Interacción Persona Ordenador

Visual Basic

Tema 1

Introducción

¿Qué es Visual Basic?

- La Palabra "***Visual***" hace referencia al método que se utiliza para crear la **I**nterfaz **G**ráfica del **U**suario (GUI). En lugar de escribir numerosas líneas de código para describir la aparencia y la ubicación de los elementos del interfaz, simplemente pueden agregarse objetos prefabricados, en su lugar, dentro de la pantalla.
- La palabra "***Basic***" hace referencia la lenguaje BASIC (***B****eginners* ***A****ll-purpose* ***S****ymbolic* ***I****nstruction* ***C****ode* - Código de Instrucciones Simbólicas de Propósito General para Principiantes), un lenguaje utilizado por más programadores que ningún otro lenguaje en la historia de la informática o la computación.
- Visual Basic dispone de las siguientes peculiaridades:
  - Una biblioteca de clases que da soporte a los objetos Windows tales como ventanas, cajas de diálogo, controles (por ejemplo, etiquetas, cajas de texto, botones de pulsación, etc.).
  - Un control que permite utilizar las cajas de diálogo más comúnmente utilizadas (abrir, guardar como, imprimir, color y fuentes).
  - Un entorno de desarrollo integrador (editor de texto, intérprete, depurador, examinador de objetos, explorador de proyectos, compilador, etc.). Visual Basic fue diseñado para ser un intérprete, lo que favorece la creación y la depuración de una aplicación, y a partir de la versión 5 incluyó también un compilador que permite generar ficheros ***.exe*** favoreciendo así la ejecución. En edición, puede crear también ficheros ***.ocx*** y ***.dll***, lo que permitirá manipular controles ***ActiveX***.
  - El editor de textos nos ayuda a completar cada una de las sentencias visualizando la sintaxis correspondiente a las mismas.

Visual Basic

- Asistentes para el desarrollo de aplicaciones; asistente para aplicaciones, asistente para barra de herramientas, asistente para formularios de datos, asistente para empaquetado y distribución, asistente para crear la interfaz pública de controles *ActiveX*, asistente para páginas de propiedades, asistente para objetos de datos, generador de clases, diseñador de complementos y asistente para migración de documentos *ActiveX*.
- Galería de objetos incrustados y vinculados (OLE – *Object Linking and Embedding*). Esto es, software autocontenido en pequeñas y potentes unidades o *componentes software* para reutilizar en cualquier aplicación.
- Visualización y manipulación de datos de otras aplicaciones Windows utilizando controles OLE.
- Una interfaz para múltiples documentos (MDI – *Multiple Document Interface*) que permite crear una aplicación con una ventana principal y múltiples ventanas de documento. Un ejemplo de este tipo de aplicación es *Microsoft Word*.
- Editar y continuar. Durante una sesión de depuración, se pueden realizar modificaciones en el código de la aplicación sin tener que salir de dicha sesión.
- Creación y utilización de bibliotecas dinámicas (DLL – *Dynamic Link Libraries*).
- Soporte para la programación de aplicaciones para Internet; forma parte de este soporte la tecnología de componentes activos (*ActiveX*).
- Soporte para el estándar COM (*Component Object Model* – Modelo de Objeto Componente; en otras palabras, Componente Software) al que pertenecen los componentes activos (*ActiveX* o formalmente controles OLE).
- Acceso a bases de datos a través del control de datos ADO, utilizando el motor de ***Access*** o controladores ODBC.
- Acceso a bases de datos utilizando OLE DB como proveedor de datos y objetos ADO (*ActiveX Data Objects* – Objetos *ActiveX* para acceso a datos), como tecnología de acceso a datos, para satisfacer los nuevos escenarios demandados por las empresas.

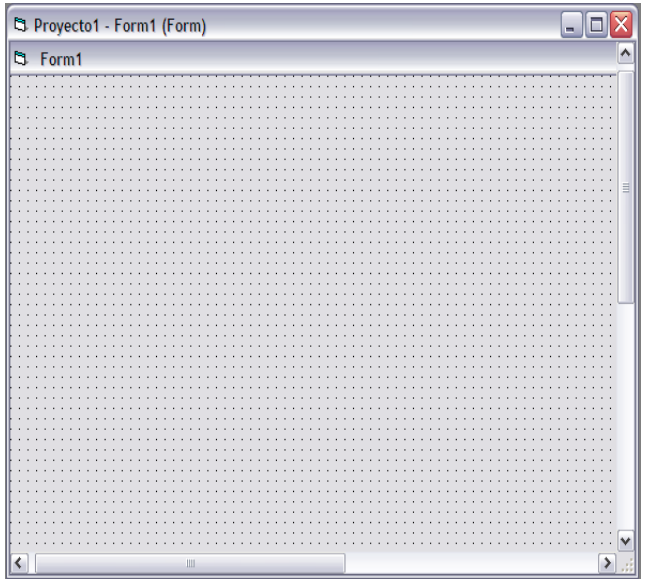
Visual Basic

- Biblioteca para SQL que permite manipular bases de datos relacionales, tales como *Microsoft Access* (SQL – *Structured Query Languaje*).
- Un *administrador visual de datos* para manipular bases de datos.
- Un programa para añadir ayuda en línea; esta herramienta permite la creación de ficheros de ayuda estilo *Windows* (*hcw.exe* – *Help Workshop*).
- **Algunas de estas características sólo están disponibles en las versiones profesional y empresarial.**

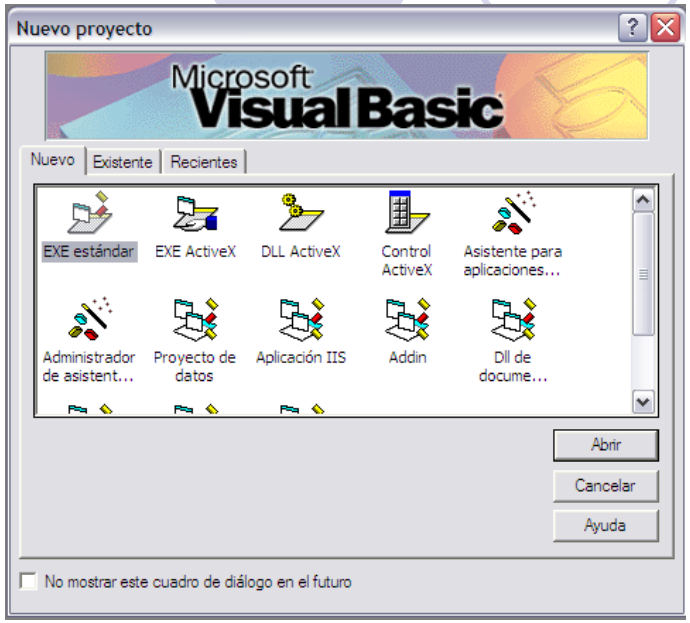
Visual Basic

- A diferencia de lo que ocurre en los intérpretes y compiladores de BASIC tradicionales, en los cuales una variable declarada en un punto del programa es accesible desde cualquier otro punto, en *Visual Basic* las variables pueden ser globales, accesibles desde cualquier punto, públicas en un módulo, accesible desde los métodos y funciones contenidas en un módulo de código, o privadas, accesibles tan sólo en el ámbito en el que se ha declarado.
- Es una práctica común que todas las variables que se utilizan en métodos y funciones sean privadas, evitando así que puedan ser manipuladas desde cualquier código externo. Este tipo de variables son creadas automáticamente cada vez que se ejecuta el procedimiento o función, y destruidas cuando éste termina.
- Sólo se necesitan unos minutos para crear una aplicación con Visual Basic. Podemos crear una Interfaz de Usuario “dibujando” controles, como cuadros de texto y botones de comando en un formulario. A continuación, estableceremos las propiedades del formulario y los controles para especificar valores como el título, el color y el tamaño. Finalmente, escribiremos el código para dar vida a la aplicación.

Visual Basic



- El **Formulario** en Visual Basic es el punto central de cualquier aplicación, pudiendo ésta contar con uno o más formularios, según las necesidades. Un formulario es una ventana Windows en la que depositaremos los controles necesarios para crear una interfaz con el usuario de la aplicación.
- El Formulario cuenta, por defecto, con los elementos habituales de cualquier ventana Windows (botones de maximizar y minimizar, borde para redimensionar la ventana, etc).
- Las propiedades que por defecto tiene un Formulario pueden ser modificadas.
- Un formulario por sí solo tiene poca utilidad, a no ser que en su interior insertemos controles.

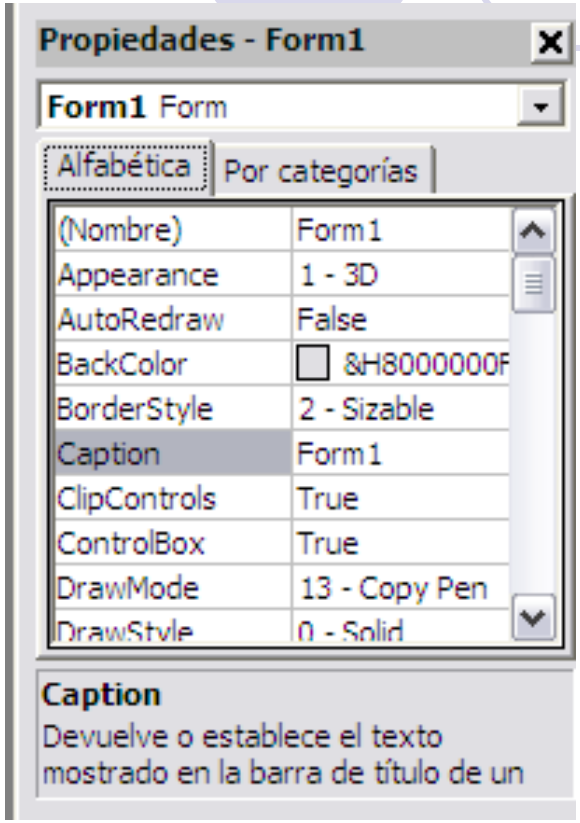


- La ventana **Nuevo Proyecto** permite seleccionar el tipo de proyecto que se desea crear:
  - *EXE estándar*. Este tipo de aplicación se construye a partir de uno ó más formularios, módulos y clases.
  - *EXE ActiveX*. Crea un componente *ActiveX* (fichero *.exe*). Un componente *ActiveX* es una unidad de código ejecutable, como un fichero *.exe*, *.dll* u *.ocx*, que sigue la especificación *ActiveX* para proporcionar código reutilizable en forma de objetos.
  - *DLL ActiveX*. Crea un componente *ActiveX* (fichero *.dll*). Los componentes proporcionan código reutilizable en forma de objetos.
  - *Control ActiveX*. Crea un control *ActiveX*. Los controles no son simplemente código, sino que tienen componente visuales como los formularios, aunque a diferencia de éstos, no pueden existir sin algún tipo de contenedor.

Visual Basic

- *EXE de documento ActiveX*. Se trata de un formulario que puede aparecer en un explorador Web.
- *Aplicación IIS (Internet Information Server)*. Se trata de una aplicación Visual Basic hecha para residir en un servidor *Web* y responder a peticiones enviadas por un explorador.
- *Aplicación DHTML*. Se trata de una o más páginas de código HTML que utilizan código Visual Basic y el modelo de objetos HTML dinámico para responder instantáneamente a las acciones que se producen en dichas páginas.
- *Asistente para aplicaciones de VB*. Genera una aplicación nueva completamente funcional desde la cual se puede generar una aplicación más compleja.
- Todos los proyectos con las extensiones *.vbp* (fichero de proyecto), *.vbz* (ficheros de asistente) o *.vbg* (grupo de proyectos) que están en el directorio …\Template\Proyects de Visual Basic.

Visual Basic



- Todos los objetos con los que trabajemos en Visual Basic, desde el propio formulario hasta cada no de los controles que podamos utilizar, tienen una serie de **Propiedades**. Estas Propiedades son las que realmente definen al objeto, indicando su posición, color, aspecto, título, valor, etc. El trabajo de diseño en Visual Basic consiste básicamente en insertar objetos en un formulario y establecer cada una de sus propiedades.
- Hay propiedades que sólo pueden ser establecidas o modificadas durante el diseño de la aplicación, otras a las que solo se puede acceder mientras la aplicación se está ejecutando, y por último hay un tercer grupo que podemos utilizar tanto en tiempo de diseño como de ejecución.

Visual Basic

- Windows es un entorno gestionado por **Eventos**, que son generados por una acción exterior por parte del usuario, como el movimiento del ratón, la pulsación de una tecla o de uno de los botones del ratón, o bien por el propio Windows.
- Cada uno de los controles que podemos insertar en el formulario, incluido el propio formulario, puede recibir una serie de eventos, en unos casos comunes y otros específicos de cada control. Por defecto Visual Basic no asocia función alguna a los controles para que respondan a estos eventos, existe una serie de procedimientos de respuesta, llamados **Métodos**, que en principio están vacíos y a los cuales podemos asociar el código que necesitemos.
- Aunque cada uno de los controles que incorpora Visual Basic es capaz de responder a multitud de eventos, por regla general sólo se definen los métodos de alguno de ellos.

Visual Basic

- En las aplicaciones tradicionales o por “procedimientos”, la aplicación es la que controla qué partes de código y en qué secuencia se ejecutan. La ejecución comienza con la primera línea de código y continúa con una ruta predefinida a través de la aplicación.
- En una aplicación controlada por Eventos, el código no sigue una ruta predeterminada; ejecuta distintas secciones de código como respuesta a los eventos. Los eventos pueden desencadenarse por acciones del usuario, por mensajes del sistema o de otras aplicaciones o incluso por la propia aplicación. La secuencia de estos eventos determina la secuencia en la que se ejecuta el código.

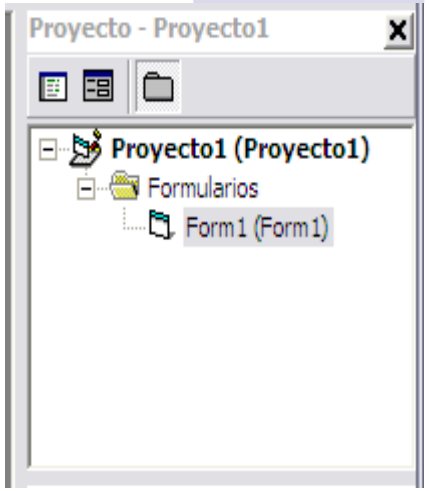
Visual Basic

```
Private Sub Form_Load()

End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)

End Sub
```

- Además del código contenido en los métodos, que es ejecutado en respuesta a un evento, en ocasiones una aplicación necesita realizar otro tipo de operaciones que por claridad y facilidad de mantenimiento no es adecuado incluir en los propios métodos. En estas ocasiones lo que se hace es crear un módulo de código, e incluir en él el código que necesitemos estructurado en Procedimientos y Funciones.
- Un **Procedimiento** es un conjunto de líneas de código al que se da un nombre, que se utiliza posteriormente para ejecutar este conjunto de líneas de código. La misma definición es válida para una **Función**, con la única diferencia ésta devuelve un valor al finalizar su ejecución, mientras que un Procedimiento no.

- Tanto los procedimientos como las funciones pueden recibir parámetros, lo que facilita el paso de datos sin necesidad de tener que utilizar variables de ámbito global. Estos parámetros pueden ser pasado por **Valor** o por **Referencia**. En el primer caso el procedimiento o función recibe una copia de la variable que se pasa como parámetro, por lo que no puede modificar la variable original, mientras en el segundo caso lo que recibe realmente es la dirección de la variable, de tal forma que cualquier modificación que se efectúe sobre ella hará que al volver del procedimiento o función, el código que realizó la llamada encuentre la variable modificada.
- Dentro de una misma aplicación podemos tener **Múltiples Módulos de Código**, conteniendo cada uno de ellos declaraciones, funciones y procedimientos. Cuando se da este caso, un procedimiento o función que se encuentre en un módulo no puede llamar a otro procedimiento o función de un módulo distinto a no ser que éste último sea público. De forma similar a lo que ocurre con las variables, tanto los procedimientos como las funciones pueden ser **Públicos**, utilizables desde cualquier otro punto de la aplicación, o **Privados**, pudiéndose llamar tan sólo desde el módulo de código en el que están definidos.

Visual Basic



- Al conjunto de formularios, conteniendo controles y métodos, módulos de código, con definición de procedimientos y funciones, y cualquier otro archivo necesario para nuestra aplicación, es a lo que se denomina **Proyecto**.
- El Proyecto establece, por lo tanto, todos los elementos que forman nuestra aplicación. Aparece como una ventana con una lista, en la que podremos ver los archivos que lo forman, cada uno con el nombre de formulario o módulo correspondiente.
- Desde la ventana de proyecto podemos seleccionar cualquiera de los componentes de la aplicación haciendo doble clic.

- **Ficheros varios, de diseño y ejecución.**

Extensión	Descripción
.bas	Módulo estándar de Visual Basic
.cls	Módulo de clase
.ctl	Código de un control *ActiveX* creado por el usuario.
.ctx	Fichero binario de un control *ActiveX* creado por el usuario
.dca	Caché del diseñador activo
.ddf	Fichero de información asociado a un fichero .cab generado por el *asistente de empaquetado y distribución*
.dep	Fichero de dependencias del *asistente de empaquetado y distribución*
.dll	Componente *ActiveX* en un proceso
.dob	Fichero de formulario de un documento *ActiveX*
.dox	Fichero binario de formulario de un documento *ActiveX*
.dsr	Fichero de un diseñador
.dsx	Fichero binario de un diseñador
.frm	Fichero de formulario
.frx	Fichero binario de formulario

- **Ficheros varios, de diseño y ejecución.**

Extensión	Descripción
.exe	Fichero ejecutable o componente *ActiveX*
.log	Fichero de registro de errores de carga
.oca	Fichero de caché de biblioteca de tipos de controles
.ocx	Control *ActiveX*
.pag	Fichero de página de propiedades
.pgx	Fichero binario de página de propiedades
.res	Fichero de recursos
.tbl	Biblioteca de tipos de automatización remota
.vbd	Fichero de estado de documento ActiveX
.vbg	Fichero de grupo de proyectos Visual Basic
.vbl	Fichero de control de licencia
.vbp	Fichero de proyecto de Visual Basic
.vbr	Fichero de registro de automatización remota
.vbw	Fichero de espacio de trabajo de un proyecto Visual Basic
.vbz	Fichero de inicio del asistente
.wct	Plantilla HTML de una clase Web (*WebClass*)

Visual Basic

- Desde siempre se ha diferenciado entre los lenguajes que utilizan **Programación Estructurada**, tipo Pascal o C, y aquellos que no lo eran, como el BASIC. Este último adolecía de las estructuras necesarias para distribuir el código de una forma clara, siendo necesario realizar saltos continuamente (lo peor, sin duda, de este lenguaje).
- Visual Basic, por el contrario, cuenta con múltiples estructuras de control que nos permiten llevar a cabo la ejecución de una aplicación completa sin la necesidad de realizar ni un solo salto. Ya hemos visto que podemos crear procedimientos y funciones, eliminando así la necesidad de utilizar uno de los saltos de BASIC, el de la instrucción GOSUB. Asimismo, contamos con varios tipos de bucles y estructuras de decisión múltiple que destierran de una vez por todas el repetitivo GOTO.
- A pesar de lo dicho antes, las instrucciones GOTO y GOSUB siguen existiendo en Visual Basic aunque su uso no es aconsejable.

Visual Basic

- Aunque Visual Basic **no** es completamente un lenguaje orientado a objetos, sí que tiene algunas características de estos lenguajes. Entre estas características está la de poder definir **Clases de Objetos**, estableciendo qué propiedades y métodos tendrán.
- Para definir una **Clase** insertaremos en nuestro proyecto un **Módulo de Clase**, en el que especificaremos cada una de sus propiedades, escribiendo el código que sea necesario para poder obtenerlas y modificarlas, así como todas las declaraciones de sus métodos. En cada archivo de módulo de clase tan sólo se puede definir una clase, pero sí es posible incluir varios módulos de este tipo en un mismo proyecto. Una **Clase** es, en cierta forma, como un plantilla del objeto que deseamos crear. En realidad al definir una clase no se está creando un **Objeto**, pero se están poniendo las bases para ello. Cuando en nuestra aplicación necesitemos un objeto de esta clase lo crearemos, y en ese momento se tomará la definición efectuada previamente para establecer las propiedades y métodos del nuevo objeto, a los que podremos acceder como si se tratase de cualquier otro control de Visual Basic.

- ¿Qué diferencia hay entre crear una Clase y definir un Procedimiento o una Función? Básicamente una, un Procedimiento es un conjunto de líneas de código que podemos ejecutar y no cuenta con propiedades ni con métodos, mientras que un Objeto encapsula en su interior todo lo necesario para su funcionamiento, no sólo código, también todas aquellas variables e incluso otros objetos que son necesarios. El Objeto no se ejecuta sin más, como un procedimiento, sino que se crean, se establecen y obtienen propiedades, y se utilizan sus métodos. Es una entidad muy superior a un Procedimiento o Función.

Visual Basic

- La creación de aplicaciones en Visual Basic se basa principalmente en el uso de distintos **Controles**. Algunos de ellos nos sirven para solicitar o presentar información al usuario, otros para generar un evento cada cierto tiempo, otros para buscar un archivo en el disco, otros para contener una imagen, etc. Estos controles se denominan tradicionalmente VB*x* (16 bits) o OLE (16 y 32 bits – se almacenan en archivos .OCX).