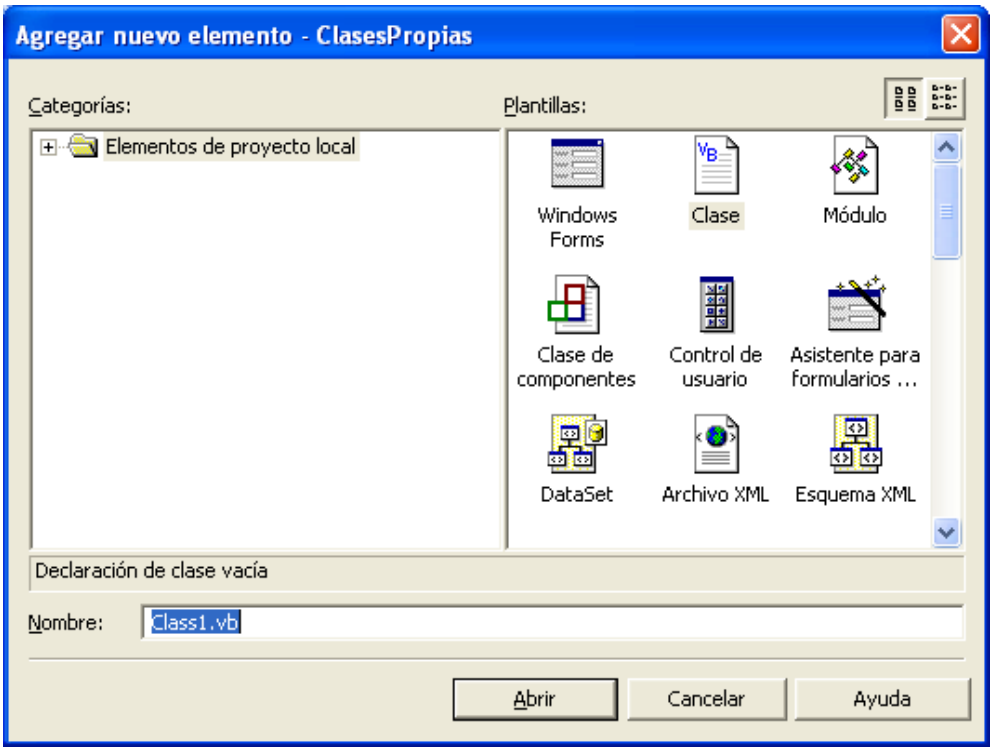
Figura 10.2 Formulario con controles modificados de la aplicación ClasesPropias.



- **Agregar nueva Clase**

Para crear una nueva clase utilice el comando **Agregar clase** del menú **Proyecto,** para visualizar la siguiente figura:

Figura 10.3 Ventana para agregar una clase.



En el cuadro de texto **Nombre** cambie el nombre a la clase por **base**, luego pulse el botón **Abrir** para visualizar un nuevo modulo de clase en el editor de código y en la lista del explorador de soluciones aparece un nuevo archivo con el nombre de la clase.

- **Escribir código**

Entre las instrucciones **Public Class base** y **End Class,** escriba el siguiente código:

```
Private nombre As String
Private apellido As String
```

Se declaran dos variables (**nombre** y **apellido**) para guardar los nombres y los apellidos digitados por el usuario respectivamente. Además se declaran como privadas para que no puedan ser utilizadas fuera de la clase.

```
Public Property nombres() As String
    Get
        Return nombre
    End Get
    Set(ByVal Valor As String)
        nombre = Valor
    End Set
End Property
```

Se define una función **Property** que retornará valores de tipo **Strings** llamada **nombres**, cuando se pulse **Enter** se generará automáticamente el bloque **Get** y **Set.** En el bloque **Get** se retorna por intermedio de la palabra reservada **Return** la información que contenga la variable **nombre.** En el bloque **Set** se recibe como parámetro la

variable **valor** de tipo **String**, la cual recibe la información del objeto **txtnombre** que es enviada por el usuario y dicha información es asignada a la variable **nombre.**

```
Public Property apellidos() As String
    Get
        Return apellido
    End Get
    Set(ByVal Valor As String)
        apellido = Valor
    End Set
End Property
```

Se define una función **Property** que retornará valores de tipo **Strings** llamada **apellidos**, cuando se pulse **Enter** se generará automáticamente el bloque **Get** y **Set.** En el bloque **Get** se retorna por intermedio de la palabra reservada **Return** la información que contenga la variable **apellido.** En el bloque **Set** se recibe como parámetro la variable **valor** de tipo **String** la cual recibe la información del objeto **txtapellido** que es enviada por el usuario y dicha información es asignada a la variable **apellido.**

Seleccione el objeto **botón**, dé doble clic para abrir el editor del procedimiento *boton_`Clicked`* y escriba el siguiente código:

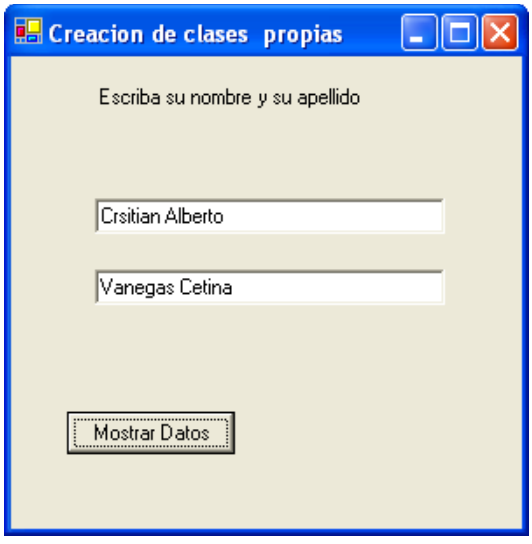
```
Dim persona As  new base
persona.nombres = txtnombre.Text
persona.apellidos = txtapellido.Text
MsgBox ("Sus nombres son: " & persona.nombres & "
          y sus apellidos     son:" & persona.apellidos)
```

En este código primero se crea una instancia de tipo **base** llamada **persona.** Luego se le asigna a las propiedades **nombres** y **apellidos** los valores correspondientes de los objetos **txtnombre** y **txtapellido**. Por último se muestra en una caja de mensajes los nombres y los apellidos que fueron digitados.

- **Ejecutar el proyecto**

Al ejecutar el proyecto en el entorno de desarrollo de visual Basic.NET se visualiza:

Figura 10.4 Ejecución aplicación ClasesPropias.



Al digitar la información requerida en los campos de texto y hacer clic en el botón **Mostrar Datos,** se visualizará la siguiente figura:

Figura 10.5 Ventana de resultados de la aplicación ClasesPropias.

10.6 Heredar una clase base

Para heredar una clase base (padre) como se dijo anteriormente es necesario utilizar la instrucción **Inherits** para incluir la clase definida previamente en una nueva clase. También se pueden agregar métodos y propiedades adicionales a la clase heredada.

Continuando con el mismo ejercicio, agréguele una segunda clase definida dentro del modulo de la clase **base**. Esta nueva clase se llamará **datos** y heredará las propiedades **nombres** y **apellidos** de la clase **base.** En esta clase se le agregarán dos propiedades: **edad** para obtener la edad y **nacimiento** para obtener la ciudad de nacimiento de la persona.

Haga clic en la clase **base** en el explorador de soluciones, sitúese debajo de la instrucción **End Class** y escriba el siguiente código:

```
Public Class base
                    .......
                    ......
End Class
Public Class datos
        Inherits base
    Private edad As Integer
    Private nacimiento As String
    Public Property edades() As Integer
        Get
            Return edad
        End Get
        Set(ByVal Valor As Integer)
            edad = Valor
        End Set
    End Property
    Public Property nacimientos() As String
        Get
            Return nacimiento
        End Get
        Set(ByVal Valor As Integer)
           nacimiento = Valor
        End Set
    End Property
End Class
```

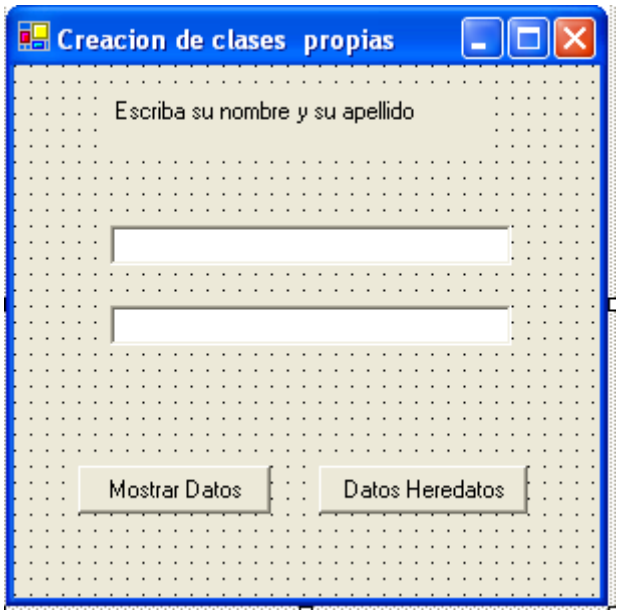
La palabra clave **Inherits** enlaza la clase base a esta nueva clase, incorporado todas sus variables, propiedades y métodos. La clase **datos** tiene dos propiedades: **edades** que permitirá capturar la edad de una persona y **nacimientos** que permitirá capturar la fecha de nacimiento de una persona.

Adicione un nuevo objeto **Button** al formulario actual, en la propiedad **Text** coloque el texto **datos heredados** y en la propiedad **Name botonheredado**. Seleccione dicho objeto, de doble clic para abrir el editor del procedimiento *botonheredado_Clicked* y escriba el siguiente el código:

```
Dim persona As New datos
persona.nombres = TextBox1.Text
persona.apellidos = TextBox2.Text
persona.edades = InputBox ("Digite su edad")
persona.nacimientos = InputBox ("Digite su fecha de nacimiento:")
MsgBox ("Sus nombres son: " & persona.nombres &
        " y sus apellidos son:" & persona.apellidos &
        " Su edad es:" & persona.edades & " años" &
        " y su fecha de nacimiento es: " & persona.nacimientos)
```

La interfaz quedaría como se muestra en la siguiente figura:

Figura 10.6 Formulario de la aplicación ClasesPropias con nuevos controles.



A continuación ejecute el proyecto y escriba en los campos de texto un nombre y un apellido respectivamente y pulse el botón **datos heredados** y capture los datos que se solicitan. Se visualizarán las siguientes figuras:

Figura 10.7 Ejecución aplicación ClasesPropias (captura edad).

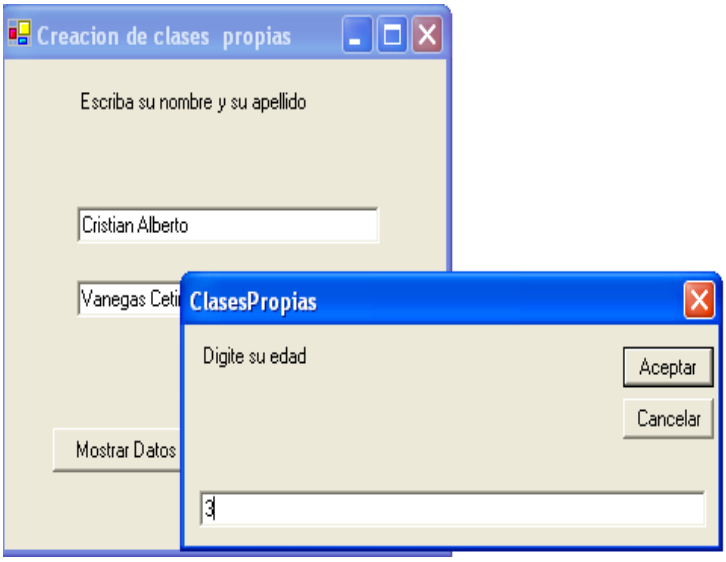


Figura 10.8 Ejecución aplicación ClasesPropias (captura fecha).

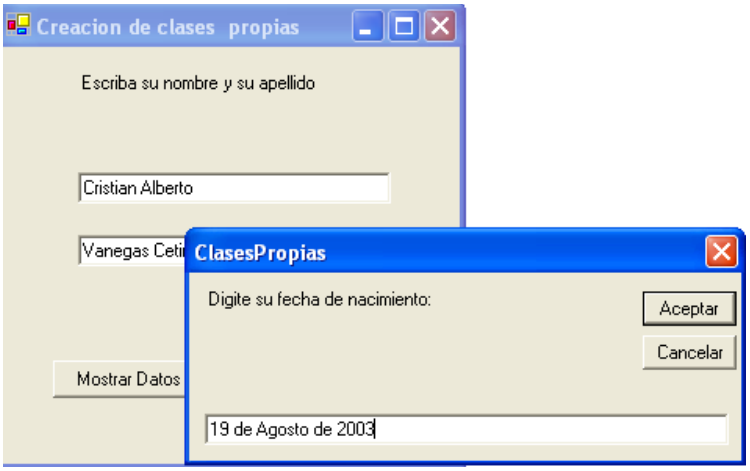
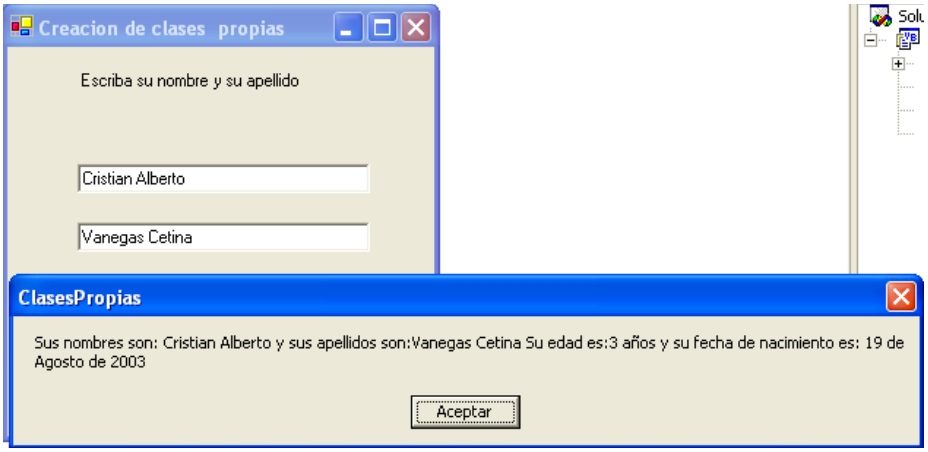


Figura 10.9 Ejecución aplicación ClasesPropias (mostrar resultados).



11. ACCESO A BASES DE DATOS CON .NET

11.1 Que es una base de datos

Una base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos (S.G.B.D. Sistema de Gestión de Bases de Datos). Cada base de datos se compone de una o más tablas que guarda un conjunto de datos. Cada tabla tiene una o más columnas y filas. Las columnas guardan una parte de la información sobre cada elemento que se quiere guardar en la tabla, cada fila de la tabla conforma un registro. Entre las principales características se pueden mencionar:

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoria.
- Respaldo y recuperación.
- Acceso a través de lenguajes de programación estándar.

11.2 Tipos de bases de datos

11.2.1 Relacionales

Son las que más se utilizan. Las bases de datos relacionales son un conjunto de tablas relacionadas entre sí, cada tabla está definida por una serie de campos. Los campos forman las columnas de las tablas; definen el tipo y la variedad de sus datos. Las filas de datos se denominan registros (tuplas), cada tipo definido en un registro se le denomina atributo. Las tablas pertenecientes a una base de datos pueden relacionarse entre sí utilizando campos clave comunes entre las tablas.

11.2.2 Enfoque orientado a objetos

El esquema de una base de datos por objetos está representado por un conjunto de clases que definen las características y el comportamiento de los objetos que poblarán la base de datos. Con una base de datos orientada a objetos, los objetos memorizados en la base de datos contienen tanto los datos como las operaciones posibles con tales datos. En cierto sentido, se podrá pensar en los objetos como en datos a los que se les ha puesto una inyección de inteligencia que les permite saber cómo comportarse, sin tener que apoyarse en aplicaciones externas.

11.3 Lenguaje de consulta estructurado

Es un lenguaje de base de datos normalizado, utilizado por los diferentes manejadores de bases de datos para realizar determinadas operaciones sobre los datos o sobre la estructura de los mismos.

El lenguaje SQL (Structure Query Languaje) está compuesto por comandos, cláusulas, operadores y funciones de agregado. Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos.

11.3.1 Comandos

Existen dos tipos de comandos SQL:

- DLL que permiten crear y definir nuevas bases de datos, campos e índices.
- DML que permiten generar consultas para ordenar, filtrar y extraer datos de la base de datos.

Tabla 11.1 Comandos DLL y DML de SQL.

Comandos DLL	
Comando	**Descripción**
CREATE	Utilizado para crear nuevas tablas, campos e índices
DROP	Empleado para eliminar tablas e índices.
ALTER	Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos.
Comandos DML	
Comando	**Descripción**
SELECT	Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado.
INSERT	Utilizado para cargar lotes de datos en la base de datos en una única operación.
UPDATE	Utilizado para modificar los valores de los campos y registros especificados.
DELETE	Utilizado para eliminar registros de una tabla de una base de datos.

11.3.2 Cláusulas

Las cláusulas son condiciones de modificación utilizadas para definir los datos que desea seleccionar o manipular.

Tabla 11.2 Clausulas SQL.

Cláusula	**Descripción**
FROM	Utilizada para especificar la tabla de la cual se van a seleccionar los registros.
WHERE	Utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar.
GROUP BY	Utilizada para separar los registros seleccionados en grupos específicos.
HAVING	Utilizada para expresar la condición que debe satisfacer cada grupo.
ORDER BY	Utilizada para ordenar los registros seleccionados de acuerdo con un orden específico.

11.3.3 Operadores lógicos

Los operadores lógicos comprueban la veracidad de alguna condición. Éstos, como los operadores de comparación, devuelven el tipo de datos **Boolean** con el valor **TRUE** o **FALSE**

Tabla 11.3 Operadores lógicos SQL.

Operador	Uso
AND	Es el "y" lógico. Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR	Es el "o" lógico. Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
NOT	Negación lógica. Devuelve el valor contrario de la expresión.
BETWEEN	Utilizado para especificar un intervalo de valores.
LIKE	Utilizado en la comparación de un patrón.
IN	Utilizado para especificar registros de una base de datos.
ALL	Devuelve True si el conjunto de comparaciones en verdad.

11.3.4 Operadores de comparación

Los operadores de comparación comprueban si dos expresiones son iguales. Se pueden utilizar en todas las expresiones excepto en las de los tipos de datos **text**, **ntext** o **image**.

Tabla 11.4 Operadores de comparación SQL.

Operador	Uso
<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor o igual que
>=	Mayor o igual que
=	Igual que

11.3.5 Funciones de agregado

Las funciones de agregado realizan un cálculo sobre un conjunto de valores y devuelven un solo valor. Si exceptuamos la función COUNT, todas las funciones de agregado ignoran los valores NULL. Las funciones de agregado se suelen utilizar con la cláusula **GROUP BY** de la instrucción **SELECT**.

Tabla 11.5 Funciones de agregado SQL.

Función	Descripción
AVG	Utilizada para calcular el promedio de los valores de un campo determinado.
COUNT	Utilizada para devolver el número de registros de la selección.
SUM	Utilizada para devolver la suma de todos los valores de un campo determinado.
MAX	Utilizada para devolver el valor más alto de un campo especificado.
MIN	Utilizada para devolver el valor más bajo de un campo especificado.

11.4 Sentencias SQL básicas

Se describirá muy brevemente algunas de las sentencias SQL para la manipulación de los datos de una tabla llamada **usuarios**. Para ello utilizaremos la siguiente estructura:

Tabla 11.6 Estructura de los campos de la tabla usuarios.

Campo	Tipo de Dato	Longitud
identificacion	Texto	15
Nombres	Texto	20
apellidos	Texto	20
dirección	Texto	25
Teléfono	Texto	20
Ciudad_nac	Texto	20
Fecha_nac	dateTime	

Y que contiene la siguiente información:

Tabla 11.7 Información de la tabla usuarios.

Identificación	Nombres	Apellidos	Dirección	Teléfono	Ciudad_nac	Fecha_nac
100	Carlos	Romero	Cra 7 # 20-10	4152584	Bogota	01/02/1980
101	María	Castro	Calle 25 # 25-10	3692581	Cali	15/03/1984
112	José	Peláez	Av. 35 # 32-45	1234567	Medellín	20/05/1960
114	Cristian	Vanegas	Cra 7 # 29-58	9874561	Manizales	31/08/1974
116	Rosa	Cetina	Calle 17 # 21-14	3571596	Buga	15/12/1985
118	Andrés	Vanegas	Tranvs 48 # 22-10	8527419	Bogota	10/04/1978
130	Angélica	Morales	Cra 68 # 21-11	6549518	Medellín	20/06/1981
150	Johana	Duarte	Cra 2 # 45-38	9637534	Bogota	12/06/1988
170	Mario	Vargas	Calle 1 # 99-18	6598743	Medellín	28/08/1980

11.4.1 Sentencia SELECT

La sentencia **SQL** que más se utiliza es la instrucción de selección **SELECT**. Como su nombre lo indica es una instrucción que permite seleccionar información de una tabla. Su formato es:

```
SELECT campos_tabla FROM nombre_tabla
```

A continuación se realizarán algunos ejemplos:

a) Para visualizar toda la información que contiene la tabla **usuarios** se puede incluir con la instrucción SELECT el carácter '*' o cada uno de los campos de la tabla.

```
SELECT * FROM usuarios
        O
SELECT identificación, nombres,…….. FROM usuarios
```

b) Para visualizar solamente la identificación del usuario

```
SELECT identificacion FROM usuarios
```

c) Si se desea obtener los registros cuya identificación sea menor o igual que 116, se debe utilizar la cláusula WHERE que especifica las condiciones que deben reunir los registros que se van a seleccionar.

```
SELECT * FROM usuarios WHERE identificación<='116'
```

d) Si se desea obtener los registros cuyos nombres sean Andrés o Cristian, se debe utilizar el operador IN que especifica registros de una tabla.

```
SELECT nombres FROM usuarios WHERE nombres IN ('Andres','Cristian')
```

O se puede utilizar el operador OR

```
SELECT * FROM usuarios WHERE nombres='Andrés' OR nombres='Cristian'
```

e) Si se desea obtener los registros cuya identificación sea menor de '130' y la ciudad sea 'Bogota', se debe utilizar el operador AND.

```
SELECT * FROM usuarios WHERE identificación<='130' AND ciudad='Bogota'
```

f) Si se desea obtener los registros cuyos nombres empiecen por la letra 'C', se debe utilizar el operador LIKE que utiliza los patrones '%' (todos) y '_' (carácter).

```
SELECT * FROM usuarios WHERE nombres LIKE 'C%'
```

g) Si se desea obtener los registros cuyos nombres contenga la letra 'i'.

```
SELECT * FROM usuarios WHERE nombres LIKE '%i%'
```

h) Si se desea obtener los registros donde la segunda letra del nombre sea una 'o'.

```
SELECT * FROM usuarios WHERE nombres LIKE _o%'
```

i) Si se desea obtener los registros cuya identificación este entre el intervalo 116 y 140, se debe utilizar la cláusula BETWEEN, que sirve para especificar un intervalo de valores.

```
SELECT * FROM usuarios WHERE identificación BETWEEN ’116’ AND ‘140’
```

11.4.2 Sentencia INSERT

La sentencia SQL de inserción de datos INSERT permite insertar información en una tabla. Su formato es:

```
INSERT INTO nombre_tabla (campo1, campo2,….) VALUES (valor1, valor2,….)
```

Para insertar un nuevo registro a la tabla **usuarios** se debería realizar la siguiente sentencia:

```
INSERT INTO usuarios (identificación, nombres, apellidos, dirección, teléfono, ciudad_nac,fecha_nac)
VALUES (‘160’,’Carmen’,’Bolivar’,’Calle 100 # 115-55’,’2014201’,’Barranquilla’,’18/11/1692’)
```

11.4.3 Sentencia DELETE

La sentencia SQL de eliminación de datos DELETE permite borrar todos o un grupo especifico de registros de una tabla. Su formato es:

```
DELETE FROM nombre_tabla
```

A continuación se realizarán algunos ejemplos:

a) Para eliminar todos los registros de la tabla usuarios.

```
DELETE FROM usuarios
```

b) Para eliminar solamente los registros cuya identificación sea mayor que ‘150’.

```
DELETE FROM usuarios WHERE identificación >’150’
```

c) Para eliminar los registros diferentes del nombre “Cristian”

```
DELETE FROM usuarios WHERE nombres NOT IN (‘Cristian’)
```

11.4.4 Sentencia ALTER

La sentencia SQL ALTER permite insertar un nuevo campo en una tabla. Su formato es:

```
ALTER TABLE  nombre_tabla ADD nombre_campo tipo_de_dato()
```

Para insertar un nuevo campo a la tabla **usuarios** llamado de tipo numérico se debería realizar la siguiente sentencia:

```
ALTER TABLE usuarios ADD creditonumeric(18,0)
```

11.4.5 Sentencia UPDATE

La sentencia SQL de actualización UPDATE permite actualizar un campo de una tabla. Su formato es:

```
UPDATE nombre_tabla SET nombre_campo=criterio
```

A continuación realizaremos algunos ejemplos:

1. Para actualizar el campo crédito con un valor de 100000 en la tabla usuarios.

```
UPDATE usuarios SET credito=100000
```

2. Para actualizar el campo credito en 200000 para los registros cuyo nombre empiecen por 'A'.

```
UPDATE usuarios SET  credito=credito +200000 WHERE nombres LIKE 'A%'
```

3. Para actualizar el campo credito en 50000 para los registros cuya ciudad sea igual a 'Bogota'.

```
UPDATE usuarios SET  credito=credito+50000 WHERE ciudad='Bogota'
```

11.5 Ado.NET

El acceso a los orígenes de datos en ADO.NET se rige a través de proveedores administrados. Respecto a las funciones, un proveedor administrado es muy parecido a un proveedor OLE DB, aunque con dos diferencias importantes. Primero, funcionan en el entorno .NET y recuperan y exponen datos a través de clases .NET, como DataReader y DataTable. Segundo, la arquitectura resulta más sencilla, puesto que se ha optimizado para .NET.

Se deben utilizar clases SQL para obtener acceso a las tablas de SQL Server, ya que se dirigen directamente a la API interna del servidor de base de datos, omitiendo el nivel intermedio representado por el proveedor OLE DB. Las clases de ADO constituyen una interfaz .NET además, de los proveedores OLE DB se utilizan el puente COM Interop para realizar esta tarea.

ADO.NET ofrece dos objetos para manipular los datos extraídos de un origen de datos. Se trata de los objetos **DataSet** y **DataReader**. El primero es una memoria caché de los registros que se pueden visitar en cualquier dirección y modificar como se desee. El segundo es un objeto optimizado para desplazarse por registros de sólo lectura y sólo hacia adelante. Observe que el aspecto de **DataSet** es similar a un cursor estático, mientras que el objeto **DataReader** es el equivalente en .NET al cursor de sólo lectura de ADO.

Tenga en cuenta que ADO.NET unifica la interfaz de programación para las clases contenedoras de datos. Independientemente del tipo de aplicación que desee desarrollar (para formularios de Windows, Web o Servicios Web), maneje los datos a través del mismo juego de clases. Tanto si el origen de datos en el servidor es una base

de datos SQL Server, un proveedor OLE DB, un archivo XML o una matriz, desplácese por el contenido y contrólelo a través de los mismos métodos y propiedades.

Si desea utilizar ADO en .NET, tendrá que enfrentarse a algunos efectos secundarios, como el código adicional que necesitará para utilizar los **RecordSets** a partir de controles enlazados a datos.

11.5.1 DataSet, DataTable y RecordSet

En ADO.NET, toda la funcionalidad del **RecordSet** de ADO se ha dividido en objetos más simples: uno de ellos es **DataReader**. El objeto **DataReader** imita el comportamiento de un cursor rápido, de sólo lectura y que avanza.

El objeto **DataTable** es un objeto simple que representa un origen de datos. Se puede crear un **DataTable** de forma manual, o bien, se puede rellenar automáticamente con comandos **DataSet**. **DataTable** desconoce el origen de los datos que contiene. Permite la manipulación de los datos en la memoria y realiza operaciones como la exploración, ordenación, edición, aplicación de filtros, creación de vistas, etc.

El objeto **DataSet** no tiene un equivalente en ADO. Se trata de una clase contenedora de datos y es el objeto clave para realizar la abstracción de datos de ADO.NET. **DataSet** agrupa uno o varios objetos **DataTable**. **DataTable** expone el contenido a través de recopilaciones genéricas, como filas y columnas. Al intentar la lectura desde una tabla de datos, puede pasar por dos capas distintas de objetos: **DataTableMapping** y **DataView**.

El objeto **DataTableMapping** contiene la descripción de una asignación entre columnas de datos en un origen de datos y un objeto **DataTable**. El objeto **DataSetCommand** utiliza esta clase al rellenar un **DataSet**. Mantiene el vínculo entre columnas abstractas en el conjunto de datos y las físicas en el origen de datos.

Una vista de la tabla se procesa a través del objeto **DataView**. Representa una vista personalizada de **DataTable** y se puede enlazar a controles especializados, como el **DataGrid** en los formularios de Windows y de Web. Este objeto es el equivalente en memoria a la instrucción **SQL CREATE VIEW**.

Todas las tablas en un objeto **DataSet** se pueden relacionar a través de un campo común. Un objeto **DataRelation** puede controlar esta relación. El modelo de exploración en ADO.NET permite el desplazamiento sencillo desde la fila maestra de una tabla a todas las secundarias.

El objeto **DataRelation** representa el equivalente en memoria a la instrucción JOIN y resulta útil para implementar relaciones primaria/secundaria con columnas que incluyen el mismo tipo de datos. Una vez establecida la relación, no se permite ningún cambio que la interrumpa y se origina una excepción de tiempo de ejecución. Las vistas y las relaciones constituyen dos métodos para implementar los esquemas maestro/detalle. Recuerde que una vista es simplemente una máscara en los registros, mientras que una relación es un vínculo dinámico establecido entre una o varias columnas de dos tablas y que con las relaciones no existe forma de cambiar las condiciones de orden o conjunto.

11.6 Ejemplo práctico bases de datos

Hacer una aplicación que permita realizar la conexión a una base de datos realizada en Access.

Para implementar la aplicación se deben realizar los siguientes pasos:

1. Crear una base de datos en Access llamada **visualnet** y guárdela dentro de la carpeta que desee. Dentro de la base de datos crear una tabla llamada **Clientes** con la siguiente estructura:

Tabla 11.8 Estructura tabla Clientes.

Nombre del campo	Tipo de dato	longitud
Identificación	Texto	13
Nombres	Texto	25
Apellidos	Texto	25
Dirección	Texto	25
Teléfono	Texto	15
Ciudad	Texto	20
Crédito	numérico	

La tabla con 6 registros quedara de la siguiente forma:

Figura 11.1 Registros de la tabla clientes.

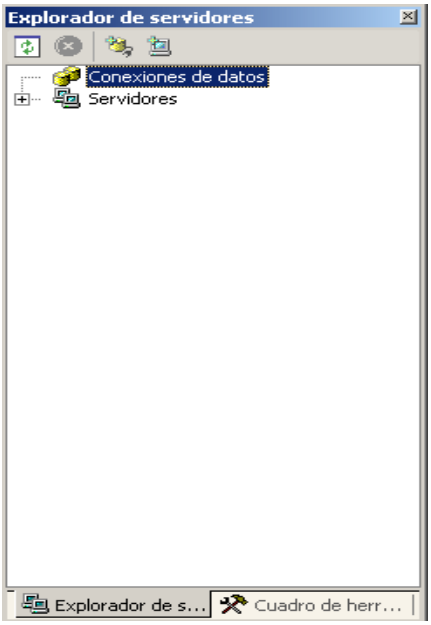
clientes

identificacic	nombres	apellidos	direccion	telefono	ciudad	credito
1	carlos alberto	vanegas	cra 24 # 24 -24	123456	bogota	1000000
2	rosa maria	cetina	cra 25 # 24 -24	654789	cali	200000
3	cristian alberto	vanegas cetina	cra 26 # 24 -24	987456	medellin	300000
4	angelita	vanegas rodrig	cra 27 # 24 -24	321456	bogota	500000
5	camilo	vanegas rodrig	cra 28 # 24 -24	325814	cali	1400000
						0

2. Hacer la conexión a la base de datos

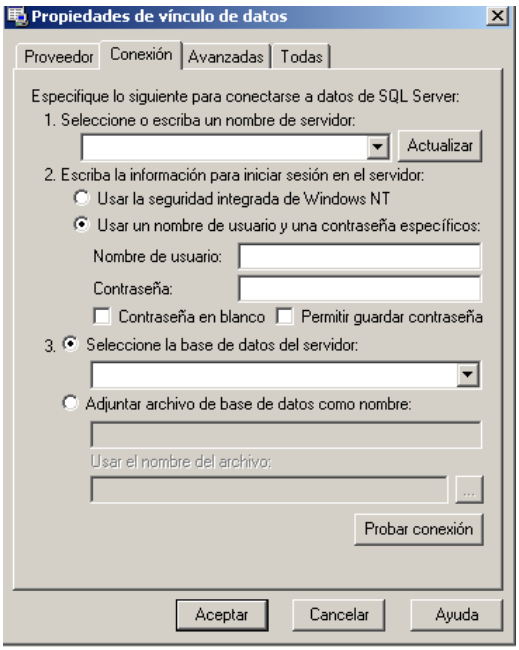
Para realizar una conexión a una base de datos existente seleccione **explorador de servidores** del menú **Ver**, donde se visualizará el siguiente cuadro de diálogo:

Figura 11.2 Ventana explorador de soluciones.



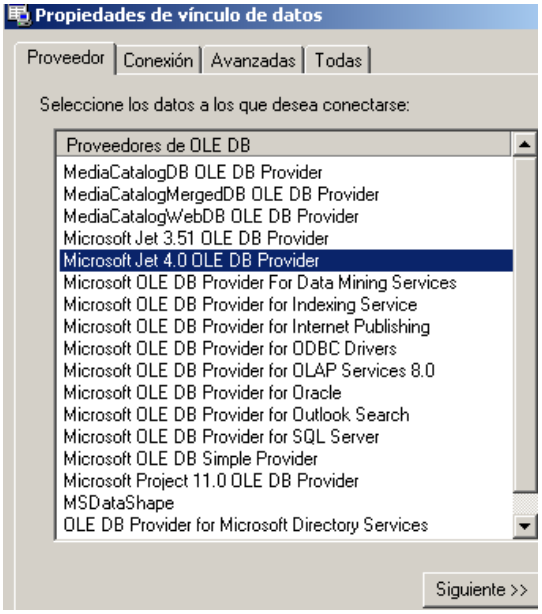
Seleccione el icono **conectar con bases de datos**  para iniciar la conexión a la base de datos y visualizar el cuadro de diálogo **Propiedades de vínculos de datos,** como se muestra en la siguiente figura:

Figura 11.3 Ventana propiedades de vínculos de datos.



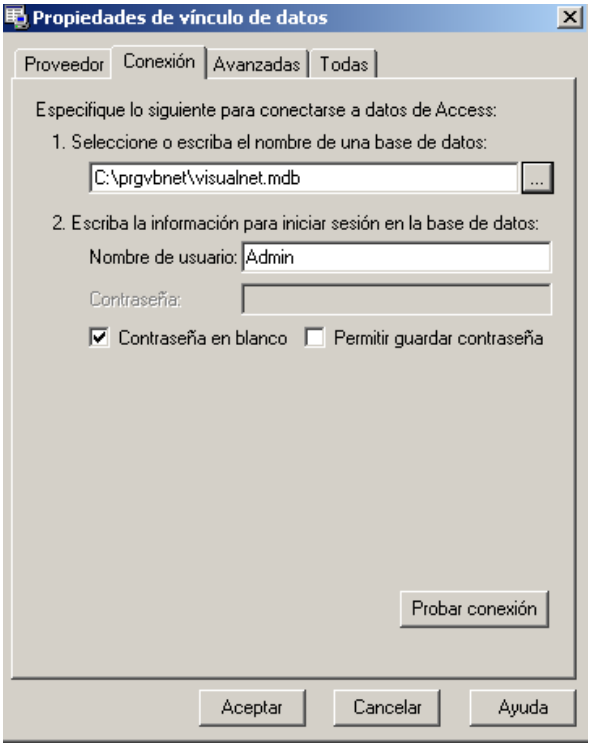
Pulse sobre la pestaña llamada **Proveedor** para seleccionar el proveedor de la base de datos. Para este caso escoja **Microsoft jet 4.0 OLE DB Provider** y pulse el botón **Siguiente >>.** Se visualizará la figura 11.4:

Figura 11.4 Ventana propiedades de vínculos de datos (proveedor).



Al pulsar el botón **Siguiente>>,** se visualizará la pestaña **Conexión** del cuadro de diálogo **propiedades de vínculo de datos,** allí en el paso 1, seleccione la base de datos que previamente se había creado, como se muestra en la siguiente figura:

Figura 11.5 Ventana propiedades de vínculos de datos (conexión).

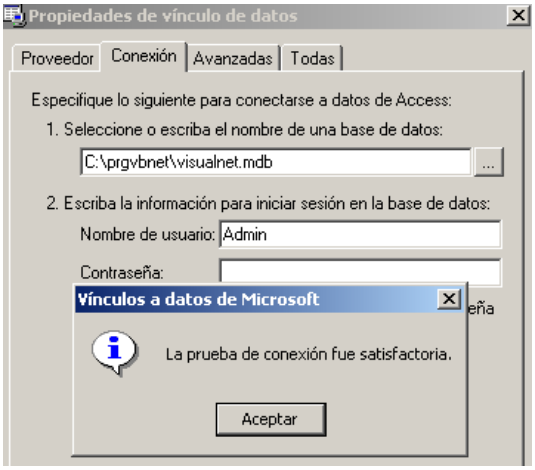


Si desea crear una contraseña a la base de datos desactive el cuadro de verificación **contraseña en blanco** y automáticamente el campo **contraseña** se

habilitará para que el usuario digite la contraseña que desee. El paso 2 es opcional.

En este momento se puede verificar la conexión pulsando el botón **Probar conexión**. Si la conexión ha sido exitosa se mostrará la siguiente figura:

Figura 11.6 Ventana prueba de conexión.

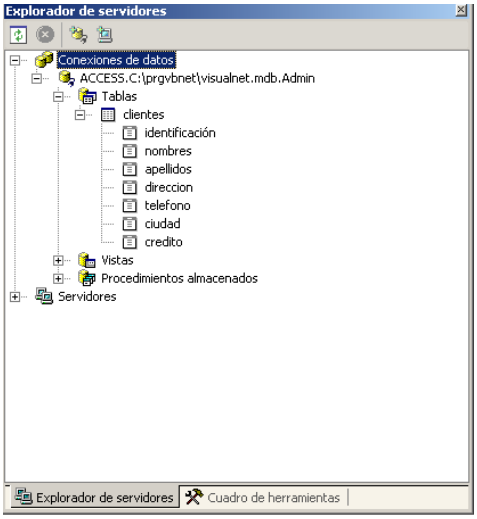


En caso de que la conexión no fuese exitosa, se debe iniciar nuevamente con cada uno de los pasos realizados hasta el momento para resolver el problema. Uno de las posibles fallas en la conexión es que la base de datos este abierta en el momento de la conexión proceda a cerrarla y realice nuevamente la conexión.

Para continuar pulse el botón **Aceptar** de la prueba de conexión, luego pulse el botón **Aceptar** del cuadro de diálogo **Propiedades de vínculo de datos** para terminar la conexión.

En este momento se puede pulsar el signo (+) a la izquierda de la leyenda **conexiones de datos** del cuadro de diálogo **Explorador de servidores** para visualizar la conexión a la base de datos **visualnet**. Cada vez que pulse el signo (+) se podrá apreciar la estructura de la base de datos, como se muestra en la siguiente figura:

Figura 11.7 Ventana explorador de servidores.

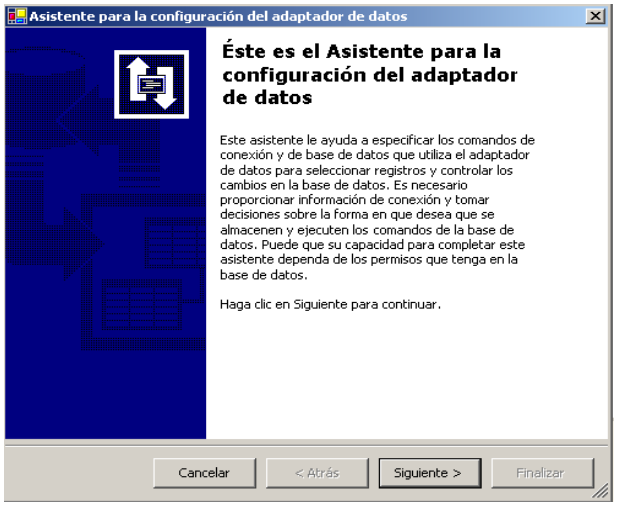


3. Seleccionar registros de un origen de datos

Para consultar los datos que se quieren mostrar de una bases de datos, se debe buscar la pestaña **Datos** del cuadro de herramientas y seleccionar el objeto **OleDbDataAdapter** que sirve para manipular datos de una base de datos de Access. . Es de aclarar que de acuerdo a las base de datos se debe escoger el adaptador, como por ejemplo, si se está trabajando con **SqlServer** el adaptador que se debería escoger seria **SqlDataAdapter**.

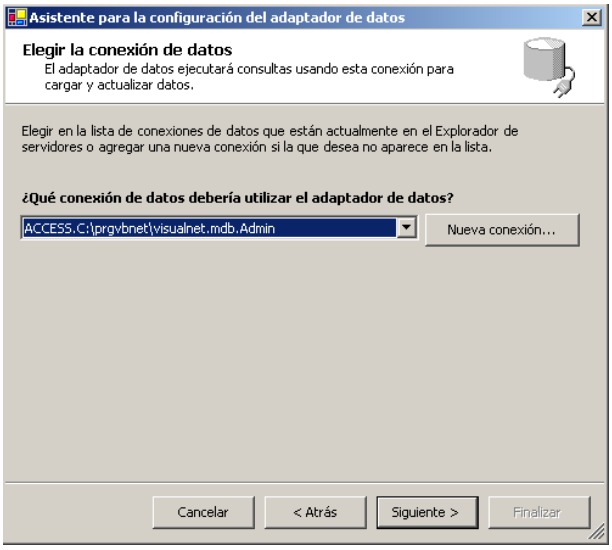
Dé doble clic sobre el objeto **OleDbDataAdapter** para visualizar el **Asistente para la configuración del adaptador de datos.**

Figura 11.8 Asistente para la configuración del adaptador de datos.



Este asistente servirá para especificar los comandos de conexión y la base de datos que el adaptador utilizará para seleccionar los registros y controlar los cambios en la base de datos. Haga clic en el botón **Siguiente>** para visualizar la ventana que sirve para elegir la conexión de datos, como se puede apreciar, en la figura 11.9.

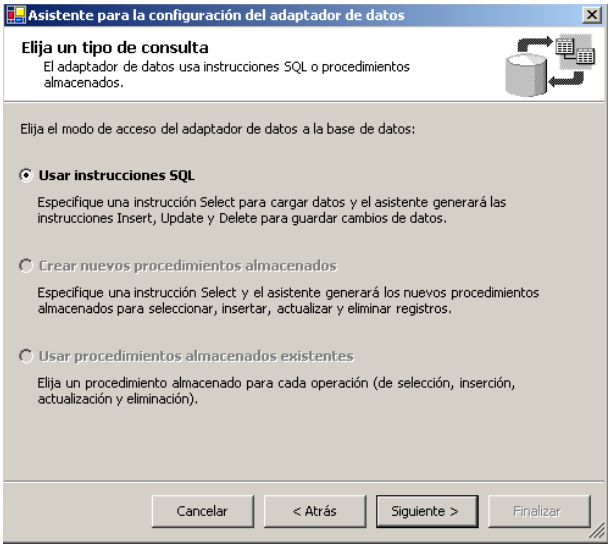
Figura 11.9 Asistente para elegir conexión de datos.



En este cuadro de diálogo del asistente se seleccionara la conexión de datos para cargar y actualizar los datos. Para continuar con el ejemplo seleccione la única conexión que hasta el momento se ha realizado como lo es la conexión: **ACCESS. C:\prgvbnet\visualnet.mdb.Admin.**

Haga nuevamente clic sobre el botón **Siguiente>,** para elegir un tipo de consulta SQL. Se visualizará la siguiente figura:

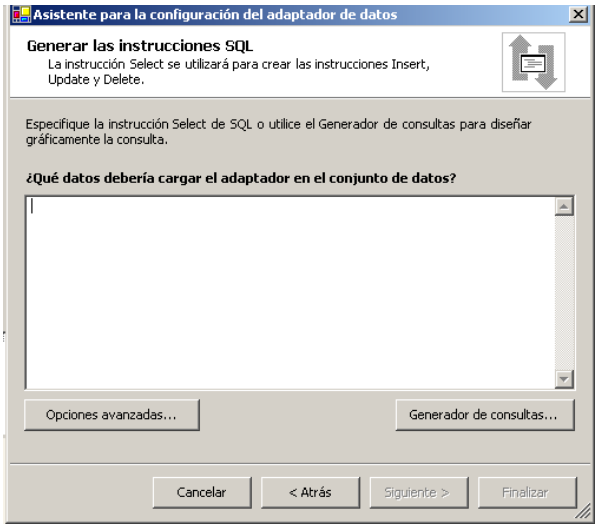
Figura 11.10 Asistente para elegir tipo de consulta.



En este cuadro de diálogo del asistente se utiliza la única opción activa como lo es **Usar instrucciones SQL**, que permitirá especificar la instrucción **Select** para cargar los datos.

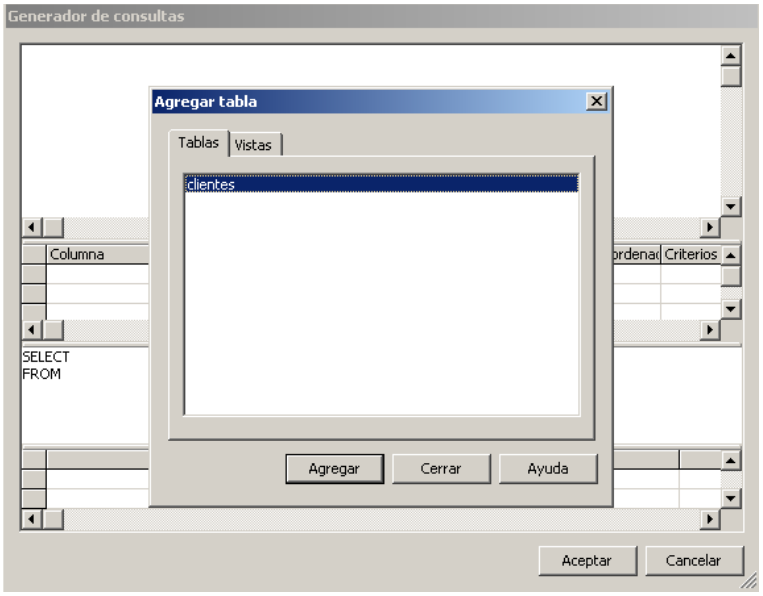
Nuevamente dé clic sobre el botón **Siguiente>,** lo cual permitirá visualizar el cuadro de diálogo del asistente para generar las instrucciones SQL. Se visualizará la siguiente figura:

Figura 11.11 Asistente para generar las instrucciones SQL.



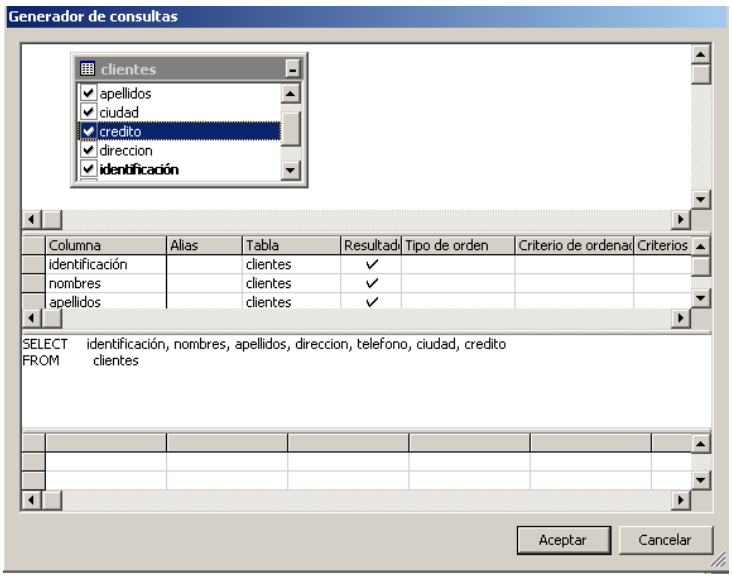
Este cuadro de diálogo permite generar las instrucciones SQL para mostrar los datos deseados. Existen dos formas de realizar las instrucciones SQL: la primera es escribir cada sentencia SQL en el cuadro de texto que aparece en el asistente; la segunda es escoger el botón **Generador de Consultas….** Escoja la segunda opción, donde se visualizará la siguiente figura:

Figura 11.12 Asistente para agregar tabla.



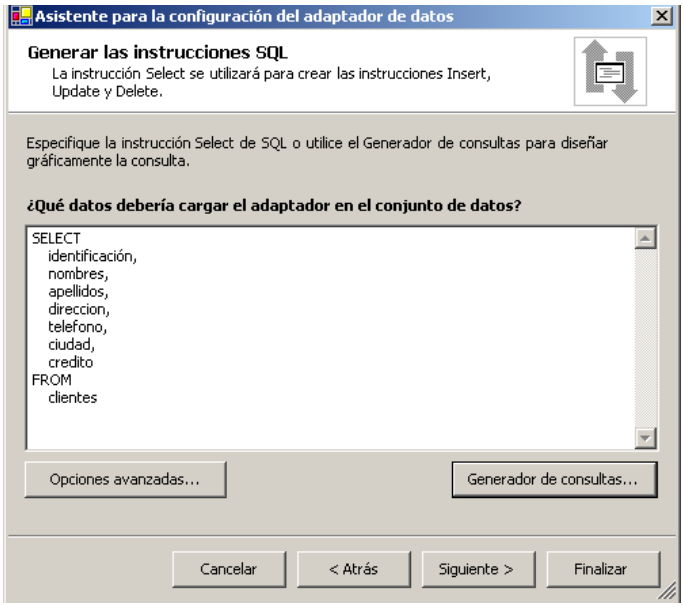
Aquí se pueden agregar las diferentes tablas o vistas que contiene los datos que se desean cargar. Se debe resaltar la tabla que se desea utilizar y dár clic sobre el botón **Agregar** para incluirla en la instrucción SQL. Al finalizar se debe dár clic sobre el botón **Cerrar**. Para el ejemplo, seleccione la tabla **clientes**, agréguela y cierre el cuadro de diálogo **Agregar Tabla**. Se visualizará la siguiente figura:

Figura 11.13 Generador de consultas.



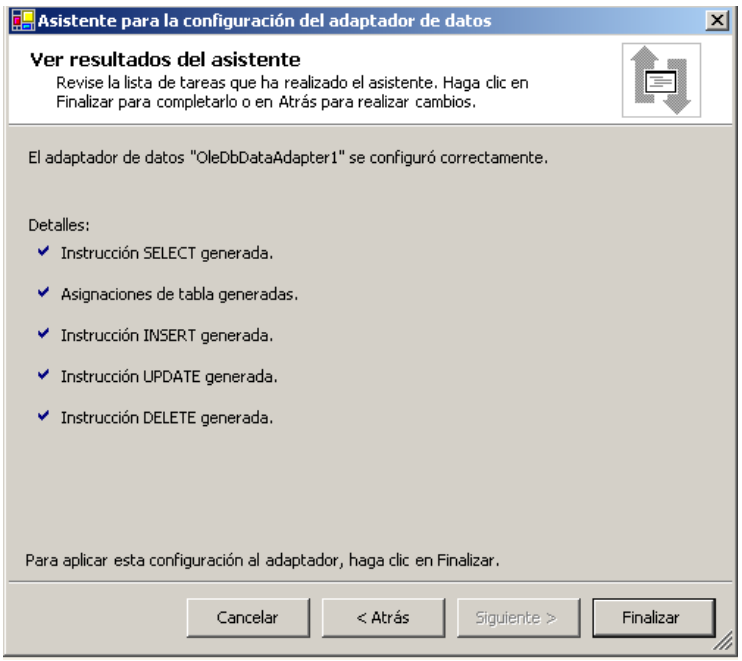
En la tabla puede seleccionar cada uno de los campos que se quieren cargar. A medida que incluya campos se irá generando la instrucción SQL. Al terminar de escoger los campos dá clic sobre el botón **Aceptar** para volver al generador de consultas SQL, como se muestra en la siguiente figura:

Figura 11.14 Generación de instrucciones SQL.



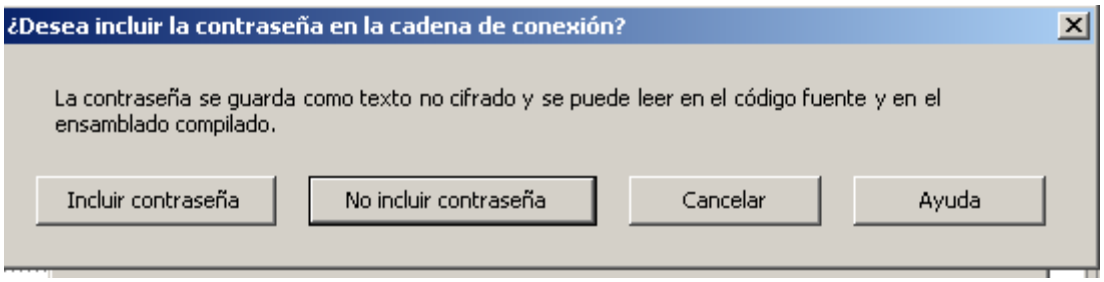
En este cuadro de diálogo se aprecia la instrucción SQL que se ha generado. Nuevamente dé clic sobre el botón **Siguiente>**, para visualizar los resultados que se han generado con el asistente, como se muestra en la siguiente figura:

Figura 11.15 Resultado generados por el asistente.



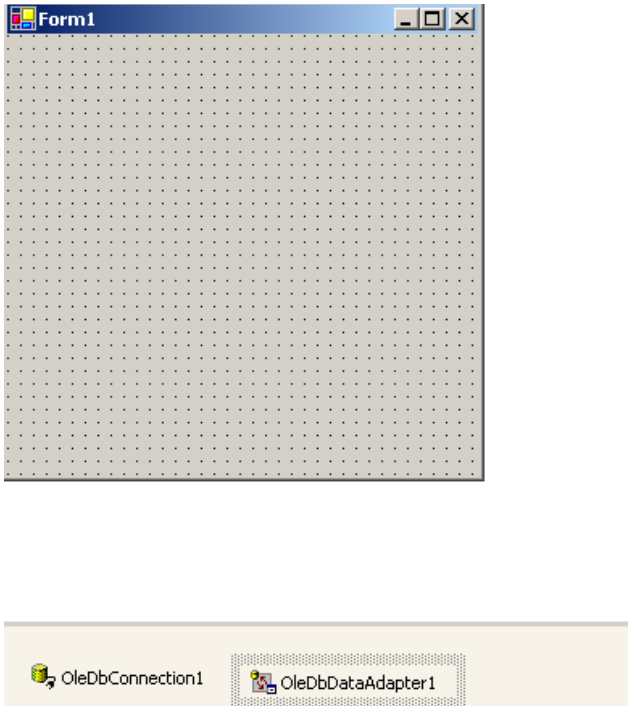
Aquí se visualizan todas las tareas que se han realizado con el asistente. Se debe hacer clic en el botón **Finalizar** para terminar el asistente y se visualizará la siguiente figura:

Figura 11.16 Ventana para contraseña en la cadena de conexión.



En este cuadro de diálogo el asistente pregunta si se desea incluir una contraseña o no. Pulse el botón **No incluir contraseña.** A la aplicación se le agregará debajo del formulario los objetos: **OleDbDataAdapter** y **OleDbConnection**. La figura que se visualizará será la siguiente:

Figura 11.17 Formulario con los objetos OleDbconecction1 y Oledbdataadapter1.

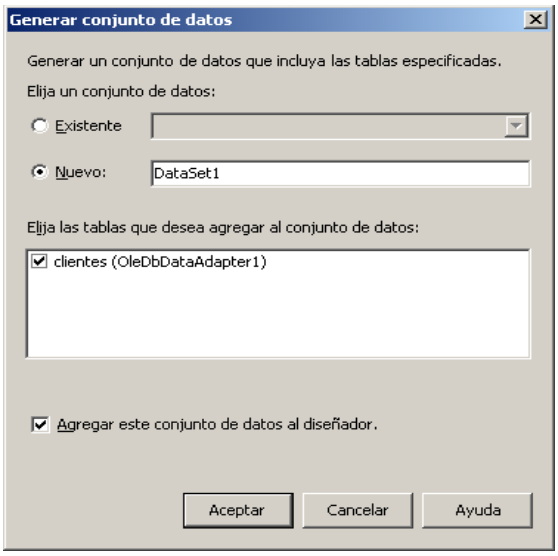


4. Seleccionar el conjunto de datos

El siguiente paso es crear un objeto que represente los datos que se quieren utilizar en la aplicación. Este objeto es llamado **DataSet** que representa los datos proporcionados por la conexión de datos y que serán extraídos por el objeto adaptador de datos. Haga clic

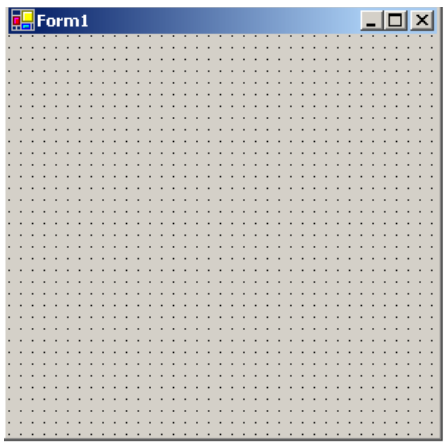
sobre el formulario para asegurarse de que este que de activo, seleccione la opción **Generar conjunto de datos** del menú **Datos**. Se visualizará la siguiente figura:

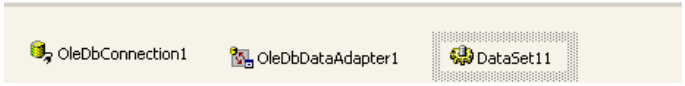
Figura 11.18 Generador de conjunto de datos.



Dé clic sobre el botón **Aceptar** para agregarlo a la aplicación como se muestra en la siguiente figura:

Figura 11.19 Formulario con el control Dataset1 (conjunto de datos).

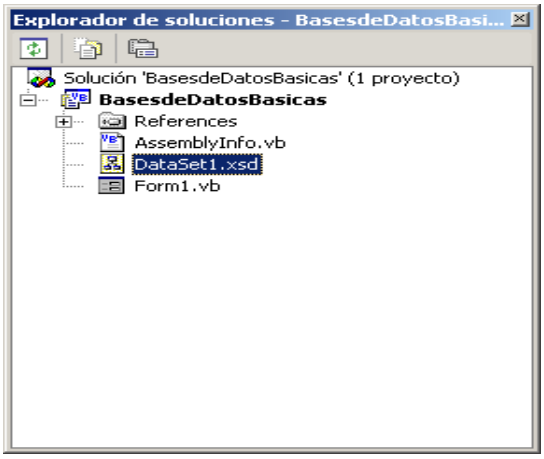




Además de los tres objetos que se han creado, también se agrega un archivo llamado **DataSet!.xsd** al explorador de soluciones, que representa el esquema de la base de datos. Este esquema es un archivo XML que describe las tablas, campos, tipos de

datos del conjunto de datos. La figura 11.20, muestra como quedaría el **explorador de soluciones** de la aplicación:

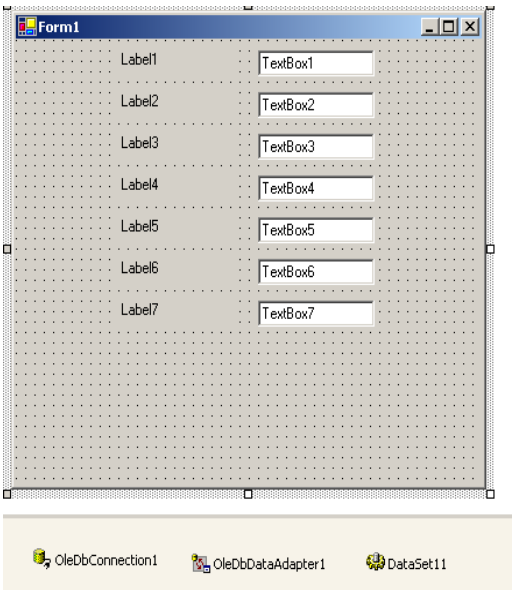
Figura 11.20 Explorador de soluciones y el objeto Dataset1 (conjunto de datos).



Después de realizar cada uno de los pasos anteriores, se está para visualizar la información de la base de datos. Se puede mostrar a los usuarios los registros que se desean ver, además se puede crear mecanismos para navegar entre los registros de una o más tablas. Para mostrar la información de una base de datos se pueden utilizar algunos de estos controles: TextBox, ComboBox, ListBox, RadioButton, DataGrid.

Para ver la información de la tabla **clientes** se utilizará el control **TextBox**. Al formulario se le debe agregar 7 **Label** y 7 **TextBox** que son el número de campos que contiene la tabla **clientes**. El formulario quedaría como se muestra en la siguiente figura:

Figura 11.21 Interfaz de usuario para conexión a una base de datos.



- **Establecer las propiedades de los objetos de la interfaz de usuario**

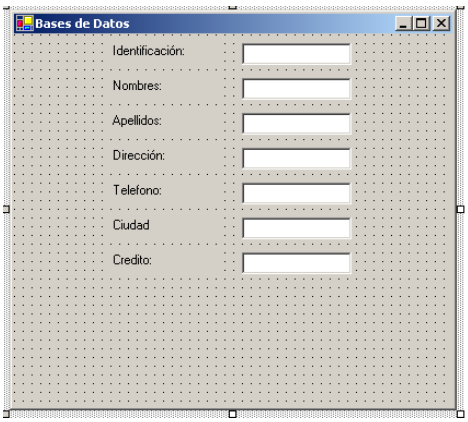
Después de colocar los controles u objetos en el formulario, establezca las siguientes propiedades a los controles:

Tabla 11.9 Propiedades de los controles de la aplicación BasesdeDatosBasicas.

Control	Propiedad	Valor
Label1	Name	lblid
	Text	Identificación:
Label2	Name	lblnombre
	Text	Nombres:
Label3	Name	lblapellido
	Text	Apellidos:
Label4	Name	lbldireccion
	Text	Dirección:
Label5	Name	lbltelefono
	Text	Telefono:
Label6	Name	lblciudad
	Text	Ciudad
Label7	Name	lblCredito
	Text	Credito
TextField1	Name	txtid
	Text	En blanco
TextField2	Name	txtnombres
	Text	En blanco
TextField3	Name	txtapellidos
	Text	En blanco
TextField4	Name	txtdireccion
	Text	En blanco
TextField5	Name	txttelefono
	Text	En blanco
TextField6	Name	txtciudad
	Text	En blanco
TextField7	Name	Txtcredito
	Text	En blanco
Form1	Name	formulario
	Text	Bases de Datos

El formulario se visualizaría como muestra la siguiente figura:

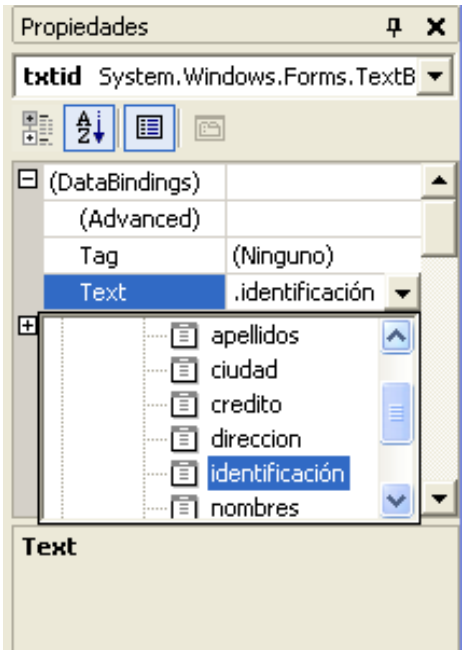
Figura 11.22 Interfaz de usuario final.



- **Escribir código**

Seleccione el objeto **TextBox** llamado **id** y abra la ventana de propiedades. Busque la categoría **DataBindings** haga clic sobre la propiedad **Text** y luego dé clic en la flecha para desplegar la fuente de datos que se va a enlazar con el cuadro de texto, haga clic en los signos (+) para expandir el conjunto de datos **Dataset1** hasta encontrar los campos de la tabla **clientes** y dé clic sobre el campo **identificación** para que en los parámetros de la propiedad **Text** aparezcan los nombres de los campos, la tabla y el conjunto de datos. La figura 11.23, muestra la ventana de propiedades del **TextBox**.

Figura 11.23 Ventana propiedades TextBox (txtid).



Realice la misma operación para cada uno de los **TextBox;** en cada propiedad **Text** quedará el campo específico de acuerdo al formulario.

Seleccione el objeto **formulario**, dé doble clic para abrir el editor del procedimiento *Form1_Load* y escriba el siguiente código:

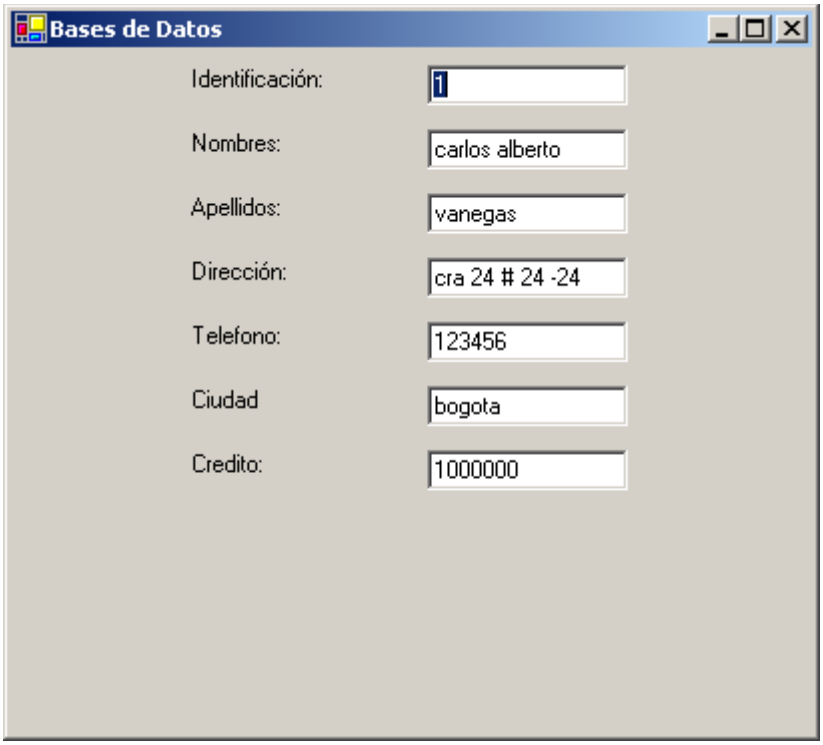
```
DataSet11.Clear()
OleDbDataAdapter1.Fill(DataSet11)
```

En la primera línea se utiliza el método **Clear** para limpiar el conjunto de datos cuando se ejecuta la aplicación. En la segunda línea se utiliza el método **Fill** del objeto **OleDbDataAdapter** para rellenar el conjunto de datos llamado **Dataset11**.

- **Ejecutar el proyecto**

Al ejecutar el proyecto en el entorno de desarrollo de Visual Basic.NET se visualiza:

Figura 11.24 Ejecución aplicación BasesdeDatosBasicas.



11.7 Controles de navegación

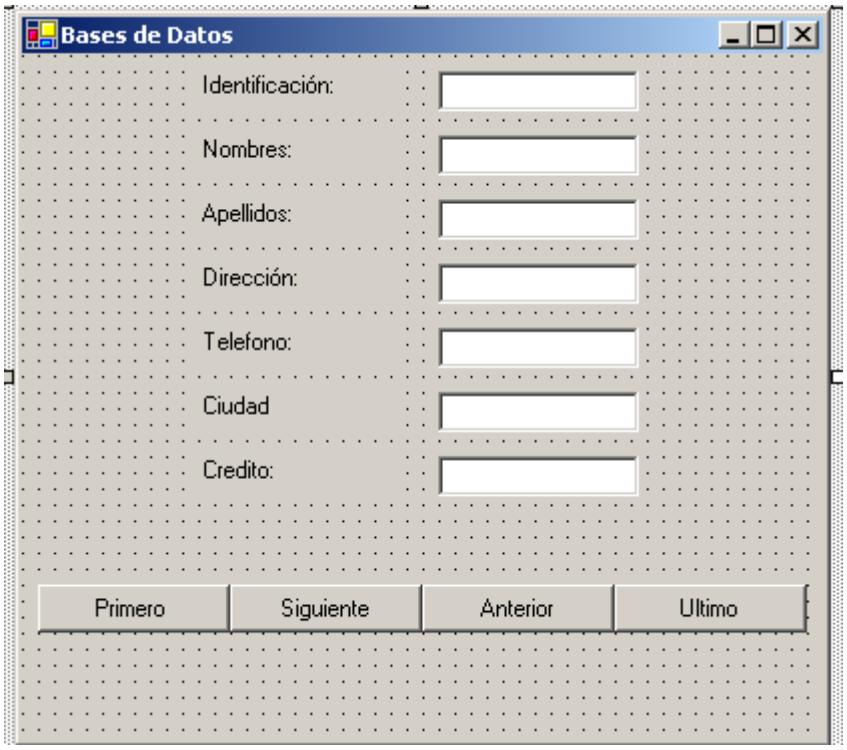
Como se puede apreciar, en la figura 11.24, se visualiza la información solamente del primer registro de la tabla C**lientes**. Para poder moverse por cada uno de los registros en necesario crear una serie de botones que permitan moverse entre los diferentes registros. Para esto se crearán los siguientes botones en el formulario: Primero, Siguiente, Anterior, Último. Al formulario adiciónele 4 botones y configure las siguientes propiedades de acuerdo a la siguiente tabla:

Tabla 11.10 Propiedades de los controles para navegación entre registros.

Control	Propiedad	Valor
Button1	Name	botonprimero
	Text	Primero
Button2	Name	botonsiguiente
	Text	Siguiente
Button3	Name	botonanterior
	Text	Anterior
Button4	Name	botonúltimo
	Text	Último

El formulario quedaría como se visualiza en la siguiente figura:

Figura 11.25 Interfaz de usuario con controles de navegación.



Seleccione el objeto **botonprimero**, dé doble clic para abrir el editor del procedimiento *botonprimero_Click* y escriba el siguiente código:

```
Me.BindingContext(DataSet11, "clientes").Position = 0
```

Se utiliza el objeto **BindingContext** el cual mantiene la posición actual para cada colección de datos. Recibe como argumento el conjunto de datos **DataSet11** y la tabla que contiene la información y se coloca en la posición cero (0) con el método **Position**, esto con el fin de que al dar clic sobre el botón **Primero**, se situé en el primer registro de la tabla **clientes**.

Seleccione el objeto **botonsiguiente**, dé doble clic para abrir el editor del procedimiento *botonsiguiente_Click* y escriba el siguiente código:

```
Me.BindingContext(DataSet11, "clientes").Position += 1
```

Cada vez que se dé clic sobre el botón **Siguiente**, se desplazará en un registro sobre la tabla que se está consultando.

Seleccione el objeto **botonanterior**, dé doble clic para abrir el editor del procedimiento *botonanterior_Click* y escriba el siguiente código:

```
Me.BindingContext(DataSet11, "clientes").Position -= 1
```

Cada vez que se dé clic sobre el botón **Anterior**, retrocederá un registro sobre la tabla que se está consultando.

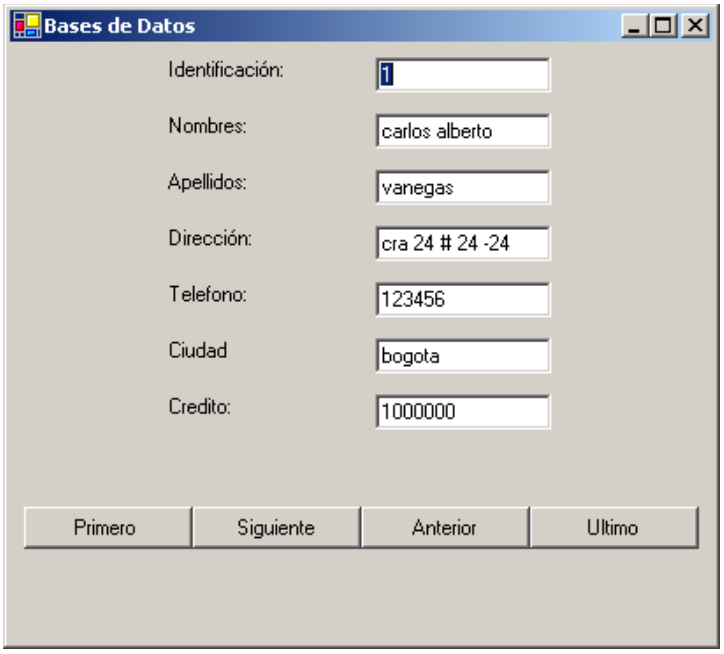
Seleccione el objeto **botonúltimo**, dé doble clic para abrir el editor del procedimiento *botonúltimo_Click* y escriba el siguiente código:

```
Me.BindingContext(DataSet11, "clientes").Position =
Me.BindingContext(DataSet11, "clientes").Count - 1
```

Cada vez que se dé clic sobre el botón **Último**, se contaran el total de registros utilizando el método **count** y se retrocederá una posición para quedar en el último registro de la tabla que se está consultando.

Al ejecutarse nuevamente la aplicación se visualizará la siguiente figura:

Figura 11.26 Aplicación BasesdeDatosBasicas con controles de navegación.



Al dar clic sobre cada uno de los botones se podrá desplazar por cada uno de los registros que se tiene en la tabla que se está consultando.

11.8 Acceder a una Base de Datos mediante el asistente de formularios

Hasta el momento la conexión de la base de datos y la manipulación de los controles se ha realizado con código casi en todos los pasos que se realizaron, pero Visual Basic .NET tiene una serie de asistentes que permiten en forma más sencilla acceder a una base de datos especifica: Para esto se utiliza el **asistente de formulario de datos** el cual se encuentra en la opción **agregar nuevo elemento** del menú **proyecto**.

11.8.1 Ejemplos de acceso a bases de datos

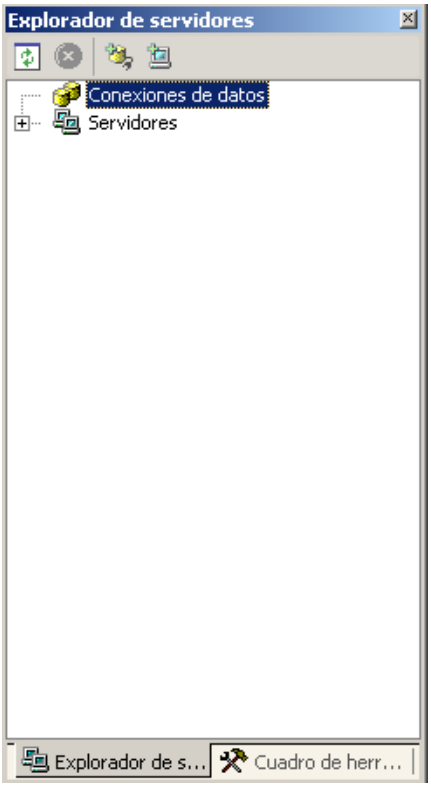
a) Acceder a una base de datos con el asistente de formularios (registro único)

En este caso se realizará el mismo ejemplo que se trabajo anteriormente, por lo cual se asume que ya existe la base de datos **visualnet.mdb** y esta cuenta con la tabla **clientes** y contiene registros. Cree un nuevo proyecto llamado **BasesdeDatosConAsistente**

1. Hacer la conexión a la base de datos

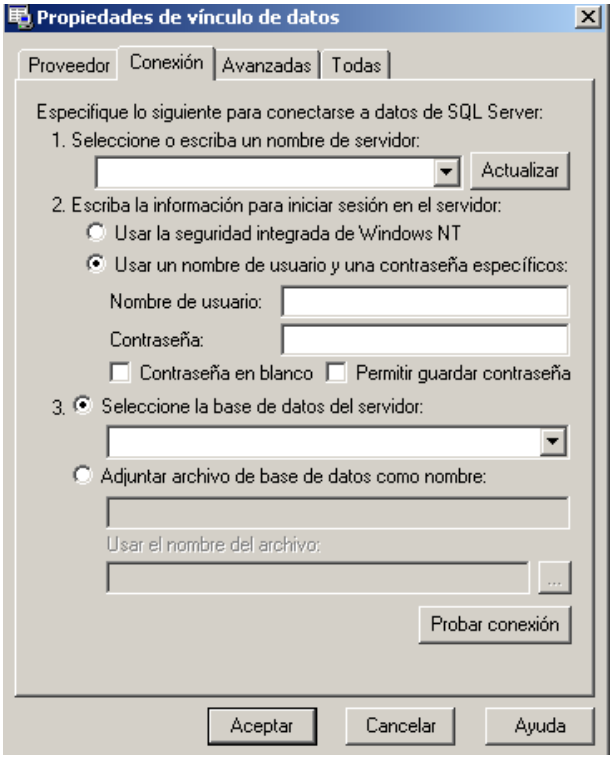
Para realizar una conexión a una base de datos existente seleccione el **explorador de servidores** del menú **Ver**, donde se visualizará el siguiente cuadro de diálogo:

Figura 11.27 Ventana explorador de soluciones.



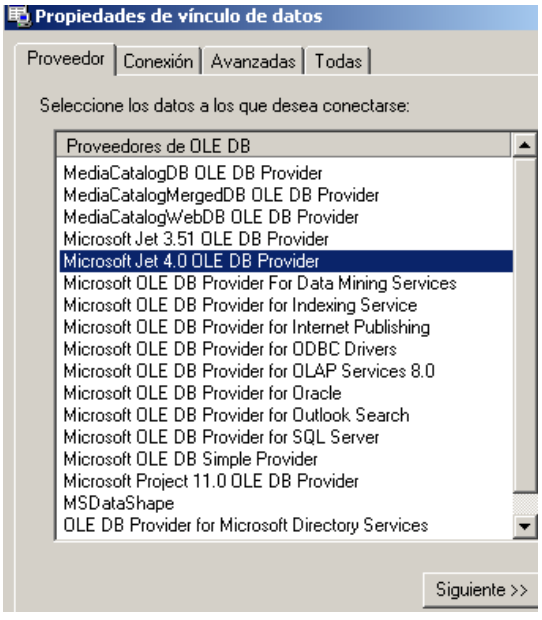
Seleccione el icono **conectar con bases de datos**  para iniciar la conexión a la base de datos y visualizar el cuadro de diálogo **Propiedades de vínculos de datos,** como se muestra en la siguiente figura:

Figura 11.28 Ventana propiedades de vínculos de datos .



Pulse sobre la pestaña llamada **Proveedor** para seleccionar el proveedor de la base de datos. Escoja **Microsoft jet 4.0 OLE DB Provider** y pulse el botón **Siguiente >>.** Se visualizará la figura 11.29:

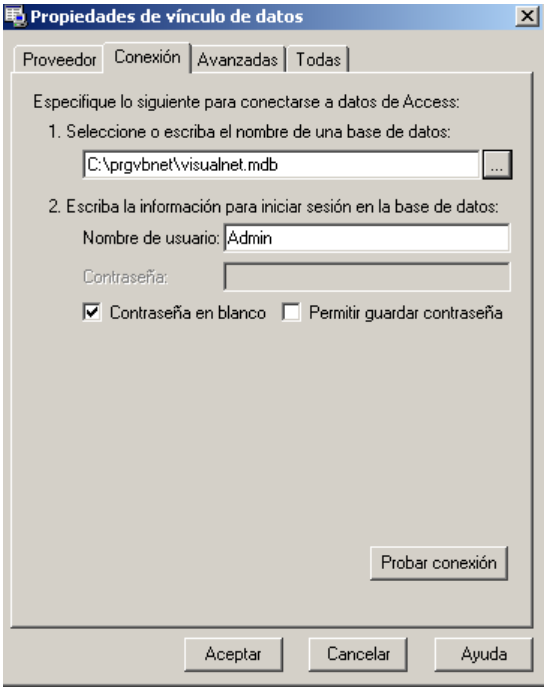
Figura 11.29 Ventana propiedades de vínculos de datos (proveedor).



Al pulsar el botón **Siguiente>>** se visualizará la pestaña **Conexión** del cuadro de diálogo **propiedades de vínculo de datos,** allí en el paso 1, seleccione la base de datos

que previamente se había creado, como se muestra en la siguiente figura:

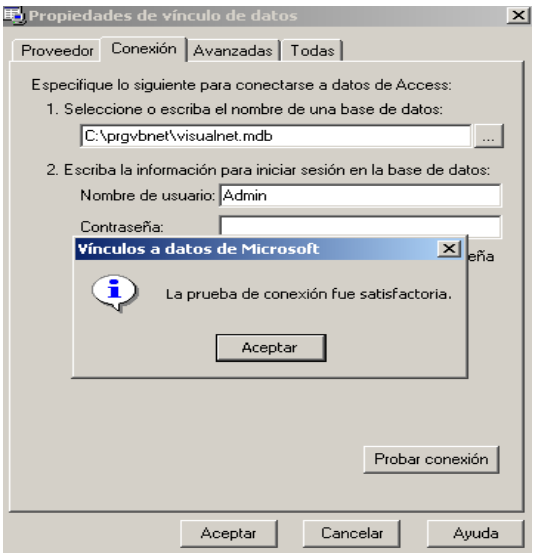
Figura 11.30 Ventana propiedades de vínculos de datos (conexión)



Si desea crear una contraseña a la base de datos desactive el cuadro de verificación **contraseña en blanco** y automáticamente el campo **contraseña** se habilitara para que el usuario digite la contraseña que desee. El paso 2 es opcional.

En este momento se puede verificar la conexión pulsando el botón **Probar conexión**. Si la conexión ha sido exitosa se mostrará la siguiente figura:

Figura 11.31 Ventana prueba de conexión.

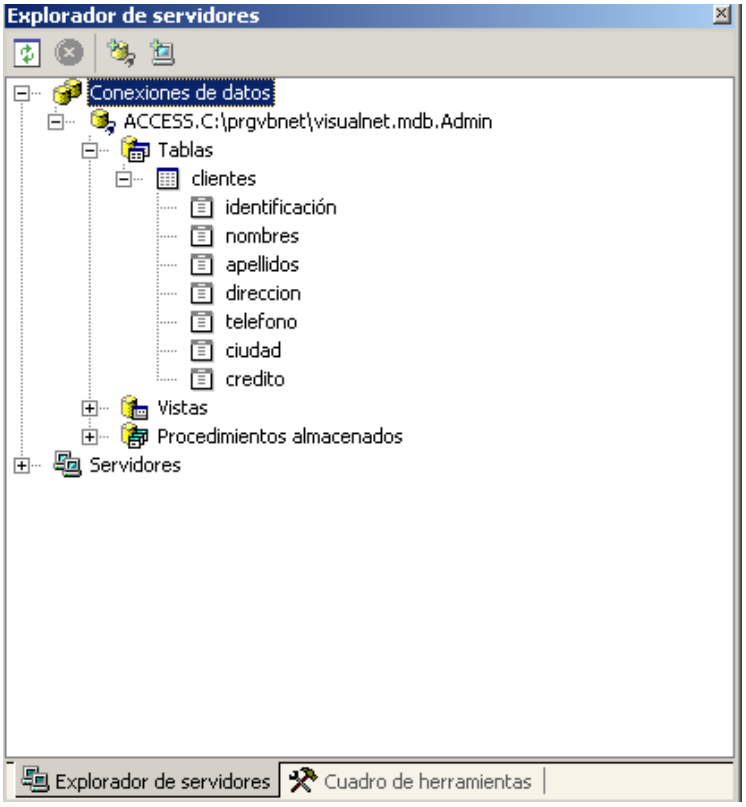


En caso de que la conexión no fuese exitosa, se debe iniciar nuevamente con cada uno de los pasos realizados hasta el momento para resolver el problema. Uno de las posibles fallas en la conexión es que la base de datos este abierta en el momento de la conexión proceda a cerrarla y realice nuevamente la conexión.

Para continuar pulse el botón **Aceptar** de la prueba de conexión, luego pulse el botón **Aceptar** del cuadro de diálogo **Propiedades de vínculo de datos** para terminar la conexión.

En este momento se puede pulsar el signo (+) a la izquierda de la leyenda **conexiones de datos** del cuadro de diálogo **explorador de servidores** para visualizar la conexión a la base de datos **visualnet.mdb**. Cada vez que pulse el signo (+) se podrá apreciar la estructura de la base de datos, como se muestra en la siguiente figura:

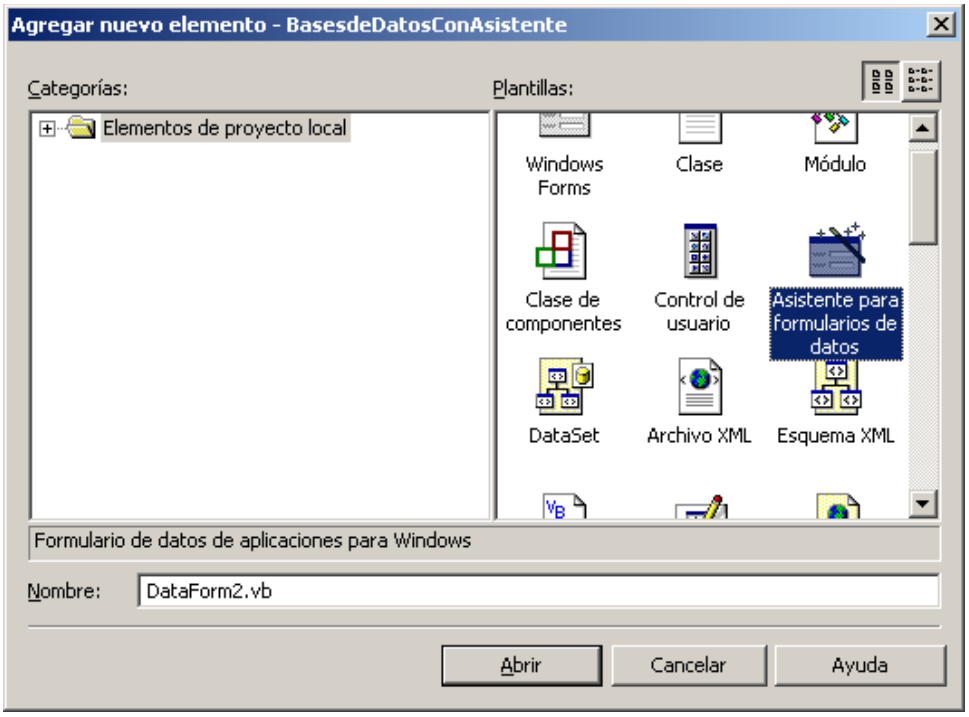
Figura 11.32 Ventana explorador de servidores.



2. Utilizar el asistente para formularios de datos para mostrar registro único

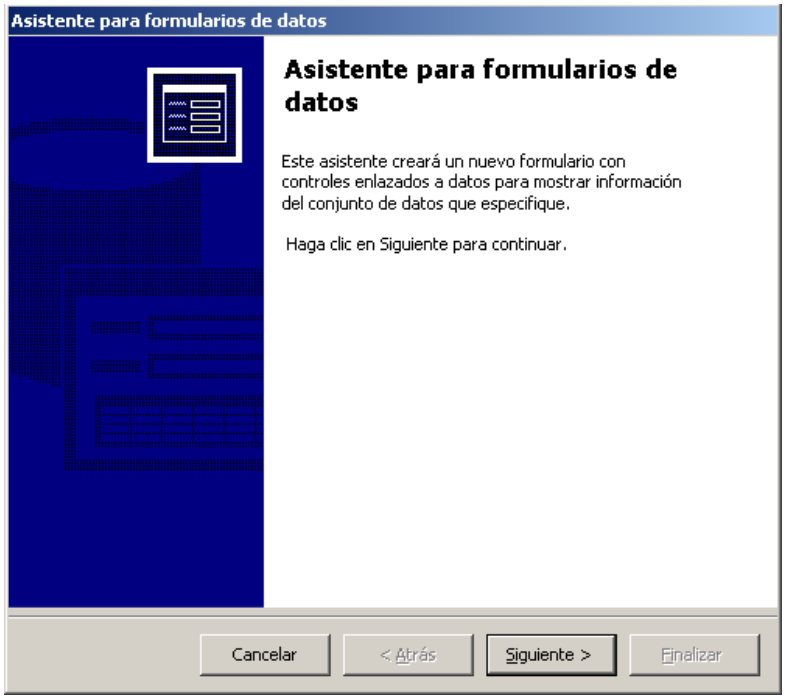
Ahora seleccione la opción **agregar nuevo elemento** del menú **Proyecto** y seleccione el icono **Asistente para formularios de datos**, como se muestra en la siguiente figura:

Figura 11.33 Ventana Agregar nuevo elemento.



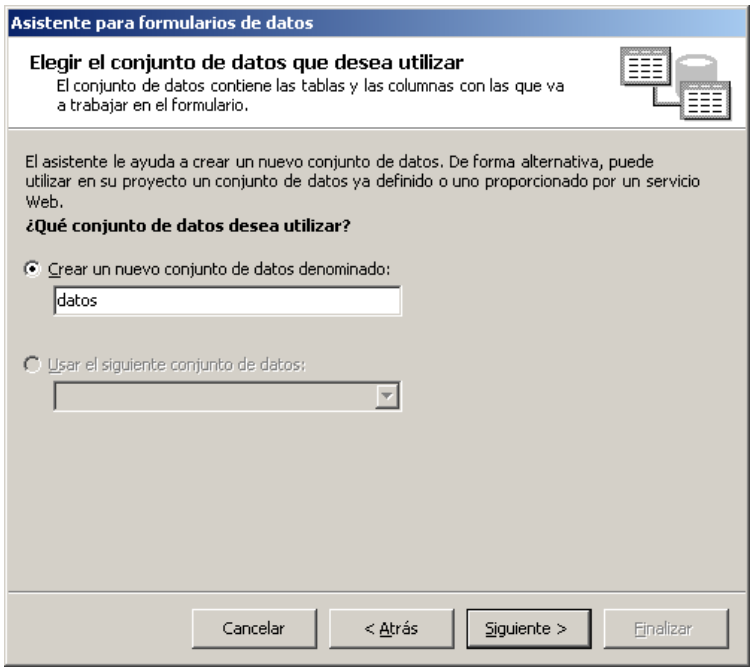
En esta ventana, en la opción **Nombre** se podrá cambiar el nombre del formulario de datos. Cambie el nombre del formulario por **formularioRegistroUnico.vb**. Al pulsar el botón **Abrir** se visualizará la siguiente ventana:

Figura 11.34 Asistente para formulario de datos.



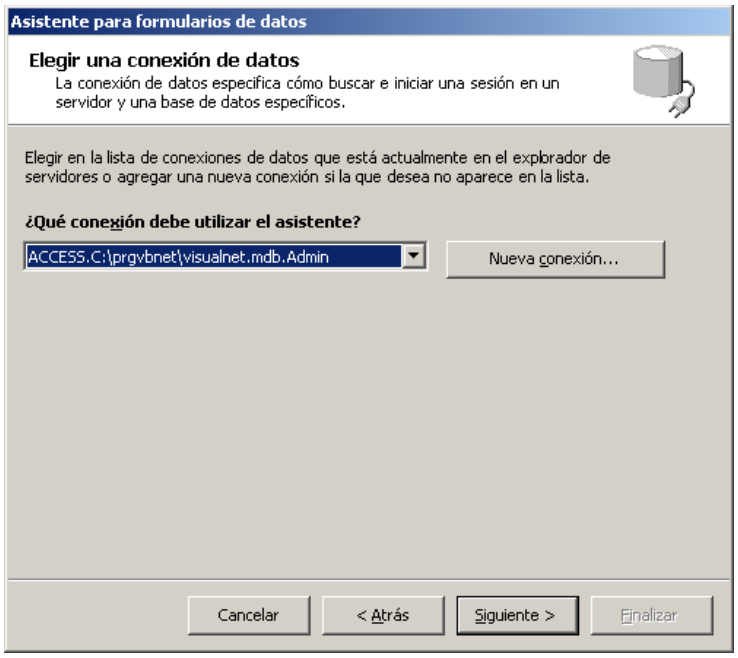
Esta pantalla es el inicio del asistente para formularios de datos, pulse el botón **Siguiente>** para visualizar la siguiente figura:

Figura 11.35 Elegir el conjunto de datos a utilizar.



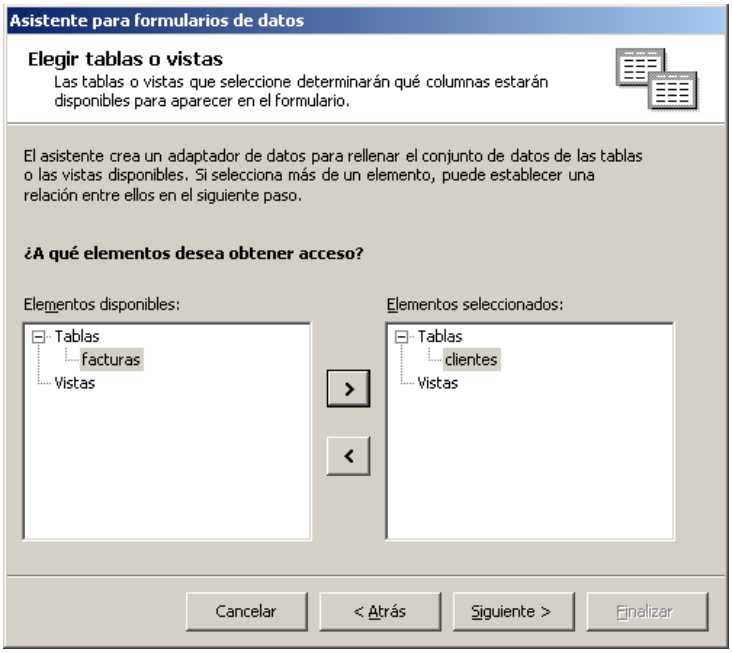
En esta ventana se selecciona el conjunto de datos que contiene las tablas y las columnas con las que va a trabajar el formulario. En la opción **crear un nuevo conjunto de datos** escriba **datos** y pulse el botón **Siguiente** para visualizar la siguiente ventana:

Figura 11.36 Asistente para elegir una conexión de datos.



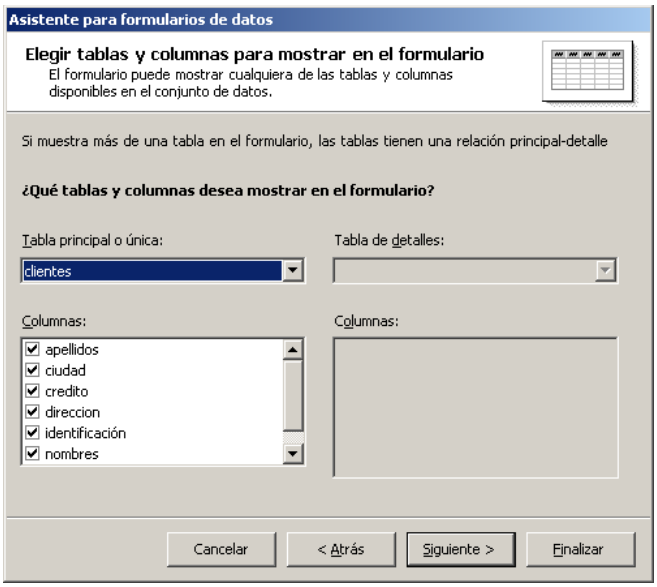
En esta ventana elija la conexión específica y seleccione la conexión que se realizo anteriormente y pulse el botón **Siguiente** > para visualizar la siguiente figura:

Figura 11.37 Asistente para elegir tablas o vistas.



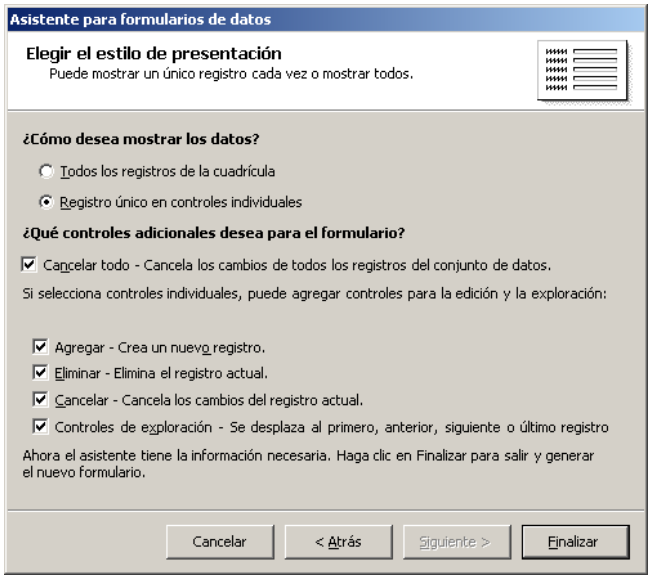
En esta ventana seleccione la tabla **clientes** y pulse el botón ">" para colocarla en la ventana **Elementos seleccionados**. Pulse el botón **Siguiente**> para visualizar la siguiente figura:

Figura 11.38 Asistente para elegir tablas y columnas.



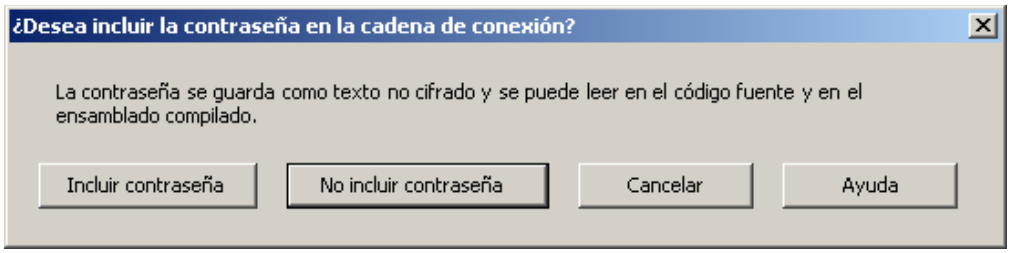
En esta ventana se debe seleccionar los campos que se quieren mostrar en el formulario. Al terminar de escoger los campos deseados pulse el botón **Siguiente** > para ver la siguiente figura:

Figura 11.39 Asistente para elegir el estilo de presentación de los datos.



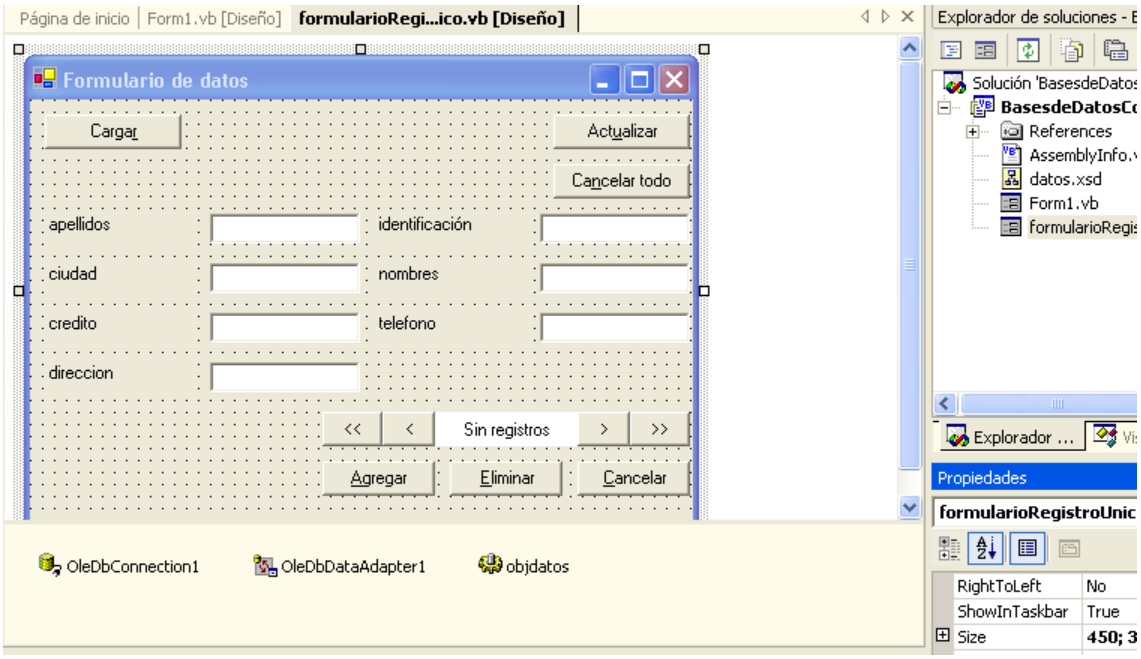
En esta ventana se podrá escoger el estilo de presentación de los datos. Existen dos formas de mostrar los datos: la primera es mostrar todos los registros en una cuadricula y la segunda es mostrar un único registro con controles individuales en el formulario. Escoja la opción **Registro único en controles individuales**. Además se puede seleccionar los controles que se quieren mostrar en el formulario. Por último pulse el botón **Finalizar** para visualizar la siguiente ventana:

Figura 11.40 Ventana para contraseña en la cadena de conexión.



En este cuadro de diálogo el asistente pregunta si se desea incluir una contraseña o no. Pulse el botón **No incluir contraseña**. A la aplicación se agregara debajo del formulario los objetos: **OleDbDataAdapter, OleDbConnection** y **objdatos**. La figura que se visualizará será la siguiente:

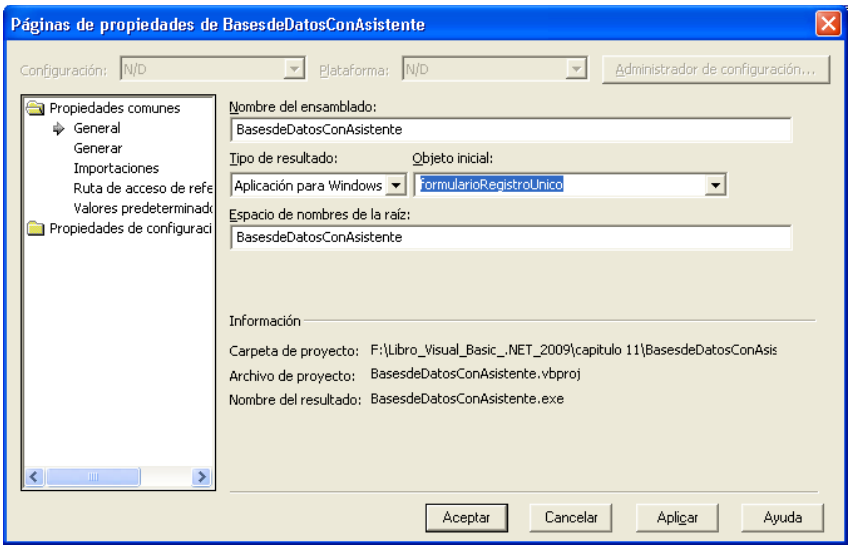
Figura 11.41 Ventana final con el asistente de formulario (registro único).



- **Ejecutar la aplicación**

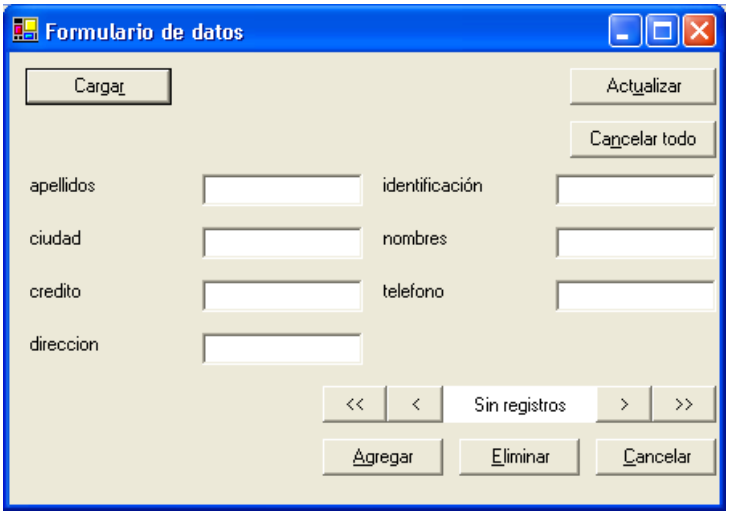
En el menú **Proyecto**, seleccione **propiedades de BasesdeDatosConAsistente,** en la opción **Objeto inicial** seleccione el formulario **formularioDatosConAsistente** y pulse el botón A**ceptar.** Se visualizará la siguiente figura:

Figura 11.42 Páginas de propiedades de BasesdeDatosConAsistente.



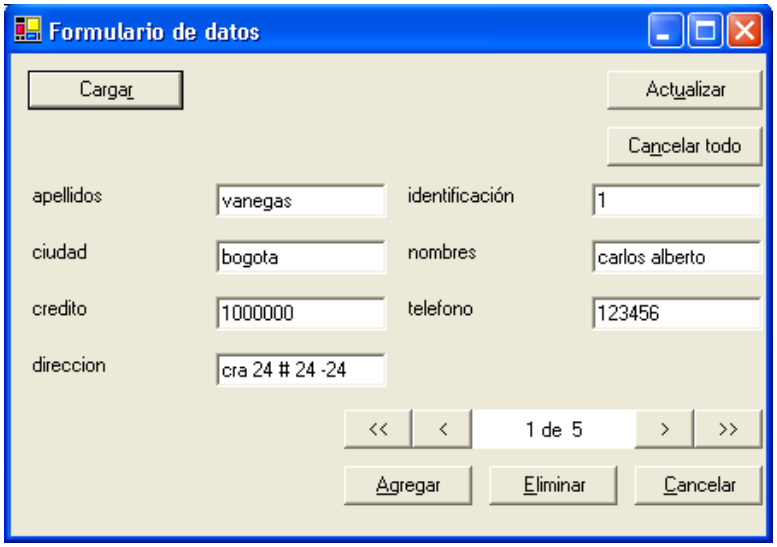
Haga clic en el botón Iniciar de la barra de herramientas estándar o presione **F5** para ejecutar el proyecto. Para visualizar la siguiente figura:

Figura 11.43 Ejecución aplicación BasesdeDatosConAsistente.



Al pulsar el botón **Cargar** se visualizará el primer registro de la tabla seleccionada, como se puede apreciar, en la figura 11.44.

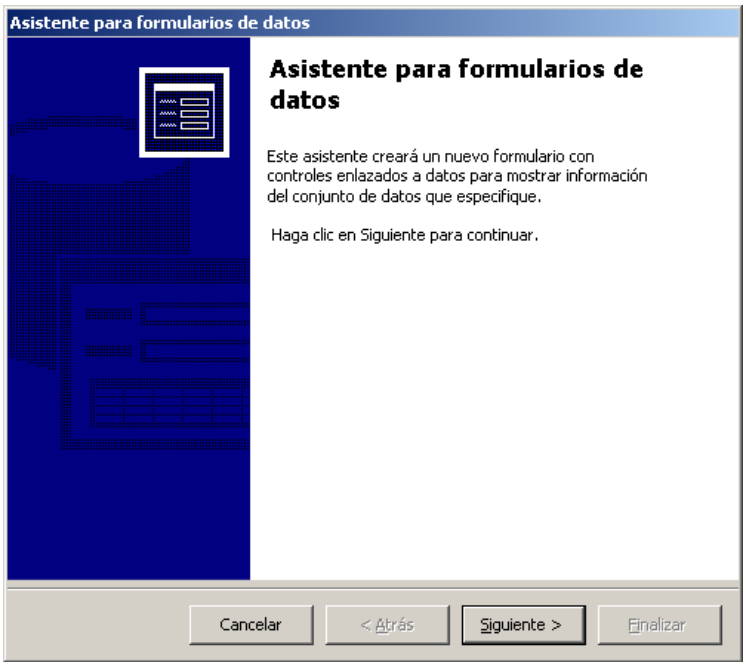
Figura 11.44 Ejecución BasesdeDatosConAsistente al pulsar el botón Cargar.



b) Utilizar el asistente para visualizar un formulario con una cuadricula de datos

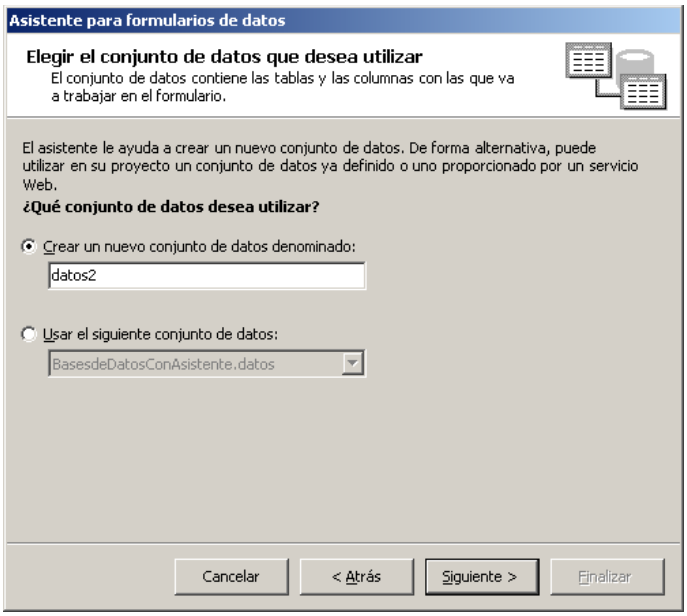
En el mismo proyecto seleccione nuevamente la opción **agregar nuevo elemento** del menú **Proyecto** y seleccione el icono **Asistente para formularios de datos**. En esta ventana en la opción **Nombre** se podrá cambiar el nombre del formulario de datos. El nombre del formulario será **formulariocuadricula.vb**. Al pulsar el botón **Abrir** se visualizará la siguiente ventana:

Figura 11.45 Asistente para formulario de datos para la cuadricula.



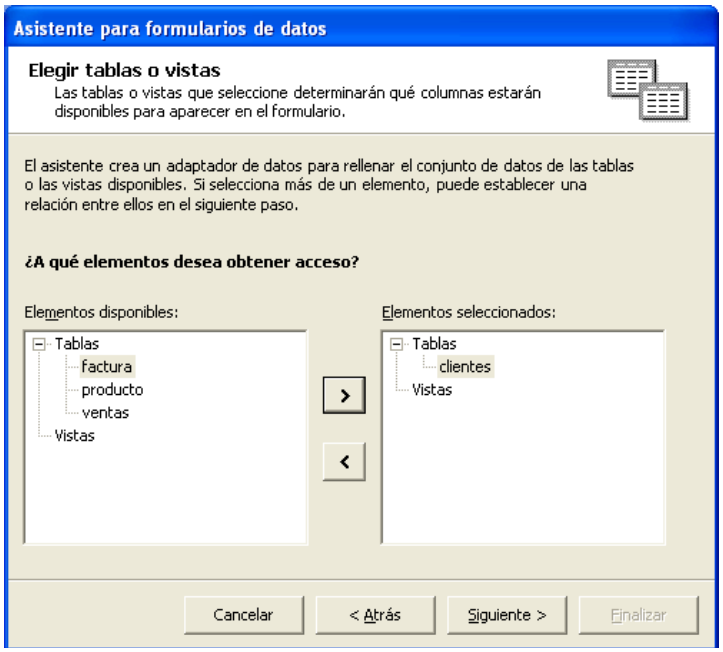
Esta pantalla es el inicio del asistente para formularios de datos, pulse el botón **Siguiente** para visualizar la siguiente figura:

Figura 11.46 Asistente para elegir un conjunto de datos (cuadricula).



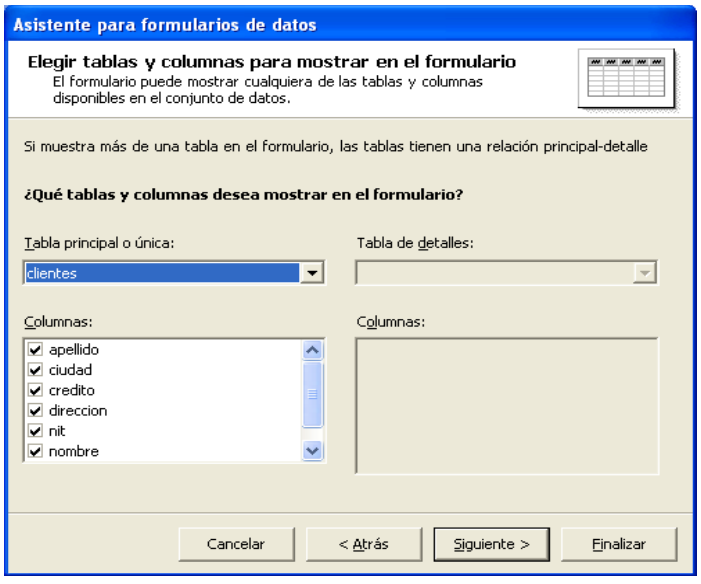
En esta ventana se selecciona el conjunto de datos que contiene las tablas y las columnas con las que va a trabajar el formulario. En la opción **crear un nuevo conjunto de datos** escriba **datos2** y pulse el botón **Siguiente >** para visualizar la siguiente ventana:

Figura 11.47 Asistente para elegir tablas o vistas.



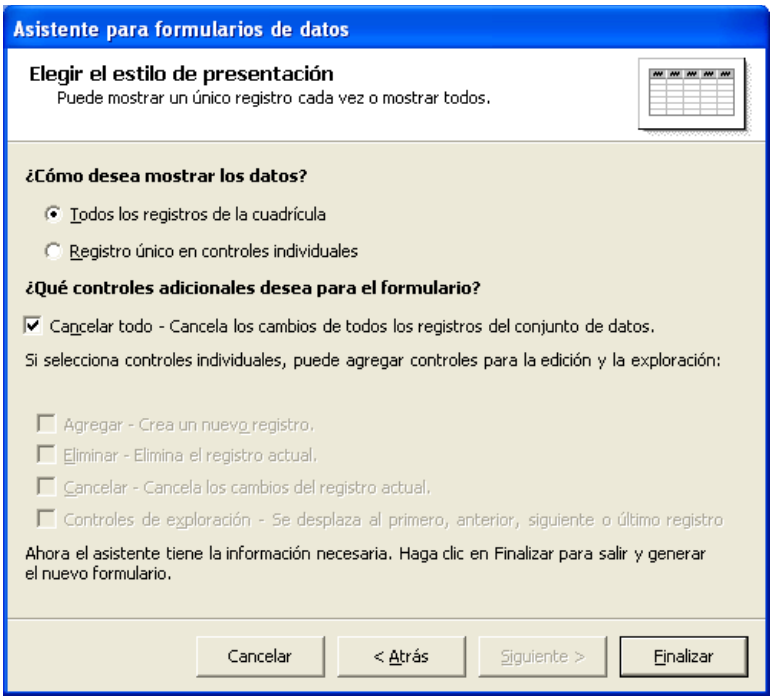
En esta ventana seleccione la tabla **clientes** y pulse el botón **">"** para colocarla en la ventana **Elementos seleccionados**. Pulse el botón **Siguiente** > para visualizar la siguiente figura:

Figura 11.48 Elegir tablas y columnas para mostrar en el formulario (cuadricula).



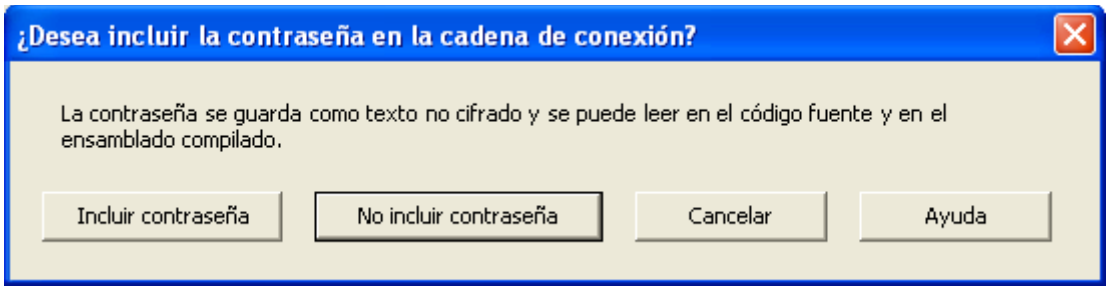
En esta ventana se debe seleccionar los campos que se quieren mostrar en el formulario. Al escoger cada uno de los campos pulse el botón **Siguiente** > para ver la siguiente figura:

Figura 11.49 Elegir estilo de presentación para mostrar datos.



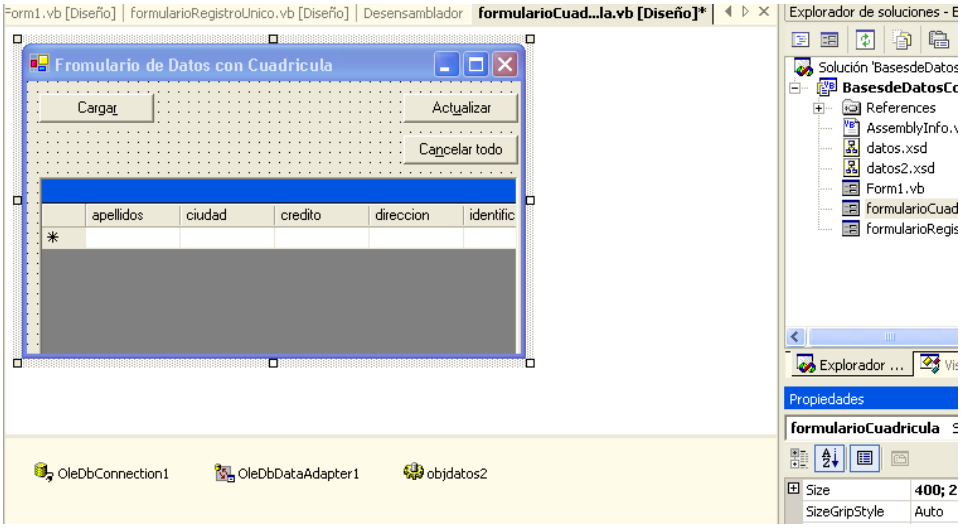
Escoja la opción de **Todos los registros de la cuadricula**. Por último pulse el botón **Finalizar** para visualizar la siguiente ventana:

Figura 11.50 Ventana para contraseña en la cadena de conexión.



En este cuadro de diálogo el asistente pregunta si se desea incluir una contraseña o no. Pulse el botón **No incluir contraseña.** A la aplicación se le agregará debajo del formulario los objetos: **OleDbDataAdapter** y **OleDbConnection** y **objdatos2**. La figura que se visualizará será la siguiente:

Figura 11.51 Formulario con una cuadricula de datos.



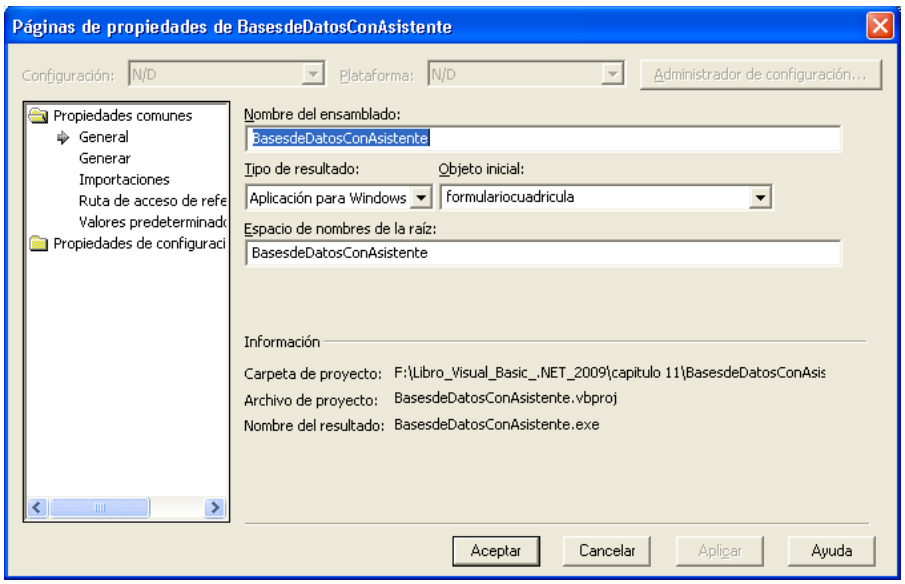
- **Ejecutar la aplicación**

Para ejecutar el proyecto en el entorno de desarrollo de visual Basic.NET, se debe realizar lo siguiente:

- En el menú **Proyecto**, seleccione **propiedades de BasesdeDatosConAsistente,** en la opción **Objeto inicial** seleccione el formulario **formulariocuadricula** y pulse A**ceptar.**

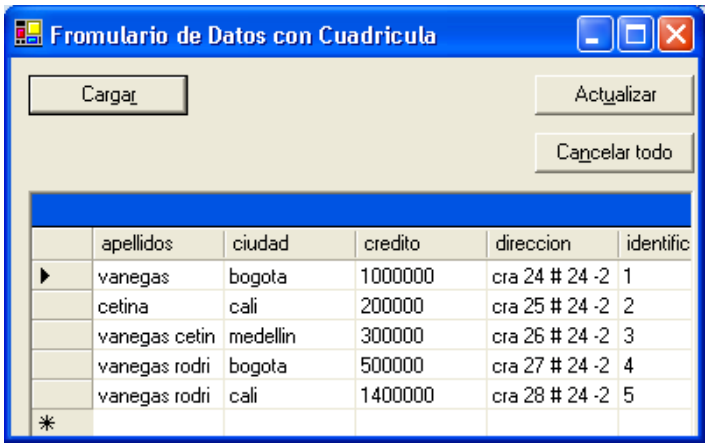
Se visualizará la siguiente figura:

Figura 11.52 Páginas de propiedades de BasesdeDatosConAsistente.



Haga clic en el botón Iniciar de la barra de herramientas estándar o presione **F5** para ejecutar el proyecto. Al visualizar el formulario pulse el botón **Cargar** para ver la siguiente figura:

Figura 11.53 Aplicación BasesdeDatosConAsistente (formulariocuadricula).



c) Utilizar el asistente para visualizar un formulario con una cuadricula de datos para dos tablas (cuadricula).

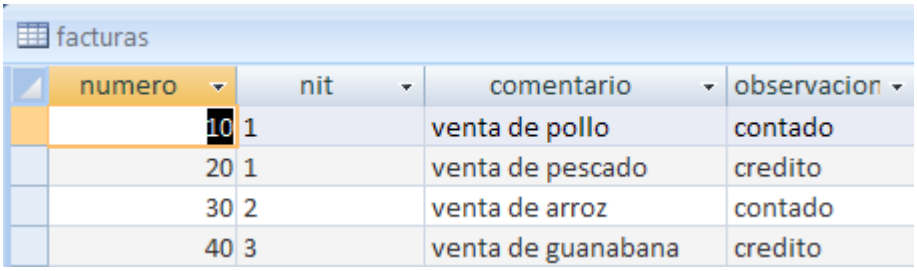
1. Crear una tabla llamada **Facturas** en la base de datos **visualnet.mdb** con la siguiente estructura:

   Tabla 11.11 Campos de la tabla Facturas.

Nombre del campo	Tipo de dato	longitud
numero	numérico	
nit	Texto	13
comentario	Texto	25
observacion	Texto	25

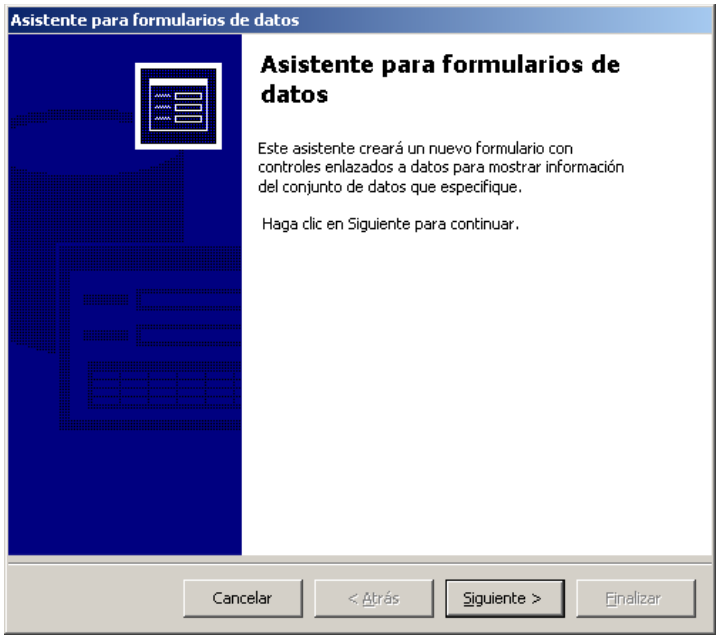
La tabla con 4 registros quedara de la siguiente forma:

Figura 11.54 Registros de la tabla Facturas.



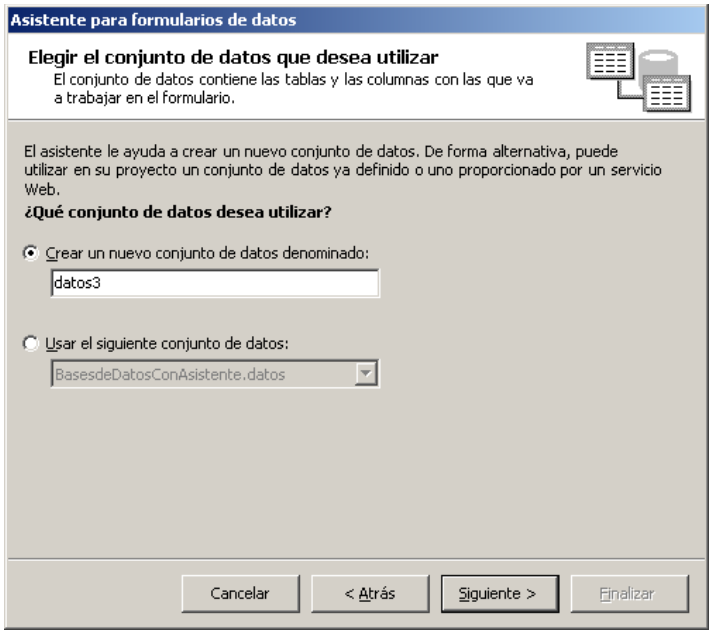
Nuevamente seleccione la opción **agregar nuevo elemento** del menú **Proyecto** y seleccione el icono **Asistente para formularios de datos**. En esta ventana, en la opción **Nombre**, cambie el nombre del formulario de datos por **formularioRelacionadoCuadricula**.vb. Al pulsar el botón **Abrir** se visualizará la siguiente ventana:

Figura 11.55 Asistente para formulario de datos con relación.



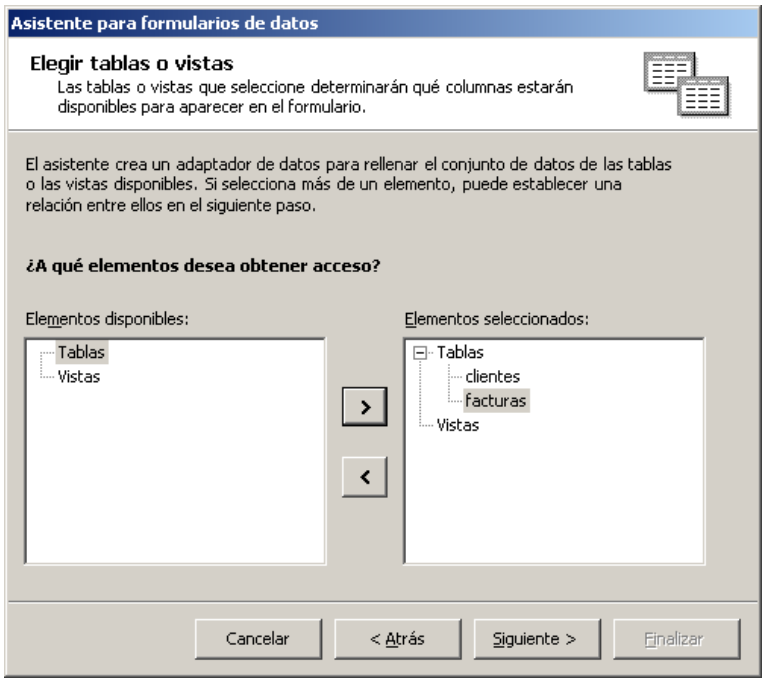
Esta pantalla es el inicio del asistente para formularios de datos, pulse el botón **Siguiente** para visualizar la siguiente figura:

Figura 11.56 Asistente para elegir el conjunto de datos a utilizar.



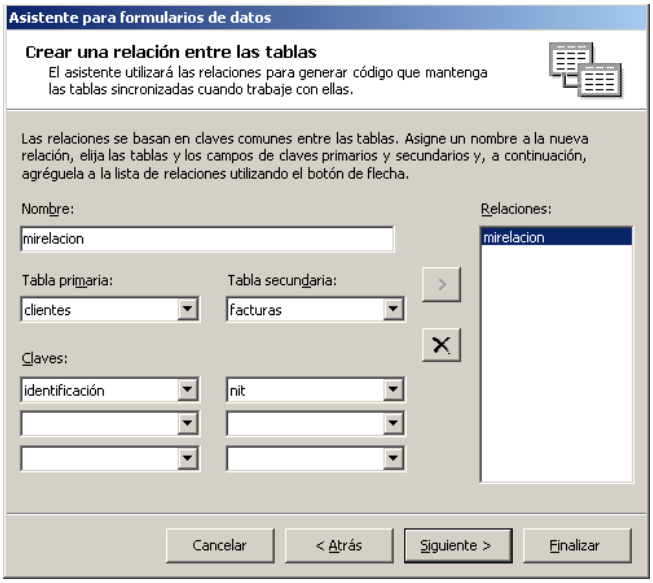
En esta ventana se selecciona el conjunto de datos que contiene las tablas y las columnas con las que va a trabajar el formulario. En la opción **crear un nuevo conjunto de datos** escriba **datos3** y pulse el botón **Siguiente** > hasta visualizar la siguiente ventana:

Figura 11.57 Asistente para elegir las tablas de clientes y fracturas.



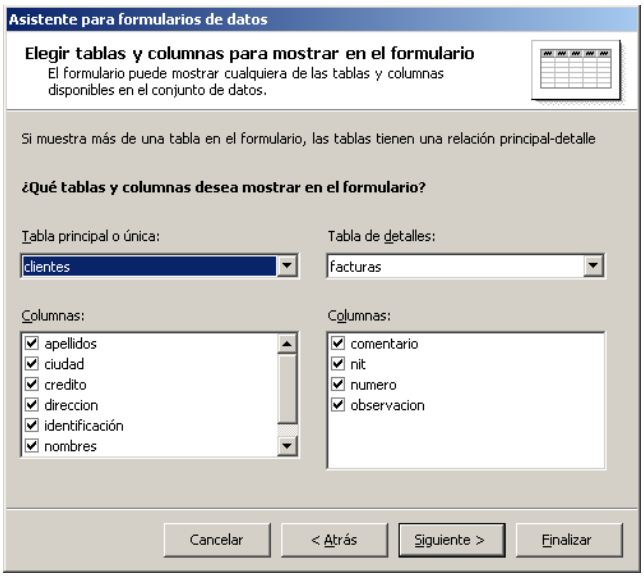
En esta ventana seleccione las tablas **clientes** y **facturas**, pulse el botón **">"** para colocarlas en la ventana **Elementos seleccionados**. Pulse el botón **Siguiente** > para visualizar la siguiente figura:

Figura 11.58 Asistente para crear una relación entre las tablas.



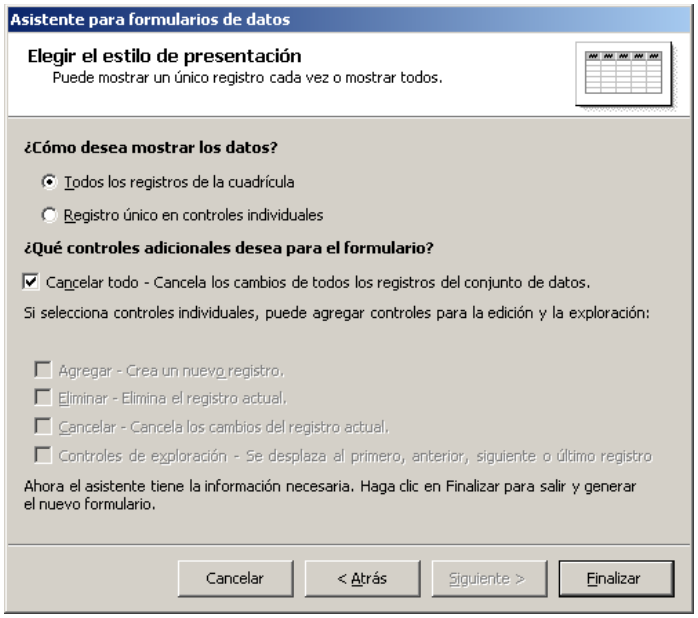
En esta ventana en el campo **Nombre** escriba **mirelacion**. Como tabla primaria seleccione la tabla **clientes** y como tabla secundaria la tabla **facturas**, en la opción de **claves** seleccione los campos **identificación** de clientes y **nit** de facturas, pulse el botón **">"** para colocar la relación en la ventana **relaciones**. Por último pulse el botón **Siguiente** > para visualizar la siguiente figura:

Figura 11.59 Elegir campos para mostrar en el formulario.



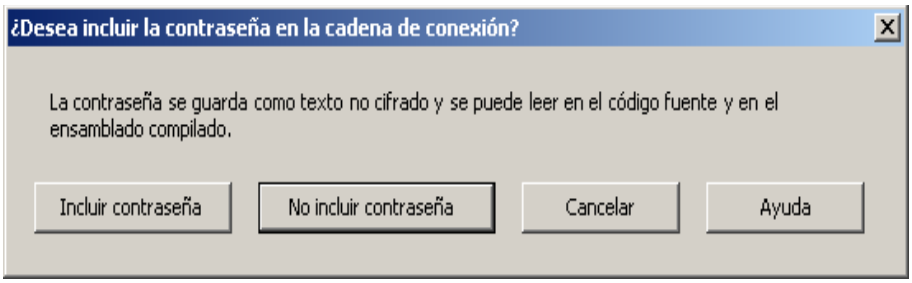
En esta ventana se debe seleccionar los campos que se quieren mostrar en el formulario. Al escoger cada uno de los campos pulse el botón **Siguiente >** para ver la siguiente figura:

Figura 11.60 Asistente para elegir el estilo de presentación.



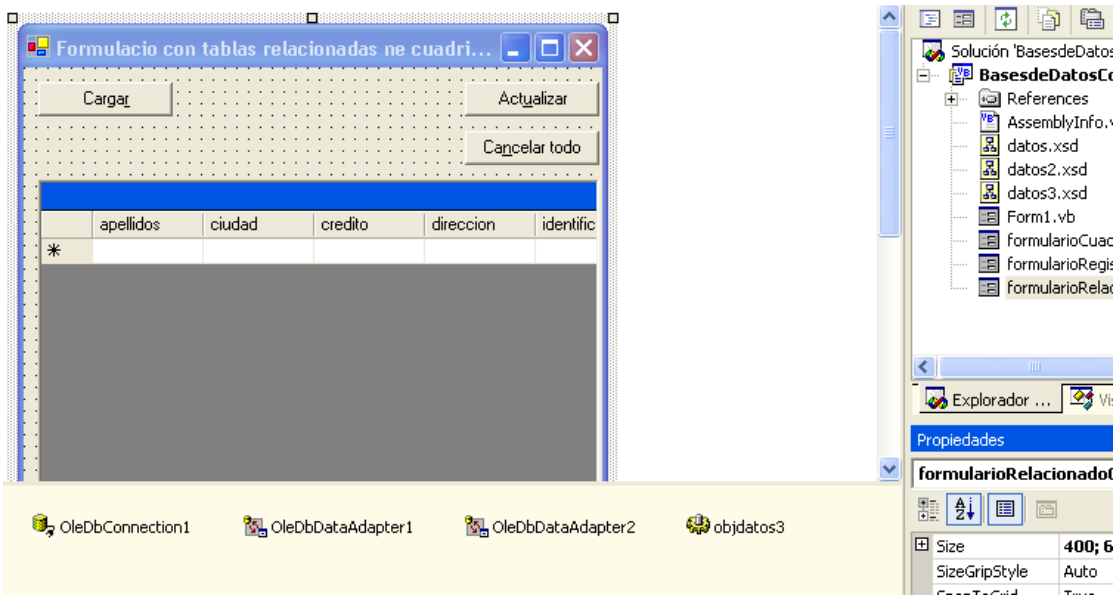
Escoja la opción de **Todos los registros de la cuadricula**. Por último pulse el botón **Finalizar** para visualizar la siguiente ventana:

Figura 11.61 Ventana para contraseña en la cadena de conexión.



En este cuadro de diálogo el asistente pregunta si se desea incluir una contraseña o no. Pulse el botón **No incluir contraseña.** A la aplicación se agregara debajo del formulario los objetos: **OleDbDataAdapter** y **OleDbConnection** y **objdatos3**. La figura que se visualizará será la siguiente:

Figura 11.62. Formulario con una cuadricula de datos para dos tablas



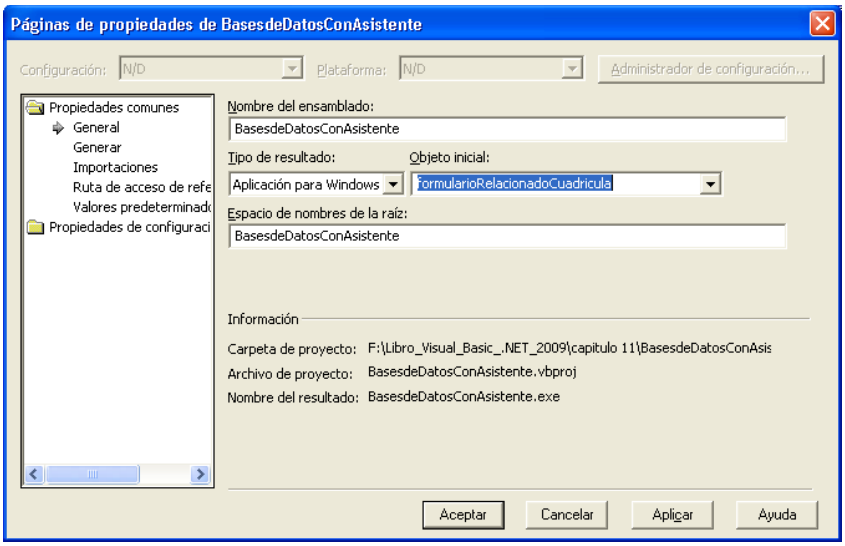
- **Ejecutar la aplicación**

Para ejecutar el proyecto en el entorno de desarrollo de visual Basic.NET, se debe realizar lo siguiente:

- En el menú **Proyecto**, seleccione **propiedades de BasesdeDatosConAsistente,** en la opción **Objeto inicial** seleccione el formulario **formularioRelacionCuadricula** y pulse el botón A**ceptar.**

Se visualizará la siguiente figura:

Figura 11.63 Páginas de propiedades de BasesdeDatosConAsistente.



Dé clic en el botón Iniciar de la barra de herramientas estándar o presione **F5** para ejecutar el proyecto. Al visualizar el formulario pulsar el botón **Cargar** para ver la siguiente figura:

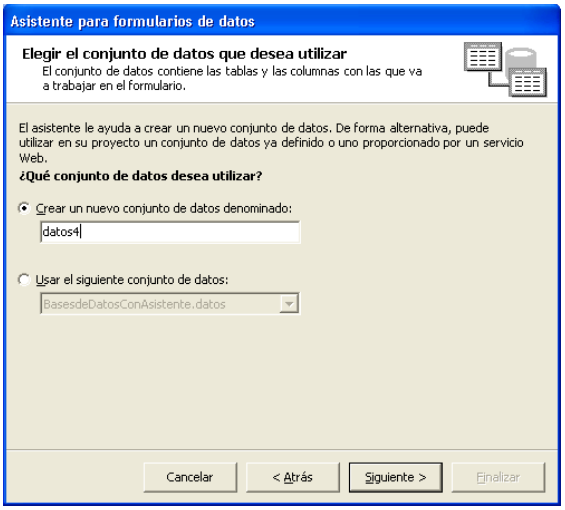
Figura 11.64 Ejecución aplicación BasesdeDatosConAsistente.

4) Utilizar el asistente para visualizar un formulario con una cuadricula de datos para dos tablas (registro único).

Realizando las operaciones anteriores seleccione nuevamente la opción **agregar nuevo elemento** del menú **Proyecto** y seleccione el icono **Asistente para formularios de datos**. En esta ventana, en la opción **Nombre** cambie el nombre del formulario de datos

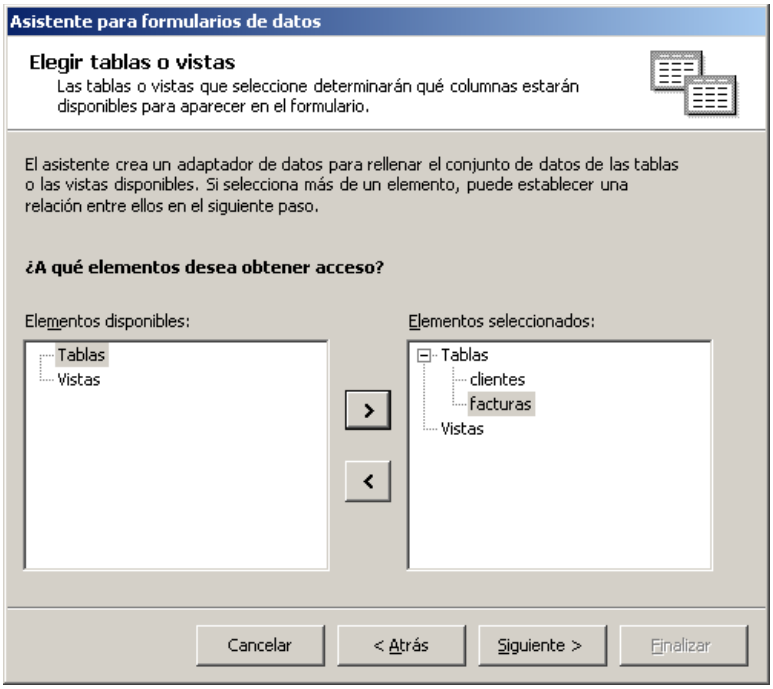
por **formularioRelacionadoRegistroUnico**.vb. Al pulsar el botón **Abrir** y después **Siguiente**> se visualizará la siguiente ventana:

Figura 11.65 Asistente para elegir el conjunto de datos a utilizar (registro único).



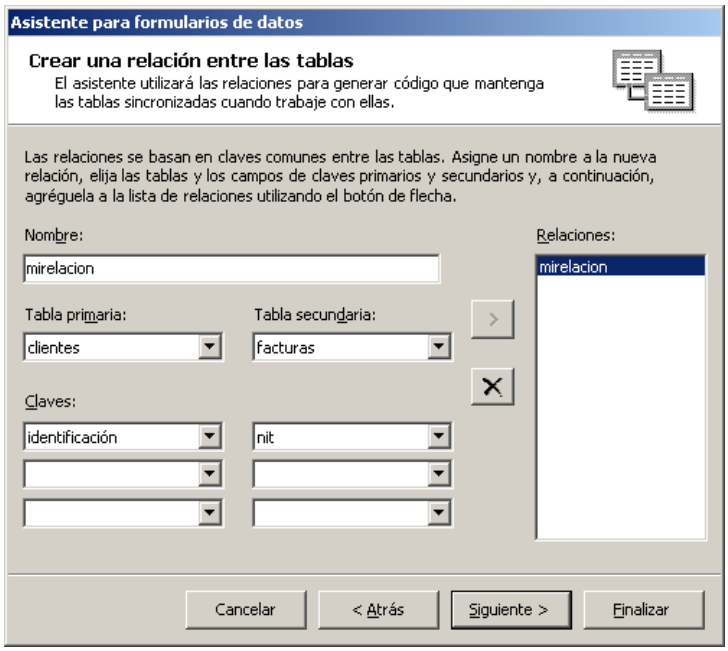
En esta ventana se selecciona el conjunto de datos que contiene las tablas y las columnas con las que va a trabajar el formulario. En la opción **crear un nuevo conjunto de datos** escriba **datos4** y pulse el botón **Siguiente** > hasta visualizar la siguiente ventana:

Figura 11.66 Asistente para elegir las tablas de clientes y fracturas.



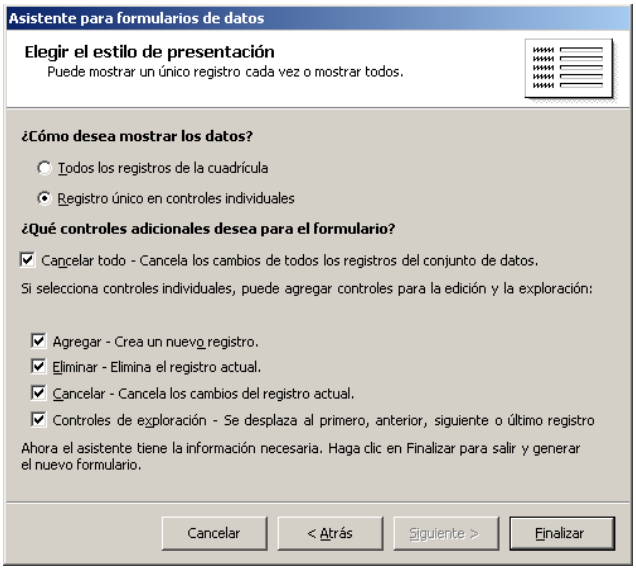
En esta ventana seleccione las tablas **clientes** y **facturas**, pulse el botón **">"** para colocarlas en la ventana **Elementos seleccionados**. Pulse el botón **Siguiente** > para visualizar la siguiente figura:

Figura 11.67 Crear una relación entre las tablas clientes y facturas.



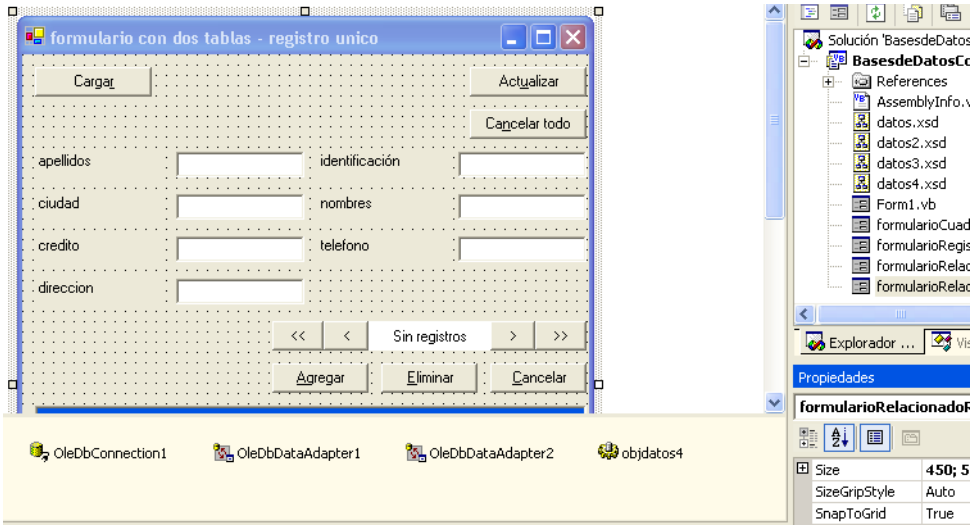
En esta ventana en el campo **Nombre** escriba **mirelacion**. Como tabla primaria seleccione la tabla **clientes** y como tabla secundaria la tabla **facturas**, en la opción de **claves** seleccione los campos **identificación** de clientes y **nit** de facturas, pulse el botón **“>”** para colocar la relación en la ventana **relaciones**. Por último pulse el botón **Siguiente** > para visualizar la siguiente figura:

Figura 11.68 Asistente para elegir el estilo de presentación.



Escoja la opción de **Registro único con controles individuales**. Por último pulse el botón **Finalizar** hasta visualizar la siguiente ventana:

Figura 11.69 Formulario tablas relacionadas con registro único.



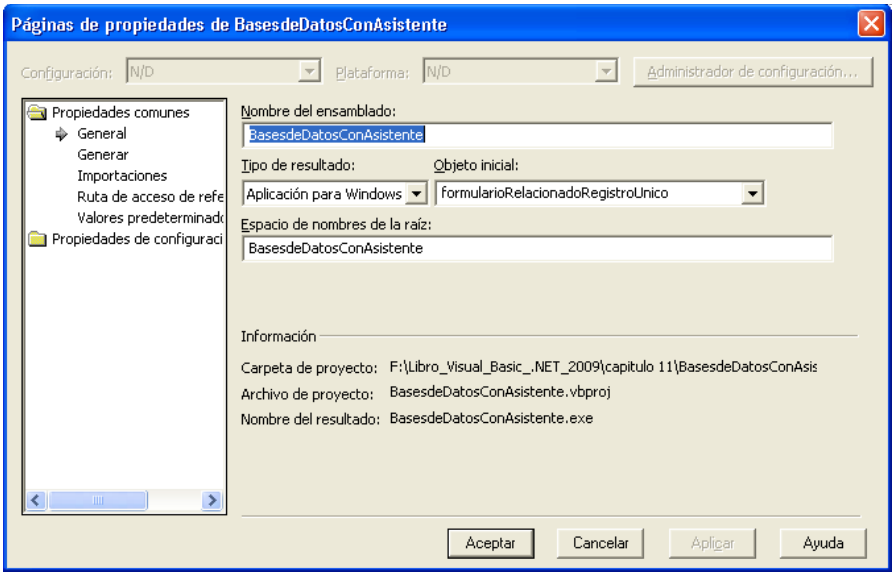
- **Ejecutar la aplicación**

Para ejecutar el proyecto en el entorno de desarrollo de visual Basic.NET, se debe realizar lo siguiente:

- En el menú **Proyecto**, seleccione **propiedades de BasesdeDatosConAsistente,** en la opción **Objeto inicial** seleccione el formulario **formularioRelacionregistroUnico** y pulse A**ceptar.**

Se visualizará la siguiente figura:

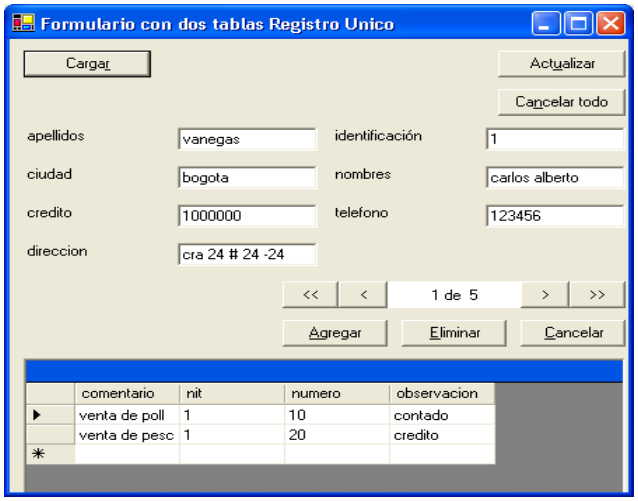
Figura 11.70 Páginas de propiedades de BasesdeDatosConAsistente.



Haga clic en el botón Iniciar de la barra de herramientas estándar o presione **F5** para

ejecutar el proyecto. Al visualizar el formulario pulse el botón **Cargar** para ver la siguiente figura:

Figura 11.71 Ejecución BasesdeDatosConAsistente formulariorleacion único.



12. ASP.NET

ASP.NET es un marco de trabajo de programación generado en Common Language Runtime que puede utilizarse en un servidor para generar eficaces aplicaciones Web. ASP.NET ofrece varias ventajas importantes acerca de los modelos de programación Web anteriores:

- **Mejor rendimiento.** ASP.NET es un código de Common Language Runtime compilado que se ejecuta en el servidor. A diferencia de sus predecesores, ASP.NET puede aprovechar las ventajas del enlace anticipado, la compilación just-in-time, la optimización nativa y los servicios de caché desde el primer momento. Esto supone un incremento espectacular del rendimiento antes de siquiera escribir una línea de código.

- **Compatibilidad con herramientas de primer nivel.** El marco de trabajo de ASP.NET se complementa con un diseñador y una caja de herramientas muy completos en el entorno integrado de programación (Integrated Development Environment, IDE) de Visual Studio. La edición WYSIWYG, los controles de servidor de arrastrar y colocar y la implementación automática son sólo algunas de las características que proporciona esta eficaz herramienta.

- **Eficacia y flexibilidad.** Debido a que ASP.NET se basa en Common Language Runtime, la eficacia y la flexibilidad de toda esa plataforma se encuentra disponible para los programadores de aplicaciones Web. La biblioteca de clases de .NET Framework, la Mensajería y las soluciones de Acceso a datos se encuentran accesibles desde el Web de manera uniforme. ASP.NET es también independiente del lenguaje, por lo que puede elegir el lenguaje que mejor se adapte a la aplicación o dividir la aplicación en varios lenguajes. Además, la interoperabilidad de Common Language Runtime garantiza que la inversión existente en programación basada en COM se conserva al migrar a ASP.NET.

- **Simplicidad.** ASP.NET facilita la realización de tareas comunes, desde el sencillo envío de formularios y la autenticación del cliente hasta la implementación y la configuración de sitios. Por ejemplo, el marco de trabajo de página de ASP.NET permite generar interfaces de usuario, que separan claramente la lógica de aplicación del código de presentación, y controlar eventos en un sencillo modelo de procesamiento de formularios de tipo Visual Basic. Además, Common Language Runtime simplifica la programación, con servicios de código administrado como el recuento de referencia automático y el recolector de elementos no utilizados.

- **Facilidad de uso.** ASP.NET emplea un sistema de configuración jerárquico, basado en texto, que simplifica la aplicación de la configuración al entorno de servidor y las aplicaciones Web. Debido a que la información de configuración se almacena como texto sin formato, se puede aplicar la nueva configuración sin la ayuda de herramientas de administración local. Esta filosofía de "administración local cero" se extiende asimismo a la implementación de las aplicaciones ASP.NET Framework. Una aplicación ASP.NET Framework se implementa en un servidor sencillamente mediante la copia de los archivos necesarios al servidor. No se requiere el reinicio del servidor, ni siquiera para implementar o reemplazar el código compilado en ejecución.

- **Escalabilidad y disponibilidad.** ASP.NET se ha diseñado teniendo en cuenta la escalabilidad, con características diseñadas específicamente a medida, con el fin de mejorar el rendimiento en entornos agrupados y de múltiples procesadores. Además, el motor de tiempo de ejecución de ASP.NET controla y administra los procesos de cerca, por lo que si uno no se comporta adecuadamente (filtraciones, bloqueos), se puede crear un proceso nuevo en su lugar, lo que ayuda a mantener la aplicación disponible constantemente para controlar solicitudes.

- **Posibilidad de personalización y extensibilidad.** ASP.NET presenta una arquitectura bien diseñada que permite a los programadores insertar su código en el nivel adecuado. De hecho, es posible extender o reemplazar cualquier subcomponente del motor de tiempo de ejecución de ASP.NET con su propio componente escrito personalizado. La implementación de la autenticación personalizada o de los servicios de estado nunca ha sido más fácil.

- **Seguridad.** Con la autenticación de Windows integrada y la configuración por aplicación, se puede tener la completa seguridad de que las aplicaciones están a salvo.

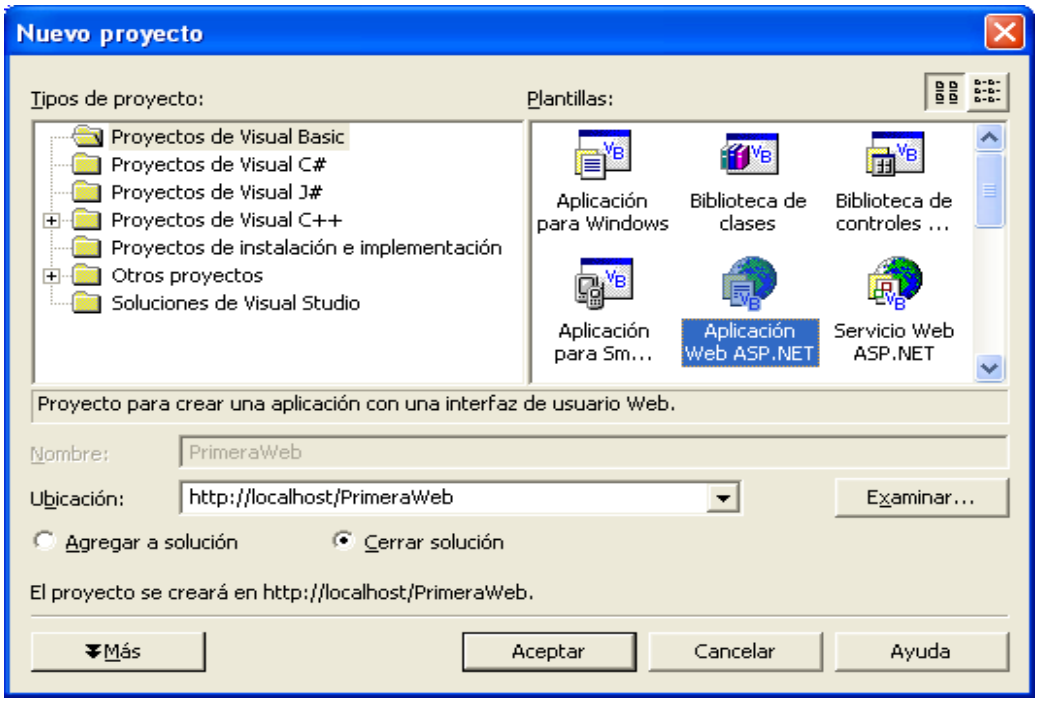
12.1 Primera aplicación ASP.NET

Realizar una aplicación Web ASP.NET que permita a un usuario digitar dos números y que se imprima la multiplicación de estos.

Para crear una aplicación ASP.NET se siguen los mismos pasos que se realizaron para una **Aplicación para Windows,** la diferencia se genera en el tipo de plantilla que se escoge, para este caso se utiliza la plantilla **Aplicación Web ASP.NET.** Entonces para crear un proyecto ASP.NET en Visual Basic .NET es necesario realizar los siguientes pasos:

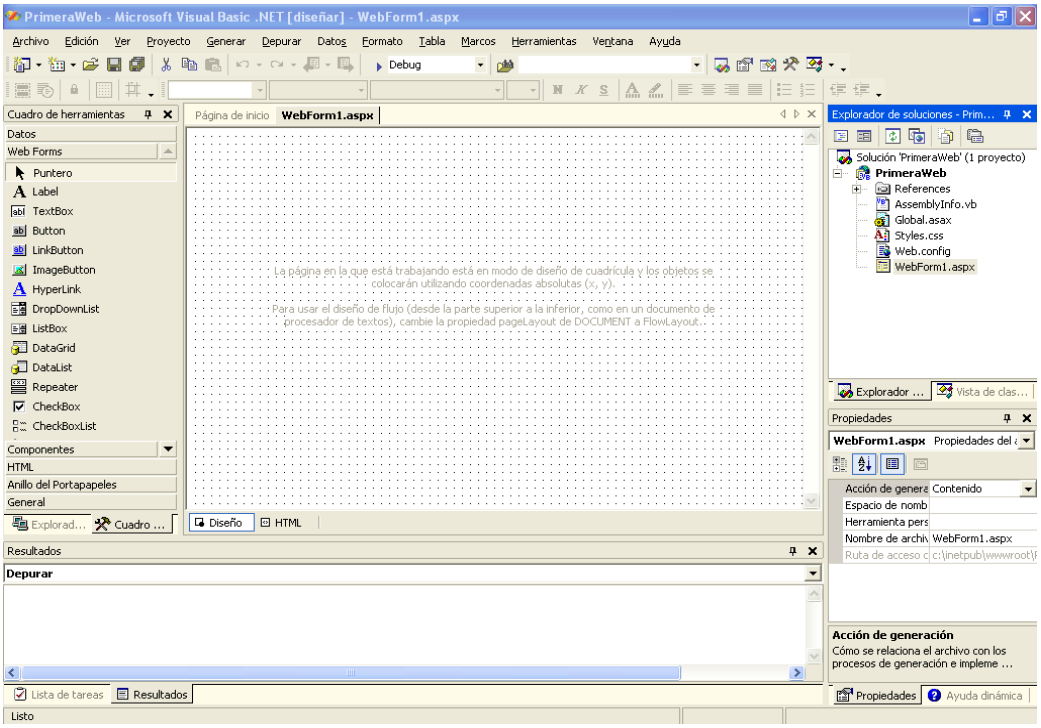
1. Abrir Visual Studio .NET.
2. En el menú **Archivo** seleccionar **Nuevo** a continuación hacer clic en **Proyecto**.
3. En el panel **Tipos de proyecto** hacer clic en **Proyectos de Visual Basic**. En el panel **Plantillas** hacer clic en **Aplicación Web ASP.NET**.
4. En el campo **Ubicación** se indica el nombre del servidor; en este caso **localhost** y como nombre de la aplicación escriba **PrimeraWeb** (http:/localhost/PrimeraWeb). La aplicación se guarda en la carpeta **Inetpub\wwwroot** que se crea cuando se instala VisualStudio .NET. todas las aplicaciones Web se guardarán en dicha carpeta.

Figura 12.1 Ventana Nuevo proyecto Web (ASP.NET).



Al hacer clic en el botón **Aceptar** se visualizará la siguiente figura:.

Figura 12.2 Ventana aplicación Web (ASP.NET).



En esta ventana se tiene la mayoría de elementos que se visualizaron en las aplicaciones para Windows. Como se puede observar se encuentra el cuadro de herramientas, la ventana de propiedades de los objetos y el explorador de soluciones, además la barra de herramientas y las opciones del menú principal. Una diferencia en que ya no existe un formulario **Form1** sino una página **WebForm1** con una apariencia distinta. La página **WebForm1** contiene dos etiquetas: **Diseño** y **HTML**. El diseño es la presentación normal de la página y HTML es el lenguaje con el que se definen las páginas Web. Básicamente se trata de un conjunto de etiquetas que sirven para definir la forma en la que se presenta el texto y otros elementos de la página. Al pulsar la etiqueta **HTML** se puede observar el siguiente código:

```
<%@ Page Language="vb" AutoEventWireup="false" Codebehind="WebForm1.aspx.vb"
Inherits="PrimeraWeb.WebForm1"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>WebForm1</title>
    <meta name="GENERATOR" content="Microsoft Visual Studio .NET 7.1">
    <meta name="CODE_LANGUAGE" content="Visual Basic .NET 7.1">
    <meta name=vs_defaultClientScript content="JavaScript">
    <meta name=vs_targetSchema content="http://schemas.microsoft.com/intellisense/ie5">
  </head>
  <body MS_POSITIONING="GridLayout">
    <form id="Form1" method="post" runat="server">
    </form>
  </body>
</html>
```

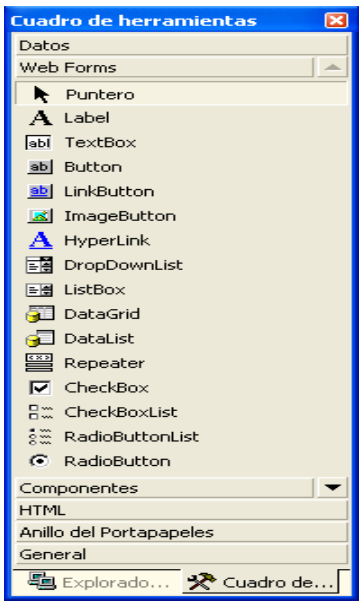
Cada vez que se le adiciona un objeto del cuadro de herramientas a la página se genera el código **HTML** necesario para representar dicho control. Por ejemplo si le adiciona un objeto **Label** a la página se genera el código **HTML** resaltado**:**

```
<%@ Page Language="vb" AutoEventWireup="false" Codebehind="WebForm1.aspx.vb"
Inherits="PrimeraWeb.WebForm1"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
        <HEAD>
                <title>WebForm1</title>
                <meta name="GENERATOR" content="Microsoft Visual Studio .NET 7.1">
                <meta name="CODE_LANGUAGE" content="Visual Basic .NET 7.1">
                <meta name="vs_defaultClientScript" content="JavaScript">
                <meta name="vs_targetSchema"
                content="http://schemas.microsoft.com/intellisense/ie5">
        </HEAD>
        <body MS_POSITIONING="GridLayout">
                <form id="Form1" method="post" runat="server">
        <asp:Label id="Label1" style="Z-INDEX: 101; LEFT: 64px; POSITION: absolute; TOP:
        56px" runat="server"
        Width="128px">Label</asp:Label>
                </form>
        </body>
</HTML>
```

Como se puede apreciar, este código que se genera es transparente para el usuario, por lo que no se ampliara más explicación, ya que no es la intención del libro hablar sobre HTML (consulte en Internet sobre este tema).

El cuadro de herramientas de la aplicación tiene dos nuevas fichas llamadas **Web Forms** y **HTML** que contiene elementos para trabajar con paginas ASP.NET. La ficha **Web Forms** contiene la mayoría de los elementos que contiene la ficha **Forms** de las aplicaciones **Windows**, La ficha **HTML** contiene elementos para trabajar con paginas HTML.

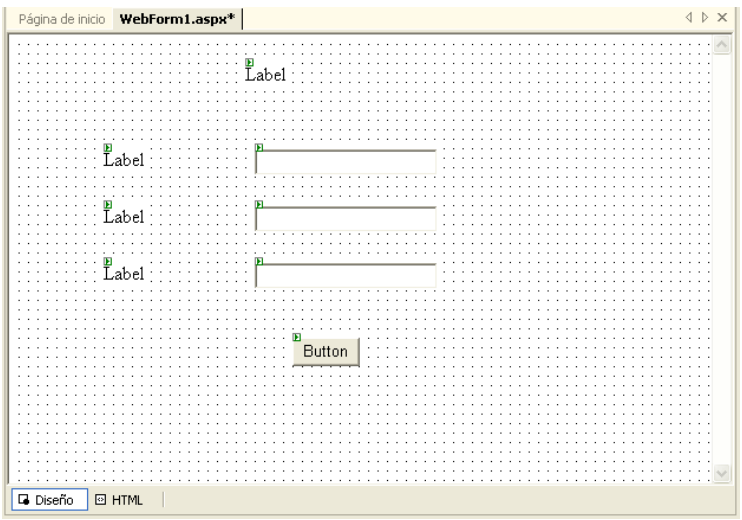
Figura 12.3 Cuadro de Herramientas de las aplicaciones Web ASP.NET.



- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas agregue al formulario: 1 Button, 3 TextBox y 4 Label. La figura muestra la interfaz de usuario:

Figura 12.4 Interfaz de usuario aplicación PrimeraWeb.



- **Establecer las propiedades de los objetos de la interfaz de usuario**

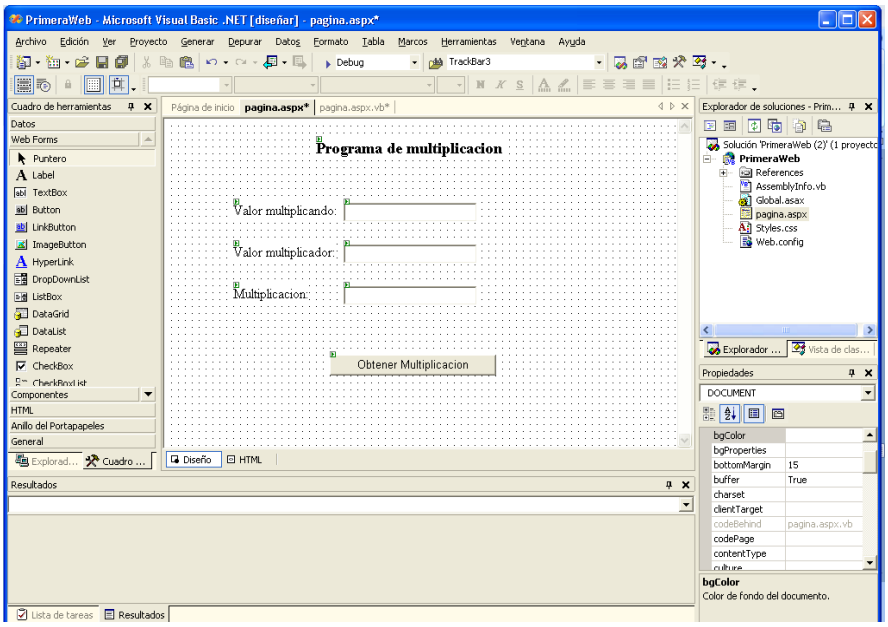
Después de colocar los controles u objetos en el formulario, establezca las siguientes propiedades a los controles:

Tabla 12.1 Propiedades de los controles de la aplicación PrimeraWeb.

Control	Propiedad	Valor
Label1	Id	lbltitulo
	Text	Programa de multiplicación
	Font/Bold	True
	Font/size	Large
Label2	Id	lblmultiplicando
	Text	Valor Multiplicando
Label3	Id	lblmultiplicador
	Text	Valor Multiplicador
Label4	Id	lblmultiplicacion
	Text	Multiplicacion
TextField1	Id	txtmultiplicando
	Text	En blanco
TextField2	Id	txtmultiplicador
	Text	En blanco
TextField3	Id	txtresultado
	Text	En blanco
Button1	Id	botón
	Text	Obtener Multiplicacion
Document	Title	Primera Web
WebForm1	Nombre del archivo	pagina

El formulario se visualizaría como muestra la siguiente figura:

Figura 12.5 Interfaz de usuario con controles modificados (PrimeraWeb).



- **Escribir código**

Seleccione el objeto **botón**, dé doble clic para abrir el editor del procedimiento `boton_Click` y escriba el siguiente código:

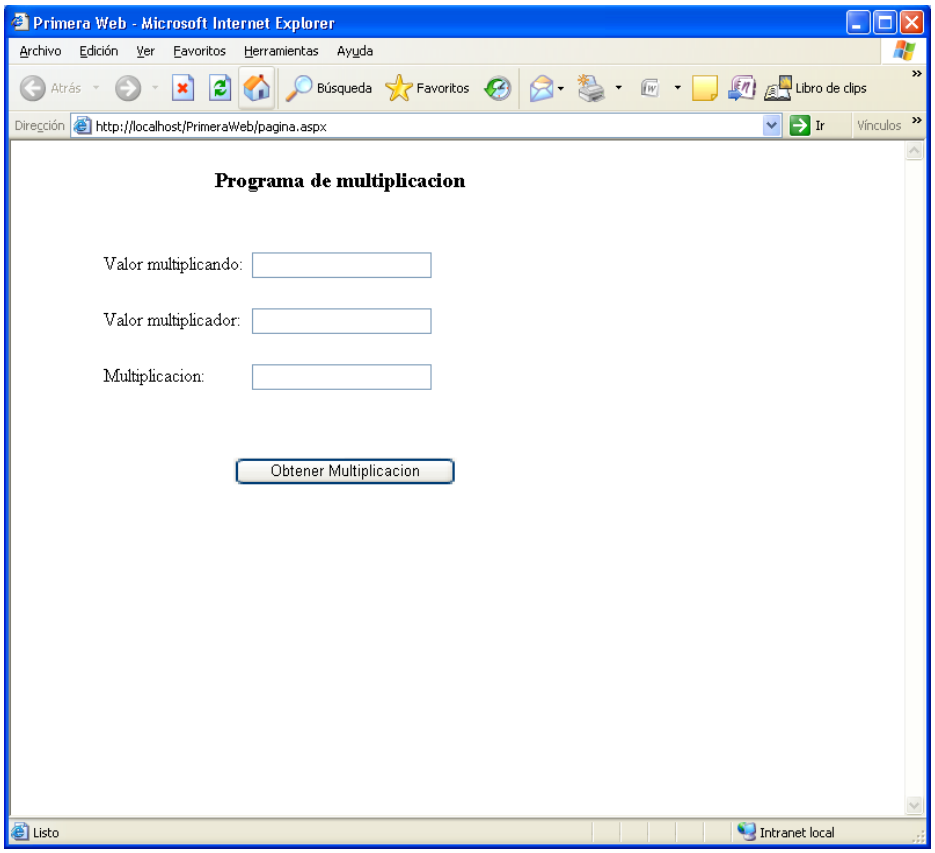
```
Dim vrmultiplicacion As Integer
vrmultiplicacion = Math.BigMul(txtmultiplicando.Text, txtmultiplicador.Text)
txtresultado.Text = vrmultiplicacion
```

Se declara una variable **vrmultiplicacion** de tipo **Integer**, a esta se le asigna el resultado que produce la función **Math.Bigmul** la cual retorna el producto de dos variables de tipo numérico. Por último se asigna el valor de **vrmultiplicacion** al objeto **txtresultado** por intermedio del método **Text.**

- **Ejecutar la aplicación**

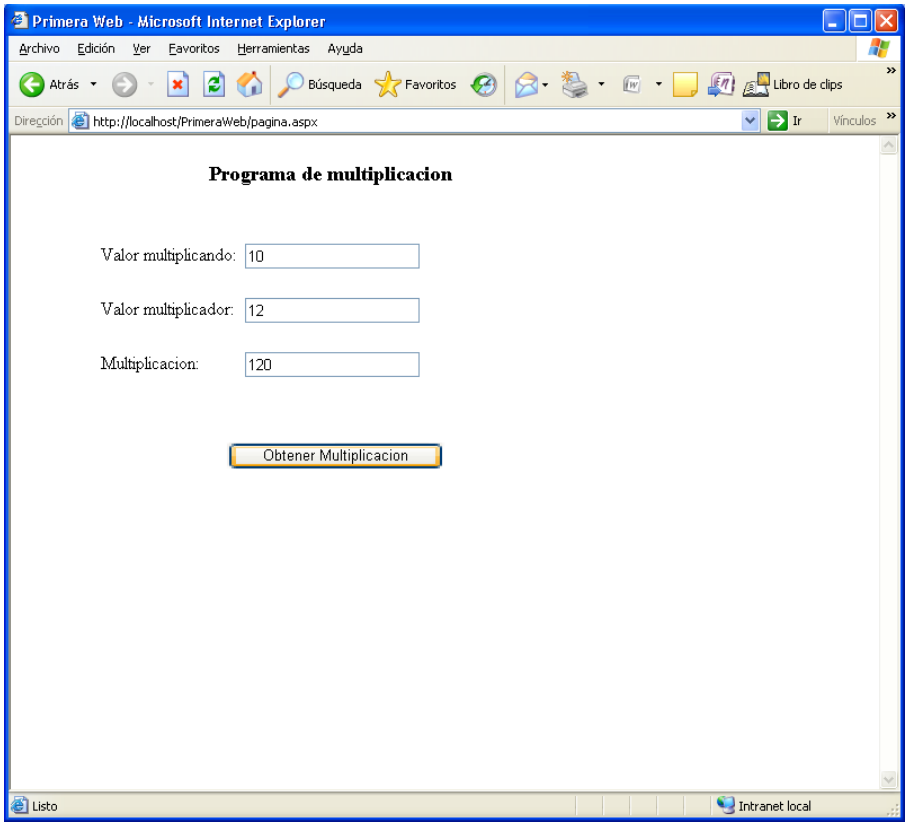
Al ejecutarse el proyecto en el entorno de desarrollo de Visual Basic.NET se muestra la siguiente figura:

Figura 12.6 Ejecución de la aplicación PrimeraWeb.



Al digitarse los valores 10 y 12 y pulsar el botón **Obtener Multiplicacion**, se visualizará la siguiente figura:

Figura 12.7 Ejecución de la aplicación PrimeraWeb.



12.2 Interfaz de usuario avanzada con Web ASP.NET

Hasta ahora se ha trabajo la interfaz de usuario con controles básicos del cuadro de herramientas como han sido los objetos **Label**, **TextBox** y **Button**. La mayoría de los controles del cuadro de herramientas se trabajaron en las aplicaciones Windows, en el capítulo 7. Nos centraremos en trabajar con los controles CheckBox, RadiButton, HyperLink, DropDownList para mirar algunas diferencias de programación.

12.2.1 Control CheckBox

Un control **CheckBox**  es una casilla de verificación que permite obtener dos estados: **verdadero** si esta activada o **falso** si esta desactivada. Para obtener el valor de verdadero o falso se hace a través de la propiedad **checked**.

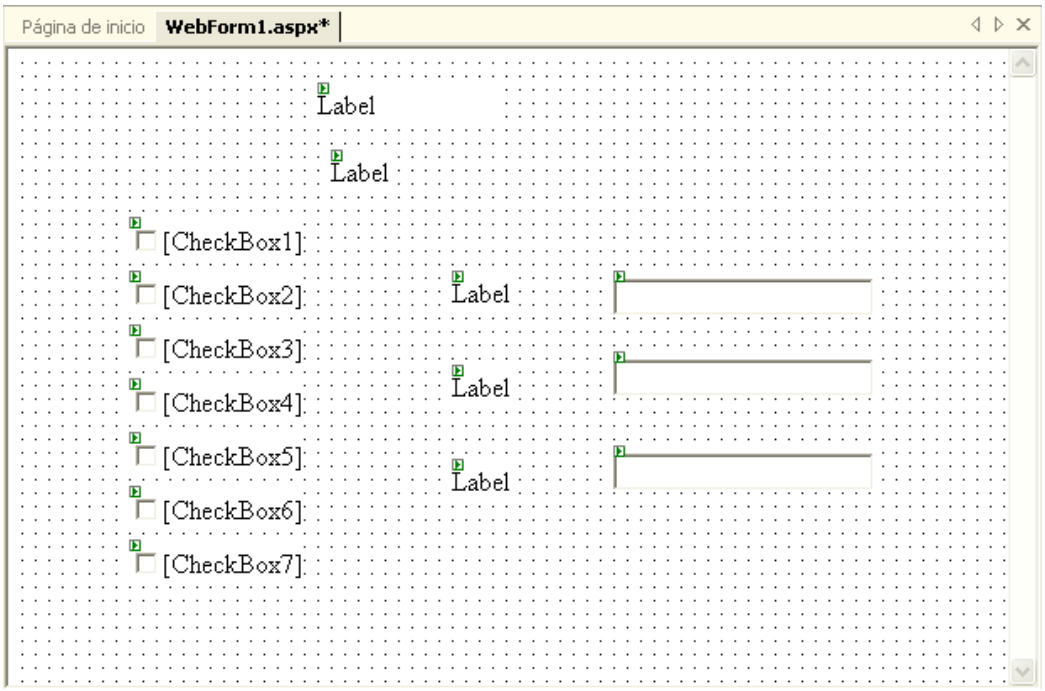
12.2.1.1 Ejemplo práctico control CheckBox

Realizar una aplicación que permita a un usuario realizar un pedido de comidas rápidas. El programa deberá permitir seleccionar los productos e ir visualizando el valor de los productos, el impuesto por ventas y neto a pagar.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control CheckBox y ubique siete controles en el formulario en la posición deseada. También seleccione cinco Label y tres TextBox y ubíquelos en el formulario. La figura 12.8 muestra la interfaz de usuario

Figura 12.8 Interfaz de usuario aplicación PedidoWeb.



- **Establecer las propiedades de los objetos de la interfaz de usuario**

Después de colocar los controles u objetos en el formulario, establezca las siguientes propiedades a los controles:

Tabla 12.2 Propiedades de los controles de la aplicación PedidoWeb.

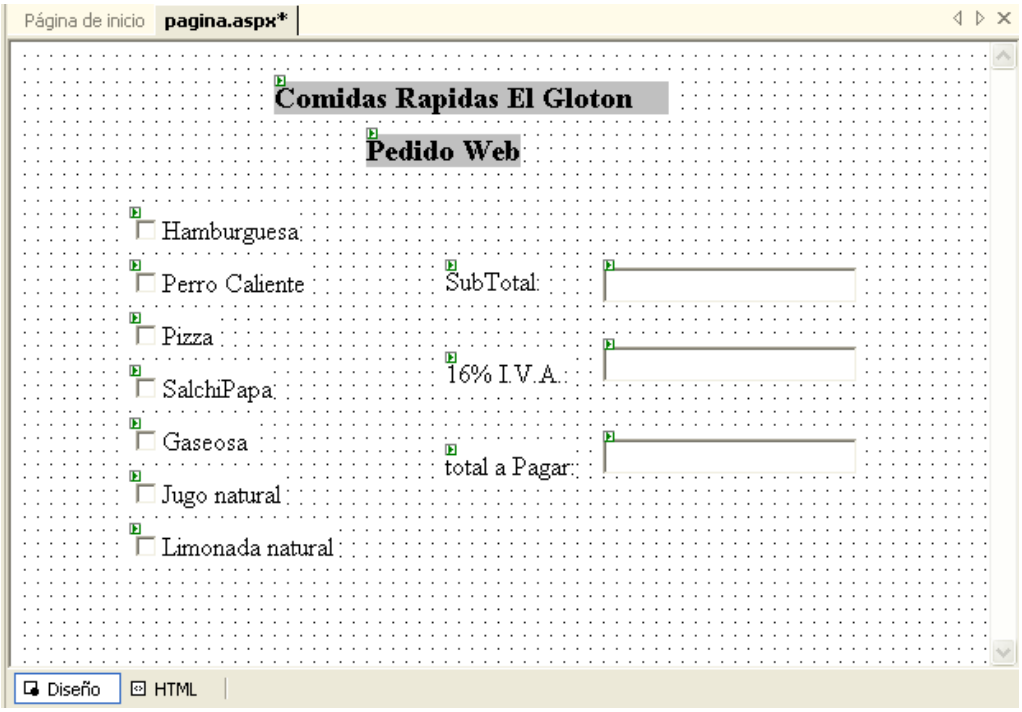
Control	Propiedad	Valor
Label1	Id	lbltitulo
	Text	Comidas rápidas El Gloton
	Backcolor	Gris
	Font	Bold
	Size	X- Large
Label2	Id	lblpedido
	Text	Valor del pedido
	Backcolor	Gris
	Font	Bold
	Size	X- Large
Label3	Id	lblsubtotal
	Text	Subtotal
Label4	Id	lbliva
	Text	16 % I.V.A.

Label5	Id	lbltotal
	Text	Total a Pagar
Checkbox1	AutoPostBack	True
	Id	validacionh
	Text	Hamburguesa
Checkbox2	AutoPostBack	True
	Id	validacionp
	Text	Perro Caliente
Checkbox3	AutoPostBack	True
	Id	validacionpi
	Text	Pizza
Checkbox4	AutoPostBack	True
	Id	validacions
	Text	Salchipapa
Checkbox5	AutoPostBack	True
	Id	validaciong
	Text	Gaseosa 350 cc
Checkbox6	AutoPostBack	True
	Id	validacionj
	Text	Jugo Natural
Checkbox7	AutoPostBack	True
	Id	Limonada Natural
	Text	validacionl
Textbox1	Id	txtsubtotal
	Text	En blanco
Textbox2	Id	txtiva
	Text	En blanco
Textbox3	Id	txttotal
	Text	En blanco
Document	Text	Comidas rápidas El gloton – Mi pedido Web
WebFrom1	Nombre del archivo	pagina

Es necesario que a todos los controles que van a sufrir modificaciones en tiempo de ejecución cambiarle el valor de la propiedad **AutoPostBack** a **True** para que devuelva datos automáticamente al servidor cuando se hace clic en el control. Como cada **CheckBox** puede estar seleccionado o no en tiempo de ejecución se le cambia el valor a dicha propiedad.

El formulario se visualizaría como muestra la siguiente figura:

Figura 12.9 Interfaz de usuario con los controles modificados (PedidoWeb).



- **Escribir código**

Declarar una variable global

```
public shared total as double =0.0
```

Se declara una variable pública global y compartida (**shared**) llamada **total** de tipo **Double**, la cual permitirá realizar operaciones cuando se seleccione o no un **CheckBox**. La variable es compartida porque se está trabajando en entorno Web y continuamente se está llamando al servidor.

Seleccione el objeto **validacionh**, dé doble clic para abrir el editor del procedimiento `validacionh_CheckedChanged` y escriba el siguiente código:

```
If (validacionh.Checked) Then
   total = total + 7000
Else
   total = total - 7000
End If
txtsubtotal.Text = total
txtiva.Text = total * 0.16
txttotal.Text = Format(Val(txtsubtotal.Text) + Val(txtiva.Text), "$#,#.00")
```

Al objeto **validacionh** se le valida si esta activado o no por intermedio de la estructura de decisión **if**. Si es verdad a la variable **total** se le asigna el valor que contiene más el nuevo valor; en este caso 7000, por el contrario (**Else**) se le asigna valor que contiene **total** menos el nuevo valor; en este caso -7000. A los objetos **TextBox** se

le asigna respectivamente: al objeto **txtsubtotal** el valor de la variable **total**, al objeto **txtiva** el valor de multiplicar la variable **total** por 0.16 y al objeto **txttotal** el valor de los objetos **txtsubtotal** más **txtiva**. Se utiliza la función **Val** () que convierte un texto en un valor numérico, como también se utiliza la función **Format** para darle el formato de pesos al valor asignado.

A cada objeto **CheckBox** se le asignarán las mismas operaciones anteriores, cambiándole el nombre del objeto **CheckBox** y los valores como se puede apreciar, a continuación:

Objeto validacionp

```
If (validacionp.Checked) Then
    total = total + 5000
Else
   total = total - 5000
End If
txtsubtotal.Text = total
txtiva.Text = total * 0.16
txttotal.Text = Format(Val(txtsubtotal.Text) + Val(txtiva.Text), "$#,#.00")
```

Objeto validacionpi

```
If (validacionpi.Checked) Then
    total = total + 3000
Else
    total = total - 3000
End If
txtsubtotal.Text = total
txtiva.Text = total * 0.16
txttotal.Text = Format(Val(txtsubtotal.Text) + Val(txtiva.Text), "$#,#.00")
```

Objeto validacions

```
If (validacions.Checked) Then
    total = total + 2500
Else
   total = total - 2500
End If
txtsubtotal.Text = total
txtiva.Text = total * 0.16
txttotal.Text = Format(Val(txtsubtotal.Text) + Val(txtiva.Text), "$#,#.00")
```

Objeto validaciong

```
If (validaciong.Checked) Then
    total = total + 1200
Else
    total = total - 1200
End If
txtsubtotal.Text = total
txtiva.Text = total * 0.16
txttotal.Text = Format(Val(txtsubtotal.Text) + Val(txtiva.Text), "$#,#.00")
```

Objeto validacionj

```
If (validacionj.Checked) Then
    total = total + 2000
Else
    total = total - 200
```

```
End If
txtsubtotal.Text = total
txtiva.Text = total * 0.16
txttotal.Text = Format(Val(txtsubtotal.Text) + Val(txtiva.Text), "$#,#.00")
```

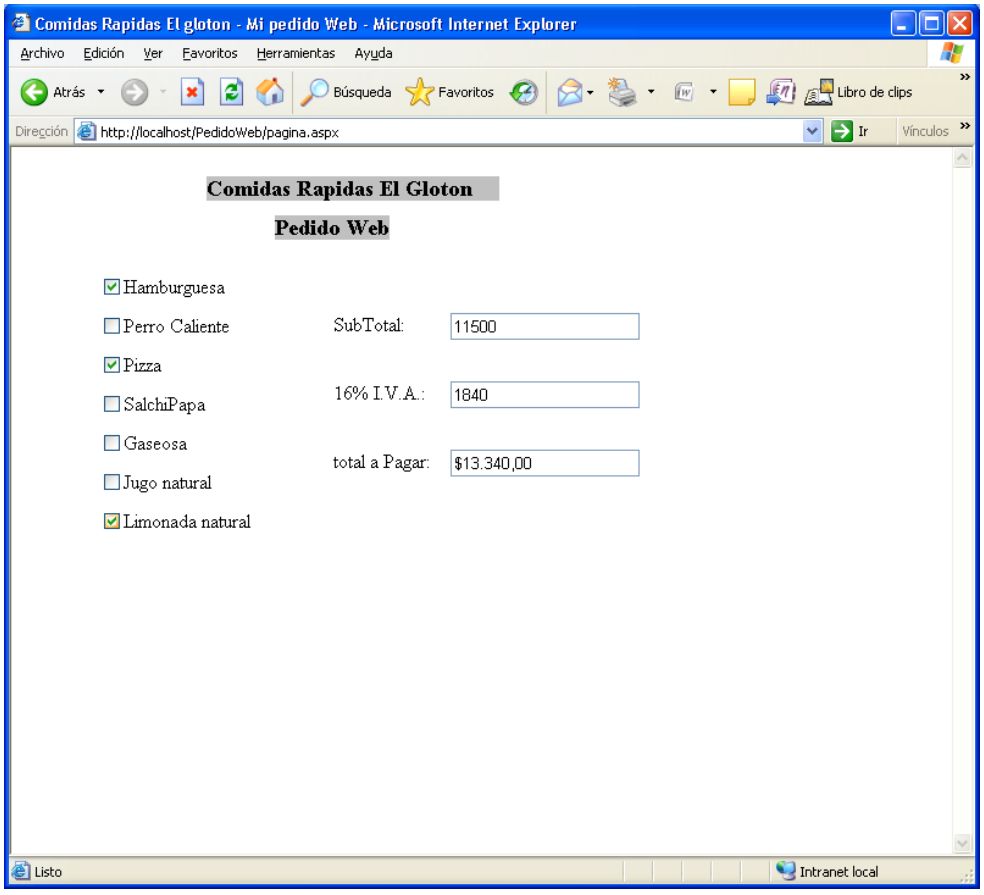
Objeto validacionl

```
If (validacionl.Checked) Then
    total = total + 1500
Else
    total = total - 1500
End If
txtsubtotal.Text = total
txtiva.Text = total * 0.16
txttotal.Text = Format(Val(txtsubtotal.Text) + Val(txtiva.Text), "$#,#.00")
```

- **Ejecutar la aplicación**

Al ejecutarse el proyecto en el entorno de desarrollo de Visual Basic.NET y seleccionar las opciones **Hamburguesa**, **Pizza** y **Limonada natural**, se visualiza:

Figura 12.10 Ejecución de la aplicación PedidoWeb.



12.2.2 Control RadioButton

Un control **RadioButton**  permite a un usuario escoger una alternativa entre varias alternativas. Al igual que el **CheckBox** por medio de la propiedad **Checked** se puede obtener dos estados: verdadero (**true**) o falso (**false**).

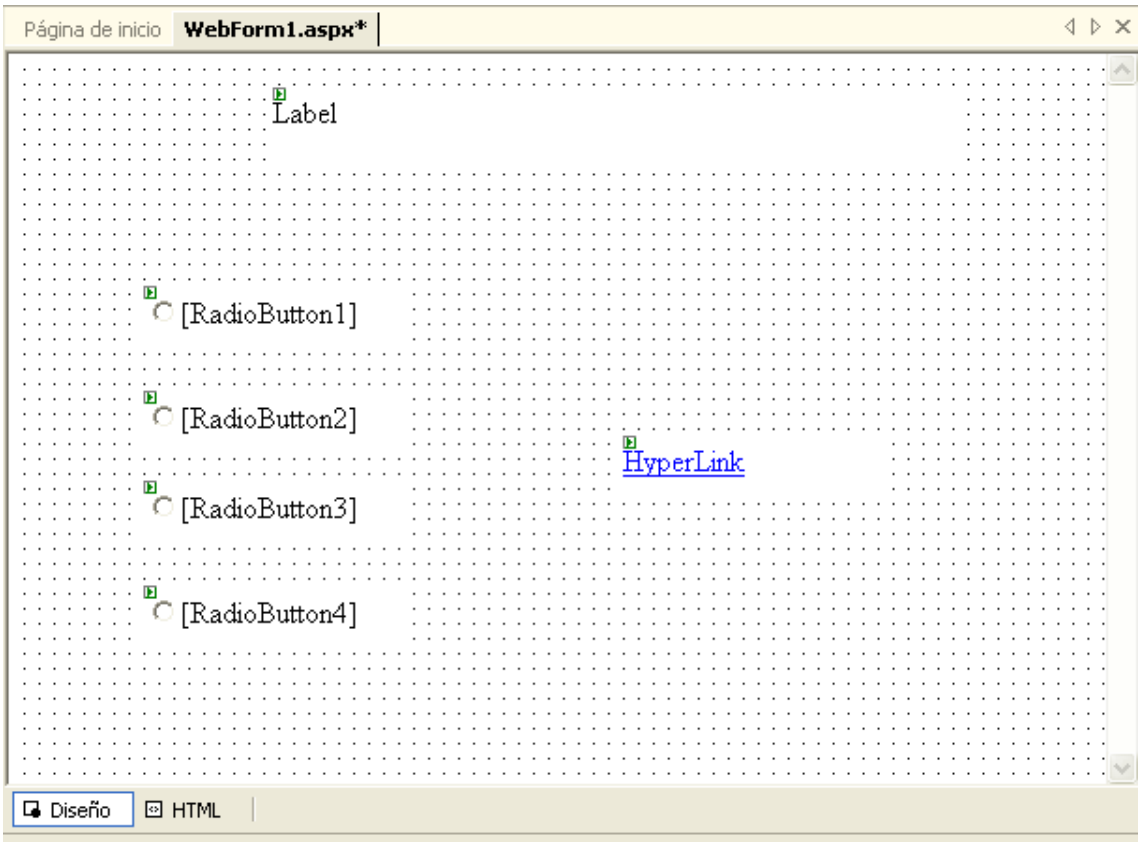
12.2.2.1 Ejemplo práctico control RadioButton

Realizar una aplicación que permita a un usuario seleccionar y abrir una página Web utilizando objetos RadioButton y HyperLink.

- **Crear la interfaz de usuario.**

Utilizando el cuadro de herramientas haga clic en el control **RadioButton** y ubique cuatro controles en el formulario en la posición deseada. También seleccione un control Label y un control HyperLink. La figura 12.11 muestra la interfaz de usuario

Figura 12.11 Interfaz de usuario aplicación SeleccionadorPaginaWeb.



- **Establecer las propiedades de los objetos de la interfaz de usuario**

Después de colocar los controles u objetos en el formulario, establezca las siguientes propiedades a los controles:

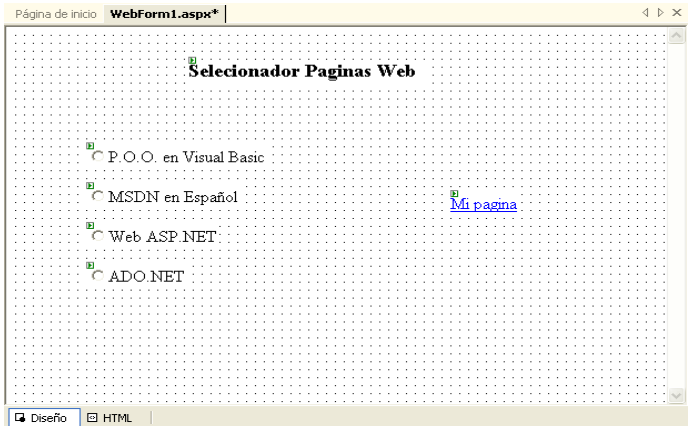
Tabla 12.3 Propiedades de los controles de la aplicación SeleccionadorpaginasWeb.

Control	Propiedad	Valor
Label1	ID	txttitulo
	Text	Seleccionador Página Web
	Backcolor	Gris
	Font	Bold
	Size	X- Large
HyperLink1	ID	vinculo
	Text	Mi pagina
RadioButton1	ID	opcionVB
	Text	P.O.O. en Visual Basic
	AutopostBack	True
	GroupName	alias
RadioButton2	ID	opcionMSDN
	Text	MSDN en Español
	AutopostBack	True
	GroupName	alias
RadioButton3	ID	opcionASP
	Text	Web ASP.NET
	AutopostBack	True
	GroupName	alias
RadioButton4	ID	opcionADO
	Text	ADO.NET
	AutopostBack	True
	GroupName	alias
WebForm1	Nombre del archivo	pagina
Document	Title	Seleccionador Paginas Web

Es necesario que a todos los controles que van a sufrir modificaciones en tiempo de ejecución cambiar el valor de la propiedad **AutoPostBack** a **True** para que devuelva datos automáticamente al servidor cuando se hace clic en el control. Como cada **RabioButton** puede estar seleccionado o no en tiempo de ejecución se le cambia el valor a dicha propiedad. También es necesario modificar la propiedad **GroupName** de los **RadioButton** por el valor de **alias** para que solamente una sola opción quede activa en un momento dato.

El formulario se visualizaría como muestra la siguiente figura:

Figura 12.12 Interfaz con controles modificados (SeleccionadorPaginaWeb).



- **Escribir código**

Seleccione el objeto **opcionVB**, dé doble clic para abrir el editor del procedimiento `opcionVB_CheckedChanged` y escriba el siguiente código:

```
vinculo.NavigateUrl = "http://msdn.microsoft.com/es-es/library/b86b82w0(VS.80).aspx"
vinculo.Text = vinculo.NavigateUrl
```

Al método **NavigateUrl** del objeto **HyperLink (**vinculo**)** se le asigna la dirección IP de la pagina Web que se quiere abrir. Al método **Text** del objeto **vinculo** se le asigna también la dirección de la pagina Web.

A cada objeto **RadioButton** se le asignarán las mismas operaciones anteriores, cambiándole el nombre de la página Web que se quiere abrir, como se puede apreciar, a continuación:

Objeto opcionMSDN

```
vinculo.NavigateUrl = "http://www.msdn.com"
vinculo.Text = vinculo.NavigateUrl
```

Objeto opcionASP

```
vinculo.NavigateUrl = "http://www.asp.net/ES/"
vinculo.Text = vinculo.NavigateUrl
```

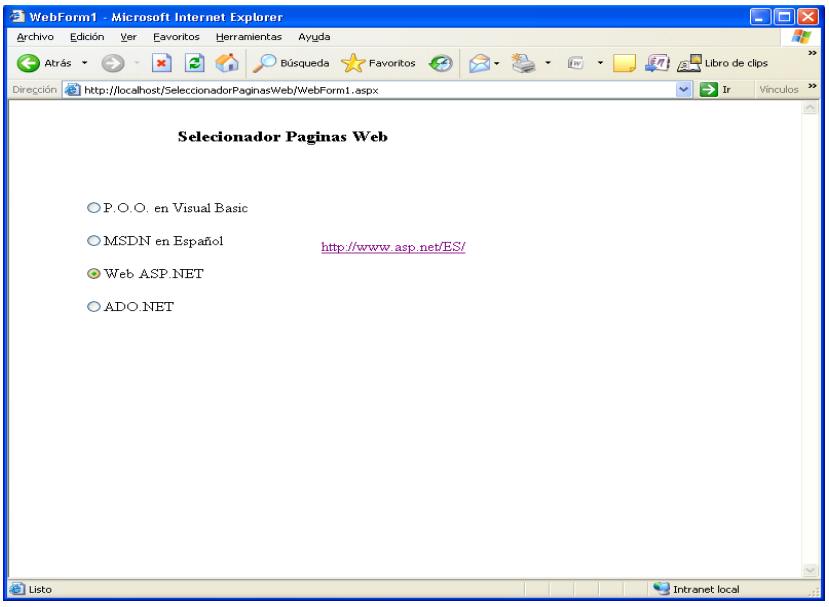
Objeto opcionADO

```
vinculo.NavigateUrl = "http://msdn.microsoft.com/es-es/library/e80y5yhx(VS.80).aspx"
vinculo.Text = vinculo.NavigateUrl
```

- **Ejecutar la aplicación**

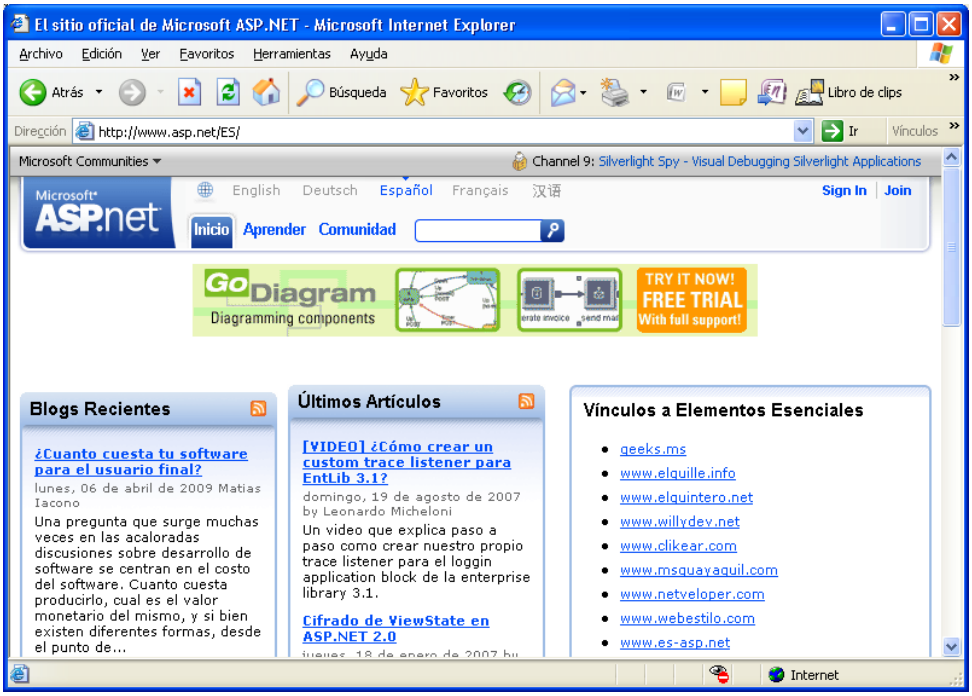
Al ejecutarse el proyecto en el entorno de desarrollo de Visual Basic.NET y seleccionar las opciones Web ASP.NET, se visualiza:

Figura 12.13 Ejecución de la aplicación SeleccionadorPaginaWeb.



Al pulsar la opción Web ASP.NET se visualizará la figura 12.14:

Figura 12.14 Ejecución opción Web ASP.NET.



12.2.3 Control DropDownList

Un control **DropDownList**  permite definir una lista de elementos donde el usuario puede seleccionar un elemento o más. Al momento de la ejecución del programa, toda la lista de elementos estarán a la vista del usuario para que este los seleccione. Para seleccionar un elemento deberá pulsar la flecha que está al lado izquierdo del encabezado.

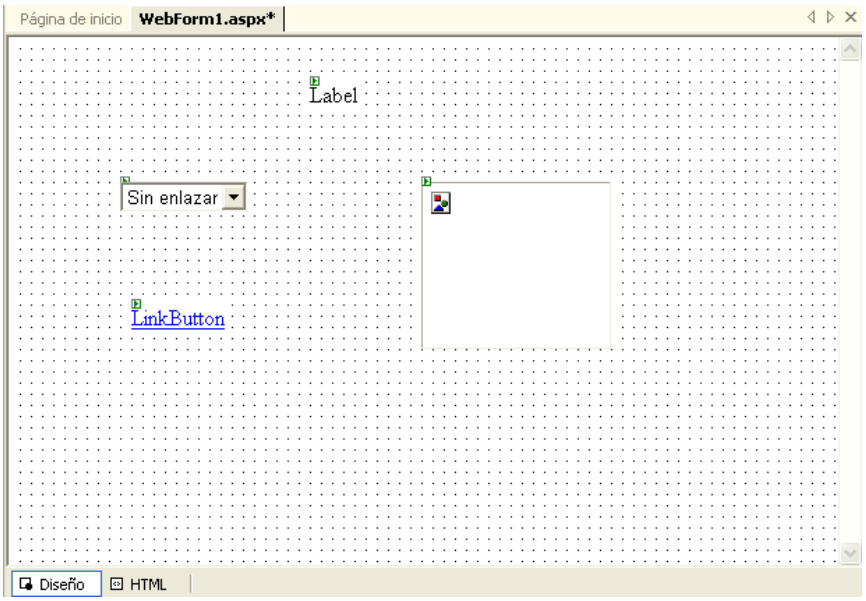
12.2.3.1 Ejemplo práctico control DropDownList

Realizar una aplicación que permita a un usuario seleccionar un medio de transporte de un objeto DropDownList, se deberá mostrar en un objeto **ButtonLink** el nombre del objeto seleccionado y al pulsar el botón se visualice la figura correspondiente.

- **Crear la interfaz de usuario.**

Utilizando el cuadro de herramientas haga clic en el control DropDownList y ubíquelo en el formulario en la posición deseada. También seleccione un control Label, un ButtonLink y un Image. La figura 12.14, muestra la interfaz de usuario:

Figura 12.15 Interfaz de usuario aplicación SeleccionarImagen.



- **Establecer las propiedades de los objetos de la interfaz de usuario**

Después de colocar los controles u objetos en el formulario establezca las siguientes propiedades a los controles:

Tabla 12.4 Propiedades de los controles de la aplicación SeleccionarImagen.

Control	Propiedad	Valor
Label1	ID	titulo
	Text	Medios de Transporte
	Backcolor	Gris
	Font	Bold
	Size	Large
LinkButton1	ID	vinculo
	Text	botón de Vinculo
Dropdownlist1	ID	lista
	Ítems	Carro, Avion, Cohete, Bicicleta
	AutoPostback	true
Image1	ID	imagen
Document	Title	Seleccionador de imágenes
WebForm1	Nombre del archivo	pagina

Al control **DropDownList** puede agregársele elementos de dos formas: la primera utilizando la propiedad **Ítems** y la segunda por medio de código desde el procedimiento **Load** del formulario. Se utilizara la segunda forma; seleccione el control **DropDownList** y haga clic sobre la propiedad **Ítems**, en la ventana pulse el botón

Agregar y escriba el nombre de cuatro medios de transporte, uno por cada línea:

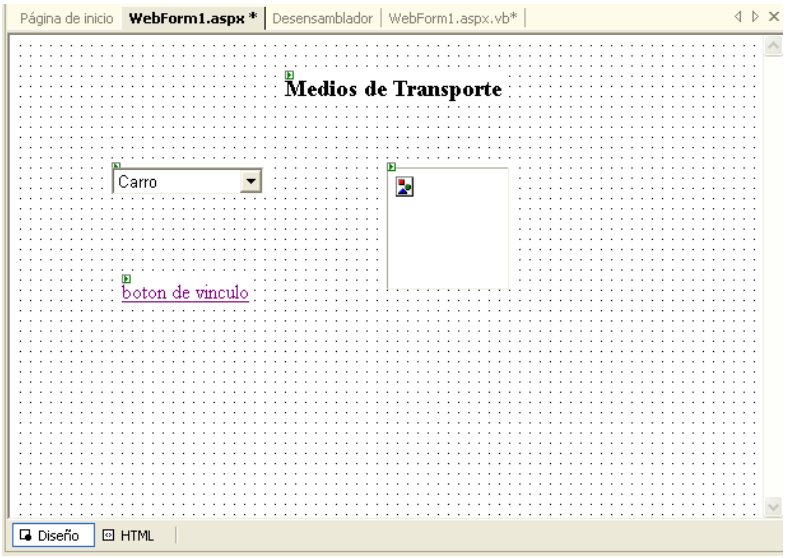
Carro
Avion
Bicicleta
Cohete

Y por último dé clic en el botón **Aceptar** para que los elementos queden agregados al control. Se visualizará la siguiente figura:

Figura 12.16 Editor con los elementos agregados en un DropDownList.

El formulario se visualizaría como muestra la siguiente figura:

Figura 12.17 Interfaz con controles modificados (SeleccionarImagen).



- **Escribir código**

Seleccione el objeto **lista**, dé doble clic para abrir el editor del procedimiento `lista_CheckedChanged` y escriba el siguiente código:

```
vinculo.Text = lista.SelectedItem.Text
```

Al método **Text** del objeto **ButtonLink (**vinculo**)** se le asigna el texto seleccionado del objeto **lista** por intermedio de la propiedad **SelectedItem**.**Text**.

Seleccione el objeto **vínculo**, dé doble clic para abrir el editor del procedimiento `vinculo_Click` y escriba el siguiente código:

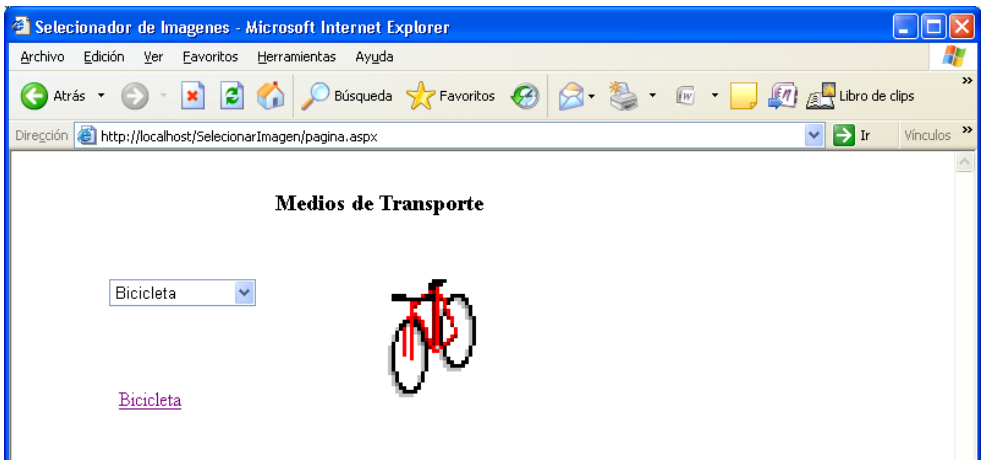
```
If lista.SelectedItem.Text = "Carro" Then
        imagen.ImageUrl = "file:///C:\Archivos de programa\Microsoft Visual Studio
        .NET 2003\Common7\Graphics\icons\Industry\cars.ICO"
End If
If lista.SelectedItem.Text = "Avion" Then
        imagen.ImageUrl = "file:///C:\Archivos de programa\Microsoft Visual Studio
        .NET 2003\Common7\Graphics\icons\Industry\Plane.ICO"
End If
 If lista.SelectedItem.Text = "Bicicleta" Then
        imagen.ImageUrl = "file:///C:\Archivos de programa\Microsoft Visual Studio
        .NET 2003\Common7\Graphics\icons\Industry\bicycle.ICO"
End If
If lista.SelectedItem.Text = "Cohete" Then
        imagen.ImageUrl = "file:///C:\Archivos de programa\Microsoft Visual Studio .NET
        2003\Common7\Graphics\icons\Industry\rocket.ICO"
End If
```

Al objeto **lista** se le valida que texto fue seleccionado por intermedio de la estructura de decisión **If**. Utilizando el método **ImageUrl** se le asigna la ruta donde está la imagen correspondiente.

Ejecutar la aplicación

Al ejecutarse el proyecto en el entorno de desarrollo de Visual Basic.NET y seleccionar la opción **Bicicleta** y pulsar el objeto **ButtonLink**, se visualiza:

Figura 12.18 Ejecución de la aplicación SeleccionarImagen.



12.3 Controles de Validación

Los controles de validación son aquellos que sirven para validar la información escrita por un usuario, es decir, que este en el formato correcto o que no dejen un control vacío. Los tipos de controles de validación son:

- **RequiredFieldValidator**: se utiliza para validar que un campo de texto contenga información no puede ser un vacío.
- **RangeValidator**: se utiliza para comparar que lo escrito por un usuario este en un valor validado por un operador de comparación (<,<,=,etc.).
- **CompareValidator**: se utiliza para comparar que el valor de un campo de texto con un valor o constante predefinida.
- **RegularExpressionValidator** : se utiliza para validar que lo escrito por un usuario sea un valor definido por una expresión regular (direcciones de correo, números de teléfonos, etc.).
- **CustomValidator**: se utiliza para validar que lo escrito por un usuario sea un valor que previamente se ha definido por el usuario.
- **ValidationSummary:** se utiliza para mostrar mensajes de errores enviados por los controles de validación.

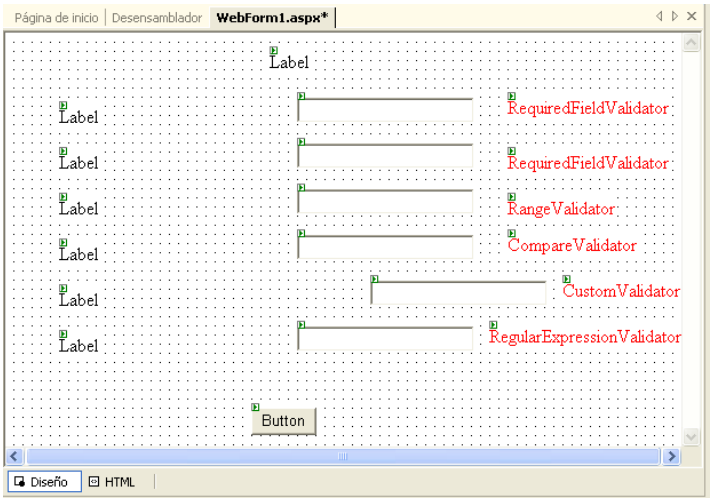
12.3.1 Ejemplo práctico controles de validación

Realizar una aplicación que permita capturar la siguiente información: nombres y apellidos del usuario (se debe validar que en estos campo contengan información), la edad (validar la edad entre 1 y 100), su nivel de ingles (), su estado civil () y un correo electrónico.

- **Crear la interfaz de usuario**

Utilizando el cuadro de herramientas haga clic en el control Label, seleccione siete controles. También seleccione seis controles TextBox, un Button, además seleccione los controles de validación de acuerdo a la figura 14.18:

Figura 12.19 Interfaz de usuario aplicación ControlesdeValidacion.



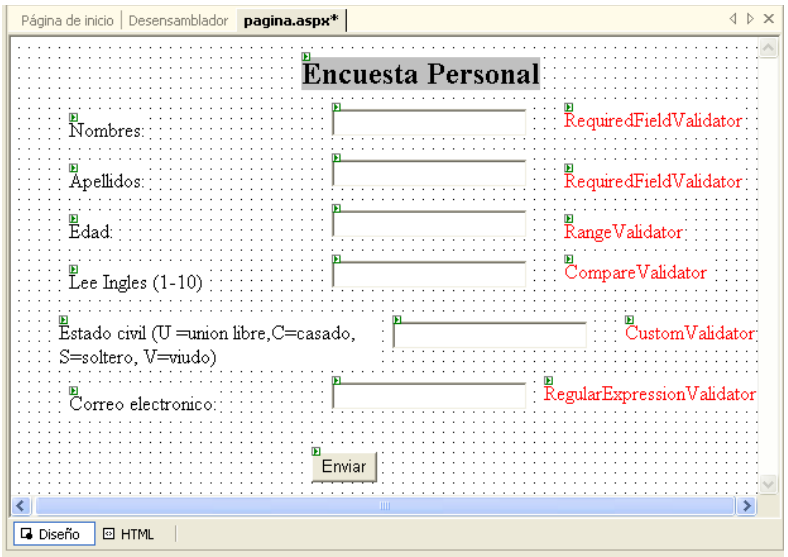
Modifique las propiedades de los controles según la siguiente tabla:

Tabla 12.5 Propiedades de los controles de la aplicación ControlesdeValidacion.

Control	Propiedad	Valor
Label1	ID	titulo
	Text	Encuesta Personal
	Backcolor	Gris
	Font	Bold
	Size	X- Large
Label2	ID	lblnombres
	Text	Nombres:
Label3	ID	lblapellidos
	Text	Apellidos:
Label4	ID	lbledad
	Text	Edad:
Label5	ID	lblingles
	Text	Lee ingles (1-10)
Label6	ID	lblestado
	Text	Estado civil (U=union libre; C casado; S soltero; V viudo)
Label7	ID	lblcorreo
	Text	Correo electrónico :
TextBox1	ID	txtnombres
	Text	En blanco
TextBox2	ID	txtapellidos
	Text	En blanco
TextBox3	ID	txtedad
	Text	En blanco
TextBox4	ID	txtingles
	Text	En blanco
TextBox5	ID	txtestado
	Text	En blanco
TextBox6	ID	txtcorreo
	Text	En blanco
WebForm1	Nombre de archivo	pagina
Document	Title	Controles de validación
Button1	ID	botón
	Text	Enviar

El formulario se visualizaría como muestra la siguiente figura:

Figura 12.20 Interfaz con controles modificados (ControlesdeValidacion).



- **Escribir código**

Dé doble clic sobre la página Web para abrir el editor del procedimiento `page_load` y escriba el siguiente código:

```
If (Not Page.IsPostBack) Then
    RequiredFieldValidator1.ControlToValidate = "txtnombres"
    RequiredFieldValidator1.ErrorMessage = "Debe digitar sus nombres"
    RequiredFieldValidator2.ControlToValidate = "txtapellidos"
    RequiredFieldValidator2.ErrorMessage = "Debe digitar sus apellidos"
    RangeValidator1.Type = ValidationDataType.Integer
    RangeValidator1.MinimumValue = 1
    RangeValidator1.MaximumValue = 100
    RangeValidator1.ErrorMessage = ("Digite una edad entre 1 y 100")
    RangeValidator1.ControlToValidate = "txtedad"
    CompareValidator1.Type = ValidationDataType.Integer
    CompareValidator1.ValueToCompare = 0
    CompareValidator1.ControlToValidate = "txtingles"
    CompareValidator1.Operator = ValidationCompareOperator.GreaterThan
    CompareValidator1.ErrorMessage = ("Digite un número mayor que 1")
    CustomValidator1.ControlToValidate = "txtestado"
    CustomValidator1.ErrorMessage = "Digite U,C,S o V"
    RegularExpressionValidator1.ControlToValidate = "txtcorreo"
    RegularExpressionValidator1.ValidationExpression = "\w+([-+.]\w+)*@\w+([-
                                                        .]\w+)*\.\w+([-.]\w+)*"
    RegularExpressionValidator1.ErrorMessage = "El formato del correo
                                        Electronico es: nombre@sitioweb.com
End If
```

Utilizando la sentencia de toma de decisiones **If** se valida cuando se envía los datos al servidor para realizar las validaciones correspondientes en cada control. Los métodos que se utilizan son:

- **ControlValidate**: es el objeto que se va a validar
- **ErrorMessage**: El mensaje que se imprimirá al ocurrir el error.
- **ValidationType**: es el tipo de dato a validar
- **MinimumValue**: es el valor mínimo del control de validación
- **MaximumValue**: es el valor máximo del control de validación
- **ValueToCompare**: es el valor mínimo que se puede escribir
- **Operator**: es el valor de comparación a validar
- **ValidationExpression**: es la expresión regular que se valida.

Seleccione el objeto **botón**, dé doble clic para abrir el editor del procedimiento `boton_Click` y escriba el siguiente código:

```
CustomValidator1.Validate()
RequiredFieldValidator1.Validate()
RequiredFieldValidator2.Validate()
RangeValidator1.Validate()
CompareValidator1.Validate()
RegularExpressionValidator1.Validate()
```

Al seleccionar el objeto **botón** se valida cada objeto de validación referente a la información que se ha escrito en cada uno de los campos de texto con el método **Validate**().

Seleccione el objeto **CustomValidator1**, dé doble clic para abrir el editor del procedimiento `CustomValidator1_ServerValidate` y escriba el siguiente código:

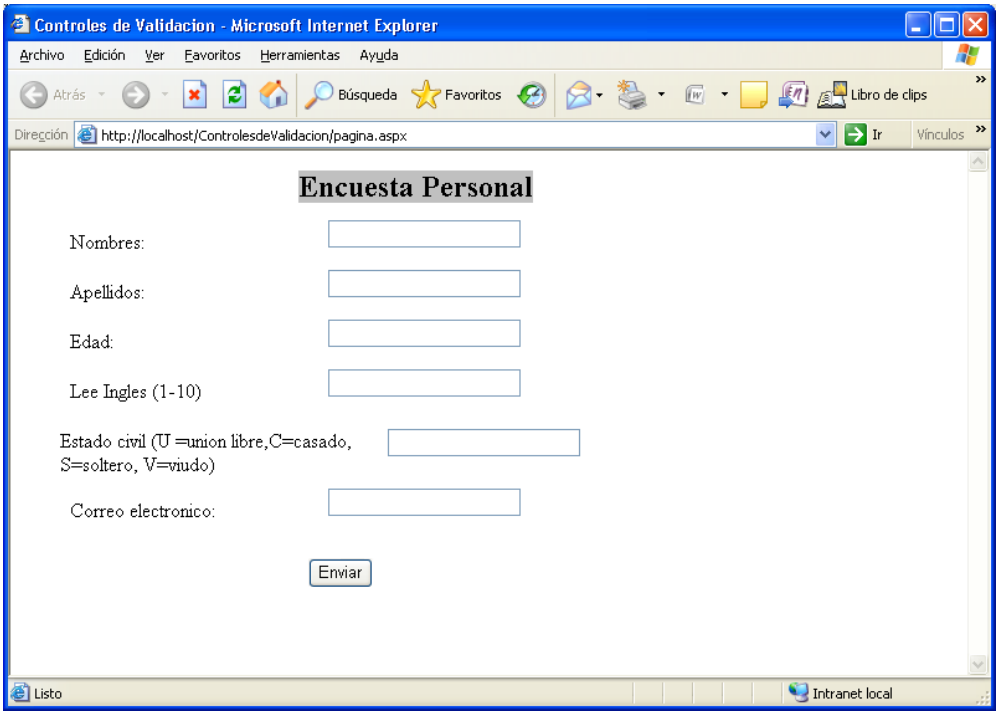
```
If (args.Value = "U") Then
        args.IsValid = True
ElseIf (args.Value = "C") Then
        args.IsValid = True
ElseIf (args.Value = "S") Then
        args.IsValid = True
ElseIf (args.Value = "V") Then
        args.IsValid = True
Else
       args.IsValid = False
End If
```

Al digitar un valor en el campo de texto **txtestado,** este es validado con la estructura de decisión **If** para determinar que su contenido sea un valor especificado por el control de validación **CustomValidator**.

- **Ejecutar la aplicación**

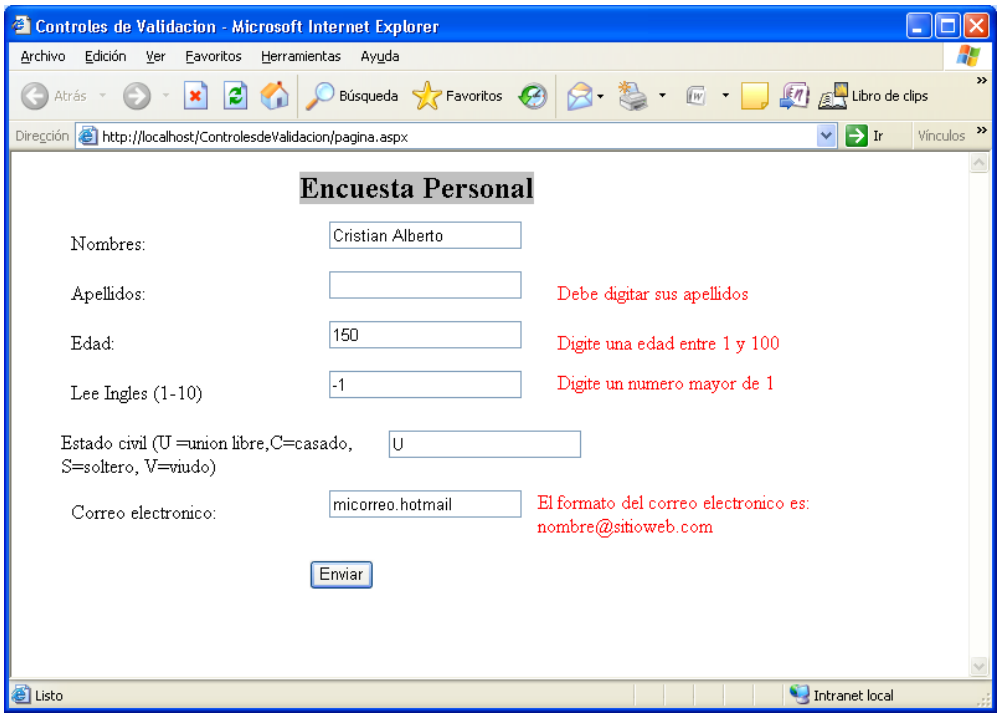
Al ejecutarse el proyecto en el entorno de desarrollo de Visual Basic.NET, se visualiza la figura 12.20:

Figura 12.21 Ejecución de la aplicación ControlesdeValidacion.



Si se digita la siguiente información en los campos correspondientes: Cristian Alberto, “ “, 150, -1, u, micorreo.hotmail.com, se visualizará la figura 12.21:

Figura 12.22 Ejecución con información de la aplicación ControlesdeValidacion.



13. ACCESO A BASES DE DATOS CON ASP.NET

En las aplicaciones Web a veces es necesario mostrar información de una base de datos. En el capítulo 11 se realizo la conexión, consulta y la operación de registros con una base de datos utilizando Windows Forms. El acceso a una base de datos desde un formulario Web es algo similar y también se pueden realizan las mismas operaciones. Se puede escribir código para el acceso a la base de datos utilizando clases del espacio de nombres **System.Data** (normalmente denominado ADO.NET).

13.1 Controles de origen de datos

Los controles de origen de datos son controles ASP.NET que administran las tareas de conexión a una base de datos. Estos no representan ningúna interfaz de usuario, sino que actúan como intermediarios entre los datos y los demás controles de la página Web ASP.NET. Dichos controles habilitan un amplio conjunto de funciones para recuperar y modificar datos, entre las que se incluyen la consulta, la actualización, la eliminación y la inserción. ASP.NET incluye los siguientes:

Tabla 13.1 Controles de datos de ASP.NET.

Control	descripción
OleDbDataAdapter	Adaptador de datos para una base de datos Access.
SqlDbDataAdapter	Adaptador de datos para una base de datos SQL.
OracleDbDataAdapter	Adaptador de datos para una base de datos Oracle.
DataSet	Conjunto de datos.
DataView	Vista de datos.

13.1.1 Ejemplo práctico bases de datos con ASP.NET

Hacer una aplicación Web que permita realizar la conexión a una base de datos realizada en Access para visualizar los registros de una tabla.

Para implementar la aplicación se deben realizar los siguientes pasos:

1. Crear una Base de Datos en Access llamada **visualnet** y guárdela dentro de la carpeta que desee. Dentro de la base de datos crear una tabla llamada **Clientes,** con la siguiente estructura:

Tabla 13.2 Estructura de la tabla Clientes.

Nombre del campo	Tipo de Dato	Longitud
Identificación	Texto	13
Nombres	Texto	25
Apellidos	Texto	25
Dirección	Texto	25
Teléfono	Texto	15
Ciudad	Texto	20
Crédito	numérico	

La tabla con 5 registros quedará de la siguiente forma:

Figura 13.1 Tabla clientes con registros.

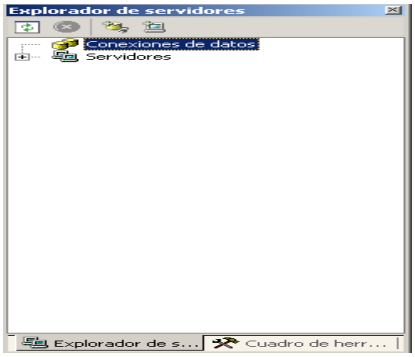
clientes

identificacic	nombres	apellidos	direccion	telefono	ciudad	credito
1	carlos alberto	vanegas	cra 24 # 24 -24	123456	bogota	100000
2	rosa maria	cetina	cra 25 # 24 -24	654789	cali	20000
3	cristian alberto	vanegas cetina	cra 26 # 24 -24	987456	medellin	30000
4	angelita	vanegas rodrig	cra 27 # 24 -24	321456	bogota	50000
5	camilo	vanegas rodrig	cra 28 # 24 -24	325814	cali	140000

2. Hacer la conexión a la base de datos

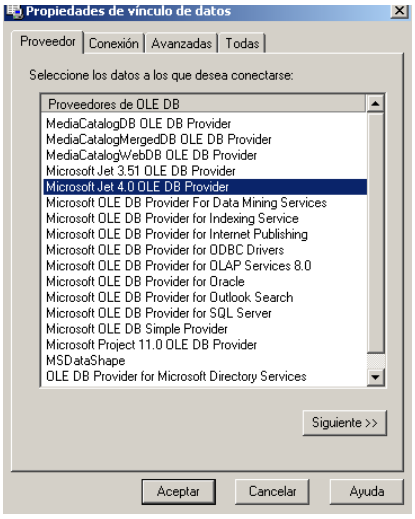
Para realizar una conexión a una base de datos existente seleccione el **explorador de servidores** del menú **Ver**, donde se visualizará el cuadro de diálogo **explorador de servidores,** como se aprecia en la siguiente figura:

Figura 13.2 Explorador de Servidores aplicación BasesdeDatosWeb.



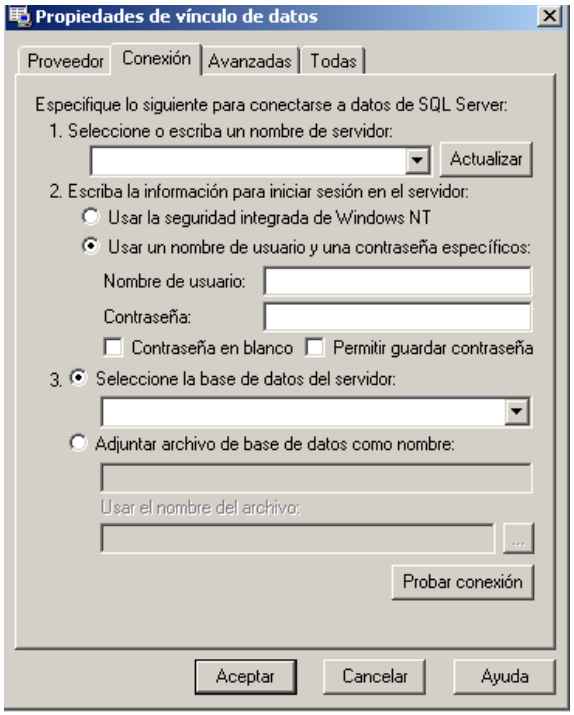
Seleccione el icono **conectar con bases de datos**  para iniciar la conexión a la base de datos y visualizar el cuadro de diálogo **Propiedades de vínculos de datos,** como se muestra en la siguiente figura:

Figura 13.3 Propiedades de vínculo de datos.



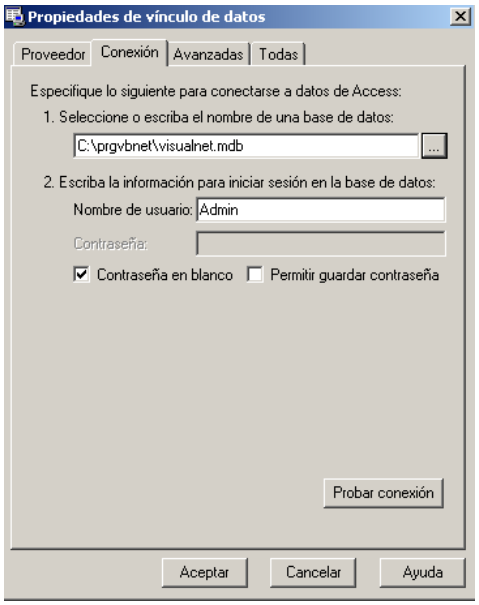
Pulse sobre la pestaña llamada **Proveedor** para seleccionar el proveedor de la base de datos. En este caso escoja **Microsoft jet 4.0 OLE DB Provider** y pulse el botón **Siguiente >>.** Se visualizará la figura 13.4:

Figura 13.4 Propiedades de vínculo de datos (pestaña conexión).



Se visualizará la pestaña **Conexión** del cuadro de diálogo **Propiedades de vínculo de datos**, allí en el paso 1, seleccione la base de datos que previamente se había creado, como se muestra en la siguiente figura:

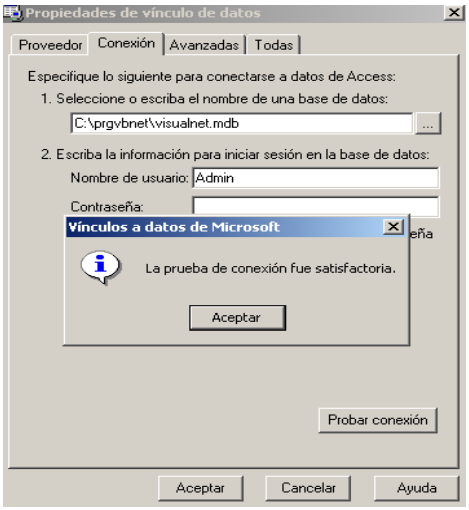
Figura 13.5 Selección de la base de datos



Si desea crear una contraseña desactive el cuadro de verificación **contraseña en blanco** y automáticamente el campo **contraseña** se habilitara para que el usuario digite la contraseña que desee. El paso 2 es opcional.

Ahora también se puede verificar la conexión pulsando el botón **Probar conexión**. Si la conexión ha sido exitosa se mostrará la siguiente figura:

Figura 13.6 Prueba de conexión.

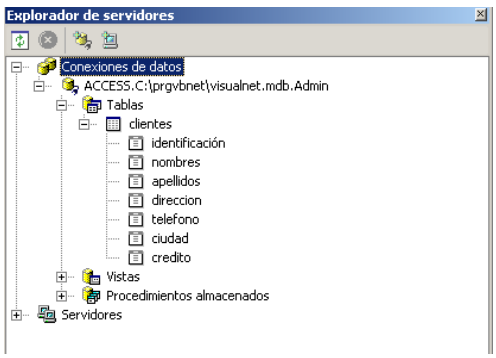


En caso de que la conexión no fuese exitosa, se debe iniciar nuevamente con cada uno de los pasos realizados hasta el momento para resolver el problema. Uno de las posibles fallas en la conexión es que la base de datos está abierta en el momento de la conexión, proceda a cerrarla y realice nuevamente la conexión.

Para continuar pulse el botón **Aceptar** de la prueba de conexión y luego nuevamente pulse el botón **Aceptar** del cuadro de diálogo **Propiedades de vínculo de datos** para terminar la conexión.

En este momento se puede pulsar el signo (+) a la izquierda de la leyenda **conexiones de datos** del cuadro de diálogo **Explorador de servidores**para visualizar la conexión a la base de datos **visualnet**. Cada vez que pulse el signo (+) se podrá apreciar la estructura de la base de datos, como se muestra en la siguiente figura:

Figura 13.7 Exploración de la base de datos.

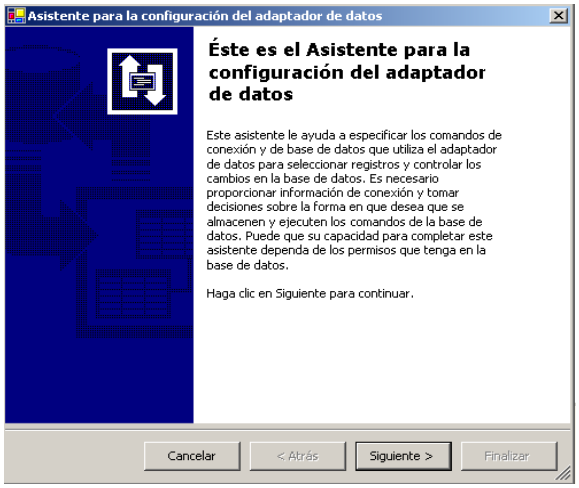


3. Seleccionar registros de un origen de datos

Para consultar los datos que se quieren mostrar de una base de datos se debe buscar la ficha **Datos** del cuadro de herramientas y seleccionar el objeto **OleDbDataAdapter** que sirve para manipular datos de una base de datos de Access. Es de aclarar, que de acuerdo a las base de datos se debe escoger el adaptador, como por ejemplo, si se está trabajando con **SqlServer** el adaptador que se debería escoger seria **SqlDbDataAdapter**.

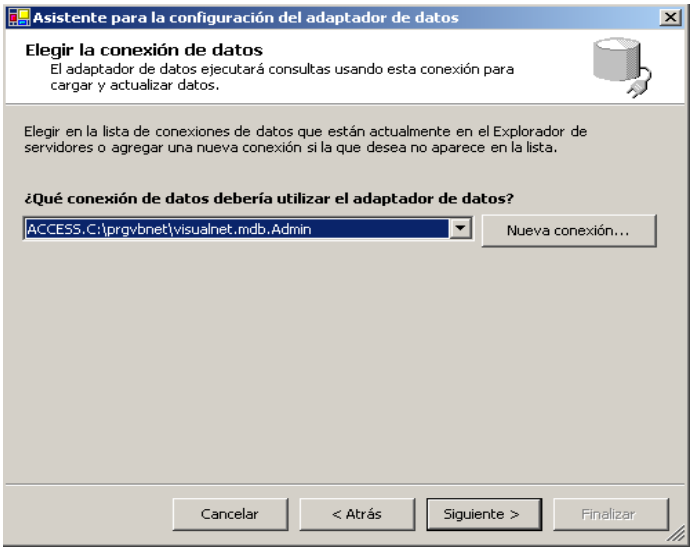
Dé doble clic sobre el objeto **OleDbDataAdapter** para visualizar el **Asistente para la configuración del adaptador de datos.**

Figura 13.8 Asistente para configuración de datos.



Este asistente servirá para especificar los comandos de conexión y la base de datos que el adaptador utilizará para seleccionar los registros y controlar los cambios en la base de datos. Haga clic en el botón **Siguiente**> para visualizar la ventana que sirve para elegir la conexión de datos, como se puede apreciar, en la figura 13.9:

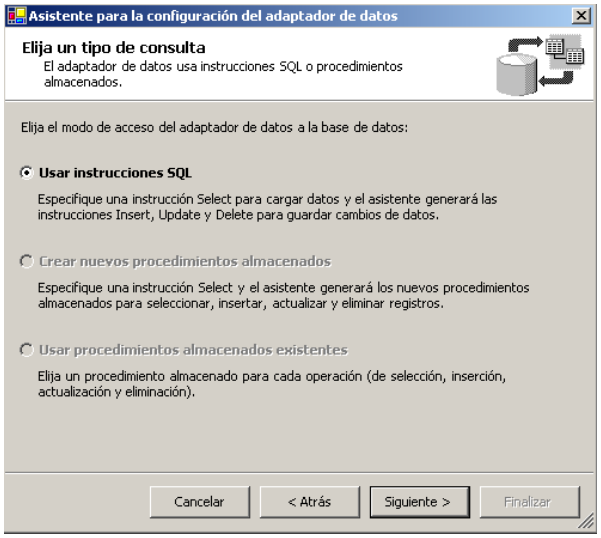
Figura 13.9 Elegir conexión de datos.



En este cuadro de diálogo del asistente se seleccionara la conexión de datos para cargar y actualizar los datos. Para continuar con el ejemplo seleccione la conexión que hasta el momento se ha trabajado: A**CCESS. C:\prgvbnet\visualnet.mdb.Admin**.

Haga nuevamente clic sobre el botón **Siguiente**>, para elegir el tipo de consulta que se realizará usando las instrucciones SQL. Se visualizará la siguiente figura:

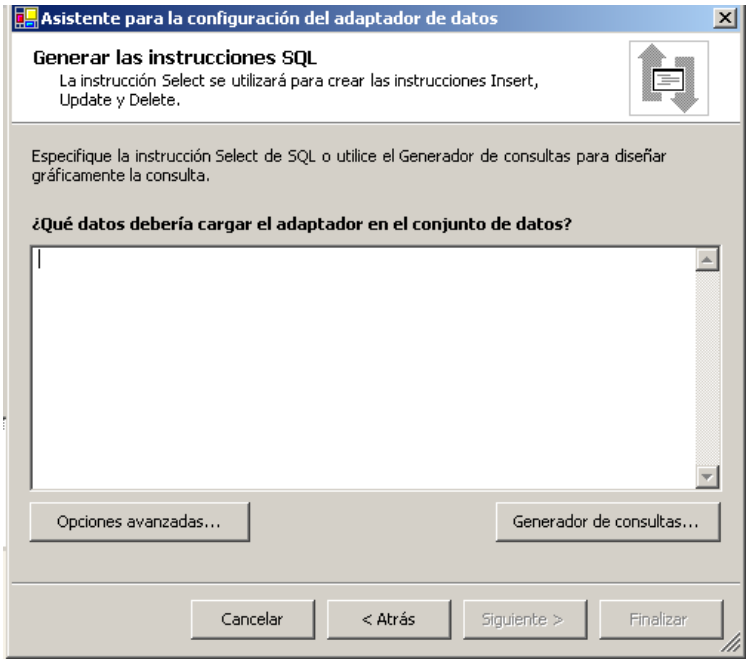
Figura 13.10 Elección del tipo de consulta.



En este cuadro de diálogo del asistente se utiliza la única opción activa **Usar instrucciones SQL** que permitirá especificar la instrucción **Select** para cargar los datos.

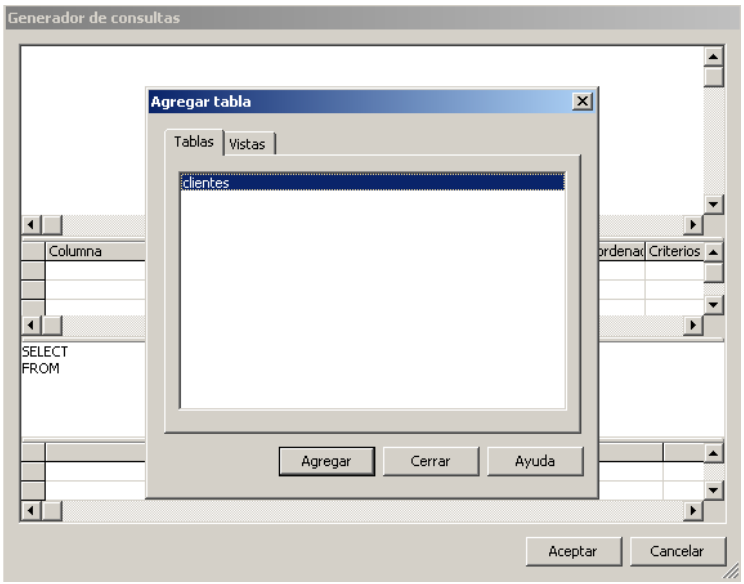
Nuevamente de clic sobre el botón **Siguiente**> lo cual permitirá visualizar el cuadro de diálogo del asistente para generar las instrucciones SQL.

Figura 13.11 Generar las instrucciones SQL.



Este cuadro de diálogo permite generar las instrucciones SQL para mostrar los datos deseados. Existen dos formas de realizar las instrucciones SQL: la primera es escribir cada sentencia SQL, en el cuadro de texto que aparece en el asistente; la segunda es escoger el botón **Generador de Consultas…** Escoja la segunda opción, donde se visualizará la siguiente figura:

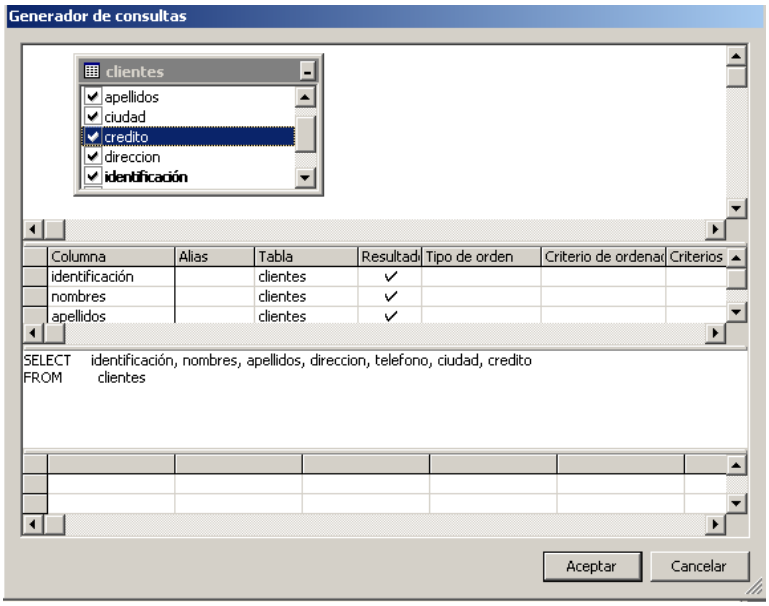
Figura 13.12 Generador de consultas.



Aquí se puede agregar las diferentes tablas o vistas que contiene la base de datos. Se debe resaltar la tabla que se desea utilizar y dar clic en el botón **Agregar** para incluirla en la instrucción SQL. Al finalizar se debe dar clic sobre el botón **Cerrar**. Para el ejemplo que se ha venido trabajando seleccione la tabla **clientes**, agréguela y cierre el cuadro de diálogo **Agregar Tabla**. Se visualizará la siguiente figura:

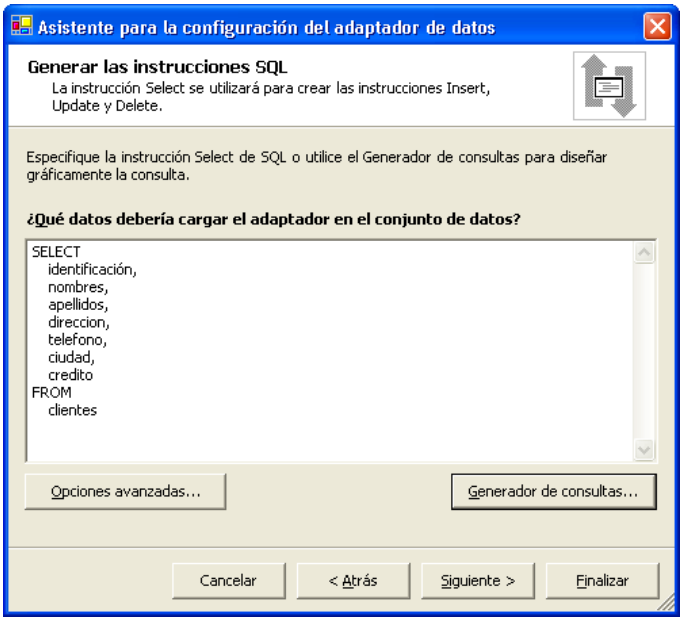
Figura 13.13 Ventana de generación de consultas con la tabla clientes.

.



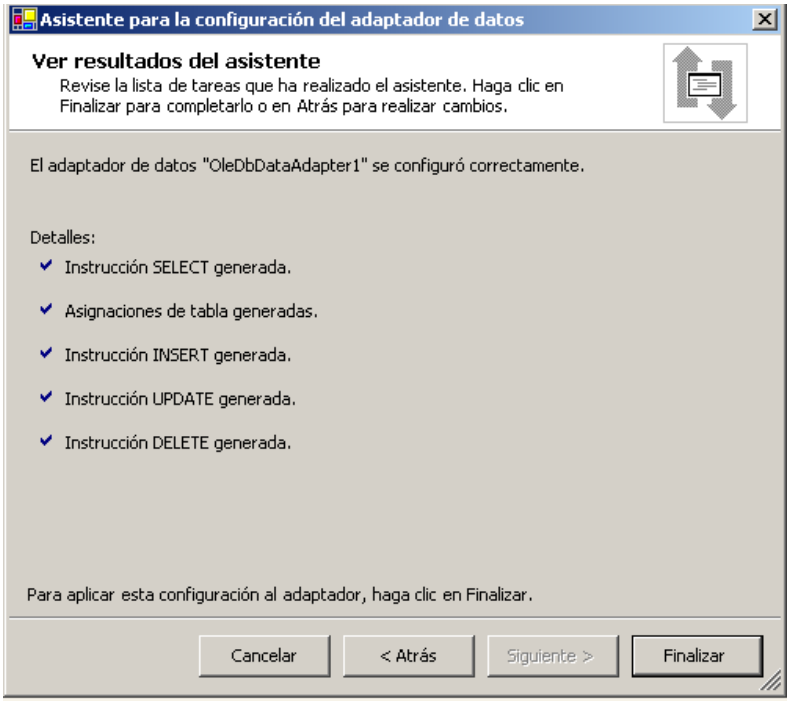
De la tabla puede seleccionar cada uno de los campos que se quieren cargar. A medida que incluya campos se irá generando la instrucción **SQL**. Al terminar de escoger los campos de clic sobre el botón **Aceptar** para volver al generador de consultas **SQL**, como se muestra en la siguiente figura:

Figura 13.14 Ventana con la instrucción SQL.



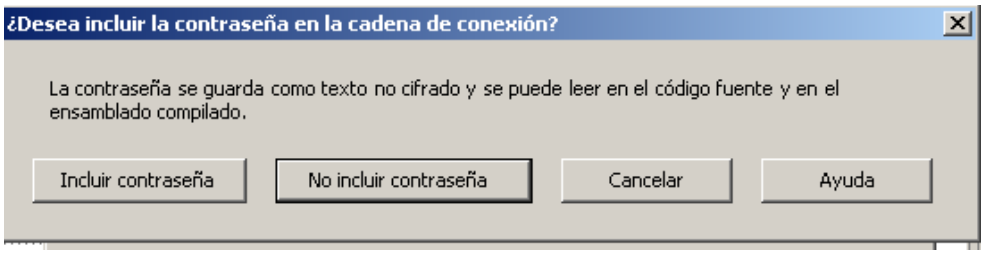
En este cuadro de diálogo se aprecia la instrucción SQL que se ha generado. Nuevamente de clic sobre el botón **Siguiente>** para visualizar los resultados que se han generado con el asistente como se muestra en la siguiente figura:

Figura 13.15 Resultados del Asistente.



Aquí se visualizan todas las tareas que se han realizado con el asistente. Se debe hacer clic en el botón **Finalizar** para terminar el asistente y visualizar la siguiente figura:

Figura 13.16 Ventana de solicitud de contraseña.



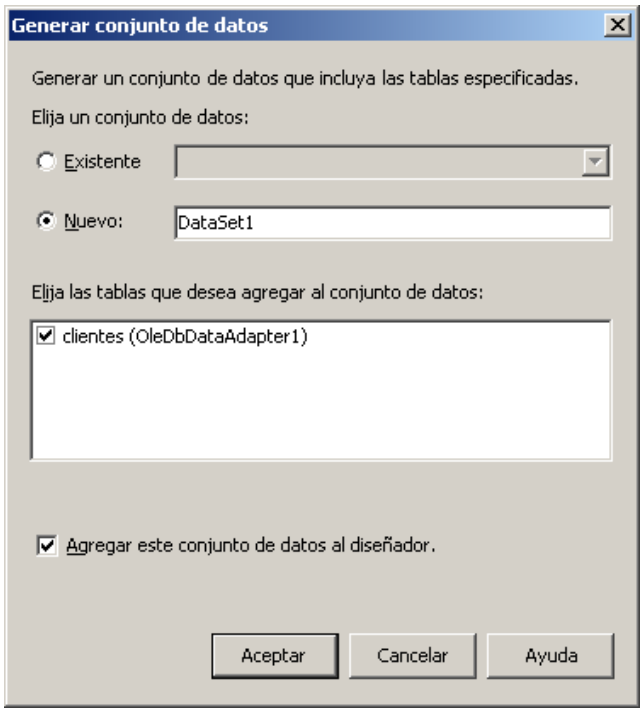
En este cuadro de diálogo el asistente pregunta si se desea incluir una contraseña o no. Pulse el botón **No incluir contraseña**. A la aplicación se le agregará debajo de la pagina los objetos: **OleDbDataAdapter** y **OleDbConnection**. La figura que se visualizará será la siguiente:

Figura 13.17 Página con los objetos OleDbDataAdapter y OleDbConnection.

4. Seleccionar el conjunto de datos

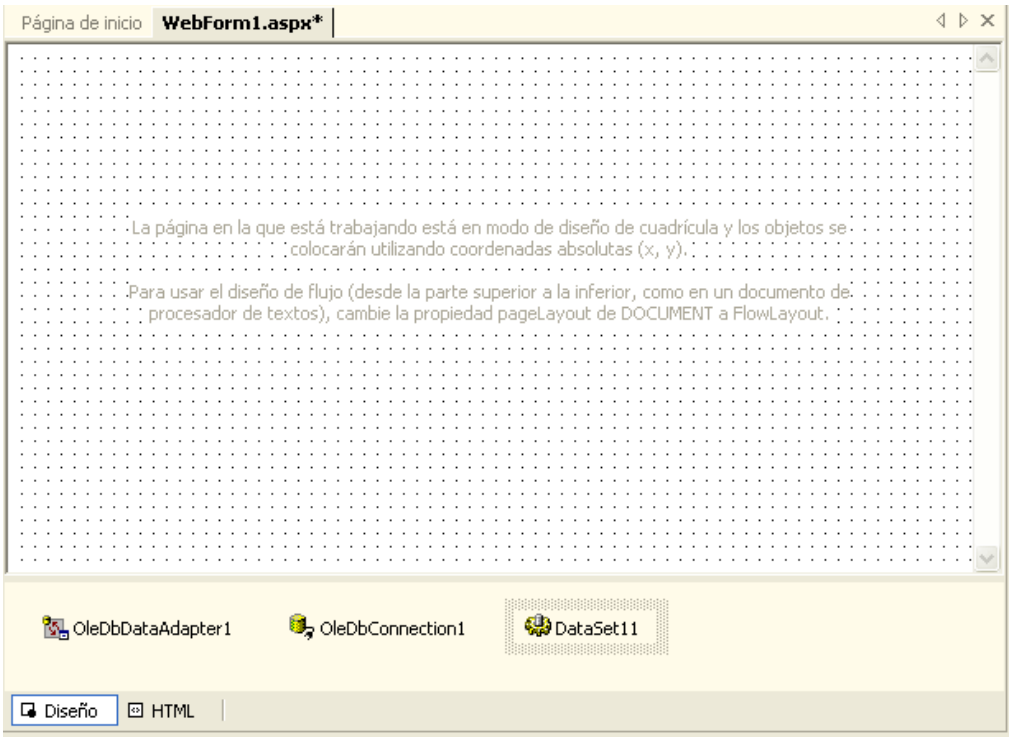
El siguiente paso es crear un objeto que represente los datos que se quieren utilizar en la aplicación. Este objeto es llamado **DataSet** que representa los datos proporcionados por la conexión de datos y que serán extraídos por el objeto adaptador de datos. Haga clic sobre el formulario para asegurarse de que este quede activo, seleccione la opción **Generar conjunto de datos** del menú **Datos**. Se visualizará la siguiente figura:

Figura 13.18 Ventana generar conjunto de datos.



Dé clic sobre el botón **Aceptar** para agregarlo a la aplicación como se muestra en la siguiente figura:

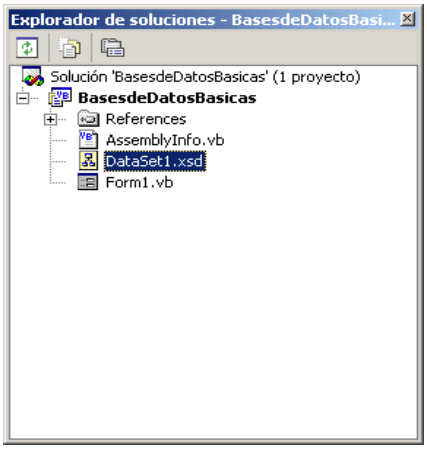
Figura 13.19 Página con el objeto DataSet.



Además de los tres objetos que se han creado, también se agrega un archivo

llamado **DataSet!.xsd** al explorador de soluciones, que representa el esquema de la base de datos. Este esquema es un archivo XML que escribe las tablas, campos, tipos de datos del conjunto de datos. La figura 13.20 muestra como quedaría el explorador de soluciones de la aplicación:

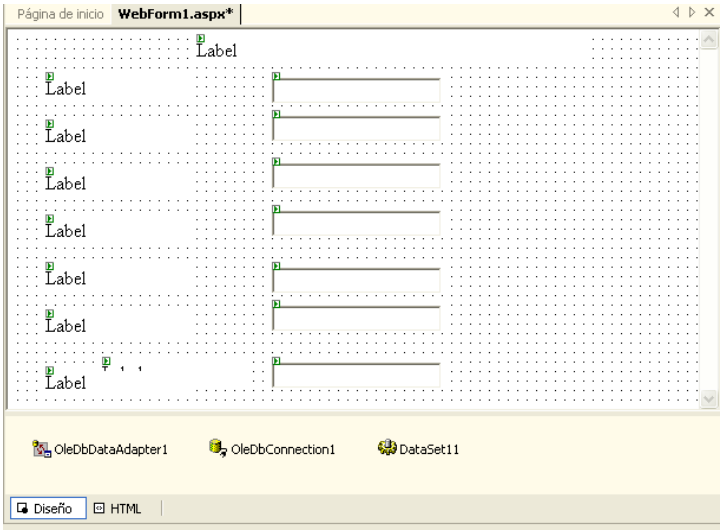
Figura 13.20 Explorador de Soluciones con el objeto DataSet!.xsd.



Después de realizar cada uno de los pasos anteriores, se está listo para visualizar la información de la base de datos. Se puede mostrar a los usuarios los registros que interesan, además se puede crean mecanismos para navegar entre los registros de una o más tablas. Para mostrar la información de una base de datos se pueden utilizar algunos de los siguientes controles: TextBox, ComboBox, ListBox, RadioButton, DataGrid.

Para ver la información de la tabla **clientes** se utilizara el control **TextBox**. A la página se le debe agregar ocho (8) **Label** y siete (7) **TextBox**, que son el número de campos que contiene la tabla **clientes**. La página quedaría como se muestra en la siguiente figura:

Figura 13.21 Página con los objetos Label y TextBox.



- **Establecer las propiedades de los objetos de la interfaz de usuario**

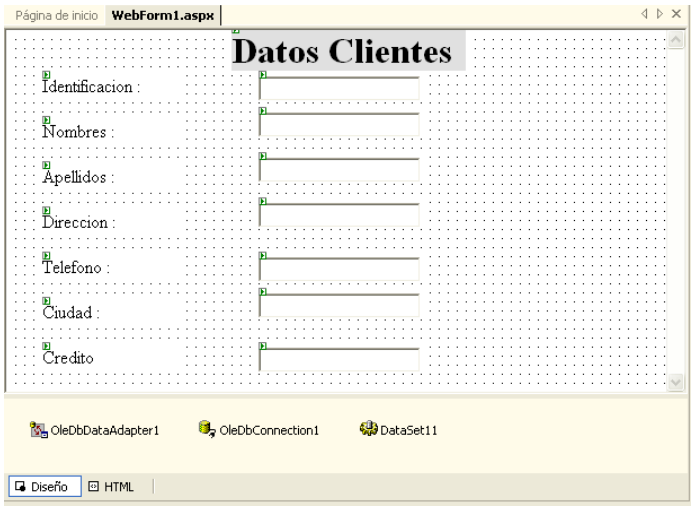
Después de colocar los controles u objetos en la página, establezca las siguientes propiedades a los controles:

Tabla 13.3 Propiedades de los controles de la aplicación BasesdeDatosWeb.

Control	Propiedad	Valor
Label1	(ID)	lbltitulo
	Text	Datos Clientes
	Backcolor	Gris
	Font	Bold
	Size	Large
Label2	(ID)	lblid
	Text	Identificación:
Label3	(ID)	lblnombre
	Text	Nombres:
Label4	(ID)	lblapellido
	Text	Apellidos:
Label5	(ID)	lbldireccion
	Text	Dirección:
Label6	(ID)	lbltelefono
	Text	Telefono:
Label7	(ID)	lblciudad
	Text	Ciudad
Label8	(ID)	lblcredito
	Text	Credito
TextField1	(ID)	txtid
	Text	En blanco
TextField2	(ID)	txtnombres
	Text	En blanco
TextField3	(ID)	txtapellidos
	Text	En blanco
TextField4	(ID)	txtdireccion
	Text	En blanco
TextField5	(ID)	txttelefono
	Text	En blanco
TextField6	(ID)	txtciudad
	Text	En blanco
TextField7	(ID)	Txtcredito
	Text	En blanco
Document	Title	Conexión a una base de datos
WebForm1	Nombre de archivo	pagina

La página se visualizaría como muestra la siguiente figura:

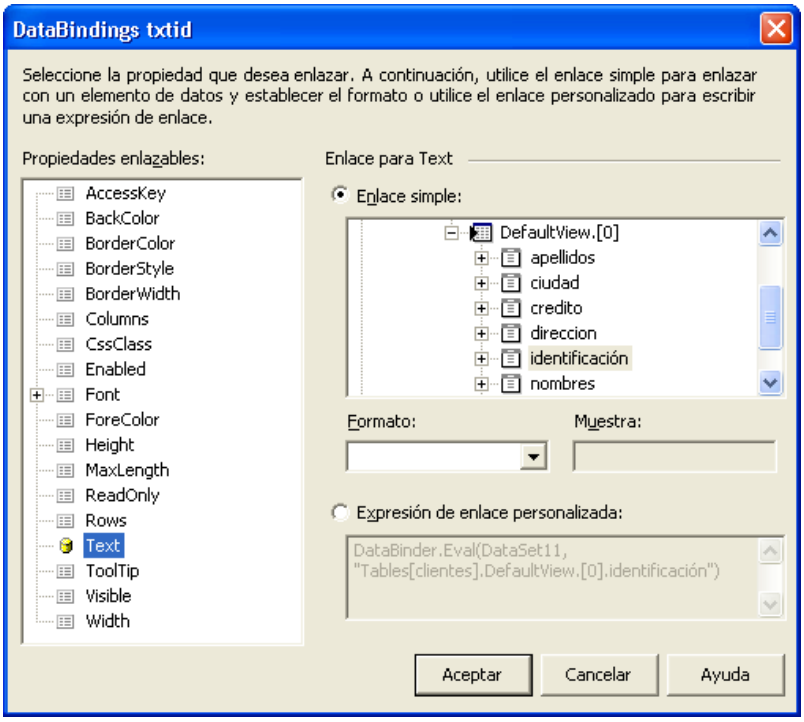
Figura 13.22. Pagina con los controles modificados (BasesdeDatosWeb)



- **Escribir código**

Seleccione el objeto **TextBox** llamado **txtid** y abra la ventana de propiedades. Busque la categoría **DataBindings**, dé clic en los tres puntos (…), luego haga clic en la flecha para desplegar la fuente de datos que se va a enlazar con el cuadro de texto, haga clic en los signos (+) para expandir el conjunto de datos **Dataset11** hasta encontrar los campos de la tabla **clientes** y dé clic sobre el campo **identificación**. Esto permitirá enlazar el objeto **txtid** con el campo **identificación**.

Figura 13.23 Propiedad DataBindings del objeto txtid.



Realice la misma operación para cada uno de los objetos **TextBox**, con el fin de incluir los demás campos de la tabla **clientes**.

Seleccione el objeto **pagina**, dé doble clic para abrir el editor del procedimiento *page_Load* y escriba el siguiente código:

```
DataSet11.Clear()
OleDbDataAdapter1.Fill(DataSet11)
txtid.DataBind ()
txtnombres.DataBind ()
txtapellidos.DataBind ()
txtdireccion.DataBind ()
txttelefono.DataBind ()
txtciudad.DataBind ()
txtcredito.DataBind ()
```

En la primera línea se utiliza el método **Clear** para limpiar el conjunto de datos cuando se ejecuta la aplicación. En la segunda línea se utiliza el método **Fill** para rellenar el objeto **OleDbDataAdatper1** con el conjunto de datos llamado **Dataset11.**

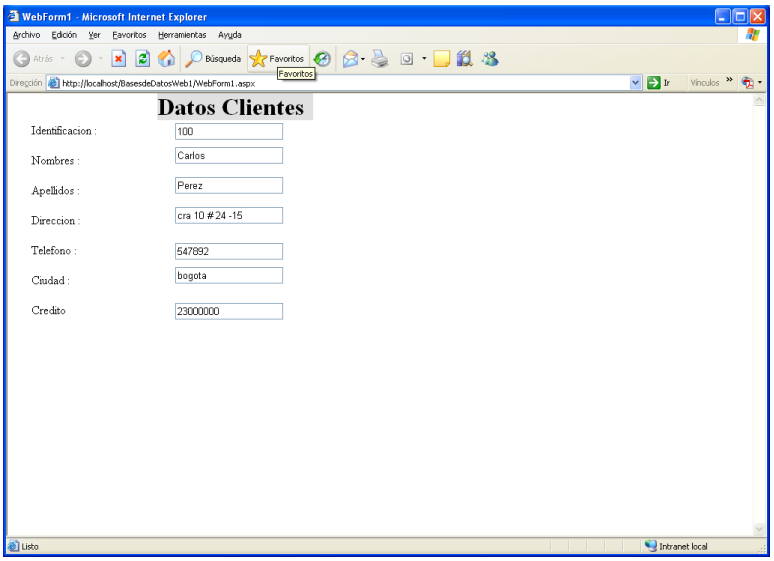
Por último se utiliza el método **DataBind** () del objeto **TextBox** para poder visualizar los registros de la tabla seleccionada.

- **Ejecutar el proyecto**

Para ejecutar el proyecto en el entorno de desarrollo de visual Basic.NET, se debe realizar lo siguiente:

- Haga clic en el botón Iniciar de la barra de herramientas estándar. También puede presionar F5 para ejecutar el proyecto. Si la aplicación se ejecuta sin errores, aparecerá una versión ejecutable del formulario, visualizándose la siguiente figura:

Figura 13.24 Ejecución aplicación BasesdeDatosWeb.



13.2 Controles de navegación en ASP.NET

Como se puede apreciar, en la figura 13.24 se visualiza la información solamente del primer registro de la tabla **clientes**. Para moverse por cada uno de los registros que contiene la tabla es necesario realizar las siguientes tareas:

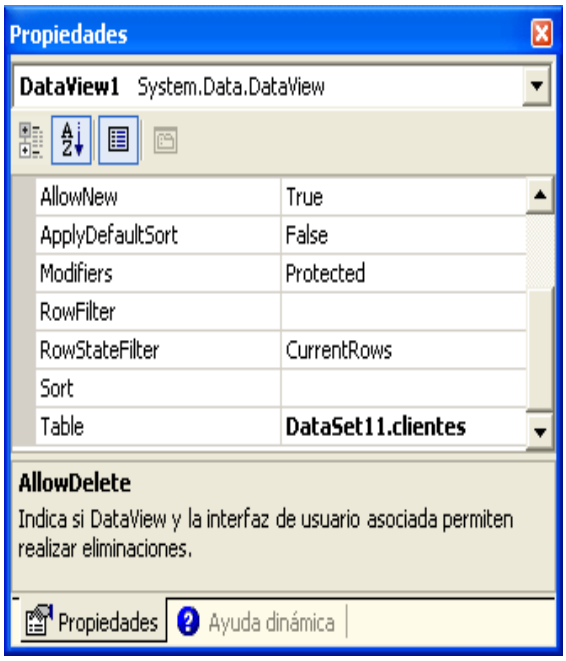
a) Agregue cuatro botones y configure las propiedades de acuerdo a la siguiente tabla:

Tabla 13.4 Propiedades de los nuevos controles de la aplicación.

Control	Propiedad	Valor
Button1	Id	botonprimero
	Text	Primero
Button2	Id	botonsiguiente
	Text	Siguiente
Button3	Id	botonanterior
	Text	Anterior
Button4	Id	botonúltimo
	Text	Último

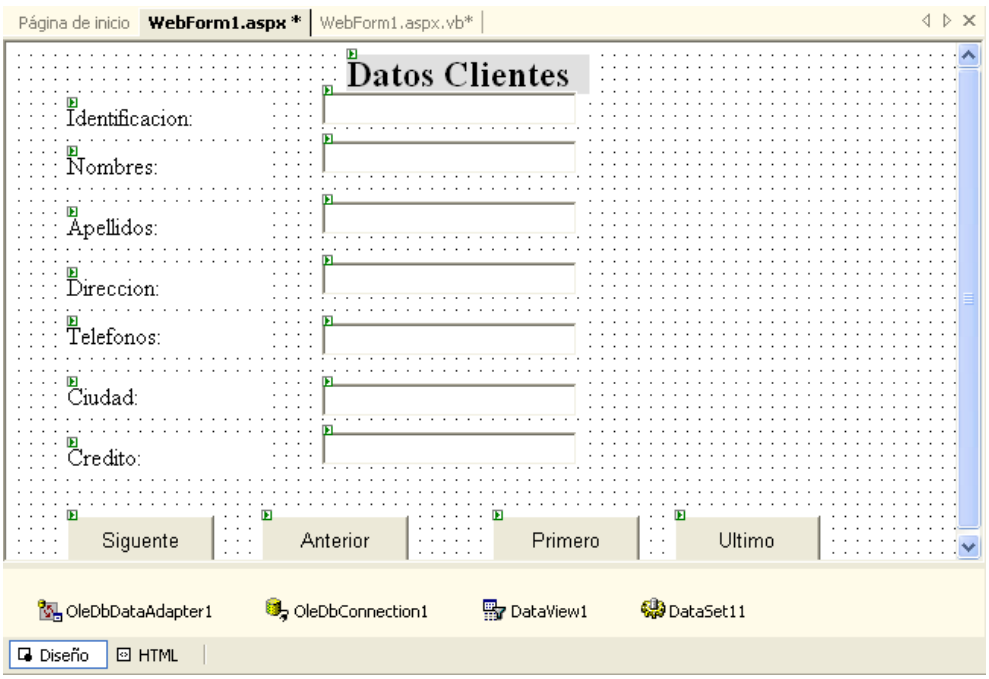
b) Seleccione de cuadro de herramientas el objeto **DataView** y arrástrelo a la página. Modifique la propiedad **Table** del **DataView** y asócielo con el **DataSet11.clientes**, el control **DataView** quedaría de la siguiente manera:

Figura 13.25 Propiedad Table del DataView.



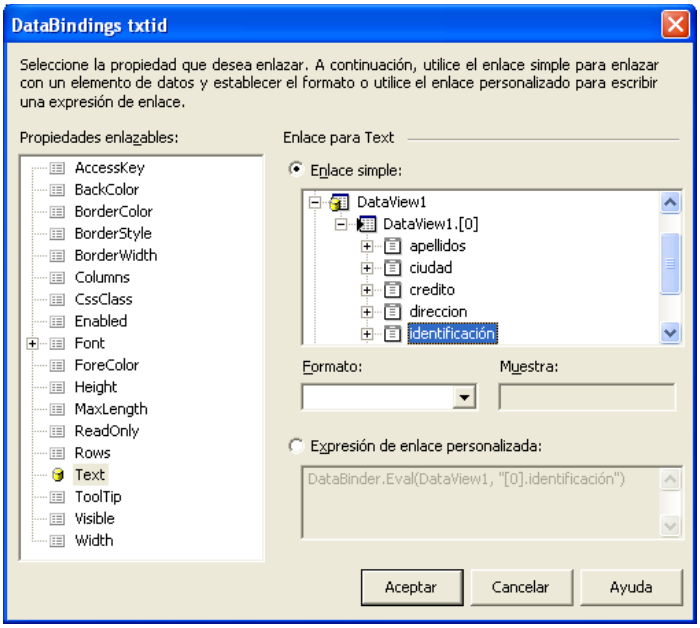
Al finalizar se visualizará la siguiente figura:

Figura 13.26 Página con los nuevos controles.



c) Modifique la propiedad **DataBindings** del objeto **txtid**, seleccione el control **DataView** y escoja el campo **identificación** de la tabla **clientes**. Realice este mismo procedimiento para cada uno de los campos.

Figura 13.27 Propiedad DataBindings del objeto txtId.



d) Codificación de los controles **Button**

Objeto botonsiguiente

```
Dim abc As Single
 Dim index As Integer
index = Me.ViewState("index")
index += 1
If index > DataSet11.Tables(0).Rows.Count - 1 Then
   index = DataSet11.Tables(0).Rows.Count - 1
   index = 0
End If
Me.ViewState("index") = index
abc = DataSet11.Tables(0).Rows(index).Item("identificación")
DataView1.RowFilter = " identificación ='" & abc & "'"
txtid.DataBind()
txtnombres.DataBind()
txtapellidos.DataBind()
txtdireccion.DataBind ()
txttelefono.DataBind ()
txtciudad.DataBind ()
txtcredito.DataBind ()
```

Se definen las variables **abc** de tipo **Single** e **index** de tipo **Integer**. A la variable **index** se le asigna la posición del registro actual por intermedio del método **ViewState** (), y se incrementa la variable **index** en 1. Con la estructura de decisión **If** se pregunta si la variable **index** es mayor que la posición del último registro. Si es verdadero se asigna a la variable **index** el valor del último registro de la tabla. Si no ingresa a la estructura **If** al método **ViewState** se le asigna el valor que contiene la variable **index**. A

la variable **abc** se le asigna el valor del campo **identificación** la cual se obtiene por medio del método **Rows** encontrado con el valor que contiene la variable **index**. Por intermedio del método **RowFilter** del objeto **DataView** se filtra la tabla para obtener el registro deseado. Por último se utiliza el método **DataBind** () del objeto **TextBox** para poder visualizar el registro seleccionado.

Objeto botonanterior

```
Dim abc As Single
Dim index As Integer
index = Me.ViewState("index")
index -= 1
Me.ViewState("index") = index
abc = DataSet11.Tables(0).Rows(index).Item("identificación")
DataView1.RowFilter = " identificación='" & abc & "'"
txtid.DataBind ()
txtnombres.DataBind ()
txtapellidos.DataBind ()
txtdireccion.DataBind ()
txttelefono.DataBind ()
txtciudad.DataBind ()
txtcredito.DataBind ()
```

Se definen las variables **abc** de tipo **single** e **index** de tipo **Integer**. A la variable **index** se le asigna la posición del registro actual por intermedio del método **ViewState** (), y se decrementa la variable **index** en 1. Al método **ViewState** se le asigna el valor que contiene la variable **index**. A la variable **abc** se le asigna el valor del campo **identificación**, la cual se obtiene por medio del método **Rows** encontrado con el valor que contiene la variable **index**. Por intermedio del método **RowFilter** del objeto **DataView** se filtra la tabla para obtener el registro deseado. Por último se utiliza el método **DataBind** () del objeto **TextBox** para poder visualizar el registro seleccionado.

Objeto botonprimero

```
Dim abc As Single
Dim index As Integer = 0
Me.ViewState("index") = index
abc = DataSet11.Tables(0).Rows(index).Item("identificación")
DataView1.RowFilter = " identificación='" & abc & " '"
txtid.DataBind ()
txtnombres.DataBind ()
txtapellidos.DataBind ()
txtdireccion.DataBind ()
txttelefono.DataBind ()
txtciudad.DataBind ()
txtcredito.DataBind ()
```

Se definen las variables **abc** de tipo **single** e **index** de tipo **Integer** inicializada en 0. Al método **ViewState** se le asigna el valor que contiene la variable **index**. A la variable **abc** se le asigna el valor del campo **identificación**, la cual se obtiene por medio del método **Rows** encontrado con el valor que contiene la variable **index**. Por intermedio del método **RowFilter** del objeto **DataView** se filtra la tabla para obtener el registro deseado. Por último se utiliza el método **DataBind** () del objeto **TextBox** para poder visualizar el registro seleccionado.

Objeto botonultimo

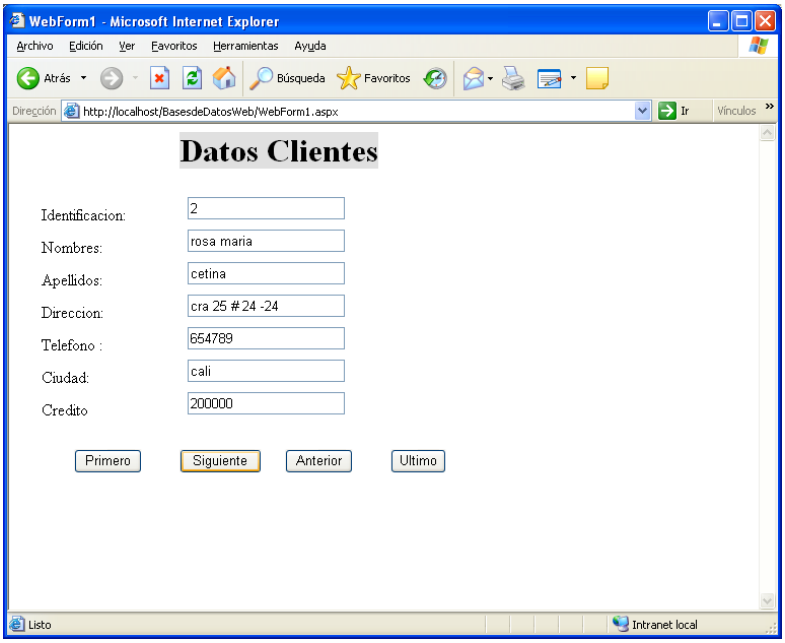
```
Dim abc As Single
Dim index As Integer
index = DataSet11.Tables(0).Rows.Count - 1
Me.ViewState("index") = index
abc = DataSet11.Tables(0).Rows(index).Item("identificación")
DataView1.RowFilter = " identificación='" & abc & "'"
txtid.DataBind ()
txtnombres.DataBind ()
txtapellidos.DataBind ()
txtdireccion.DataBind ()
txttelefono.DataBind ()
txtciudad.DataBind ()
txtcredito.DataBind ()
```

Se definen las variables **abc** de tipo **single** e **index** de tipo **Integer**. A la variable **index** se le asigna el valor del último registro de la tabla utilizando el método **count** el cual permite contar el total de filas de la tabla. Al método **ViewState** se le asigna el valor que contiene la variable **index** y a la variable **abc** se le asigna el valor del campo **identificación** la cual se obtiene por medio del método **Rows** encontrado con el valor que contiene la variable **index**. Por intermedio del método **RowFilter** del objeto **DataView** se filtra la tabla para obtener el registro deseado. Por último se utiliza el método **DataBind** () del objeto **TextBox** para poder visualizar el registro seleccionado.

- **Ejecutar el proyecto**

Al ejecutarse nuevamente la aplicación se visualizará la siguiente figura:

Figura 13.28 Ejecución de la aplicación BasesdeDatosWeb con controles.



Al dar clic sobre cada uno de los botones se podrá desplazar por cada uno de los registros que se tiene en la tabla que se está consultando.

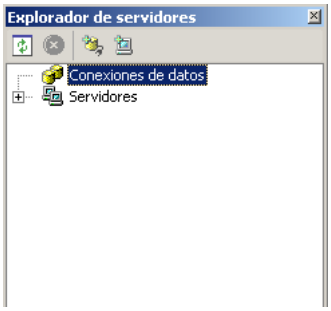
13.3 Acceder a una base de datos mediante el asistente de formularios

Existe una forma más sencilla de acceder a una base de datos especifica, y es utilizando el **asistente de formulario de datos**, el cual se encuentra en la opción **agregar nuevo elemento** del menú **proyecto**. Para esto se seguirá trabajando en la misma aplicación **BasesdeDatosWeb** y se debe realizar los siguientes pasos:

1. Hacer la conexión a la base de datos

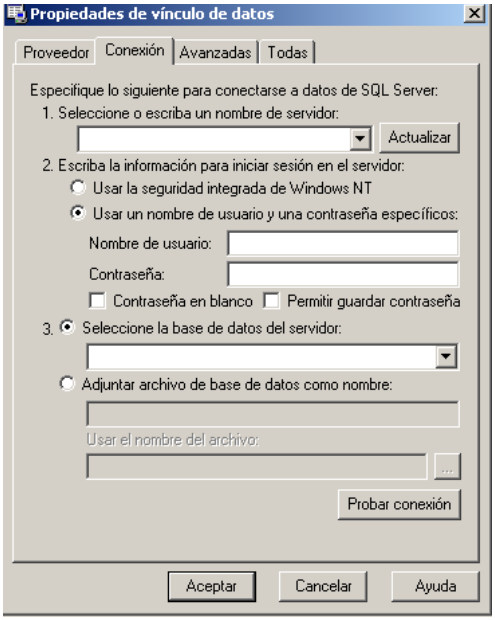
Para realizar una conexión a una base de datos existente seleccione **Explorador de servidores** del menú **Ver**, donde se visualizará el cuadro de diálogo **Explorador de servidores**, como se aprecia en la siguiente figura:

Figura 13.29 Ventana Explorador de servidores.



Seleccione el icono C**onectar con bases de datos**  para iniciar la conexión a la base de datos y visualizar el cuadro de diálogo **Propiedades de vínculos de datos**, como se muestra en la siguiente figura:

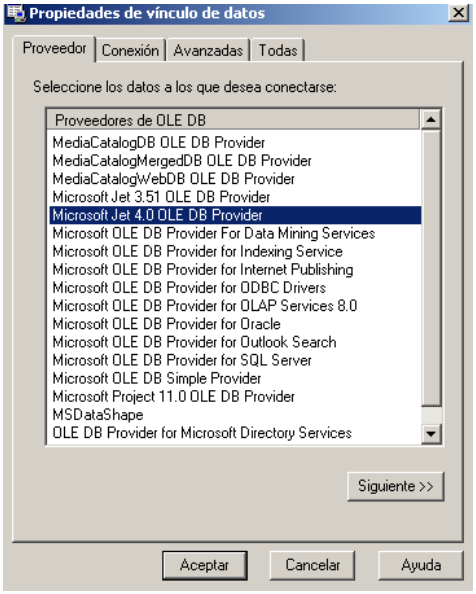
Figura 13.30 Propiedades de vínculos de datos



Pulse sobre la pestaña llamada **Proveedor** para seleccionar el proveedor de la

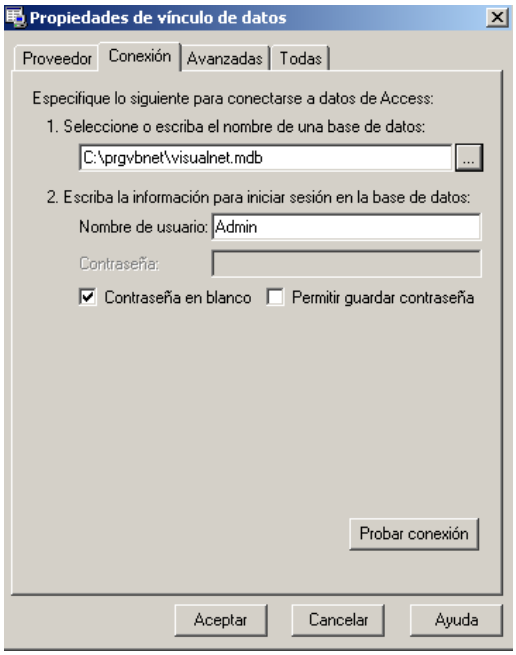
base de datos. Escoja **Microsoft jet 4.0 OLE DB Provide**, y pulse el botón **Siguiente >**.

Figura 13.31 Propiedades de vínculos de datos (proveedor).



Se visualizará la pestaña **Conexión** del cuadro de diálogo **propiedades de vínculo de datos**, allí en el paso 1, seleccione la base de datos que previamente se había creado, como se muestra en la siguiente figura:

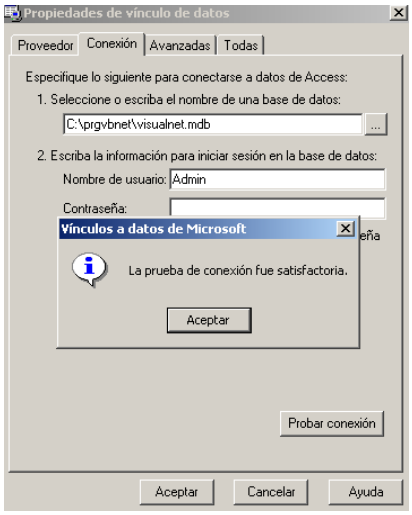
Figura 13.32 Propiedades de vínculos de datos (conexión).



Si desea crear una contraseña a la base de datos desactive el cuadro de verificación **contraseña en blanco** y automáticamente el campo **contraseña** se habilitara para que el usuario digite la contraseña que desee. El paso 2 es opcional.

Ahora también se puede verificar la conexión pulsando el botón **Probar conexión**. Si la conexión ha sido exitosa se mostrará la siguiente figura:

Figura 13.33 Mensaje con una conexión exitosa.

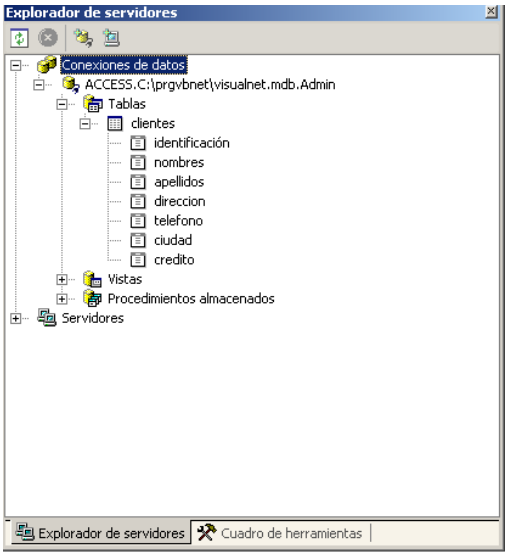


En caso de que la conexión no fuese exitosa, se debe iniciar nuevamente con cada uno de los pasos realizados hasta el momento para resolver el problema. Uno de las posibles fallas en la conexión es que la base de datos está abierta en el momento de la conexión proceda a cerrarla y realice nuevamente la conexión.

Para continuar pulse el botón **Aceptar** de la prueba y conexión y luego pulse el botón **Aceptar** del cuadro de diálogo **Propiedades de vínculo de datos** para terminar la conexión.

En este momento se puede pulsar el signo (+) a la izquierda de la leyenda **conexiones de datos** del cuadro de diálogo **Explorador de servidores** para visualizar la conexión a la base de datos **visualnet**. Cada vez que pulse el signo (+) se podrá apreciar la estructura de la base de datos, como se muestra en la siguiente figura:

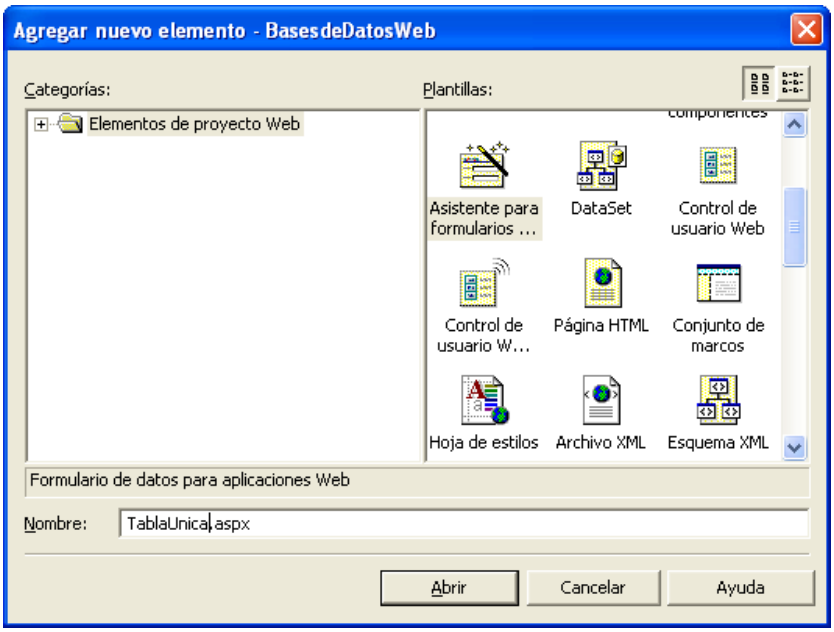
Figura 13.34 Ventana explorador de servidores con una conexión.



2. Utilizar el asistente para formularios de datos

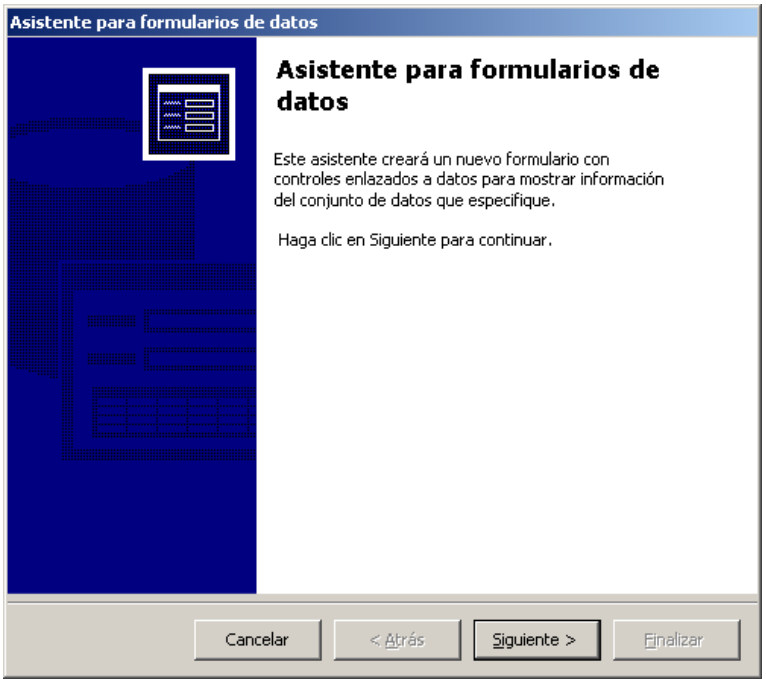
Ahora seleccione la opción **agregar nuevo elemento** del menú **Proyecto** y seleccione el icono **Asistente para formularios de datos**, como se muestra en la siguiente figura:

Figura 13.35 Agregar nuevo elemento.



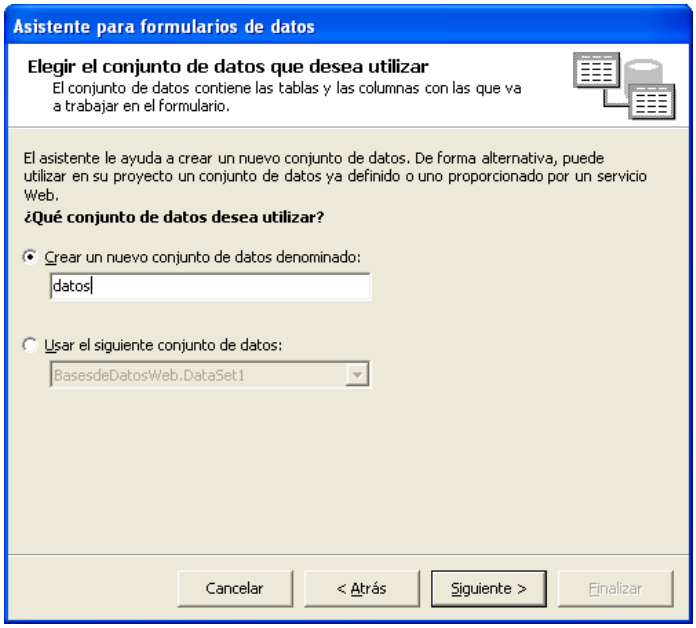
En esta ventana, en la opción **Nombre** se podrá cambiar el nombre de la página. Para este ejemplo el nombre de la página será **TablaUnica**.vb. Al pulsar el botón **Abrir**, se visualizará la siguiente ventana:

Figura 13.36 Asistente para formulario de Datos.



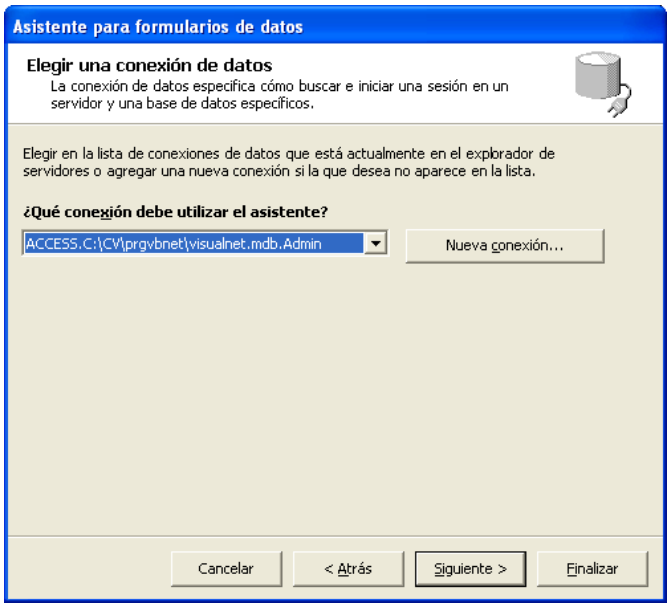
Esta pantalla es el inicio del asistente para formularios de datos, pulse el botón **Siguiente**> para visualizar la siguiente figura:

Figura 13.37 Elegir el conjunto de datos.



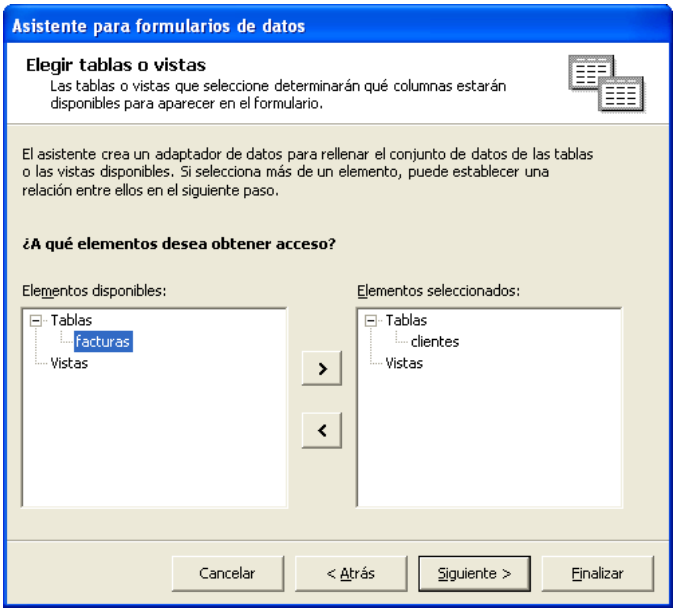
En esta ventana se selecciona el conjunto de datos que contiene las tablas y las columnas con las que va a trabajar la página. En la opción crear un nuevo conjunto de datos escriba **datos** y pulse el botón **Siguiente**> para visualizar la siguiente ventana:

Figura 13.38 Elegir conexión de datos.



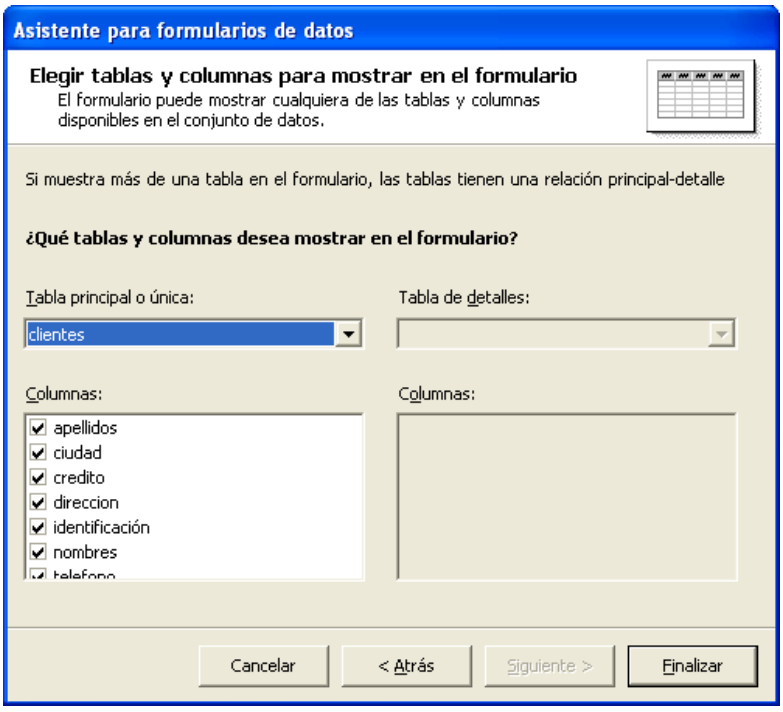
En esta ventana se elige la conexión específica, seleccione la conexión que se realizo anteriormente y pulse el botón **Siguiente**> para visualizar la figura:

Figura 13.39 Elegir tablas o vistas.



En esta ventana seleccione la tabla **clientes** y pulse el botón **">"** para colocarla en la ventana **Elementos seleccionados**. Luego pulse el botón **Siguiente**> para visualizar la figura:

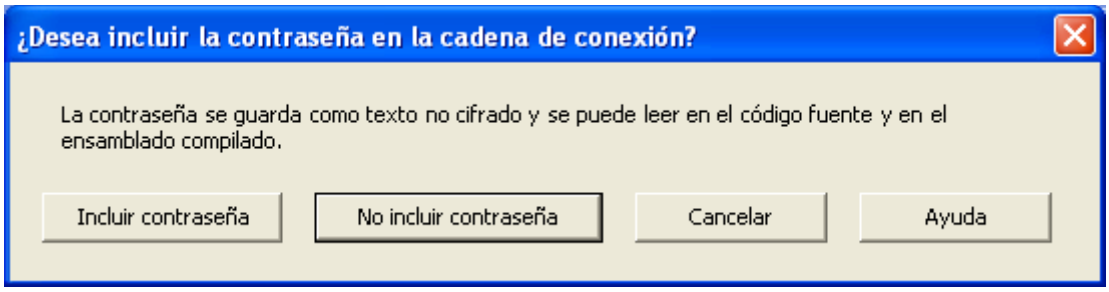
Figura 13.40 Elegir tablas y columnas para mostrar en el formulario.



En esta ventana se debe seleccionar los campos que se quieren mostrar en la página. Al escoger cada uno de los campos, pulse el botón **Finalizar** para ver la

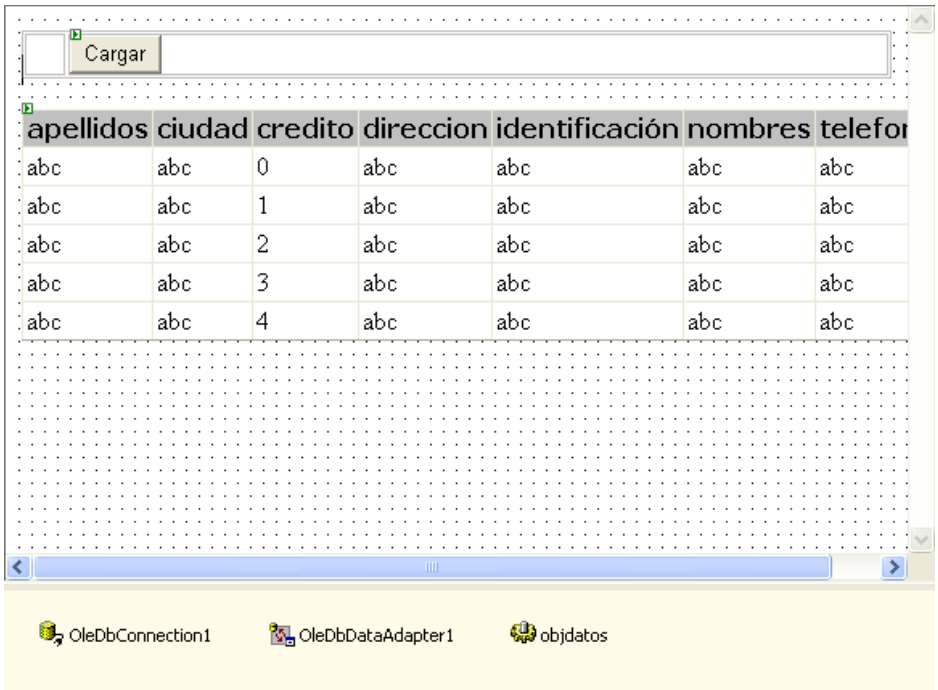
siguiente figura:

Figura 13.41 Ventana para la contraseña.



En este cuadro de diálogo el asistente pregunta si se desea incluir una contraseña o no. Pulse el botón **No incluir contraseña**. A la aplicación se le agregará debajo de la página los objetos: **OleDbDataAdapter1, OleDbConnection1 y objdatos**. La figura que se visualizará será la siguiente:

Figura 13.42 Página final realizada con el asistente de formularios.

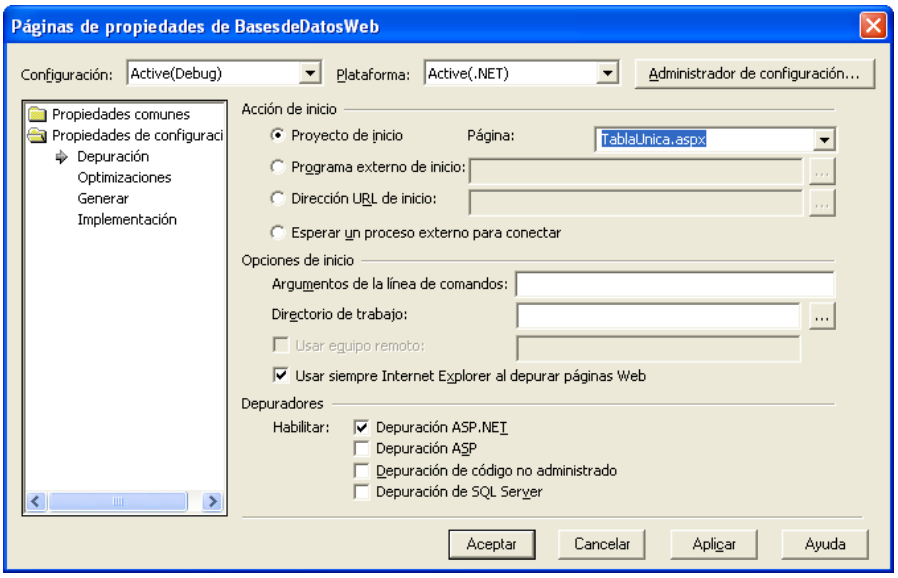


- **Ejecutar la aplicación**

Para ejecutar el proyecto en el entorno de desarrollo de visual Basic.NET, se debe realizar lo siguiente:

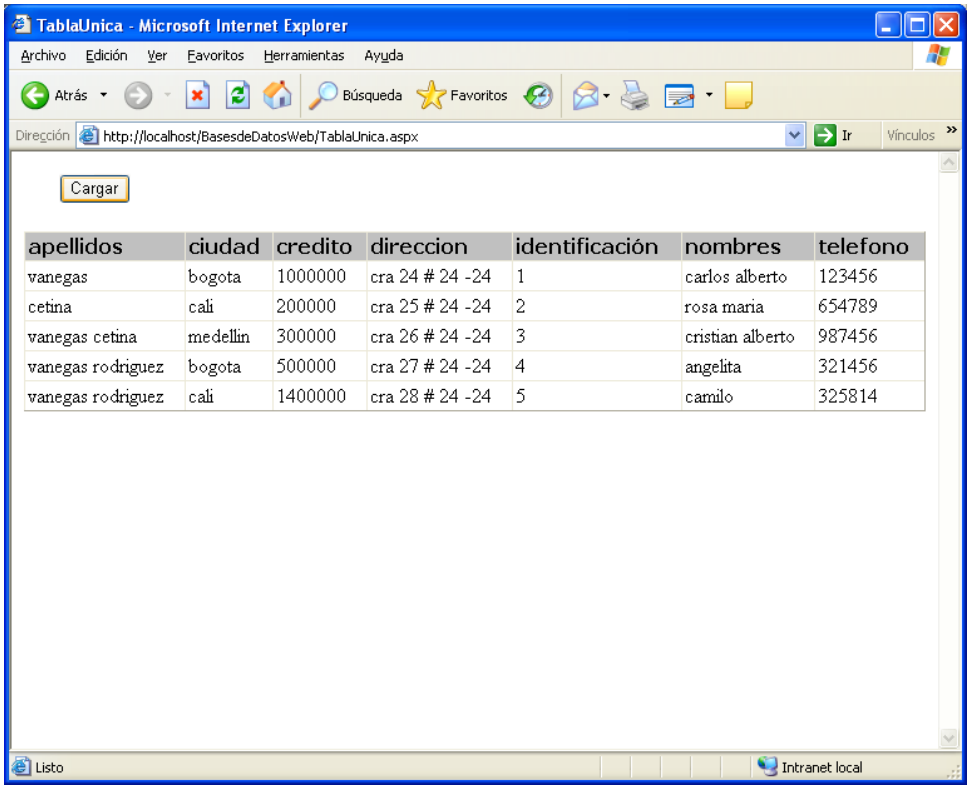
- En el menú **Proyecto**, seleccione **propiedades de BasesdeDatosWeb,** en la opción **Propiedades de configuración** seleccione la página **Tablaunica.aspx** y pulse A**ceptar.**

Figura 13.43 Propiedades de BasesdedatosWeb.



Haga clic en el botón Iniciar de la barra de herramientas estándar o presione **F5** para ejecutar el proyecto. Se visualizará la siguiente figura:

Figura 13.44 Ejecución de la aplicación BasededatosWeb.



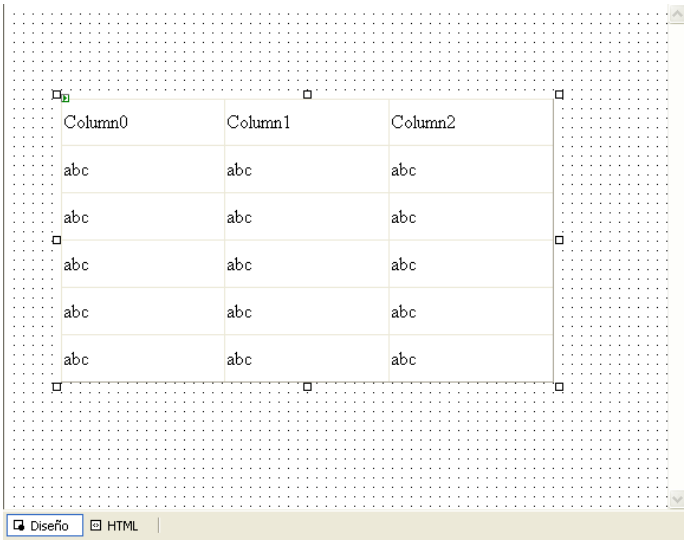
apellidos	ciudad	credito	direccion	identificación	nombres	telefono
vanegas	bogota	1000000	cra 24 # 24 -24	1	carlos alberto	123456
cetina	cali	200000	cra 25 # 24 -24	2	rosa maria	654789
vanegas cetina	medellin	300000	cra 26 # 24 -24	3	cristian alberto	987456
vanegas rodriguez	bogota	500000	cra 27 # 24 -24	4	angelita	321456
vanegas rodriguez	cali	1400000	cra 28 # 24 -24	5	camilo	325814

Pulsar el botón **Cargar** para visualizar los registros de la tabla **clientes**.

13.4 Acceder a una base de datos con un control DataGrid

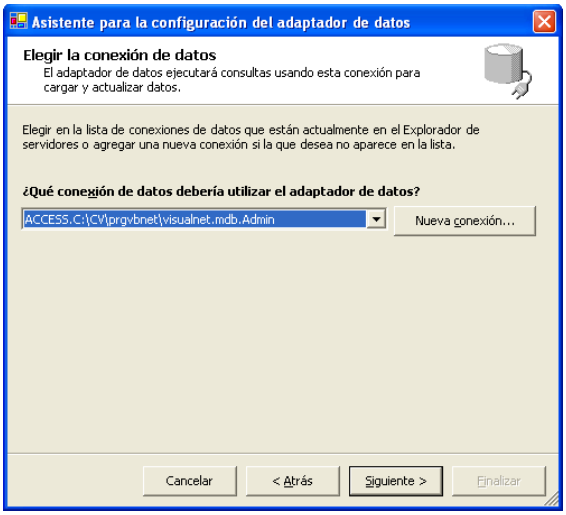
Como se dijo anteriormente para acceder a los datos de una base de datos es posible utilizar varios controles de ASP.NET. En este caso se realizará un ejercicio donde se utilizará el control **DataGrid** para visualizar datos de una tabla de una base de datos. Continuando con la aplicación **BasesdeDatosWeb, s**eleccione la opción **agregar nuevo elemento** del menú **Proyecto** y seleccione el icono **WebForms.** En esta ventana, en la opción **Nombre** cambie el nombre de la página por **PaginaConDataGrid**.vb. Al pulsar el botón **Abrir**, y al arrastrar el control **DataGrid** del cuadro de herramientas hacia la página se visualizará la siguiente ventana:

Figura 13.45 Página con un control DataGrid



Seleccione la ficha **Datos** del cuadro de herramientas y escoja el objeto **OleDbDataAdapter** que sirve para seleccionar datos de una base de datos de Access. Dé doble clic sobre el objeto **OleDbDataAdapter** para visualizar el **Asistente para la configuración del adaptador de datos**, pulse el botón **Siguiente**> para visualizar la ventana para elegir la conexión de datos.

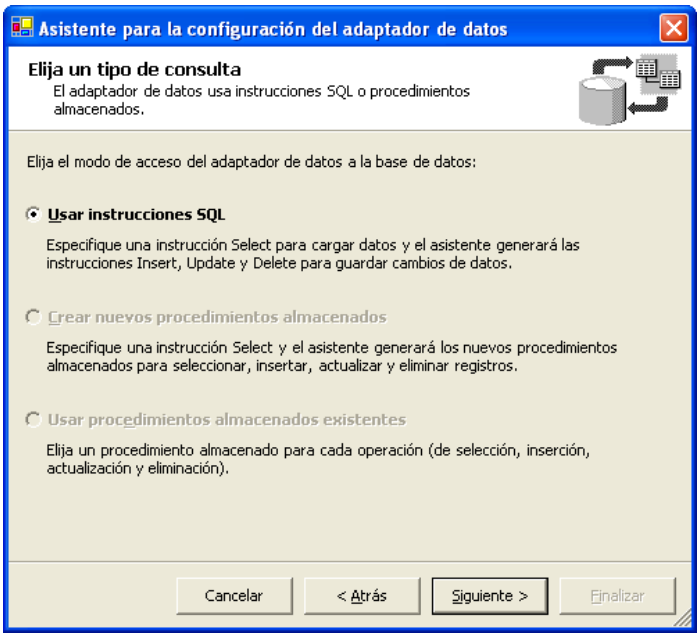
Figura 13.46 Asistente para configuración de datos (DataGrid).



Para continuar con el ejemplo seleccione la única conexión que hasta el momento se ha realizado como lo es la conexión de ACCESS: **C:\prgvbnet\visualnet.mdb.Admin**.

Dé nuevamente clic sobre el botón **Siguiente**>, para elegir el tipo de consulta que se realizará usando las instrucciones SQL. Se visualizará la siguiente figura:

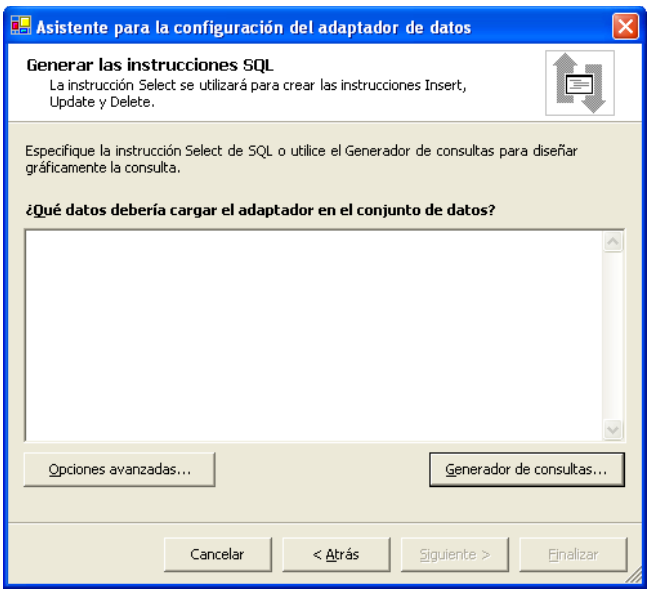
Figura 13.47 Elegir tipo de consulta (DataGrid).



En este cuadro de diálogo del asistente se utiliza la única opción activa **Usar instrucciones SQL** que permitirá especificar la instrucción **Select** para cargar los datos.

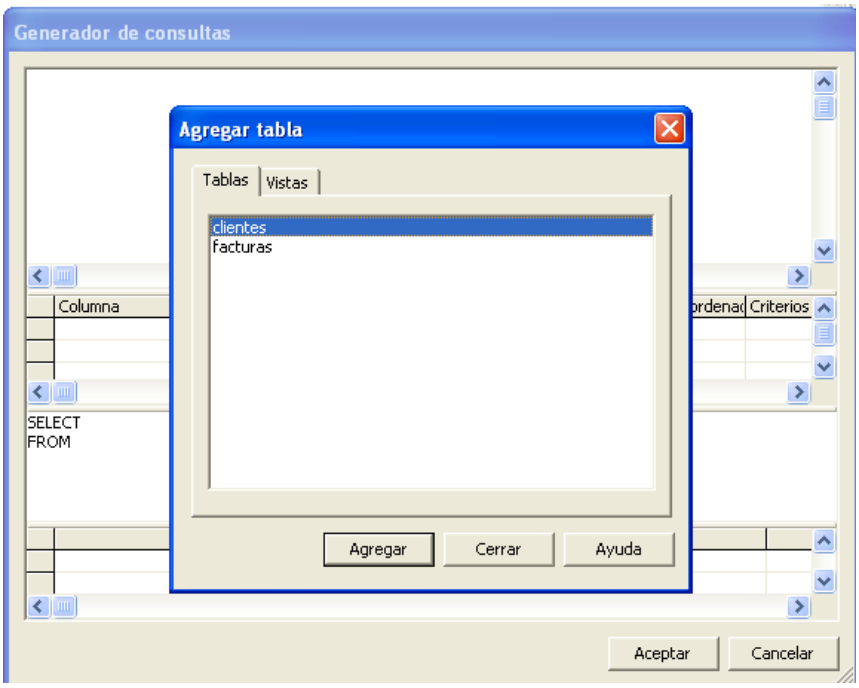
Nuevamente de clic sobre el botón **Siguiente**>, lo cual permitirá visualizar el cuadro de diálogo del asistente para generar las instrucciones SQL.

Figura 13.48 Generar las instrucciones SQL (DataGrid).



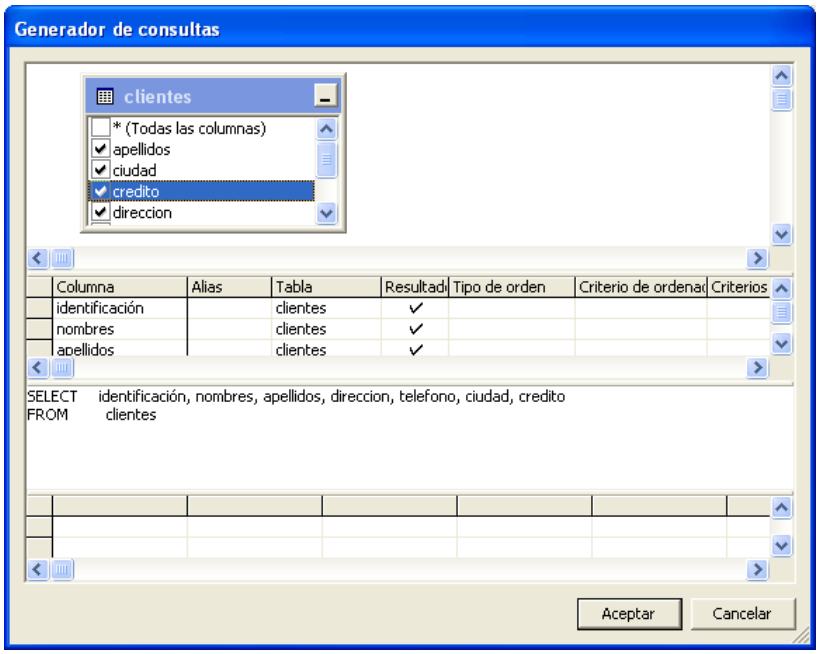
Este cuadro de diálogo permite generar las instrucciones SQL para mostrar los datos deseados. Pulse el botón **Generador de Consultas…** Para visualizar la siguiente figura:

Figura 13.49 Generar de consultas (DataGrid).



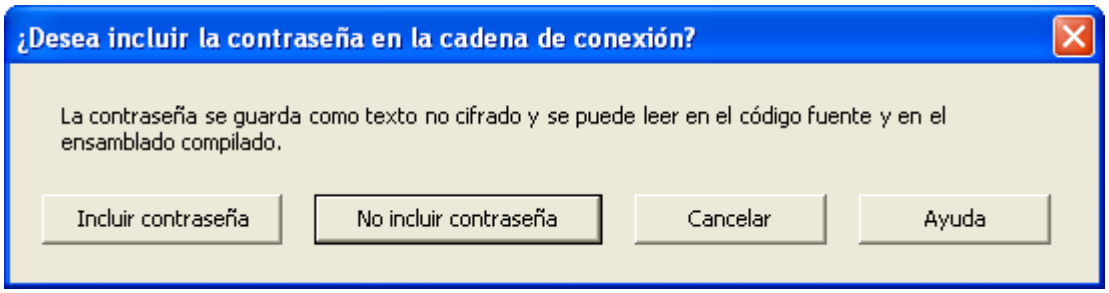
Para el ejemplo que se ha venido trabajando seleccione la tabla **clientes**, agréguela y cierre el cuadro de diálogo **Agregar Tabla**. Se visualizará la siguiente figura:

Figura 13.50 Generar de consultas con tabla seleccionada (DataGrid).



Pulse el botón **Siguiente**> hasta visualizar la siguiente figura:

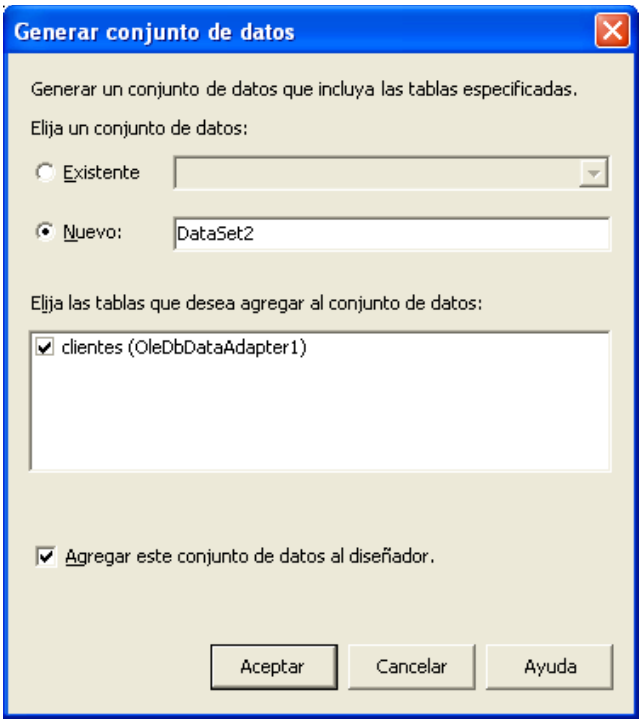
Figura 13.51 Ventana de contraseñas (DataGrid).



En este cuadro de diálogo el asistente pregunta si se desea incluir una contraseña o no. Pulse el botón **No incluir contraseña**. A la aplicación se le agregará debajo de la pagina los objetos: **OleDbDataAdapter** y **OleDbConnection**.

El siguiente paso es crear un objeto que represente los datos que se quieren utilizar en la aplicación. Seleccione la opción **Generar conjunto de datos** del menú **Datos,** escoja la opción **Nuevo** y escriba como nombre del conjunto de datos **DataSet2,** pulse el botón **Aceptar**.

Figura 13.52 Generar conjunto de datos (DataGrid).

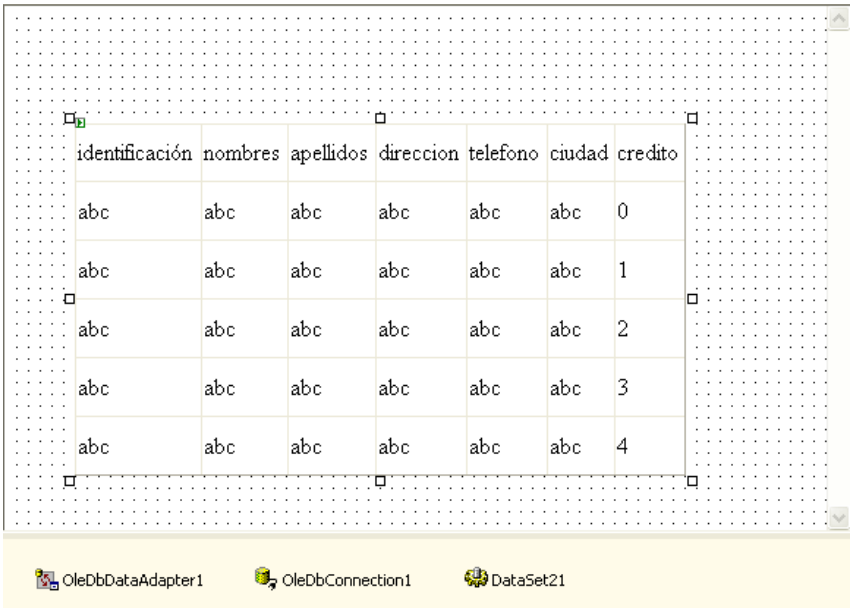


Ahora seleccione el objeto **DataGrid** y busque la propiedad **DataSource** y asígnele el objeto **Dataset21**, como lo indica la figura:

Figura 13.53 Propiedad DataSource del control DataGrid.

El **DataGrid** se rellenará con los campos de la tabla **clientes**, como se puede apreciar, en la siguiente figura:

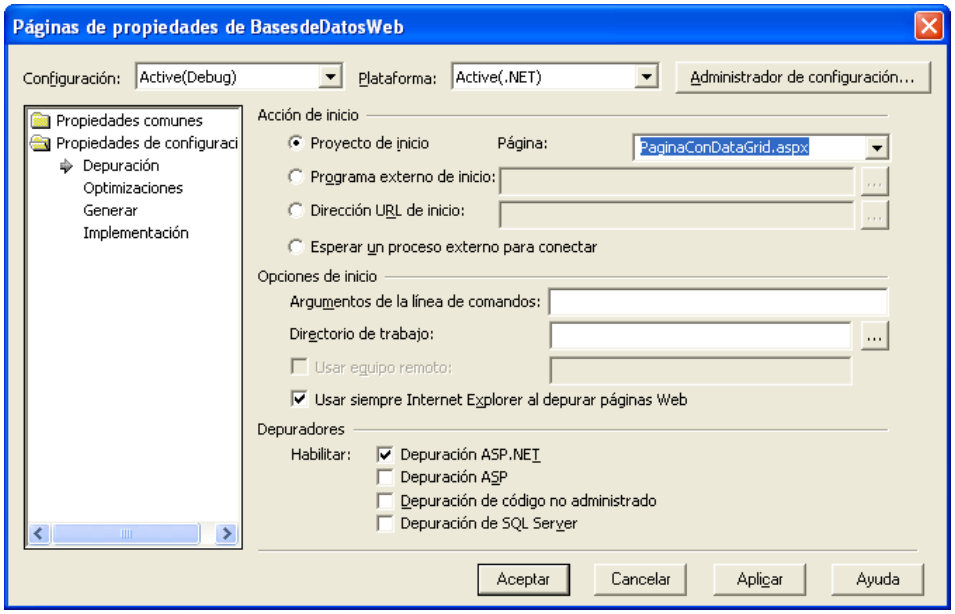
Figura 13.54 DataGrid con los campos de la tabla



- **Ejecutar la aplicación**

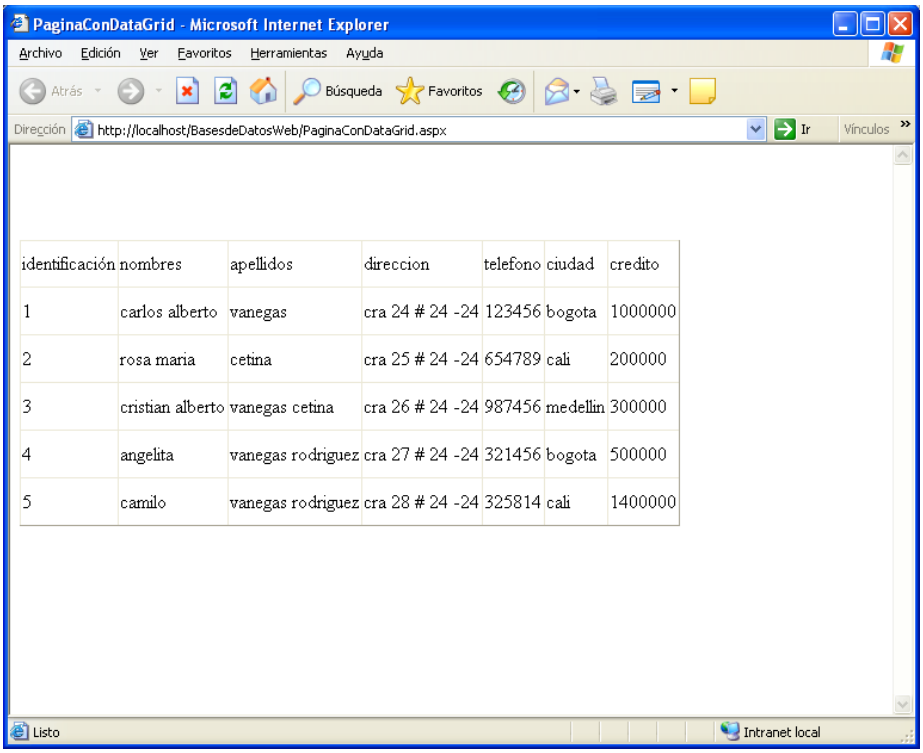
En el menú **Proyecto**, seleccione **propiedades de BasesdeDatosWeb,** en la opción **Propiedades de configuración** seleccione la página **PaginaConDataGrid.aspx** y pulse A**ceptar.**

Figura 13.55 Propiedades de la PaginaConDataGrid.aspx.



Haga clic en el botón Iniciar de la barra de herramientas estándar o presione **F5** para ejecutar el proyecto. Se visualizará la siguiente figura:

Figura 13.56. Ejecución de la PaginaConDataGrid.aspx.



identificación	nombres	apellidos	direccion	telefono	ciudad	credito
1	carlos alberto	vanegas	cra 24 # 24 -24	123456	bogota	1000000
2	rosa maria	cetina	cra 25 # 24 -24	654789	cali	200000
3	cristian alberto	vanegas cetina	cra 26 # 24 -24	987456	medellin	300000
4	angelita	vanegas rodriguez	cra 27 # 24 -24	321456	bogota	500000
5	camilo	vanegas rodriguez	cra 28 # 24 -24	325814	cali	1400000

13.5 Consultar dos o más tablas de una base de datos

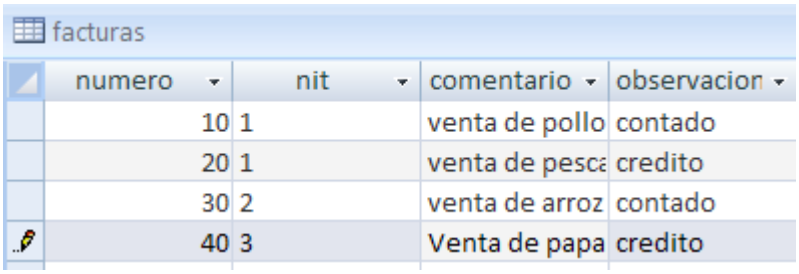
Hasta ahora se ha trabajado con una sola tabla de una base de datos. En el ejercicio que se realizará a continuación y siguiendo con la misma aplicación seleccionaremos dos tablas de la base de datos para consultarlas utilizando el asistente para formularios. Si no ha creado la tabla llamada **facturas** en la base de datos que se está trabajando cree la siguiente estructura de la nueva tabla:

Tabla 13.5 Estructura de la tabla Facturas.

Nombre del campo	Tipo de Dato	Longitud
Numero (llave principal)	numérico	
nit	Texto	13
comentario	Texto	25
observacion	Texto	25

La tabla con 4 registros quedará de la siguiente forma:

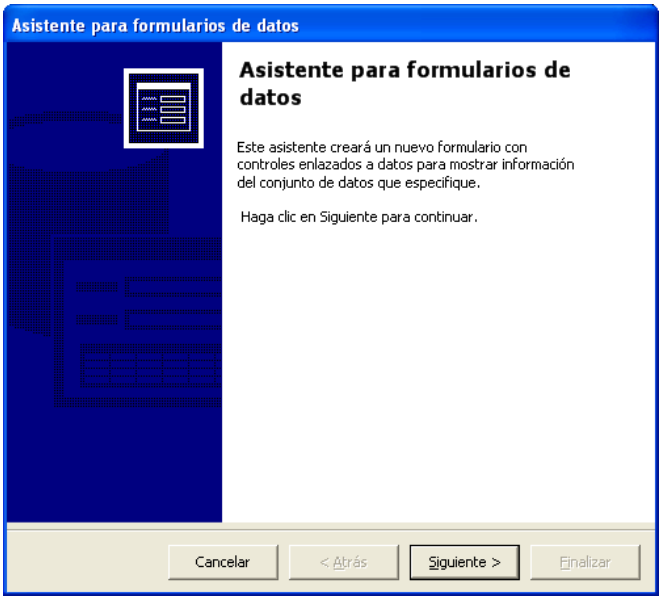
Figura 13.57 Tabla Facturas con registros.



- **Hacer la conexión a la base de datos**

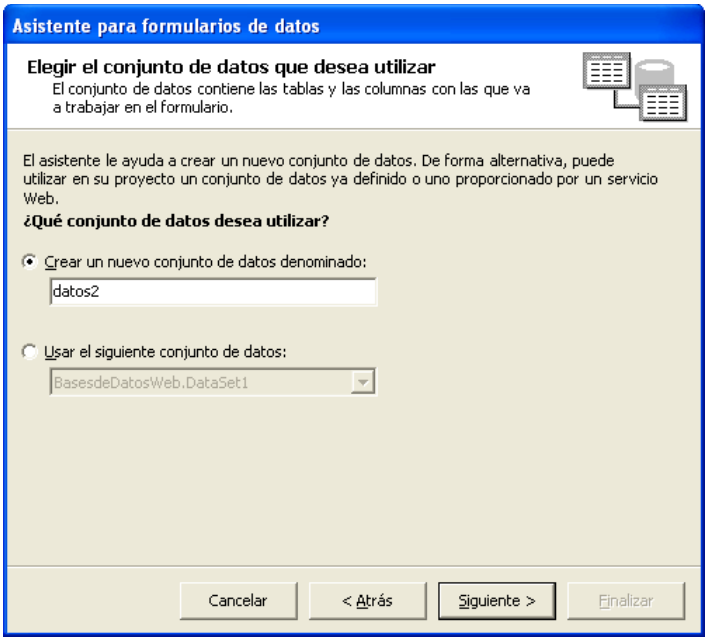
Seleccione la opción **agregar nuevo elemento** del menú **Proyecto** y seleccione el icono **Asistente para formularios de datos.** En esta ventana, en la opción **Nombre**, cambie el nombre de la página por **TablasRelacionadas**.vb. Al pulsar el botón **Abrir**, se visualizará la siguiente ventana:

Figura 13.58 Asistente para formulario de datos (TablasRelacionadas).



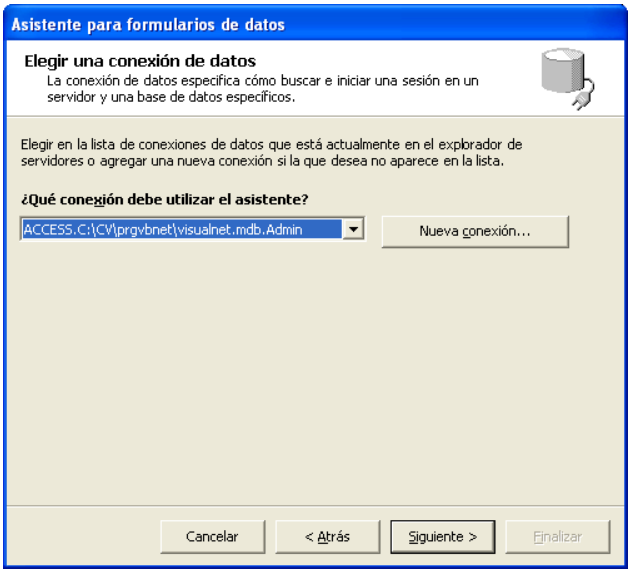
Esta pantalla es el inicio del asistente para formularios de datos, pulse el botón **Siguiente**> para visualizar la siguiente figura:

Figura 13.59 Elegir el conjunto de datos (tablasRelacionadas).



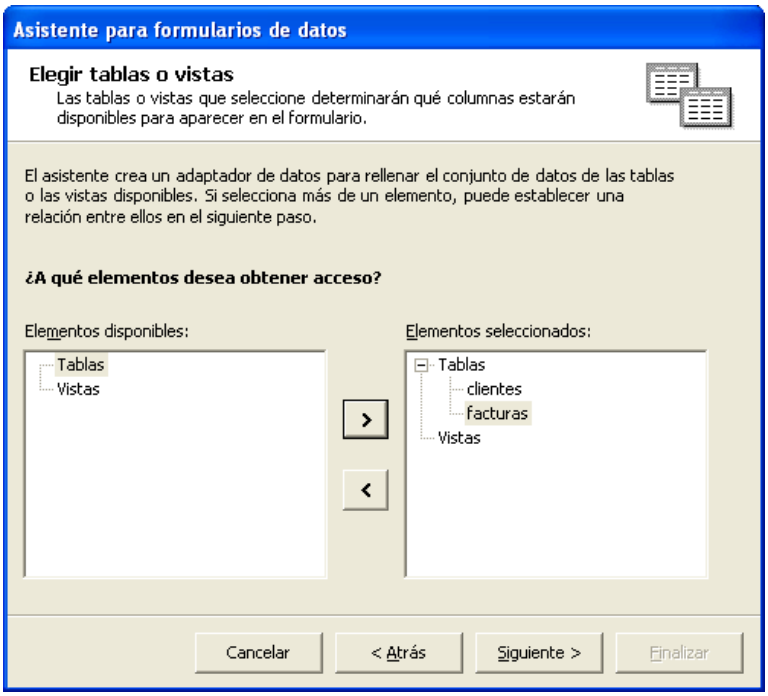
En esta ventana se selecciona el conjunto de datos que contiene las tablas y las columnas con las que va a trabajar la página. En la opción crear un nuevo conjunto de datos escriba **datos2** y pulse el botón **Siguiente**> para visualizar la siguiente ventana:

Figura 13.60 Elegir conexión de datos (TablasRelacionadas).



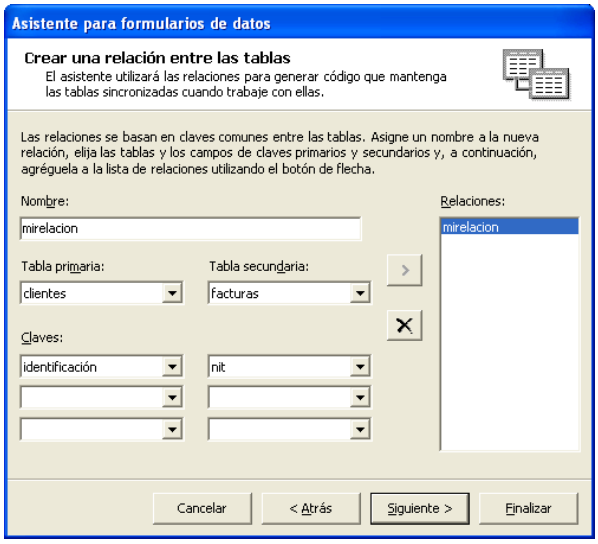
En esta ventana se elige la conexión específica, para el ejemplo seleccione la conexión que se realizo anteriormente y pulse el botón **Siguiente**> para visualizar la figura:

Figura 13.61 Elegir tablas o vistas (tablasRelacionadas).



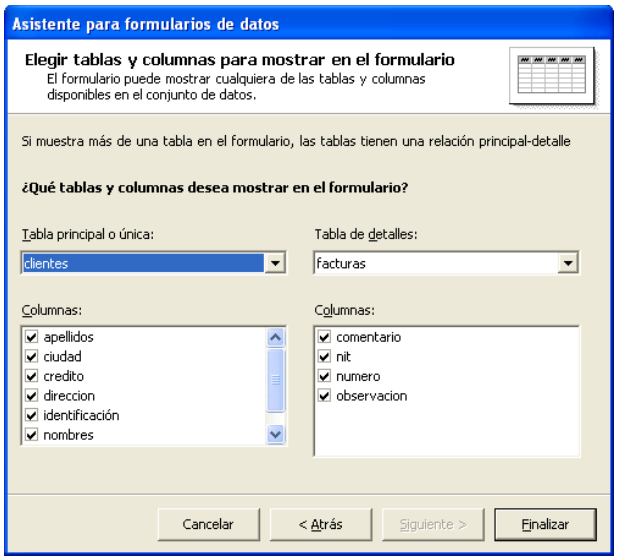
En esta ventana seleccione las tablas **clientes, facturas** y pulse el botón **">"** por cada tabla seleccionada para colocarla en la ventana **Elementos seleccionados**. Luego pulse el botón **Siguiente**> para visualizar la figura:

Figura 13.62 Crear una relación entre las tablas (TablasRelacionadas).



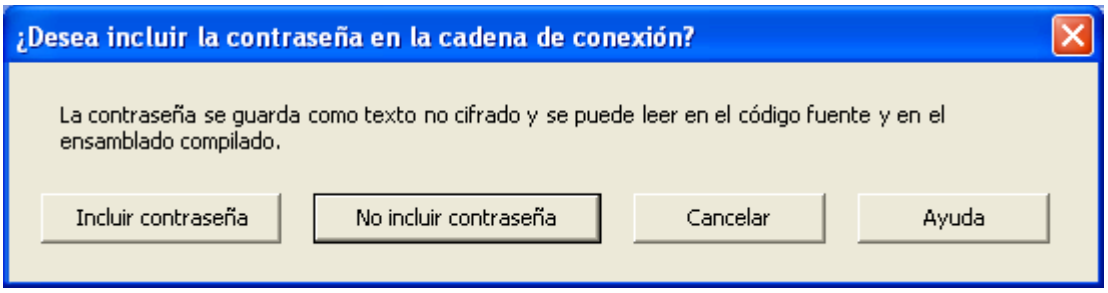
En esta ventana en el campo **Nombre** escriba **mirelacion**. Como tabla primaria seleccione la tabla **clientes** y como tabla secundaria la tabla **facturas,** después seleccione el campo **identificación** de la tabla **cliente**s como clave de la tabla primaria y **nit** de la tabla **facturas** como clave de la tabla secundaria. Por último pulse el botón **">"** para colocarla la relación de las tablas en la ventana **Relaciones**. Luego pulse el botón **Siguiente**> para visualizar la figura:

Figura 13.63. Elección de tablas y columnas (TablasRelacionadas)



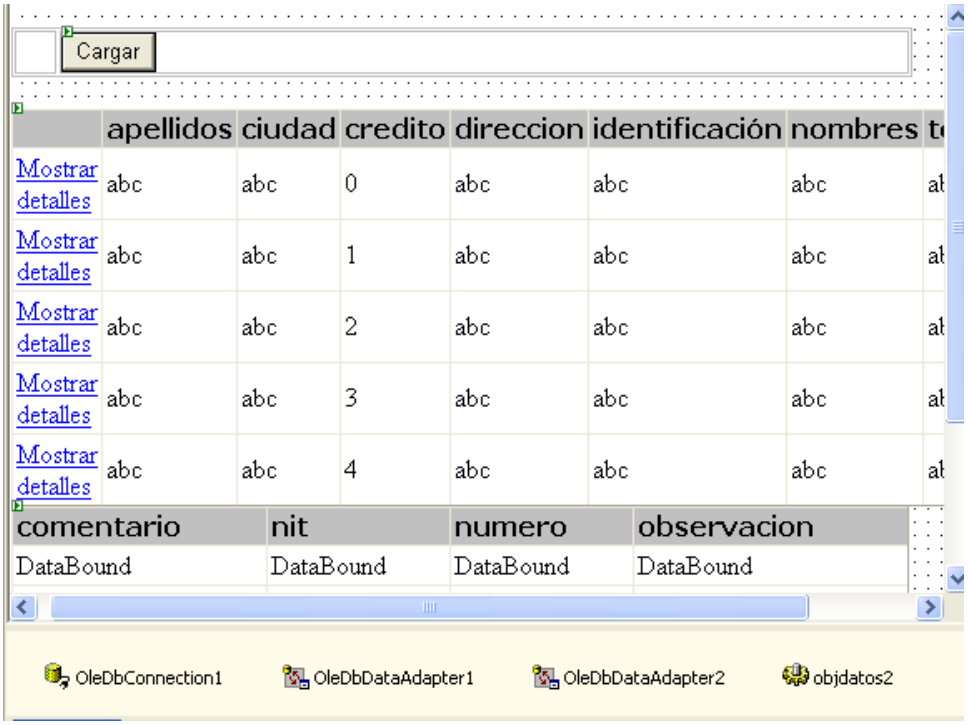
En esta ventana se debe seleccionar los campos que se quieren mostrar en la página. Al escoger cada uno de los campos pulse el botón **Finalizar** para ver la siguiente figura:

Figura 13.64 Ventana de contraseñas (TablasRelacionadas).



En este cuadro de diálogo el asistente pregunta si se desea incluir una contraseña o no. En este caso pulse el botón **No incluir contraseña**. A la aplicación se agregara debajo del formulario los objetos: OleDbDataAdapter2 y objdatos2. La figura que se visualizará será la siguiente:

Figura 13.65 Pagina final con tablas relacionadas.

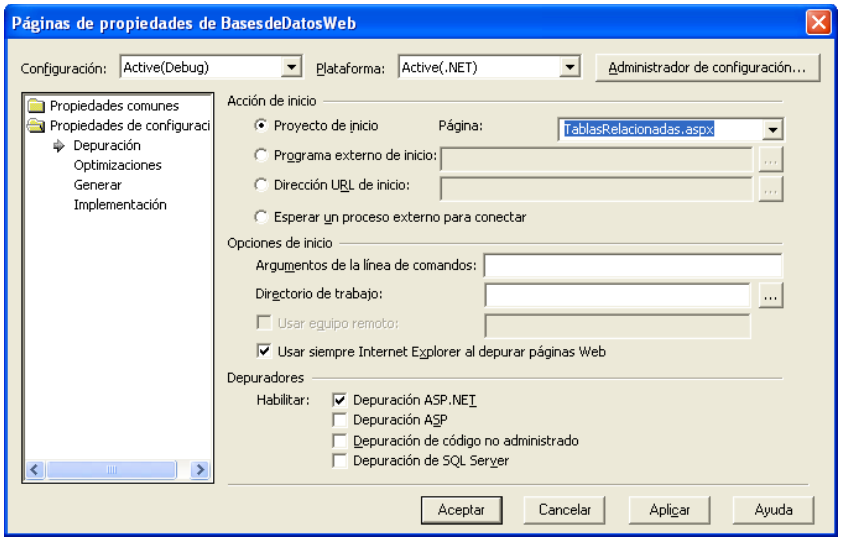


- **Ejecutar la aplicación**

Para ejecutar el proyecto en el entorno de desarrollo de visual Basic.NET se debe realizar lo siguiente:

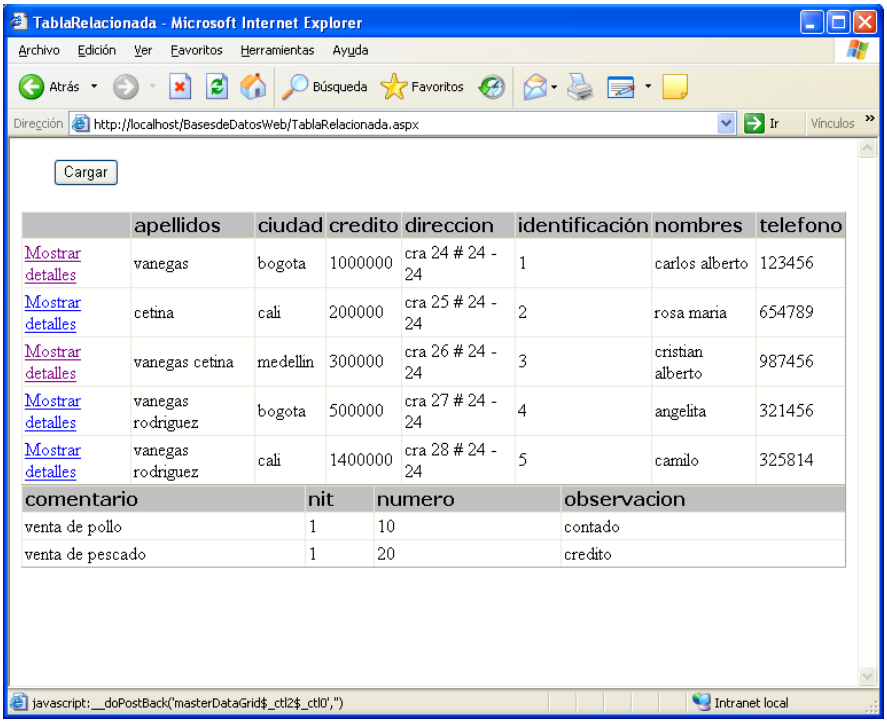
- En el menú **Proyecto**, seleccione **propiedades de BasesdeDatosWeb,** en la opción **Propiedades de configuración** seleccione la página **TablasRelacionadas.aspx** y pulse A**ceptar.**

Figura 13.66 Propiedades de BasesdeDatosWeb (TablasRelacionadas).



Haga clic en el botón Iniciar de la barra de herramientas estándar o presione **F5** para ejecutar el proyecto. Se visualizará la siguiente figura:

Figura 13.67 Ejecución de la BasesdeDatosWeb (pagina TablasRelacionadas).



	apellidos	ciudad	credito	direccion	identificación	nombres	telefono
Mostrar detalles	vanegas	bogota	1000000	cra 24 # 24 - 24	1	carlos alberto	123456
Mostrar detalles	cetina	cali	200000	cra 25 # 24 - 24	2	rosa maria	654789
Mostrar detalles	vanegas cetina	medellin	300000	cra 26 # 24 - 24	3	cristian alberto	987456
Mostrar detalles	vanegas rodriguez	bogota	500000	cra 27 # 24 - 24	4	angelita	321456
Mostrar detalles	vanegas rodriguez	cali	1400000	cra 28 # 24 - 24	5	camilo	325814

comentario	nit	numero	observacion
venta de pollo	1	10	contado
venta de pescado	1	20	credito

Al pulsar el botón **Cargar** se visualizarán los datos de la tabla **clientes**. Si se quiere mostrar los datos relacionados se debe pulsar en **Mostrar detalle** y si existen registros relacionados se visualizará la información de la tabla **facturas**.

14. SERVICIOS WEB

Un servicio Web es un servicio, con un interfaz definida y conocida al que se puede acceder a través de Internet. Igual que una página Web está definida por un URL (Uniform Resource Locator), un servicio Web está definido por un URI (Uniform Resource Identification) y por su interfaz a través del cual se puede acceder a él.

Los servicios Web se dividen en servicios de transporte (los protocolos del nivel más bajo que codifican la información independientemente de su formato y que pueden ser comunes a otros servicios), mensajería, descripción y descubrimiento. En la parte más baja se encuentran los servicios de transporte, que establecen la conexión y el puerto usado. Generalmente se usa HTTP (protocolo de la WWW), pero se puede usar también SMTP (protocolo del correo electrónico), FTP (File Transfer Protocol).

El protocolo más usado en esta capa es el SOAP. Este protocolo puede usar cualquiera de los transportes anteriores, se pueden escribir clientes y servidores en cualquier lenguaje y usa XML como lenguaje para especificar los mensajes.

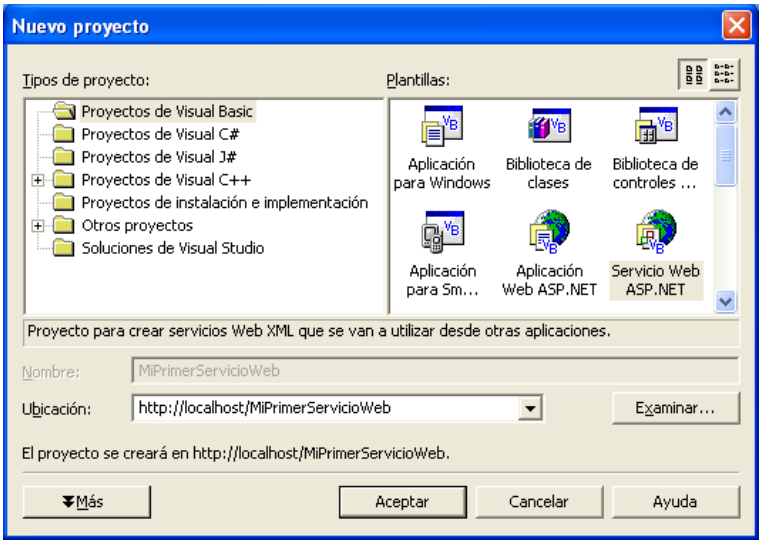
Los beneficios de los servicios Web son:

- Comunicación entre diferentes aplicaciones
- Reutilización del servicio
- Rápido desarrollo
- La distribución de la información en varios clientes

14.1 Creación de servicio web con Visual Basic .NET

La creación de servicios Web desde el entorno de Visual Basic.NET es similar a la creación de un proyecto Windows Forms o Web Forms. Entonces seleccione crear un nuevo proyecto y elija el icono **Servicio Web ASP.NET.**

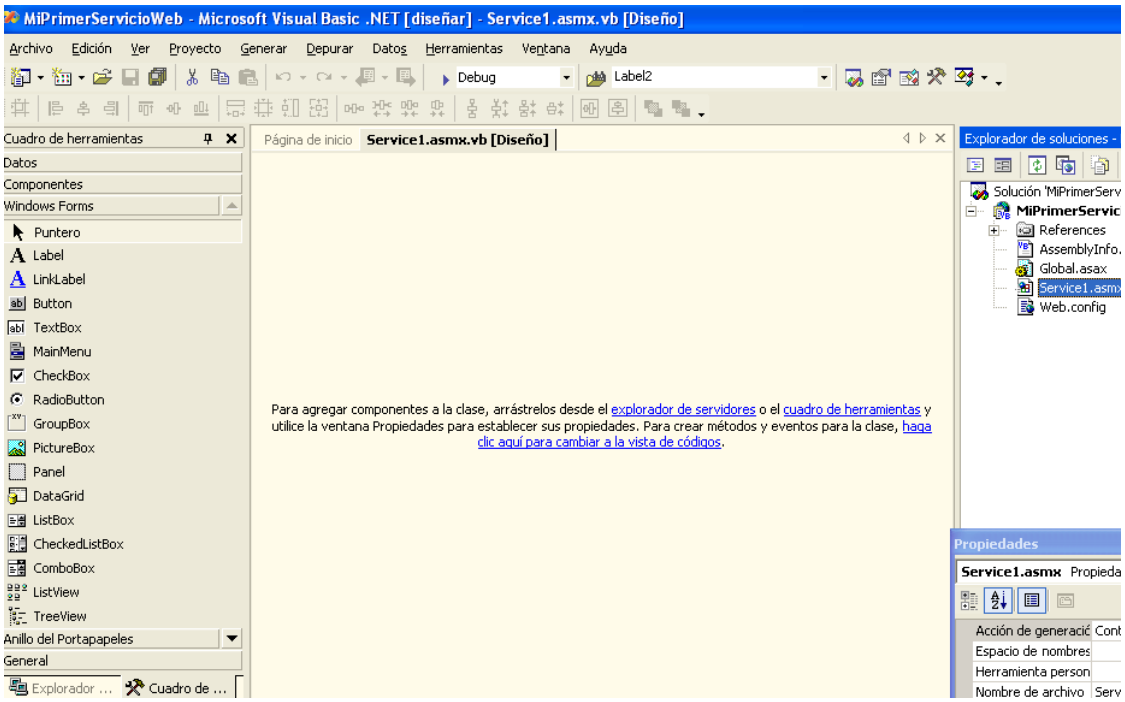
Figura 14.1 Ventana para la creación de un servicio Web desde Visual Basic.NET.



En el campoEn el En

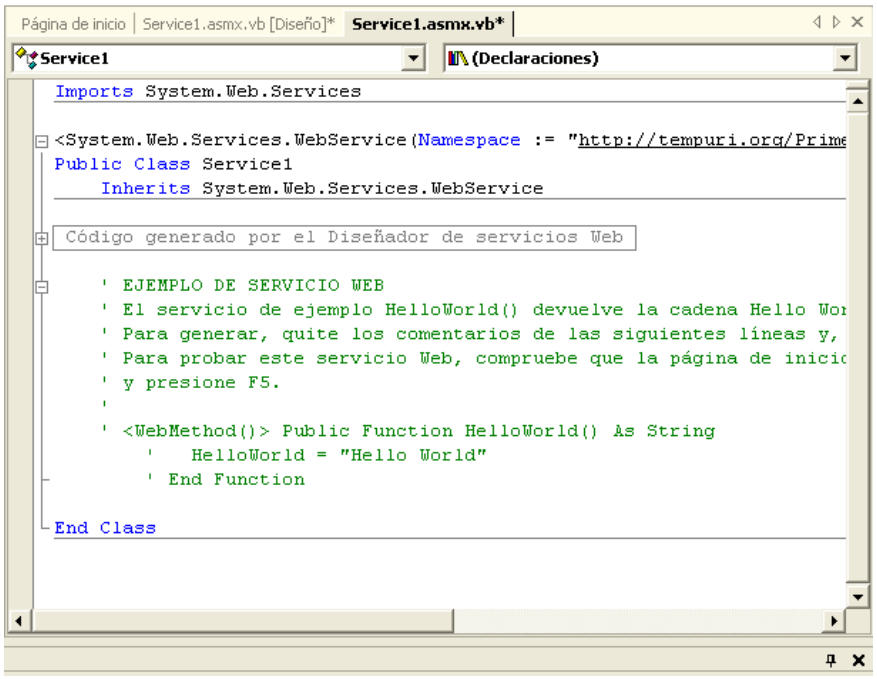
En el campo **Ubicación** cambie el nombre **Webservice1** por **MiPrimerServicioWeb** y pulse el botón **Aceptar** para visualizar la figura 14.2.

Figura 14.2 Página del servicio Web MiPrimerServicioWeb.



En el explorador de soluciones se visualizará el archivo **Service1.asmx** el cual define la clase que permitirá las operaciones que ofrecerá el servicio a sus clientes. Dé clic sobre el hipervínculo **haga clic para cambiar a la vista de código.** Se observará la siguiente figura:

Figura 14.3 Código del servicio Web MiPrimerServicioWeb.



El archivo **service1.asmx** importa el espacio de nombres **System.Web.services**, como también crea un espacio de nombres **NamesSpace** donde se define la dirección HTTP. Además se creará la clase **Service1** que hereda de **System.web.services.WebService** y dentro de está se definen cada una de las funciones que contendrá el servicio Web. Por omisión siempre se crea la función **HelloWord** ().

Las funciones de un servicio que estructuran de la siguiente manera:

<WebMethod ()>: Entre los paréntesis se puede hacer un comentario del servicio Web,

Como por ejemplo:

<**WebMethod** (Description:=“Servicio Web para consultar una base de datos”)>

La definición normal de una función es:

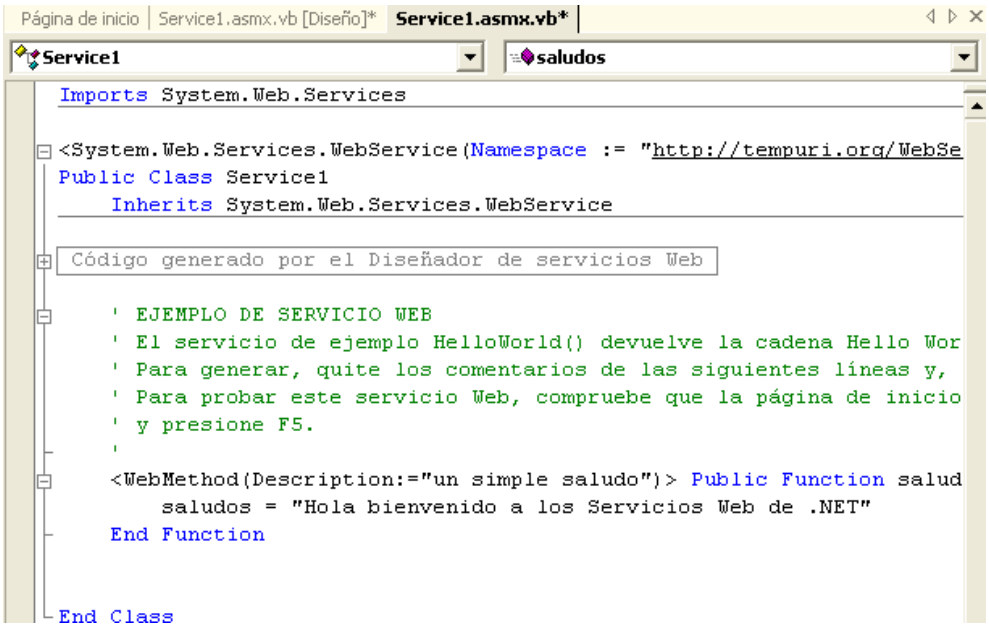
- Modificador de acceso: public, prívate
- Function: palabra reservada para declarar una función
- Nombre_de la funcion: HelloWord
- Lista de parámetros: ()
- Tipo de retorno: String
- Cuerpo de la función: Helloworld =”Hello World”
- Fin de la función : end Function

Ahora puede borrar la función HelloWord y crear una nueva función llamada saludos que contiene lo siguiente:

```
<WebMethod (Description:="un simple saludo")>
 Public Function saludos () As String
                    saludos = "Hola bienvenido a los Servicios Web de .NET"
 End Function
```

El servicio quedaría como se muestra en la figura 14.4.

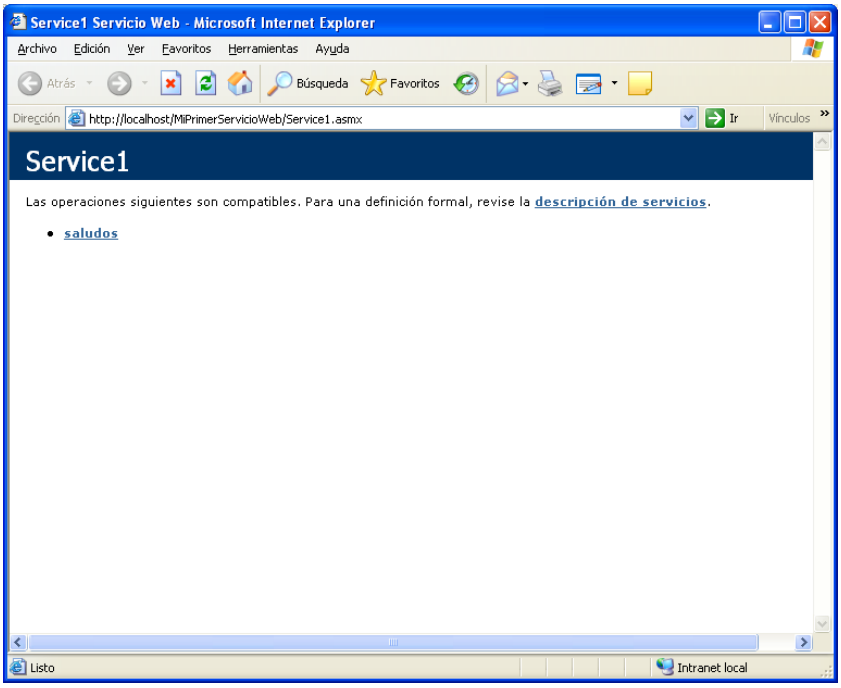
Figura 14.4 Servicio Web con la función saludos ().



- **Ejecutar el Servicio Web**

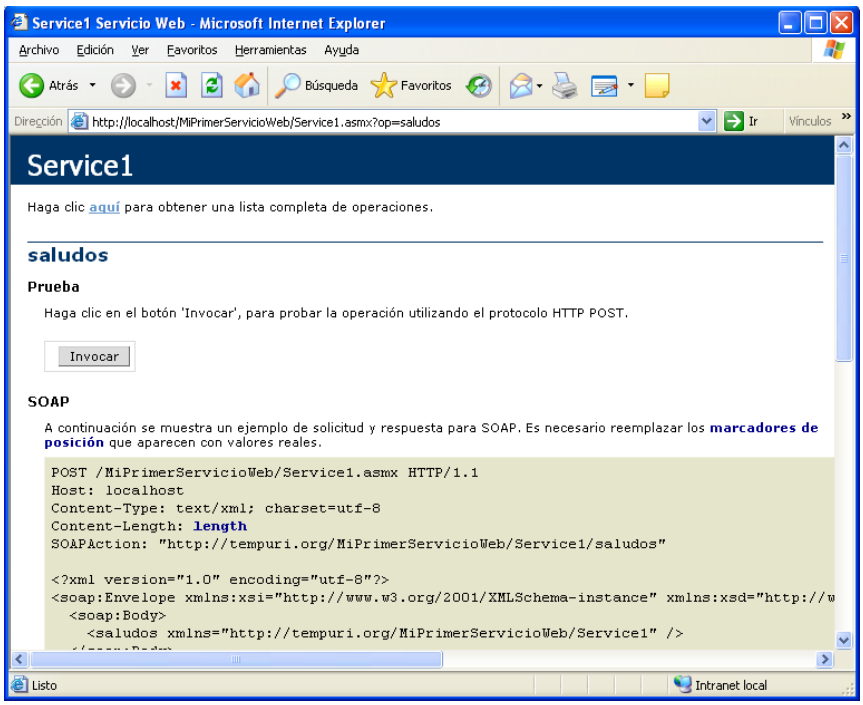
Haga clic en el botón Iniciar de la barra de herramientas estándar o presione **F5** para ejecutar el proyecto. Se visualizará la siguiente figura:

Figura 14.5 Ejecución del Servicio Web MiPrimerServicioWeb.



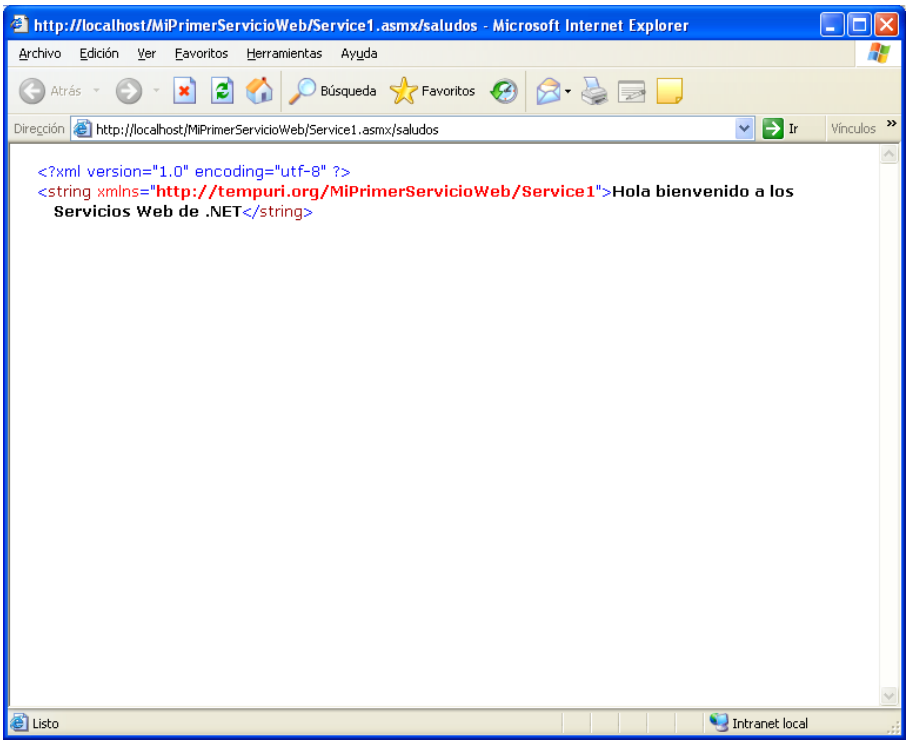
Haga clic sobre el hipervínculo **saludos**, para visualizar la siguiente figura:

Figura 14.6 Ejecución del hipervínculo saludos.



En esta ventana se debe hacer clic en el botón **Invocar** para probar la operación que realiza la función **Saludos.**

Figura 14.7 Resultado de la operación realizada por el servicio Web saludos.



El resultado arrojado por el servicio Web es el mensaje "**Hola bienvenido a los Servicios Web de .NET".**

14.2 Acceder a un servicio Web desde ASP.NET

Como se pudo apreciar crear un servicio Web es muy sencillo, ahora se creará un nuevo proyecto Web y desde dicho proyecto se consumirá un servicio Web que previamente debemos crear.

La aplicación deberá capturar dos números en cajas de texto y en una tercera caja de texto al pulsar un botón deberá visualizar el producto de los dos números. Dicha operación la debe realizar el servicio Web.

1) Crear el servicio Web

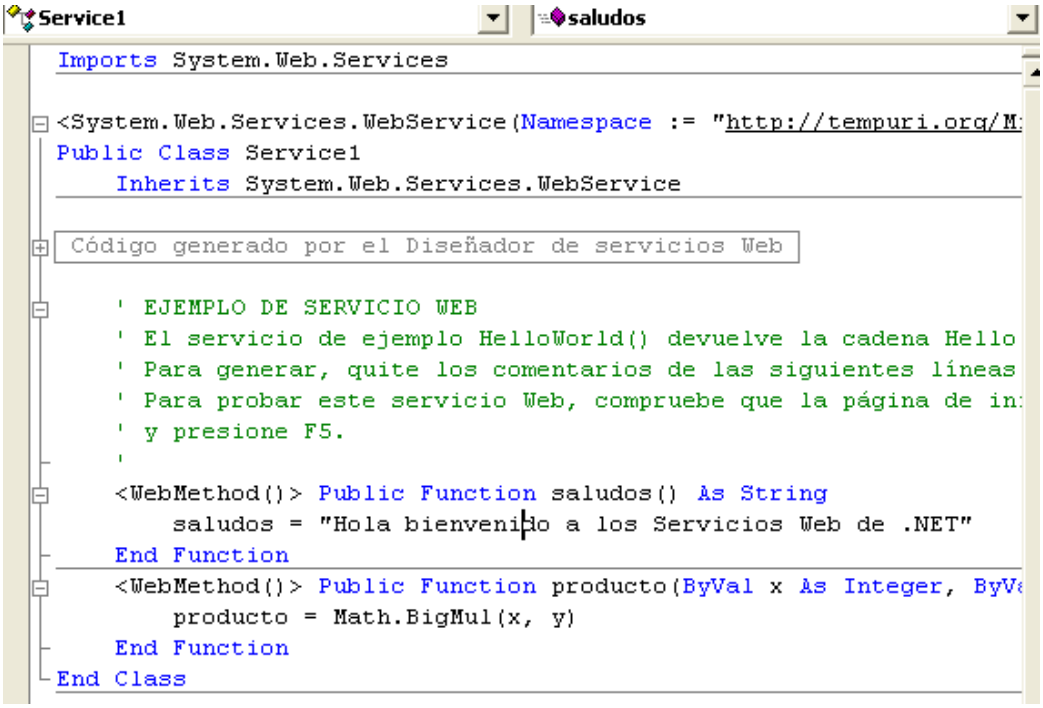
En el servicio que se creo anteriormente agregue una nueva función llamada **producto**, su estructura es:

```
<WebMethod ()> Public Function producto (ByVal x As Integer, ByVal y As Integer) As Integer
        producto = Math.BigMul(x, y)
End Function
```

En el código anterior se crea la función llamada **producto** que recibe como parámetros las variables **x** y **y** de tipo **Integer** y que retorna un valor de tipo **Integer**. Se utiliza para retornar el producto de los dos números el método **BigMul** () de la clase **Math** que permite realizar el producto de dos números.

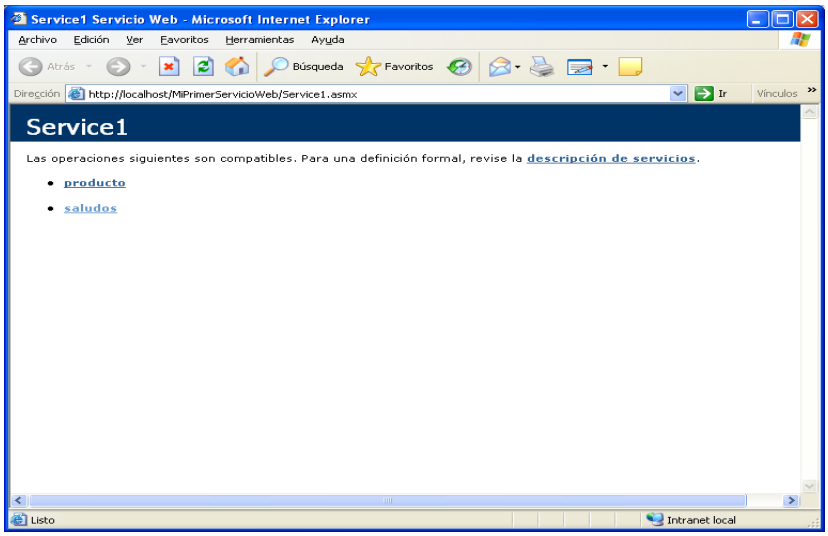
El servicio Web quedaría como se muestra en la siguiente figura:

Figura 14.8 Servicio Web con la función producto.



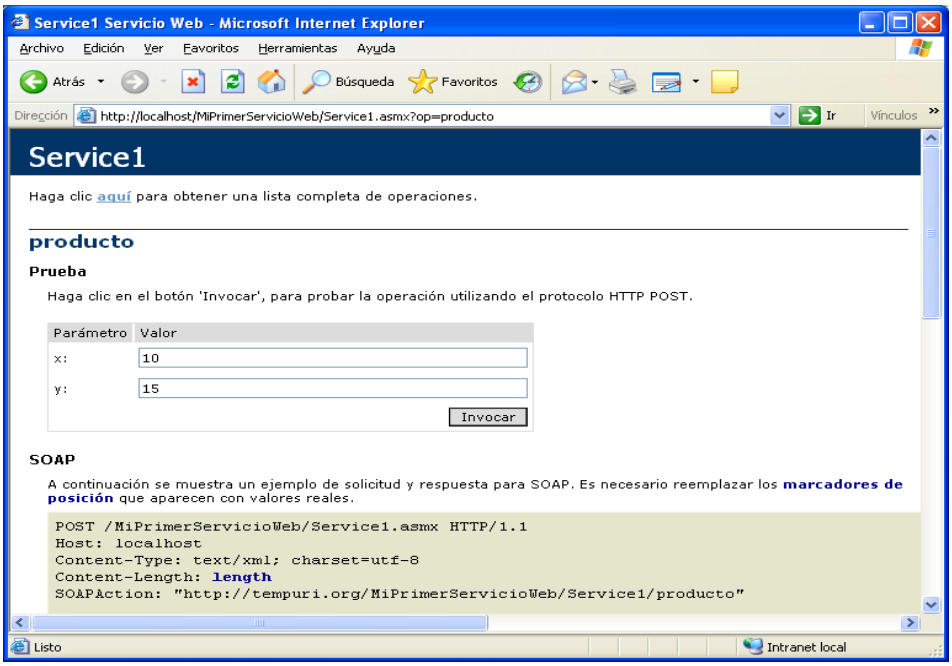
Al ejecutarse nuevamente el servicio Web se visualizarán las dos funciones como se aprecia en la figura 14.9.

Figura 14.9 Servicio Web con las funciones saludos y producto.



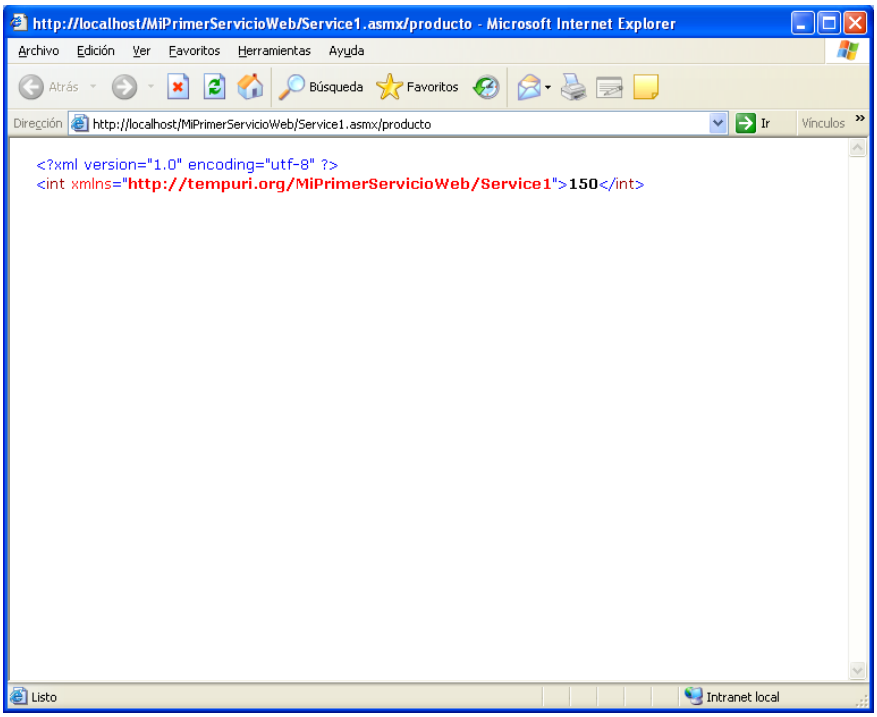
Al pulsar el hipervínculo producto se visualiza la siguiente figura:

Figura 14.10 Ejecución de la funcion producto.



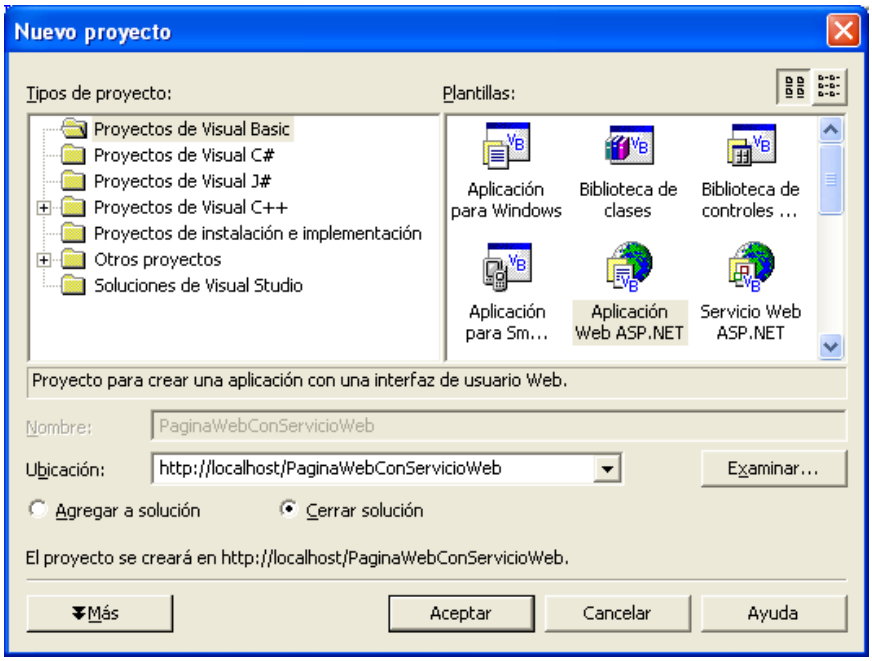
Si se digitan los números 10 y 15 respectivamente, al invocar el servicio se aprecia la siguiente figura:

Figura 14.11 Resultados del servicio Web (función producto).



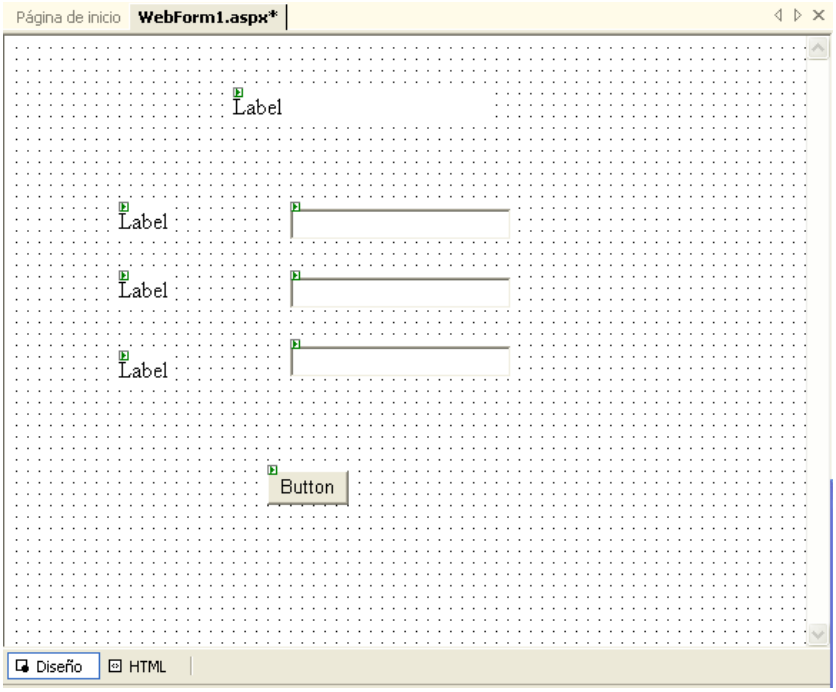
Ahora cree un nuevo proyecto Web y seleccione un **Aplicación Web ASP.NET.** En esta ventana, en la opción **Ubicación**, cambie el nombre de la página por **PaginaWebConServicioWeb**.

Figura 14.12 Proyecto Web PaginaWebConServicioWeb.



Al pulsar el botón **Aceptar** y al arrastrar 4 **Label**, 3 **TextBox** y un **Button** del cuadro de herramientas hacia la página se visualizará la siguiente ventana:

Figura 14.13 Pagina Web con controles (PaginaWebConServicioWeb).



Establezca las siguientes modificaciones en la propiedad correspondiente a cada uno de los siguientes controles:

Tabla 14.1 Propiedades de los controles (PaginaWebConServicioWeb).

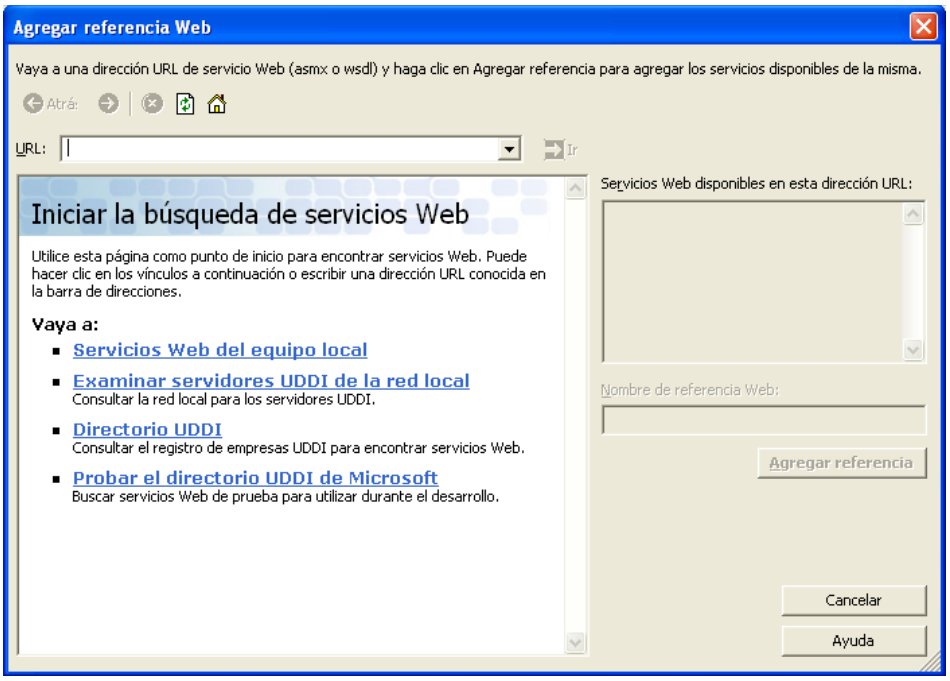
Control	Propiedad	Valor
Label1	(ID)	lbltitulo
	Text	Programa para Multiplicar
	Backcolor	Gris
	Font	Bold
	Size	Large
Label2	(ID)	lblmultiplicando
	Text	Multiplicando:
Label3	(ID)	lblmultiplicadro
	Text	Multiplicador:
Label4	(ID)	lblproducto
	Text	Producto:
TextField1	(ID)	txtmultiplicando
	Text	En blanco
TextField2	(ID)	txtmultiplicador
	Text	En blanco
TextField3	(ID)	txtproducto
	Text	En blanco
Button1	(ID)	botón
	Text	Enviar
Document	Title	Programa con Servicio Web

Al realizarse los cambios en las propiedades de los controles, se muestra la siguiente figura:

Figura 14.14 Pagina Web con los controles modificados.

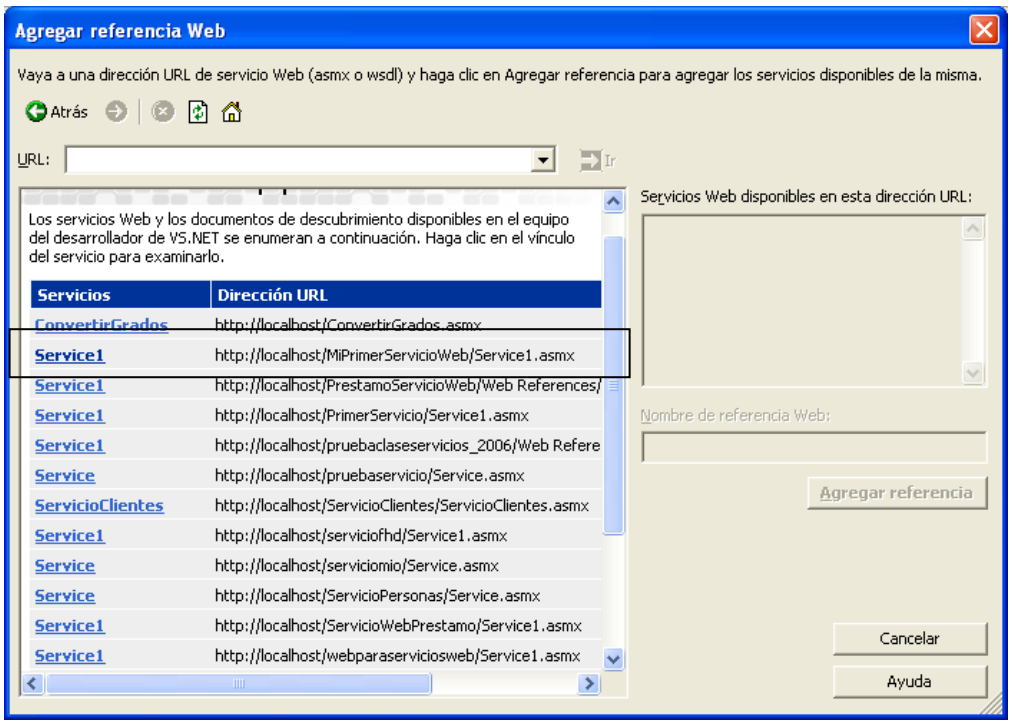
Ahora se debe agregar el servicio Web. Para esto en el menú **Proyecto** seleccione la opción **Referencia Web** para ver la siguiente figura:

Figura 14.15 Ventana para agregar un servicio Web.



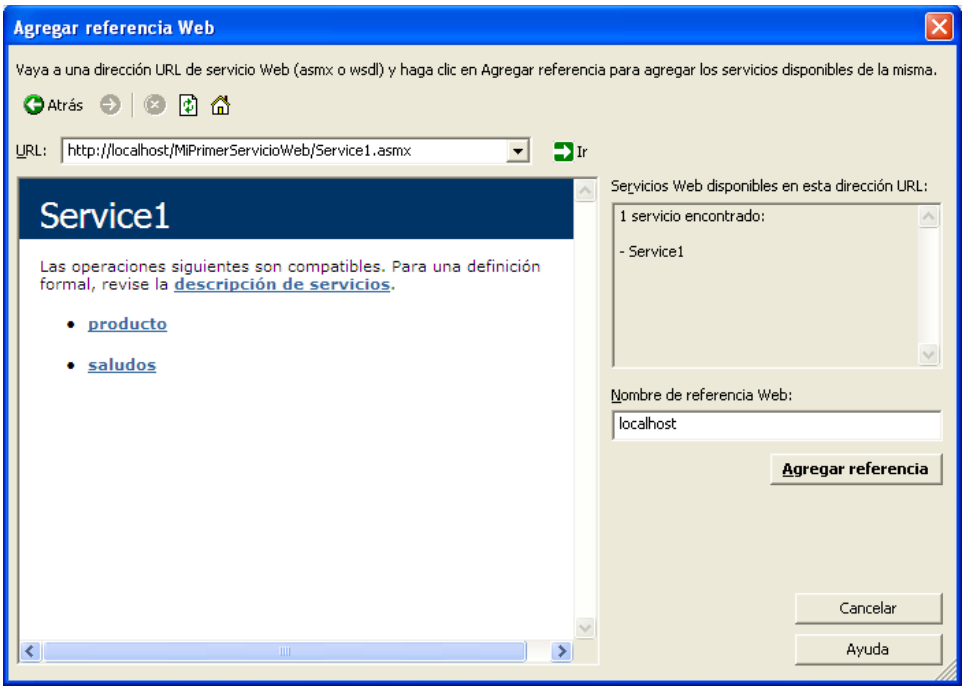
Haga clic sobre **Servicios Web del equipo local**, se visualiza una ventana similar a la siguiente figura (depende de los servicios que tenga el equipo local):

Figura 14.16 Ventana para seleccionar un servicio Web.



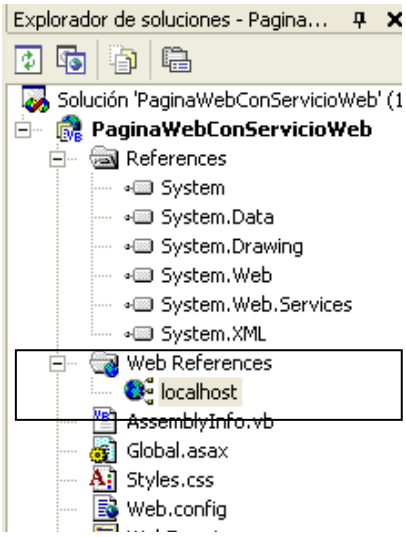
Seleccione el servicio llamado **MiPrimerServicioWeb**, se visualizará la siguiente figura:

Figura 14.17 Funciones del servicio Web MiPrimerServicioweb.



En esta ventana se muestran todas las funciones que tiene el servicio Web **MiPrimerServicioWeb**. En el campo **Nombre de referencia Web** aparece por omisión el texto **localhost** (cámbielo si desea), pulse el botón **Agregar referencia** para volver a la página Web. En el **explorador de soluciones** se podrá visualizar el servicio Web integrado al proyecto

Figura 14.18 Servicio Web integrado al proyecto.

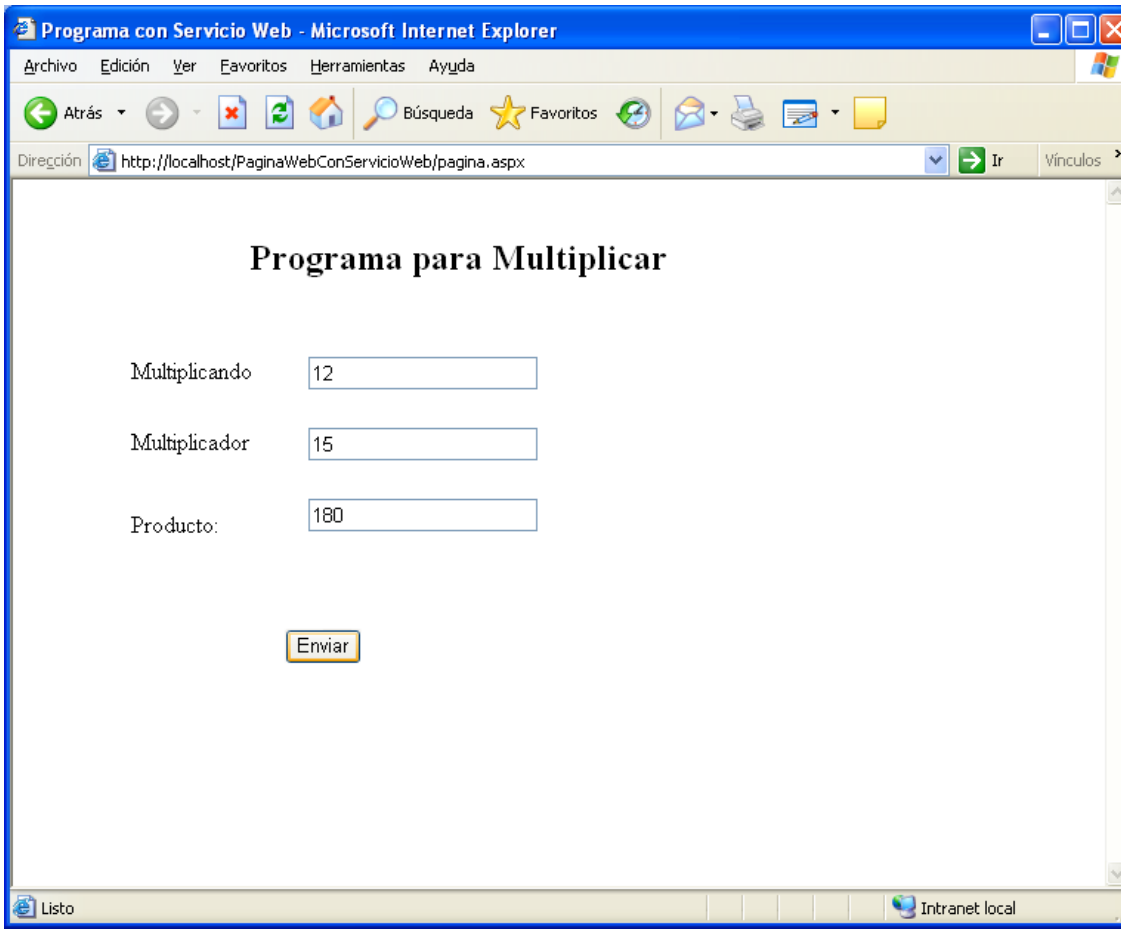


Ahora se deberá digitar el siguiente código en la ventana de código del botón **Enviar**

```
Dim servicio As New localhost.Service1
txtproducto.Text = servicio.producto(txtmultiplicando.Text, txtmultiplicador.Text)
```

Al ejecutarse el proyecto y digitar 12 y 15 en la respectiva caja de texto y pulsar el botón **Enviar** se llamara al servicio Web y se visualizará en la tercera caja de texto el producto de los dos números capturados.

Figura 14.19 Ejecución de la aplicación PaginaWebconServicioWeb.



ÍNDICE

BIBLIOGRAFÍA

- CEBALLOS, Javier. Microsoft Visual Basic .NET Lenguaje y aplicaciones. México, Editorial Alfaomega Ra-Ma. 2da edición, 2007.
- CHARTE, Francisco. Programación con Visual Basic .NET. México, Editorial Anaya, 2003.
- HARWRYSZKIEWYCZ, I T. Análisis y diseño de base de datos. Editorial Megabyte. Noriega Editores. México. 1994.
- JAMSA, Kris. Superutilidades para Visual Basic .NET. Editorial Mc Graw Hill, 1ra. Edición. España. 2003.

INFOGRAFÍA

- http://www.elquintero.net/Manuales.aspx?Cat=2&SubCat=6&jscript=true
- http://www.willydev.net/descargas/Cursos/vbnet/index.html
- http://www.dotnetspider.com/tutorials/AspNet-Tutorials.aspx
- http://www.programacionfacil.com/visual_basic_net/start
- http://www.programacion.com/asp/articulo/datosaspnet/
- http://www.es-asp.net/tutoriales-asp-net/tutorial-61-81/efectuando-acceso-a-datos.aspx
- http://www.es-asp.net/tutoriales-asp-net/tutorial-61-96/enlazando-a-bases-de-datos.aspx.
- http://msdn.microsoft.com/es-es/library/y8c0cxey.aspx
- http://msdn.microsoft.com/es-es/library/ya3sah92(VS.80).aspx
- http://msdn.microsoft.com/es-es/library/ff855828(VS.80).aspx
- http://msdn.microsoft.com/es-es/library/a127sbc5(VS.80).aspx
- http://msdn.microsoft.com/es-es/library/h974h4y2(VS.80).aspx
- http://social.msdn.microsoft.com/forums/es-ES/vbes/thread/087a5f2c-9eda-44a5-9a58-6008b65c9a8e
- http://www.recursosvisualbasic.com.ar/htm/tutoriales/datagrid-dbgrid.htm
- http://www.recursosvisualbasic.com.ar/htm/tutoriales/controles_visual_basic_menu.htm