VISUAL BASIC .net

1

Introducción

A modo de introducción, repasaremos brevemente la historia del lenguaje, desde sus comienzos, en 1991, hasta la actualidad, la versión Visual Basic .NET. Además, se presentarán los fundamentos de la programación visual, necesarios para quien nunca haya programado bajo Windows.

Historia del lenguaje

Aunque parezca aburrido, siempre es conveniente conocer la historia de las herramientas con las que trabajamos, para ubicarnos en el tiempo y saber dónde estamos situados. **Visual Basic** es un lenguaje de programación, y como tal, sirve para crear programas o aplicaciones. Un lenguaje de programación está formado por un conjunto de sentencias (comprensibles por los humanos) que representan órdenes que se le dan a la computadora.

Pero Visual Basic no es el único lenguaje de programación que existe en la actualidad, y tampoco nació de la noche a la mañana. La versión original del lenguaje **Basic** (*Beginner's All-purpose Symbolic Instruction Code*) fue creada en 1964, en *Dartmouth College*. Nació con el objetivo de servir como lenguaje para aquellas personas que deseaban introducirse por primera vez en el mundo de la programación, y luego fue sufriendo modificaciones, hasta que en 1978 se estableció el Basic estándar. Más adelante, en 1987, llegó una de las versiones más populares del lenguaje: el viejo y querido **QuickBasic**, una joya de oro de los tiempos del **MS-DOS**, con la cual muchos nos desvelamos más de una noche. Sin embargo, a la hora de programar, siempre existieron alternativas a Basic: lenguajes como **C**, **Pascal** o **COBOL** eran muy populares entre los programadores, mientras que había una especie de desprecio hacia Basic, por tratarse de un lenguaje "para principiantes".

Hasta aquí, todo eran oscuras pantallas de texto y aplicaciones de consola. Con la llegada de Windows, todo resultó muy diferente. Las antiguas aplicaciones de **MS-DOS** mucho distaban ya de las modernas y visuales de Windows, pero, claro, hacer aplicaciones para ese nuevo sistema operativo era un real privilegio, que muy pocos estudiosos de la programación en **C** podían experimentar. Afortunadamente, esto cambió en **1991**. Ese año, **Visual Basic 1.0** vio la luz (**Figura 1**), y revolucionó el desarrollo de aplicaciones para Windows, especialmente por su facilidad y por la rapidez con la que permite crearlas.

Si bien muchas personas hicieron de Visual Basic una realidad, un empleado de Microsoft, llamado Alan Cooper, fue considerado el verdadero padre del lenguaje. Hoy, con más de diez años del lenguaje en el mercado, Alan es considerado una eminencia.

BASIC

Basic es la sigla de *Beginner's All-purpose Symbolic Instruction Code*.

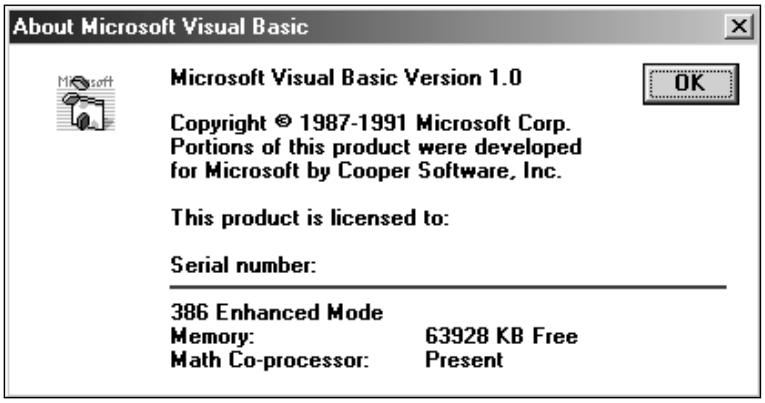

***Figura 1**. La mítica pantalla de apertura de Visual Basic 1.0. Una pieza de museo en estos días.*

Visual Basic es un lenguaje de **propósito general**: se pueden crear aplicaciones de escritorio, utilitarios, juegos, aplicaciones multimedia, sistemas con manejo de bases de datos, componentes reutilizables, aplicaciones distribuidas y mucho más. Muchos critican esta generalidad, aduciendo que sirve para tantas cosas que, al final, no cumple bien con ninguna... Qué equivocados que están.

Luego de la primera versión, el lenguaje creció y empezó a volverse muy popular. A partir de la versión 3, ya se incluían herramientas para el acceso a datos y una interfaz gráfica más cómoda e intuitiva. Después, llegó la versión 4, que podía compilar ejecutables tanto de 16 bits como de 32; y, finalmente, el exilio a los 32 bits arribó con las versiones 5 y 6. Actualmente, Visual Basic combina la sencillez de Basic con el poder de un lenguaje de programación visual que permite desarrollar robustas aplicaciones de 32 bits. Visual Basic ya no es sólo un lenguaje para los más novatos, sino que representa una excelente alternativa para programadores de todos los niveles.

VB.NET, un cambio profundo

El lenguaje siguió evolucionando hasta que, el **13 de febrero de 2002**, justo un día antes de San Valentín, nació oficialmente **Visual Basic .NET**, junto con el resto de la familia de **Visual Studio .NET**.

¿Simplemente una nueva versión? Para nada. **VB.NET** no es un simple upgrade; es un cambio realmente profundo y radical, que lo convierte en uno de los lenguajes más poderosos de la actualidad, con características avanzadas, como verdadera orientación a

objetos, *multi-threading*, y la posibilidad de crear *Web Services*, por nombrar sólo tres aspectos. Todos aquellos que todavía tengan la idea (equivocada) de que Visual Basic es un lenguaje para principiantes, finalmente tendrán que callar y agachar sus cabezas.
Obviamente, no es un cambio que se produjo de la noche a la mañana. La **plataforma .NET**, base de este nuevo lenguaje, se viene gestando en **Microsoft** desde hace ya un par de años, y forma parte de una nueva estrategia impulsada por esta empresa para conquistar el mercado del desarrollo y de Internet, y seguir creciendo.

Hasta dónde se puede llegar con VB.NET

Visual Basic .NET es la versión más poderosa de este lenguaje jamás concebida. Muchas barreras se han abierto, y las posibilidades se expanden. Sólo por mencionar algunas, podríamos citar la siguiente lista:

- Aplicaciones y utilitarios de cualquier índole para Windows.
- Sistemas que manejen bases de datos de cualquier tamaño y envergadura.
- Aplicaciones multimedia, con imágenes, gráficos y sonidos.
- Juegos de mediana envergadura.
- Sistemas distribuidos.
- Componentes reutilizables.
- Servicios web aptos para ser usados a través de Internet en un santiamén.

Y la lista es casi infinita. Con Visual Basic .NET, se puede llegar tan lejos como la imaginación lo permita. Éste es un lenguaje tan amplio y expansible, que muchas veces le otorga al programador el incomparable placer de experimentar soluciones ingeniosas a problemas nuevos, basándose en combinaciones de técnicas sumamente sencillas.

Pero ¿cuál es el límite de este lenguaje? Hoy en día, Visual Basic .NET no permite, o no es la mejor opción para:

- Desarrollar drivers para Windows o librerías de bajo nivel.
- Crear juegos comerciales de última generación. Si bien es posible utilizar las libre-

VISUAL BASIC, UN LENGUAJE EN SERIO

Desafortunadamente, muchas personas tienen la idea equivocada de que Visual Basic es un lenguaje poco serio, de juguete, quizás. La realidad indica todo lo contrario: si se lo sabe explotar, Visual Basic es un lenguaje realmente poderoso.

rías de DirectX en VB.NET, la lógica y el cálculo intensos de este tipo de juegos requieren lenguajes como Visual C++, que puedan salir de la plataforma .NET sobre la cual corre Visual Basic, para correr rápidamente en un nivel más bajo.

Como habrá deducido, Visual Basic .NET es una excelente opción, y si todavía está leyendo, es porque seguramente ya ha decidido incursionar en él, así que basta de vueltas, y comencemos.

Fundamentos de la programación visual

Si ha tenido la oportunidad de programar bajo DOS, habrá notado que no es tarea sencilla. La principal dificultad para los programadores era lograr una buena interfaz gráfica. Además, los usuarios tenían una complicación adicional: los programas no eran parecidos entre sí, ni en su estética ni en su operatividad.

Con la llegada de Windows, esto cambió radicalmente. Gracias a la programación visual, los programadores cuentan con la posibilidad de diseñar interfaces más amigables, y a los usuarios les resulta más transparente el paso de una aplicación a otra, ya que la mayoría de los programas poseen elementos comunes (menúes, cajas de texto, etc.).

Pero no sólo cambió la parte estética de los programas, sino también la forma de construirlos. Al programar para DOS, había que seguir un esquema bien definido: todas las aplicaciones comenzaban en cierto punto y finalizaban en otro punto. Era tarea del programador definir todo lo que ocurría en el medio, y ocuparse de la entrada, la validación y la salida de datos, así como de establecer todo lo que el usuario podría hacer durante la ejecución.

Afortunadamente, programar para Windows con Visual Basic es totalmente distinto:

- La primera tarea que se debe realizar es diseñar los **formularios** (ventanas) con los que interaccionará el usuario.
- Luego, deben agregarse objetos (**controles**) a esta ventana, y definir las **propiedades** de cada uno.
- Finalmente, se deben establecer los **eventos** a los que va a responder cada uno de los objetos de estas ventanas, y escribir el **código fuente** que determine cómo se comportarán (por ejemplo, definir cómo va a responder un botón cuando el usuario haga un clic sobre él).

Cuando pienso en programación visual, siempre me viene a la mente una imagen, que trata de ilustrar cómo se diseña una aplicación. En la **Figura 2** puede ver la idea. Lo que intenta representar es que cada objeto (control) en la ventana (formulario) tiene un código asociado que determina cómo se comporta ante determinado suceso (el clic del mouse, por ejemplo).

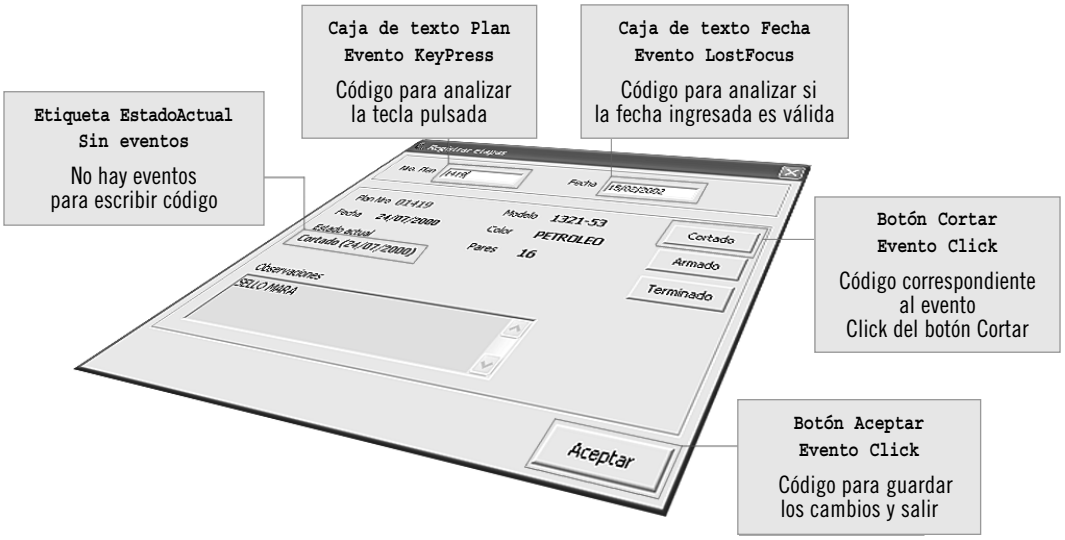

***Figura 2**. Un intento de ilustrar lo que es la programación visual.*

En este esquema, mientras el usuario o el sistema no dispare ningún evento (supongamos que no hizo acciones), ningún código se ejecutará. Cuando el usuario hace clic sobre un botón, por ejemplo, dispara el evento **Click**, y entonces su código asociado se ejecuta.

Formularios

Los formularios son las ventanas mediante las cuales los usuarios interaccionan con la aplicación. Lo primero que se debe hacer al comenzar un nuevo proyecto es definir todas las ventanas que formarán parte del programa, y luego establecer su

NO CONFUNDIR

Aunque muchas veces se utilicen ambos términos como sinónimos, no hay que confundir el significado de controles con el de objetos. Un control es un objeto que se inserta en un formulario o ventana (por lo tanto, todo control es un objeto). Un objeto no siempre es un control, ya que puede estar formado simplemente por código.

apariencia y su comportamiento. En general, se trabaja con los formularios en tiempo de diseño, es decir, no bien se comienza a modelar el programa. En tiempo de ejecución, lo que se suele hacer es ocultar o redefinir el tamaño de un formulario.

Controles

Los controles son todos los objetos que se insertan en los formularios, mediante los cuales los usuarios realizan sus acciones visualmente (interaccionan con ellos). A través de los controles se puede mostrar texto (etiquetas), recibir texto (TextBoxs, o cajas de texto), usar botones, insertar y manipular imágenes, mostrar almanaques, listados y mucho más. Visual Basic .NET incluye muchos controles listos para usar, pero lo más llamativo es la posibilidad de crear controles personalizados.

Propiedades

Las propiedades son atributos que tienen los objetos. Por ejemplo, los formularios poseen dos propiedades, llamadas **Width** y **Height**, que determinan su ancho y su altura, respectivamente. Cada objeto posee una lista particular de propiedades, que difiere del resto. En **tiempo de diseño** se establecen las propiedades que tendrán los controles al iniciarse el programa, y luego, en **tiempo de ejecución**, se puede escribir código para modificar sus valores.

Eventos

Los eventos son sucesos a los que debe responder el programa. Gran parte del código que hay que escribir en Visual Basic se debe a procedimientos de suceso, que definen cómo va a actuar el programa ante un evento. Por ejemplo, los botones de comando tienen un evento llamado **Click**, que se dispara cada vez que el usuario hace un clic del mouse sobre él. Si queremos que se muestre un mensaje cuando el usuario haga clic sobre ese botón, tendremos que escribir código en su evento.

PROPIEDADES

Las propiedades de un objeto reflejan una característica de éste; por ejemplo, su ancho (Width). Generalmente, el valor de las propiedades se puede leer y escribir, aunque también existen propiedades de sólo lectura (Read-Only), cuyos valores pueden ser leídos, pero no modificados.

Métodos

Los métodos son funciones propias de cada objeto. Así como las propiedades determinan cómo son los objetos, los métodos ejecutan acciones propias de éstos. Los métodos sólo se utilizan en tiempo de ejecución (a diferencia de las propiedades, que también se pueden establecer en tiempo de diseño). Por ejemplo, los formularios poseen un método llamado **Hide**, que se encarga de ocultarlos de la pantalla; es decir, **realiza una acción** (los oculta).

El código fuente

El código es donde se vuelca toda la información, y allí es donde más se notan las diferencias entre distintos lenguajes. Cada uno tiene su sintaxis, su forma de declarar variables, de hacer ciclos y estructuras de decisión, de trabajar con objetos, etc. En el caso de Visual Basic, afortunadamente el lenguaje es muy ameno, ya que se asemeja bastante al humano. Otros lenguajes como **C** son muy poderosos, pero cuentan con sintaxis más complicadas de escribir, de leer y de interpretar.

SINTAXIS DEL LENGUAJE

Si bien hay muchos aspectos nuevos en Visual Basic .NET, la sintaxis del lenguaje no ha cambiado radicalmente, y por lo tanto el programador de Visual Basic 6 no tendrá problemas en adaptarse a ella.