UNIVERSIDAD NACIONAL DE EDUCACIÓN
Enrique Guzmán y Valle
Alma Máter del Magisterio Nacional

FACULTAD DE CIENCIAS
Escuela Profesional de Matemática e Informática

MONOGRAFÍA

PROGRAMACIÓN VISUAL.NET

Conceptos de programación visual, fundamentos del Visual NET, elementos de MS Visual, sentencias de control, procedimientos y matrices de controles, acceso a una base de datos, asistente para aplicaciones, aplicaciones

Examen de Suficiencia Profesional Resolución N° 1064-2018-D-FAC

Presentada por

León Pedro LAUREANO JULCA

Para optar al Título Profesional de Licenciado en Educación

Especialidad: Matemática e Informática

Lima, Perú

2018

MONOGRAFÍA

PROGRAMACIÓN VISUAL.NET

Conceptos de programación visual, fundamentos del Visual NET, elementos de MS Visual, sentencias de control, procedimientos y matrices de controles, acceso a una base de datos, asistente para aplicaciones, aplicaciones

Designación de Jurado Resolución N° 1064-2018-D-FAC

Dr. Richard Santiago QUIVIO CUNO
PRESIDENTE

Dr. Lolo José CABALLERO CIFUENTES
SECRETARIO

Dr. Guillermo Pastor MORALES ROMERO
VOCAL

Línea de Investigación: Tecnología y Soportes Educativos

Agradecimiento

El presente trabajo agradezco a Dios por ser mi guía y acompañarme en el transcurso de mi vida, brindándome paciencia y sabiduría para culminar con éxito mis metas propuestas.

Dedicatoria

Con todo mi cariño y mi amor para las personas que hicieron todo lo que estuvo a su alcance, para que yo pudiera lograr uno de mis sueños, por motivarme y levantarme cuando sentía que el camino se terminaba; por ellos por siempre mi corazón y mi agradecimiento.

Mi familia

Tabla de Contenidos

Pág.

Índice de Tablas

Pág.

Índice de Figuras

Pág.

Introducción

Se llama programación a la implementación de un algoritmo en un determinado lenguaje de programación, para realizar un programa. Algoritmo es una secuencia no ambigua, finita y ordenada de instrucciones que han de seguirse para resolver un problema. Programa (Software en inglés) es una secuencia de instrucciones que una computadora puede interpretar y ejecutar. El proceso de creación de software es materia de la ingeniería del software, una de las ramas propias de la Ingeniería Informática.

Se han propuesto diversas técnicas de programación, cuyo objetivo es mejorar tanto el proceso de creación de software como su mantenimiento. Entre ellas se pueden mencionar las programaciones lineales, estructurada, modular y orientada a objetos.

Lenguaje de programación es el idioma utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

Hay muchos lenguajes de programación, pero para programar no es necesario conocer todos los lenguajes, es igual que cuando hablamos, podemos comunicarnos en español, aunque no sepamos alemán. Aunque la palabra debería ser idioma ya que lenguaje realmente abarca todos los idiomas, pero en computación equivocadamente se usa el término lenguaje cuando el término correcto es idiomas de programación.

En la actualidad los lenguajes de programación están escritos para ser comprensibles por el ser humano, a este código se le llama código fuente, pero no es comprendido por la máquina ya que esta solo maneja el lenguaje binario.

Visual Basic. Net (VB.NET) es un lenguaje de programación orientado a objetos que se puede considerar una evolución de Visual Basic implementada sobre el framework .NET. Su introducción resultó muy controvertida, ya que debido a cambios significativos en el

lenguaje VB.NET no es retro compatible con Visual Basic, pero el manejo de las instrucciones es similar a versiones anteriores de Visual Basic, facilitando así el desarrollo de aplicaciones más avanzadas con herramientas modernas. Para mantener eficacia en el desarrollo de las aplicaciones, la gran mayoría de programadores de VB.NET utilizan el entorno de desarrollo integrado Microsoft Visual Studio en alguna de sus versiones (desde el primer Visual Studio .NET hasta Visual Studio .NET 2017, que es la última versión de Visual Studio para la plataforma .NET), aunque existen otras alternativas, como SharpDevelop (que además es libre).

Capítulo I: Conceptos Generales sobre Programación

1.1. Programación Visual

1.1.1. Definición

La programación visual se define comúnmente como el uso de expresiones visuales (tales como gráficos, animación o iconos) en el proceso de la programación, pueden ser utilizadas para formar la sintaxis de los nuevos lenguajes de programación visuales que conducen a los nuevos paradigmas tales como programación por la demostración; o pueden ser utilizadas en las presentaciones gráficas del comportamiento o de la estructura de un programa.

La programación visual brinda los conocimientos necesarios para diseñar y desarrollar aplicaciones con un entorno visual amigable y fácil de utilizar por el usuario. Los lenguajes de programación visual, como Visual Basic, hacen sencilla la tarea de los programadores porque antes constituía una gran demora tiempo en el diseño de ventanas o formularios.

1.1.2. Lenguajes visuales

La programación visual se basa en el uso de lenguajes visuales (LVP). Un lenguaje de programación visual puede definirse como:

- Un lenguaje de programación que usa una representación visual (tal como gráficos, dibujos, animaciones o iconos, parcial o completamente).
- Un lenguaje visual manipula información visual o soporta interacción visual, o permite programar con expresiones visuales.
- Un lenguaje visual es un conjunto de arreglos espaciales de símbolos de texto y gráficos con una interpretación semántica que es usada para comunicar acciones en un ambiente.
- Los LVPs son lenguajes de programación donde se usan técnicas visuales para expresar relaciones o transformaciones en la información.

Por ejemplo, un objeto visual que representa un proceso de adición (suma) toma dos entradas y produce una salida. En un LVP típico de flujo de datos, el usuario simplemente selecciona un valor de entrada y selecciona un puerto de entrada al objeto para establecer una relación entre los datos y el proceso.

El génesis de los LPV vino en 1975 con la publicación de David Canfield Smith "Pygmalion: A Creative Programming Environment". Por ejemplo, Pygmalion incorporó un paradigma de programación basado en iconos en el cual los objetos creados podían ser modificados y conectados juntos, con las características definidas para realizar cómputos.

Muchos LPV modernos emplean un acercamiento basado en iconos como el de Smith. Pygmalion también hizo uso el concepto de programación, por ejemplo, en donde el usuario muestra al sistema cómo realizar una tarea en un caso específico y el sistema utiliza esta información para generar un programa el cuál realiza la tarea en casos generales.

¿Por qué insistimos en comunicarnos con las computadoras usando lenguajes de programación textuales? ¿No sería mejor comunicarnos con las computadoras usando una representación que aproveche nuestra naturaleza visual?

Obviamente, los autores de los lenguajes de programación visuales (LPV) discuten que la respuesta a ambas preguntas es sí. Las principales motivaciones para la mayoría de la investigación en LPV son:

- Mucha gente piensa y recuerda cosas en términos de cuadros.
- Ella se relaciona con el mundo de una manera intrínsecamente gráfica y utiliza imágenes como componente primario del pensamiento creativo.
- Además, los lenguajes de programación textuales han demostrado ser algo difíciles para que mucha gente creativa e inteligente aprenda utilizar con eficacia.

- La reducción o eliminación de la necesidad de traducir ideas visuales en representaciones textuales puede ayudar a atenuar este problema de la curva del aprendizaje.
- Además, una variedad de aplicaciones, incluyendo la visualización científica y la simulación interactiva se prestan bien a los métodos visuales de desarrollo.

Un LPV no es un ambiente integrado de desarrollo (o IDE). La diferencia es que un VPL debe ser capaz de llevar a cabo todas las tareas de programación de forma visual, sin tener que recurrir a la representación textual.

1.1.3. Clasificación de los lenguajes visuales

Los lenguajes visuales se dividen en las siguientes categorías:

- Lenguajes puramente visuales.
- Sistemas híbridos de texto y elementos visuales.
- Sistema de programación por ejemplo (Programming-by-example)
- Sistemas orientados a restricciones (Constraint-oriented systems)
- Sistemas basados en formas (entradas tipo “hoja de cálculo” animada)

1.1.4. Principales conceptos de los lenguajes visuales

Los principales conceptos relativos a los lenguajes visuales son:

- Abstracción procedural (descomposición del programa en módulos)
- Abstracción de datos, consiste en encapsular los tipos de datos abstractos definidos por el usuario, permitiendo su acceso solo por medio de operaciones definidas.
- Sistema de tipos.
- Control de flujo.
- Estructuras de control (selección, iteración)
- Formatos y estándares. Se refiere a la definición de reglas que permitan especificar los elementos del LPV (p.ej. GXL, XGMML, GML)

- Gramáticas de lenguajes visuales. Los lenguajes visuales se especifican por una tripleta (ID, G, B), donde:
  - ID es el diccionario de iconos, el cual es un conjunto de iconos generalizados, cada uno representado por un par (Xm,Xi) , donde Xm es la parte lógica (significado) y Xi es la parte física (imagen).
  - G es una gramática que especifica cómo pueden construirse objetos compuestos a partir de iconos simples usando operadores de relación espaciales.
  - B es una base de conocimiento de dominio específico, la cual contiene la información necesaria para construir el significado de la sentencia visual (nombres de eventos, relaciones, referencias a los objetos, etc.).

1.1.5. Caso de estudio implementación de una neurona artificial con Sanscript

Para demostrar el uso de un LPV, se realizó el desarrollo de una neurona artificial (NA) usando la herramienta Sanscript. Cabe mencionar que Sanscript implementa las siguientes abstracciones fundamentales:

- Funciones, que son los bloques con los que se construyen las aplicaciones en Sanscript. Una función tiene entradas, ejecuta un cálculo y produce salidas.
- Flujogramas (flowgrams), que son diagramas de funciones ligadas con funciones donde se especifica una acción (equivalentes a un programa textual, solo que visual).
- Conexiones, que son los vínculos (relaciones) entre funciones.
- Aplicaciones, que son programas que se ejecutan fuera de la herramienta

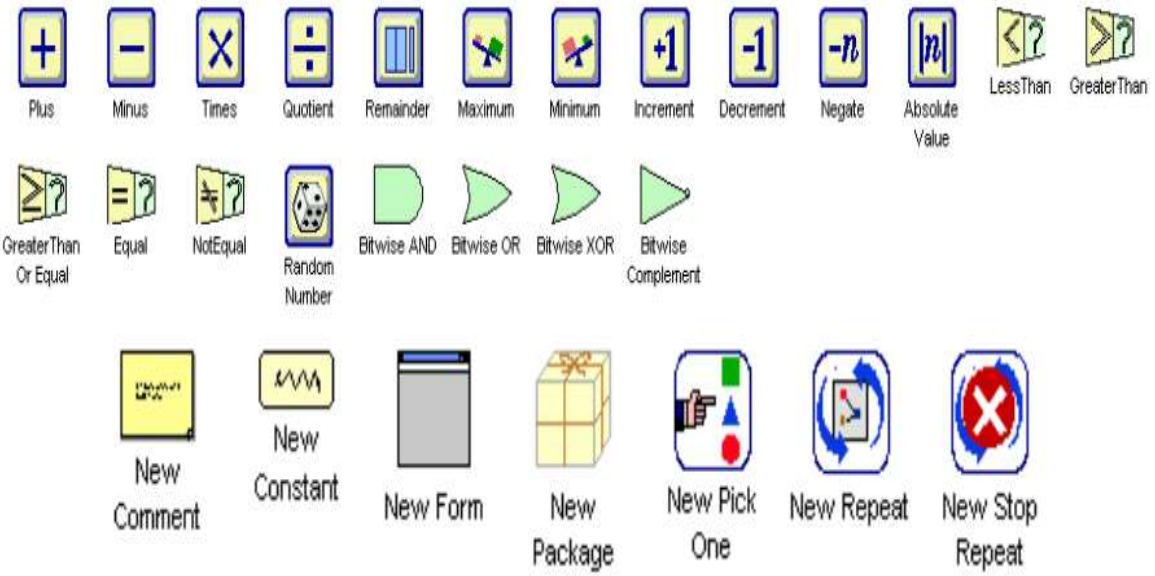


Figura 1 Elementos de programación Sanscript

Fuente: Elaboración propia

Una neurona artificial consta básicamente de 4 partes, que son:

- Las entradas.
- Los pesos.
- Una función que calcula la suma ponderada de los pesos y las entradas.
- La función de transferencia.

La neurona artificial implementada solo tiene 2 entradas y una salida, pero el modelo puede ampliarse fácilmente.

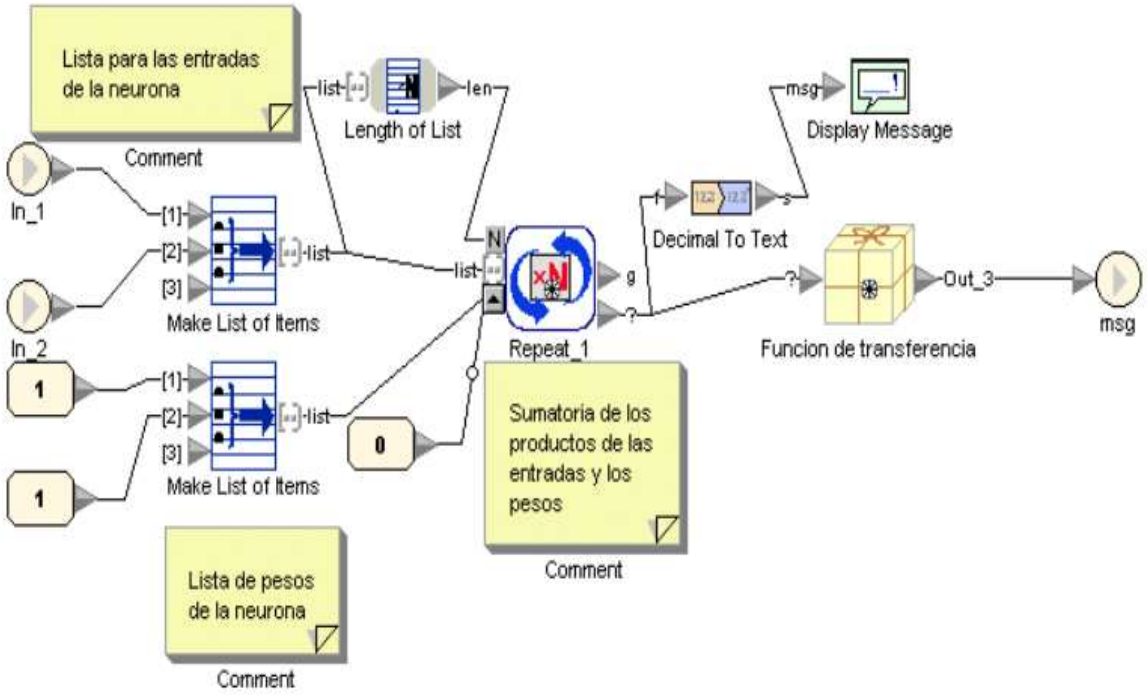


Figura 2 Flujograma del modelo de una neurona artificial

Fuente: Elaboración propia

Cada una de estas partes se implementó de la siguiente manera:

- Lista de entradas. Debido a que podemos tener N entradas en la neurona lo más conveniente es tener una lista para que tengamos las entradas en una sola estructura de datos y poder obtener de esa estructura los valores para realizar la suma de los pesos por las entradas.
- Lista de pesos. Como debe tenerse la misma cantidad de entradas que de pesos para poder realizar la suma ponderada, hay que contar con una estructura de datos igual a la de las entradas, pero ahora con los pesos.

1.2. Lenguajes de Programación

Todo lo que vemos ni bien pulsamos el botón de encendido de nuestra computadora, es decir el sistema operativo, sus aplicaciones, y las partes más pequeñas que lo conforman

como cuadros de diálogo, menús, ventanas y botones, tienen su nacimiento en los llamados lenguajes de programación, los cuales son básicamente programas con la habilidad, mediante una serie de reglas sintácticas y semánticas compuestas por palabras, números y expresiones matemáticas, de crear el llamado código fuente, el cual una vez compilado, se convertirá en un programa o software y podrá ser ejecutado en nuestra computadora sin necesidad de que el usuario lleve a cabo ningún otro paso.

También la palabra programación se define como el proceso de creación de un programa de computadora, mediante la aplicación de procedimientos lógicos, a través de los siguientes pasos:

- El desarrollo lógico del programa para resolver un problema en particular.
- Escritura de la lógica del programa empleando un lenguaje de programación específico (codificación del programa).
- Ensamblaje o compilación del programa hasta convertirlo en lenguaje de máquina.
- Prueba y depuración del programa.
- Desarrollo de la documentación.

Existe un error común que trata por sinónimos los términos 'lenguaje de programación' y 'lenguaje informático'. Los lenguajes informáticos engloban a los lenguajes de programación y a otros más, como por ejemplo HTML (lenguaje para el marcado de páginas Web que no es propiamente un lenguaje de programación, sino un conjunto de instrucciones que permiten estructurar el contenido de los documentos).

Permite especificar de manera precisa sobre qué datos debe operar una computadora, cómo deben ser almacenados o transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural. Una característica relevante de los lenguajes de programación es precisamente que más de un programador pueda usar un conjunto común

de instrucciones que sean comprendidas entre ellos para realizar la construcción de un programa de forma colaborativa.

1.2.1. Evolución de los lenguajes de programación

La historia de los lenguajes de programación se remonta hasta 1840, cuando Ada Lovelace, considerada una pionera en este ámbito, dejara en un reconocido trabajo llamado "Notas" el puntapié inicial de todo lo que hoy conocemos en materia de lenguajes de computadoras, y que ha sido muy importante en los orígenes de esta materia. Los años han pasado, y desde esos primeros esbozos, los lenguajes de programación han avanzado muchísimo, como así también sus capacidades y el resultado de lo que pueden ofrecer.

Si bien Ada Lovelace fue la primera persona que trabajó en este tipo de investigaciones, los verdaderos avances se hicieron muchos años después, a partir de la década de 1940, momento en que la computación comenzaba a desarrollarse.

Precisamente en 1946 surgió ENIAC, iniciales de "Electronic Numerical Integrator And Computer", que en español significa "Computadora e Integrador Numérico Electrónico", y que fuera utilizada por el Laboratorio de Investigación Balística del Ejército de los Estados Unidos. Obviamente, con estas primeras computadoras surgió la necesidad de programarlas para que hicieran lo que se les solicitaba.

Para la inmensa mayoría de los usuarios, el uso de una computadora es lo más sencillo y transparente que existe, y no debe preocuparse por aspectos técnicos relacionados con cómo los programas que utiliza a diario son desarrollados y diseñados. Esto es así desde hace años, y es totalmente aceptable que sea de esta manera, ya que lo que subyace debajo de las ventanas y cuadros de diálogo de una aplicación no debe interferir nunca con la productividad de quien use el software, sino que es responsabilidad de los ingenieros y desarrolladores.

Todo lo que vemos ni bien pulsamos el botón de encendido de nuestra computadora, es decir el sistema operativo, sus aplicaciones, y las partes más pequeñas que lo conforman como cuadros de diálogo, menús, ventanas y botones, tienen su nacimiento en los llamados lenguajes de programación, los cuales son básicamente programas con la habilidad, mediante una serie de reglas sintácticas y semánticas compuestas por palabras, números y expresiones matemáticas, de crear el llamado código fuente, el cual una vez compilado, se convertirá en un programa o Software y podrá ser ejecutado en nuestra computadora sin necesidad de que el usuario lleve a cabo ningún otro paso.

Lenguaje de máquina

El lenguaje de máquina, también llamado código máquina es un sistema de códigos diseñado para ser reconocido y ejecutado en forma directa por un circuito micro programable, tal como el de un microprocesador de una computadora o de microcontroladores en máquinas de control numérico, por ejemplo. Básicamente, este lenguaje de máquina está compuesto por un set de instrucciones que determinan el comportamiento de una maquinaria o controlador. Un programa en lenguaje máquina es una cadena de estas instrucciones más los datos sobre los que arrojará los resultados.

Sin embargo, el lenguaje de máquina es expresado completamente en código binario, es decir 1 y 0, y por lo tanto muy complejo de implementare, y debe ser traducido para que los humanos puedan interrelacionarse con el mismo. Fue para ello que se desarrolló el lenguaje ensamblador, que posibilita traducir estas extensas cadenas numéricas en palabras como “Add”, “Sub”, “Mul” y “Call”, a las que posteriormente se les denominó “Instrucciones” y que operaban directamente a nivel de hardware.

Lenguaje ensamblador

Assembly Language o Lenguaje Ensamblador por su traducción al castellano, es un lenguaje de programación para todo tipo de procesadores y controladores, que es capaz de

interpretar y manipular mediante una representación simbólica de los códigos de máquina binarios y hacerlos de alguna manera “más accesibles” a los programadores.

```
; Example of IBM PC assembly language
; Accepts a number in register AX;
; subtracts 32 if it is in the range 97-122;
; otherwise leaves it unchanged.

SUB32  PROC         ; procedure begins here
       CMP  AX,97   ; compare AX to 97
       JL   DONE    ; if less, jump to DONE
       CMP  AX,122  ; compare AX to 122
       JG   DONE    ; if greater, jump to DONE
       SUB  AX,32   ; subtract 32 from AX
DONE:  RET          ; return to main program
SUB32  ENDP         ; procedure ends here
```

Figura 3 Lenguaje ensamblador

Fuente: Elaboración propia

El lenguaje ensamblador fue utilizado en los albores de las ciencias de la computación, cuando todavía no habían sido desarrollados lenguajes más potentes y flexibles. No obstante, todavía es utilizado a nivel académico y cuando es necesario tener acceso directo al hardware, como en el caso de los sistemas operativos y los controladores de dispositivos para impresoras, escáneres y otros tipos de periféricos. Cabe destacar que el lenguaje ensamblador es considerado como un lenguaje de bajo nivel.

Lenguajes de alto y bajo nivel

También existe un segundo tipo de lenguaje de programación, o “lenguaje de Alto Nivel”, que se distingue del primero debido a que tiene la capacidad de poder expresarse de manera análoga al lenguaje de los humanos, es decir que pueden representar los algoritmos de una manera adecuada a la capacidad cognitiva de las personas.

El primer lenguaje de programación de alto nivel que les permitió a los programadores una flexibilidad nunca antes vista fue Fortran, creado en el año 1957, precisamente como una alternativa de lenguaje de alto nivel al lenguaje ensamblador para programar la mainframe IBM 704, lo que permitiría agilizar los tiempos de programación de dichas máquinas. Este debe considerase como un verdadero hito en la historia de los lenguajes de programación, ya que antes de Fortran, los programas sólo se desarrollaban en lenguaje ensamblador.

1.2.2. Implementación de los lenguajes de programación

A lo largo de los años, y a medida que eran necesarios lenguajes de programación más potentes y flexibles para llevar a cabo las tareas complejas que las computadoras modernas podían procesar, al lenguaje ensamblador y a Fortran le siguieron LISP, COBOL, ALGOL, PASCAL, BASIC, C, dBASE, ADA, JAVA, PHP, C++, DELPHI y otros, que le abrieron la puerta a la computación tal y como la conocemos ahora.

En la actualidad, existen alrededor de 2000 lenguajes de programación, lo que demuestra que existe un gran interés en este tipo de herramientas de diseño de software, tanto de los desarrolladores como de sus clientes, sin embargo, la mayoría de ellos son implementaciones de lenguajes más antiguos.

Más a pesar de este gran abanico de posibilidades, no existe ningún lenguaje de programación que se destaque por sobre el resto, ya que cada uno de los lenguajes de programación ofrecen ventajas y desventajas, y será cada desarrollador el que deba seleccionar el que mejor le sea conveniente para el tipo de desarrollo que llevará a cabo.

1.3. Paradigmas de Programación

Un paradigma de programación es un estilo de desarrollo de programas. Es decir, un modelo para resolver problemas computacionales. Los lenguajes de programación, necesariamente, se encuadran en uno o varios paradigmas a la vez a partir del tipo de

órdenes que permiten implementar, algo que tiene una relación directa con su sintaxis. Los principales son:

1.3.1. Paradigma imperativo:

Describe la programación como una secuencia instrucciones o comandos que cambian el estado de un programa. El código máquina en general está basado en el paradigma imperativo. Su contrario es el paradigma declarativo. En este paradigma se incluye el paradigma procedimental (procedural) entre otros.

PARADIGMAS	CARACTERÍSTICAS	LENGUAJE EJEMPLO
IMPERATIVO Los lenguajes imperativos son lenguajes controlados por mandatos u orientados a enunciados (instrucciones).	- Comandos o Instrucciones. - Orientados a la utilización por programadores profesionales. - Requiere especificación sobre cómo ejecutar una tarea. - Se deben especificar todas las alternativas. - Requiere gran número de instrucciones de procedimiento. - El código puede ser difícil de leer, entender y mantener. - Lenguaje creado originalmente para operación por lotes. - Puede ser difícil de aprender. - Difícil de depurar. - Orientados comúnmente a archivos.	• Fortran (FORmula TRANslation) • Basic (Beglnner's All-purpose Symbolic Instruction Code) • Cobol (COmmon Business-Oriented Language) • Pascal • Ada • ALGOL • Smalltalk • PL/I • C

Figura 4 Paradigmas
Fuente: Elaboración propia

1.3.2. Paradigma declarativo

No se basa en el cómo se hace algo (cómo se logra un objetivo paso a paso), sino que describe (declara) cómo es algo. En otras palabras, se enfoca en describir las propiedades de la solución buscada, dejando indeterminado el algoritmo (conjunto de instrucciones) usado para encontrar esa solución. Es más complicado de implementar que el paradigma

imperativo, tiene desventajas en la eficiencia, pero ventajas en la solución de determinados problemas.

1.3.3. Paradigma funcional

Este paradigma concibe a la computación como la evaluación de funciones matemáticas y evita declarar y cambiar datos. En otras palabras, hace hincapié en la aplicación de las funciones y composición entre ellas, más que en los cambios de estados y la ejecución secuencial de comandos (como lo hace el paradigma procedimental). Permite resolver ciertos problemas de forma elegante y los lenguajes puramente funcionales evitan los efectos secundarios comunes en otro tipo de programaciones.

FUNCIONAL	
La programación funcional se basa en el uso de las funciones matemáticas para producir mejores efectos y que sus resultados sean más eficientes.	♥ ML(MetaLanguaje) ♥ LISP

Figura 5 Programación funcional

Fuente: Elaboración propia

1.4. Programación Estructurada y Modular

***Programación estructurada*.** La programación estructurada es un paradigma de programación orientado a mejorar la claridad, calidad y tiempo de desarrollo de un programa de computadora, utilizando únicamente subrutinas y tres estructuras: secuencia, selección (if y switch) e iteración (bucles for y while), considerando innecesario y contraproducente el uso de la instrucción de transferencia incondicional (GOTO), que podría conducir a "código espagueti", que es mucho más difícil de seguir y de mantener, y era la causa de muchos errores de programación.

Surgió en la década de 1960, particularmente del trabajo de Böhm y Jacopini, y una famosa carta, (la sentencia goto considerada perjudicial), de Edsger Dijkstra en 19682 y fue reforzado teóricamente por el teorema del programa estructurado, y prácticamente por la aparición de lenguajes como ALGOL con adecuadas y ricas estructuras de control.

Programación modular. La programación modular es un paradigma de programación que consiste en dividir un programa en módulos o subprogramas con el fin de hacerlo más legible y manejable. Se presenta históricamente como una evolución de la programación estructurada para solucionar problemas de programación más grandes y complejos de lo que ésta puede resolver.

Al aplicar la programación modular, un problema complejo debe ser dividido en varios subproblemas más simples, y estos a su vez en otros subproblemas más simples. Esto debe hacerse hasta obtener subproblemas lo suficientemente simples como para poder ser resueltos fácilmente con algún lenguaje de programación. Esta técnica se llama refinamiento sucesivo, divide y vencerás o análisis descendente (Top-Down).

1.5. Programación Visual

Los lenguajes de programación visual son aquellos en donde podemos crear programas, por medio de objetos (iconos, gráficos, etc.), que interactúan unos con otros; que es completamente opuesto a la programación tradicional (programación textual), en la que el programador tenía que seguir ciertas rutinas e instrucciones para el computador.

Un lenguaje de programación visual expresa el diseño de programas con gráficos, colores, etc. Los objetos que hay se desprenden de manera visual, y pasan de un estado a otro, al asignarles ciertas propiedades, y se comportan de cierto modo al asignarles métodos que funcionan se lleva a cabo un evento (interfaz entre el usuario y la maquina).

Entonces lo más importante que nos muestra la programación visual, son las notaciones gráficas, los componentes del Software que son manipulables y usamos las expresiones visuales en el proceso de programación.

En un lenguaje de programación visual lo más importante son los objetos (iconos), los cuales son manipulables e interactúan entre ellos. Un icono es un objeto que tiene dos representaciones, la imagen (que es la parte física) y el significado de la imagen (que es la

parte lógica). Los iconos se encuentran estructurados en un sistema de iconos (caja de herramientas).

Entre los lenguajes de programación visual tenemos: C++, fox., Basic; que usan interfaces gráficas para hacer más amigable la programación

1.6. Programación Orientada a Objetos

La Programación Orientada a Objetos (POO), es una forma de programación que utiliza objetos (similares a los objetos del mundo real) para la solución de problemas. La POO permite descomponer un problema en bloques relacionados. Cada bloque pasa a ser un objeto auto contenido que posee sus propios datos e instrucciones. De esta manera, la complejidad se reduce y se pueden realizar programas más largos de una manera sencilla, mediante la combinación de varios bloques.

1.6.1. Elementos básicos de la programación orientada a objetos

Los elementos básicos de la programación orientada a objetos son:

- Objetos
- Propiedades
- Métodos
- Eventos
- Mensajes
- Clases

a. ***Objetos.*** Un objeto o entidad que tiene atributos propios (propiedades) y unas formas de operar sobre ellos (métodos). Por tanto, un objeto contiene variables que especifican su estado y operaciones que definen su comportamiento. Son ejemplos de objetos en Visual Basic: Formularios, botones de comando, cuadro de texto, etiqueta, etc.

Nombres de objetos. En principio cada objeto de Visual Basic debe tener un nombre, por medio del cual se hace referencia a dicho objeto. El nombre puede ser el que el usuario desee, e incluso Visual Basic proporciona nombres por defecto para los diversos controles. Estos nombres por defecto hacen referencia al tipo de control y van seguidos de un número que se incrementa a medida que se van introduciendo más controles de ese tipo en el formulario (por ejemplo, VScroll1, para una barra de desplazamiento - scroll bar- vertical, HScroll1, para una barra horizontal, etc.). Los nombres por defecto no son adecuados porque hacen referencia al tipo de control, pero no al uso que de dicho control está haciendo el programador.

La tabla muestra las abreviaturas de los controles más usuales, junto con la nomenclatura inglesa de la que derivan. En este mismo capítulo se verán unos cuantos ejemplos de aplicación de estas reglas para construir nombres.

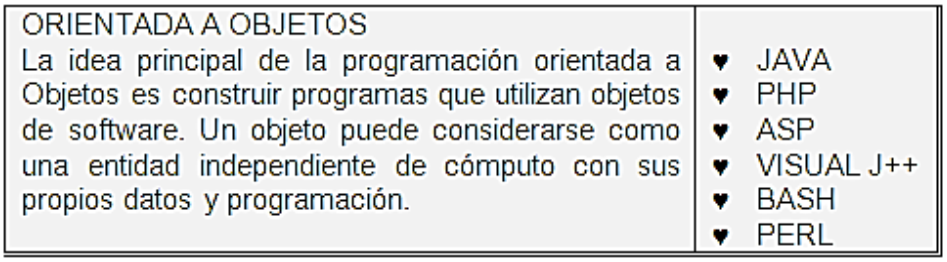


Figura 6 Programación Orientada a Objetos

b. ***Propiedades.*** Las propiedades representan las características del objeto. Hay propiedades particulares, como Caption que las poseen los botones de comando, por ejemplo, y genéricas como Name que la poseen todos los objetos.

c. ***Métodos.*** Los métodos son funciones asociadas a un objeto; pero a diferencia de los procedimientos no son programadas por el usuario; sino que vienen ya programadas con el lenguaje de programación. Cada tipo de objeto tiene sus propios métodos.

d. ***Eventos.*** Un evento es la capacidad de un objeto de reaccionar cuando ocurre una determinada acción (acción y reacción). Como respuesta a un evento se envía un mensaje y se ejecuta un determinado método (procedimiento).

e. ***Mensajes.*** Un mensaje es una llamada a un método, de tal forma que cuando un método recibe un mensaje, la respuesta a ese mensaje es ejecutar el procedimiento asociado. Cuando se ejecuta un programa orientado a objetos, los objetos están constantemente recibiendo, interpretando y respondiendo a mensajes de otros objetos.

f. ***Clases.*** Una clase es una descripción para producir objetos de esa clase o tipo. Es decir, se trata de una generalización de un tipo específico de objetos. En otras palabras, un objeto es una variable del tipo definido por una clase. Por ejemplo, si se pensara en hacer un molde para pasteles, el molde es la clase y los pasteles los objetos.

1.6.2. Características de la programación orientada a objetos

Las características fundamentales de la programación orientada a objetos son:

- Abstracción
- Encapsulamiento
- Herencia
- Polimorfismo

a. ***Abstracción.*** La abstracción permite no detenerse en los detalles concretos del funcionamiento de las cosas, sino centrarse en los aspectos que realmente nos importan y son útiles en un determinado momento. En cierta medida, se podría decir que es "úsese el objeto y olvídese de cómo funciona en forma interna". Por ejemplo, para manejar una computadora no se necesita saber cómo funcionan sus circuitos electrónicos, en términos de corriente, tensión, etc.

b. ***Encapsulamiento.*** Esta característica permite ver un objeto como una "caja negra" auto contenida en la que se ha metido de alguna manera toda la información que

maneja dicho objeto. Esto permite manipular los objetos como unidades básicas, permaneciendo oculta su estructura interna.

c. ***Herencia.*** La herencia es la característica que permite compartir automáticamente propiedades y métodos entre objetos. Es decir, se pueden crear nuevas clases de objetos en base a clases existentes. Más concreto, un objeto puede heredar un conjunto general de propiedades y métodos a las que puede añadir aquellas características que son específicas suyas. El usuario de Visual Basic no dispone de esta característica.

d. ***Polimorfismo.*** Polimorfismo, cuyo significado es "muchas formas", es la característica que permite implementar múltiples formas de un mismo método, dependiendo cada una de ellas de la clase sobre la que se realiza la implementación. Esto hace posible que se puede acceder a una variedad de métodos distintos (todos con el mismo nombre) utilizando exactamente el mismo medio de acceso.

1.7. Visual .Net

Es un lenguaje de programación desarrollado por el alemán Alan Cooper para Microsoft. El lenguaje de programación es un dialecto de BASIC, con importantes agregados. Su primera versión fue presentada en 1991, con la intención de simplificar la programación utilizando un ambiente de desarrollo completamente gráfico que facilitara la creación de interfaces gráficas y, en cierta medida, también la Programación misma. Desde el 2001 Microsoft ha propuesto abandonar el desarrollo basado en la API Win32 y pasar a trabajar sobre un framework o marco común de librerías independiente de la versión del sistema operativo, NET Framework, a través de Visual Basic, NET (y otros lenguajes como C Sharp (C#) de fácil transición de código entre ellos).

El Visual.Net es un lenguaje de programación que actualmente respeta prácticamente todas las características de la P. O. O., anteriormente la generación de un programa ejecutable con Basic generaba un ejecutable, actualmente genera un archivo

intermedio que se ejecuta posteriormente por la máquina virtual de Studio Net, esta característica, que hoy incorporan otros lenguajes de programación, convierte a VB en un lenguaje portable en el momento que exista una máquina virtual para otros sistema operativos distintos de Windows.

VB.NET, al estar basado en el .NET Framework, se conocerá este marco de desarrollo tan igual que otros lenguajes .NET, tales como C#, ya que, al fin y al cabo, el corazón de los lenguajes .NET es el .NET Framework. Para ir aclarando ideas, veamos algunos conceptos que habrá que tener claros desde el principio: Visual Basic .NET usa una jerarquía de clases que están incluidas en el .NET Framework, por tanto, conocer el .NET Framework nos ayudará a conocer al propio Visual Basic .NET, aunque también necesitarás conocer la forma de usar y de hacer del VB ya que, aunque en el fondo sea lo mismo, el aspecto sintáctico es diferente para cada uno de los lenguajes basados en .NET Framework.

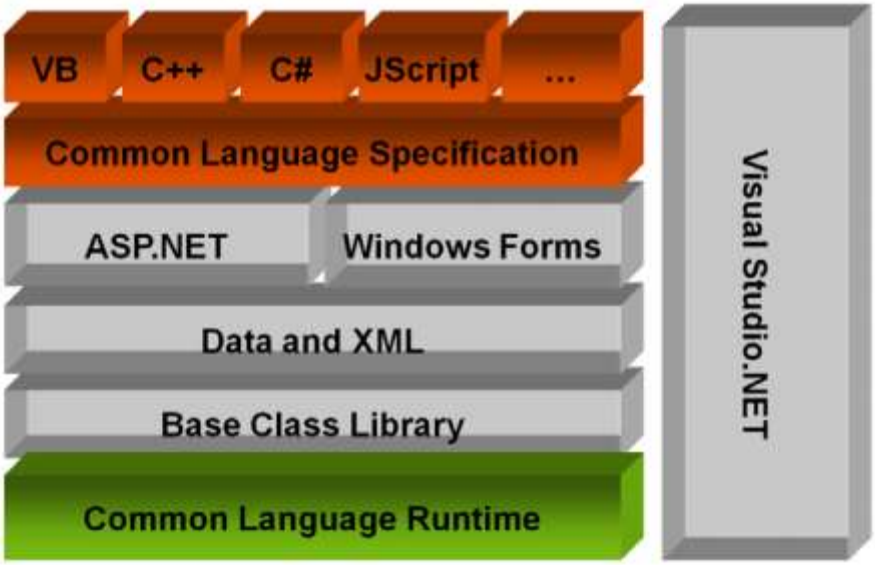


Figura 7 Visual Studio. NET

Fuente: Elaboración propia

Los principales beneficios de utilizar la tecnología .NET son la utilización de una plataforma de desarrollo de programación orientada a objetos, una gestión automática de la memoria y la integración en una herramienta de diferentes lenguajes de programación.

A pesar de la gran cantidad de usuarios que utilizan Autocad sólo una minoría programa sus propias aplicaciones. Esto hace que haya una carencia de libros y publicaciones sobre VB.NET y Autocad. Además, las aplicaciones más interesantes para el Sistema de Información Monumental estaban basadas en comandos específicos de Autodesk Map por lo que buscar bibliografía específica ha sido tarea casi imposible.

No obstante, la red es una comunidad de usuarios formidable y hay grupos de discusión o blogs en los que se pueden consultar dudas.

Capítulo II: Fundamentos de Visual.Net

2.1. Introducción Microsoft.Net

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

2.1.1. Características de .Net I

Según Microsoft.NET es una plataforma Microsoft para Servicios Web XML.

Sin embargo, es mucho más:

- .NET es una nueva plataforma para el desarrollo y explotación de aplicaciones "gestionadas" (managed) modernas y orientadas a objetos.
- Las aplicaciones .NET se pueden desarrollar en cualquier lenguaje de programación que se ajusta a .NET
- .NET soporta una extensa framework de librerías de clases independientes del lenguaje de programación.
- .NET soporta la creación de componentes auto-describibles
- .NET ofrece integración multi-lenguaje, reutilización de componentes, y herencia entre componentes desarrollados en diferentes lenguajes.

2.1.2. Características de .Net II

- .NET ofrece una nueva manera de desarrollar aplicaciones de sobremesa usando las clases Windows Forms.
- .NET ofrece una nueva manera de desarrollar aplicaciones basadas en navegador Web a través de ASP. NET.

- Las clases ADO.NET proveen una arquitectura desconectada para acceso a datos a través de Internet .
- .NET soporta la creación de Servicios Web XML independientes de la plataforma, a través de SOAP (Simple Object Access Protocol) y WSDL (Web Services Description Language) .
- .NET ofrece una nueva arquitectura para el desarrollo y explotación de objetos remotos .NET convierte a varias tecnologías y técnicas Windows en obsoletas.

2.1.3. Componentes del .Net

Microsoft.NET está compuesto de:

2.1.3.1. Plataforma .NET

- El concepto en el que se basa .NET no es nuevo.
- Java y su entorno de ejecución (JVM) ya utilizan el concepto de encapsulamiento del sistema operativo para permitir la interoperabilidad entre diferentes sistemas operativos.

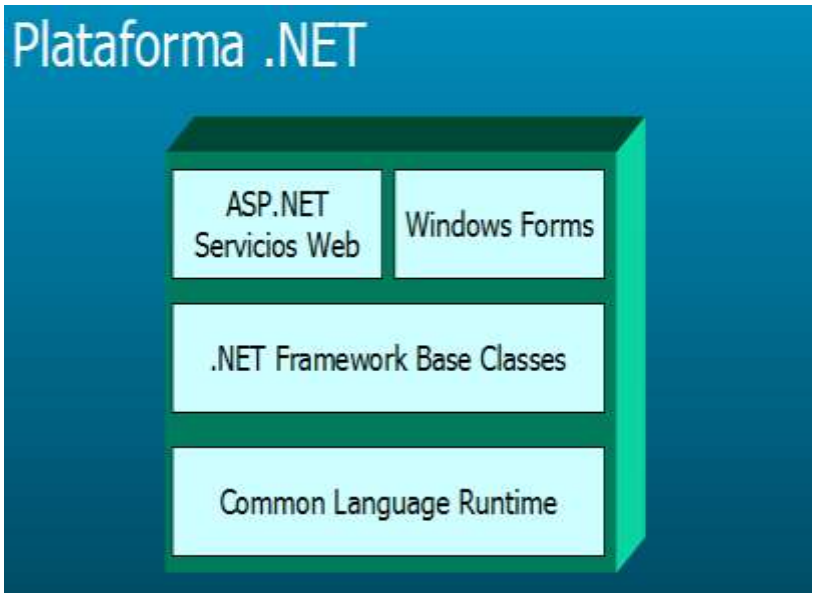


Figura 8 Plataforma .NET

Fuente: Elaboración propia

2.1.3.2. Componentes:

- ***Common Language Runtime*(CLR).** Entorno de ejecución de la plataforma. Common Language Runtime Un Runtime no es más que un entorno en el que se ejecutan los programas. De esta forma, el CLR es el entorno donde se ejecutarán las aplicaciones .NET que han sido compiladas a un lenguaje común llamado Microsoft Intermediate Language (MSIL).

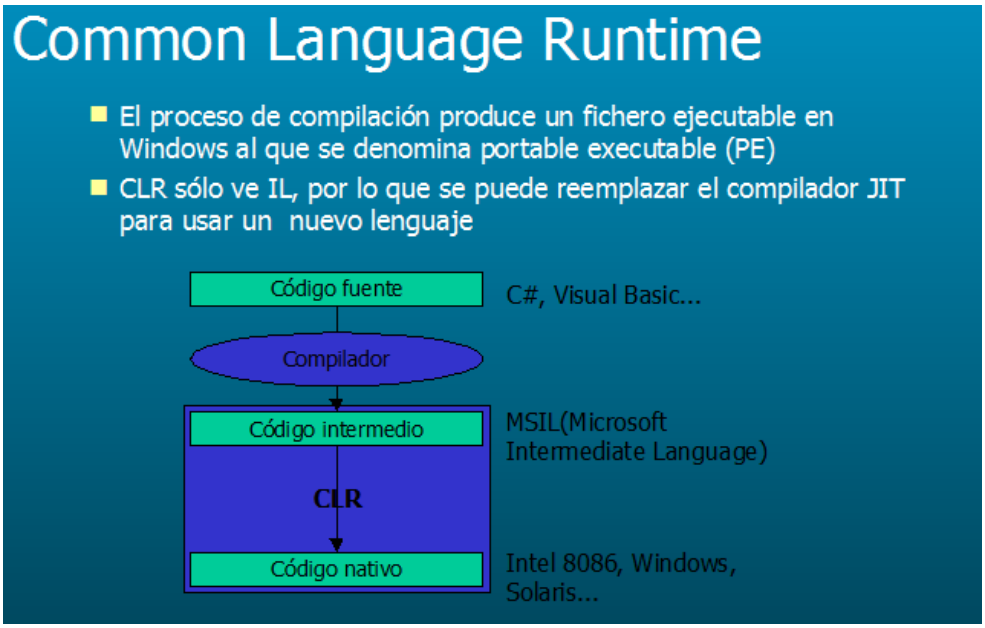


Figura 9 Entorno de ejecución de la plataforma

Fuente: Elaboración propia

- ***.NET Framework Base Classes o FCL* (Framework Class Library).** Añaden funcionalidad.
- ***ASP.NET.*** Versión .Net de ASP. Incluye los servicios Web.

  (Es la versión para la plataforma .NET de la tecnología ASP (Active Server Pages).

  Dentro de esta capa podemos distinguir tres partes muy diferenciadas:

  - Web Forms
  - Server Controls

- Web Services

- ***Windows Forms***
  - Los Windows Forms son una forma avanzada e integrada de crear aplicaciones de escritorio Win32 estándar.
  - Los WinForms descienden de las Windows Foundation Classes (WFC) de Microsoft.
  - Cualquier lenguaje de la plataforma .Net puede utilizar WinForms.
  - De hecho, actualmente, las WinForms forman parte de las clases de la plataforma .NET en el espacio de nombres System.Windows.Forms.

2.1.3.3. .NET Framework SDK

Microsoft distribuye este kit de desarrollo como parte del paquete .NET. Disponible en: http://msdn.microsoft.com/netframework/technologyinfo/howtoget/default.aspx

En este kit podemos encontrar documentación sobre la plataforma, ejemplos y código fuente, y una serie de utilidades que sirven para desarrollo y prueba de aplicaciones .NET.

- NET Framework

.NET Framework es un entorno para construir, instalar y ejecutar servicios Web y otras aplicaciones. Se compone de tres partes principales:

- Common Language Runtime.
- Clases Framework.
- ASP.NET.

El .NET Framework es el corazón de .NET, cualquier cosa que queramos hacer en cualquier lenguaje .NET debe pasar por el filtro cualquiera de las partes integrantes del .NET Framework.

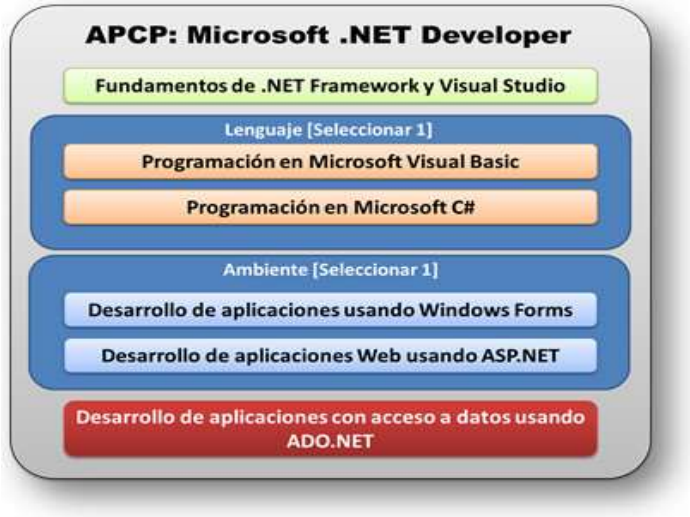


Figura 10 Microsoft .NET

Fuente: Elaboración propia

***El Common Lenguage Runtime* (CLR).** Es una serie de librerías dinámicas (DLLs), también llamadas assemblies, que hacen las veces de las DLLs del API de Windows, así como las librerías runtime de Visual Basic o C++. Cualquier ejecutable depende de una forma u otra de una serie de librerías, ya sea en tiempo de ejecución como a la hora de la compilación. Pues el CLR es eso, una serie de librerías usadas en tiempo de ejecución para que nuestros ejecutables o cualquiera basado en .NET puedan funcionar. Se acabó eso de que existan dos tipos de ejecutables:

- Los que son autosuficientes y no dependen de librerías externas
- Los que necesitan de librerías en tiempo de ejecución para poder funcionar, tal es el caso de las versiones anteriores de Visual Basic.

Librería de clases de .NET Framework. Proporciona una jerarquía de clases orientadas a objeto disponibles para cualquiera de los lenguajes basados en .NET, incluido el Visual Basic. Esto quiere decir que a partir de ahora Visual Basic tendrá a su disposición todas las clases disponibles para el resto de los lenguajes basados en .NET, (o casi), con lo cual sólo nos diferenciará del resto de programadores en la forma de hacer las cosas más fáciles.

- ***Características de Visual.Net***

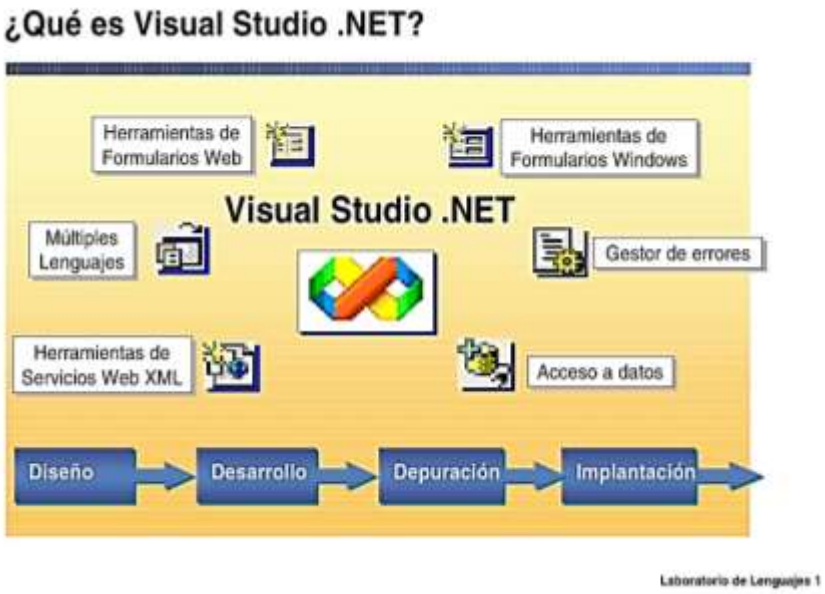


Figura 11 Visual Studio .NET

Fuente: Elaboración propia

2.1.3.4. Visual Studio.Net

Es un entorno gráfico que permite a los desarrolladores crear, probar y depurar aplicaciones desarrolladas o no para la plataforma .NET.

Inicialmente se llamó Visual Studio 7.0 ya que es la evolución del Visual Studio 6.0.

La última versión a 02/2004, es el Visual Studio 2003.

SharpDevelop es un IDE open source para .NET que representa una buena alternativa a Visual Studio.NET: http://www.icsharpcode.net/opensource/sd/

2.1.3.5. Servicios Web (Microsoft.Net MyServices)

.NET My Services Microsoft ha creado una serie de servicios Web a los que llama ".NET My Services" o "HailStorm".

.Net My Services son un conjunto de servicios Web XML que permiten al usuario almacenar y recuperar información confidencial (contactos, agenda, documentos...).

http://msdn.microsoft.com/myservices

2.1.3.6. Servidores para empresas (SQL Server.NET.)

La evolución de.NET ha obligado a Microsoft a realizar cambios en los servidores para empresas (Enterprise Servers) que distribuye.

Los nuevos servidores, ".NET Enterprise Servers" tratan de ayudar a las empresas a integrar y orquestar servicios y aplicaciones en una sola solución comprensible.

Algunos de estos servidores son: Application Center, BizTalk, Commerce Server, Exchange...

Extensión de los ficheros de código.

En Visual Basic .NET a diferencia de lo que ocurría en las versiones anteriores de Visual Basic, sólo existe un tipo de fichero de código, el cual tiene la extensión .vb, en este tipo de fichero pueden coexistir distintos tipos de elementos, por ejemplo: un módulo de clase, un formulario, un módulo de código, un control, etc.; mientras que, en las versiones anteriores de Visual Basic, cada uno de estos elementos tenían su propio tipo de fichero con su respectiva extensión.

Tipos de ejecutables.

Con Visual Basic .NET puedes crear básicamente estos dos tipos de ejecutables: de consola, no gráfico y gráficos, como los que normalmente estamos acostumbrados a ver en Windows. Existen otros tipos de aplicaciones que se pueden crear con Visual Basic .NET: aplicaciones ASP.NET, (realmente no es una aplicación o ejecutable, sino un compendio de distintos tipos de elementos...), servicios Web, servicios Windows, etc.

2.2. Estructura y Elementos del Visual.Net

2.2.1. Visual Studio 6.0

Se lanzó en 1998 y fue la última versión en ejecutarse en la plataforma Win9x. Los números de versión de todas las partes constituyentes pasaron a 6.0, incluyendo Visual J++ y Visual InterDev, que se encontraban en las versiones 1.1 y 1.0 respectivamente. Esta

versión fue la base para el sistema de desarrollo de Microsoft para los siguientes cuatro años, en los que Microsoft migró su estrategia de desarrollo al .NET Framework.

Visual Studio 6.0 fue la última versión en que Visual Basic se incluía de la forma en que se conocía hasta entonces; versiones posteriores incorporarían una versión muy diferente del lenguaje con muchas mejoras, fruto de la plataforma .NET. También supuso la última versión en incluir Visual J++, que proporcionaba extensiones de la plataforma Java, lo que lo hacía incompatible con la versión de Sun Microsystems. Esto acarreó problemas legales a Microsoft, y se llegó a un acuerdo en el que Microsoft dejaba de comercializar herramientas de programación que utilizaran la máquina virtual Java.

Aunque el objetivo a largo plazo de Microsoft era unificar todas las herramientas en un único entorno, esta versión en realidad añadía un entorno más a Visual Studio 5.0: Visual J++ y Visual Interdev se separaban del entorno de Visual C++, al tiempo que Visual FoxPro y Visual Basic seguían manteniendo su entorno específico.

2.2.2. Visual Studio.Net 2002

En esta versión se produjo un cambio sustancial, puesto que supuso la introducción de la plataforma .NET de Microsoft. .NET es una plataforma de ejecución intermedia multilenguaje, de forma que los programas desarrollados en .NET no se compilan en lenguaje máquina, sino en un lenguaje intermedio (CIL - Common Intermediate Language) denominado Microsoft Intermediate Language (MSIL). En una aplicación MSIL, el código no se convierte a lenguaje máquina hasta que se ejecuta, de manera que el código puede ser independiente de la plataforma (al menos de las soportadas actualmente por .NET). Las plataformas han de tener una implementación de Infraestructura de Lenguaje Común(CLI) para poder ejecutar programas MSIL. Actualmente se pueden ejecutar programas MSIL en Linux y Mac OS X usando implementaciones de .NET que no son de Microsoft, tales como Mono y DotGNU.

Visual Studio .NET 2002 supuso también la introducción del lenguaje C#, un lenguaje nuevo diseñado específicamente para la plataforma .NET, basado en C++ y Java. Se presentó también el lenguaje J# (sucesor de J++), el cual, en lugar de ejecutarse en una máquina virtual Java, se ejecuta únicamente en el framework .NET. El lenguaje Visual Basic fue remodelado completamente y evolucionó para adaptarse a las nuevas características de la plataforma .NET, haciéndolo mucho más versátil y dotándolo con muchas características de las que carecía. Algo similar se llevó a cabo con C++, añadiendo extensiones al lenguaje llamadas Managed Extensions for C++ con el fin de que los programadores pudieran crear programas en .NET. Por otra parte, Visual FoxPro pasa a comercializarse por separado.

Todos los lenguajes se unifican en un único entorno. La interfaz se mejora notablemente en esta versión, siendo más limpia y personalizable.

Visual Studio .NET puede usarse para crear programas basados en Windows (usando Windows Forms en vez de COM), aplicaciones y sitios web (ASP.NET y servicios web), y dispositivos móviles (usando el .NET Compact Framework).

Esta versión requiere un sistema operativo basado en NT. La versión interna de Visual Studio .NET es la 7.0.

2.2.3. Visual Studio.Net 2003.

Visual Studio .NET 2003 supone una actualización *menor* de Visual Studio .NET. Se actualiza el .NET Framework a la versión 1.1. También se añade soporte con el fin de escribir aplicaciones para determinados dispositivos móviles, ya sea con ASP.NET o con el .NET Compact Framework. Además, el compilador de Visual C++ se mejora para cumplir con más estándares: el Visual C++ Toolkit 2003.

Visual Studio 2003 se lanza en cuatro ediciones: Academic, Professional, Enterprise Developer y Enterprise Architect. La edición Enterprise Architect incluía una

implementación de la tecnología de modelado Microsoft Visio, que se centraba en la creación de representaciones visuales de la arquitectura de la aplicación basadas en UML. También se introdujo "Enterprise Templates", para ayudar a grandes equipos de trabajo a estandarizar estilos de programación e impulsar políticas de uso de componentes y asignación de propiedades.

Microsoft lanzó el *Service Pack 1* para Visual Studio 2003 el 13 de septiembre de 2006.

La versión interna de Visual Studio .NET 2003 es la 7.1, aunque el formato del archivo que emplea es el de la 8.0.

Es compatible solo con Windows XP, Windows Server 2003 o anteriores

2.2.4. Visual Studio.Net 2005

Visual Studio 2005 se empezó a comercializar a través de Internet a partir del 4 de octubre de 2005, y la versión en inglés llegó a los comercios a finales del mes de octubre. En castellano no salió hasta el 4 de febrero de 2006. Microsoft eliminó la coletilla *.NET* de su nombre, pero eso no indica que se alejara de la plataforma .NET, de la cual se incluyó la versión 2.0.

La actualización más importante que recibieron los lenguajes de programación fue la inclusión de tipos genéricos, similares en muchos aspectos a las plantillas de C++. Con esto se consigue encontrar muchos más errores en la compilación en vez de en tiempo de ejecución, incitando a usar comprobaciones estrictas en áreas donde antes no era posible. C++ tiene una actualización similar con la adición de C++/CLI como sustituto de C# manejado.

Se incluye un diseñador de implantación, que permite que el diseño de la aplicación sea validado antes de su implantación. También se incluye un entorno para publicación web y pruebas de carga para comprobar el rendimiento de los programas bajo varias condiciones de carga.

Visual Studio 2005 también añade soporte para arquitecturas de 64 bits. Aunque el entorno de desarrollo sigue siendo una aplicación de 32 bits, Visual C++ 2005 soporta compilación para x86-64 (AMD64, Intel 64) e IA-64 (Itanium). El SDK incluye compiladores de 64 bits, así como versiones de 64 bits de las librerías.

Visual Studio 2005 tiene varias ediciones radicalmente distintas entre sí: Express, Standard, Professional, Tools for Office y cinco ediciones Visual Studio Team System. Estas últimas se proporcionaban conjuntamente con suscripciones a MSDN cubriendo los cuatro principales roles de la programación: Architects, Software Developers, Testers y Database Professionals. La funcionalidad combinada de las cuatro ediciones Team System se ofrecía como la edición Team Suite. Por otra parte, Tools for the Microsoft Office System está diseñada para extender la funcionalidad a Microsoft Office.

Las ediciones Express se han diseñado para principiantes, aficionados y pequeños negocios, todas disponibles gratuitamente a través de la página de Microsoft.[4] Se incluye una edición independiente para cada lenguaje: Visual Basic, Visual C++, Visual C#, Visual J# para programación .NET en Windows y Visual Web Developer para la creación de sitios web ASP.NET. Las ediciones Express carecen de algunas herramientas avanzadas de programación, así como de opciones de extensibilidad.

Se lanzó el Service Pack 1 para Visual Studio 2005 el 14 de diciembre de 2006.

La versión interna de Visual Studio 2005 es la 8.0, mientras que el formato del archivo que emplea es el de la 9.0.

2.2.5. Visual Studio.Net 2008

Permite trabajar con los Frameworks:

- .NET Framework 2.0
- .NET Framework 3.0
- .NET Framework 3.5

2.2.6. Visual Studio.Net 2010

El IDE se rediseña para una mejor legibilidad. Se han eliminado gradientes y líneas innecesarias para hacer más simple su uso.

Las ventanas de documentos tales como el editor de código y la ventana de la vista diseño ahora pueden colocarse fuera de la ventana IDE. Por ejemplo, puede arrastrar el editor de código en el IDE de modo que se puede ver la ventana de la vista de diseño al lado.

Permite trabajar con los Frameworks:

- .NET Framework 2.0
- .NET Framework 3.0
- .NET Framework 3.5
- .NET Framework 4.0

2.2.7. Visual Studio.Net 2012

- NET Framework 2.0
- .NET Framework 3.0
- .NET Framework 3.5
- .NET Framework 4.0
- .NET Framework 4.5

2.2.8. Visual Studio.Net 2013

Fue la primera revisión de Visual Studio en incluir una versión "Community", que básicamente ofrece las mismas capacidades que la versión "Professional" pero limitando su uso a empresas de pequeño tamaño, desarrolladores de software libre y estudiantes. La gran ventaja de esta versión de Visual Studio es que es gratuita.

Permite trabajar con los frameworks:

- .NET Framework 2.0
- .NET Framework 3.0
- .NET Framework 3.5
- .NET Framework 4.0
- .NET Framework 4.5
- .NET Framework 4.5.1
- .NET Framework 4.5.2

2.2.9. Visual Studio.Net 2015

Permite trabajar con los frameworks:

- .NET Framework 2.0
- .NET Framework 3.0
- .NET Framework 3.5
- .NET Framework 4.0
- .NET Framework 4.5
- .NET Framework 4.5.1
- .NET Framework 4.5.2
- .NET Framework 4.6
- .NET Framework 4.6.1

2.2.10. Visual Studio.Net 2017

Permite trabajar con los frameworks:

- .NET Framework 2.0
- .NET Framework 3.0
- .NET Framework 3.5
- .NET Framework 4.0

- .NET Framework 4.5
- .NET Framework 4.5.1
- .NET Framework 4.5.2
- .NET Framework 4.6
- .NET Framework 4.6.1
- .NET Framework 4.7
- .NET Framework 4.7.1
- .NET Framework 4.7.2

La compatibilidad es la misma que en Visual Studio 2015, salvo que se agregan algunas funciones extra.

2.3. Instalación de Visual Studio.Net

2.3.1. Preparación del entorno de trabajo

Antes de poder comenzar a escribir aplicaciones para .NET Framework, debemos instalar en nuestra máquina de trabajo las herramientas que nos permitirán el desarrollo de programas para este entorno de ejecución.

2.3.2. .NET Framework SDK

Se trata del kit de desarrollo de software para .NET Framework (Software Development Kit o SDK), que contiene la propia plataforma .NET y un conjunto de herramientas independientes, algunas funcionan en modo comando (en una ventana MS-DOS) y otras en modo gráfico. Los elementos imprescindibles para poder desarrollar aplicaciones para .NET están contenidos en este conjunto de herramientas.

2.3.3. Visual Studio .NET

Es la nueva versión de la familia de herramientas de desarrollo de software de Microsoft, naturalmente orientadas hacia su nuevo entorno de programación: .NET Framework. Si bien es posible la escritura de programas empleando sólo el SDK de .NET

Framework, este último, al estar compuesto de herramientas independientes, constituye un medio más incómodo de trabajo.

Visual Studio .NET (VS.NET a partir de ahora), al tratarse de un entorno de desarrollo integrado (Integrated Development Environment o IDE como también lo denominaremos a lo largo del texto), aúna todas las herramientas del SDK: compiladores, editores, ayuda, etc., facilitando en gran medida la creación de programas. Por este motivo, todas las explicaciones y ejemplos desarrollados a lo largo de este texto se harán basándose en este entorno de programación.

2.3.4. Requisitos básicos

Requerimiento de la PC

- Procesador de 1,6 GHz o superior
- 1 GB de RAM (1,5GB si es ejecutable en máquina virtual)
- 10 GB de espacio disponible en disco duro
- Unidad de disco duro de 5400 rpm
- Tarjeta de video compatible con Directx9 con una resolución de 1024x768 o superior
- Window 8.1 y probado también en Windows 10

Tiempo estimado para completar este laboratorio. 60 minutos

2.3.5. Instalar Visual Studio

2.3.5.1. Pasos para la instalación

Primera forma

- **Paso 1**

Luego de haber descargado el archivo .iso en el cual se encuentra el programa listo para ser instalado se ejecuta con un programa que maneje este tipo de archivos para poder abrir

y poder instalar el programa, ejemplo de estos programas son Daemontools o UltraIso, en este caso usaremos Daimontools

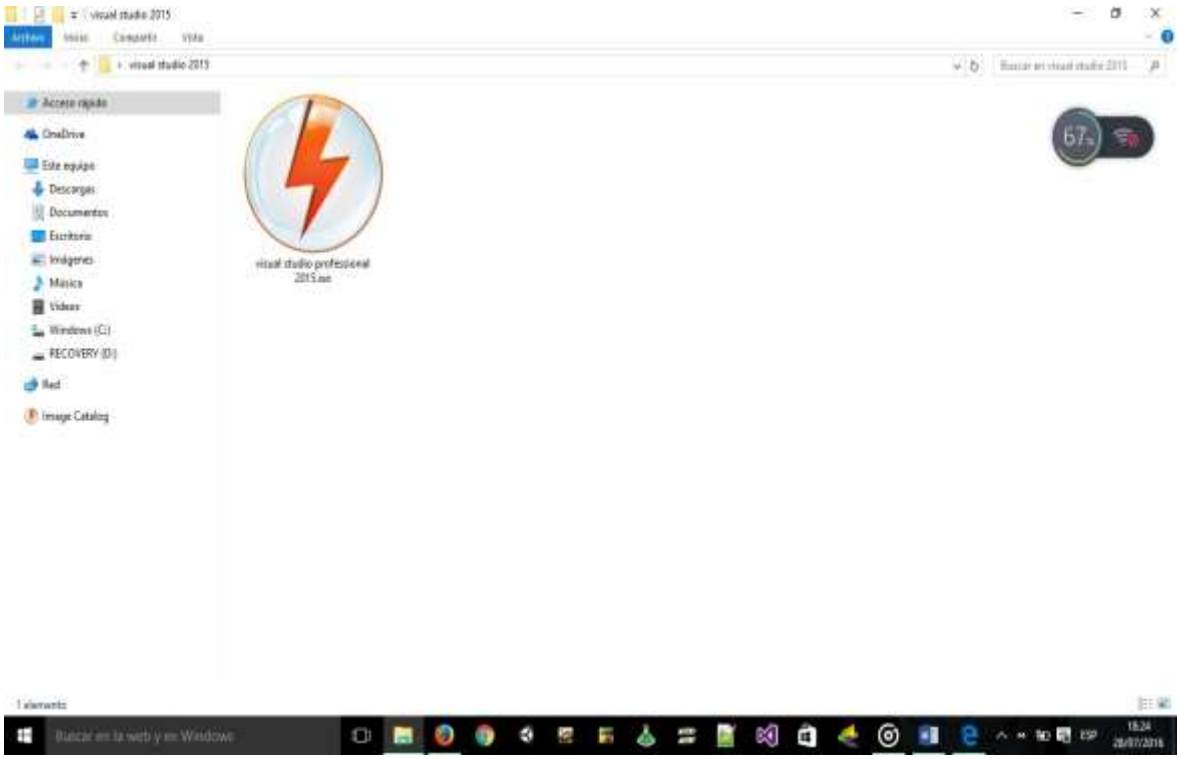


Figura 12 Ventana icono daimontools 1

- **Paso 2**

Se ejecuta el archivo .iso con Daimontools para ver su contenido y se crea un dispositivo de almacenamiento virtual o UltraIso. En este caso usaremos Daimontools.

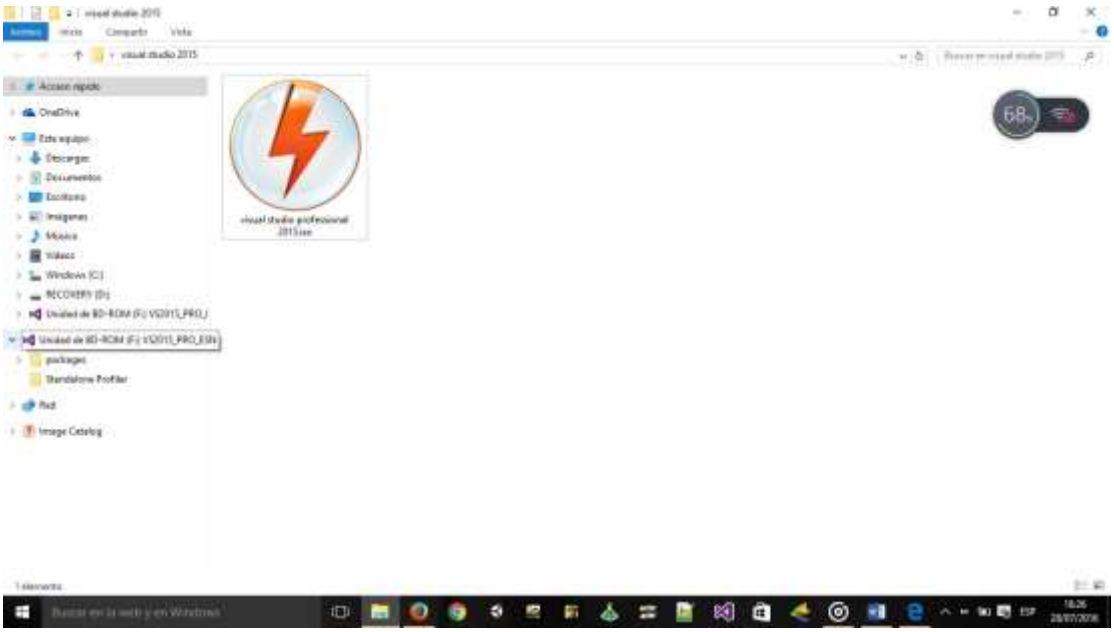


Figura 13 Ventana Icono daimontools 2

- **Paso 3**

Una vez veamos el contenido de dicho .iso veremos el instalador el cual tendremos que ejecutar en el paso siguiente.

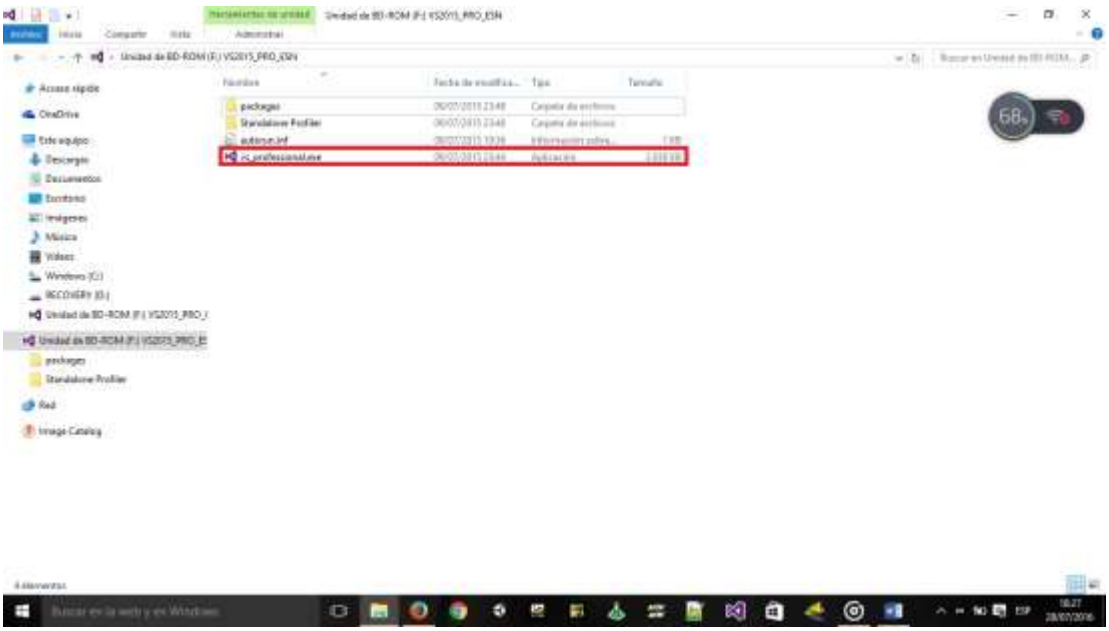


Figura 14 Icono de instalador

- **Paso 4**

Una vez ejecutado el instalador aparecerá la siguiente pestaña de información en el cual nos da opciones de instalación que tenemos que escoger una de las dos.

Típica.

Incluye las características básicas de Visual Studio con las cuales podemos empezar hacer uso adecuado del mismo, además eso es la recomendada para las personas que empiezan en Visual Studio.

Personalizada.

Es la opción para usuarios más experimentados en la cual pueden personalizar a su gusto los componentes que deseen instalar.

Nota: La instalación personalizada se usa comúnmente con los programadores expertos que saben el uso de cada componente por eso no es recomendable con principiantes ya que, además de lograr que la instalación sea mucho más tediosa y que dure mucho más puede incluso a llegar a ser fastidiosa por el motivo de confusión al buscar entre tantos componentes.

La pestaña de componentes al usar la instalación personalizada (están marcadas directamente los componentes básicos para Visual Studio).

Figura 15 Ventana opciones

Figura 16 Ventana opción típica

En esta ocasión usaremos la instalación predeterminada no solo para ahorrar tiempo de instalación sino también para lograr un mayor entendimiento del uso del programa y a medida conozcamos más el programa podremos instalar más componentes.

Figura 17 Ventana opciones predeterminada

Pero antes de dar clic al botón de instalar debemos de saber que son los componentes que estamos instalando.

-La instalación típica incluye la posibilidad de usar herramientas web que puedes ser programables en muchos lenguajes de programación además de un paquete de lenguajes predeterminado del programa.

-Algunos lenguajes compatibles con Visual Studio son: C#, C++, JavaScript, TypeScript, Python y otros.

-Además de que tiene la función de acoplarse a cada lenguaje y así puede guiar al programador en su escritura de código detectando errores.

-Además de ser personalizable descargando paquetes para abrir aún más funciones (aunque esto se puede hacer directamente en la instalación personalizada)

- **Paso 5**

Llegado a este punto es hora de dar clic en el botón instalar, no sin antes haber revisado la ruta en la cual el usuario desea que termine el programa

Figura 18 Ventana opción instalar

- **Paso 6**

Luego de ver y comprobar si existen errores damos clic en el apartado que dice iniciar cargar a otra pestaña de visual studio en la que analizará los componentes para poder lograr correr el programa adecuadamente.

Figura 19 Ventana Visual Instalación

Bueno y si es la primera vez que lo instalan aparecerá la pestaña siguiente en la cual si ustedes tienen cuenta de visual studio pueden iniciar y si no tienen pueden omitirla dando clic en "de momento no"

Figura 20 Ventana Visual 1

- **Paso 7(opcional)**

Si posees un código de Visual Studio lo podrás disfrutar sin restricciones haciendo la siguiente guía de pasos sencillos.

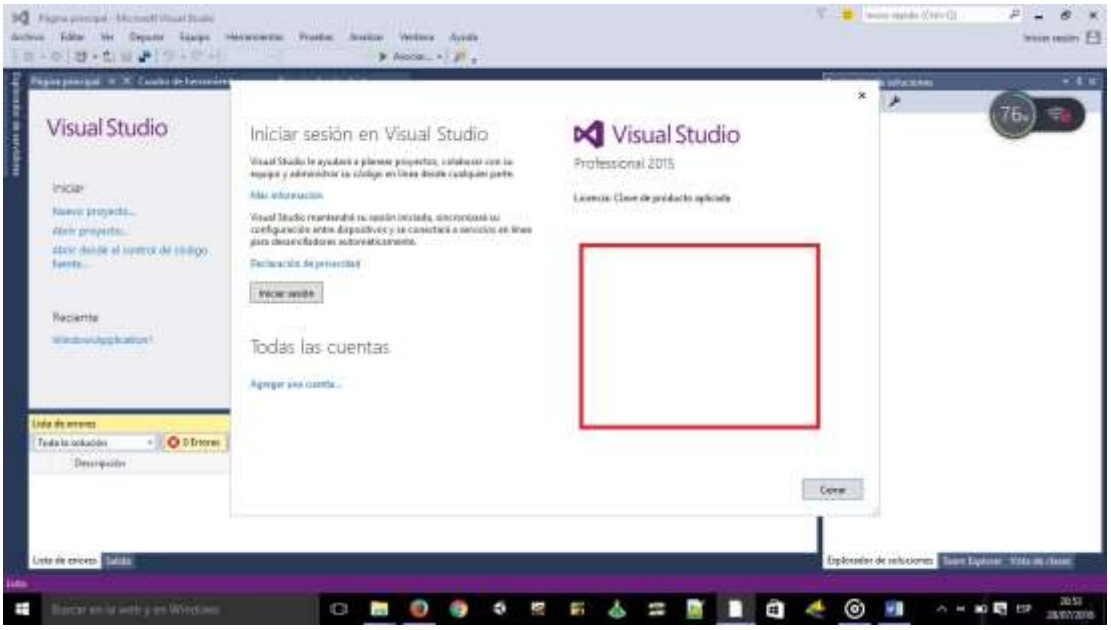


Figura 21 Ventana Visual 2

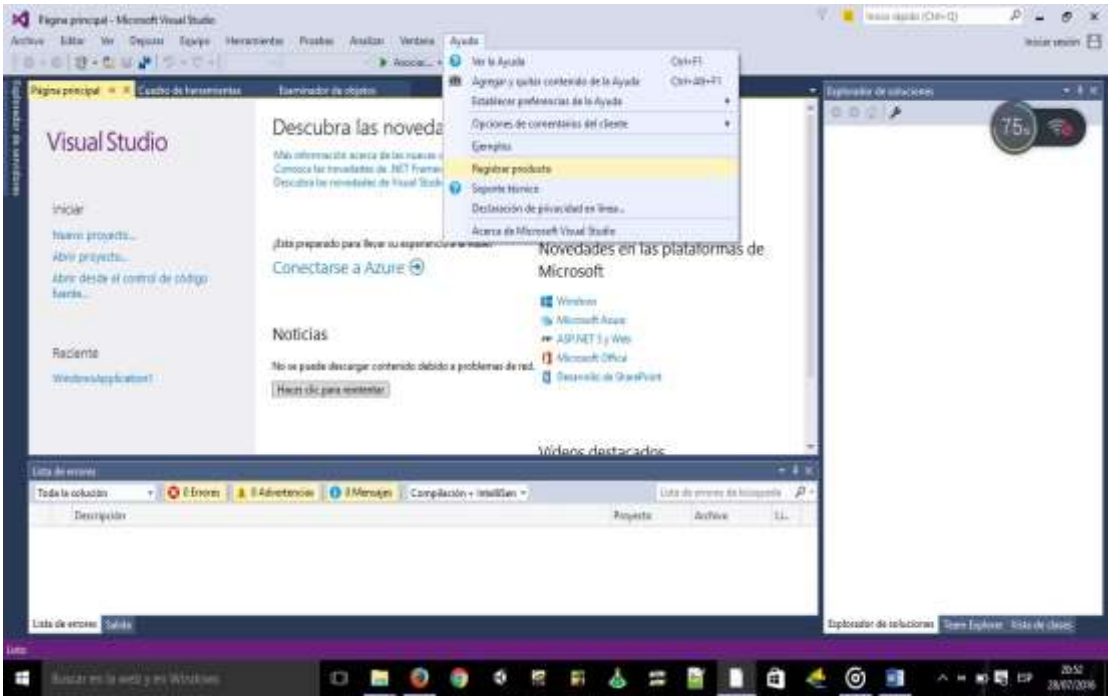


Figura 22 Ventana Visual 3

Y así se le da fin a la de instalación de Visual Studio.

2.3.6. 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:

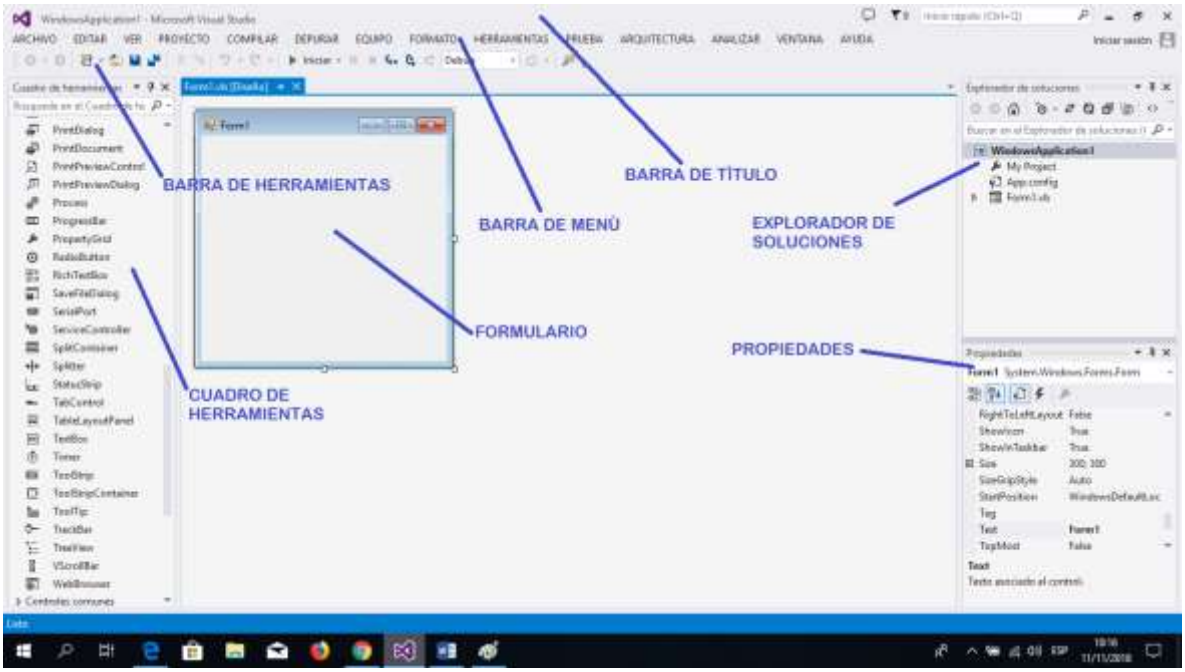


Figura 23 Página de inicio de Visual

2.3.7. 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.

2.3.8. 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.

2.4. Visual Basic.Net

2.4.1. Introducción

Visual Basic es un ambiente gráfico de desarrollo de aplicaciones para el sistema operativo Microsoft Windows. Las aplicaciones creadas con Visual Basic están basadas en objetos y son manejadas por eventos. Visual Basic se deriva del lenguaje Basic, el

cual es un lenguaje de programación estructurado. Sin embargo, Visual Basic emplea un modelo de programación manejada por eventos.

En una aplicación Visual Basic, el programa está formado por una parte de código puro y otras partes asociadas a los objetos que forman la interface gráfica. Es por tanto un término medio entre la programación tradicional, formada por una sucesión lineal de código estructurado, y la programación orientada a objetos. Cambian ambas tendencias, sin embargo, no puede decirse que visual Basic pertenezca por completo a uno de estos dos tipos de programación

2.4.2. La evolución hacia. NET

Los motivos que han llevado a Microsoft al desarrollo de .NET han sido tanto tecnológicos como estratégicos. Respecto a las motivaciones tecnológicas, la necesidad de poner a disposición del programador una plataforma de desarrollo con plena potencia para abarcar los requerimientos de las nuevas aplicaciones que están a punto de llegar, y que no soporte incómodos las tres derivados de antiguos modelos de programación, ha desembocado en una tecnología totalmente nueva, que no arrastra pesadas incompatibilidades, pero que sin embargo, permite la ejecución de componentes basados en el anterior modelo de programación.

Esto es .NET, una nueva arquitectura para el futuro del desarrollo de aplicaciones, y no, como en un principio pudiera pensarse, una operación más de marketing, que proporciona las herramientas ya conocidas con algunas remodelaciones y lavados de cara.

2.4.3. Características de VB.NET

VB.NET aporta un buen número de características que muchos programadores de VB han demandado desde hace largo tiempo. En cierto modo, alguna de estas incorporaciones se la agradece a la plataforma .NET, ya que, al integrar VB dentro del

conjunto de lenguajes de .NET Framework, dichos cambios han sido necesarios, no ya porque los necesitara VB, sino porque eran requisitos derivados de la propia arquitectura de .NET.

Entre las novedades aportadas por VB.NET existen plenas capacidades de orientación a objetos (Full-OOP), incluyendo por fin, herencia; Windows Forms o la nueva generación de formularios para aplicaciones Windows; soporte nativo de XML; gestión de errores estructurada; un modelo de objetos para acceso a datos más potente con ADO.NET; posibilidad de crear aplicaciones de consola (ventana MS-DOS); programación para Internet mediante Web Forms; un entorno de desarrollo común a todas las herramientas de .NET, etc.

2.4.4. Definición de .NET

.NET es toda una nueva arquitectura tecnológica, desarrollada por Microsoft para la creación y distribución del software como un servicio. Esto quiere decir, que, mediante las herramientas de desarrollo proporcionadas por esta nueva tecnología, los programadores podrán crear aplicaciones basadas en servicios para la web.

2.4.5. .NET Framework

.NET Framework constituye la plataforma y elemento principal sobre el que se asienta Microsoft .NET. De cara al programador, es la pieza fundamental de todo este nuevo modelo de trabajo, ya que proporciona las herramientas y servicios que necesitará en su labor habitual de desarrollo. .NET Framework permite el desarrollo de aplicaciones a través del uso de un conjunto de herramientas y servicios que proporciona, y que pueden agruparse en tres bloques principales: el Entorno de Ejecución Común o Common Language Runtime (CLR a partir de ahora); la jerarquía de clases básicas de la plataforma o .NET Framework Base Classes; y el motor de

generación de interfaz de usuario, que permite crear interfaces para la web o para el tradicional entorno Windows, así como servicios para ambos entornos operativos.

2.4.6. Objetos y clases de Visual Basic.net

***Un objeto*.** es una combinación de código y datos que puede tratarse como una unidad. Un objeto puede ser una porción de una aplicación, como un control o un formulario. Una aplicación entera también puede ser un objeto.

Clase. cada objeto de Visual Basic está definido por una clase. Una clase describe las variables, propiedades, procedimientos y eventos de un objeto. Los objetos son instancias de clases; pueden crearse tantos objetos como sean necesarios una vez que se defina una clase.

Tipos de objetos.

Existen muchos tipos de objetos con los que trabaja Visual Basic, entre ellos podemos hablar de los formularios y controles.

Un formulario es una ventana mediante la cual los usuarios interactúan con la aplicación. En dicha ventana depositaremos los controles necesarios para crear nuestra interfaz con el usuario de la aplicación. La finalidad principal de un formulario es agrupar una serie de controles por medio de los cuales poder presentar y solicitar información al usuario.

Los controles son todos los objetos que se colocan en los formularios, mediante los cuales se realizan las acciones. A través de los controles se pueden escribir y recibir texto (etiquetas y cajas de texto), usar botones de comando, insertar o manipular imágenes, utilizar cuadros de diálogo, etc.

Proyecto. Un proyecto es un conjunto de formularios, conteniendo controles, objetos, las propiedades de esos objetos y el código Visual Basic de programación. Es decir que en un proyecto se encuentran enlazados todos los componentes que Visual Basic utiliza para desarrollar un programa.

Métodos. Los métodos son funciones propias de cada objeto. Así como las propiedades afectan cómo son los objetos, los métodos ejecutan acciones propias del mismo. Los métodos afectan el comportamiento de los objetos de un programa, y solamente se los utiliza en tiempo de ejecución. Por ejemplo, los formularios poseen un método llamado Show que se encarga de mostrarlos por pantalla.

Eventos. Un evento es una acción reconocida por un objeto (formulario o control). El evento puede ser causado por el usuario (por ejemplo: cuando pulsar una tecla), por el sistema (por ejemplo: transcurrió un determinado tiempo), o indirectamente por el código (por ejemplo: cuando el código carga un formulario se da el evento Load). Los eventos son sucesos a los que debe responder el programa.

2.5. Características de Programación en Visual Basic .NET.

Tabla 1

Características 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.

2.6. Entorno Visual Basic.Net

Después de haber creado un Proyecto de Windows Forms Application, se mostrará la ventana de Visual Basic:

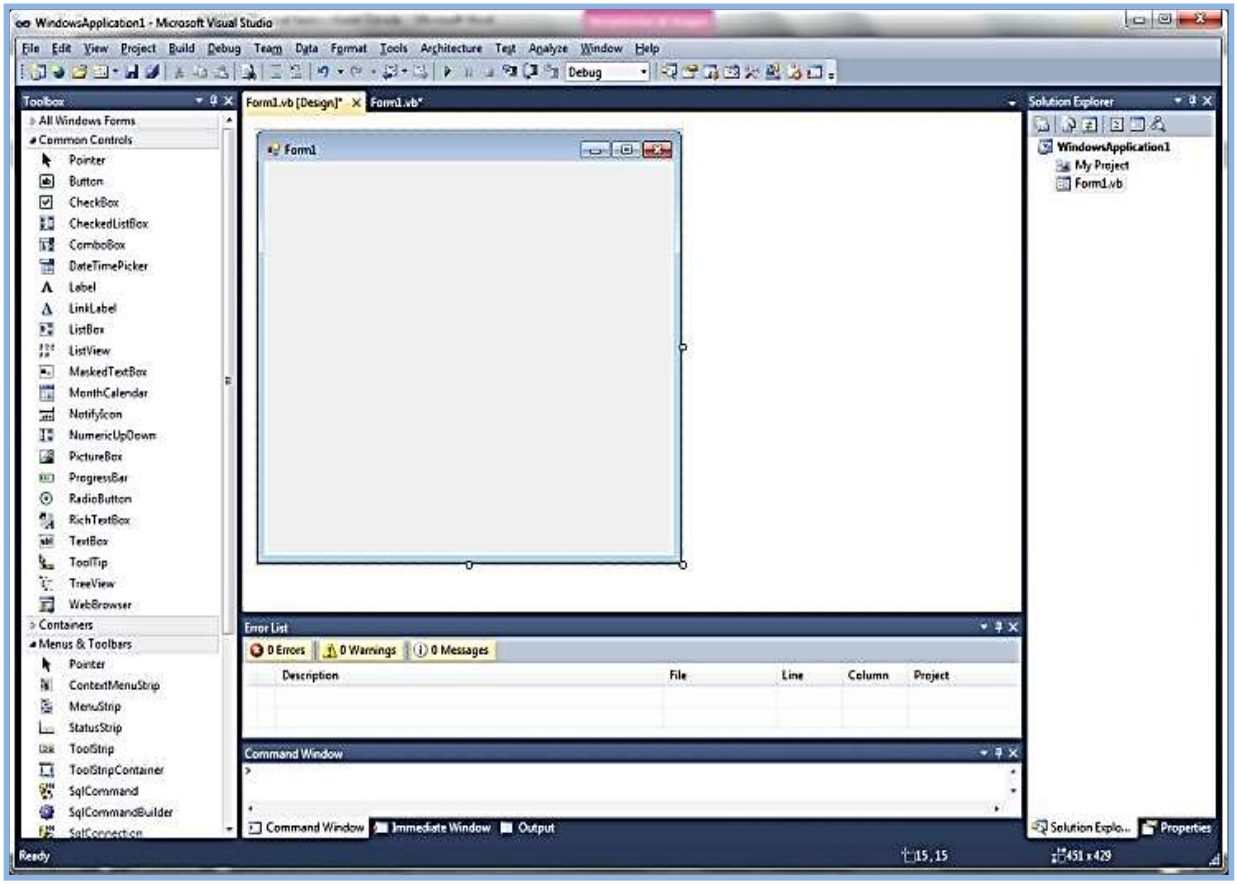


Figura 24 Entorno Visual Basic

2.6.1. Ventana exploradora de proyecto

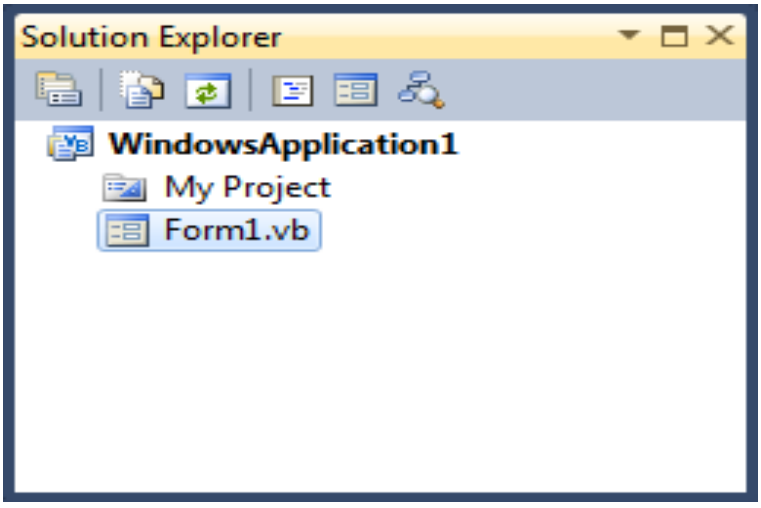


Figura 25 Ventana exploradora de proyectos

2.6.2. Cuadro de herramientas

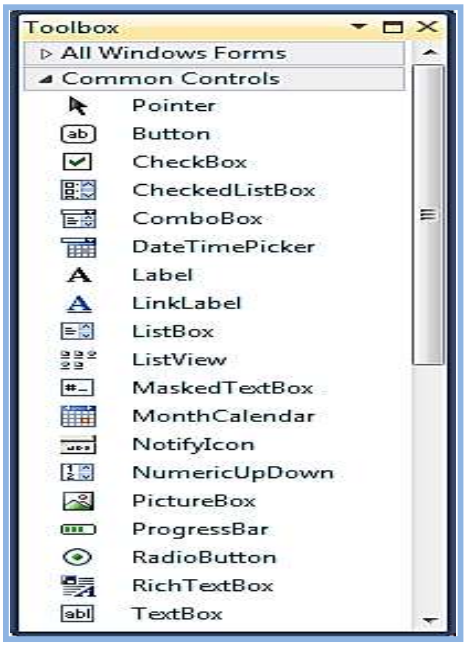


Figura 26 Ventana cuadro de herramientas

2.6.3. Ventana de propiedades

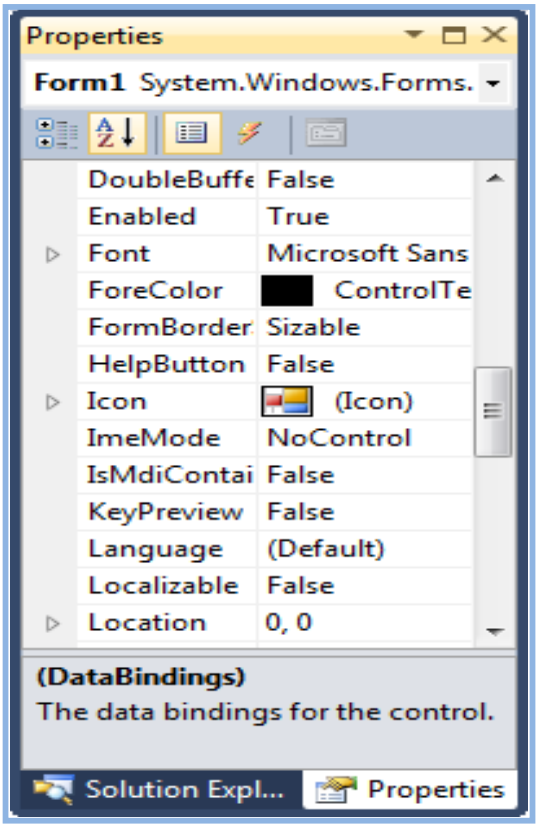


Figura 27 Ventana de propiedades

2.6.4. Ventana editor de código

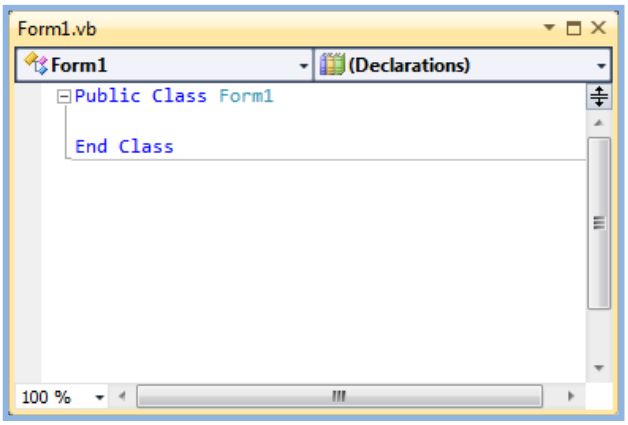


Figura 28 Ventana editor de código

2.6.5. Ventana de depuración

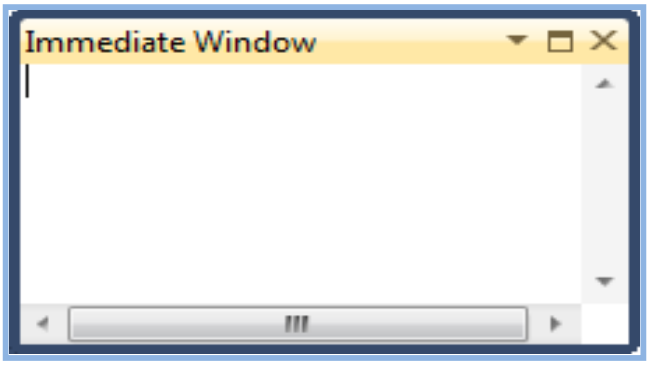


Figura 29 Ventana de depuración

2.6.6. Ventana de formulario

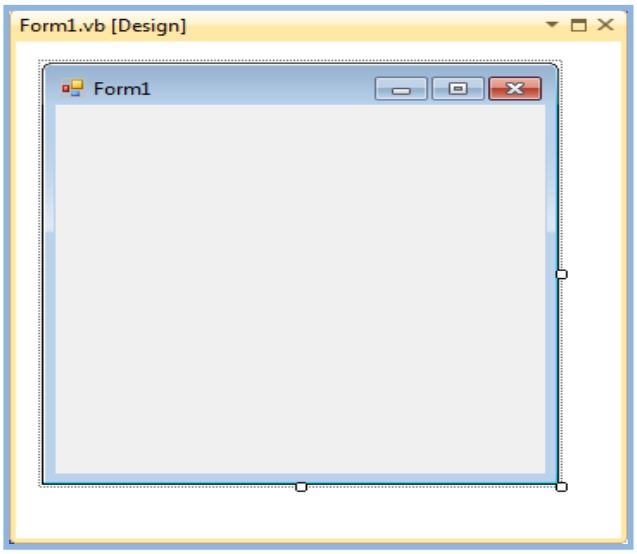


Figura 30 Ventana de formulario

Nota: VB .NET crea una nueva carpeta para el proyecto y guarda los ficheros del proyecto dentro de ella, incluso antes de que la edite. El IDE guarda los cambios en los ficheros del proyecto por defecto cada vez que lo ejecutamos.

2.7. Área del desarrollo de Visual.Net

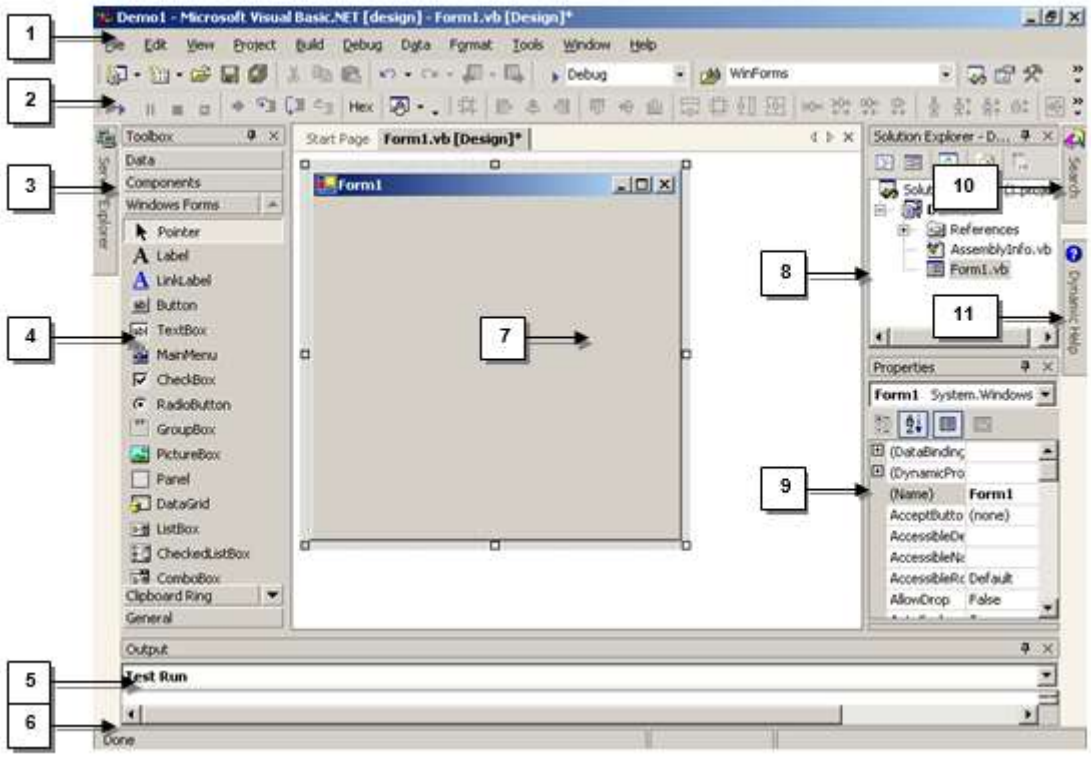


Figura 31 Ventana de área de desarrollo de Visual .Net

Entre las partes del nuevo IDE de Visual Studio .NET tenemos:

1. Menu Bar
2. ToolBars
3. Server Explorer Window (Ctrl + Alt + S)
4. ToolBox (Ctrl + Alt + X)
5. Output Window (Ctrl + Alt + O)
6. Status Bar
7. Windows Form Designer

8. Solution Explorer Window (Ctrl + R)
9. Properties Window (F4)
10. Search Window (Ctrl + Alt + F3)
11. Dynamic Help Window (Ctrl + F1)

Existen nuevas ventanas en Visual Studio .NET entre las cuales tenemos:

- Class View (Ctrl + Shift + C)
- Resource View (Ctrl + Shift + E)
- Macro Explorer (Alt + F8)
- Document Outline (Ctrl Alt + T)
- Task List (Ctrl + Alt + K)
- Command Window (Ctrl + Alt + A)
- Find Symbol Results (Ctrl +Alt + Y)

Nota: Para mostrar u ocultar cualquier ventana del IDE elegir el menú "View"

2.8. Terminologías en Visual.Net

- ***Objetos.***Instancia de la clase, tiene propiedades atributos.
- ***Clase.***Concepto, idea, las características y comportamientos comunes de los objetos.
- ***Propiedades.***Características de los objetos, calificativo.
- ***Métodos.***Se programa.
- ***Eventos.***Es una acción que se aplica a los objetos.

Algunos Objetos y Controles

- Formularios (Form)
- Botones de comando (Button)
- Etiquetas (Label)
- Cuadros de textos (TextBox)

- CheckBox
- RadioButton
- ListBox

Algunas Propiedades

- Name (nombre)
- Caption (título)
- Text (texto)
- Font (fuente)
- Fore color (color de primer plano)
- Backcolor (color de fondo)
- Enabled (disponible)

2.9. Administración de Ventanas

El manejo de ventanas en Visual Studio .NET es más simple y rápido pudiendo acceder a cualquier elemento de manera fácil, debido a las nuevas características de Administración de ventanas, tales como:

Auto ocultar. Esta característica es nueva en Visual Studio .NET y permite ocultar una ventana permitiendo liberar espacio en el IDE, para mostrar nuevamente la ventana solo hay que ubicar el mouse cerca del nombre de la ventana que aparece en una ficha.

Ventanas acoplables. Al igual que Visual Basic 6, esta nueva versión permite acoplar ventanas las cuales estarán fijas en el IDE. Podemos elegir si una ventana se va a "Auto Ocultar" o si se va a "Acoplar". Al acoplar la ventana tendremos la posibilidad de ver siempre su contenido.

Fichas de documentos. En la versión anterior de Visual Studio el trabajo con varios documentos era tedioso porque para acceder a un documento abierto (por ejemplo, un

módulo de formulario) había que hacerlo mediante el menú “Window” o dando clic en el botón “View Code” o doble clic sobre el nombre del objeto. Ahora el acceso es muy rápido a través de las fichas que hay en la parte superior del Editor.

Navegación a través del IDE. Podemos navegar a través de los documentos visitados usando la barra Web, pudiendo ir hacia “Atrás”, “Adelante”, “Detener”, “Actualizar”, “Ir al inicio” como si se tratase de un Browser y si navegáramos a través de páginas Web, lo que facilita la búsqueda de una página ya abierta.

Ventana de ayuda Rápida. Una de las características más importantes de Visual Studio .NET es la “ayuda inteligente” o “ayuda rápida” que permite mostrar en una ventana todos los tópicos relacionados a donde se encuentre el cursor (si está en el editor) o al objeto seleccionado (si estamos en el diseñador de formulario), por ejemplo, si estamos en el editor escribiendo una función aparecerán los tópicos relacionados a ésta, si nos encontramos seleccionando un control, aparecerán los temas referentes a éste.

Todas estas nuevas características hacen que el trabajo del desarrollador sea más productivo, centrándose en la lógica de la aplicación y no en el mantenimiento de ésta ya que es más fácil al utilizar las nuevas características de Administración de Ventanas, anteriormente comentadas.

Diseñadores.

Para realizar la construcción de aplicaciones o creación de componentes o servicios disponemos de diseñadores que facilitan la labor de construcción de interfaces, creación de sentencias, etc.

La mayoría de diseñadores se habilitan al elegir una plantilla de Visual Studio .NET y casi todos generan código al diseñar controles sobre el contenedor respectivo;

característica totalmente distinta a la forma de trabajo en Visual Basic 6, que ocultaba el código generado por el diseñador.

Entre los diseñadores que trae Visual Studio .NET tenemos:

Windows Form Designer. Se muestra al elegir cualquiera de dos plantillas: "Windows Application" o "Windows Control Library", habilitando en el Toolbox los controles para Windows que serán usados para construir la interfase de la aplicación arrastrando dichos controles hacia el formulario o control de usuario.

Web Form Designer. Se muestra al elegir la plantilla "Web Application" habilitando en el Toolbox los controles para Web y los controles HTML que serán usados para construir la página Web que correrá en el servidor IIS (archivo aspx) arrastrando dichos controles hacia el formulario Web.

Component Designer. Este diseñador se muestra al elegir una de dos plantillas: "Class Library" o "Windows Service" y también trabaja con los controles para Windows, creando una interfase reusable desde otra aplicación.

Web Service Designer. Sirve para diseñar servicios Web y es mostrado al elegir una plantilla "Web Service", también trabaja con los controles para Windows, componentes, etc.

Existen más diseñadores, pero que lo trataremos en la categoría de herramientas de datos, debido al tipo de trabajo que realizan con datos, el cual se trata como tema siguiente.

Herramientas de datos.

Si se quiere realizar un trabajo rápido con datos, tal como modificar la estructura de la base de datos, crear tablas, consultas, procedimientos almacenados, etc., existen herramientas que permiten realizar esta labor reduciendo enormemente el proceso de desarrollo en el caso de hacerse por otros medios.

Entre las principales herramientas que trabajan con datos tenemos:

Server Explorer. Sin duda es una de las principales herramientas de Visual Studio .NET y no solo para acceder a datos sino para mostrar y administrar los diferentes servidores o recursos del sistema, tales como Base de Datos, Servicios Web, Aplicaciones COM+, etc. Con solo arrastrar el objeto éste puede ser usado en una aplicación. También se tratará con mayor detalle en el módulo de acceso a datos.

DataAdapter Wizard. Es un Asistente que permite crear un DataAdapter que es un Comando (Select, Insert, Update, Delete) con el cual se podrá generar un conjunto de registros o Dataset. La misma función puede ser cubierta por el Server Explorer con solo arrastrar los campos hacia el formulario.

Query Designer. Es un Diseñador que permite crear Consultas SQL de manera sencilla arrastrando tablas o consultas sobre éste y eligiendo los campos que se verán en la consulta de datos, también se puede realizar filtros o especificar criterios de selección. Además, no solo se pueden construir consultas Select sino también se pueden crear consultas Insert, Update o Delete, etc.

DataBase Project. Es un tipo de plantilla que sirve para trabajar con una Base de Datos, para lo cual debe existir una conexión a un origen de datos, este tipo de proyecto da la posibilidad de crear y modificar scripts de creación de tablas, consultas, vistas, desencadenantes, procedimientos almacenados, etc.

Editor de Script. Uno de las principales herramientas para trabajar con base de datos remotas como SQL Server, Oracle, etc., es utilizar el editor de Scripts que permite crear tablas, consultas, vistas, etc. Mostrando con colores las sentencias o palabras reservadas del lenguaje Transact-SQL.

Depurador de procedimientos almacenados. Visual Studio .NET incorpora un Depurador de Store Procedure que puede realizar seguimientos paso a paso por línea de

código, por sentencia o por instrucción, además crea puntos de interrupción, permitiendo un mayor control y seguimiento del código en caso de errores.

Todas estas herramientas mencionadas, son nuevas en Visual Studio .NET, a excepción del Query Builder que es el mismo de la versión anterior de Visual Studio. Como se habrá dado cuenta muchas herramientas de acceso a datos de Visual Basic 6 han sido eliminadas, tales como: Data Environment, Data View, Data Report, y otras más, pero en su reemplazo existen las que ya hemos mencionado.

Tabla 2

Tipos de datos

Tipo V. Basic	**Estructura Tipo .NET Runtime**	**Tamaño Almac.**	**Rango de Valores**
Boolean	System.Boolean	4 bytes	True o False
Byte	System.Byte	1 byte	0 to 255 (sin signo)
Char	System.Char	2 bytes	0 to 65535 (sin signo)
Date	System.DateTime	8 bytes	Enero 1, 1 CE hasta diciembre 31, 9999
Decimal	System.Decimal	12 bytes	+/- 79,228,162,514,264,337,593,543,950,335 sin punto decimal; +/-7.9228162514264337593543950335 con 28 posiciones a la derecha del decimal; número más corto (no 0) es +/-0.0000000000000000000000000001

Double (doble-precisión punto-flotante)	System.Double	8 bytes	-1.79769313486231E308 hasta -4.94065645841247E-324 para valores negativos; 4.94065645841247E-324 hasta 1.79769313486232E308 para valores positivos
Integer	System.Int32	4 bytes	-2,147,483,648 to 2,147,483,647
Long (Entero largo)	System.Int64	8 bytes	-9,223,372,036,854,775,808 hasta 9,223,372,036,854,775,807
Object	System.Object (class)	4 bytes	Cualquier tipo de dato
Short	System.Int16	2 bytes	-32,768 to 32,767
Single (simple precisión punto-flotante)	System.Single	4 bytes	-3.402823E38 hasta -1.401298E-45 para valores negativos; 1.401298E-45 hasta 3.402823E38 para valores positivos
String (tamaño-variable)	System.String (class)	10 bytes + (2 * tamaño cadena)	0 hasta aproximadamente 2 billones de caracteres Unicode
User-Defined Type (estructura)	(heredado desde System.ValueType)	Suma de tamaños de sus miembros	Cada miembro de la estructura tiene un rango determinado, es decir pueden tener sus propios tipos de datos distintos unos de otros

Notas: Se ha eliminado el tipo de dato Variant y es reemplazado por Object, también el dato Currency ahora es Decimal y el Type ahora es Structure. Además, no existen String de tamaño fijo, sino que todos son dinámicos.

Variables.

Una variable es un dato temporal en memoria que tiene un nombre, un tipo de dato, un tiempo de vida y un alcance, los cuales lo dan la forma como se declare esta.

Una variable debe cumplir con las siguientes reglas:

- Debe iniciar con un carácter alfabético.
- Debería contener solo caracteres alfabéticos, dígitos y carácter de subrayado.
- El nombre no debe exceder a 255 caracteres, etc.

Declaración de variables.

A diferencia de Visual Basic 6, en VB.NET se pueden declarar varias variables en una sola instrucción y además se puede asignar directamente sus valores. Otra observación es que es necesario definir el tipo de declaración y el tipo de dato (antes si no se hacía se asumía un tipo de declaración y un tipo de dato variante, que ahora no existe).

Sintaxis: <Tipo de Declaración> <Variable(s)> As <Tipo de Dato>[=<Valor>]

Existen varios tipos de declaración que detallamos a continuación en la siguiente tabla:

Tabla 3

Declaración de variables

Declaración	**Lugar de Declaración**	**Alcance o Ámbito**
Public	Módulo o Clase	Global, en todo el proyecto.

Protected	Clase	En la clase declarada o en una derivada.
Friend	Clase	En el Assemblie.
Private	Módulo	Solo en el módulo.
Dim	Procedimiento	Solo en el procedimiento.
Static	Procedimiento	Solo en el procedimiento.

Alcance de variables.

Para las variables declaradas a nivel de procedimiento (Dim y Static) existe un nuevo alcance que es a nivel de estructura o bloque, que puede ser For – Next, If – End If, Do – Loop, etc. Las variables definidas dentro de un bloque solo valdrán en este bloque.

Opciones de trabajo con variables.

Por defecto en VB NET es necesario declarar las variables usadas (Option Explicit) y también es necesario que se asigne el mismo tipo de dato a la variable (Option Strict), si deseamos Declaración implícita (por defecto Object) y Conversión Forzosa de tipos (ForeCast), aunque no es lo recomendable por performance, podemos conseguirlo de dos formas: mediante “Propiedades” del proyecto, opción “Build” y elegir “Off” en las listas de “Option Explicit” y Option Strict” o mediante declaración al inicio de todo el código:

Option Explicit Off

Option Strict Off

Capítulo III: Operaciones con Visual.Net

3.1.Estructuras de control

Las sentencias de control denominadas también estructuras de control, permiten tomar decisiones y realizar un proceso repetidas veces. Visual Basic.Net dispone de las siguientes estructuras:

Estructura de decisión.

a. Sentencia If .... Then

Permite tomar una decisión referente al camino a seguir o acción a ejecutar en un proceso basándose en el resultado (verdadero o falso) de una condición. Su sintaxis es:

```
If condición then
accion1
end if
```

Donde la condición debe ser una expresión numérica, relacional o lógica. Si la condición es verdadera se ejecuta la acción1.

b. Sentencia If... Then Else

Permite tomar una decisión referente al camino a seguir o acción a ejecutar en un proceso basándose en el resultado (verdadero o falso) de una condición. Su sintaxis es:

```
If condición then
accion1
Else
accion2
End if
```

Donde la condición debe ser una expresión numérica, relacional o lógica. Si la condición es verdadera se ejecuta la ccion1 y si es falsa se ejecutará la accion2.

c. Sentencia SELECT

Esta expresión permite ejecutar una de varias acciones en función del valor de una expresión. Es una alternativa a If... Then Elself cuando lo que se necesita es comprobar es la misma expresión con diferentes valores. Su sintaxis es:

Select case expresión

case lista1

sentencias

case Lista2

sentencias

case else

sentencias n

end select

Donde la expresión es una expresión numérica o alfanumérica, y lista1 y lista2.... representan una lista que puede tener cualquiera de las formas siguientes:

expresión [, expresión]...

expresión to expresión

Is operador-de-relación expresión

combinación de las anteriores separadas por comas

3.2. Estructuras de Repetición

a. Do Loop

Un Loop (buc|e) repite la ejecución de un conjunto de sentencias mientras una condición dada sea cierta, o basta que una condición dada sea cierta. La condición puede ser verificada antes o después de ejecutarse el conjunto de sentencias:

Formatol

Do

[{WhiIe|Unti|} condición]

[sentencias]

[Exit do]

[sentencias]

Loop

Formato2

Do

[sentencias]

[Exit do]

[sentencias]

Loop[{While|Unti|} condición]

Donde condición es cualquier expresión que se evalué a True o a False.

b. Sentencias: For ... Next

La sentencia for da lugar a un bucle que permite ejecutar un conjunto de sentencias cierto número de veces. Su sintaxis es:

for variable = expresionl to expresion2[Step expresion3]

[sentencias]

[Exit for]

[sentencias]

Next [variab|e[variab|e...]]

Cuando se ejecuta una sentencia For en la que el valor de la expresién3 es positivo o no se ha especificado, primero se asigna él valor de la expresión 1 a la variable y a continuación se comprueba si la variable es mayor que la expresión 2, en cuyo caso se

salta el cuerpo de| bucle y se continúa en la línea que este a continuación de la sentencia Next. En otro caso, se ejecutan las líneas de programa que haya entre la sentencia For y la sentencia Next. Por último, la variable, se incrementa en él valor de la expresi6n 3, o en 1 si Step no se especifica, volviéndose a efectuar la comparación entre la variable y la expresión 2, y así sucesivamente. La sentencia exit for permite salir del bucle for... next antes de que este finalice.

3.3.Procedimientos y funciones

Procedimiento.

Los procedimientos son muy interesantes y útiles en la programación. Nos sirven para realizar una tarea concreta que probablemente se vaya a ejecutar varias veces a lo largo de la vida del programa. Esta tarea se especifica en un bloque de código de manera independiente y cuando se desean realizar las acciones del procedimiento se llama al procedimiento o función. Una vez realizadas las acciones pertinentes se devuelve el flujo de| programa al lugar desde donde se invocó ese procedimiento.

Lo primero que debemos hacer al crear un procedimiento es pensar las cosas que se desean hacer dentro de la función, la información que necesitaremos (y que tendremos que recibir como parámetros) y la información que devolveré.

Con estas ideas claras se pueden construir los procedimientos y funciones sin mucha dificultad, siguiendo estas estructuras:

Para un procedimiento.

```
Sub nombre (parametro1, parametro2...)
        …. Código de| procedimiento
end Sub
```

Funciones.

Una función en Visual Basic net es un módulo de un programa separado del cuerpo principal, que realiza una tarea específica y que puede regresar un valor a la parte principal del programa u otra función o procedimiento que la invoque.

La forma general de una función es:

```
Function Nom_fun(parametros)
        instrucciones
        nomfun = cargarlo porque es quien regresa el dato
End Function
```

La lista de parámetros formales es una lista de variables separadas por comas (,) que almacenaran los valores que reciba la función estas variables actúan como locales dentro de| cuerpo de la función. Aunque no se ocupen parámetros los paréntesis son requeridos.

Dentro del cuerpo de la función deber haber una instrucción que cargue el NOMFUNCION para regresar el valor, de esta manera se regresan los datos.

Sin embargo, es de considerar que NOMFUNCION puede regresar un dato, una variable o una expresión algebraica (no ecuación o formula)

3.4.Arreglos

Unidimensionales.

También son llamados arrays unidimensionales y lo podríamos definir como un conjunto de variables de| mimo tipo y tamaño que ocupan posiciones consecutivas en la memoria del computador. El tamaño en memoria que ocupa un array es siempre fijo y no puede variar. Para calcular el tamaño en memoria que puede ocuparnos un array solo tenemos que multiplicar el número de elementos de nuestro array por el tamaño en bytes de| tipo de este.

La estructura más simple es el arreglo unidimensional, que consiste de una columna de localizaciones de memoria. El siguiente arreglo es un arreglo unidimensional llamado AGE. Los elementos dados de| arreglo son similares a los nombres de referencia, dado que el primer elemento (con el dato 32) es la caja 1 (en lugar de 0) del arreglo. Los nombres de referencia se escriben como AGE(1), AGE(2), y así sucesivamente. El número entre paréntesis es solo un número de referencia y puede ser una constante, una variable o una expresión.

Arreglo unidimensional.

Variable: Array

Reference	AGE	Name
1	32	AGE(1)
2	54	AGE(2)
3	25	AGE(3)
4	36	AGE(4)
5	45	AGE(5)
6	20	AGE(6)
7	28	AGE(7)
8	50	AGE(8)
9	42	AGE(9)

El número entre paréntesis hace referencia o apunta al número de la caja en el arreglo, que es el número del elemento.

Arreglos bidimensionales (matrices).

Un array bidimensional (también llamado tabla o matriz) es un array con dos índices. Al igual que los vectores deben ser ordinales. Se declaran de igual manera que los arrays de una dimensión.

Un array bidimensional recoge valores de una tabla de doble entrada. Cada uno de los elementos se identifica y se asigna mediante una variable ($nombre) seguida de dos () que contienen los índices del array. Los índices puede ser escalares, equivaldrían al número de fila y columna que la celda ocuparía en la tabla, o puede ser asociativo que equivaldría en alguna medida a usar como índices los nombres de la fila y de la columna.

Un array bidimensional (tabla o matriz) es un array con dos índices, al igual que los vectores que deben ser ordinales o tipo subrango.

	Columnas				
1	2	3	4	5	
A[1,1]	A[1,2]	A[1,3]	A[1,4]	A[1,5]	1
A[2,1]	A[2,2]	A[2,3]	A[2,4]	A[2,5]	2
A[3,1]	A[3,2]	A[3,3]	A[3,4]	A[3,5]	3 **Filas**
A[4,1]	A[4,2]	A[4,3]	A[4,4]	A[4,5]	4

Para localizar o almacenar un valor en el array se deben especificar dos posiciones (dos subíndices), uno para la fila y otro para la columna.

Formato:

identificador = array [indice1, índice 2] of tipo de elemento

identificador = array [índice 1] of array [índice 2] of tipo de elemento

3.5.Acceso de Base de Datos

Los programadores suelen necesitar crear bases de datos mediante programación. Describiremos cómo utilizar ADO.NET y Visual Basic .NET para crear una base de datos de Microsoft SQL Server mediante programación.

Pasos para crear una base de datos:

1. Cree un nuevo proyecto de aplicación para Windows en Visual Basic.NET. Se agregará Form1 al proyecto de forma predeterminada.
2. Coloque un botón Command en Form1 y cambie su propiedad Name a btnCreateDatabase y su propiedad Text a Crear base de datos.
3. Copie y pegue la línea de código siguiente en la sección de declaraciones generales de Form1: Imports System.Data.SqlClient
4. Copie y pegue el código siguiente después de la región "Código generado por el Diseñador de Windows Forms":

```
        Private Sub btnCreateDatabase_Click(ByVal sender Byval As
    System.Object, _
        ByVal e As System.EventArgs) Handles btnCreateDatabase.Click
            Dim str As String
        Dim myConn As SqlConnection = New
    SqlConnection("Server=(local)\netsdk;"    &
    _"uid=sa;pwd=;database=master")
  str = "CREATE DATABASE MyDatabase ON PRIMARY " & _
    "(NAME = MyDatabase_Data, " & _
```

```
            " FILENAME = 'D:\MyFolder\MyDatabaseData.mdf', " & _
            " SIZE = 2MB, " & _
            " MAXSIZE = 10MB, " & _
            " FILEGROWTH = 10%) " & _
            " LOG ON " & _
            "(NAME = MyDatabase_Log, " & _
            " FILENAME = 'D:\MyFolder\MyDatabaseLog.ldf', " & _
            " SIZE = 1MB, " & _
            " MAXSIZE = 5MB, " & _
            " FILEGROWTH = 10%) "
    Dim myCommand As SqlCommand = New SqlCommand(str, myConn)
    Try
        myConn.Open()
        myCommand.ExecuteNonQuery()
        MessageBox.Show("Base de datos se está creando", _
                    "Mi Programa", MessageBoxButtons.OK, _
                     MessageBoxIcon.Information)
      Catch ex As Exception
                        MessageBox.Show(ex.ToString())
      Finally
                        If (myConn.State = ConnectionState.Open) Then
                    myConn.Close()
                        End If
      End Try
End Sub
```

5. Cambie la cadena de conexión para que señale a SQL Server y asegúrese de que el argumento Database está establecido en Master o está en blanco.
6. Presione F5 o CTRL+F5 para ejecutar el proyecto y, a continuación, haga clic en Crear base de datos.

Notas adicionales.

- Este código crea una base de datos personalizada con determinadas propiedades.
- La carpeta que contendrá los archivos .mdf y .ldf creados ya deberá existir antes de ejecutar el código. En caso contrario, se generará una excepción.
- Si desea crear una base de datos similar a la base de datos modelo de SQL Server y en la ubicación predeterminada, cambie la variable str en el código:

  str = "CREATE DATABASE Mi_Base_de_Datos"

3.6.Aplicaciones

La pantalla principal de Visual Basic .NET es el IDE de Visual Studio .NET. La ventana principal es el diseñador de formularios, la superficie gris (Formulario) es la ventana de la nueva ventana en modo diseño.

Procedimiento para crear una nueva aplicación de Consola.

- Click sobre el Menú Archivo
- Seleccionar Nuevo
- Click sobre proyecto
- Verificar que se encuentre seleccionado Visual Basic – Windows (Tipos de proyectos)
- Click sobre Aplicación de Consola en Plantillas
- Digitar el Nombre de la aplicación a crear (opcional), por ejemplo, Ejercicio2
- Click sobre el botón Aceptar

Se visualiza:

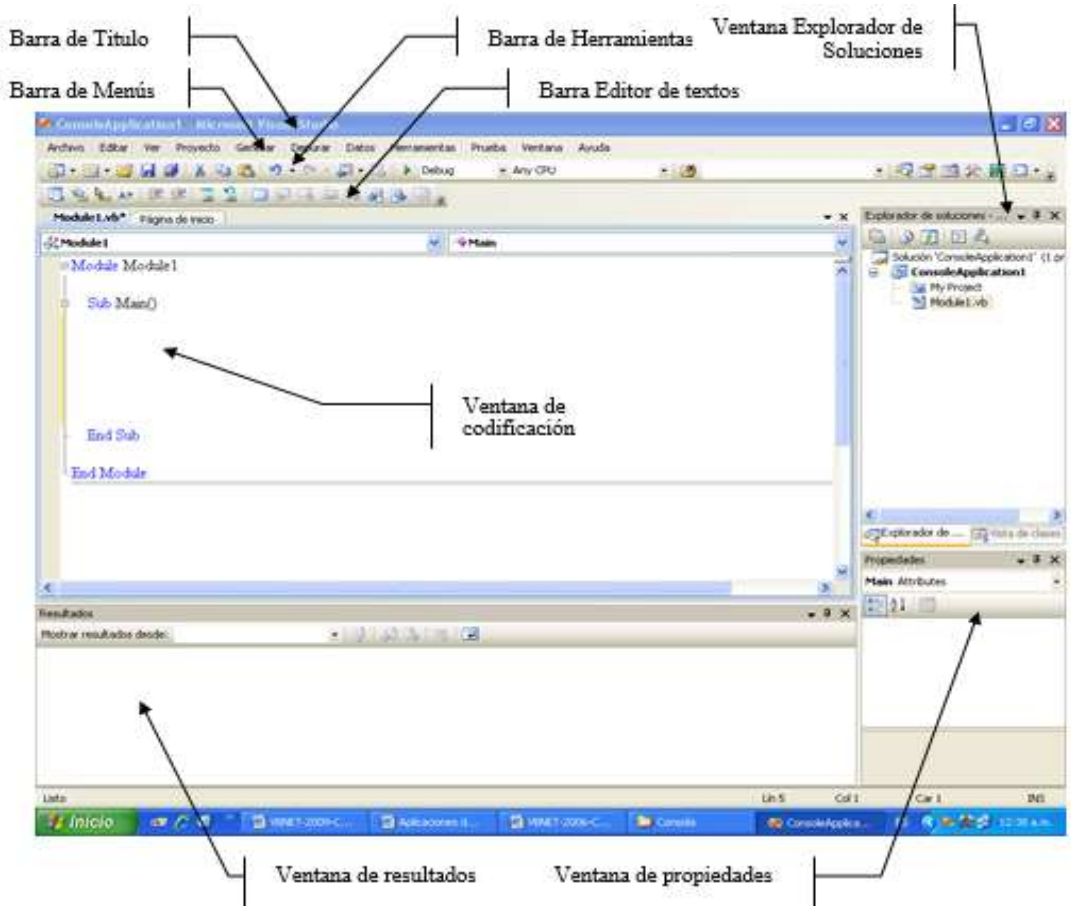


Figura 32 Ventana Visual Basic.net

```
Module Module1

    Sub Main()

    End Sub

End Module
```

Es decir, se crea un módulo llamado Module1 y un procedimiento llamado Sub Main, que por otro lado es el que sirve como punto de entrada al programa. Por ahora

sólo debes saber que los procedimientos Sub son como instrucciones y cuando se usan en otras partes del programa, se ejecuta el código que haya en su interior.

Nota:

- Una aplicación de Consola es un Módulo. Main () es el nombre de la subrutina que se ejecutara automáticamente cuando se inicia la aplicación de Consola. El código a ejecutar debe estar situado entre las sentencias Sub Main () y End Sub.
- Una aplicación de consola usa la misma codificación a la aplicación Windows, con la diferencia que esta usa la sentencia Console.Writeline para mostrar los mensajes en la ventana comandos.
- Una aplicación de consola no reacciona ante eventos porque no tiene una interfaz visible, sin embargo, es fácil añadir elementos de una interfaz de Windows a una aplicación de consola.
- La utilidad de las aplicaciones de consola es la de poder comprobar una determinada característica del lenguaje sin tener que construir una interfaz de usuario.
- Las dos últimas líneas son iguales para todas las aplicaciones de Consola utilizando la sentencia Console.ReadLine. Sin ellas, la ventana de comandos se cerraría tan pronto como se llegará a la sentencia End Sub, y no tendría la oportunidad de ver el resultado.

Primera aplicación:

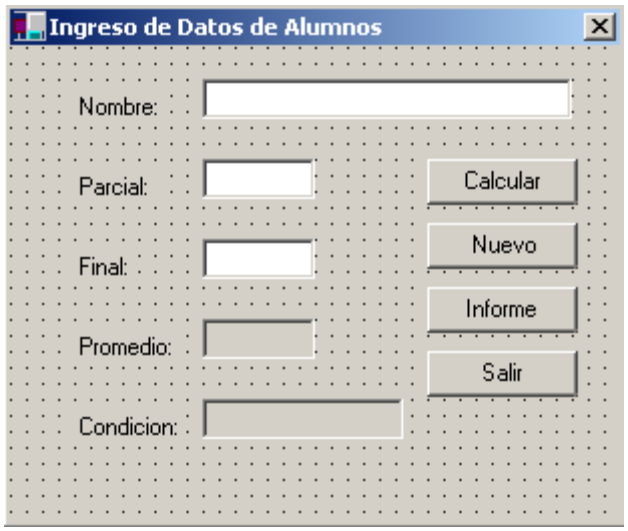


Figura 33 Formulario de ingreso de datos

Para realizar los cálculos de la aplicación vamos a crear variables y funciones públicas en un módulo estándar, para lo cual del menú "Project" elegimos la opción "Add Module" y luego escribimos en el nombre "Módulo.vb"

Escribir el siguiente código en el módulo:

```
Module Módulo
    Public NTotal, NAprobados, NDesaprobados As Byte
    Public Suma, PromTot As Single
    Public Function Calcular_Promedio(ByVal Par As Byte, _
      ByVal Fin As Byte) As Single
        Return ((Convert.ToSingle(Par + 2 * Fin)) / 3)
     End Function

    Public Function Calcular_Condicion (ByVal Pro As Single) _
      As String
```

```
        NTotal = +1
        Suma = +Pro
        If Pro > 10.5 Then
            NAprobados = +1
            Return ("Aprobado")
        Else
            NDesaprobados = +1
            Return ("Desaprobado")
        End If
    End Function
End Module
```

Regresar al formulario y proceder a escribir código en los eventos de los botones, tal como se muestra a continuación:

```
    Protected Sub btnCalcular_Click(ByVal sender As Object, …)
        Dim Par, Fin As Byte
            Dim Pro As Single
            Par = Convert.ToByte(txtParcial.Text)
             Fin = Convert.ToByte(txtFinal.Text)
             Pro = Calcular_Promedio(Par, Fin)
             txtPromedio.Text = Format(Pro, "#.00")
             txtCondicion.Text = Calcular_Condicion(Pro)
    End Sub
Protected Sub btnNuevo_Click(ByVal sender As Object, …)
        Dim X As Control
         For Each X In Me.Controls
```

```
                If TypeOf X Is TextBox Then X.Text = ""
        Next
        txtNombre.Focus()
    End Sub
Protected Sub btnInforme_Click(ByVal sender As Object, …)
        Dim X As New frmInforme()
        X.ShowDialog()
    End Sub
Protected Sub btnSalir_Click(ByVal sender As Object, …)
        Me.Close
    End Sub
```

Segunda aplicación

Para mostrar el informe de situación de alumnos ir al menú “Project” y elegir la opción “Add Windows Form” y en el nombre escribir “frmInforme.vb”. Luego realizar el diseño del formulario, tal como se muestra en la figura de abajo.

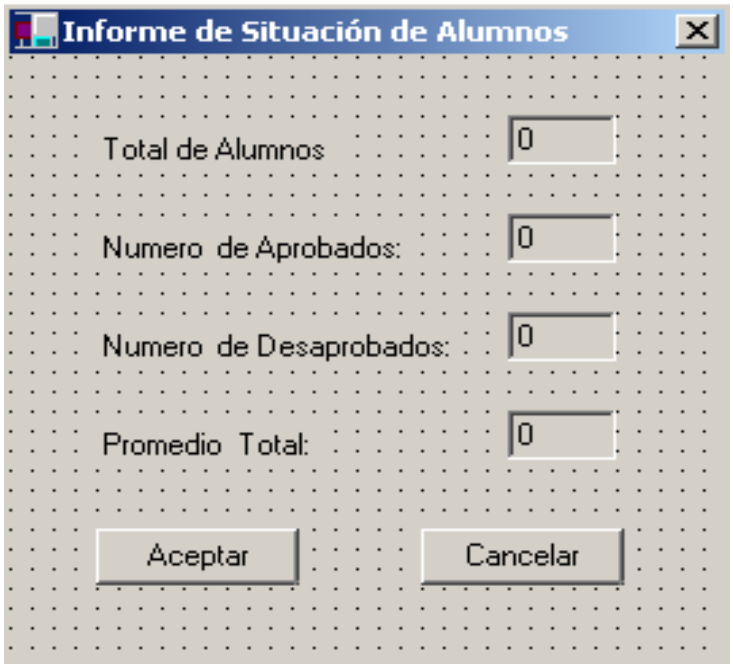


Figura 34 Diseño de formulario

Ingresar al código del formulario “frmInforme” y proceder a escribir código en el evento “Load” para que se muestre las estadísticas de los alumnos:

```
Public Sub frmInforme_Load(ByVal sender As Object, …)
    txtTotal.Text = NTotal.ToString
    txtAprobados.Text = NAprobados.ToString
    txtDesaprobados.Text = NDesaprobados.ToString
    txtPromedioTotal.Text = Format((Suma / NTotal), "#.00")
End Sub
```

Luego programar en los botones de “Aceptar” y “Cancelar” el regreso y la finalización de la aplicación respectivamente, similar al código mostrado abajo:

```
Protected Sub btnAceptar_Click(ByVal sender As Object, …)
    Me.Close()
End Sub
Protected Sub btnCancelar_Click(ByVal sender As Object, …)
    End
End Sub
```

Grabar y ejecutar la aplicación; para probar ingresar como mínimo los datos de dos alumnos, mostrar el informe y finalizar.

Diríjase al “Explorador de Windows” y analice el contenido de la carpeta “Lab01_1”, observe que existen varios tipos de extensiones, sln para la solución, vbproj para la aplicación y vb para los formularios y el módulo.

Finalmente, ingrese a la carpeta “bin” de “Lab01_1” y observe dos archivos: uno ejecutable (extensión exe) y otro archivo intermedio (extensión pdb).

Aplicación didáctica

Sesión de Aprendizaje

Programa en Visual Basic.Net

I. DATOS GENERALES DEL CURSO

1. Curso : Computación
2. Grado y Secc : 3ª C Nivel Secundaria
3. Tema : Visual Basic.Net
4. Fecha :
5. Duración : 2 Horas
6. Turno : Diurno
7. Profesor : León Pedro Laureano Julca

II. OBJETIVOS GENERALES

- Primer programa en Visual Basic.Net.
- Utilización de comandos procedimientos y eventos.

III. CONTENIDOS

- Aplicación con Visual Basic.Net.
- Aplicación de comandos procedimientos y eventos.

IV. PROCESO DE ENSEÑANZA – APRENDIZAJE

<table>
<tr><th>FASE</th><th>ACTIVIDAD</th><th>T</th><th>PREVENCIÓN DE AYUDA</th></tr>
<tr><td>INICIO</td><td>EL PROFESOR:
➢ Presenta las actividades a realizar.
➢ Vista panorámica de lo que va a tratar.
➢ Explica las herramientas a utilizar.
➢ Explica la lógica de la clase.</td><td>10’</td><td rowspan="3">ESTRATEGIAS METODOLÓGICAS

Las clases se desarrollan en los laboratorios haciendo una exposición clara de los temas e incentivando la participación del grupo, los métodos que empleara el docente serán dinámicos y examinara junto con el alumno.

MEDIOS Y MATERIALES DIDÁCTICOS</td></tr>
<tr><td>PROCESO</td><td>EL ESTUDIANTE:
➢ Sigue las indicaciones correspondientes del profesor.
➢ Conoce las herramientas a utilizar.
➢ Entiende la lógica del diseño de programación.
➢ Realizan las inserciones de los archivos.</td><td>35’</td></tr>
<tr><td>EVALUACIÓN</td><td>Se evaluará al alumno con criterios, indicadores e instrumentos de evaluación.</td><td>25’</td></tr>
</table>

CULMINACIÓN	El profesor, hace el reforzamiento o retroalimentación del tema y pregunta a los estudiantes lo que han entendido.	10’	➢ Computadoras ➢ Memoria USB ➢ Guía de practica ➢ Pizarra acrílica ➢ Plumones ➢ Mota ➢ Multimedia ➢ Lenguaje de Programación Visual Basic.Net
EXTENSIÓN	El profesor explica el trabajo de investigación que deben de presentar los estudiantes en la próxima clase.	10’	

REFERENCIA

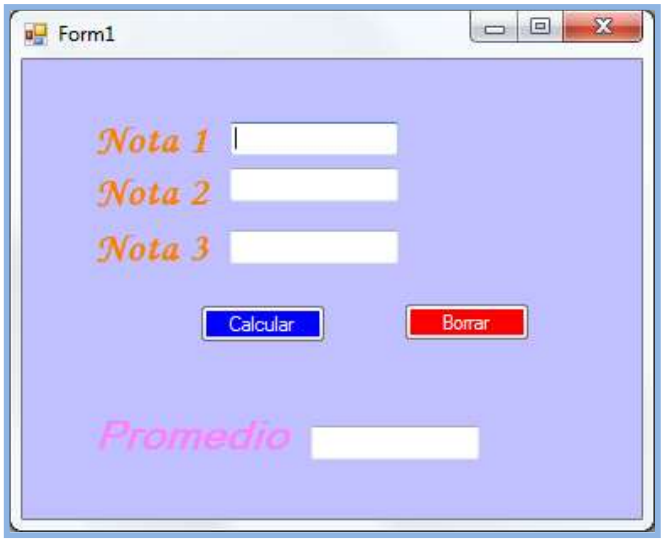
- http://www.aulaclic.es/
- Coronel, E. (1994). *Creando soluciones* VB.Net. Macro. Lima, 2015.

APLICACIÓN 1: APÉNDICE

Guía de Laboratorio

Ejercicio 1:

Programa que permite el ingreso de notas y permite calcular el promedio

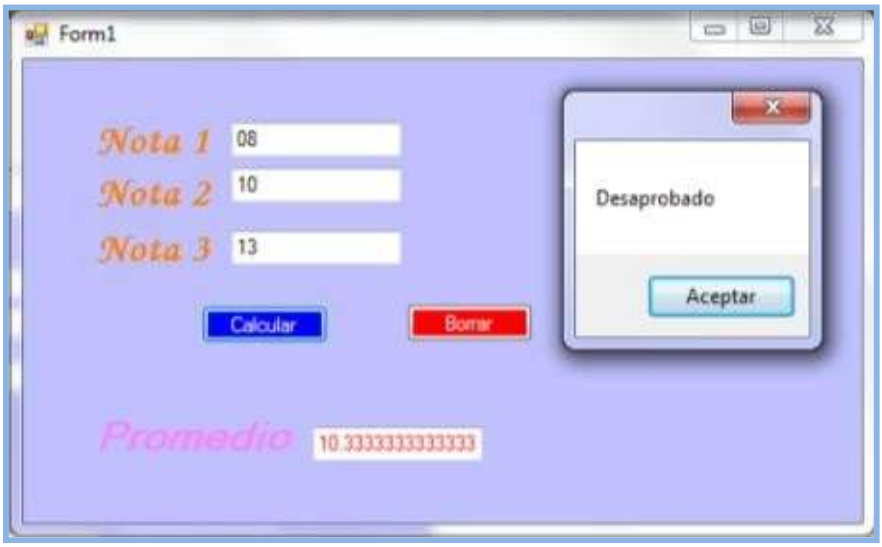


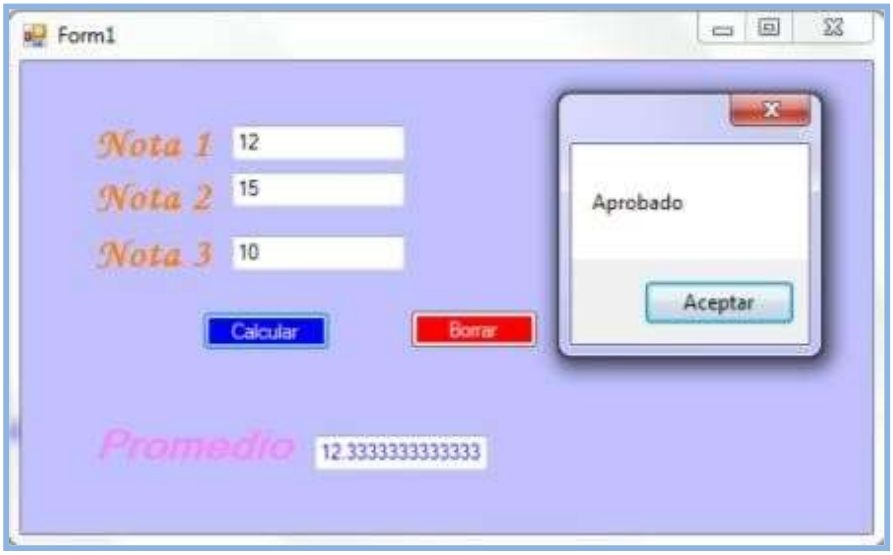

```
Form1.vb*
(General)                    (Declarations)

    Public Class Form1


        Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
            Dim vn1, vn2, vn3, vnp As Double
        'ingresar        notas
            vn1 = CDbl(TextBox1.Text)
        vn2 = CDbl(TextBox2.Text) vn3
        = CDbl(TextBox3.Text) vnp =
        (vn1 + vn2 + vn3) / 3


            TextBox5.Text =
        vnp If vnp >= 10.5
100 %
```





APLICACIÓN 2: en APÉNDICE

Síntesis

La programación orientada a objetos (POO) es un paradigma de programación más orientado a como vemos las cosas en la vida real que otros tipos de programación (programación lógica, funcional, imperativa, etc.)

Este tipo de programación permite solventar algunos de los problemas que ha tenido el desarrollo del Software desde sus comienzos, como por ejemplo la falta de portabilidad y baja reusabilidad del código, junto con la dificultad en la modificación y desarrollo del mismo. Además, es una técnica de codificación bastante más intuitiva que el resto. La programación orientada a objetos tiene tres propiedades básicas: Debe estar basado en objetos, estar basado en clases y debe de ser capaz de mantener una herencia entre clases. La mayoría de los lenguajes cumplen una o dos de estas propiedades, pero pocos logran cumplir las tres. En particular, la herencia.

Aprender esta técnica no es complicado, pero es una manera subjetiva de programar que depende del desarrollador. Aunque podemos dar diferentes soluciones a un mismo problema, no todas ellas son válidas. La dificultad no radica en aprender esta técnica, si no en ejecutarla bien. Solo programando bien podemos aprovechar todas las ventajas que nos ofrece la programación orientada a objetos.

El Visual.Net es un lenguaje de programación que actualmente respeta prácticamente todas las características de la P. O. O., anteriormente la generación de un programa ejecutable con Basic generaba un ejecutable, actualmente genera un archivo intermedio que se ejecuta posteriormente por la máquina virtual de Studio Net, esta característica, que hoy incorporan otros lenguajes de programación, convierte a VB en un lenguaje portable en el momento que exista una máquina virtual para otros sistema operativos distintos de Windows.

Como se puede ver Visual Basic está teniendo mucha trascendencia en el ámbito tecnológico y, por ende, cada vez surgen nuevos retos y oportunidades en relación a este lenguaje, los cuales, en gran parte, benefician a la productividad empresarial, donde el hombre en algunas ocasiones se vuelve cada vez más dependiente de estas aplicaciones; sin embargo, no se puede prescindir de ellas por el mismo hecho de que vivimos en un entorno globalizado, donde la tecnología es una parte sustancial de nuestras vidas.

Apreciación crítica y sugerencias

Pese a las grandes ventajas que ofrece la programación orientada a objetos gracias a características propias como la herencia y la encapsulación, aún quedan problemas por solucionar. No hay un lenguaje dominante que elimine la dependencia que existe entre unos y otros, y las definiciones de las clases pueden resultar erróneas después de meses de trabajo. Por otro lado, todos los elementos del programa deben de ser vistos como objetos y la transición a este tipo de programación hace que sea necesario el reciclaje de muchos de los programadores.

En conclusión, se puede decir que la forma de programar, ha evolucionado con el paso del tiempo, en la actualidad, existen diversos programas los cuales nos facilitan nuestras actividades que realizamos en nuestra vida diaria, es un gran trabajo el que hacen los programadores, es importante mencionar que no cualquiera puede hacer un programa, se debe tener una preparación para realizar estos.

Al exponer todos estos temas de Programación en Visual Basic nos podemos percatar que la programación es sumamente útil en nuestra vida cotidiana, desde ir a pagar el recibo de impuestos, sacar dinero del banco o incluso nuestros celulares o reproductores de música están empleados con programas unos más complejos que otros, pero accesibles para nosotros como usuarios.

Resolver un problema bajo el paradigma de la programación orientada a objetos implica determinar y caracterizar los diferentes objetos que intervienen en el problema, definir sus propiedades y métodos y ponerlos a interactuar.

La Programación Orientada a Eventos facilita el proceso de programación visual por su practicidad al generar rápidamente algunas aplicaciones basadas en objetos visuales.

Ayuda al programador novato en el rápido aprendizaje de desarrollo de nuevos programas con alta calidad en muy poco tiempo.

Se sugiere que este lenguaje de programación sea considerado como un curso obligatorio en la educación superior.

Referencias

Betancourt, G. A. (2000). *Programación estructurada antes de programación orientada a objetos*. [Documento Word]. Disponible en internet: http://www.ilustrados.com/publicaciones/EpZVVllyAyovOwMHjf.php

Franco, A. (2012). *Programación en lenguaje Java* [Sitio Web]. Disponible en Internet: https://www.ibm.com/developerworks/ssa/java/tutorials/j-nItrotojava1/index.html

Guzmán, L. (2015). *Lenguajes de programación.* [Página]. Disponible en Internet: http://www3.uji.es/~mmarques/f47/apun/node37.html

Lucas, (2006). *Programación orientada a objetos*. [Página]. Disponible en Internet: J. Arias. *Arquitectura de Software: Conceptos y Definiciones. Presentación de la clase Arquitectura de Software*. Universidad de los Andes, Bogotá – Colombia.

Marqués, M. (2008). *Lenguajes de cuarta generación*. Disponible en Internet: https://www.monografias.com/trabajos/objetos/objetos.shtml

Méndez, J. (2014). *Las tendencias de los lenguajes de programación*. [Página]. Disponible en Internet: http://es.wikipedia.org/wiki/Generaciones_de_lenguajes_de_programaci%C3%B3n

Microsoft .Net. (1995). *Programación con C# Net*, [Documento PDF]. Disponible en internet: http://ohm.utp.edu.co/gustavoa/res/Documentos/prog_estruct.doc

Pool, I. A. (1991). *Lenguaje Ensamblador*. [Página]. Disponible en Internet: P. Norton y J. Socha. Nueva Guía del programador en ensamblador para IBM PC/XT/AT y

compatibles. Anaya Multimedia, S.A. http://es.wikipedia.org/wiki/Lenguaje_m%C3%A1quina

Som, C. G. (2002).*Visual Basic.Net.(*2° ed.).Paraguay.Anaya

Wikipedia. (1999). *Generaciones de los lenguajes de programación*. [Página]. Disponible en Internet: http://www.monografias.com/trabajos26/lenguajes-programacion/lenguajes-programacion.shtml#estand

Wikipedia. (2001). *Lenguaje ensamblador*. [Página] disponible en Internet: http://www.monografias.com/trabajos/tendprog/tendprog.shtml

Wikipedia. (2001). *Lenguaje máquina*. [Página]. Disponible en Internet: http://es.wikipedia.org/wiki/Lenguaje_ensamblador

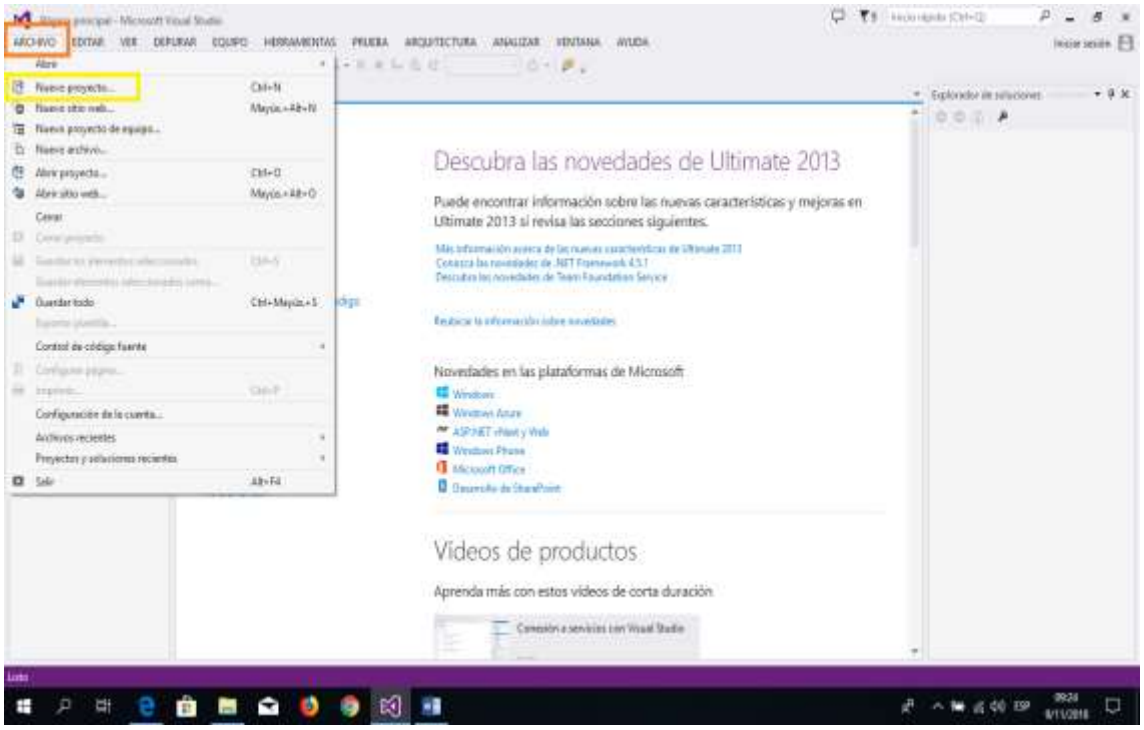
APÉNDICE

APLICACIÒN 2: Utilizando procedimientos y eventos en Visual Basic.Net se diseña una calculadora básica de cuatro operaciones

1. Crear un proyecto de aplicación Window Forms.

Click en archivo

Clic nuevo proyecto

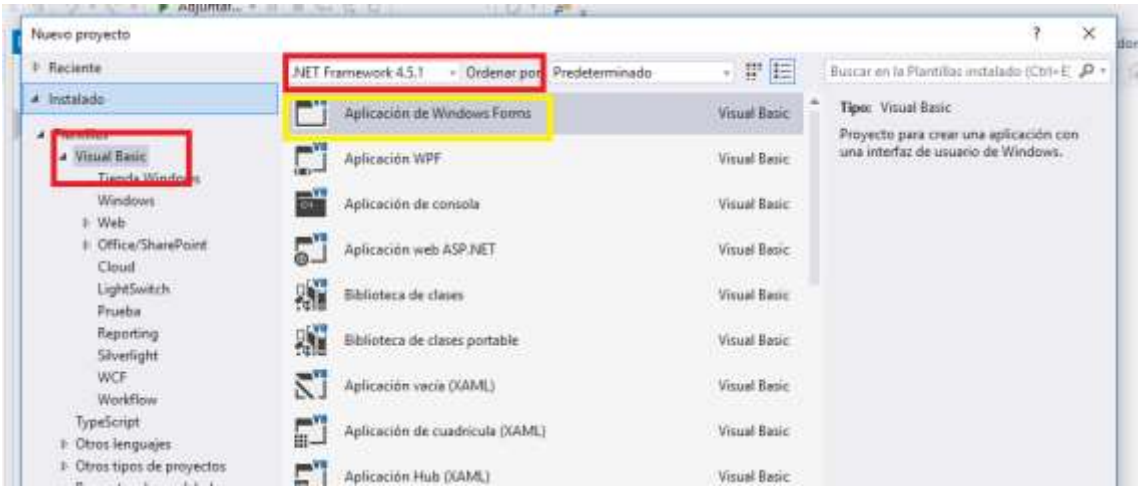


Clic en visual Basic

Clic net framework

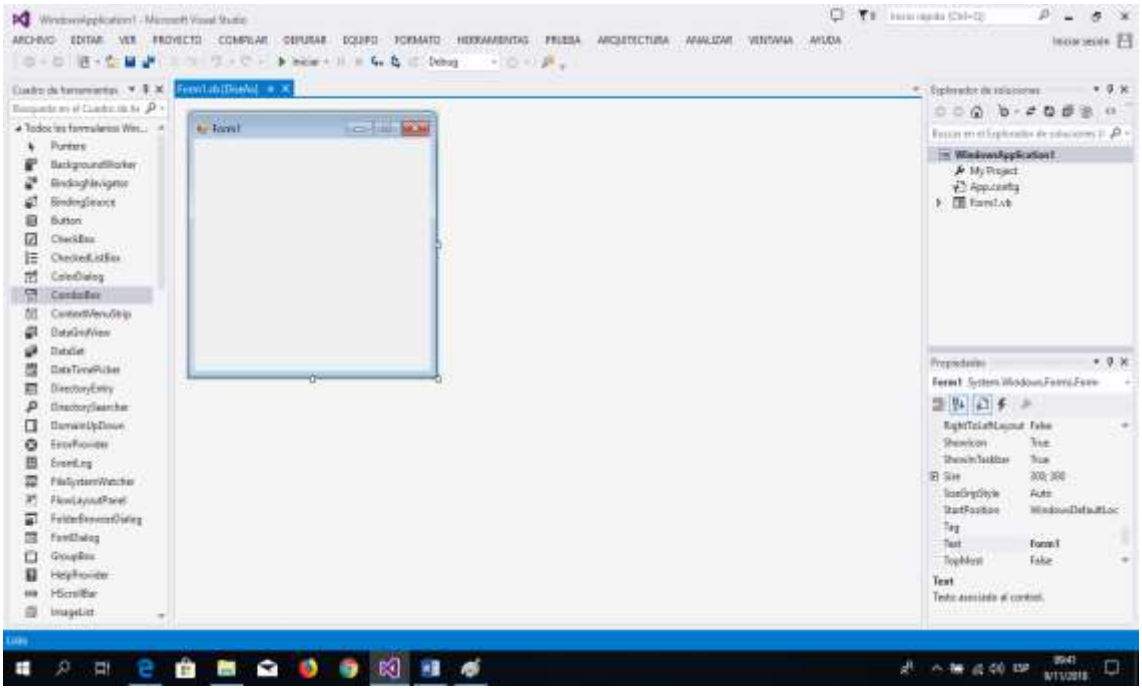
Clic en Aplicaciòn de Window Forms

Clic en aceptar



2. Entorno de Visual Basic.

Podemos observar la Form 1

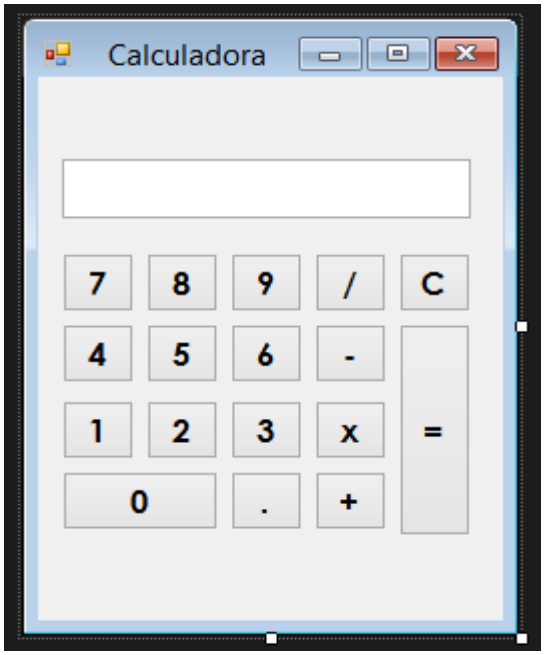


Para comenzar nuestro post de cómo **hacer** una calculadora en Visual Basic .NET, en el formulario que hemos creado destinado para nuestra calculadora, vamos

a **arrastrar** los elementos necesarios para hacer la calculadora. Básicamente son necesarios:

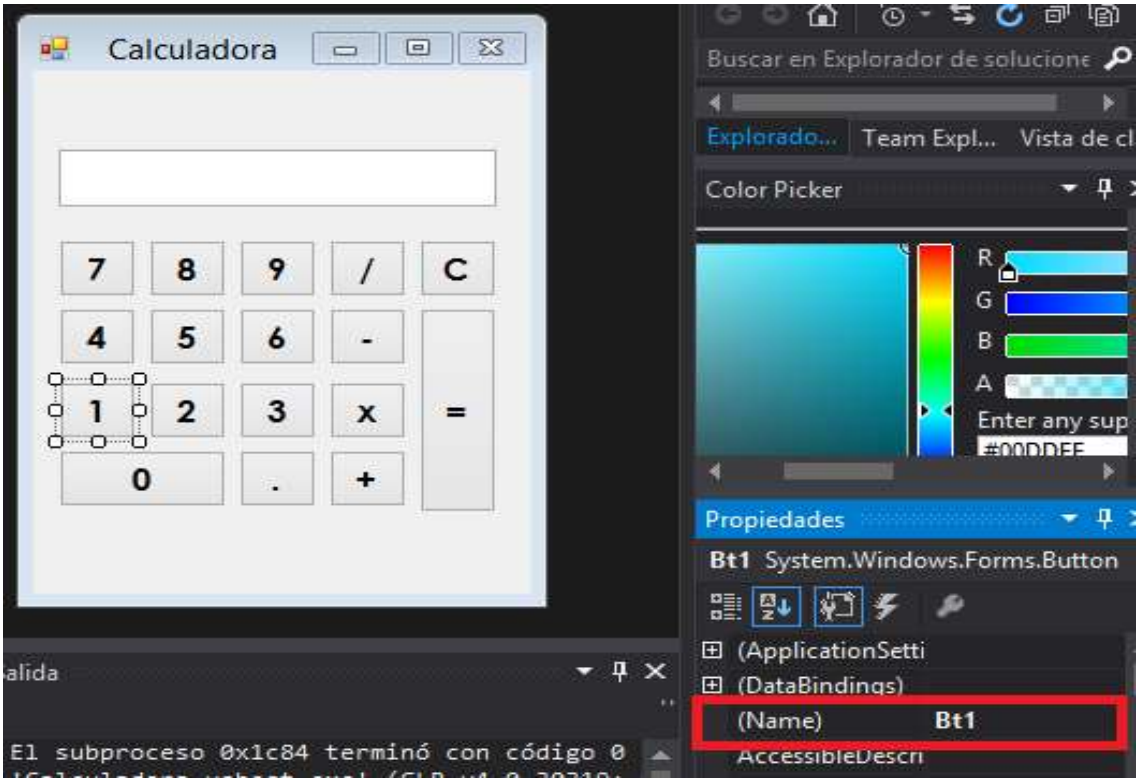
17 *"Buttons"* correspondientes a los números y a las operaciones,

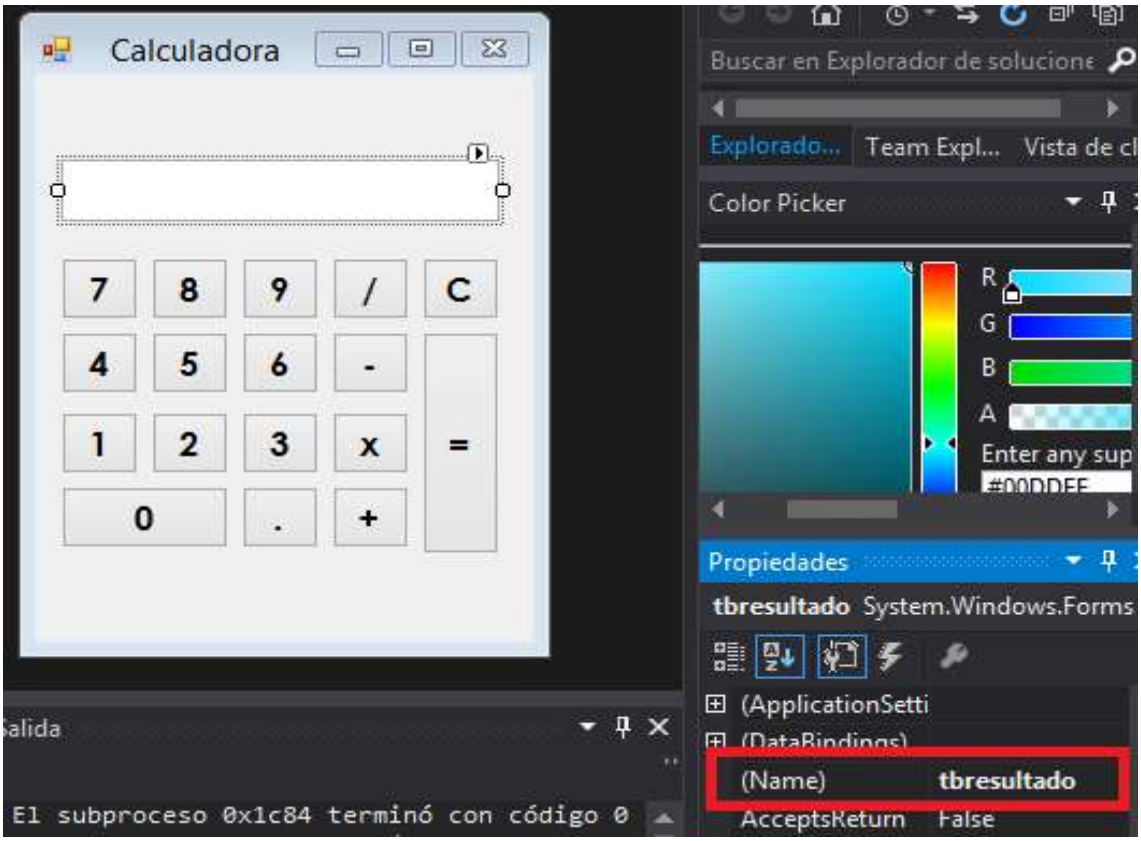
un *"TextBox". P*ara reflejar el resultado. Nos quedaría así:



3. Asignarle el "Name" a los elementos.

Una vez tengamos todos los elementos en orden, es necesario asignarles un nombre a nuestros "Buttons" y a nuestro "TextBox" por cuestiones de orden y estandarización.


Calculadora
7 8 9 / C
4 5 6 - =
1 2 3 x
0 . +
Buscar en Explorador de solucione
Explorado... Team Expl... Vista de cl
Color Picker
R
G
B
A
Enter any sup
#00DDEE
Propiedades
Bt1 System.Windows.Forms.Button
(ApplicationSetti
(DataBindings)
(Name) Bt1
AccessibleDescri
alida
El subproceso 0x1c84 terminó con código 0


Calculadora
7 8 9 / C
4 5 6 - =
1 2 3 x
0 . +
Buscar en Explorador de solucione
Explorado... Team Expl... Vista de cl
Color Picker
R
G
B
A
Enter any sup
#00DDEE
Propiedades
tbresultado System.Windows.Forms
(ApplicationSetti
(DataBindings)
(Name) tbresultado
AcceptsReturn False
alida
El subproceso 0x1c84 terminó con código 0

En nuestro caso, los *"Buttons"* que cumplen la función de los números, llevan el *"Name"* de *"bt1, bt2, btsuma, btigual ETC* ". Nuestro *"TextBox"* llevará el *"Name"* de *"tbresultado"*. Esto lo hacemos con el fin de establecer un orden y trabajar de forma correcta.

4. Declaración de las variables.

En la parte superior de nuestro código, dentro de la clase publica de nuestro formulario, vamos a declarar 4 variables públicas:

```
Public Class Form1
    Dim Operacion As String
    Dim ValorResultado As Nullable(Of Double) = Nothing
    Dim Valor2 As Nullable(Of Double) = Nothing
    Dim Bandera As Boolean
```

La variable *"Operacion"* de tipo *"String"* es la que viajará al procedimiento (que haremos en la parte final de nuestro código) que realizará la operación, esta **contendrá** el símbolo de nuestra operación (*"* + – * / *"*).

A continuación:

Se declara la variable "ValorResultado" como *"Nullable(Of Double)"* que quiere decir qué dicha variable va a aceptar valores nulos (que no tienen ningún valor) y va a ser de tipo *"Double"*. Esa variable será igual a *"Nothing* , o sea a nada.

Luego

"Valor2" será la encargada de contener el segundo valor que se introduzca. Al igual que la anterior, será igual a *"Nothing" y de tipo "Nullable(Of Double)"*.

Por último,

La variable *"Bandera"* nos servirá de bandera (*interruptor*) y será de tipo *"Boolean*"

5. *Código de los botones.*

- Botones Numéricos:

En esta parte vamos a asignarle a cada *"Button"* un código, para que nuestra calculadora pueda funcionar de forma correcta.

A los botones numéricos le vamos a asignar el siguiente código:

```
Private Sub bt1_Click(sender As Object, e As EventArgs) Handles bt1.Click
    DeterminarConcatenar()
    tbresultado.Text &= "1"
End Sub
```

```
0 referencias
Private Sub bt2_Click(sender As Object, e As EventArgs) Handles bt2.Click
    DeterminarConcatenar()
    tbresultado.Text &= "2"
End Sub
```

Esto se lo colocaremos a todos los números, así sucesivamente hasta llegar al número "0".

En el código anterior, vemos lo que se ejecutará cuando se haga click en cada "Button" numérico.

- En este caso *"DeterminarConcatenar ()"* es un procedimiento que explicaremos al final, que en pocas palabras evaluará el estado de nuestro "TextBox", limpiará y concatenará para introducir el siguiente número.
- Luego de eso (tbresultado.Text &= "1"), le especificamos la propiedad *"Text"* a nuestro *"tbresultado",* nuestro *"TextBox"* gracias al signo ampersand *("&") contendrá* la suma de todos los números introducidos anteriormente, más (*"+"*) el número *"1".* Esto lo haremos con todos los *"Buttons",* solo debemos cambiar el numero dependiendo del *"Button"* al que se le está asignando el código.

- **Botones de operaciones matemáticas:**

En esta parte, a los botones que hacen referencia a las operaciones matemáticas, se les asignará el siguiente código:

```
Private Sub btsuma_Click(sender As Object, e As EventArgs) Handles btsuma.Click
    OperacionProceso()
    Operacion = "+"
End Sub

0 referencias
Private Sub btresta_Click(sender As Object, e As EventArgs) Handles btresta.Click
    OperacionProceso()
    Operacion = "-"
End Sub
```

Se ejecutará el procedimiento *"operación proceso ()",* que es aquel que efectuará la operación dependiendo el signo. Luego, a la variable que creamos en la parte superior llamada "Operación", le asignaremos un operador que solo cambiará el signo dependiendo la operación.

Botón punto:

Este botón nos ayudará es especificar números decimales, y así obtener un resultado decimal. Su código es el siguiente:

```
DeterminarConcatenar()
 If InStr(tbresultado.Text, ".", CompareMethod.Text) = 0 Then
 tbresultado.Text &= "."
 End If
```

Este condicional se hace para poder establecer solo una vez un punto en nuestro número. Con *"InStr(tbresultado.Text, ".", CompareMethod.Text) = 0"* lo que hacemos es obtener el numero entero del texto del *"TextBox"* con nombre *"tbresultado"* y verificar el número de veces que dentro de ese número hay un punto, eso lo hacemos con el método "CompareMethod.Text". Si esa condición se cumple, entonces el texto de *"tbresultado"* será igual al contenido de "tbresultado" más un punto.

Botón borrar:

Este botón nos limpiará el *"TextBox"* con nombre *"tbresultado"*, el código es el siguiente:

```
tbresultado.Text = "0"
Valor2 = Nothing
ValorResultado = Nothing
```

Al hacer click en el este en el *"Button"* con nombre *"btborrar"*, la propiedad texto de *"tbresultado"* será igual a cero *("0").* Luego las variables "Valor2" y "ValorResultado" serán iguales a "Nothing" (Nada).

Botón igual:

Este botón nos dará el resultado de la operación. Su código es sencillo y es el siguiente:

```
OperacionProceso ()
Operacion = ""
```

Simplemente lo que hacemos es llamar al procedimiento *"operación proceso ()"* que es el que realiza la operación. Luego la variable *"Operación"* es igual a Vacío; esto lo hacemos para no entre en ningún *"Case" d*el *"Select Case"* que hay en el procedimiento anteriormente mencionado.

6. Procedimientos.

Procedimiento "Determinar Concatenar ()"

Como lo dijimos anteriormente, este procedimiento nos servirá para **limpiar** el "TextBox" que contiene el resultado final. Si se presiona un operador, es necesario que se limpie nuestra caja de texto para poder ingresar el segundo número; de esto se encarga este procedimiento. Ahora veamos el código:

```
Public Sub DeterminarConcatenar()
    If Bandera = True Then
        tbresultado.Text = ""
        Bandera = False
    ElseIf tbresultado.Text = "0" Then
        tbresultado.Text = ""
    End If
End Sub
```

Primer condicional:

```
If Bandera = True Then
 tbresultado.Text = ""
 Bandera = False
```

Vemos que está compuesto por dos condicionales. El primer *"If"* cuenta con la condición que cuando la variable *"Bandera"* sea verdadera, el *"TextBox"* del resultado se limpiará e instantáneamente recibirá el segundo número.

Luego tenemos el segundo y último condicional:

```
ElseIf tbresultado.Text = "0" Then
 tbresultado.Text = ""
 End If
```

Nuestro segundo condicional, el *"ElseIf"* nos ilustra que, cuando la caja de texto tenga únicamente el número cero (*"0"*), esta se limpie y permita introducir el siguiente número.

Procedimiento "Operación Proceso ()":

En este procedimiento se realiza la operación y enviamos el resultado a la caja de texto. Este segmento de código, está compuesto por un *"If"* y por un *"Select Case"*, los *"Case" o* casos de este *"Select Case"* reciben el signo del operador para realizar la respectiva operación. El código es el siguiente:

```
Public Sub OperacionProceso()
    Bandera = True
    Valor2 = Val(tbresultado.Text)
    If ValorResultado IsNot Nothing Then
        Select Case Operacion
            Case "+"
                ValorResultado = ValorResultado + Valor2
            Case "-"
                ValorResultado -= Valor2
            Case "*"
                ValorResultado *= Valor2
            Case "/"
                ValorResultado /= Valor2
        End Select
        tbresultado.Text = ValorResultado
    Else
        ValorResultado = Valor2
    End If
End Sub
```

En este código, cada vez que se ejecute, nuestra variable *"Bandera"* tomará un *"True" (Verdadero)* como valor. Siguiente a esto la variable *"Valor2"* toma el valor numérico que contiene el *"TextBox"* del resultado.

Luego de esto, está compuesto por dos condicionales, uno dentro de otro:

```
If ValorResultado IsNot Nothing Then
Select Case Operacion
Case "+"
ValorResultado = ValorResultado + Valor2
Case "-"
ValorResultado -= Valor2
Case "*"
ValorResultado *= Valor2
Case "/"
ValorResultado /= Valor2
End Select
tbresultado.Text = ValorResultado
Else
```

```
    ValorResultado = Valor2
    End If
```

Primero se evalúa si la variable *"Valor Resultado"* es *"IsNot Nothing"*, o sea si no se encuentra vacía, si ya tiene un muero como valor. Si esto es verdadero, el *"Select Case"* toma la variable "Operación" y comienza a evaluar. La variable *"Operación"* recibe como valor el signo de la operación que hemos presionado.

En un caso hipotético, si se presionó el operador de suma, la variable *"Operación"* tomará como valor: "+"; ese signo entraría en el primer *"Case"* y se efectuaría la suma.

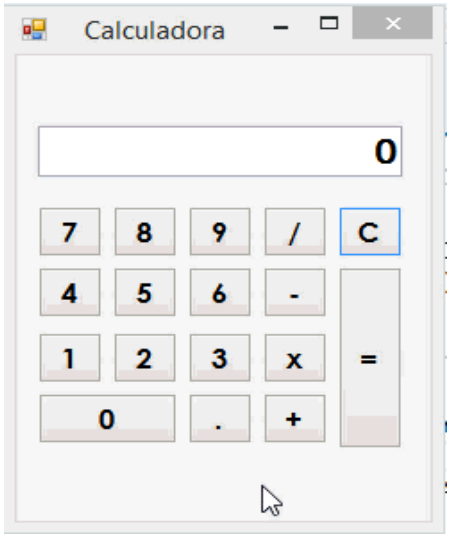
En cada "Case" la variable *"Valor Resultado"* será igual a la operación escogida entre *"Valor Resultado"* y *"Valor2"*.

Ejemplo: Si se entra en el *"Case"* de multiplicación, la variable *"Valor Resultado"* será igual a la multiplicación de ella misma por la variable *"Valor2"*.

Luego de que se haya salido del *"Case"*, se finaliza *el "Select Case"* y el *"TextBox"* será igual a la variable *"Valor Resultado"* que es la que contiene el resultado de la operación.

Posteriormente, el condicional que contiene nuestro *"Select Case"*, tiene un "Else" el cual se ejecutará si la condición *"Valor Resultado IsNot Nothing"*no se cumple; si esto pasa la variable *"Valor Resultado"* será igual a *"Valor2"*.

Resultado Final:



Y este sería el resultado final de nuestra calculadora. Espera muy pronto más posts relacionado con Visual Basic .NET.