

Cuaderno de Apuntes

Fundamentos de
Lenguaje de
Programación



www.aiep.cl

Estimado Estudiante de AIEP, en este Cuaderno de Apuntes, junto a cada Aprendizaje Esperado que se te presenta y que corresponde al Módulo que cursas, encontrarás **“Conceptos, Ideas Centrales y Aplicaciones”** que reforzarán el aprendizaje que debes lograr.

Esperamos que estas Ideas Claves entregadas a modo de síntesis te orienten en el desarrollo del saber, del hacer y del ser.

Mucho Éxito.-

Dirección de Desarrollo Curricular y Evaluación

VICERRECTORÍA ACADÉMICA AIEP

Módulo: FUNDAMENTOS DE LENGUAJE DE PROGRAMACION

PRIMERA UNIDAD: El lenguaje de programación Visual Basic.NET

1. Aprendizajes esperados: Identifican qué es la Programación Orientada a Objetos (POO) y las herramientas de la plataforma del lenguaje .NET disponibles para el desarrollo de proyectos.

- Un lenguaje de programación es un idioma artificial diseñado para expresar procesos que pueden ser llevados a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. (O'Reilly Media, Inc. (ed.): «Learning Python, Fourth Edition» (libro). O'Reilly 2011).

Diferencias entre lenguajes estructurados, orientados a objeto:

Estructurados	Orientado a Objetos
Es leído en secuencia	Fomenta la reutilización y extensión del código.
Programas más sencillos y más rápido de seguir en las pruebas, lo que implica reducción de esfuerzo en estas.	Permite crear sistemas más complejos.
Los programas quedan mejor documentados internamente.	Facilita la creación de programas visuales, agiliza el desarrollo de software.
Resuelve un problema de principio a fin en una sola estructura de código.	Facilita el mantenimiento del software.
Ejemplo de lenguajes estructurados son Pascal, Basic, Cobol, C.	Identifica sus objetos, con sus propiedades y acciones.
	Ejemplo de lenguajes orientados a objetos son vb.Net, Java, C#, J#, C++, Php, etc.

- Programación Orientada a Objetos (POO) plantea resolver problemas de la realidad a través de identificar objetos, y relaciones de colaboración entre ellos. Entendiendo que el objeto y sus mensajes son los elementos principales de esta.
- Un objeto es una representación abstracta de la realidad, la cual tiene un conjunto de datos llamados atributos, y una serie de funcionalidades que son llamadas métodos. Cada objeto posee una estructura, y es parte de una organización jerárquica o de otro tipo.

Ejemplo de ello es:



Objeto: Automóvil.

Atributos o propiedades: Modelo, Marca, Color, Marcha, Velocidad

Métodos o funcionalidades: Cambiar de marcha, frenar, acelerar

Otro ejemplo de ello es:



Objeto: Persona

Atributos o propiedades: Edad, Altura, Color de pelo

Métodos o funcionalidades: Cambiar color de pelo, Ver edad.

- Una clase es una abstracción de un objeto, es decir que la definición de un objeto es la clase. En programación al definir un objeto con sus características y funcionalidades, lo que hacemos es programar una clase. Entonces una clase es una clasificación, en base atributos y funcionalidades comunes.

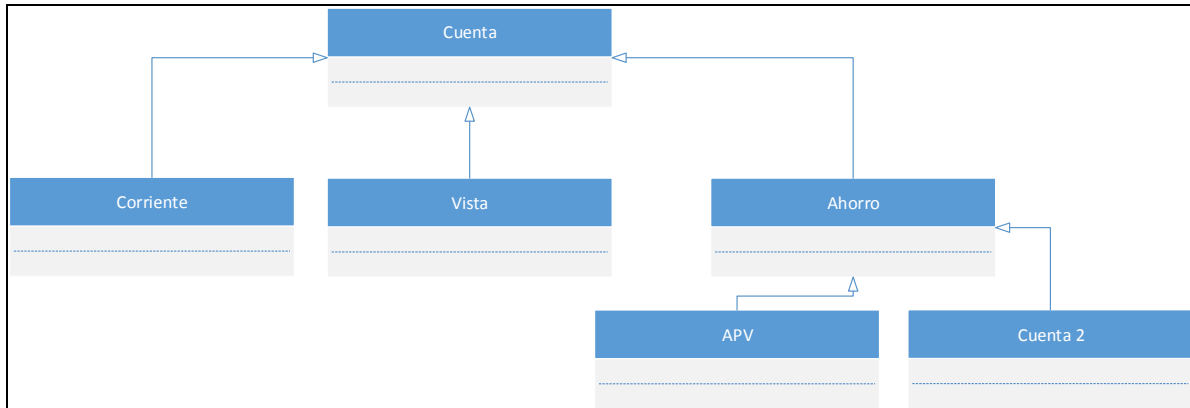
Los atributos toman estados, como el ejemplo del automóvil, el atributo color puede cambiar de rojo a azul.

Un método, que es la funcionalidad del objeto, modifica el comportamiento de este, como por ejemplo Cambiar de Marcha de Cuarta a Quinta.

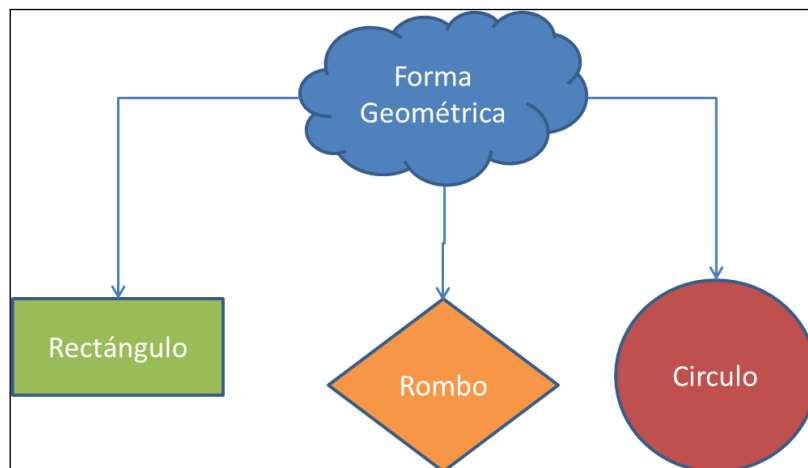
ACTIVIDAD: Crear los posibles objetos de un Taller de Bicicletas, con sus atributos y métodos.

- Instancia, en programación orientada a objetos es crear un objeto a partir de una clase.
- Herencia es un tipo de relación que consiste en que una clase puede heredar sus atributos y métodos a varias subclases. La clase que hereda es la superclase o clase padre, y la clase hija separadamente de poseer sus propios atributos y métodos, adiciona los atributos y métodos de la clase padre.

Ejemplo de esto es:



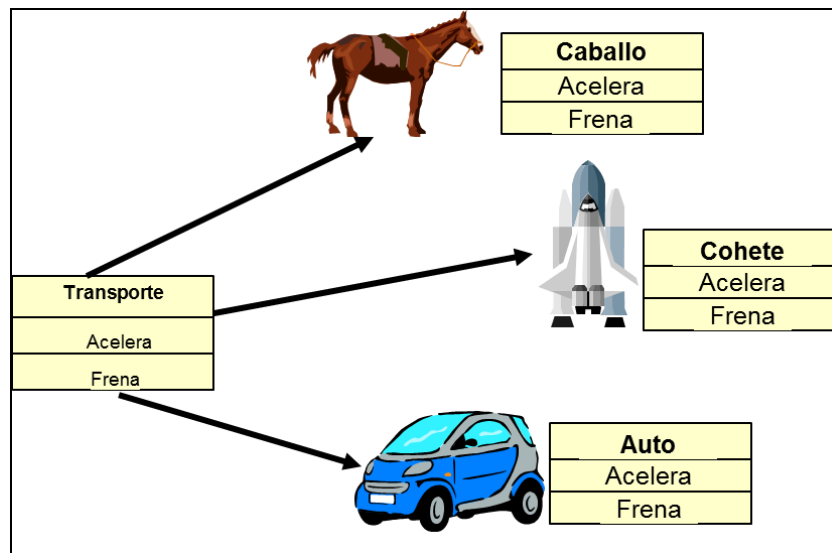
- Encapsulamiento es la ocultación del estado, esta característica es la que expresa la capacidad del objeto de responder a peticiones a través de sus métodos sin la necesidad de exponer los medios utilizados para llegar a brindar estos resultados en sus propiedades o atributos. Entonces por qué utilizar encapsulamiento, va por la facilidad para manejar la complejidad, ya que el encapsulamiento también es llamado “ocultamiento de la información”, esto asegura que los objetos no pueden cambiar el estado interno de otros objetos de maneras inesperadas, solamente los propios métodos internos del objeto pueden acceder a su estado.



- Abstracción permite no preocuparse de los detalles no esenciales. Es la capacidad de un objeto de cumplir sus funciones independientemente del contexto en que se lo utilice. Por ejemplo un objeto, automóvil siempre mostrará sus mismas propiedades o atributos, y dará los mismos resultados a través de sus eventos. Existe en casi todos los lenguajes de programación. Las estructuras de datos y los tipos de datos son un ejemplo de abstracción. Los procedimientos y funciones son otro ejemplo.

- Polimorfismo es la capacidad que tienen los objetos de una clase de responder al mismo mensaje o evento en función de los parámetros utilizados durante su invocación. En algunos lenguajes, el término polimorfismo es también conocido como ‘Sobrecarga de parámetros’ ya que las características de los objetos permiten aceptar distintos parámetros para un mismo método (diferentes implementaciones) generalmente con comportamientos distintos e independientes para cada una de ellas.

Existen 2 tipos de polimorfismo, dinámico y estático, el dinámico es el que el código no incluye ningún tipo de especificación sobre el tipo de datos; estático es el que los tipos que se aplica el polimorfismo deben ser explicitados y declarados uno por uno antes de utilizarlos.



- Que es el framework, según su traducción literal es un marco de trabajo. Este marco de trabajo, ofrece a quien lo utiliza, una serie de herramientas para facilitar la realización de determinada tarea.

Un framework puede estar compuesto por librerías de clases, documentación y ayuda, ejemplos, tutoriales e incluso foros de discusión. Es posible que se utilicen varios frameworks a la vez, o incluso que algunos sean soporte de otros.

- **.Net Framework** es un componente Windows que soporta el desarrollo y ejecución de aplicaciones Windows y Web Services. El propósito de este componente es proveer al usuario con un entorno de programación orientada a objetos consistente, donde el código pueda estar almacenado localmente o de forma remota. Versionado de software y promueve la ejecución de código seguro. El .NET Framework tiene dos componentes principales. La Common Runtime (CLR) y la Class Library (CLS).

- CLS es la librería de clases de .NET Framework, expone características en tiempo de ejecución y provee otros servicios útiles para todos los desarrolladores. Las clases simplifican el desarrollo basado en .NET. Los desarrolladores pueden extenderlas creando sus propias librerías de clases. Las librerías de clases base implementan el .NET Framework. Todas las aplicaciones (web, windows, web services) acceden a las mismas clases base. Estas están almacenadas en namespaces. Los diferentes lenguajes acceden a las mismas librerías.

- CLR simplifica el desarrollo de aplicaciones, provee un entorno de ejecución robusto y seguro, soporta varios lenguajes y simplifica el despliegue y la administración. La CLR es un entorno administrado (managed), en el cual los servicios comunes, como garbage collection y seguridad, son provistos automáticamente

- MSIL que es un conjunto de instrucciones independiente de la CPU que se pueden convertir de forma eficaz en código nativo. MSIL incluye instrucciones para cargar, almacenar, inicializar y llamar a métodos en los objetos, así como instrucciones para operaciones lógicas y aritméticas, flujo de control, acceso directo a la memoria, control de excepciones y otras operaciones.

2. Aprendizajes esperados: Implementan rutinas de acceso, búsqueda, validación y operaciones aritméticas en arreglos de diverso tipo.

RECORDAR: la herramienta de desarrollo VisualStudio en sus versiones 2008 o posteriores, para crear cualquier proyecto, se debe seleccionar en el Menú, la opción Archivo, luego seleccionar Nuevo Proyecto. Luego elegir lenguaje de programación, que en este caso es Visual Basic, y seleccionar el tipo de proyecto Windows (Consola, Windows Forms, etc), Web, entre otros.

Al crear un proyecto de tipo aplicación de consola, la herramienta VisualStudio por defecto crea la clase del programa con su método principal.

Donde la clase va a estar identificada con la palabra Module [Nombre Modulo], o bien al crear una clase como tal, se creara como "Public Class Class1", donde Public indica que es una clase no tiene restricciones, y puede ser utilizada por cualquier otra clase, Class indica que es de tipo Clase, Class1 es el nombre de la clase.

Las clases pueden ser públicas, privadas, protegidas, que indican lo siguiente:

- ⇒ Public: indica que es una clase no tiene restricciones, y puede ser utilizada por cualquier otra clase.
- ⇒ Private: es accesible sólo dentro de su contexto de declaración.
- ⇒ Protected: los elementos protegidos son sólo accesibles dentro de la clase, o de una clase derivada.

Ejemplo de clase y métodos en una aplicación de consola, en donde se identifica claramente la utilización de clase, con un método principal, y ejecución del código en el método.

```
Imports System

Module Module1 ← Nombre de la clase

    Sub Main() ← Método principal
        Console.Write("Cuál es su nombre?")
        Console.WriteLine("")
        Dim a As String = Console.ReadLine()
        Console.Write(a.ToUpper())
        Console.WriteLine(" Bienvenido!!!")
        Console.ReadLine()
    End Sub

End Module
```

Código a ejecutar en la aplicación

Un ejemplo simple de las operaciones básicas aritméticas en lenguaje vb.Net, ingresando 2 números por una aplicación de consola.

```
Ingrese primer valor
6
Ingrese segundo valor
4
El resultado de la suma es: 10
El resultado de la resta es: 2
El resultado de la multiplicación es: 24
El resultado de la división es: 2
El resto de la división es: 2
```

Método principal de la clase, en donde se instancian los otros métodos de esta.

```
Module Module1

    Sub Main()
        Console.WriteLine("Ingrese primer valor")
        Dim num1 As String = Console.ReadLine()

        Console.WriteLine("Ingrese segundo valor")
        Dim num2 As String = Console.ReadLine()

        Console.Write("El resultado de la suma es: ")
        Sumar(num1, num2)

        Console.Write("El resultado de la resta es: ")
        Restar(num1, num2)

        Console.Write("El resultado de la multiplicación es: ")
        Multiplicar(num1, num2)

        Console.Write("El resultado de la división es: ")
        Dividir(num1, num2)

        Console.Write("El resto de la división es: ")
        Resto(num1, num2)
        Console.ReadLine()
    End Sub
```

Instancia del método Restar

Instancia del método Sumar

Métodos en los cuales se realizan las operaciones básicas aritméticas en vb.Net, las cuales son llamadas desde el método principal.

```
Private Sub Sumar(num1 As String, num2 As String)
    Dim res As Integer = Int(num1) + Int(num2)
    Console.WriteLine(res)
End Sub

Private Sub Restar(num1 As String, num2 As String)
    Dim res As Integer = Int(num1) - Int(num2)
    Console.WriteLine(res)
End Sub

Private Sub Multiplicar(num1 As String, num2 As String)
    Dim res As Integer = Int(num1) * Int(num2)
    Console.WriteLine(res)
End Sub

Private Sub Dividir(num1 As String, num2 As String)
    Dim res As Integer = Int(num1) / Int(num2)
    Console.WriteLine(res)
End Sub

Private Sub Resto(num1 As String, num2 As String)
    Dim res As Integer = Int(num1) Mod Int(num2)
    Console.WriteLine(res)
End Sub
```

Las operaciones básicas aritméticas en vb.Net son:

- ⇒ Suma, representada por el símbolo +
- ⇒ Resta, representada por el símbolo –
- ⇒ Multiplicación, representada por el símbolo *
- ⇒ División, representada por el símbolo /
- ⇒ Resto de la división, representada por la palabra clave “Mod”

Operador	Símbolo	Ejemplo
Multiplicación	*	2 * 3
División	/	3 / 2
Resto	Mod	10 Mod 3
Suma	+	1 + 2
Resta	-	1 - 2
Concatenación	&, +	

3. Aprendizajes esperados: Operan con sentencias básicas del lenguaje .Net.

RECORDAR:

VARIABLE, es un espacio de almacenamiento de dato que cambia de acuerdo a su utilización en el programa.

CONSTANTE, es un espacio de almacenamiento que se mantiene en el programa.

Existen varias formas de declarar variables en vb.Net, pero siempre siguen la misma regla. Se utiliza la palabra reservada **Dim**, que indica que es un espacio de memoria; luego se da un nombre a la variable; en seguida se utiliza la palabra reservada **As**, que indica que tipo será esta; inmediatamente el tipo de variable, que pueden ser por ejemplo numérica, alfanumérica, etc.

Dim nombreVariable As Tipo	Sólo se está declarando
Dim nombreVariable As Tipo = Valor	Se declara y se asigna un valor para inicializarla.
Dim variable1, variable2, variable3 As Tipo	Se declara una serie de variables del mismo tipo

Ejemplos

⇒ Dim numeroEntero, otroEntero As Integer

⇒ Dim numeroPi As Double = 3.14159

⇒ Dim fecha As Date = "12/12/2005"

La declaración de constantes es de la siguiente forma:

⇒ **Const nombreConstante As Tipo = Valor**

Donde Const indica que es de tipo constante; luego se identifica el nombre de la constante; en seguida la palabra reservada As que indica de que tipo será la constante; inmediatamente se indica el tipo de dato que será la constante.

Ejemplo

⇒ Const constante1 As Integer = 10

Los tipos de datos que se pueden utilizar en vb.Net, para declaración de variables y constantes.

Tipo de dato en VB.NET	Tipo Framework .NET	Tamaño
Boolean	System.Boolean	1 byte
Byte	System.Byte	1 byte
Char	System.Char	2 bytes
Date	System.DateTime	8 bytes
Decimal	System.Decimal	12 bytes
Double	System.Double	8 bytes
Integer	System.Int32	4 bytes
Long	System.Int64	8 bytes
Short	System.Int16	2 bytes
Single (punto flotante con precisión simple)	System.Single	4 bytes
Object	System.Object	4 bytes
String (cadena de longitud variable)	System.String	10 bytes + (2 * longitud de la cadena)

Los operadores de relación en vb.Net permiten evaluar las relaciones (igualdad, mayor, menor, etc) entre un par de operando, o expresiones de comparación. Siempre devuelven un verdadero o falso (True o False). Estos son:

Operador	Símbolo
Menor que	<
Menor o igual que	<=
Mayor	>
Mayor o igual que	>=
Igual	=

Distinto	<>
----------	----

Los operadores lógicos evalúan que se cumpla o no una cierta condición, a lo cual devuelven un resultado. Estos son:

Operador	Símbolo	Ejemplo
Y lógico	And	
O lógico	Or	
No Lógico	Not	
Y lógico avanzado	AndAlso	Si la primera expresión es verdadera evalúa la siguiente
O lógico avanzado	OrElse	Si la Primera expresión es falsa evalua la Siguiente

En el siguiente ejemplo podemos ver:

```
Sub Main()
'Declaración de varias variables del mismo tipo en una sólo fila
Dim intUno, intDos, intTres As Integer

'Declaración de una variable, asignándole un valor en la línea de declaración
Dim nombre As String = ""

'Declaración de una constante
Const saludo As String = "Gracias!"

Console.Write("Hola!, diganos su nombre: ")
nombre = Console.ReadLine 'Asignación de lo escrito en consola en la variable nombre

Console.Write(nombre & " indiquenos sus 3 números favoritos, primero: ")
intUno = Console.ReadLine 'Asignación de lo escrito en consola en la variable de tipo entero
Console.Write("Segundo número: ")
intDos = Console.ReadLine 'Asignación de lo escrito en consola en la variable de tipo entero
Console.Write("Tercer número: ")
intTres = Console.ReadLine 'Asignación de lo escrito en consola en la variable de tipo entero

Console.Write(nombre & " indiquenos sus fecha de nacimiento: ")
Dim fec_Nac As Date = Console.ReadLine 'Declaración y asignación de un valor de lo escrito en consola

'Declaración de variable y asignándole el resultado de una operación aritmética.
Dim prom As Integer = (intUno + intDos + intTres) / 3
Dim mult As Integer = intUno * intDos * intTres

Console.WriteLine("El promedio de sus números es: " & prom) 'Concatenación de una cadena de caracteres con una variable
Console.WriteLine(saludo & " por entregarnos sus datos!")
Console.ReadLine()
End Sub
```

Utilización de sentencias condicionales Si entonces – Sino; Según haga

RECORDAR:

Un condicional es una estructura de se puede ejecutar o no en función del valor de una condición.

- Estructura condicional Si Entonces – Sino

Cuando se quiere ejecutar un bloque de código de acuerdo a una condición o varias (condiciones anidadas), entonces se utiliza la estructura If ... Then ... Else, la estructura es la siguiente:

Decisión

If condiciones1 **Then**

sentencias1 // Si la condición a evaluar es verdadera

Elseif condicion2 **Then**

sentencias2 // Si la primera condición es falsa, y si la segunda condición a evaluar es verdadera

Else

sentencias3 // Si las anteriores condiciones no se cumplen se ejecuta este código por defecto

End If // Termina el condicional

Estos condicionales pueden evaluar dos condiciones en un IF, utilizando los operadores lógicos, además se pueden utilizar operadores matemáticos, por lo tanto la secuencia en que se evalúan es, operadores lógicos y luego los operadores aritméticos, la estructura es la siguiente:

If condición1 **And** condición2 **Then** // Se evalúa si se cumplen ambas condiciones

Sentencias 1 // Se ejecuta si se cumplen ambas condiciones

Elseif condicion1 **OR** condición2 **then** // Se evalúa si se cumple alguna de las dos condiciones

Sentencias 2 // Se ejecuta si se cumple alguna de las dos condiciones

Else

Sentencias3 // Si las anteriores condiciones no se cumplen se ejecuta este código por defecto

End If

Un ejemplo de ello es:

```
Sub numMayor()  
    Console.Write("Ingrese dos números: ")  
  
    Dim numero1 As Integer = Console.ReadLine  
    Dim numero2 As Integer = Console.ReadLine  
  
    'Condicion 1  
    If numero1 > numero2 Then  
        'Sentencia 1  
        Console.WriteLine("El número 1 es mayor que el número 2")  
    'Condicion 2  
    ElseIf numero2 < numero2 Then  
        'Sentencia 2  
        Console.WriteLine("El número 1 es mayor que el número 2")  
    Else  
        ' Se ejecuta si no se cumplen las dos condiciones anteriores  
        Console.WriteLine("Son iguales")  
    End If  
End Sub
```

Ejemplo de condicional utilizando operadores lógicos:

```
Sub condicionLogicos()  
    Dim numero As Integer = 74  
  
    'Evalúa dos expresiones, y se deben cumplir ambas para que se ejecute la sentencia  
    If numero > 0 And numero < 50 Then  
        Console.WriteLine("El número es mayor que 0 y menor que 50")  
    'Evalúa dos expresiones, y se deben cumplir alguna de las dos para que se ejecute la sentencia  
    ElseIf numero >= 50 Or numero <= 100 Then  
        Console.WriteLine("El número es mayor o igual a 50 ó el número es menor o igual a 100")  
    End If  
End Sub
```

- Estructura condicional Según, se ejecuta uno de varios grupos de instrucciones, según el valor de una expresión, la estructura es la siguiente:

Select Case nombreVariable

 Case Valor1

 sentencias1

 Case Valor2

 sentencias2

 Case Else // Por omisión

 sentencias

End Select

Ejemplo de esta estructura es la siguiente:

```
Sub diaSemana()  
    Dim dia As Integer = 2  
  
    Select Case dia  
        Case 1 'Si el numero es 1  
            Console.WriteLine("Lunes")  
        Case 2 'Si el numero es 2  
            Console.WriteLine("Martes")  
        Case 3 'Si el numero es 3  
            Console.WriteLine("Miercoles")  
        Case 4 'Si el numero es 4  
            Console.WriteLine("Jueves")  
        Case 5 'Si el numero es 5  
            Console.WriteLine("Viernes")  
        Case 6 'Si el numero es 6  
            Console.WriteLine("Sábado")  
        Case 7 'Si el numero es 7  
            Console.WriteLine("Domingo")  
    End Select  
End Sub
```


Otro ejemplo

```
Sub condicionSegun()  
    Dim numero As Integer = 78  
  
    'Evalua si el numero cumple alguna de las siguientes condiciones  
    Select Case numero  
        Case 1 To 10 'Si esta entre esos valores  
            Console.WriteLine("Entre 1 y 10")  
        Case 11 To 20  
            Console.WriteLine("Entre 11 y 20")  
        Case 21 To 30  
            Console.WriteLine("Entre 21 y 30")  
        Case 31 To 40  
            Console.WriteLine("Entre 31 y 40")  
        Case 41, 42 ' Si es igual a alguno de los valores indicados  
            Console.WriteLine("Igual a 41 o 42")  
        Case Else  
            Console.WriteLine("Ninguna de las anteriores")  
    End Select  
End Sub
```

Ejercicios

1. Crear una aplicación de consola que permita ingresar dos números y calcule las operaciones básicas sobre ellos.
2. Crear una aplicación de consola en donde se ingrese día, mes y año de nacimiento y evalúe si es mayor de edad, e indique la edad de la persona.

4. Aprendizajes esperados: Usan bucles y arrays de acuerdo a requerimientos y sintaxis del lenguaje.

RECORDAR:

Los ciclos de repetición son estructuras cíclicas o repetitivas que ejecutara un bloque de código o instrucciones, tantas veces que sea necesario hasta que la condición se cumpla.

- Las estructuras de bucles de Visual Basic permiten ejecutar una o varias líneas de código de forma repetitiva. Puede repetir las instrucciones de una estructura de bucles hasta que una condición sea True, una condición sea False, un número de veces especificado o una vez para cada objeto de una colección (MSDN Español).
- La estructura **Para** ejecuta un número fijo de repeticiones, y utiliza una variable denominada contador, que controla las veces en que se repetirá el bucle. Además de especificar hasta donde se repetirá, y la cantidad en la que se incrementará el contador entre una repetición y otra.

La nomenclatura para este ciclo es:

For contador = valorInicial **To** valorFinal [**Step** paso]

<sentencias>

Next

Ejemplo simple del uso del ciclo For

```
Sub cicloSimple()  
    Dim i, res As Integer 'Declaración de variables  
  
    'Ciclo de repetición en que la variable i inicia en 1,  
    'y se repetirá hasta 5  
    For i = 1 To 5  
        res += i 'Suma a la variable res el valor de i  
    Next  
  
    Console.WriteLine(res)  
End Sub
```

Otro ejemplo con incrementación con Step, en donde la palabra clave Step indica que la variable i incrementará su valor en 5 hasta llegar a 50.

```
Sub cicloCinco()  
    Dim i As Integer 'Declaracion variable tipo entero  
  
    'Ciclo de repetición  
    For i = 0 To 50 Step 5 'Incrementación en 5  
        Console.WriteLine(i)  
    Next  
End Sub
```

- Estructura de repetición Mientras, la construcción **While...End While** ejecuta un conjunto de instrucciones mientras la condición especificada en la instrucción While sea verdadera. La sintaxis es la siguiente:

While condición

Sentencias

End While

Ejemplo básico de cómo utilizar While

```
Sub mientrasBs()  
    Dim i As Integer ' Declaración variable  
  
    'Ciclo mientras,  
    'que indica que se repitá mientras i sea menor a 6  
    While i < 6  
        i += 1 'incremento en 1 de la variable i por cada repetición  
    End While  
End Sub
```

Otro ejemplo de uso de While

```
Sub mientrasSalida()  
    Dim index As Integer = 0 'Declaración de variable  
  
    'Repetición del ciclo mientras la variable index sea menor a 10000  
    While index < 10000  
        index += 1  
  
        ' Si la variable index está entre 5 y 7  
        'continua con la proxima repetición  
        If index >= 5 And index <= 8 Then  
            Continue While  
        End If  
  
        ' Muestra el valor de la variable index.  
        Console.Write(index.ToString & " ")  
  
        ' Si la variable index es 10, sale del ciclo de repetición  
        If index = 10 Then  
            Exit While 'Salida del ciclo de repetición  
        End If  
    End While  
End Sub
```

- Estructura de repetición **Do ... Loop**, permite probar una condición al comienzo o al final de una estructura de bucle. También puede especificar si repite el bucle mientras la condición sigue siendo verdadera o hasta que se convierta en verdadera. La sintaxis es la siguiente:

Do While | Until Condición
 Sentencias

Loop

O bien

Do
 Sentencias
Loop While | Until Condición

Ejemplo de uso de Do ... Loop

```
Sub repLopp()  
    Dim i As Integer 'Declaración de variable  
  
    'Evalua la condición  
    Do While i < 5  
        i += 1  
    Loop  
End Sub
```


Otro ejemplo del uso de Do ... Loop

```
Sub repLopp()  
    Dim i As Integer 'Declaración de variable  
  
    'En este caso primero ejecuta y  
    'luego evalúa si se cumple la condición  
    Do  
        i += 1  
    Loop While i < 5  
End Sub
```

ACTIVIDAD

Desarrollar una aplicación de consola que permita calcular el fibonacci de un número.

Dada la fórmula:

$$f_n = f_{n-1} + f_{n-2}$$

Ejemplo, el Fibonacci de 8 es la siguiente secuencia:

0, 1, 1, 2, 3, 5, 8, 13

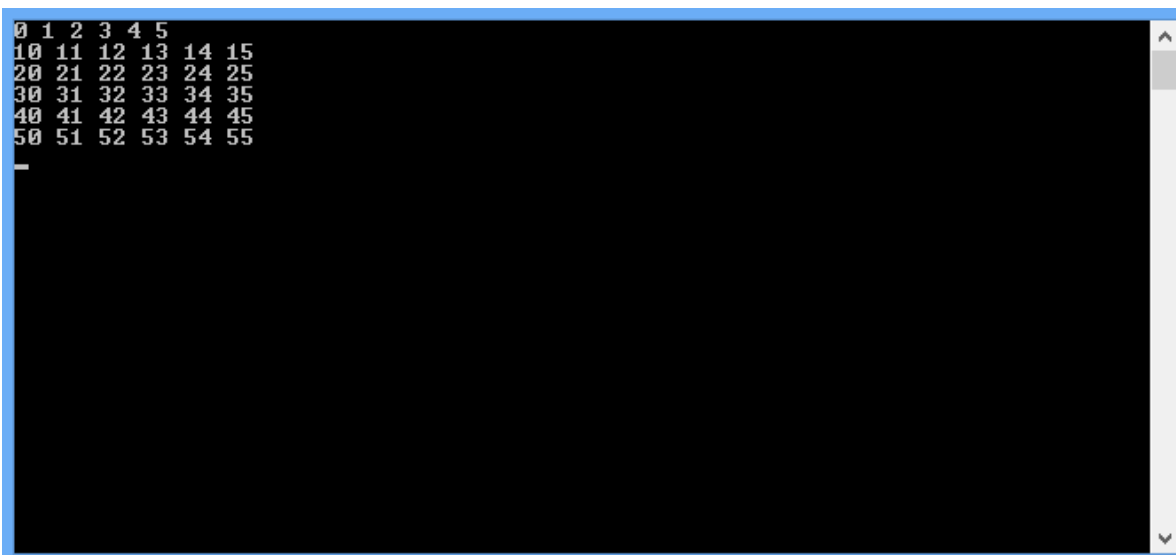
- Un arreglo puede definirse como un grupo o una colección homogénea y ordenada de elementos. Los arreglos pueden ser unidimensionales y multidimensionales.
- Los arreglos unidimensionales o también conocidos como vector, es un tipo de datos estructurado, que es una colección finita y ordenada de datos del mismo tipo. Importante saber que los arreglos el primer elemento está posicionado en el índice 0. Ejemplo de estos en vb.Net.

```
Sub vectores()
    ' Declaración de un vector de tipo String
    Dim vectorCadena As String() = {"valor 1", "valor 2", "valor 3"}
    Dim i As Integer
    ' Recorro el vector y obtengo cada uno de sus valores
    ' desde el índice 0
    For i = 0 To vectorCadena.Length - 1
        Console.WriteLine(vectorCadena(i).ToString)
    Next

    ' Declaración de un vector de tipo Fecha
    Dim vectorFecha As Date() = {CDate("12/12/2001"), CDate("02/11/2007")}
    ' Muestro el valor del vector en el índice 1
    Console.WriteLine(vectorFecha(1).ToString)

    ' Declaración de un vector de tipo entero
    Dim vectorEntero As Integer() = {2, 6, 4, 8, 23, 39}
    Dim res As Integer = 0
    ' Recorro el vector y sumo cada uno de sus valores
    For i = 0 To vectorEntero.Length - 1
        res += Int(vectorEntero(i).ToString)
    Next
    ' Muestro el valor de la suma de los valores en el vector que es 82
    Console.WriteLine(res)
End Sub
```

- Los arreglos multidimensionales, es un tipo de dato estructurado, que está compuesto por **n** dimensiones. Para hacer referencia a cada componente del arreglo es necesario utilizar **n** índices, uno para cada dimensión. Al igual que los arreglos unidimensionales, el primer elemento se encuentra ubicado en la dimensión [0,0]. Ejemplo de estos es:



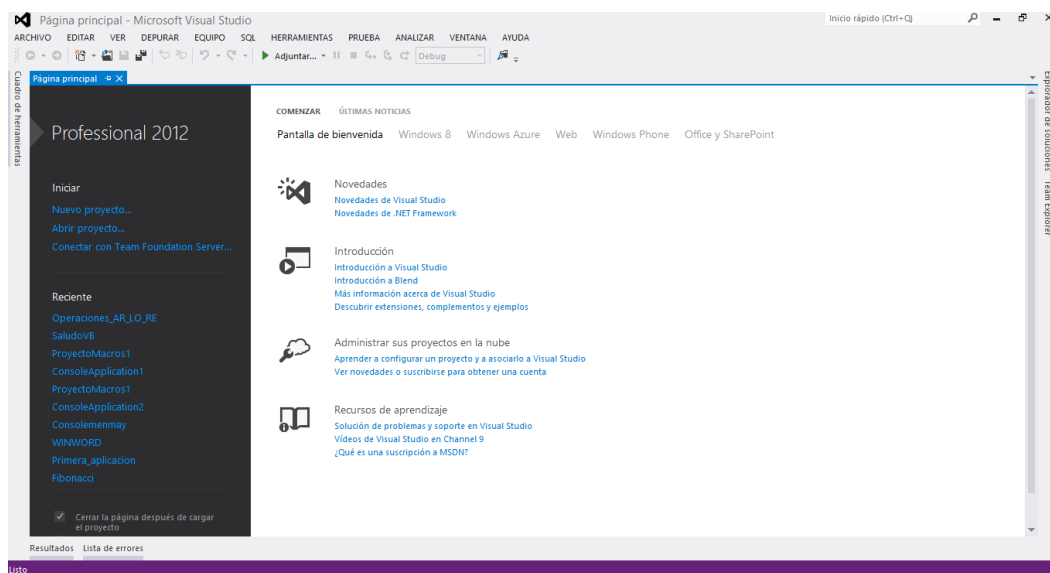
```
0 1 2 3 4 5
10 11 12 13 14 15
20 21 22 23 24 25
30 31 32 33 34 35
40 41 42 43 44 45
50 51 52 53 54 55
```

```
Sub matrix()  
Dim matriz(5, 5) As Integer 'Declaración de matriz de 5 filas y 5 columnas  
Dim i, j As Integer 'Declaración de variables  
  
'UBound obtiene el índice más alto de la dimensión indicada en la matriz  
Dim maxDim0 As Integer = UBound(matriz, 1)  
Dim maxDim1 As Integer = UBound(matriz, 2)  
  
'Recorre la Matriz  
For i = 0 To maxDim0  
    For j = 0 To maxDim1  
        matriz(i, j) = (i * 10) + j  
        Console.Write(matriz(i, j) & " ")  
    Next j  
    Console.WriteLine("")  
Next i  
End Sub
```

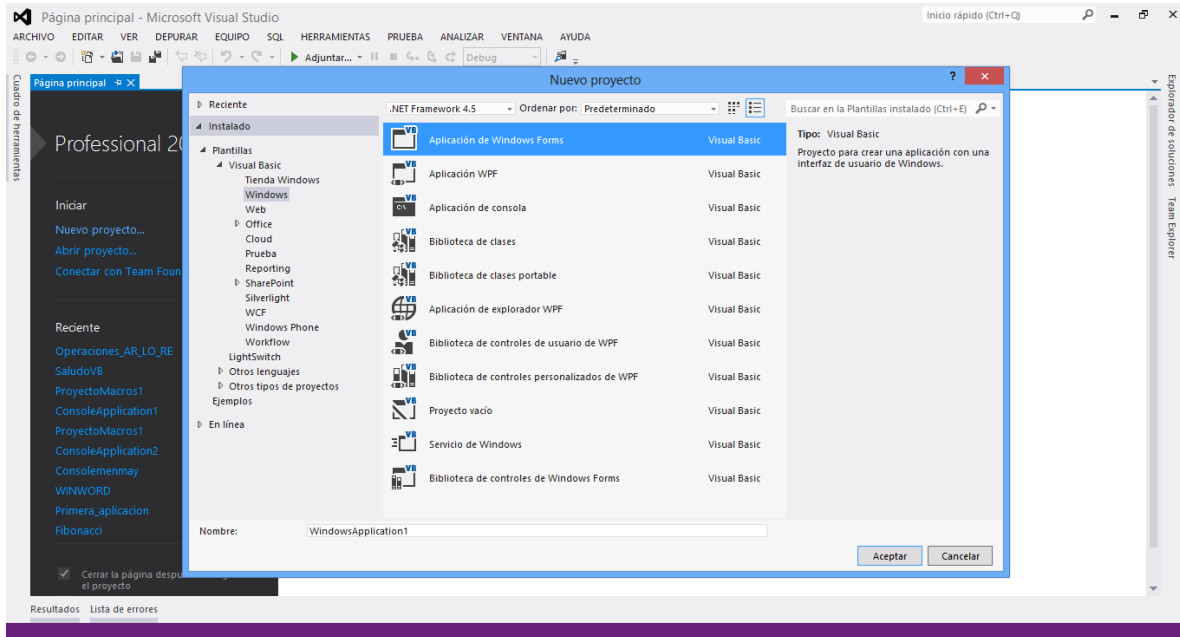
5. Aprendizajes esperados: Desarrollan proyectos .NET, aplicando las sentencias y sintaxis, según requerimientos.

El ambiente de trabajo en la herramienta Visual Studio es similar en todas las versiones, destacando que se pueden crear proyectos nuevos o abrir proyectos o archivos creados previamente. Esto se puede realizar mediante el menú Archivo, o bien con los botones de la barra.

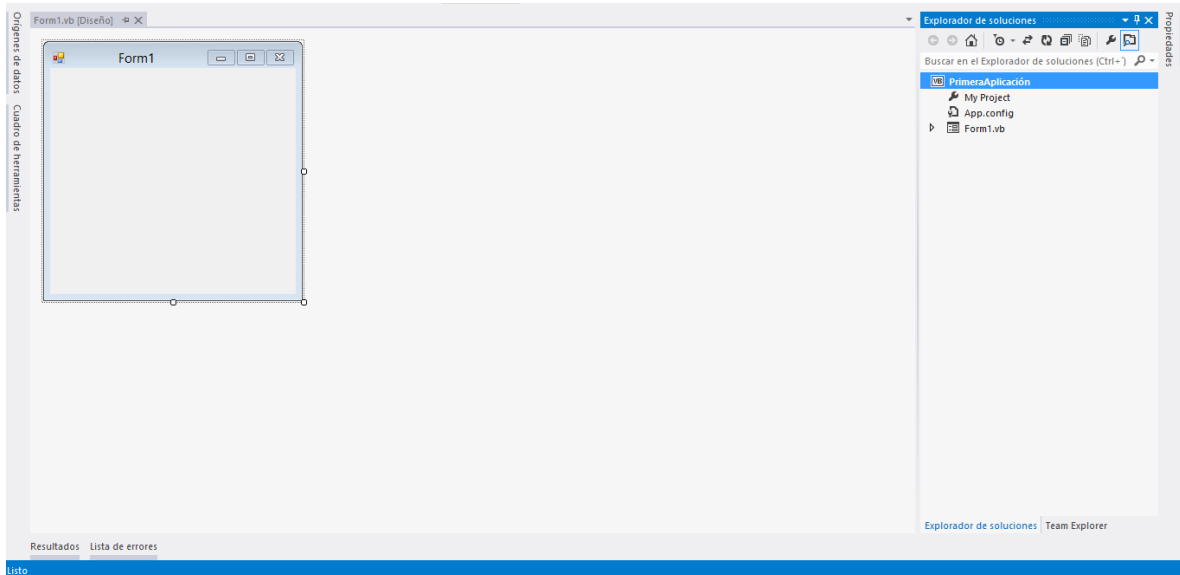
Estos son los ambientes en Visual Studio 2010 y en visual Studio 2012



- Las aplicaciones Windows Form son aplicaciones cliente que se ejecutan en el equipo del usuario, muestran información, solicitan entrada de los usuarios y se comunican con equipos remotos en una red. Para crear una aplicación Windows Form es necesario crear un nuevo proyecto, seleccionar el lenguaje de programación base, que en este caso sería vb.Net, y luego seleccionar Aplicación Windows Form, en seguida dar un nombre a la aplicación, y aceptar. Tal como lo muestra la siguiente imagen en visual studio 2012.



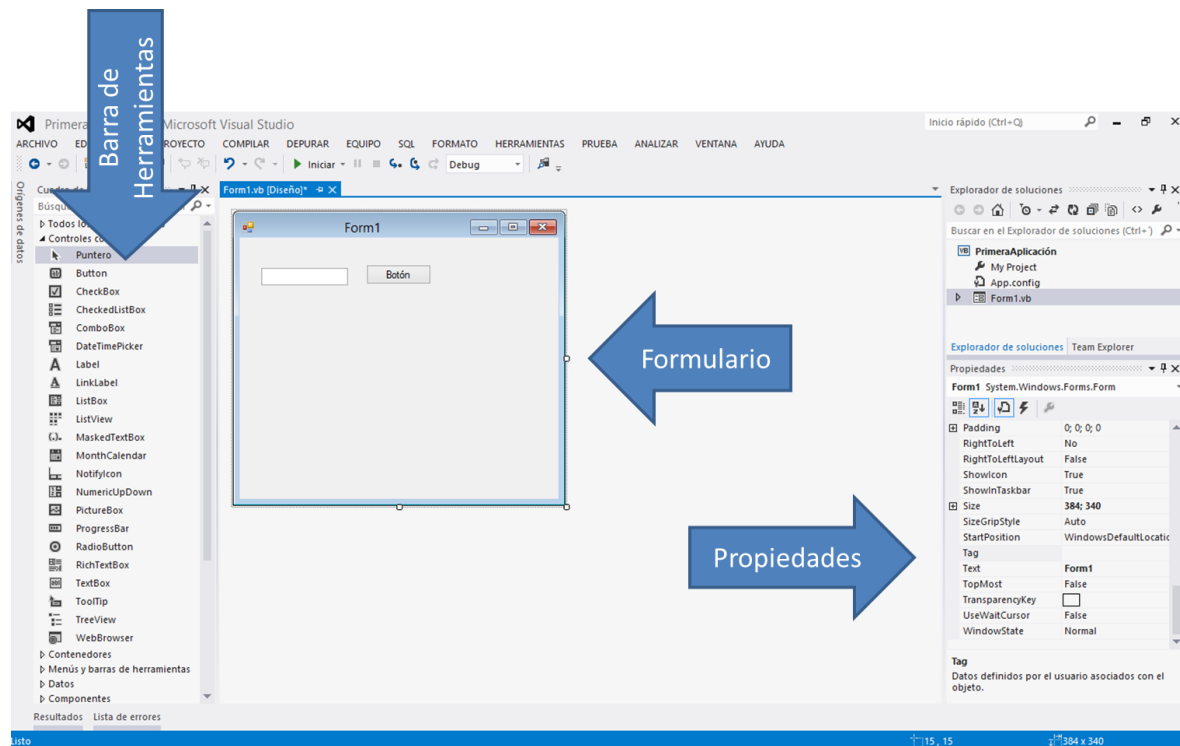
Al crear la aplicación, se genera automáticamente la solución, que puede verse en el Explorador de Soluciones, además de crear por defecto el primer formulario de nombre Form1.



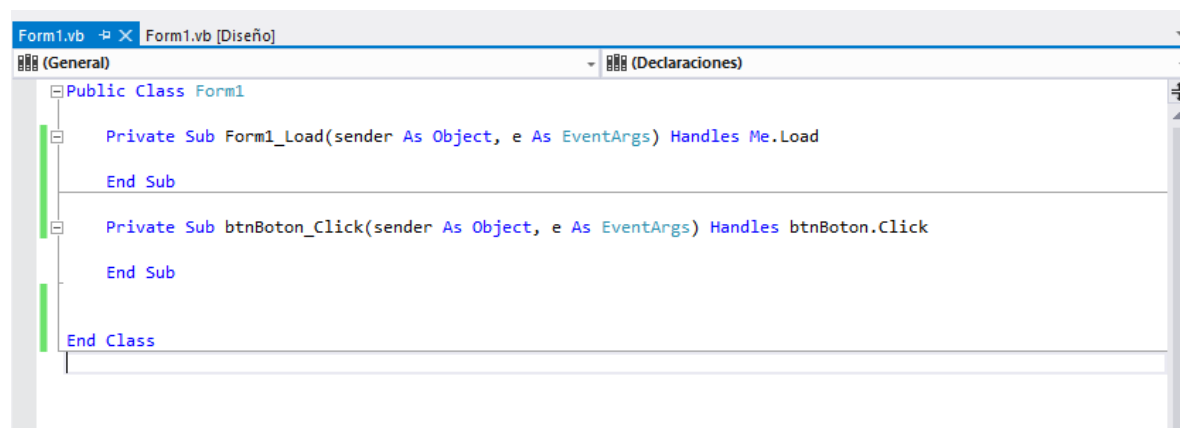
Al crear un formulario se creará la clase del formulario, que se llamará Form1.vb, teniendo la extensión vb por el lenguaje de programación.

En el entorno de trabajo con formularios Windows se tiene a disposición una serie de herramientas que permiten trabajar con los formularios, y generar los eventos (mensajes en POO). Además de contar con el editor de propiedades de un control.

En la siguiente imagen, se puede ver el editor de propiedades, un formulario con 2 controles, una caja de texto (TextBox), y un Botón (Button).

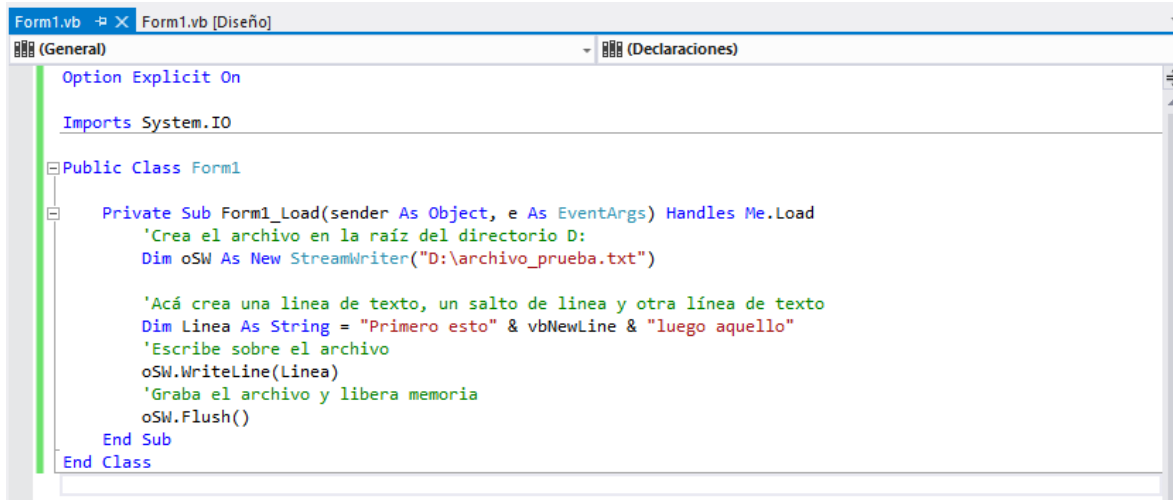


En la siguiente imagen se puede ver el código de la clase con dos eventos, el primero Load que se ejecuta cuando se carga el formulario, y el siguiente es el evento click del botón, el cual se ejecuta cuando el usuario presiona el botón.



- El trabajar con archivos de texto en vb.Net es relativamente fácil, se les conoce como ficheros, o bien como archivos de texto plano (txt). Con algunas sentencias propias de .Net se puede leer su contenido o bien escribir sobre él.

Un ejemplo de cómo crear y escribir sobre él.

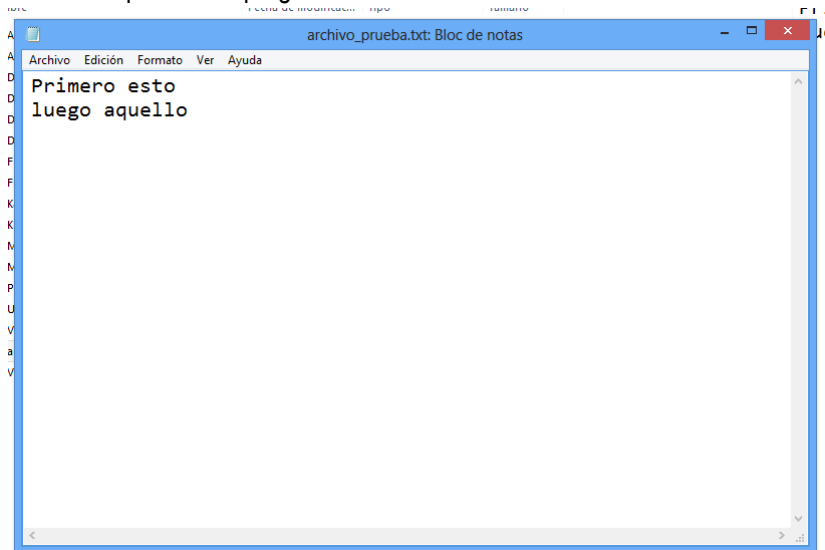


```
Form1.vb [Diseño]
(General)
Option Explicit On
Imports System.IO

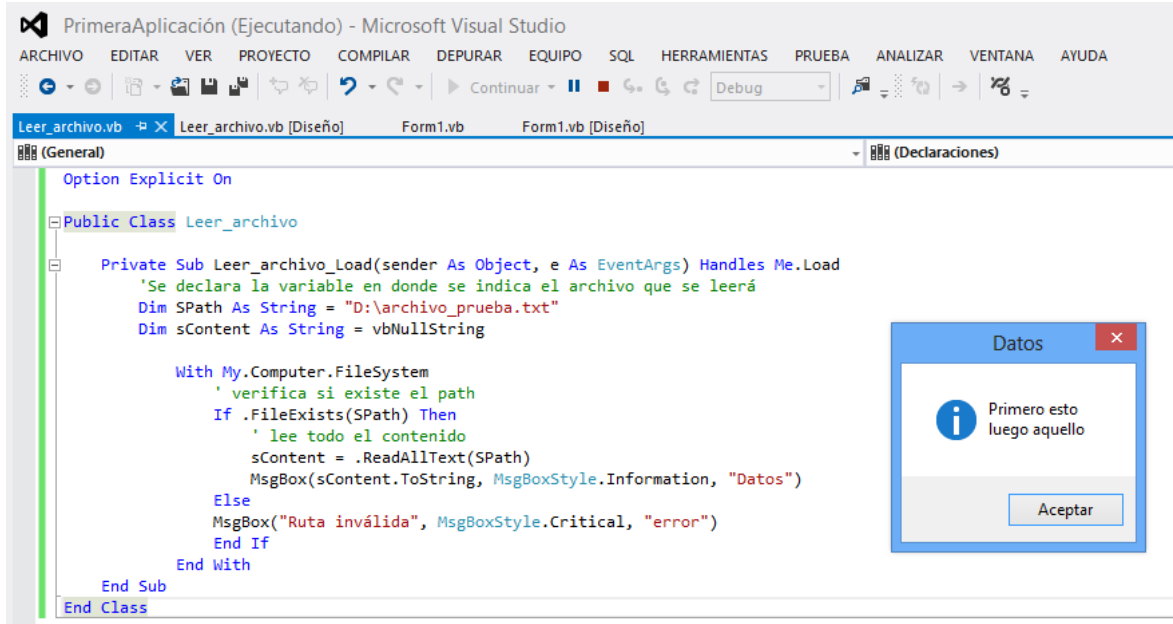
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles Me.Load
        'Crea el archivo en la raíz del directorio D:
        Dim oSW As New StreamWriter("D:\archivo_prueba.txt")

        'Acá crea una línea de texto, un salto de línea y otra línea de texto
        Dim Linea As String = "Primero esto" & vbCrLf & "luego aquello"
        'Escribe sobre el archivo
        oSW.WriteLine(Linea)
        'Graba el archivo y libera memoria
        oSW.Flush()
    End Sub
End Class
```

Este es el archivo de texto que crea el programa

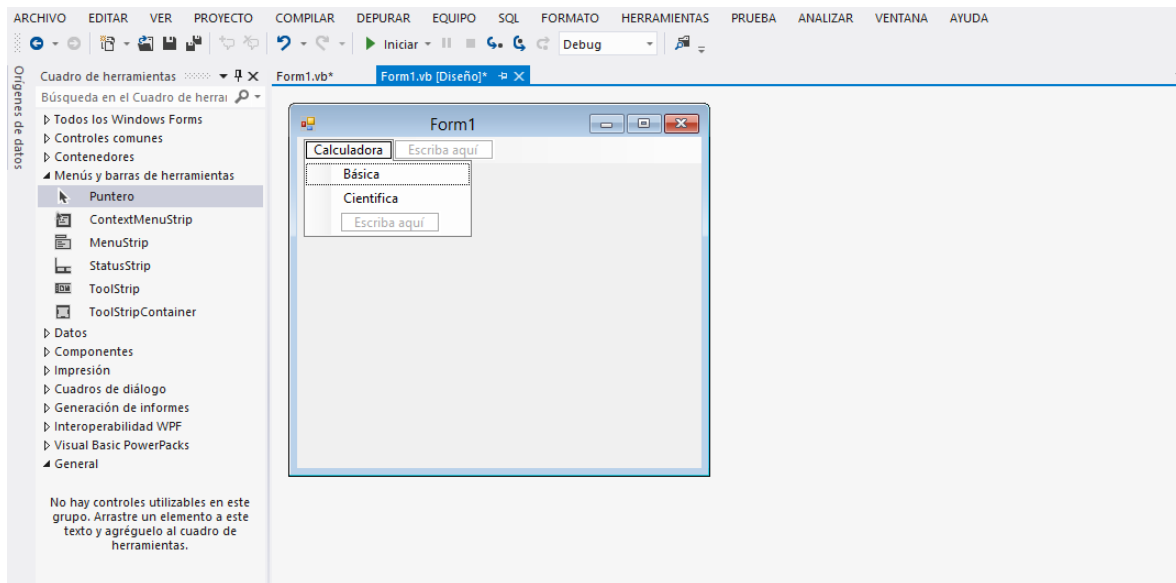


En el siguiente ejemplo se lee el contenido de un archivo, luego lo muestra por un cuadro de mensajes.

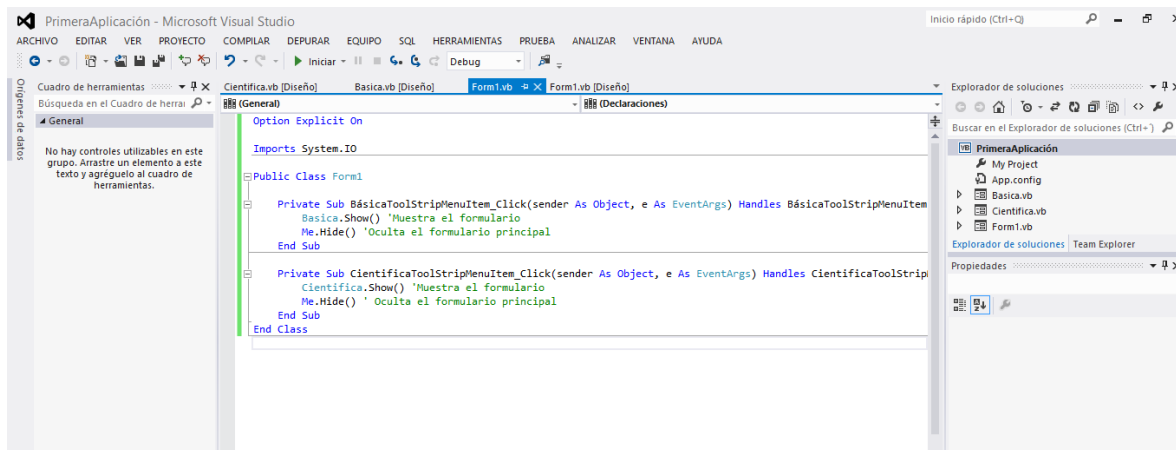


6. Aprendizajes esperados: Construyen programas de aplicación de baja complejidad para resolver problemas específicos, aplicando lenguaje .net.

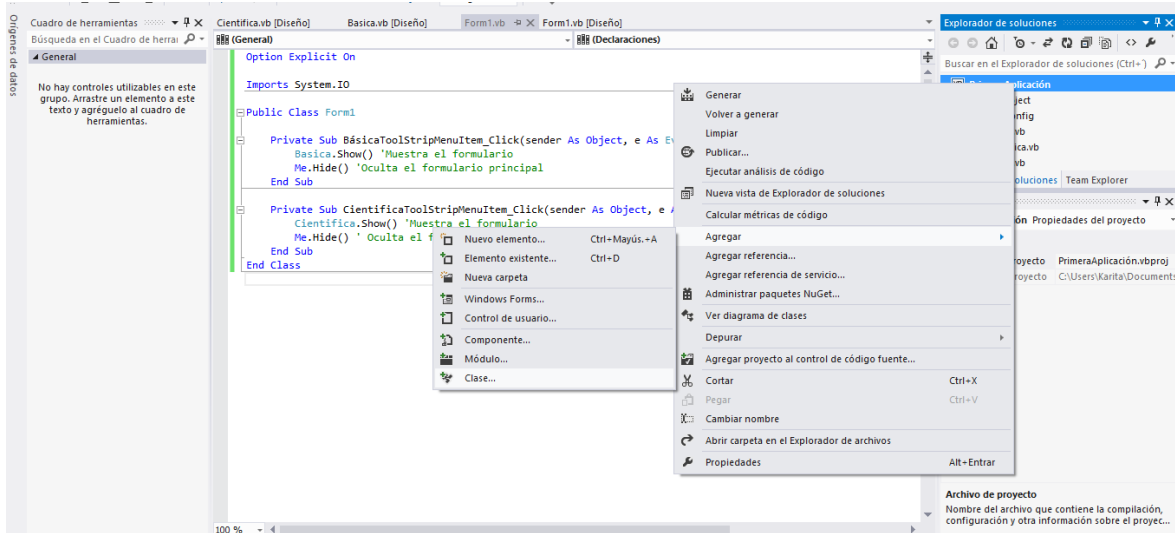
- En un proyecto se pueden incluir varios formularios, clases, y otros tipos de objetos propios de .Net. Acá se verá cómo crear un proyecto que incluya varios formularios, sus clases asociadas, navegabilidad entre ellos con algunas propiedades, además de la creación de clases para demostrar la reutilización de código.
- Lo primero se debe crear un proyecto en el cual se tendrá un formulario principal, en el que se incluirá un control de tipo Menú, llamado MenuStrip, el cual permite crear opciones de navegación y en distintos niveles, tal como lo muestra la siguiente imagen, en el cuál crearemos una calculadora científica y una básica.



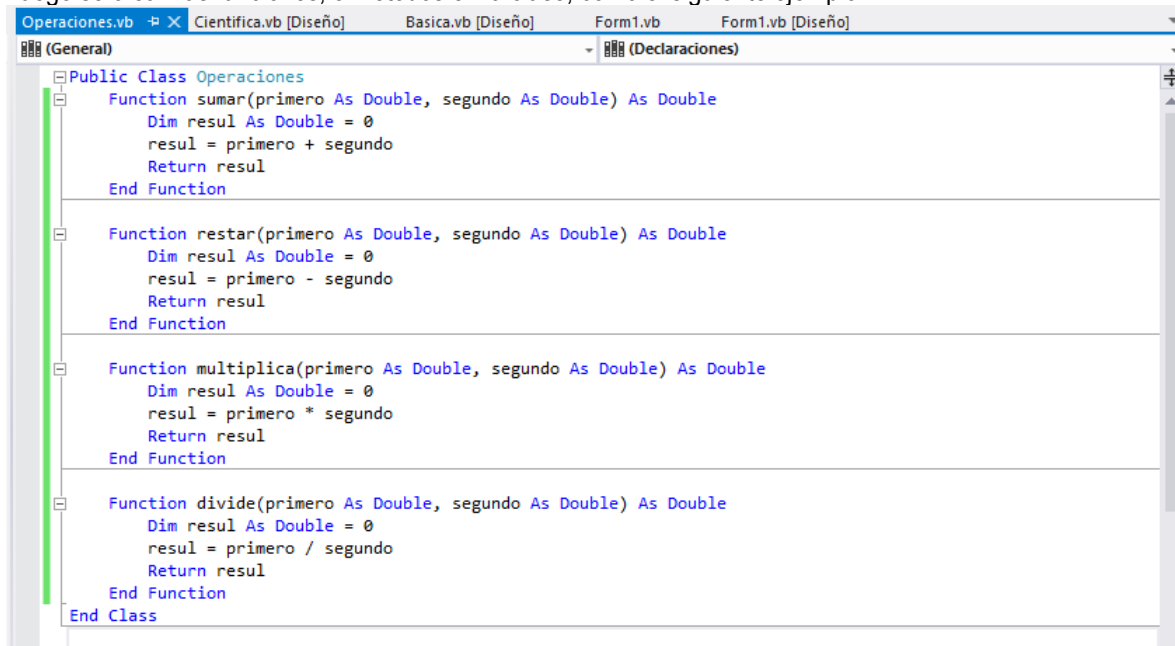
Se debe crear 2 formularios, uno para Básica, y otro para Científica, para llamar ambos formularios a través de código se debe crear los eventos del menú, como muestra la imagen.



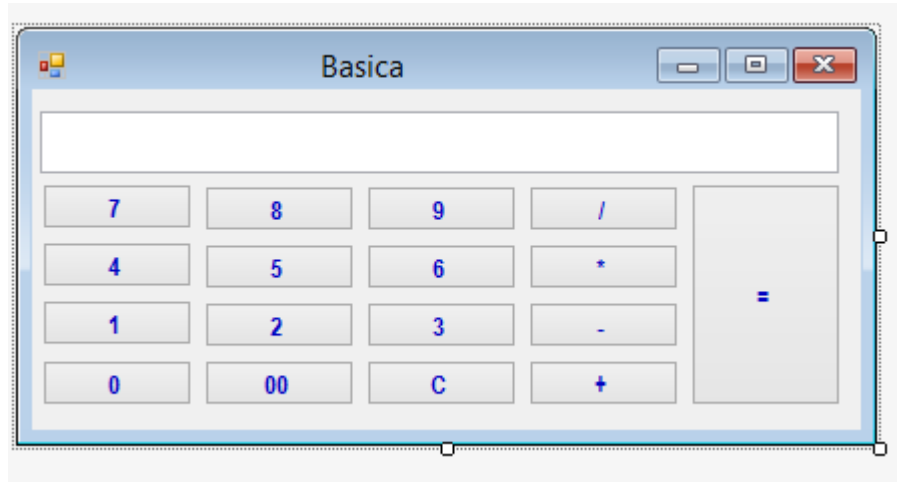
Luego se creará una clase en el proyecto que permitirá la reutilización de código, llamada de distintos formularios. Para crear la clase se debe seleccionar con el botón derecho sobre el proyecto en el explorador de soluciones, agregar una nueva Clase, como lo muestra la siguiente imagen.



Luego se crean las funciones, o métodos en la clase, como el siguiente ejemplo:



Un ejemplo de cómo instanciar la clase desde un formulario. Creamos la interfaz del formulario, incorporando los controles necesarios para emular una calculadora. Como lo muestra la imagen.



Luego en el evento Click del botón sumar (+), se instancia la Clase, y se utiliza su función Sumar, pasando los dos valores a sumar, como lo muestra el siguiente ejemplo.

```

Basica.vb | Operaciones.vb | Cientifica.vb [Diseño] | Basica.vb [Diseño] | Form1.vb | Form1.vb [Diseño]
(General)
Private Sub btnIgual_Click(sender As System.Object, e As System.EventArgs) Handles btnIgual.Click
    Dim segundo As Double = Double.Parse(txtResultado.Text)
    Dim resultado As Double = 0

    Select Case oper
    Case 1
        'Se instancia la clase Operaciones
        Dim objOper As New Operaciones
        'Se instancia la función sumar pasando los parámetros
        'y retorna el resultado, asignandoselo al control textbox
        ' en su propiedad Text
        txtResultado.Text = objOper.sumar(primer, segundo)
    Exit Select
    Case 2
        'Se instancia la clase Operaciones
        Dim objOper As New Operaciones
        'Se instancia la función restar pasando los parámetros
        'y retorna el resultado, asignandoselo al control textbox
        ' en su propiedad Text
        txtResultado.Text = objOper.restar(primer, segundo)
    Exit Select
    Case 3
        'Se instancia la clase Operaciones
        Dim objOper As New Operaciones
        'Se instancia la función multiplicar pasando los parámetros
        'y retorna el resultado, asignandoselo al control textbox
        ' en su propiedad Text
        txtResultado.Text = objOper.multiplica(primer, segundo)
    Exit Select
    Case 4
        'Se instancia la clase Operaciones
        Dim objOper As New Operaciones
        'Se instancia la función dividir pasando los parámetros
        'y retorna el resultado, asignandoselo al control textbox
        ' en su propiedad Text
        txtResultado.Text = objOper.dividir(primer, segundo)
    Exit Select
    End Select
End Sub

```

SEGUNDA UNIDAD: Programación Avanzada con Visual Basic.NET

7. APRENDIZAJE ESPERADO: Construyen proyectos que permitan mantención a archivos de textos desde VB.NET, de acuerdo a requerimientos.

- Anteriormente se vio como leer y escribir en un archivo de texto, ahora se verá como trabajar con estos archivos en edición, eliminación, mover de carpetas, obtener sus propiedades, contar la cantidad de archivos en directorio específico.

Para crear un archivo vacío se debe utilizar la función `System.IO.File.Create`, como lo muestra el siguiente ejemplo:

```
Public Class Crear_Nuevo

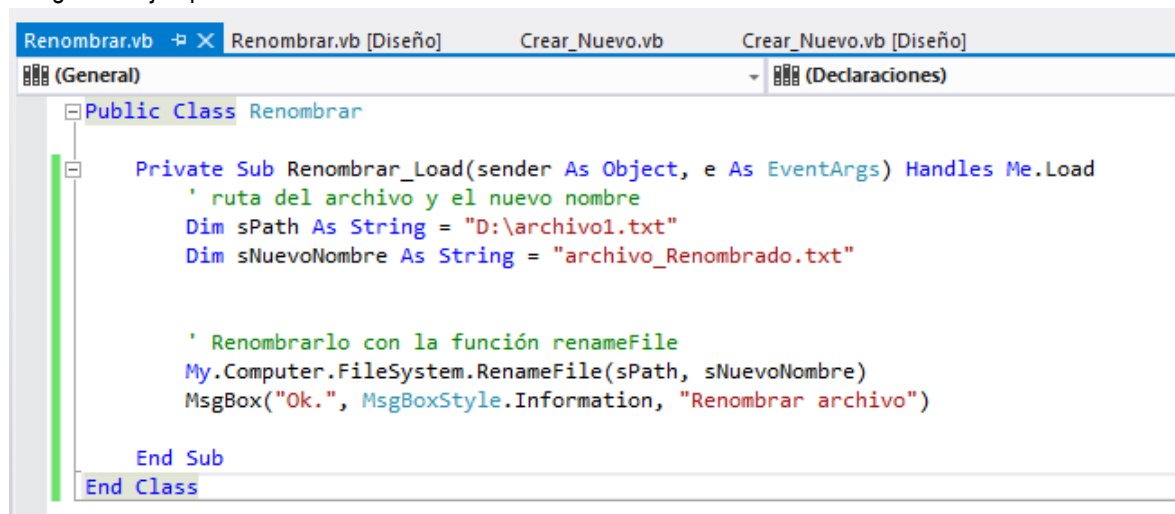
    Private Sub Crear_Nuevo_Load(sender As Object, e As EventArgs) Handles Me.Load

        Dim Archivo As System.IO.FileStream
        ' crea un archivo vacío archivo_prueba.txt
        Archivo = System.IO.File.Create("D:\archivo_prueba.txt")

    End Sub

End Class
```

Para renombrar un archivo se debe utilizar la función `My.Computer.FileSystem.RenameFile`, como lo muestra el siguiente ejemplo:



```
Renombrar.vb [Diseño]  Crear_Nuevo.vb  Crear_Nuevo.vb [Diseño]
(General)  (Declaraciones)

Public Class Renombrar

    Private Sub Renombrar_Load(sender As Object, e As EventArgs) Handles Me.Load
        ' ruta del archivo y el nuevo nombre
        Dim sPath As String = "D:\archivo1.txt"
        Dim sNuevoNombre As String = "archivo_Renombrado.txt"

        ' Renombrarlo con la función renameFile
        My.Computer.FileSystem.RenameFile(sPath, sNuevoNombre)
        MsgBox("Ok.", MsgBoxStyle.Information, "Renombrar archivo")

    End Sub

End Class
```

Para eliminar archivos y carpetas se debe utilizar, en archivos la función `My.Computer.FileSystem.DeleteFile`, y en carpetas la función `My.Computer.FileSystem.DeleteDirectory`, como lo muestra el siguiente ejemplo:

```

Eliminar.vb  ➤ ✕
⚡ (Eliminar eventos)  ⚡ Load

Public Class Eliminar

    Private Sub Eliminar_Load(sender As Object, e As EventArgs) Handles Me.Load

        ' Eliminar el archivo, mostrando el cuadro
        ' de diálogo de eliminar de windows para confirmar
        Dim sdir As String = "D:\Nueva carpeta"
        Dim Spath As String = "D:\archivo1.txt"

        ' Archivo
        My.Computer.FileSystem.DeleteFile( _
            Spath, _
            FileIO.UIOption.AllDialogs, _
            FileIO.RecycleOption.SendToRecycleBin, _
            FileIO.UICancelOption.DoNothing)

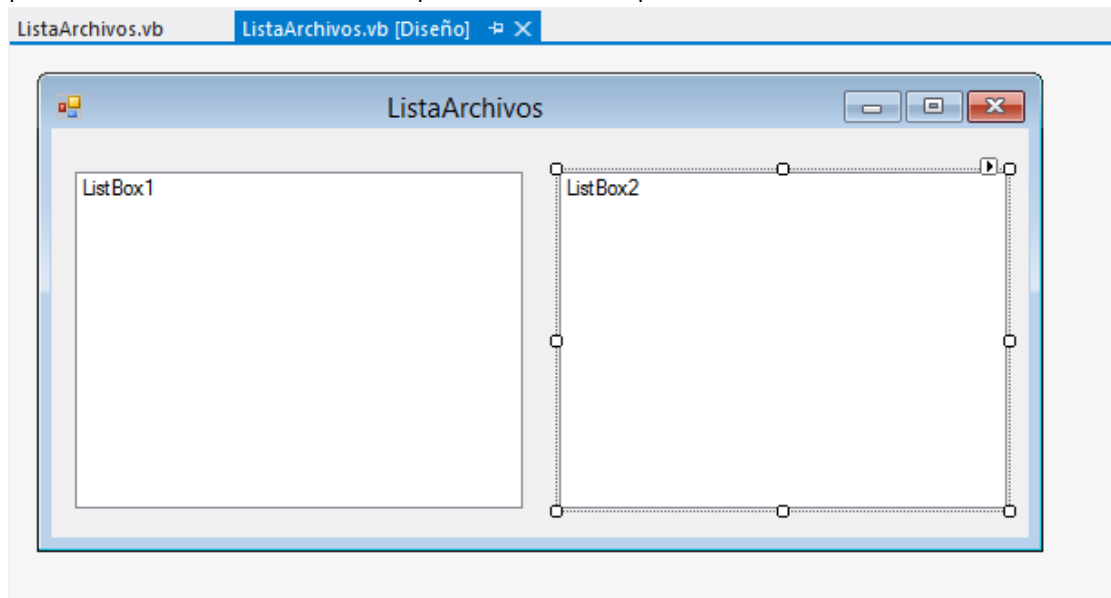
        ' carpeta
        My.Computer.FileSystem.DeleteDirectory( _
            sdir, _
            FileIO.UIOption.AllDialogs, _
            FileIO.RecycleOption.SendToRecycleBin, _
            FileIO.UICancelOption.DoNothing)

    End Sub
End Class

```

Para cargar directorios y archivos en un control de tipo ListBox, se debe utilizar las funciones Directory.GetFiles, además de agregar cada uno de ellos en el ListBox, como muestra el siguiente ejemplo:

Lo primero es crear un formulario, e incorporar un control de tipo ListBox, como a continuación.



Luego se debe crear el código, como se muestra a continuación.

```

ListaArchivos.vb [Diseño]
(ListaArchivos eventos) Load

Option Explicit On
Option Strict On

'Importar la librería del framework que permite trabajar con entrada y salida de datos.
Imports System.IO

Public Class ListaArchivos

    Private Sub ListaArchivos_Load(sender As Object, e As EventArgs) Handles Me.Load
        ' Obtener todos los archivos .pptx del directorio ( incluyendo subdirectorios )
        For Each archivos As String In Directory.GetFiles("D:\AIEP\FLP", _
            "*.pptx", _
            SearchOption.AllDirectories)

            ' extraer el nombre de la ruta
            archivos = archivos.Substring(archivos.LastIndexOf("\") + 1).ToString
            ' Agregar el valor al listbox
            ListBox1.Items.Add(archivos.ToString)
        Next

        ' Obtener todos los directorios del directorio C: ( un solo nivel )
        For Each archivos As String In Directory.GetDirectories("C:\", "*.*", SearchOption.TopDirectoryOnly)
            ' extraer el nombre de la carpeta de la ruta completa
            archivos = archivos.Substring(archivos.LastIndexOf("\") + 1).ToString
            ' Agregar el valor
            ListBox2.Items.Add(archivos.ToString)
        Next
    End Sub
End Class

```

Para contar archivos en un directorio, se debe utilizar la función System.Collections.ObjectModel.ReadOnlyCollection, como se muestra a continuación.

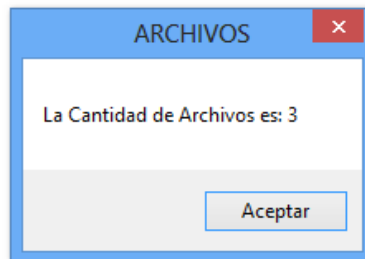
```

Contar.vb [Diseño]
(Contar eventos) Load

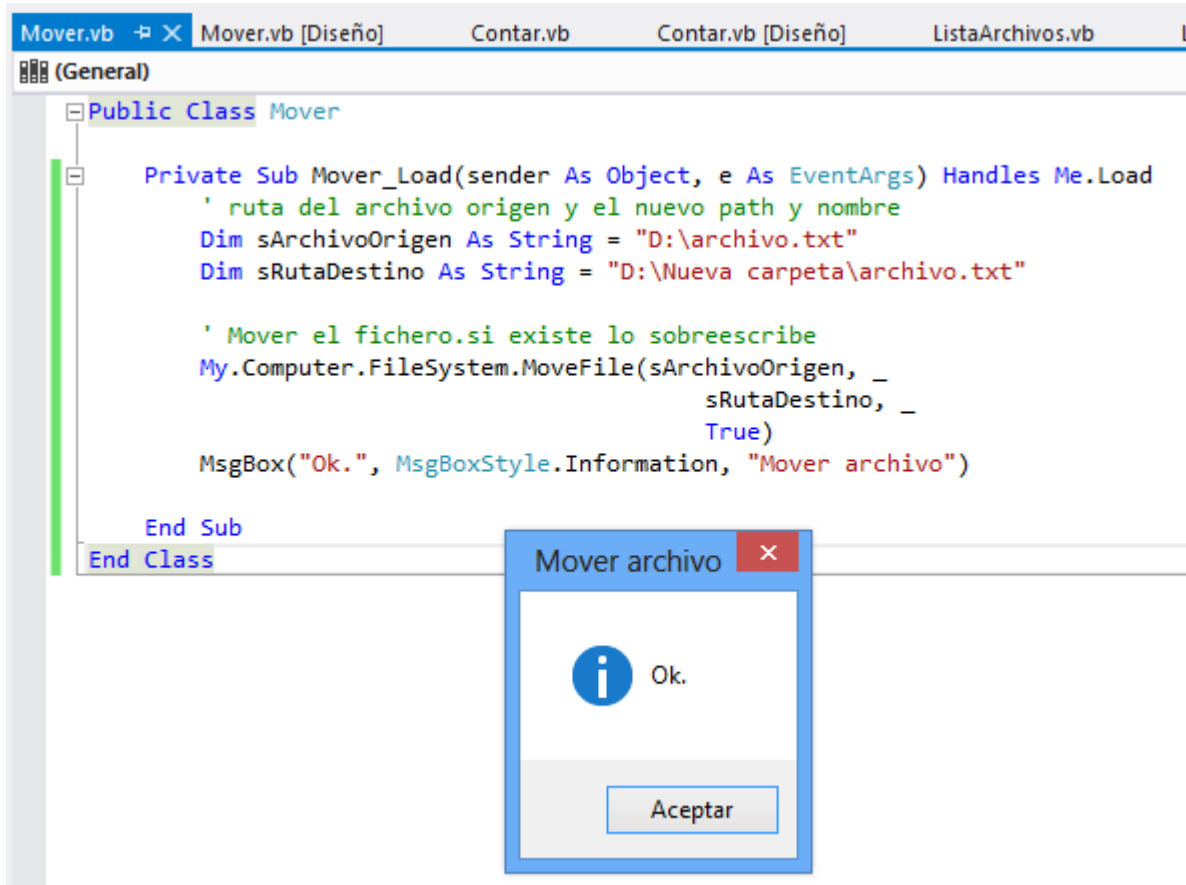
Public Class Contar

    Private Sub Contar_Load(sender As Object, e As EventArgs) Handles Me.Load
        Dim ContadorDeArchivos As System.Collections.ObjectModel.ReadOnlyCollection(Of String)
        'le indicamos el path que queremos
        ContadorDeArchivos = My.Computer.FileSystem.GetFiles("D:\AIEP\FLP")
        'nos devuelve la cantidad de archivos
        MsgBox("La Cantidad de Archivos es: " & CStr(ContadorDeArchivos.Count))
    End Sub
End Class

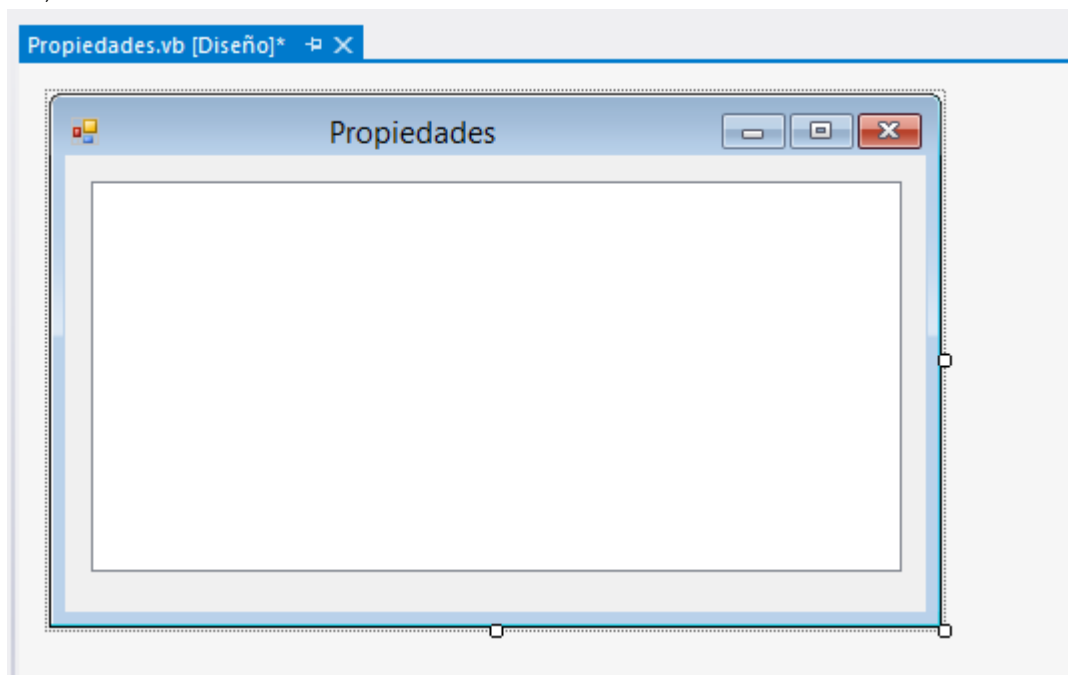
```



Para mover archivos de un directorio a otro se debe utilizar la función My.Computer.FileSystem.MoveFile, como lo muestra el siguiente ejemplo.



Para obtener las propiedades de los archivos se debe crear un formulario, e incorporar un control de tipo ListView, como se muestra a continuación.



Para incorporar los archivos en el listView, e indicar sus propiedades, se utilizan las siguientes funciones, como muestra el ejemplo.

The screenshot shows a Visual Basic IDE with a class named 'Propiedades' and a preview window titled 'Propiedades'.

Propiedades (Diseño)

```

Public Class Propiedades
    Private Sub Propiedades_Load(sender As Object, e As EventArgs) Handles Me.Load
        ' agregar columnas
        With ListView1
            .Columns.Add("Nombre", 150)
            .Columns.Add("Fecha y hora de modificación", 150)
            .Columns.Add("Tamaño - bytes", 100)
            .Columns.Add("Extensión", 80)
            .View = View.Details
            .GridLines = True
        End With
        ListView1.Items.Clear()
        ' recorrer los ficheros en la colección
        For Each sFichero As String In Directory.GetFiles("D:\AIEP\FLP", "*.pptx", SearchOption.TopDirectoryOnly)
            ' Crear nuevo objeto FileInfo
            Dim Archivo As New FileInfo(sFichero)
            ' Crear nuevo objeto ListViewItem
            Dim item As New ListViewItem(Archivo.Name.ToString)
            ' cargar los datos y las propiedades
            With item
                ' LastWriteTime - fecha de modificación
                .SubItems.Add(Archivo.LastWriteTime.ToShortDateString & " " & Archivo.LastWriteTime.ToShortTimeString)
                ' Length - tamaño en bytes
                .SubItems.Add(Archivo.Length.ToString)
                ' Extension - extensión
                .SubItems.Add(Archivo.Extension.ToString)
                ListView1.Items.Add(item) ' añadir el item
            End With
        Next
    End Sub
End Class

```

Propiedades

Nombre	Fecha y hora de modificaci...	Tamaño - bytes	Extensión
FLP_II.pptx	20/07/2013 13:15	1850905	.pptx
FP_II.pptx	17/07/2013 0:06	2096335	.pptx
FP_III.pptx	27/08/2012 15:46	351013	.pptx

8. APRENDIZAJE ESPERADO: Diseñan y programan proyectos de mediana complejidad para aplicaciones que dan mantención a una base de datos.

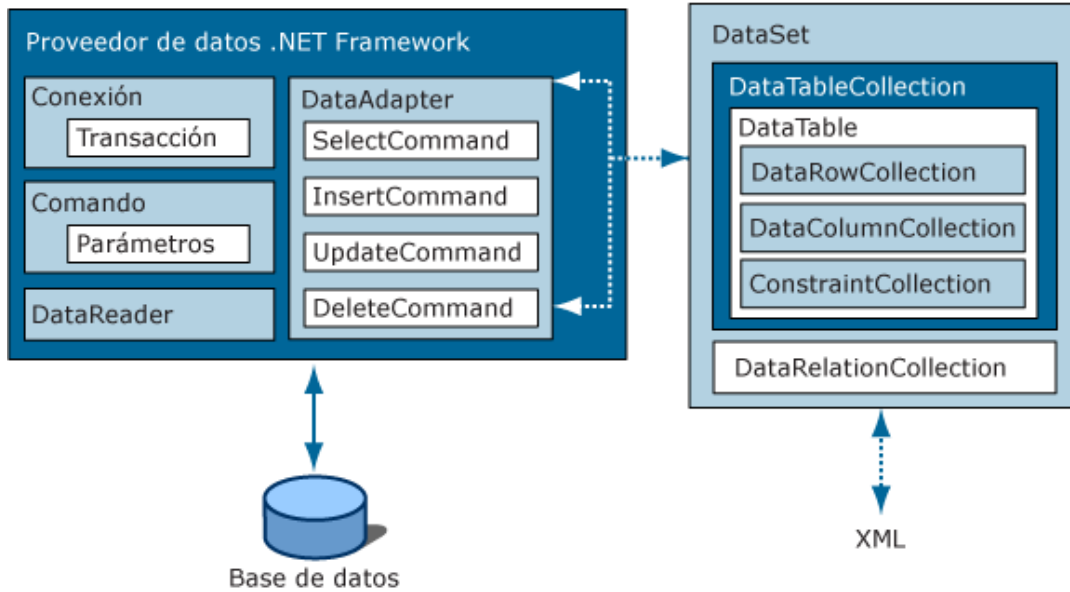
- ADO.Net es un conjunto de clases que exponen servicios de acceso a datos, y que forma parte integral de .NET Framework, el cual permite desarrollar aplicaciones con acceso a datos de distintos orígenes: bases de datos relacionales, XML, archivos, etc.
- ADO .NET incluye proveedores de datos administrados de .NET Framework: para conectarse a una base de datos, ejecutar comandos y recuperar resultados, y objetos DataSet de ADO.NET que tiene una estructura análoga a una Base de Datos en memoria.
- **System.Data** es el espacio de nombres de ADO .NET. Entre sus tareas habituales en la programación de aplicaciones que gestionan bases de datos está establecer una conexión con la Base de Datos, solicitar al Servidor datos específicos, el Servidor retorna los datos solicitados, el Usuario modifica los datos, y envía las actualizaciones al Servidor, se cierra la conexión, la gestión de datos se puede realizar en línea, o en un proceso desconectado.

-

ADO.Net permite trabajar con entornos conectados y desconectados

- Un entorno conectado es aquel en que los usuarios están conectados continuamente a una fuente de datos (base de datos); por ejemplo, en una red local.
Ventajas: El entorno es más fácil de mantener, la concurrencia se controla más fácilmente, datos actualizados en línea.
Desventajas: Debe existir una conexión de red constante, escalabilidad limitada.
- Un entorno desconectado es aquel en el que los datos pueden modificarse de forma independiente y los cambios se escriben posteriormente en la base de datos
Ventajas: Las conexiones se utilizan durante el menor tiempo posible, permitiendo que menos conexiones den servicio a más usuarios, un entorno desconectado mejora la escalabilidad y el rendimiento de las aplicaciones.
Inconvenientes: Los datos no siempre están actualizados en línea, pueden producirse conflictos de cambios de datos entre distintos usuarios.
- ADO .NET brinda servicios que permiten trabajar en ambos esquemas. Elegir el esquema a utilizar depende de la concurrencia de usuarios, la necesidad de contar con datos actualizados en línea y la disponibilidad de acceso al Servidor (red).

Los componentes ADO.Net son proveedores de datos de .NET Framework, y DataSet, el siguiente esquema ejemplifica.



ADO.NET se basa en los siguientes espacios de nombres de accesos a datos:

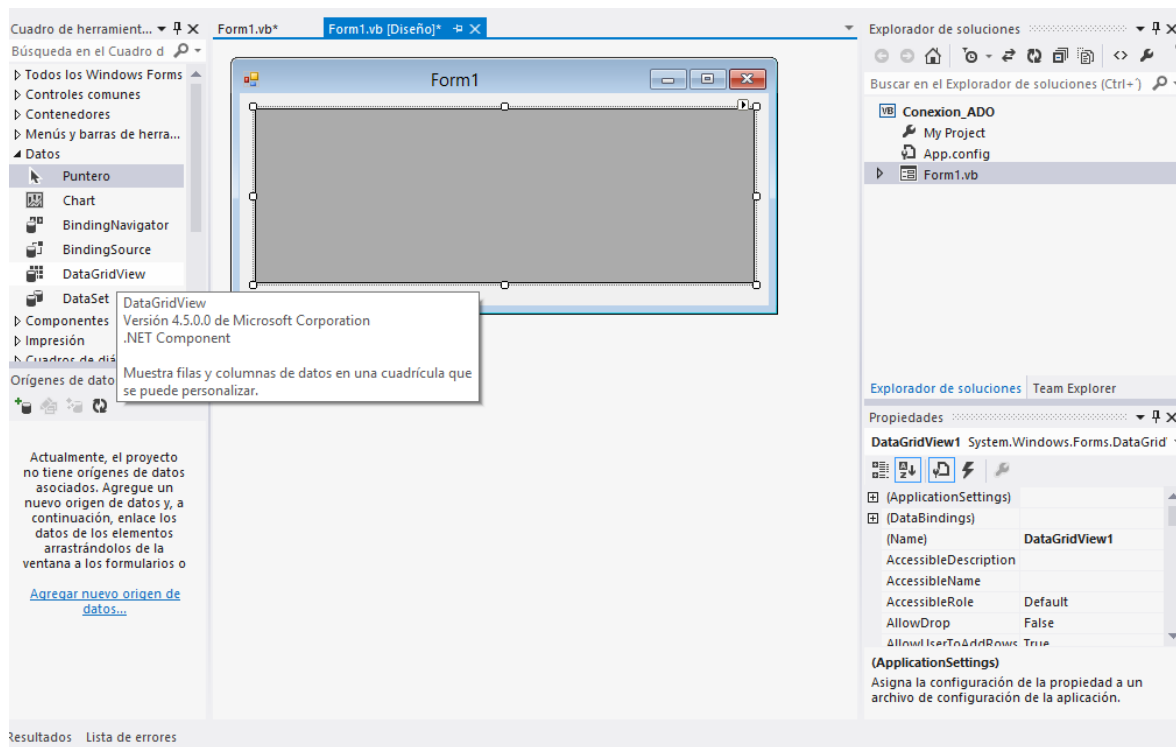
- ⇒ System.Data que proporciona las clases de acceso a datos generales.
- ⇒ System.Data.Common que contiene las clases compartidas por los proveedores de datos.
- ⇒ System.Data.OleDb que almacena las clases del proveedor de datos OLE DB.
- ⇒ System.Data.SqlClient que expone las clases del proveedor de datos para SQL Server.

Los objetos de los proveedores de datos de ADO.NET

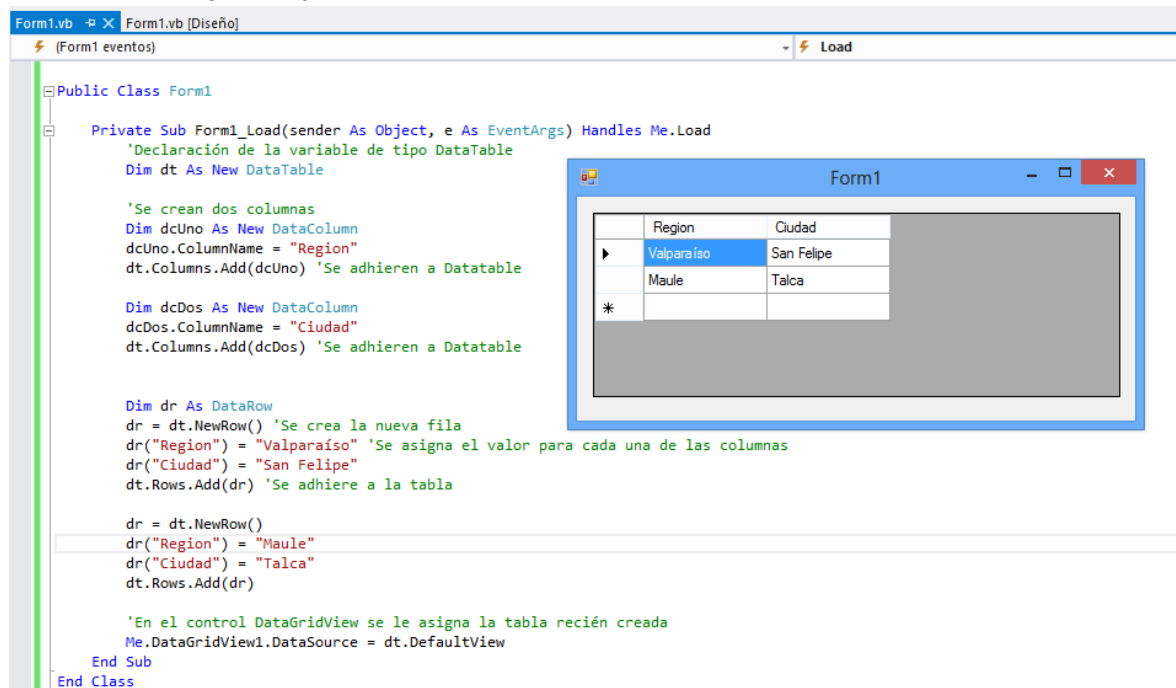
OBJETO	DESCRIPCIÓN
Connection	Establece una conexión a un origen de datos determinado. La clase base para todos los objetos Connection es DbConnection.
Command	Ejecuta un comando en un origen de datos. Expone Parameters y puede ejecutarse en el ámbito de un objeto Transaction de Connection . La clase base para todos los objetos Command es DbCommand.
DataReader	Lee una secuencia de datos de sólo avance y sólo lectura desde un origen de datos. La clase base para todos los objetos DataReader es DbDataReader.
DataAdapter	Llena un DataSet y realiza las actualizaciones necesarias en el origen de datos. La clase base para todos los objetos DataAdapter es DbDataAdapter.

- Para trabajar con la estructura DataTable, que permite crear una estructura de una tabla, incorporando Columnas y Filas a través del código del programa.

Un ejemplo de ello, se crea un formulario y se le incorpora un objeto de tipo DataGridView, que se encuentra en la lista Datos del cuadro de herramientas, como muestra la siguiente imagen.



En el código se utilizan los objetos DataTable para definir una tabla, DataColumn para definir las columnas que serán parte de esta tabla, y luego DataRow para crear cada uno de los registros de esta tabla. Esto se demuestra en el siguiente ejemplo.



- Para crear un DataSet, se debe utilizar el objeto DataSet, al cual se le pueden asignar los valores de un Datatable, o bien asignarles los valores desde una conexión a una Base de Datos. Ejemplo de uso de un DataSet, asignándoles el valor de una DataTable, el mismo del ejemplo anterior.

The screenshot displays the Visual Basic IDE with the 'Form1.vb [Diseño]' window active. The design view on the right shows a form titled 'Form1' containing a table with two columns: 'Region' and 'Ciudad'. The table has two data rows: 'Valparaíso' and 'San Felipe', and 'Maule' and 'Talca'. Below the table is a text area with an asterisk. The code view on the left shows the following code:

```
'Se crean dos columnas
Dim dcUno As New DataColumn
dcUno.ColumnName = "Region"
dt.Columns.Add(dcUno) 'Se adhieren a Datatable

Dim dcDos As New DataColumn
dcDos.ColumnName = "Ciudad"
dt.Columns.Add(dcDos) 'Se adhieren a Datatable

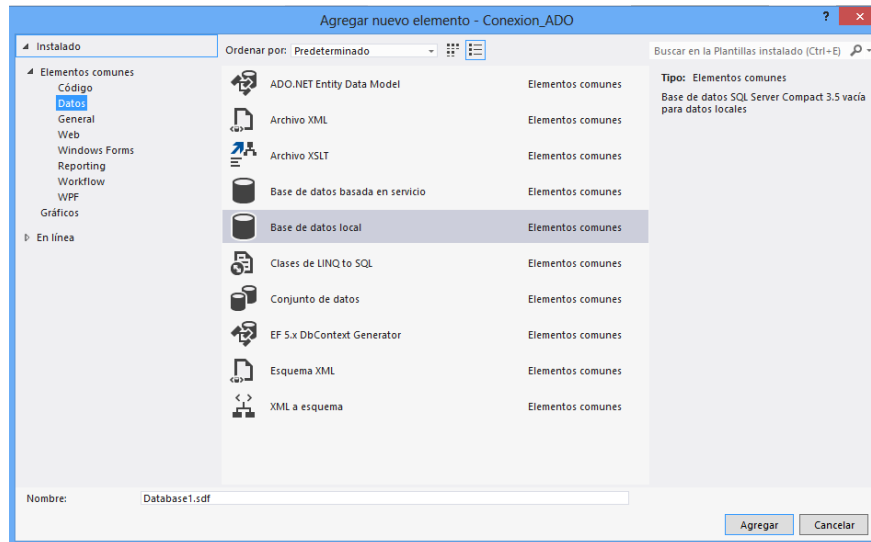
Dim dr As DataRow
dr = dt.NewRow() 'Se crea la nueva fila
dr("Region") = "Valparaíso" 'Se asigna el valor para cada una de las columnas
dr("Ciudad") = "San Felipe"
dt.Rows.Add(dr) 'Se adhiere a la tabla

dr = dt.NewRow()
dr("Region") = "Maule"
dr("Ciudad") = "Talca"
dt.Rows.Add(dr)

'Crea el dataset y le asigna la tabla creada
Dim set1 As DataSet = New DataSet("Ciudad")
set1.Tables.Add(dt)

'En el control DataGridView se le asigna la tabla recién creada
Me.DataGridView1.DataSource = set1.Tables(0).DefaultView
End Sub
End Class
```

- Otra forma de crear y utilizar un DataSet es creándolo como elemento del proyecto, pero para eso primero se debe tener una conexión a una base de datos, ya sea local o externa. En este caso se creará una base de datos local sdf. Se debe agregar un nuevo elemento en el proyecto, de tipo Datos llamado Base de Datos Local. Los pasos a seguir se muestran a continuación.



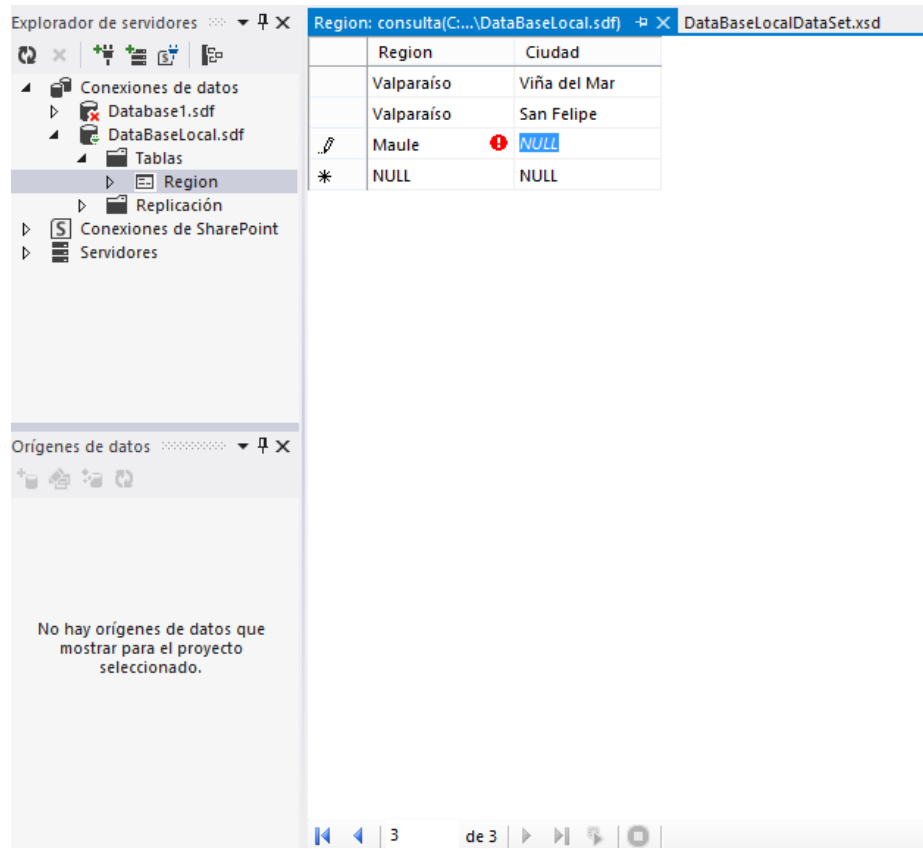
El asistente dará una serie de pasos para configurar y crear esta nueva base de datos local, los pasos a seguir son. Elección del modelo de base de datos, se elegirá Conjunto de Datos (DataSet), establecerá la conexión que usaremos que el asistente crea por defecto, y luego se le da un nombre a la Cadena de Conexión, luego un nombre al DataSet, como se muestra a continuación.

The figure displays four sequential screenshots of the 'Asistente para la configuración de orígenes de datos' (Data Source Configuration Assistant) window, which is used for configuring data sources in a database application.

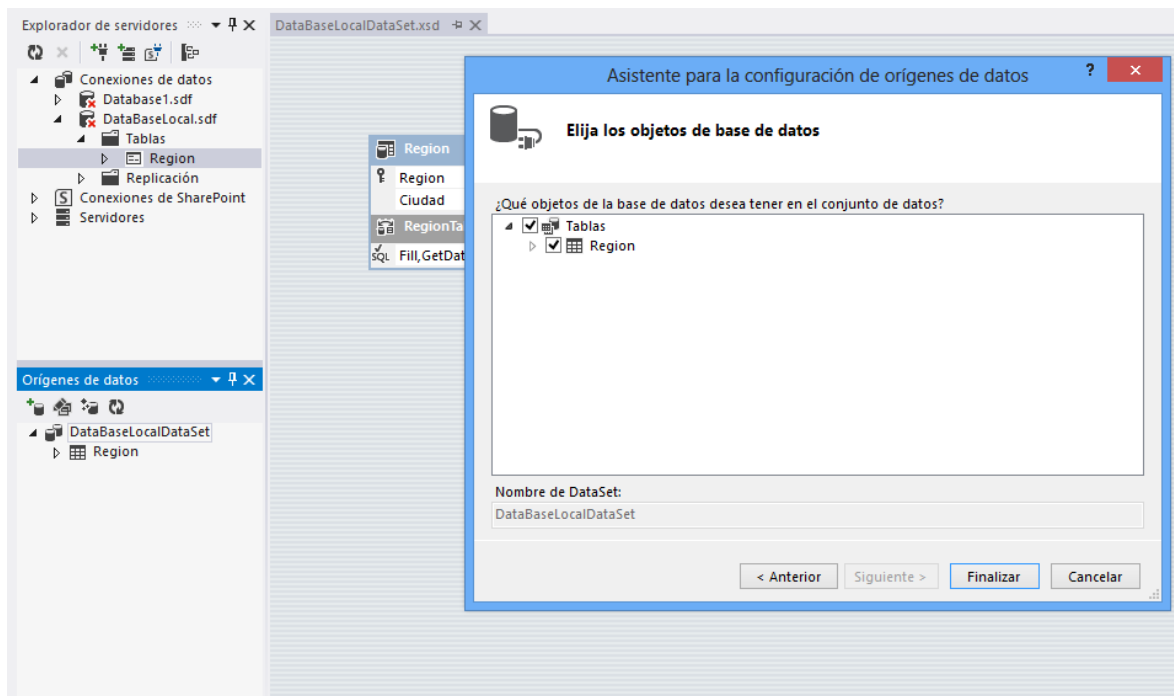
- Screenshot 1: Elegir un modelo de base de datos (Choose a database model).** The user is asked '¿Qué tipo de modelo de base de datos desea usar?' (Which type of database model do you want to use?). Two options are available: 'Conjunto de datos' (Data Set) and 'Entity Data Model'. The 'Conjunto de datos' option is selected. A note states: 'El modelo de base de datos que elija determina los tipos de objetos de datos que utiliza el código de la aplicación. Se agregará un archivo de conjunto de datos al proyecto.' (The database model you choose determines the types of data objects that the application code uses. A data set file will be added to the project.)
- Screenshot 2: Elegir la conexión de datos (Choose the data connection).** The user is asked '¿Qué conexión de datos debería utilizar la aplicación para conectarse a la base de datos?' (Which data connection should the application use to connect to the database?). The 'DataBaseLocal.sdf' connection is selected. A note explains that this connection might contain confidential data (e.g., passwords) and that storing such data in the connection string could pose a security risk. The user is asked if they want to include this data in the connection string. The 'No, excluir los datos confidenciales de la cadena de conexión. Estableceré esta información en el código de mi aplicación.' (No, exclude confidential data from the connection string. I will set this information in the code of my application.) option is selected.
- Screenshot 3: Guardar cadena de conexión en el archivo de config. de la aplicación (Save connection string in the application config file).** The user is asked '¿Desea guardar la cadena de conexión en el archivo de configuración de la aplicación?' (Do you want to save the connection string in the application configuration file?). The 'Sí, guardar la conexión como:' (Yes, save the connection as:) checkbox is checked. The text 'DataBaseLocalConnectionString' is entered in the provided field.
- Screenshot 4: Elija los objetos de base de datos (Choose the database objects).** The user is asked '¿Qué objetos de la base de datos desea tener en el conjunto de datos?' (Which database objects do you want in the data set?). A note states: 'La base de datos que ha seleccionado es nueva o no contiene objetos. Haga clic en Finalizar para crear un conjunto de datos vacío. Cuando agregue objetos de base de datos a la base de datos, abra el conjunto de datos vacío en el Diseñador de Dataset para volver a configurar el conjunto de datos.' (The database you have selected is new or does not contain objects. Click Finish to create an empty data set. When you add database objects to the database, open the empty data set in the Dataset Designer to reconfigure the data set.) The 'Nombre de DataSet:' (Data Set Name:) field contains 'DataSetBD'.

En el explorador de Servidores, y explorador de base de datos, se expande el nodo con la base de datos creada, bajo el nodo de Conexiones de Datos. En el menú tablas, se debe crear una nueva tabla, y se abrirá el diseñador de tablas. Una vez creados los campos, las claves, se guarda la tabla y se da un nombre para ella.

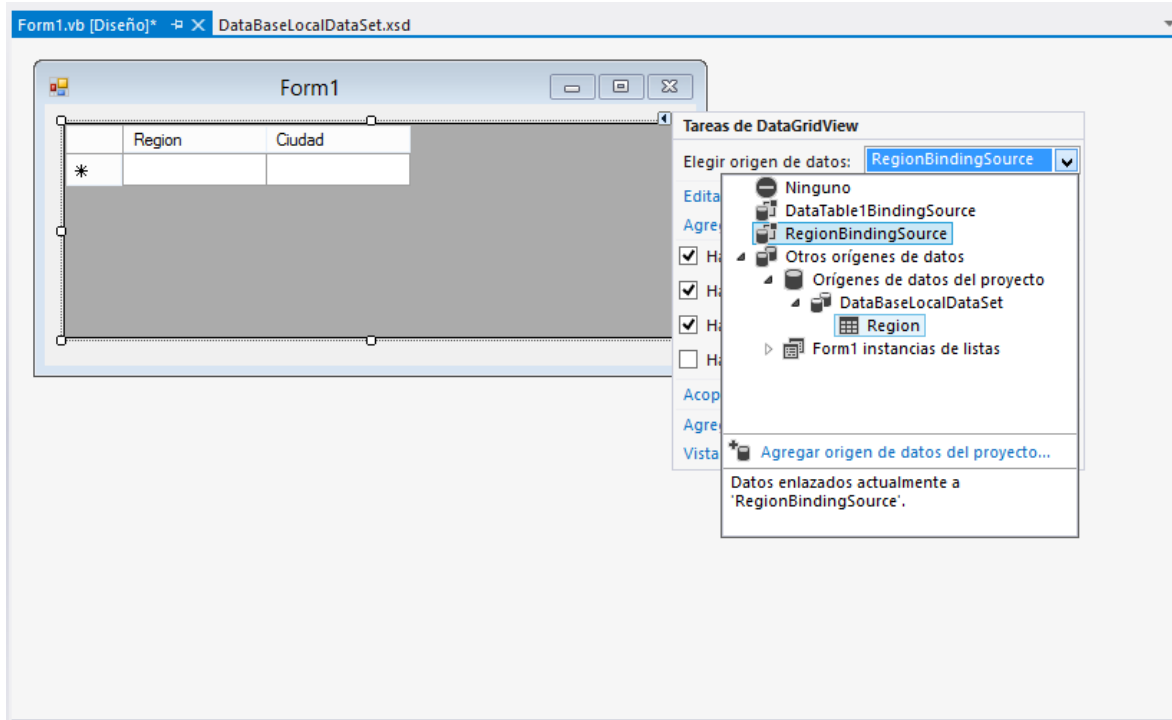
Para llenar la tabla con datos, se debe seleccionar la Tabla creada en el nodo Tablas de la base de datos, luego con el botón derecho seleccionar mostrar datos de tabla, el cual mostrará los datos que contenga, además de permitir ingresar nuevos registros, como se muestra a continuación.



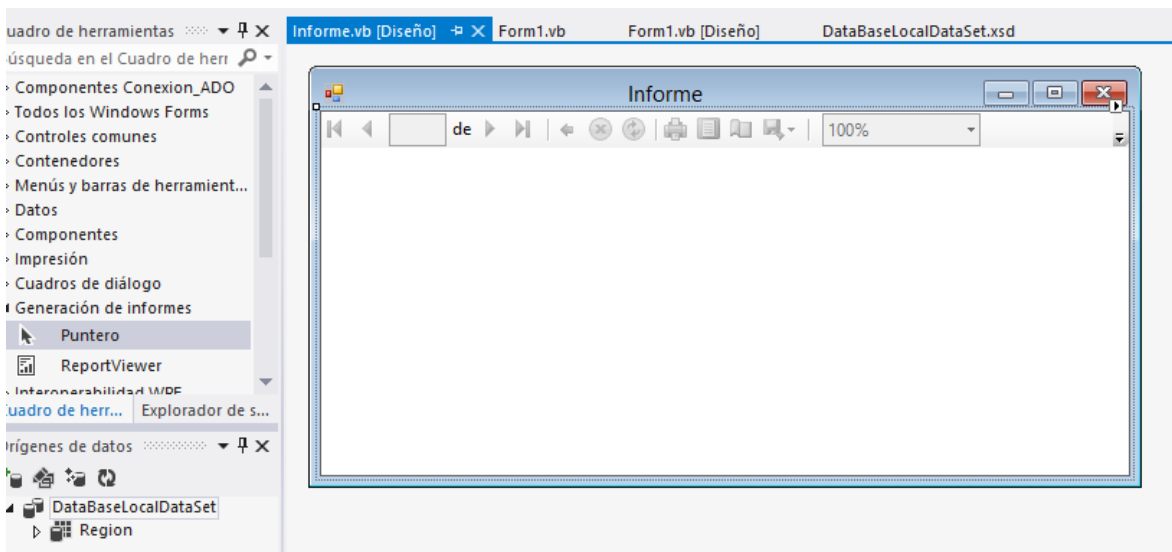
Luego en Explorador de Origen de datos se selecciona el DataSet, se selecciona configurar origen de datos con el asistente, el cual permitirá indicar de donde obtendremos la información, en la siguiente imagen se muestra este asistente.



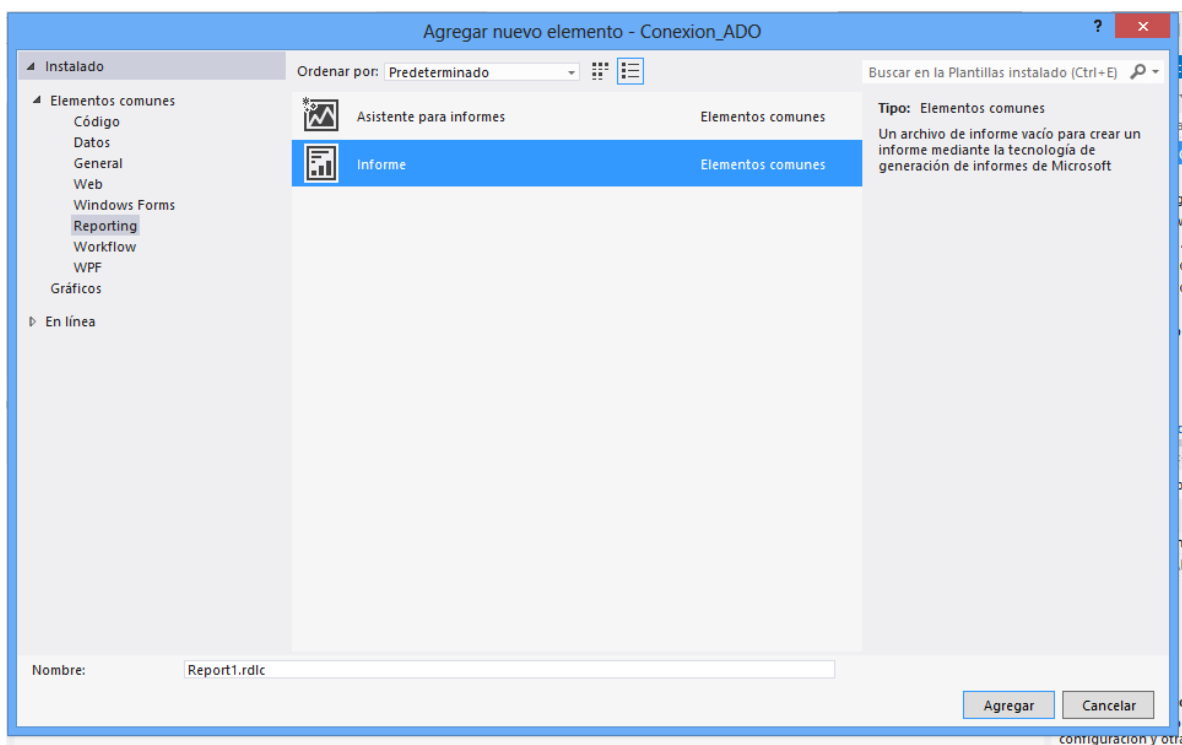
Una vez creada la base de datos, y el DataSet, en el formulario, en el vértice superior derecho del control DataGridView (grilla), se selecciona Elegir origen de datos, luego en Otro origen de datos, se selecciona la base de datos local, y el data set asociado, como se muestra a continuación. Con esto al ejecutar la aplicación, al momento de cargar el formulario se conectará con la base de datos local, y mostrara los datos según sea la consulta en el DataSet.



Además estos DataSet se pueden utilizar para crear informes. Para ello se debe incorporar en un formulario un control de tipo Generación de Informes llamado ReportViewer.

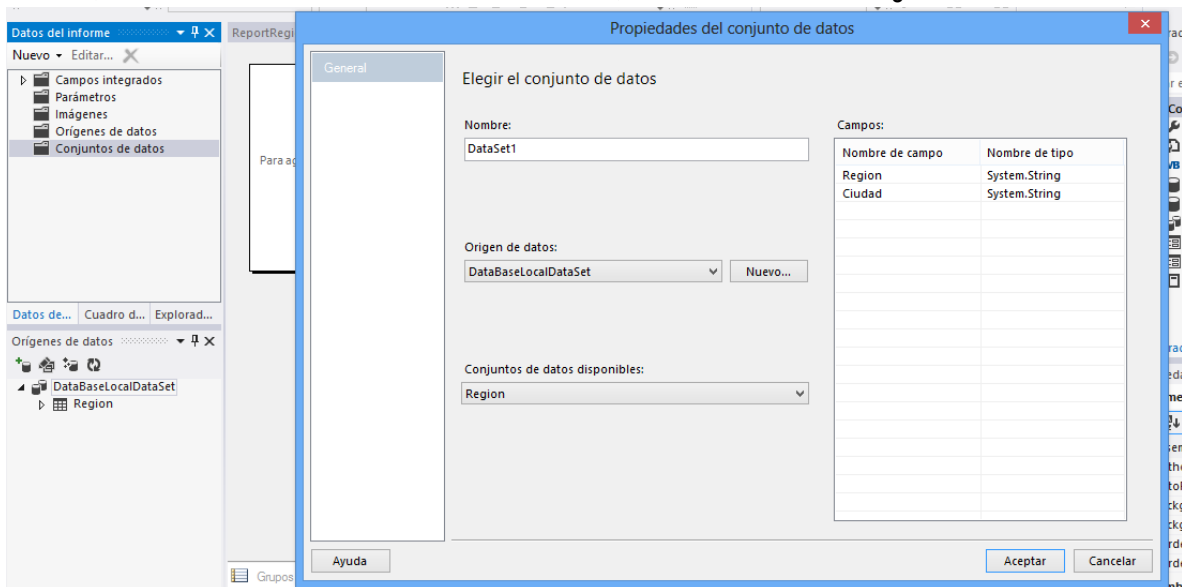


Una vez incorporado este control al formulario, se debe crear un reporte en el proyecto, como nuevo elemento de tipo Reporting llamado Informe, se le da un nombre, y se agrega al proyecto, como se muestra a continuación.

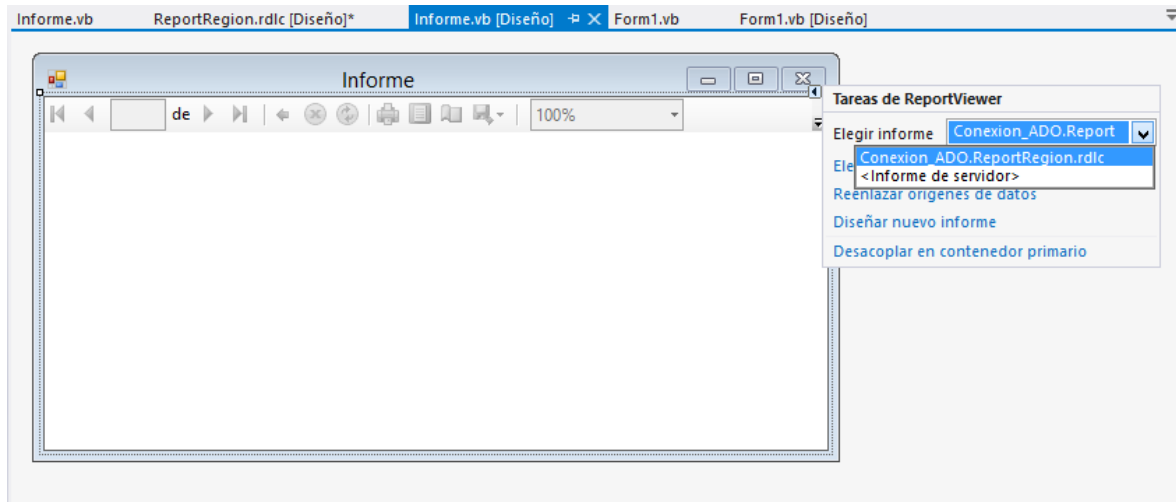


Luego abre el área de diseño del reporte, en el cual se puede incorporar encabezado, y pie de página, además de incorporar otro tipo de elementos como imágenes, líneas, dar formato de fuentes, entre otras posibilidades.

Para obtener la información, en Datos de informe, se agrega un nuevo conjunto de datos con el botón derecho, en el cual se mostrara un asistente de conexión, como se muestra en la figura.



Una vez creada la conexión se muestran los campos disponibles del DataSet, estos campos se arrastran al informe en donde queramos aparezcan. Una vez incorporado los elementos en el informe, se guardan los cambios, y en el control Report Viewer del formulario se elige el reporte creado, como se muestra a continuación.



9. APRENDIZAJE ESPERADO: Diseñan y programan proyectos de alta complejidad para aplicaciones que dan mantención a una base de datos.

- Como se ha visto podemos crear una base de datos local de tipo sdf, así como también se puede acceder a una base de datos externa. Para esto se debe crear una cadena de conexión, esta depende del modo de autenticación que tenga la base de datos a la que se quiere acceder. Esta cadena de conexión se crea automáticamente con el asistente de conexión a través de un control, o bien a través de código en una clase, importando las librerías necesarias para trabajar.

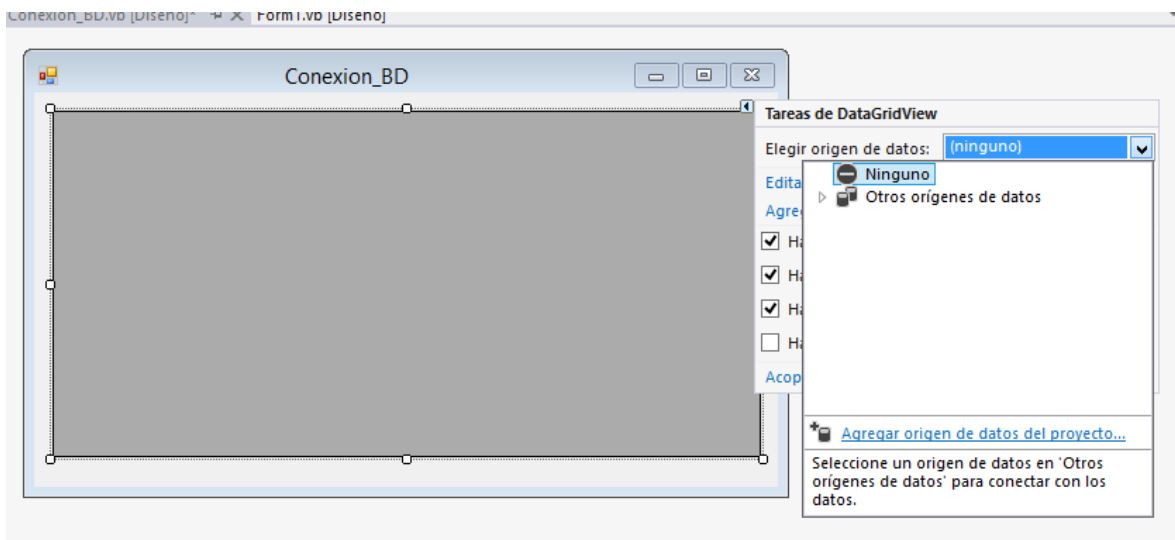
- Modo autenticación Windows se debe crear la siguiente cadena de conexión.

Data Source = **ServidorSQL**; Initial Catalog = **BaseDatos**; Integrated Security = True

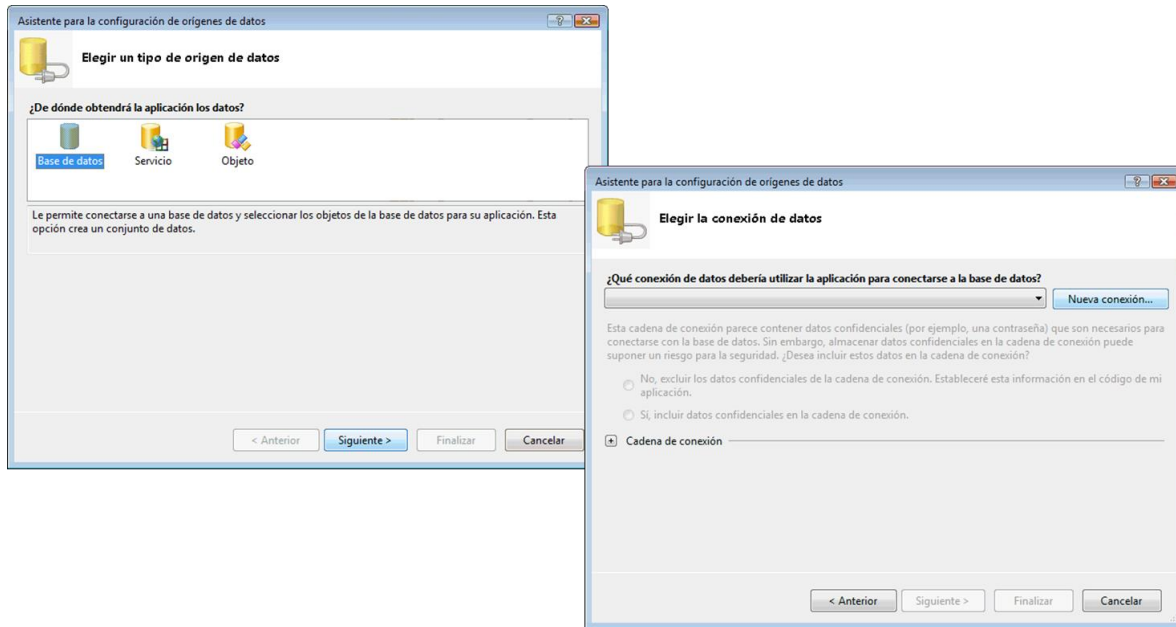
- Modo de autenticación con SQL Server

data source = **ServidorSQL**; initial catalog = **BaseDatos**; user id = **Usuario**;
password = **Contraseña**

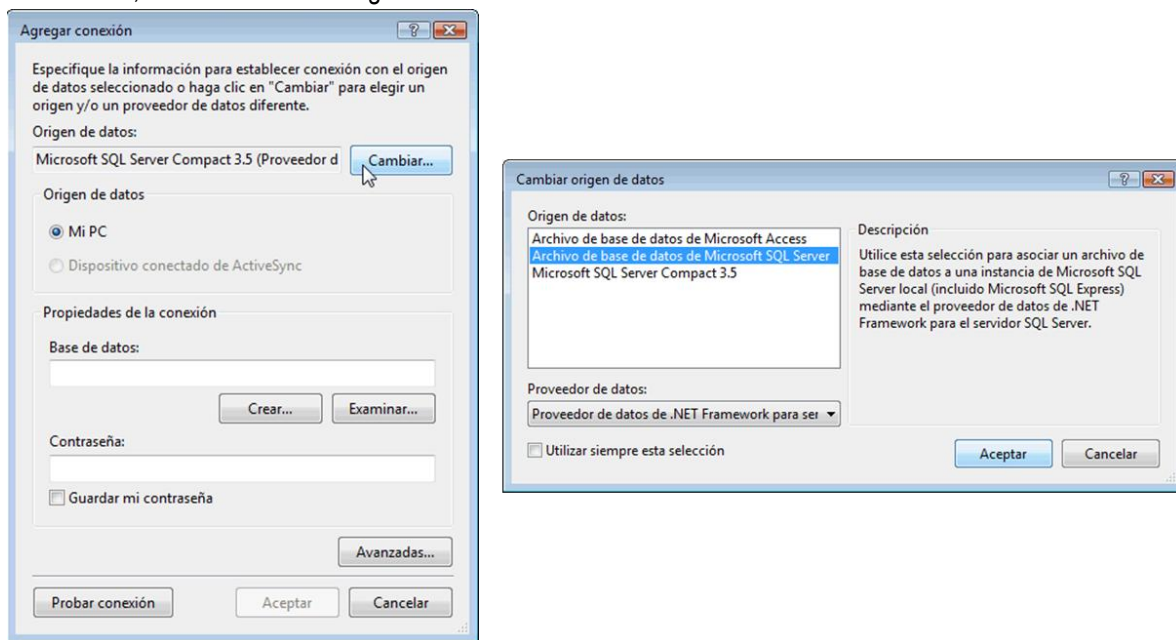
Conectar a través de un control a través de un asistente, en un formulario con un control de tipo grilla, se selecciona Elegir Origen de Datos en el vértice superior derecho, seleccionar Agregar origen de datos del proyecto.



Se despliega el asistente de conexión, el cual permite crear una cadena de conexión como lo muestra los siguientes pasos.



Luego se debe seleccionar el proveedor de conexión a base de datos, que será Archivo de Base de Datos SQL Server, como lo muestra la figura



Luego se crea o se selecciona la base de datos con la que se trabajará, y por último se le da un nombre a la conexión, como muestra la imagen.

Agregar conexión

Especifique la información para establecer conexión con el origen de datos seleccionado o haga clic en "Cambiar" para elegir un origen y/o un proveedor de datos diferente.

Origen de datos:
Archivo de base de datos de Microsoft SQL Serv Cambiar...

Nombre del archivo de la base de datos (nuevo o existente):
nueva Examinar...

Conexión con el servidor

☒ Usar autenticación de Windows

☐ Usar autenticación de SQL Server

Nombre de usuario:

Contraseña:

☐ Guardar mi contraseña

Avanzadas...

Probar conexión Aceptar Cancelar

Asistente para la configuración de orígenes de datos

Guardar cadena de conexión en el archivo de config. de la aplicación

El almacenamiento de las cadenas de conexión del archivo de configuración de aplicación facilita el mantenimiento y la implementación. Para guardar la cadena de conexión en el archivo de configuración de la aplicación, escriba un nombre en el cuadro y, a continuación, haga clic en Siguiente.

¿Desea guardar la cadena de conexión en el archivo de configuración de la aplicación?

☒ Sí, guardar la conexión como:

nuevaConnectionString

< Anterior Siguiente > Finalizar Cancelar

Luego se seleccionan la o las tablas de donde se obtendrá la información, además de las columnas que se mostrarán para armar la consulta.

Asistente para la configuración de orígenes de datos

Elegir la conexión de datos

¿Qué conexión de datos debería utilizar la aplicación para conectarse a la base de datos?

nuevaConnectionString (MySettings) Nueva conexión...

Esta cadena de conexión parece contener datos confidenciales (por ejemplo, una contraseña) que son necesarios para conectarse con la base de datos. Sin embargo, almacenar datos confidenciales en la cadena de conexión puede suponer un riesgo para la seguridad. ¿Desea incluir estos datos en la cadena de conexión?

☐ No, excluir los datos confidenciales de la cadena de conexión. Estableceré esta información en el código de mi aplicación.

☐ Sí, incluir datos confidenciales en la cadena de conexión.

☒ Cadena de conexión

Data Source=.\SQLEXPRESS;Initial Catalog=Northwind;Integrated Security=True

< Anterior Siguiente > Finalizar

Asistente para la configuración de orígenes de datos

Elija los objetos de base de datos

¿Qué objetos de la base de datos desea tener en el conjunto de datos?

☒ Tablas

- ☐ Categories
- ☐ CustomerCustomerDemo
- ☐ CustomerDemographics
- ☐ Customers
- ☒ Employees
- ☐ EmployeeTerritories
- ☐ Order Details
- ☐ Orders
- ☐ Products
- ☐ Region
- ☐ Shippers
- ☐ Suppliers
- ☐ Territories

☐ Vistas

☐ Procedimientos almacenados

Nombre de DataSet:
NorthwindDataSet

< Anterior Siguiente > Finalizar Cancelar

Luego al ejecutar la aplicación cada vez que se cargue el formulario se mostraran los datos de la conexión que se creó con el asistente de conexión.

Otra forma de conexión a base de datos es a través de código, utilizando las librerías, el siguiente código ejemplifica como utilizar la conexión y las sentencias SQL que se pueden utilizar.

Lo primero es crear una clase, que genere la conexión a la base de datos, y luego es utilizar los comandos de conexión, ejecución a una base de datos de tipo SQL Server, como a continuación se demuestra.

```
Productor.vb  X
Imports System.Data
Imports System.Data.SqlClient

Public Class Productor
    Function conexion() As String 'funcion que retorna un string
        'Se arma el string de conexion
        Dim str As String = "Data Source=SERVIDOR;Initial Catalog=AgroProd;Integrated Security=True"
        Return str
    End Function

    Function getDatosProductor(ByVal id As String) As DataSet 'funcion que retorna un conjunto de datos
        'Declaracion de variable para la conexión a Base de Datos de tipo SQL Server
        Dim sqlCon As New SqlConnection
        Dim Con As New Conexion
        sqlCon.ConnectionString = conexion() 'Se le asigna el string de conexion instanciando la funcion
        sqlCon.Open() 'se abre la conexión

        Dim sqlCom As New SqlCommand 'Declara variable que genera los comandos a ejecutar en la base de datos.
        sqlCom.Connection = sqlCon
        sqlCom.CommandType = CommandType.Text
        'Acá se crea la sentencia SQL
        sqlCom.CommandText = "select NOMBRE_PRODUCTOR,NUMERO_TELEFONO from productor where ID_PRODUCTOR = '" & id & "'"

        'Variable que ejecuta la sentencia y llena el dataset con datos
        Dim sqlDpa As New SqlDataAdapter(sqlCom)
        Dim ds As New DataSet
        sqlDpa.Fill(ds)
        sqlCon.Close() 'Se cierra la conexión
        Return ds 'Retorna el dataSet
    End Function
```

Otras funciones con sentencias SQL, como insert y update, se demuestran en el siguiente ejemplo.

```

Productor.vb
Productor
updateProductor

Function insertProductor(ByVal id As String, ByVal nombre As String, ByVal telefono As Integer) As Integer
    Dim sqlCon As New SqlConnection
    Dim Con As New Conexion
    sqlCon.ConnectionString = conexion()
    sqlCon.Open()

    Dim sqlCom As New SqlCommand
    sqlCom.Connection = sqlCon
    sqlCom.CommandType = CommandType.Text
    sqlCom.CommandText = "insert into PRODUCTOR (ID_PRODUCTOR,nombre_productor,Numero_Telefono) " & _
        "values('" & id & "','" & nombre & "','" & telefono & "')"

    Dim i As Integer = sqlCom.ExecuteNonQuery()
    sqlCon.Close()
    Return i
End Function

Function updateProductor(ByVal id As String, ByVal nombre As String, ByVal telefono As Integer) As Integer
    Dim sqlCon As New SqlConnection
    Dim Con As New Conexion
    sqlCon.ConnectionString = conexion()
    sqlCon.Open()

    Dim sqlCom As New SqlCommand
    sqlCom.Connection = sqlCon
    sqlCom.CommandType = CommandType.Text
    sqlCom.CommandText = "update PRODUCTOR set nombre_productor = '" & nombre & "',Numero_Telefono = '" & _
        "& telefono & "where ID_PRODUCTOR = '" & id & "'"

    Dim i As Integer = sqlCom.ExecuteNonQuery()
    Return i
End Function

```

Para usar estas funciones y llenar una grilla con datos se utiliza a través de un evento, como por ejemplo el evento click de un botón, como de la siguiente forma.

```

Public Class ConsultarProductor

    Private Sub Volver_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Volver.Click
        Principal.Show()
        Me.Hide()
    End Sub

    Private Sub botonConsultar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles botonConsultar.Click
        Dim objProd As New Productor
        'Se instancia la clase productor, y se utiliza la funcion
        'que obtiene los datos, pasando los parametros necesarios.
        Dim ds As DataSet = objProd.getDatosProductor(Me.txtIdUser.Text)
        'Se evalúa si el conjunto de datos contiene registros
        If ds.Tables(0).Rows.Count > 0 Then
            'Si tiene datos se traspasan los datos a la grilla
            Me.DataGridView1.DataSource = ds.Tables(0).DefaultView
        End If
    End Sub
End Class

```