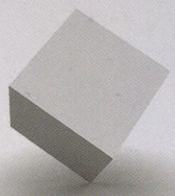


Recursosinformáticos

VBA Access (versiones 2019 y Office 365)

Programar en Access

Descarga

bases de datos

Jean-Philippe ANDRÉ

eni

VBA Access (versiones 2019 y Office 365)

Programar en Access

Este libro sobre **VBA Access (versión 2019 y Office 365)**, se dirige tanto a las personas con conocimientos de Access, como a los **desarrolladores principiantes** o **aquellos con más experiencia**. Cada uno de ellos encontrará la información necesaria para transformar sus herramientas "caseras", en una aplicación robusta o para **personalizar y optimizar de manera gráfica** las soluciones existentes. Conocer el funcionamiento de Access y utilizarlo de manera habitual, es un requisito previo imprescindible para obtener el máximo beneficio de este libro.

En este libro se tratan tanto los aspectos **básicos de VBA, como el uso de APIs externos**. El autor cubre los diferentes aspectos de la programación en Visual Basic. Este desarrollo, que tiene una dificultad progresiva, permite a los lectores reconocer la necesidad de usar este lenguaje, para entregar soluciones potentes y eficaces. Después de **la sintaxis básica en VBA**, el autor trata la noción de **programación orientada a objetos**, para más adelante seguir con los modelos **de acceso a los datos DAO y ADO**, el **lenguaje SQL** aplicado a Access, los **eventos Access**, las **interfaces** de usuario y cómo optimizarlas con la **cinta de opciones de Access**, el control de otras aplicaciones Office con **la automatización**, las **interacciones con Internet** o la **programación con Windows**. Al final del libro, se ofrece una mini-aplicación.

Las bases de datos de ejemplo para implementar los diferentes aspectos que se tratan en el libro, están disponibles para su descarga en esta página.

Jean-Philippe ANDRÉ

Jean-Philippe ANDRÉ ha sido desarrollador y consultor senior de las herramientas de la suite de Microsoft durante muchos años. Jean-Philippe ANDRÉ interviene en la actualidad en el apoyo a los desarrolladores en Quebec, especializado en tecnologías Microsoft. Ha sido profesor durante 10 años en las escuelas de ingenieros y en la universidad, y en este libro transmite toda su experiencia técnica y pedagógica, para permitir al lector dominar la potencia del lenguaje VBA, aplicado a las soluciones de gestión de bases de datos Access 2019.

Introducción

Microsoft Access 2019 es la herramienta de administración de base de datos del paquete Microsoft Office. El lenguaje nativo es Visual Basic for Application (VBA) y permite enriquecer las posibilidades ya presentes, para aprovechar mejor las capacidades de la herramienta.

El lenguaje VBA es común a varias aplicaciones del paquete Office, pero tiene aspectos específicos para Access, que es importante entender para poder asimilarlos. El lenguaje es muy potente y simple de entender, aunque llevará tiempo controlarlo.

Objetivos

Este libro puede ser útil tanto a desarrolladores experimentados como a usuarios que deseen iniciarse en el lenguaje y crear aplicaciones en Microsoft Access 2019.

El objetivo de este libro es presentar la gama de soluciones que ofrece el lenguaje VBA, aplicado a Microsoft Access. Cada capítulo dispone de una base de datos de ejemplo, a la que se puede acceder desde el sitio web de Ediciones ENI.

No es un requisito previo tener un conocimiento importante del lenguaje VBA, aunque se recomienda conocer la aplicación Access, ya se trate de la versión 2019 o una anterior, suscribiéndose a Office 365 o realizando una compra definitiva.

Al inicio se mencionará el entorno de desarrollo de VBA y después las nociones básicas de VBA, avanzando poco a poco en VBA orientado a la aplicación Access. El libro aborda estos temas por etapas, empezando por las más sencillas, hasta llegar a la personalización más potente de la aplicación y a la llamada a librerías externas, dentro de VBA. Estas diferentes etapas son las siguientes:

- Los fundamentos del lenguaje VBA, común a las diferentes aplicaciones de Microsoft Office, con variables, bucles, pruebas e interacciones básicas con el usuario.
- Los objetos en VBA y la manera de utilizarlos en una herramienta compleja.
- El acceso a los datos, utilizando las dos principales librerías llamadas DAO y ADO. Se explicarán los diferentes componentes de cada una de estas librerías.
- Se examinará el lenguaje SQL que se aplica a Access, para permitir a los lectores entender la utilidad de las diferentes sintaxis posibles.
- Los eventos en Access, piedra angular de la interacción con el usuario y la automatización de procesos.
- A continuación, se analizarán los formularios e informes en Access para explicar lo que pueden aportar a una aplicación completa.
- Una vez que se dominen las interfaces, es interesante continuar con la personalización de las interfaces, y fundamentalmente la cinta de opciones del menú de Access, que es completamente configurable.
- La posibilidad de controlar desde Access otras aplicaciones disponibles en el universo de Microsoft Office, también se tratará en este libro, analizando los principales componentes de las herramientas Excel, Word y Outlook.
- Más adelante se menciona el lenguaje XML, para permitir entender la interacción de la Web en las aplicaciones desarrolladas para las que hoy en día el acceso a Internet suele ser un requisito previo habitual en las definiciones de las necesidades de la aplicación.
- El uso de las API Windows y de la interacción posible entre Microsoft Access y el sistema operativo.
- Para terminar este libro, se mostrará un ejemplo de miniaplicación, que permite gestionar un refugio para animales, con sus adopciones.

Una vez que haya terminado de leer la totalidad de este manual, podrá comprobar en qué punto es necesario llamar al lenguaje VBA para obtener los mejores resultados posibles, tanto en términos de interacción con los usuarios de sus herramientas como en términos de posible rendimiento.

Analogías y comparaciones

A lo largo de todo el libro, se harán analogía con los perros, para facilitar la comprensión de las nociones abordadas, que adornarán los capítulos a medida que se mencionan las novedades.

Los objetos de Access

Una base de datos de Access 2019 es un archivo con extensión .accdb. Se compone de los siguientes objetos.

1. Tablas

Las tablas son contenedores que almacenan los datos de la base de datos. Pueden estar directamente almacenadas en el archivo de Access (tablas en local), así como pertenecer a otras bases de datos (tablas relacionadas).

Para crear una nueva tabla, puede ir a la pestaña **Crear** y seleccionar, del grupo **Tablas**, el icono **Tabla** o **Creación de tabla**.

2. Consultas

Las consultas permiten:

- seleccionar datos u ordenar los datos de las tablas,
- añadir, actualizar o eliminar los datos de las tablas,
- realizar cálculos y agrupaciones con los datos,
- crear, modificar o eliminar tablas,
- ejecutar instrucciones SQL.

Para crear una nueva consulta, puede ir a la pestaña **Crear** y seleccionar, del grupo **Consultas**, el icono **Asistente Consulta** o **Creación de consulta**.

3. Formularios

Los formularios son las interfaces de usuario dinámicas, que permiten proporcionar un soporte visual entre el usuario y los datos de la base de datos.

Para crear un nuevo formulario, puede ir a la pestaña **Crear** y seleccionar, del grupo **Formularios**, los iconos **Creación de formulario**, **Formulario vacío** o **Asistente Formulario**.

4. Informes

Los informes son las interfaces estáticas destinadas a imprimirse. Pueden contener elementos muy sencillos, como listas de datos, así como cuadros de mando con cálculos más complejos.

Para crear un nuevo informe, puede ir a la pestaña **Crear** y seleccionar, del grupo **Informes**, los iconos **Creación de informe**, **Informe vacío** o **Asistente Informe**.

5. Macros

Las macros son series de instrucciones que sirven para automatizar las tareas programadas en un lenguaje propio de Access, pero diferente de VBA.

Para crear una nueva macro, puede ir a la pestaña **Crear** y seleccionar, del grupo **Macros y código**, el icono **Macro**.

6. Módulos

Los módulos (y módulos de clase) son los archivos en los que se codifican los programas en VBA.

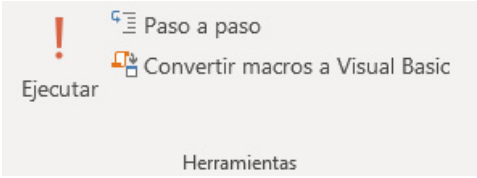
Para crear un nuevo módulo (y módulo de clase), puede ir a la pestaña **Crear** y seleccionar, del grupo **Macros y código**, el icono **Módulo** (respectivamente **Módulo de clase**).

Pasar de las macros a VBA

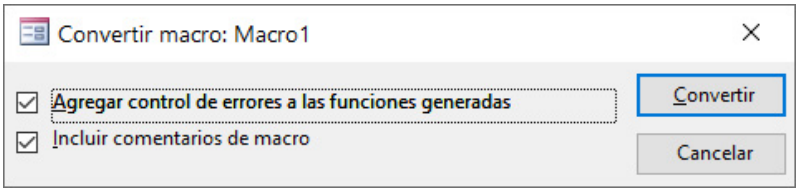
Si dispone ya de una base de datos en la que ha creado macros, es posible transformarlas en código VBA. El código VBA es más elaborado que el de las macros, y permitirá una mejor administración de las operaciones y de los errores que pueda generar Access.

1. Conversión de las macros en VBA

Para convertir las macros ya existentes en VBA, es necesario abrir la macro en modo creación, en la pestaña **Creación**, grupo **Herramientas**, y hacer clic en el icono **Convertir macros a Visual Basic**.



Aparece un cuadro de diálogo como el que se muestra a continuación:

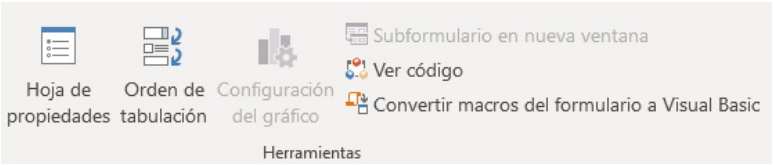


Puede seleccionar la administración automática de los errores marcando la primera casilla y mantener los comentarios marcando la segunda.

Haciendo clic en el botón **Convertir**, se crea un módulo automáticamente con el código VBA que corresponde a las instrucciones de la **Macro convertida**, automáticamente llamada **Macro convertida**, seguido del nombre de la macro.

2. Conversión de las macros de un formulario en VBA

Para convertir las macros que pertenecen a un formulario, es posible abrir el formulario en modo creación, seleccionar la pestaña **Creación** y hacer clic, en el grupo **Herramientas**, en el icono **Convertir macros del formulario a Visual Basic**.



El cuadro de diálogo es igual al que se ha mostrado anteriormente.

Se creará un módulo automáticamente llamado **Macro convertida**, seguido del nombre de la macro.

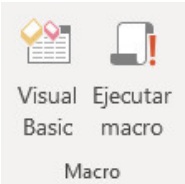
Introducción

El entorno de desarrollo o IDE (*Integrated Development Environment*), que se utiliza con Microsoft Access 2019, se llama **Visual Basic Editor** o VBE. Este es el entorno donde podrá escribir, probar y modificar su código VBA.

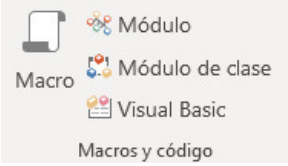
Cómo acceder al entorno

Para acceder a la interfaz de desarrollo, hay varios métodos posibles.

En la interfaz de Microsoft Access, puede hacer clic en el icono **Visual Basic** del grupo **Macro**, en la pestaña **Herramienta de base de datos**.



También es posible acceder mediante por la pestaña **Crear**, haciendo clic en el icono **Visual Basic** del grupo **Macros y código**.

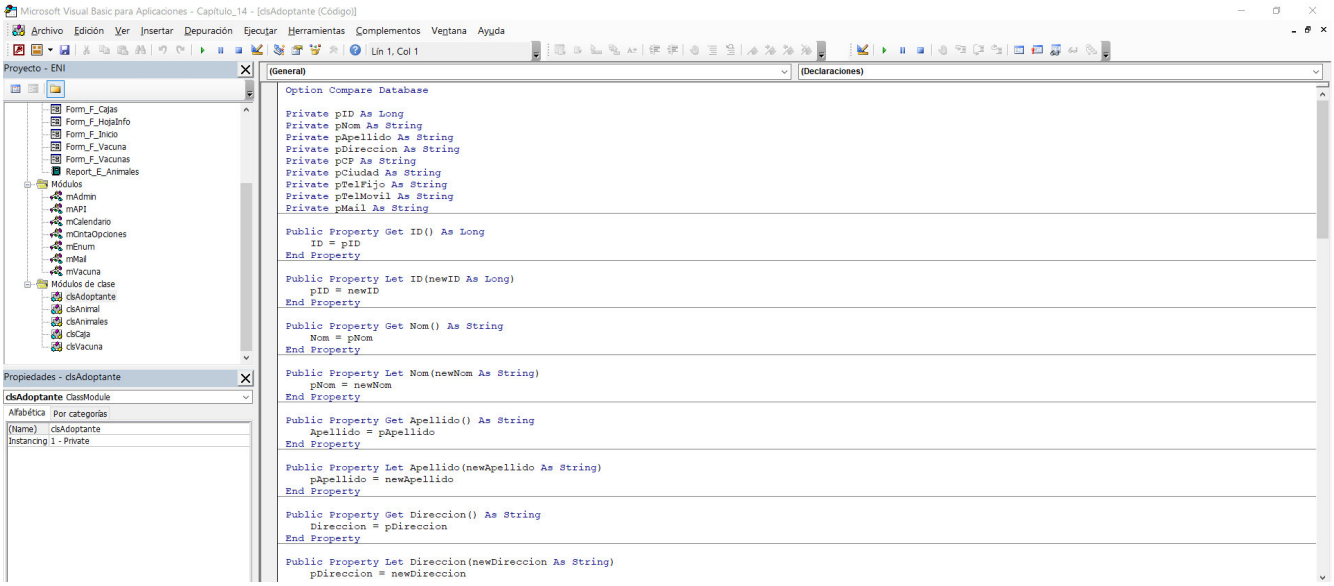


Además puede acceder al VBE desde la pantalla de propiedades de los formularios o de los informes, en la pestaña **Eventos**, seleccionando un procedimiento de evento.

Para terminar, puede abrir directamente la interfaz de VBE, gracias al acceso directo de teclado [Alt][F11].

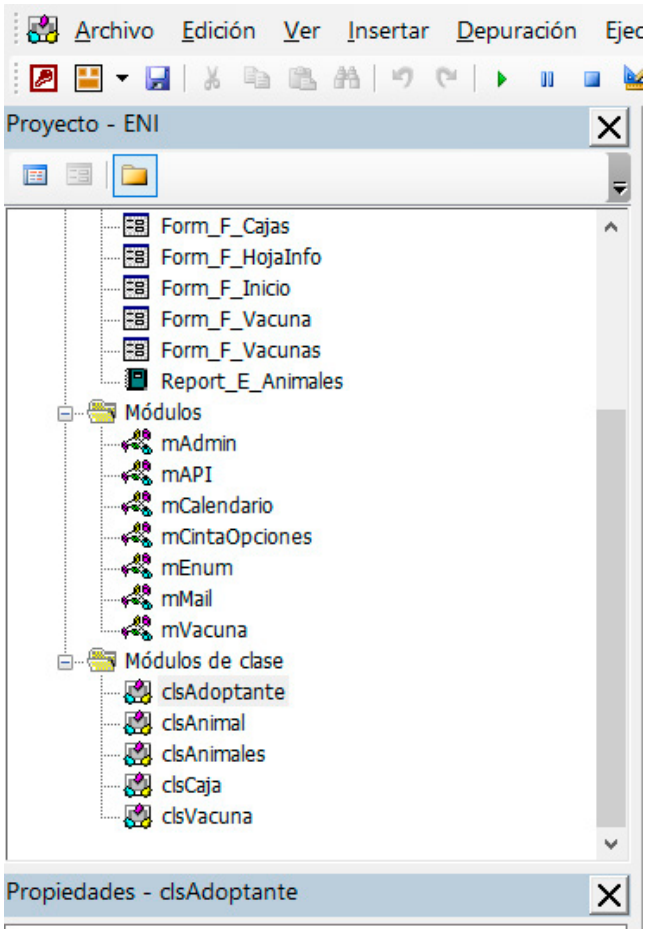
Las interfaces

La interfaz generada en VBE es la siguiente. Vamos a estudiar las diferentes zonas y menús que la componen.



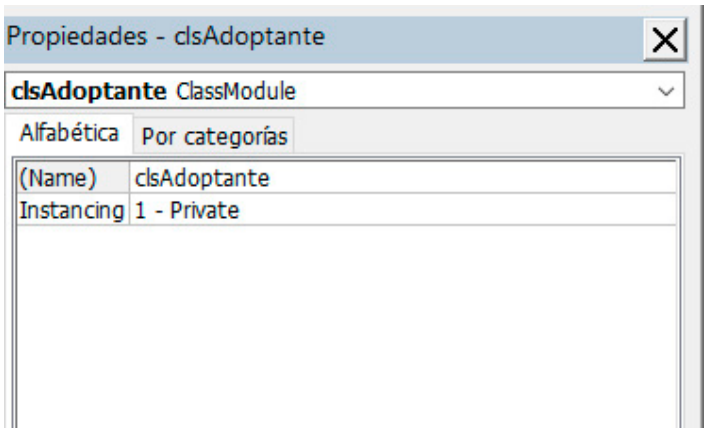
1. El explorador de proyectos

En la zona de explorador de proyectos, podemos ver el conjunto de objetos que están o pueden estar relacionados con la programación VBA, en su aplicación. Por tanto, podemos mostrar los formularios e informes, así como los módulos y los módulos de clase. Haciendo doble clic en el objeto en la ventana de la arborescencia, se accede al código en la zona de edición.



2. Las propiedades

Debajo de la ventana de la arborescencia, podemos encontrar la ventana de propiedades, que mostrará, según el objeto activo, las diferentes propiedades de este, permitiendo al desarrollador modificar su valor si lo desea.



3. La zona de edición

La ventana a la derecha de la ventana de VBE es la zona de edición, en la que se muestra el código y que modifica el desarrollador. Esta ventana se presenta en forma de un bloc de notas, utilizando la coloración sintáctica, es decir, que los diferentes elementos que están escritos tomarán automáticamente un color según se trate de una palabra clave, un comentario o código estándar.

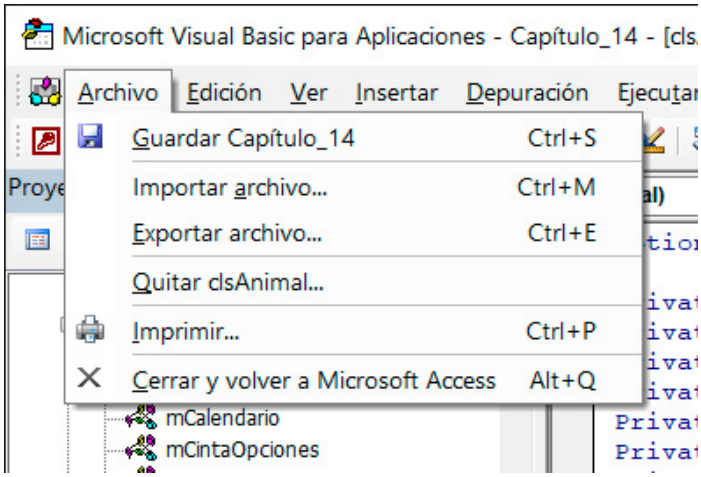
```
Public Sub Guardar(Salir As Boolean)
    'si el animal tiene todo los elementos necesarios, se guarda la información
    If oAnimal.EstaValidado Then
        pANI_ID = oAnimal.Guardar
        If Salir Then
            DoCmd.Close acForm, Me.Name
        Else
            SetParam "ANI_ID", pANI_ID
            SetParam "ADO_ID", oAnimal.AdoptanteID
            SetParam "DT_ADOPCION", oAnimal.FechaAdopcion
        End If
        'si el formulario de lista de animales está abierto, se actuaiza la lista de la información.
        If CurrentProject.AllForms("F_Animales").IsLoaded Then
            Form_F_Animales.LstAnimales.Requery
        End If
    Else
        MsgCaja "La información está incmpleta o es incorrecta", vbCritical + vbOKOnly
    End If
End Sub
```

Encima de la zona de edición, hay dos zonas de listas desplegables en las que podemos encontrar los diferentes objetos de formularios e informes (zona de lista de la izquierda) y los eventos de los objetos, así como las funciones y procedimientos (zona de lista de la derecha).

4. Los menús

Los diferentes menús visibles en la barra de menú principal son los siguientes.

a. Archivo

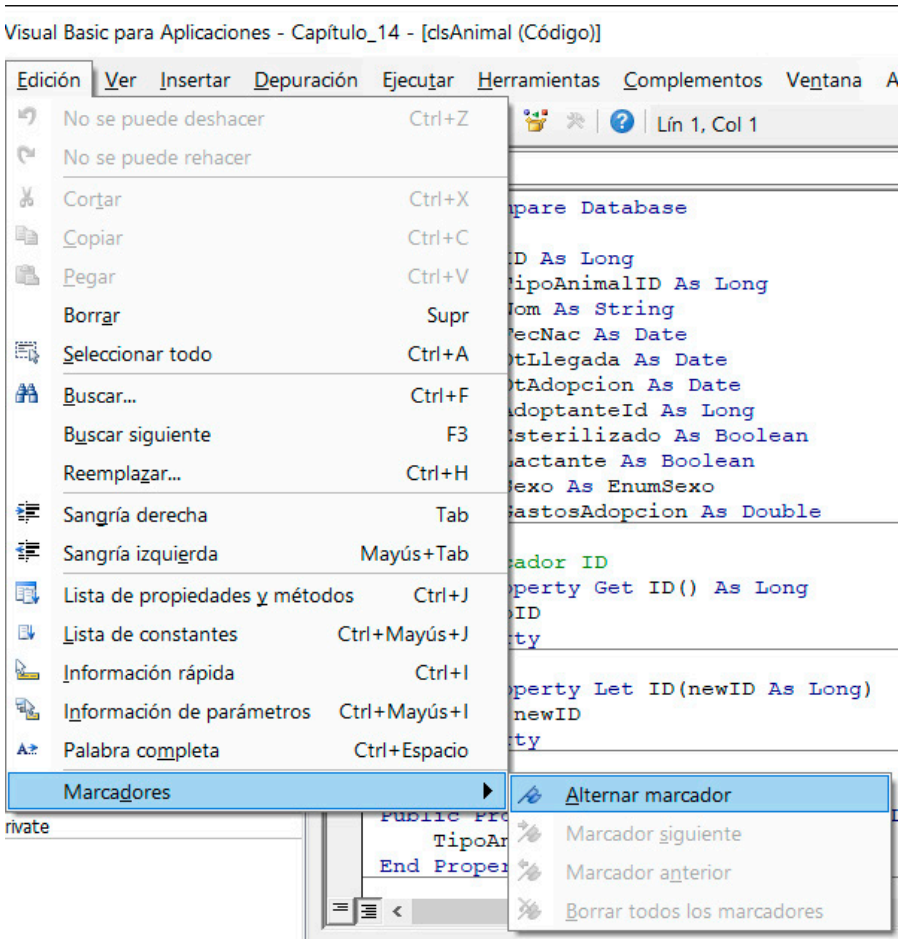


El menú **Archivo** permite las siguientes acciones:

Elemento	Acceso directo de teclado	Descripción
Guardar [nombreBaseDatos]	[Ctrl] **S**	Permite guardar las modificaciones en la base de datos.
Importar archivo...	[Ctrl] **M**	Permite importar los formularios, informes, módulos o módulos de clase en la base de datos.
Exportar archivo...	[Ctrl] **E**	Permite exportar los formularios, informes, módulos o módulos de clase, en los archivos con las extensiones respectivas frm, bas y cls.
Quitar [nombreObjeto]		Permite eliminar el objeto activo. Algunos objetos no se pueden eliminar desde la interfaz de VBE,

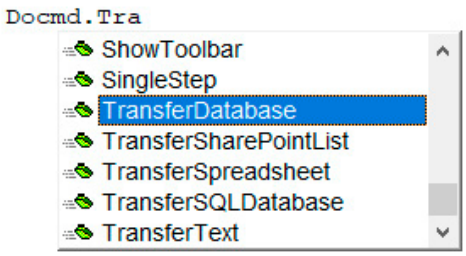
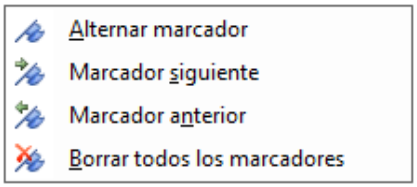
		como por ejemplo los formularios, que solo se pueden eliminar manualmente desde la interfaz principal de Microsoft Access.
Imprimir	[Ctrl] **P**	Permite imprimir el contenido del formulario, informe, módulo o módulo de clase.
Cerrar y volver a Microsoft Access	[Alt] **Q**	Permite cerrar la IDE y volver a la interfaz de Microsoft Access.

b. Edición

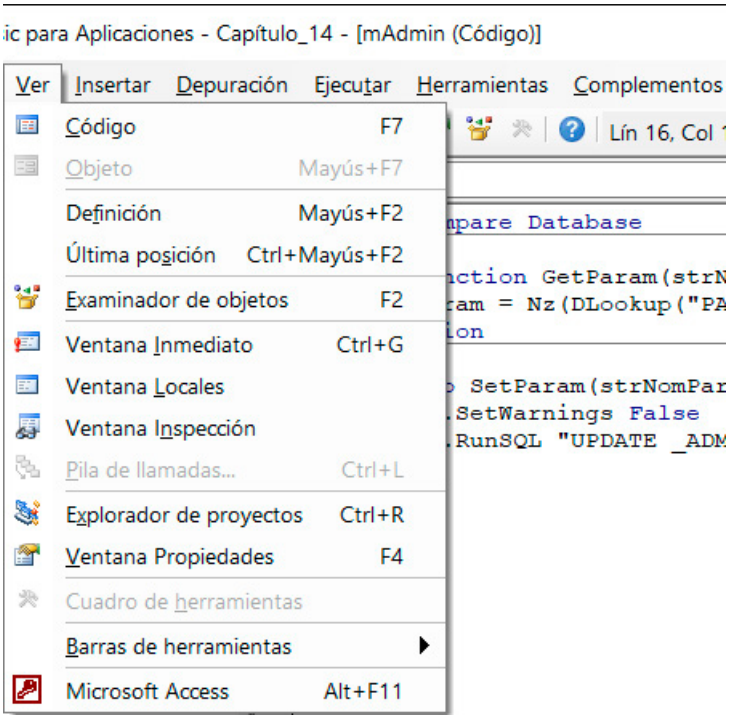


En este menú encontramos elementos similares al grupo **Portapapeles** de la pestaña **Bienvenida** de Microsoft Access.

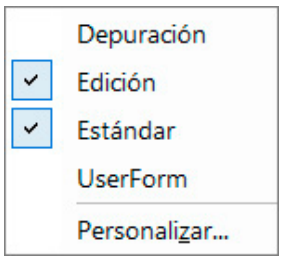
Elemento	Acceso directo de teclado	Descripción
Deshacer	[Ctrl] **Z**	Permite anular la última acción (copiar, pegar, selección de texto o mover el objeto).
Rehacer		Permite repetir la última acción. Por supuesto, algunas acciones no se pueden repetir (inserción de objetos fundamentalmente).
Cortar	[Ctrl] **X**	Permite cortar el texto.
Copiar	[Ctrl] **C**	Permite copiar el texto.
Pegar	[Ctrl] **V**	Permite pegar el texto.

Borrar	[Supr]	Permite eliminar el texto.
Seleccionar todo	[Ctrl] **A**	Permite seleccionar todo el texto de un formulario, informe, módulo o módulo de clase.
Buscar	[Ctrl] **F**	Permite buscar texto a diferentes niveles en el programa.
Buscar siguiente	[F3]	Permite continuar la búsqueda.
Reemplazar	[Ctrl] **H**	Permite sustituir el texto a diferentes niveles en el programa.
Sangría derecha	[Tab]	Permite indentar el texto hacia la derecha, para mejorar la legibilidad del código.
Sangría izquierda	[Shift][Tab]	Permite indentar el texto a la izquierda.
Lista de propiedades y métodos	[Ctrl] **J**	Permite visualizar en la ventana de edición la lista de las propiedades y métodos disponibles (AutoCompletion).
Lista de constantes	[Ctrl][Shift] **J**	Permite visualizar en la ventana de edición la lista de las constantes disponibles (AutoCompletion).
Información rápida	[Ctrl] **I**	Permite visualizar la sintaxis general del elemento seleccionado (función, procedimiento, instrucción, etc).
Información de parámetros	[Ctrl][Shift] **I**	Permite visualizar en la ventana de edición la información de los argumentos que se deben proporcionar. Representa una ayuda a la redacción del código.
Palabra completa	[Ctrl][Espacio]	Permite completar la palabra que se está escribiendo. Permite evitar tener que escribir la totalidad del nombre de una función o variable, con un nombre largo. Ejemplo, si escribe "Docmd.Tran", VBE le propondrá lo siguiente: 
Marcadores		Permite cambiar y desplazarse entre los marcadores en el programa. 

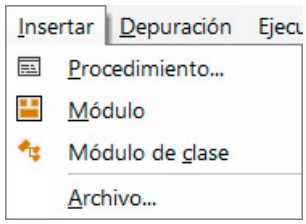
c. Ver

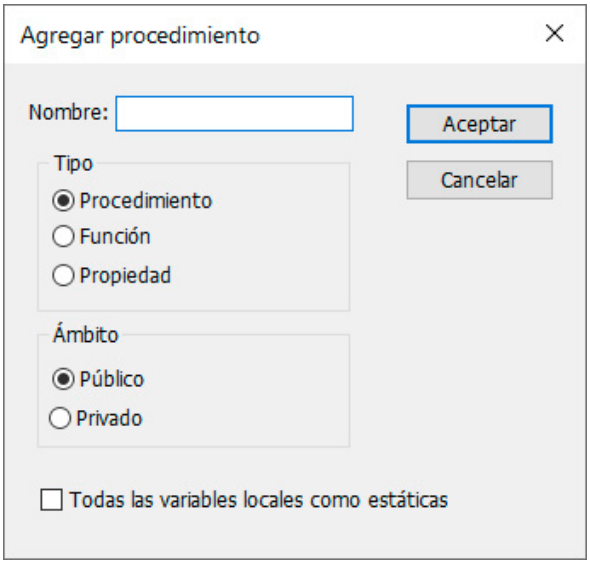


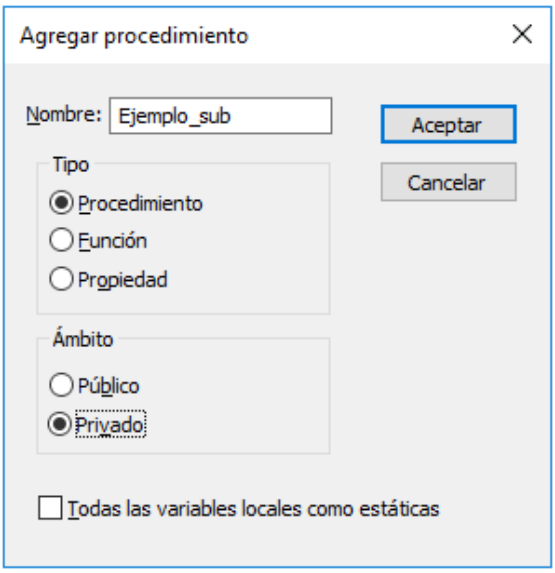
Elemento	Acceso directo de teclado	Descripción
Código	[F7]	Permite visualizar el código del objeto seleccionado en la ventana de edición.
Objeto	[Shift][F7]	Permite cambiar en la interfaz de Access con el objeto, cuyo código aparece en la ventana de edición.
Definición	[Shift][F2]	Permite acceder a la ubicación del código en la ventana que define el procedimiento, en el que se encuentra el cursor.
Última posición	[Ctrl][Shift][F2]	Permite volver a la última posición en el código. Solo puede volver 8 posiciones como máximo en VBE.
Examinador de objetos	[F2]	Permite visualizar el explorador de objetos.
Ventana Inmediato	[Ctrl] **G**	Permite visualizar la ventana de ejecución. Ver la sección Las ventanas, en este capítulo.
Ventana Locales		Permite visualizar las variables del procedimiento actual de ejecución y su valor. Ver la sección Las ventanas, en este capítulo.
Ventana Inspección		Permite visualizar la ventana de inspección. Ver la sección Las ventanas, en este capítulo.
Pila de llamadas	[Ctrl] **L**	Permite visualizar la pila de llamadas durante la ejecución de un código. Ver la sección Las ventanas.
Explorador de proyectos	[Ctrl] **R**	Permite visualizar el explorador de proyectos.
Ventana Propiedades	[F4]	Permite visualizar la ventana de propiedades.
Cuadro de herramientas		Permite visualizar la caja de herramientas (no funciona en Microsoft Access, la interfaz de creación y modificación de los formularios e informes se hace en Access y no en VBE).
Barras de herramientas		Permite visualizar u ocultar las diferentes barras

		de herramientas. Ver la sección Las barras de herramientas, en este capítulo. 
Microsoft Access	[Alt][F11]	Permite cambiar la visualización a Microsoft Access.

d. Insertar

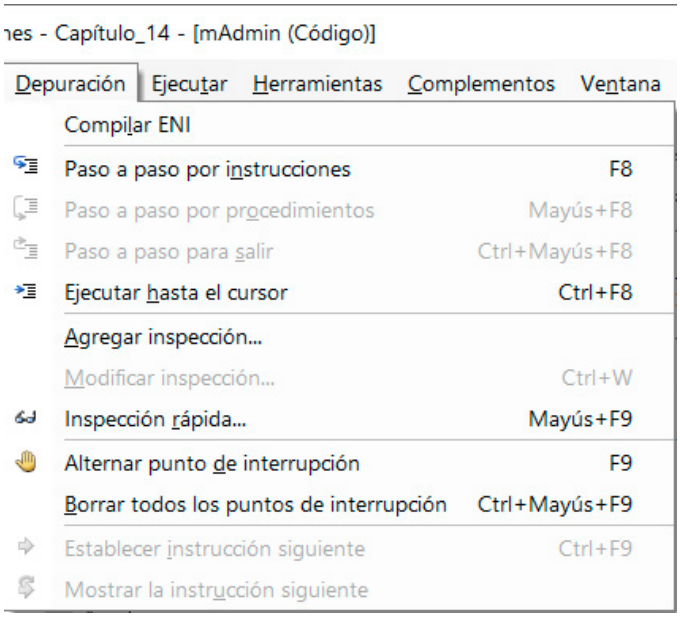


Elemento	Acceso directo de teclado	Descripción
		Permite insertar un procedimiento. Se muestra la siguiente interfaz y permite al desarrollador añadir los elementos que desea. 
		Determina el nombre, tipo y ámbito del procedimiento:

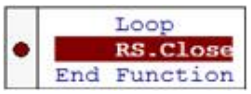
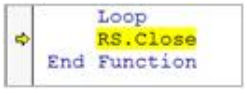
Procedimiento		
		El resultado en la zona de edición será el siguiente:
		`Private Sub Ejemplo_sub()`
		`End Sub`

Módulo		Permite insertar una hoja de código de tipo Módulo.
Módulo de clase		Permite insertar una hoja de código de tipo Módulo de clase.
Archivo		Permite insertar una hoja de código a partir de un archivo. Este elemento no está disponible si no hay ningún objeto activo en la zona Explorador de proyectos.

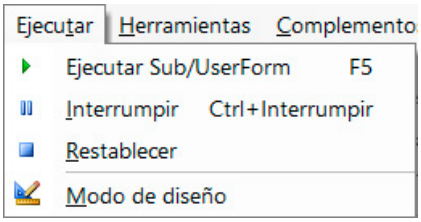
e. Depuración

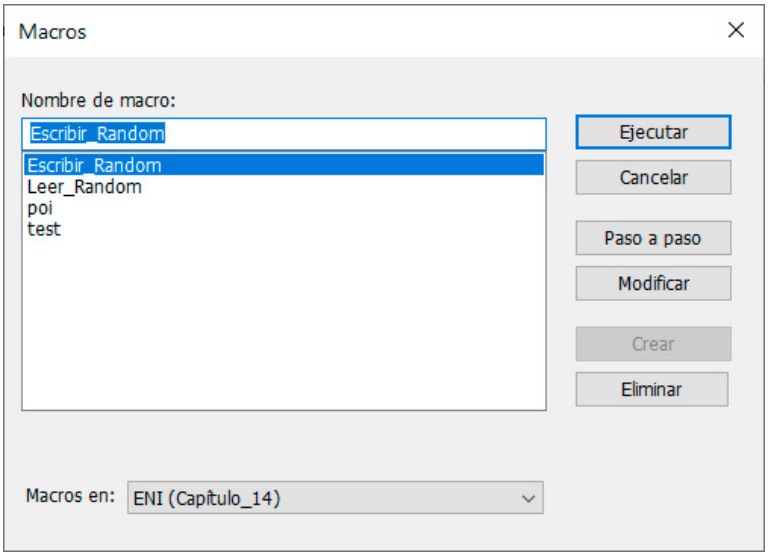


Elemento	Acceso directo de teclado	Descripción
Compilar [nombre del Proyecto]		Permite realizar una compilación del proyecto.

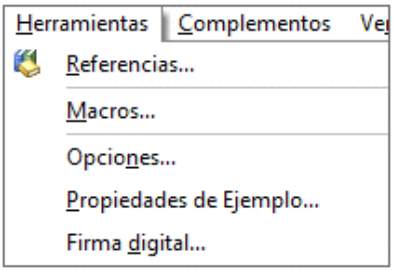
		Todos los procedimientos se comprobarán en términos de coherencia.
Paso a paso por instrucciones	[F8]	Permite ejecutar un procedimiento, instrucción a instrucción.
Paso a paso por procedimientos	[Shift][F8]	Permite ejecutar un procedimiento instrucción a instrucción, pero sin entrar en los bucles y subprocedimientos.
Paso a paso para salir	[Ctrl][Shift][F8]	Permite ejecutar el conjunto de líneas restantes del procedimiento donde se sitúa el punto de ejecución actual.
Ejecutar hasta el cursor	[Ctrl][F8]	Permite ejecutar un procedimiento hasta la ubicación del cursor.
Agregar inspección		Permite crear una nueva inspección. Ver la sección Las ventanas, en este capítulo.
Modificar inspección	[Ctrl] **W**	Permite modificar una inspección existente. Ver la sección Las ventanas, en este capítulo.
Inspección rápida	[Shift][F9]	Permite visualizar la ventana inspección rápida. Ver la sección Las ventanas, en este capítulo.
Alternar punto de interrupción	[F9]	Permite definir o eliminar un punto de ruptura en la línea actual. No es posible crear un punto de ruptura en una línea sin código (incluidos los comentarios). Por defecto, la línea estará precedida de un círculo de color burdeos.  Para la administración de los colores, ver la sección Formato del editor, en este capítulo.
Borrar todos los puntos de interrupción	[Ctrl][Shift][F9]	Permite eliminar todos los puntos de ruptura del programa. No es posible anular esta operación.
Establecer instrucción siguiente	[Ctrl][F9]	Permite indicar la siguiente línea que se ejecutará por el programa. Esta técnica permite no ejecutar ningún código intermedio entre la línea actual y la seleccionada. No es posible definir la instrucción siguiente en un procedimiento diferente al procedimiento actual de ejecución.
Mostrar la instrucción siguiente		Permite resaltar la siguiente línea que se ejecutará. Por defecto, la línea que contiene la instrucción siguiente se indicará por una flecha amarilla.  Para la administración de los colores, ver la sección Formato del editor, en este capítulo).

f. Ejecutar



Elemento	Acceso directo de teclado	Descripción
Ejecutar	[F5]	Permite visualizar la ventana de las macros disponibles en el proyecto o iniciar la ejecución del procedimiento en el que se encuentra el cursor. 
Interrumpir	[Ctrl][Detener]	Permite detener la ejecución del programa.
Restablecer		Permite cerrar la ejecución del programa.
Modo de diseño		Permite cambiar el modo de edición a Modo de Diseño.

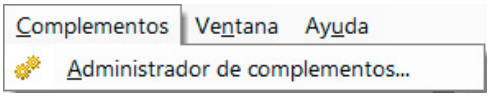
g. Herramientas



Elemento	Acceso directo de teclado	Descripción
Referencias		Permite visualizar la ventana de Referencias. Ver la sección Las ventanas, en este capítulo.
Macros		Permite visualizar la ventana de macros. Ver la sección Los menús - Ejecutar, en este capítulo.
Opciones		Permite visualizar la ventana de opciones. Ver la sección Las opciones de VBE, en este capítulo.

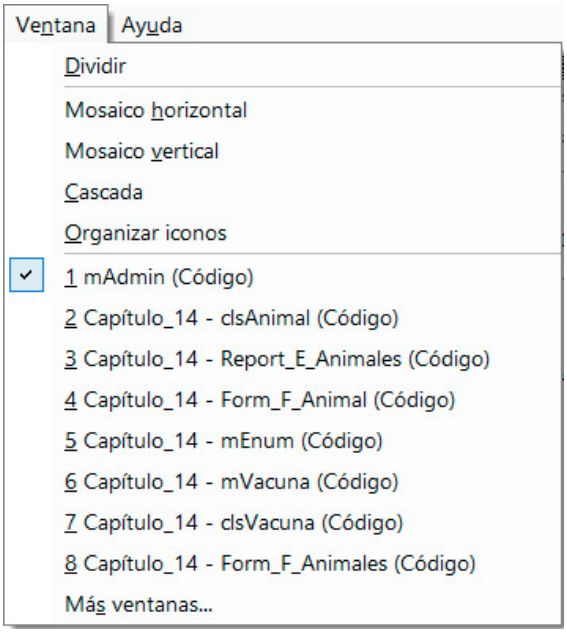
Propiedades de [Base de datos]		Permite visualizar la ventana de propiedades del Proyecto. Ver la sección Las ventanas en este capítulo.
Firma digital		Permite visualizar la información de la firma electrónica del proyecto. Ver la sección Microsoft Access y el paquete firmado, en este capítulo.

h. Complementos



Elemento	Acceso directo de teclado	Descripción
Administrador de complementos		Permite visualizar la lista de complementos. Ver la interfaz Completado en Microsoft Access: **Archivo - Opciones - Complementos**.

i. Ventana

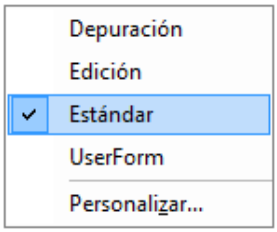


Elemento	Acceso directo de teclado	Descripción
Dividir		Permite organizar la ventana de la zona de edición en dos partes. Esta técnica permite visualizar dos partes del código al mismo tiempo.
Mosaico horizontal		Permite visualizar todas las ventanas de edición de código disponibles, dividiendo la ventana horizontalmente, en dos partes de igual tamaño.
Mosaico vertical		Permite visualizar todas las ventanas de edición de código disponibles, dividiendo la ventana verticalmente, en dos partes de igual tamaño.
Cascada		Permite visualizar en cascada todas las ventanas de

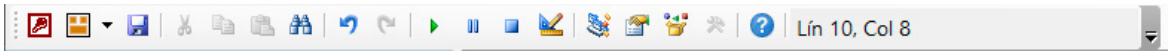
		edición de código.
Organizar iconos		Permite reorganizar los iconos.
Lista de las ventanas abiertas		Permite visualizar las ventanas abiertas.
Más ventanas		Aparece si hay más de ocho ventanas abiertas al mismo tiempo.
Cerrar las ventanas		Permite cerrar las ventanas abiertas.

5. Las barras de herramientas

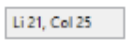
Además de los diferentes menús visibles, se pueden mostrar varias barras de herramientas. Son accesibles desde el menú **Ver - Barras de herramientas**.



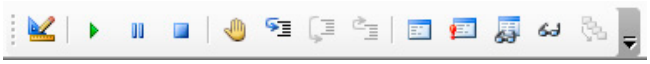
a. Estándar



Elemento	Icono	Descripción
Ver Microsoft Access		Permite volver a la interfaz principal de Microsoft Access 2019.
Agregar módulo		Permite añadir un módulo, un módulo de clase o un procedimiento (Sub/Function).
Guardar		Permite guardar la base de datos.
Cortar		Permite cortar el texto seleccionado.
Copiar		Permite copiar el texto seleccionado.
Pegar		Permite pegar el texto del portapapeles a la ubicación del cursor.
Buscar		Permite buscar texto.
Deshacer		Permite anular la última acción (si es posible).

Rehacer		Permite repetir la última acción (si es posible).
Ejecutar		Permite ejecutar código.
Interrumpir		Permite detener el código.
Restablecer		Permite reiniciar el código.
Modo de diseño		Permite pasar del modo Edición al modo Creación.
Explorador de proyectos		Permite visualizar el explorador de proyectos.
Ventana Propiedades		Permite visualizar la ventana de Propiedades.
Explorador de objetos		Permite visualizar la ventana del explorador de objetos.
Cuadro de herramientas		Permite visualizar la caja de herramientas (no disponible en Microsoft Access).
Ayuda de Microsoft VBA		Permite visualizar la ayuda de Microsoft.
Ubicación actual en la ventana de código		Permite ver el número de línea y columna donde se sitúa el cursor, dentro de la zona de edición actual.

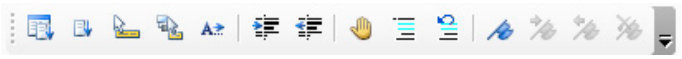
b. Depuración



Elemento	Icono	Descripción
Modo de diseño		Permite pasar del modo Edición al modo Creación.
Ejecutar		Permite ejecutar código.
Interrumpir		Permite detener el código.
Restablecer		Permite reiniciar el código.

Alternar punto de interrupción		Permite visualizar o eliminar un punto de ruptura.
Paso a paso por instrucciones		Permite ejecutar el código en modo paso a paso.
Paso a paso por procedimientos		Permite ejecutar el código en modo paso a paso, sin entrar en los bucles ni los subprocedimientos.
Paso a paso para salir		Permite ejecutar las últimas líneas de un procedimiento.
Ventana Locales		Permite visualizar la ventana Variables locales.
Ventana Inmediato		Permite visualizar la ventana Ejecución.
Ventana Inspección		Permite visualizar la ventana Inspección.
Inspección rápida		Permite visualizar la ventana Inspección rápida.
Pila de llamadas		Permite visualizar la pila de llamadas.

c. Edición



Elemento	Icono	Descripción
Lista de propiedades y métodos		Permite visualizar las posibles propiedades. Ver la sección Los menús - Edición, en este capítulo.
Lista de constantes		Permite visualizar las constantes disponibles. Ver la sección Los menús - Edición, en este capítulo.
Información rápida		Permite visualizar la información rápida. Ver la sección Los menús - Edición, en este capítulo.
Información de parámetros		Permite visualizar la información de los argumentos. Ver la sección Los menús - Edición, en este capítulo.
Palabra completa		Permite completar automáticamente la palabra actual de edición. Ver la sección Los menús - Edición, en este capítulo.

Sangría derecha		Permite insertar una indentación en el punto en el que se encuentre el cursor. Ver la sección Los menús - Edición, en este capítulo.
Sangría izquierda		Permite insertar una indentación negativa en el punto en el que se encuentre el cursor. Ver la sección Los menús - Edición, en este capítulo.
Alternar punto de interrupción		Permite añadir o eliminar un punto de ruptura en el punto en el que se encuentre el cursor. Ver la sección Los menús - Edición, en este capítulo.
Comentar bloque		Permite comentar todas las líneas seleccionadas en la zona de edición.
Descomentar bloque		Permite eliminar los comentarios de todas las líneas seleccionadas en la zona de edición.
Alternar marcador		Permite añadir y eliminar un marcador en el punto en el que se encuentre el cursor. Ver la sección Los menús - Edición, en este capítulo.
Marcador siguiente		Permite pasar al marcador siguiente. Ver la sección Los menús - Edición, en este capítulo.
Marcador anterior		Permite pasar al marcador anterior. Ver la sección Los menús - Edición, en este capítulo.
Borrar todos los marcadores		Permite eliminar todos los marcadores del proyecto. Ver la sección Los menús - Edición, en este capítulo.

d. UserForm

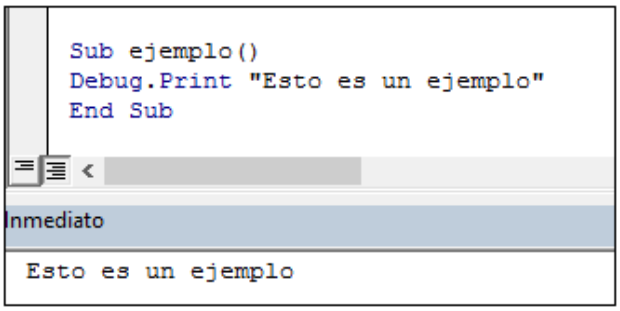
Esta barra de herramientas no está activada en Microsoft Access 2019; las modificaciones de los formularios e informes se hacen en la interfaz principal, y no en VBE.

6. Las ventanas

Varias ventanas completan el panel de información que puede mostrar Visual Basic Editor.

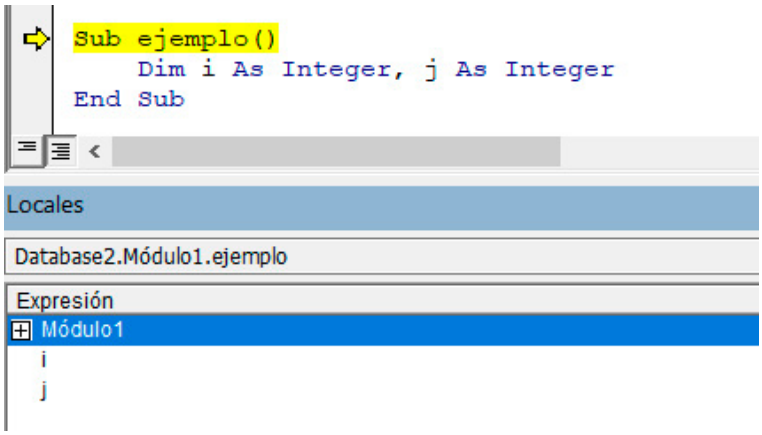
a. Ejecución

Inicialmente, la ventana **Ejecución** aparece en la parte inferior en la interfaz de VBE. A partir de esta zona, se pueden ejecutar los procedimientos y funciones. También se muestra en esta ventana la información del procedimiento Debug.Print.



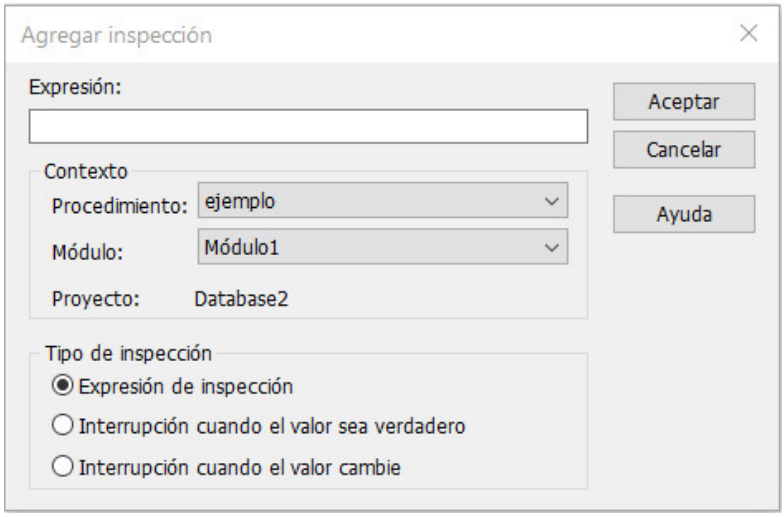
b. Variables locales

La ventana de las variables locales aparece en la parte inferior de la interfaz de VBE. En esta ventana se muestran las diferentes variables declaradas en el procedimiento o función que se está ejecutando, así como aquellas accesibles en el ámbito actual (ver el capítulo El lenguaje VBA - Ámbito y ciclo de vida de las variables). También es visible la información del nombre de variable, así como el valor y el tipo de dato.



c. Inspección

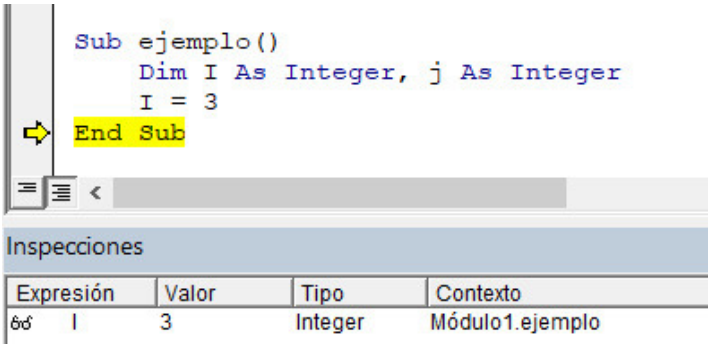
La ventana **Inspección** aparece en la parte inferior en la interfaz de VBE. Esta ventana permite visualizar los valores de las diferentes inspecciones ubicadas en el código. Para añadir una inspección, puede ir al menú **Depuración - Agregar inspección**.



Es posible introducir la expresión que queremos inspeccionar e indicar el contexto (nombre del procedimiento y nombre del módulo). El tipo de inspección permite seleccionar entre las siguientes posibilidades:

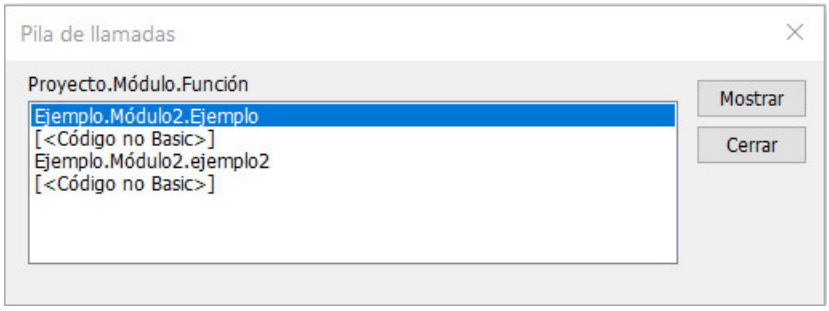
Tipo de inspección	Descripción
Expresión de inspección	La inspección muestra en momento del programa el valor de la expresión inspeccionada. Si no hay código actual, el valor por defecto mostrado es **<Fuera de contexto>**.
Interrupción cuando el valor sea verdadero	La inspección detiene el programa cuando la expresión (que devuelve un booleano) sea verdadera.
Interrupción cuando el valor cambie	La inspección detiene el programa cuando el programa modifique el valor de la expresión.

En la ventana son visibles la expresión, su valor, tipo y contexto `(NombreModulo.NombreProcedimiento)`.



d. Pila de llamadas

La ventana **Pila de llamadas** es una ventana modal (que recibe el foco sobre VBE y se debe cerrar antes de poder habilitar de nuevo la interfaz de VBE). Permite visualizar la pila de las diferentes funciones que se están ejecutando, indicando una pila.



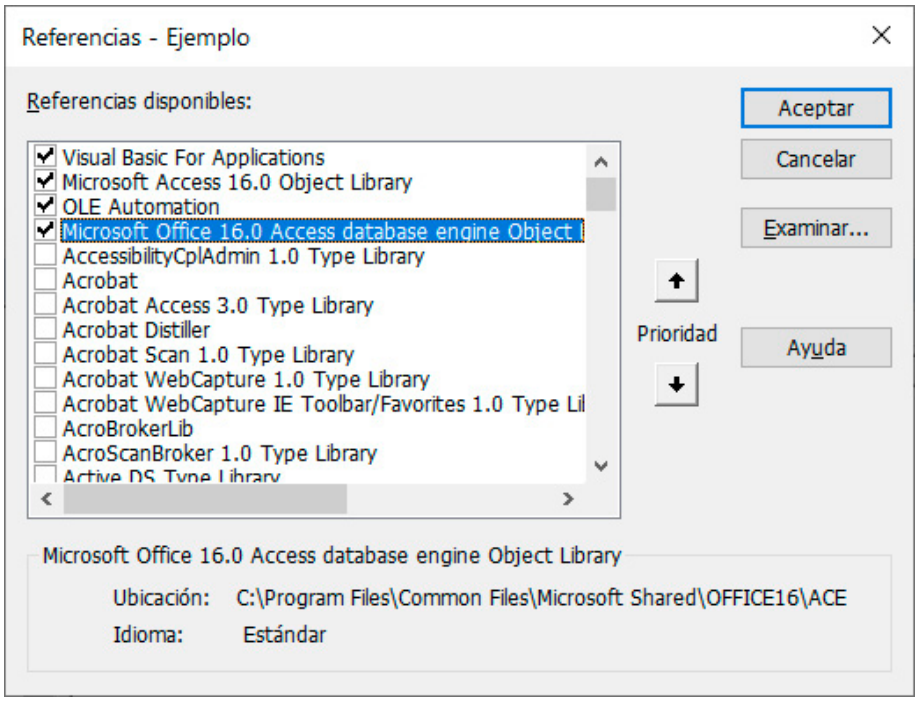
Como una pila de platos, es necesario leer la información con la instrucción actual en la parte superior de la pila y la instrucción que la haya llamado, debajo (principio LIFO: *Last In First Out*, último en entrar, primero en salir).

Así, en la captura de pantalla podemos ver el procedimiento **Ejemplo**, llamado por **Ejemplo2**. Seleccionando **Ejemplo.Modulo2.Ejemplo2** y haciendo clic en **Ver**, tendremos la ventana **Pila de llamadas**, que desaparecerá y el cursor se situará en la línea actual en **Ejemplo2**:

```
Sub Ejemplo2()
    Ejemplo
End Sub
```

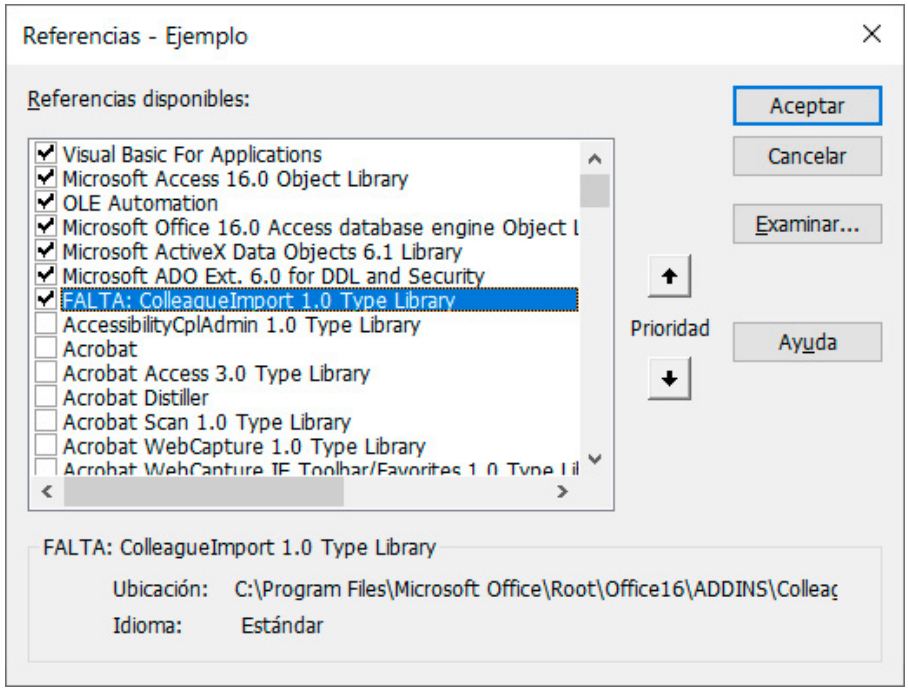
e. Referencias

La ventana de las referencias se abre desde el menú **Herramientas - Referencias**. Esta interfaz muestra el conjunto de librerías que se utilizan en el proyecto.



Para añadir una librería, es posible marcar su casilla de selección si aparece en la lista o, en caso contrario, hacer clic en **Examinar** y seleccionar el archivo fuente.

Si una de las librerías no está disponible en el puesto en el que se abre la base de datos, se mostrará un mensaje AUSENTE. En el ejemplo siguiente, podemos ver que la librería **Microsoft Scripting Runtime** no está disponible en el puesto.

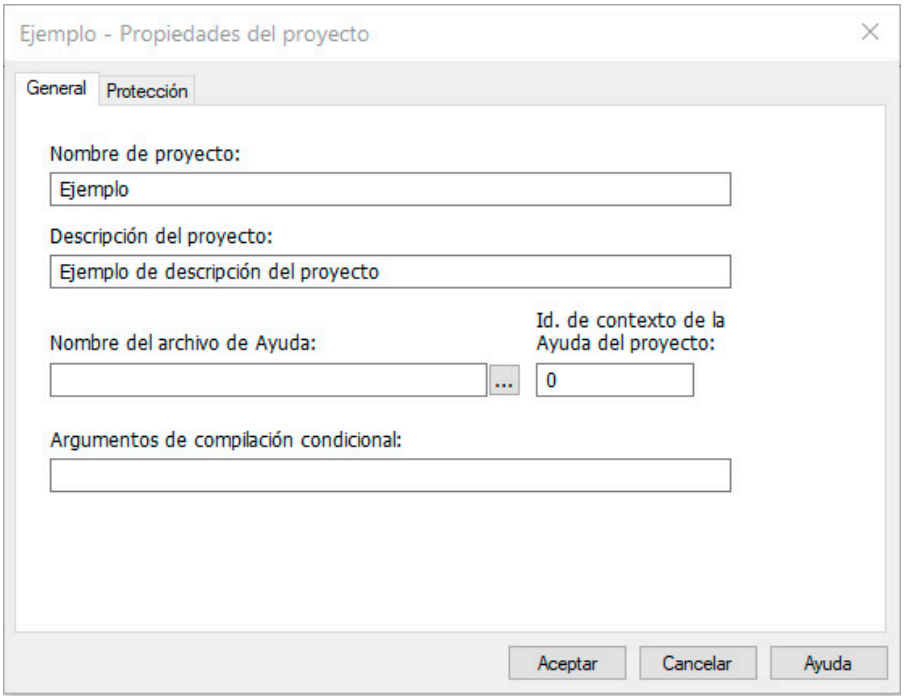


En este caso, será necesario deshabilitar la librería, lo que potencialmente puede dejar la aplicación no operativa,

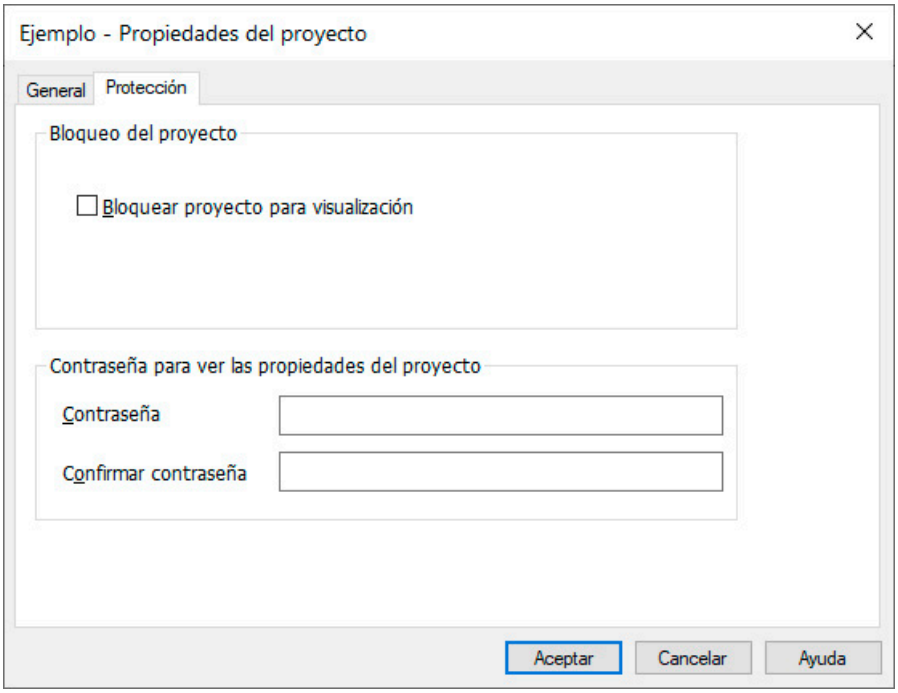
o bien instalar la librería en el puesto.

f. Propiedades del proyecto

La ventana **Propiedades del proyecto** aparece desde el menú **Herramientas - Propiedades de [NombreDelProyecto]**.



La pestaña **General** permite visualizar el nombre y la descripción del proyecto. También es posible indicar un archivo global de ayuda, así como eventuales argumentos de compilación.

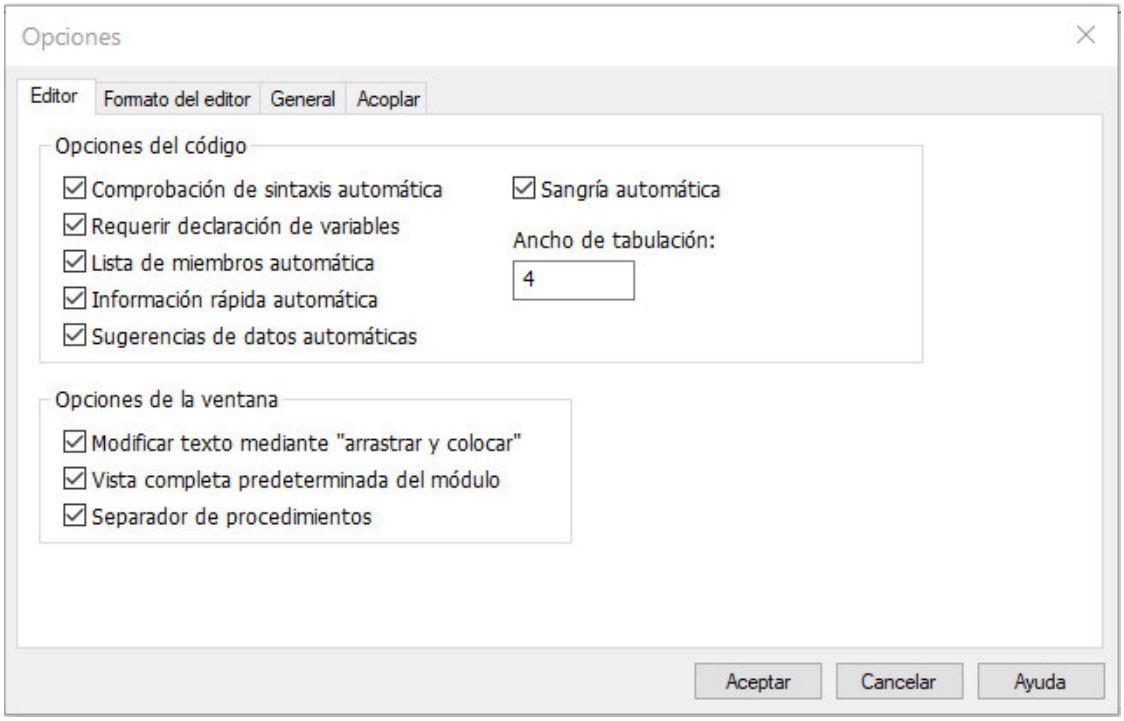


La pestaña **Protección** permite bloquear el proyecto si se desea, indicando una contraseña para desbloquearlo.

7. Las opciones de VBE

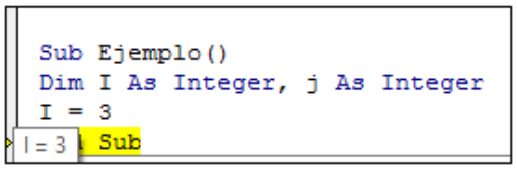
Es posible personalizar la interfaz de VBE. El desarrollador puede modificar las diferentes opciones.

a. Editor



En esta pestaña **Editor**, es posible configurar los siguientes elementos del código.

Opciones del código

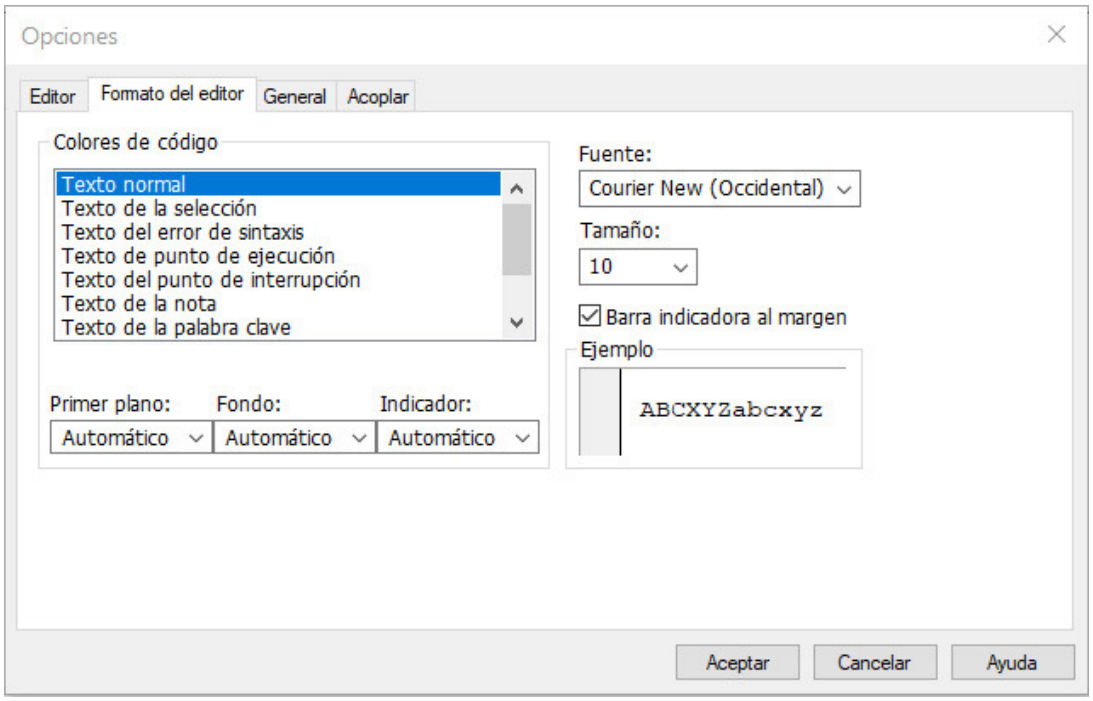
Elemento	Descripción
Comprobación de sintaxis automática	Permite comprobar la sintaxis durante la escritura del programa.
Requerir declaración de variables	Permite forzar al desarrollador a declarar las variables que utiliza en el programa. Cuando esta casilla está marcada, la instrucción `Options Explicit` aparecerá automáticamente en la parte superior de cada nuevo módulo.
Lista de miembros automática	Permite visualizar la lista de los elementos que pueden completar el código actual de edición.
Información rápida automática	Permite visualizar los argumentos del procedimiento actual de edición.
Sugerencias de datos automáticas	Permite visualizar los valores de las variables cuando el código se está ejecutando, cuando se pasa el ratón sobre una variable. 

Sangría automática	Permite ubicar cada línea al mismo nivel que la línea anterior.
Ancho de tabulación	Permite definir la longitud de las tabulaciones.

Opciones de la ventana

Elemento	Descripción
Modificar texto mediante "arrastrar y colocar"	Permite ejecutar la acción arrastrar-soltar para el texto.
Vista completa predefinida del módulo	Permite visualizar todo el módulo. Si la opción no está marcada, solo se mostrará el procedimiento actual en la zona de edición.
Separador de procedimientos	Permite ubicar una línea horizontal para delimitar los diferentes procedimientos.

b. Formato del editor



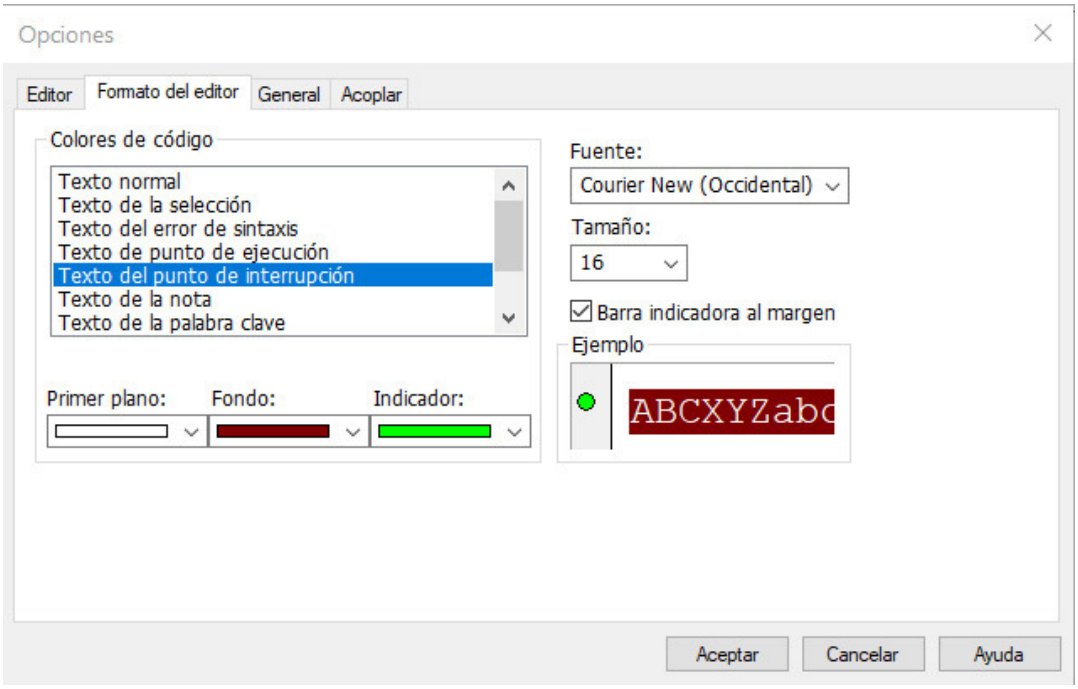
Esta pestaña permite personalizar el conjunto de tipos de letra, su tamaño y colores, que se utilizan en la zona de edición. Podemos modificar el color del texto (**Primer plano**), el del fondo (**Fondo**), así como el de las pestañas que aparecen en al margen (**Indicador**).

El tipo de letra de los caracteres utilizado, así como el tamaño, se pueden modificar.

Para terminar, podemos visualizar u ocultar la barra de indicadores (**Barra indicadora al margen**).

La vista previa global de las opciones aparece en la ventana **Ejemplo**.

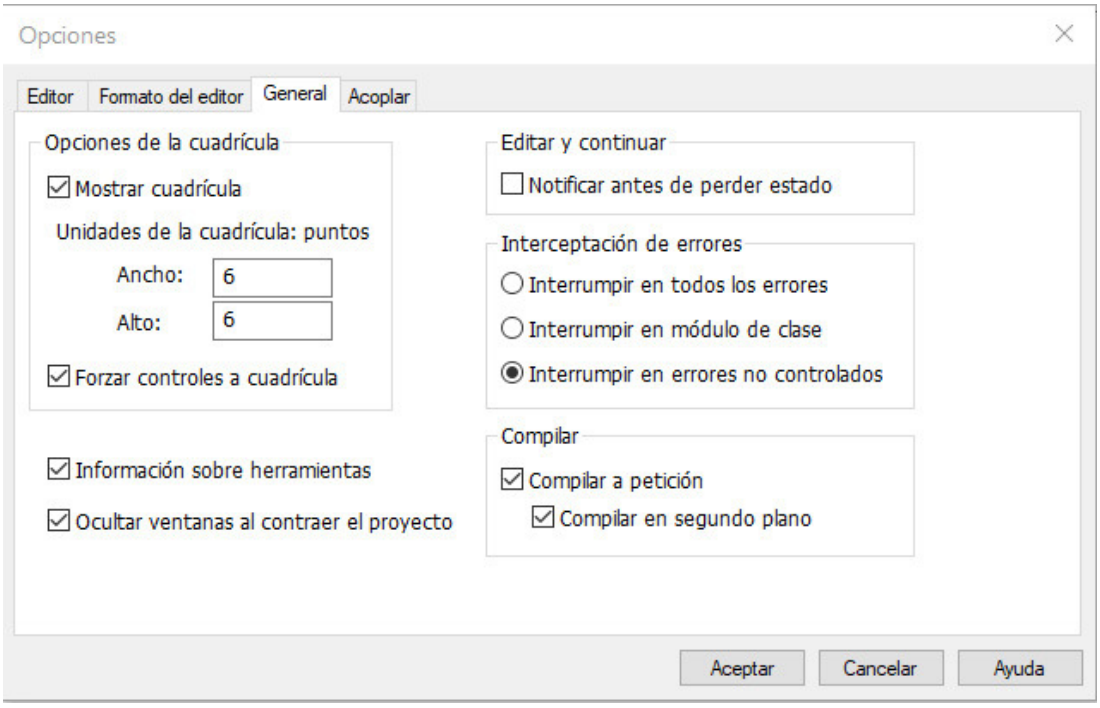
A continuación, un ejemplo de personalización:



Se pueden personalizar los siguientes elementos:

Elemento	Descripción
Texto normal	Representa el código más sencillo, como los caracteres de puntuación o el texto entre comillas.
Texto de la selección	Representa el código seleccionado por el desarrollador.
Texto del error de sintaxis	Representa el código cuando VBE detecta un error.
Texto del punto de ejecución	Representa el código que se va a ejecutar en modo paso a paso.
Texto del punto de interrupción	Representa el código cuando hay un punto de ruptura.
Texto de la nota	Representa el comentario en el código.
Texto de la palabra clave	Representa el código con las palabras clave VBA.
Texto del identificador	Representa el código con los nombres de variables, constantes, funciones y procedimientos.
Texto del marcador	Representa el código cuando hay un marcador.
Texto del retorno de la llamada	Representa el código cuando la pila de llamadas apunta al cursor.

c. General



La pestaña **General** permite personalizar las opciones generales del proyecto.

Las **Opciones de la cuadrícula** no tienen utilidad en VBE Access, ya que están relacionadas con la operación de control en los formularios e informes.

Si se está realizando la modificación, la opción **Notificar antes de perder estado** permite impedir que el desarrollador reinicie el programa.

El cuadro **Interceptación de errores** permite definir la manera en la que comportará el programa en caso de error, pasando a modo Detener. El comportamiento será el siguiente según las opciones:

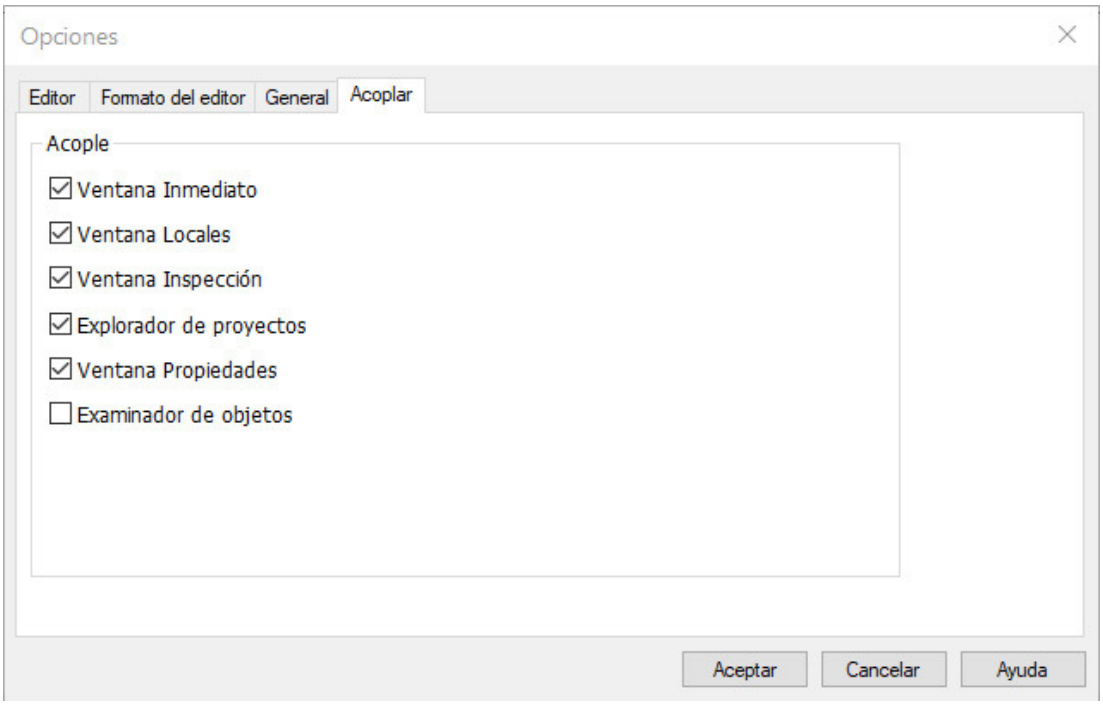
Opción	Descripción
Interrumpir en todos los errores	Pasa a modo Detener, independientemente del error.
Interrumpir en módulo de clase	Pasa a modo Detener en caso de error no gestionado, en un módulo de clase.
Interrumpir en errores no controlados	Pasa a modo Detener si no hay activo ningún administrador de (**On Error**). Esta opción es la que se recomienda más habitualmente.

La opción **Información sobre herramientas** permite visualizar información de las barras de herramientas.

La opción **Ocultar ventanas al contraer el proyecto** permite definir si las diferentes ventanas de código reducen su tamaño cuando el proyecto también reduce su tamaño, en la zona **Explorador de proyectos**.

Las opciones de compilación **Compilar a petición** y **Compilar en segundo plano** permiten respectivamente compilar el proyecto antes de su ejecución y compilar el proyecto durante los períodos de inactividad.

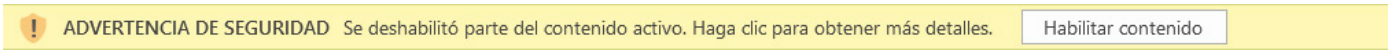
d. Acoplar



La pestaña **Acoplar** permite definir si las diferentes ventanas que se visualizan se anclan, es decir, quedan adheridas a un borde de la ventana.

Las bases de datos de Access y la seguridad

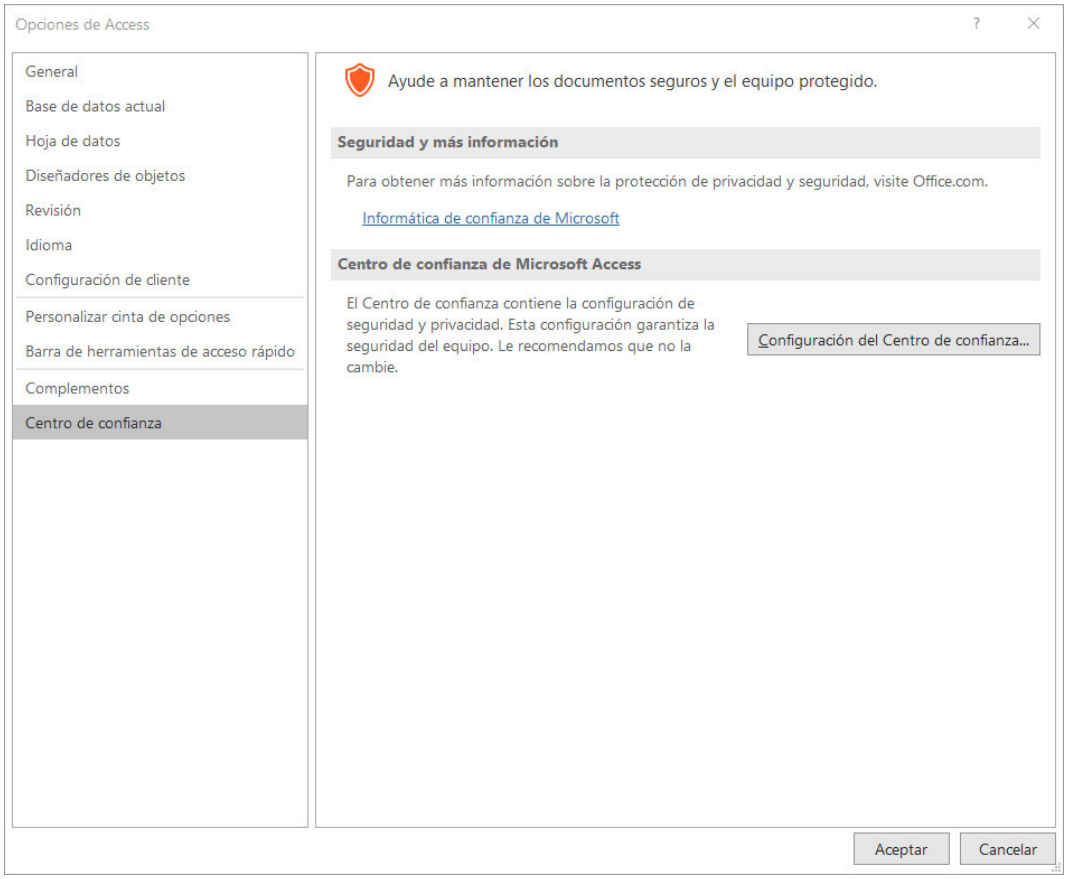
Cuando se instala Microsoft Access 2019 y se abren nuevas bases de datos, es normal que aparezca un mensaje de alerta sobre un fondo amarillo.



La administración de la seguridad incorporada en la aplicación permite evitar la ejecución incontrolada de programas que puedan ser peligrosos para la máquina. Esta sección va a tratar los diferentes niveles de seguridad que es posible modificar a su conveniencia en Microsoft Access 2019.

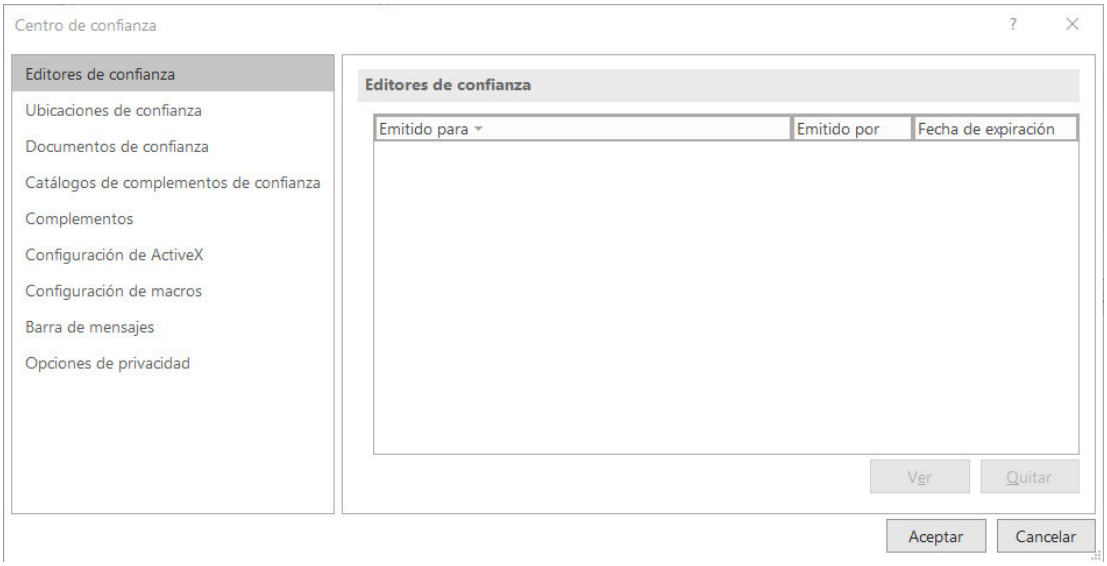
1. Los argumentos de seguridad

Para modificar los argumentos de seguridad, es necesario pasar por el **Centro de confianza**. Se puede acceder a él desde la pestaña **Archivo - Opciones**.



En la pestaña de navegación de la izquierda, seleccione **Centro de confianza** y haga clic en el botón **Configuración del Centro de confianza**.

Se muestra la siguiente interfaz:



2. Editores de confianza

a. ¿Qué es un editor de confianza?

Un editor es un desarrollador que ha creado una aplicación, una macro, un control ActiveX o un complemento para otras aplicaciones. Se considera que un editor es de confianza cuando responde a los siguientes elementos:

- El código está firmado por firma digital.
- La firma digital es válida.
- La firma digital no ha caducado.
- El certificado asociado a la firma digital fue emitido por una autoridad de certificación reconocida.
- El desarrollador que ha firmado el código es un editor de confianza.

b. ¿Cómo añadir un editor de confianza?

Cuando el usuario intenta abrir una base de datos por primera vez, se muestra un mensaje de alerta que pregunta si desea habilitar las macros o **Aprobar todos los documentos desde el editor**.

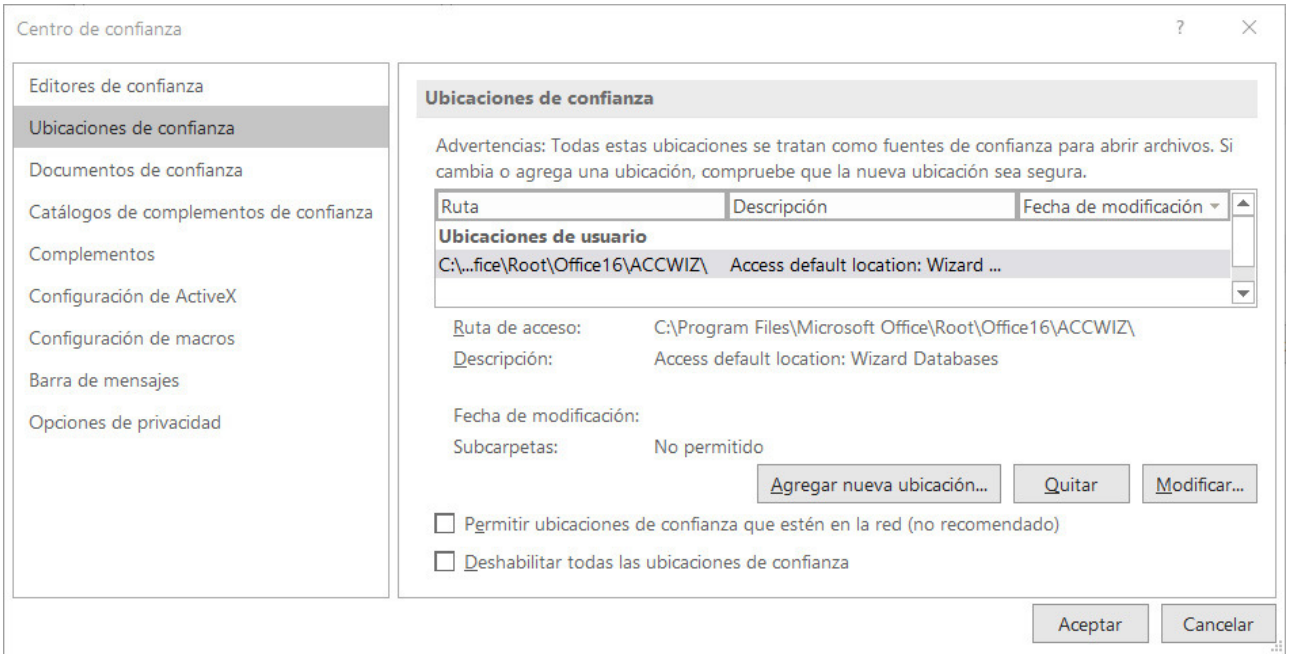
c. ¿Cómo acceder a la lista de editores de confianza?

La lista de los diferentes editores de confianza aparece en la ventana **Centro de confianza** (ver la sección Los argumentos de seguridad).

d. ¿Cómo eliminar un editor de confianza?

En la ventana **Centro de confianza**, es posible hacer clic en el botón **Eliminar**.

3. Ubicaciones de confianza

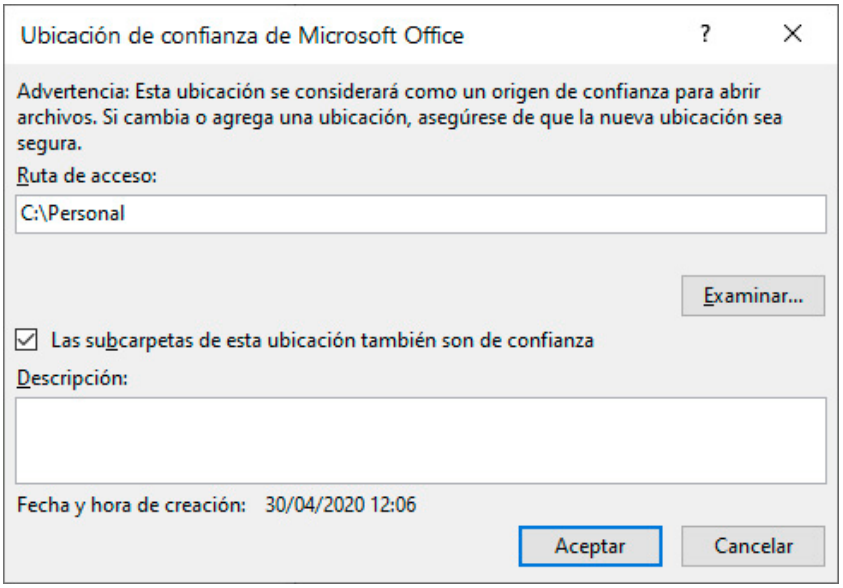


a. ¿Qué es una ubicación de confianza?

Una ubicación de confianza es una carpeta situada en un disco o en una unidad de red. Cualquier archivo guardado en una ubicación de confianza se puede abrir sin la intervención del **Centro de confianza**. Algunas ubicaciones se pueden definir como tales durante la instalación de Microsoft Office 2019.

b. ¿Cómo añadir una ubicación de confianza?

Para añadir una ubicación de confianza, es suficiente con hacer clic en el botón **Agregar nueva ubicación**. Sustituyendo la ruta de acceso o haciendo clic en el botón **Examinar**, es posible indicar la ubicación que se desea añadir, especificando si las subcarpetas contenidas en esta carpeta también se considerarán como aprobadas.

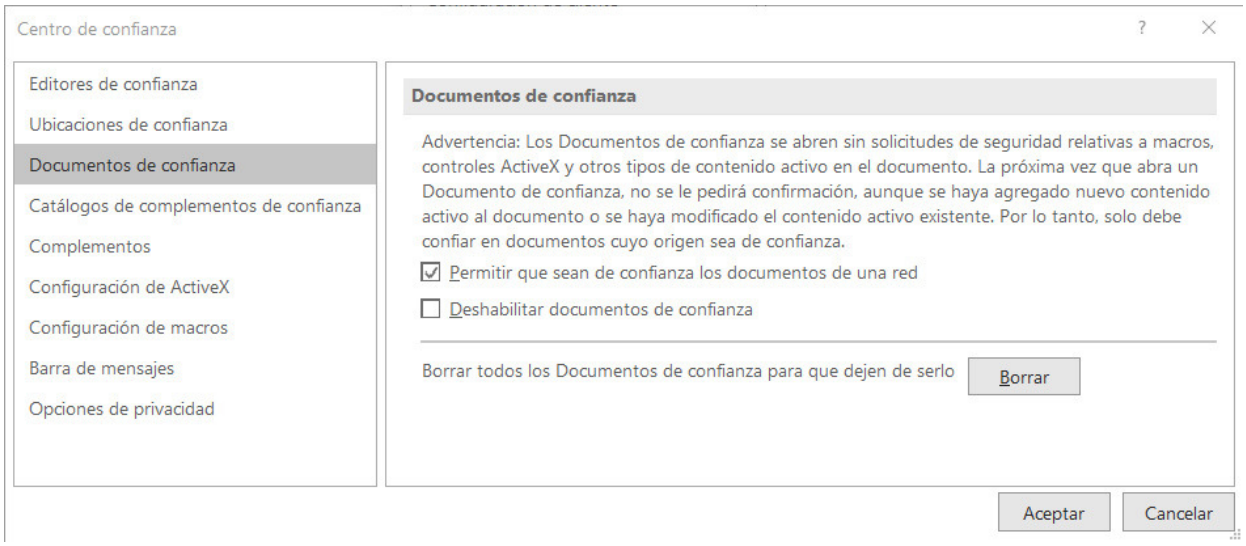


Para hacer referencia a una ubicación situada en la red, será necesario marcar la casilla de selección **Permitir ubicaciones de confianza que estén en la red (no recomendado)**.

c. ¿Cómo eliminar una ubicación de confianza?

Para eliminar una ubicación de confianza, es suficiente con seleccionarla en la interfaz y hacer clic en el botón **Quitar**.

4. Documentos de confianza



a. ¿Qué es un documento de confianza?

Aprobar un documento permite evitar que se muestre la barra de alerta durante su apertura. Por tanto, los documentos de confianza son los que considere como fiables.

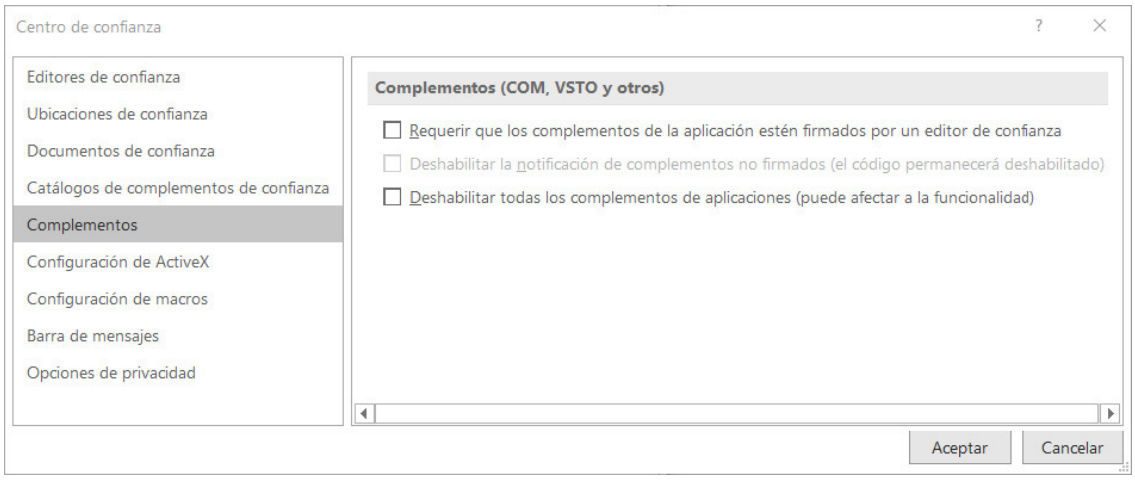
b. ¿Cómo aprobar un documento?

Durante la apertura de una nueva base de datos, se muestra la barra de alerta. Yendo a la pestaña **Archivo**, podemos ver la siguiente interfaz:

Haciendo clic en el icono **Habilitar contenido**, podemos seleccionar entre **Habilitar todo el contenido** u **Opciones avanzadas**. La primera solución aprueba el documento, la segunda solo lo aprueba una vez para la sesión actual.

Las bases de datos almacenadas en las ubicaciones temporales, como las descargadas de Internet, no se pueden aprobar.

5. Complementos



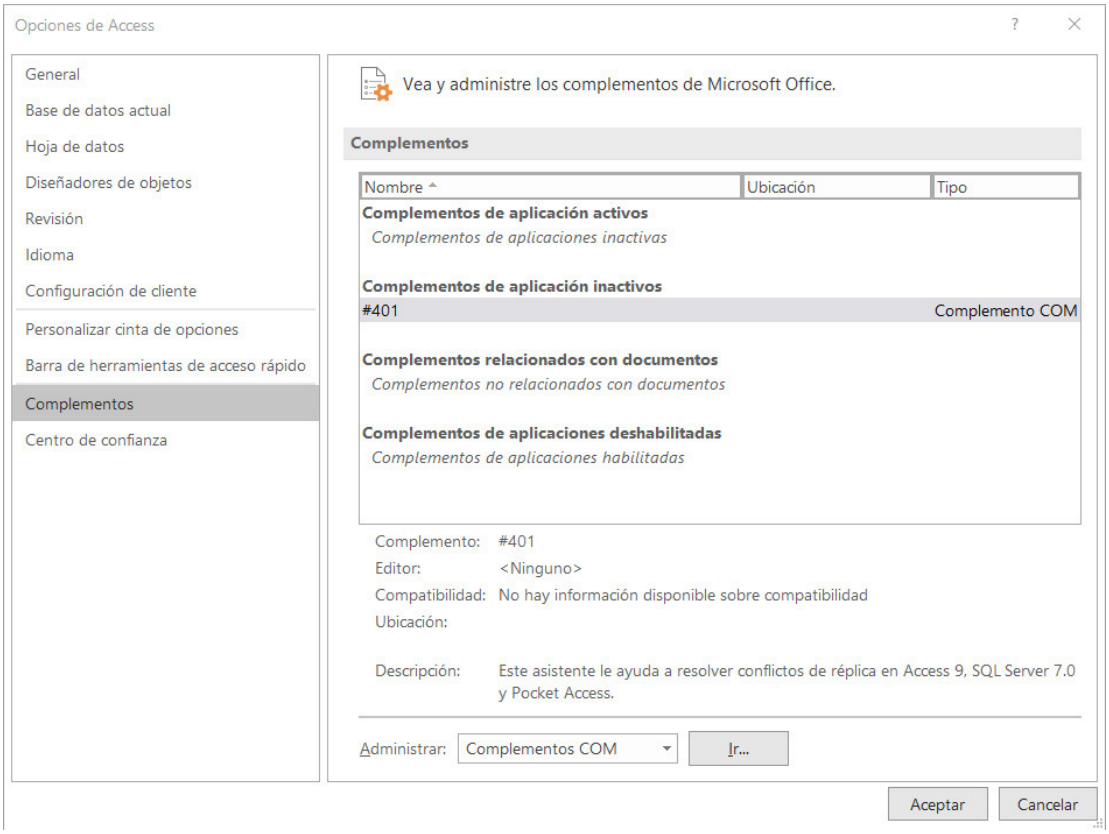
Las opciones de **Complementos** permiten definir el comportamiento de Microsoft Access 2019 durante la adición de un editor de confianza.

El hecho de **Requerir que los complementos de la aplicación estén firmados por un editor de confianza** fuerza la firma aprobada durante la adición. Si la firma del editor no es de confianza, Microsoft Access 2019 no cargará el complemento y la barra de mensajes mostrará una notificación indicando que el complemento se ha desactivado.

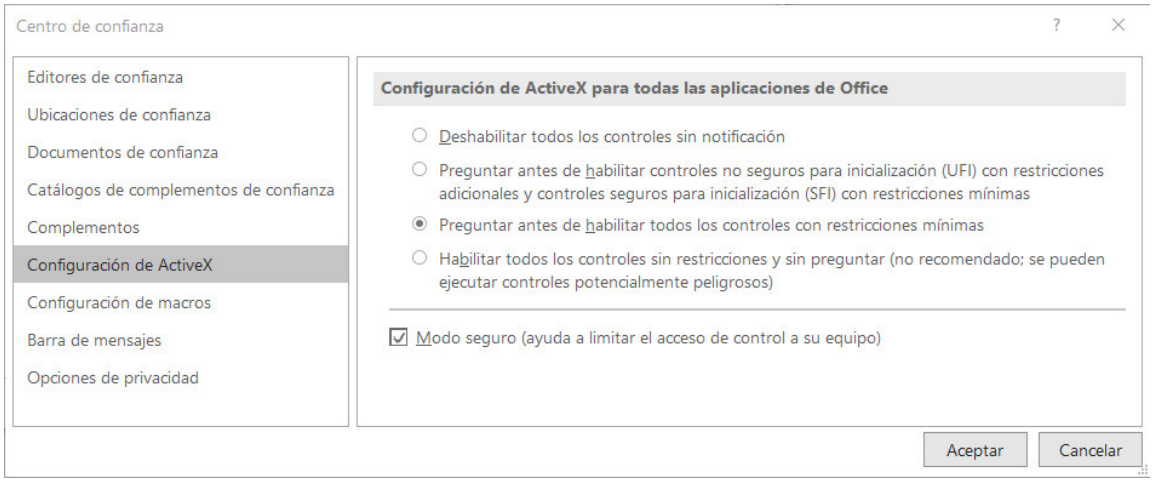
En caso de que esta casilla esté marcada, también es posible **Deshabilitar la notificación de complementos no firmados**, si el complemento no está firmado.

Si la casilla **Deshabilitar todos los complementos de aplicaciones** está marcada, las otras dos casillas de selección están deshabilitadas.

Es posible visualizar los complementos desde la pestaña **Archivo - Opciones** y seleccionando **Complementos**, en la pestaña de navegación.



6. Configuración de ActiveX



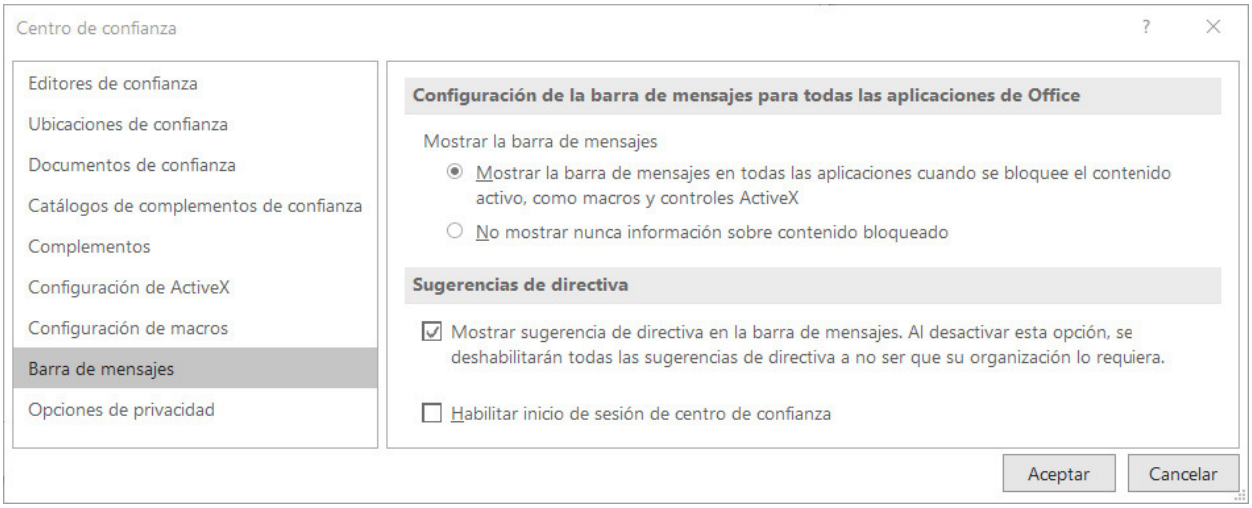
Los argumentos de ActiveX se pueden gestionar durante su activación en Microsoft Access 2019. Por tanto, es posible:

- **Deshabilitar todos los controles sin notificación**.
- **Preguntar antes de habilitar controles no seguros...**
- **Preguntar antes de habilitar todos los controles...**
- **Habilitar todos los controles sin restricciones y sin preguntar...**

7. Configuración de macros

De la misma manera que es posible gestionar los argumentos de ActiveX, los argumentos de las macros permiten definir su grado de activación durante la apertura de una base de datos que incluye macros VBA.

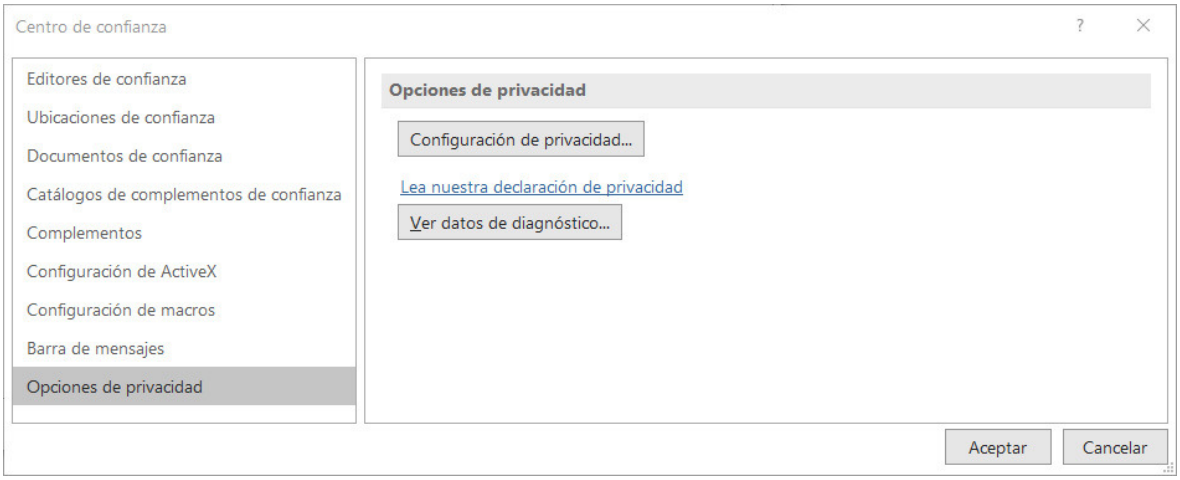
8. Barra de mensajes



La barra de alerta de los mensajes se puede mostrar o no durante la apertura de una base de datos en Microsoft Access 2019.

En esta pestaña, también es posible ver los consejos de estrategia en la barra de mensajes (ejemplo al inicio de este capítulo), y activar el registro de trazas del **Centro de confianza**.

9. Opciones de privacidad

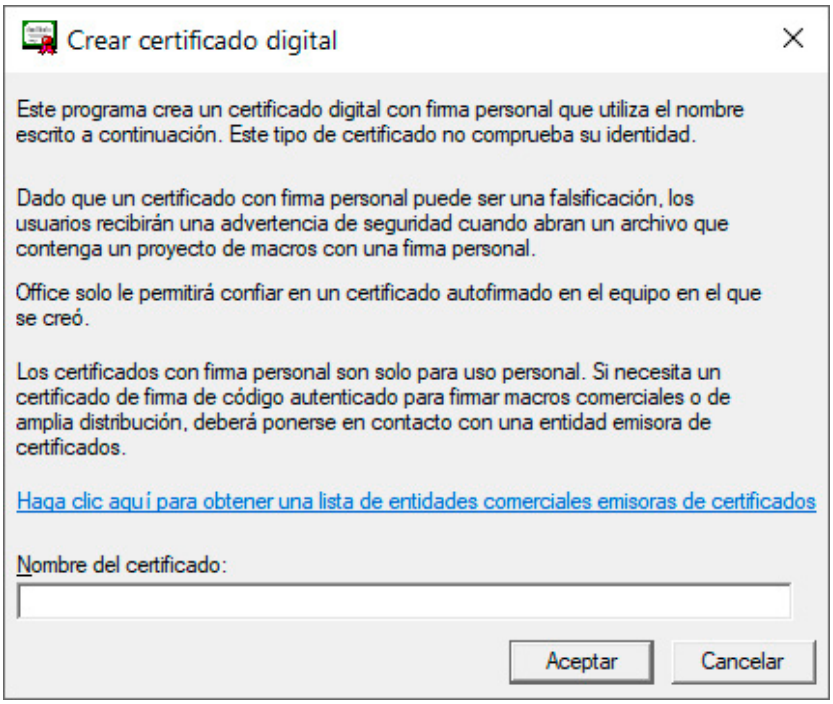


En la pestaña **Opciones de privacidad**, se muestran los diferentes elementos relacionados con la difusión y descarga de información desde y hacia el exterior de la aplicación.

10. Microsoft Access y el paquete firmado

a. ¿Cómo crear un certificado?

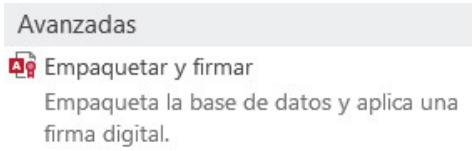
En primer lugar, para crear un paquete firmado, es necesario crear un certificado. Para esto puede ir al menú **Inicio - Todos los programas - Microsoft Office - Herramientas de Microsoft Office - Certificado digital para los proyectos VBA**. También puede ejecutar la aplicación **SELFCERT.EXE**, distribuida con Microsoft Access 2019, directamente desde su ubicación en la máquina. Normalmente, su ubicación es la siguiente: C:\Program Files (x86) \Microsoft Office\root\Office17



Introduzca el nombre del certificado y haga clic dos veces en el botón **Aceptar**.

b. ¿Cómo crear un paquete en Access?

En la pestaña **Archivo - Guardar en**, es necesario seleccionar el formato **Avanzadas - Empaquetar y firmar**.



Seleccione el certificado anteriormente creado y haga clic en **Aceptar**.

Microsoft Access 2019 propondrá guardar la base de datos con una extensión **accdc**, que corresponde al **Paquete firmado Microsoft Access**.

Una programación secuencial

VBA es un lenguaje de programación secuencial, es decir, que las instrucciones que se codifican se deben ejecutar en su orden de aparición en el programa. Una instrucción solo se puede ejecutar si la que la precede se ha ejecutado.

Un perro seguirá las instrucciones en su orden de aparición: "Sentado", "Arriba", "Tumbarse".

Sintaxis posibles

En VBA, las instrucciones se separan por un retorno de carro o por el signo dos puntos ":".

Por ejemplo, consideremos el siguiente programa:

```
Instrucción_1
Instrucción_2
Instrucción_3
```

La ejecución de este programa implicará la ejecución de `Instrucción_1`, después `Instrucción_2` y para terminar `Instrucción_3`. El programa se podría haber escrito de manera equivalente como sigue:

```
Instrucción_1: Instrucción_2: Instrucción_3
```

Si una instrucción es demasiado larga para escribirse en una única línea, o si desea dividirla en varias líneas para simplificar la lectura, es posible pasar a la siguiente línea usando el símbolo subrayado (underscore) " _ ". De esta manera, la ejecución del programa arranca la ejecución de la instrucción `Instrucción_a`:

```
Instruc_
ción_A
```

Estructura de un programa

Un programa VBA se descompone en una serie de procedimientos y funciones, escritos con el objetivo de realizar una o varias operaciones. Las instrucciones se deben codificar dentro de estos procedimientos o funciones. Se trata de declarar procedimientos (y funciones).

Por tanto, en un módulo, el programa está formado por una serie de declaraciones de procedimientos y funciones. Una vez estos procedimientos se declaren, se ejecutarán cuando el programa los llame.

Para ejecutar un programa VBA, se ejecuta una macro, que es un procedimiento particular, la cual contiene las instrucciones que hacen la llamada a otras funciones y procedimientos. Para poder funcionar, un programa VBA debe tener, al menos, una macro declarada.

A continuación se muestra un ejemplo de programa VBA dentro de un módulo.

```
Sub Procedimiento_AProcedimiento_A()
...
End Sub
Sub Procedimiento_BProcedimiento_B()
...
End Sub
Function Funcion_C...
...
End Function
Sub Macro_1()
Procedimiento_AProcedimiento_A
...
End Sub
```

En este ejemplo, podemos comprobar la declaración de los procedimientos `Procedimiento_A` y `Procedimiento_B`, de la función `Funcion_A` y de la macro `Macro_1`. La primera instrucción de `Macro_1` es una llamada al procedimiento `Procedimiento_A`. Por tanto, la primera operación ejecutada durante la ejecución de la macro `Macro_1` será la ejecución del procedimiento `Procedimiento_A`.

Preste atención, porque le nombre que asigne a las macros, procedimientos y funciones, no tiene impacto sobre su naturaleza. Por ejemplo, puede tener:

```
Sub MiFuncion()
...
End Sub
```

Aunque es evidente que esto no le facilitará la vida.

Las variables

Las variables son recipientes que permiten almacenar información en todo momento, durante la ejecución de un programa, y utilizarla en cualquier otro momento. En VBA, una variable queda definida por medio de dos atributos:

- Su nombre, que se utilizará para acceder a la información que contiene; para obtener información adicional sobre las convenciones de nomenclatura, consulte Convenciones de nomenclatura y tipografía del código VBA.
- El tipo de dato que almacena.

1. La sintaxis de declaración

Como sucede con las funciones y los procedimientos, para que una variable se pueda llamar y utilizar, es necesario declararla. La sintaxis de declaración de una variable es la siguiente:

```
Dim NombreDeVariable As TipoVariable
```

La palabra clave `Dim` sirve para declarar una variable. Viene seguida del nombre de la variable. Después de la palabra clave `As`, encontramos el tipo de la variable. Hay varios tipos de datos, que es posible manipular en VBA.

Al presentar a su perro, se indica su nombre y su raza ("Milou, fox-terrier"), y para usar una variable, se llama por su nombre.

2. Los tipos de datos

Hay varios tipos de datos (también tenemos las constantes).

De la misma manera que existen varias razas de perros.

a. Los tipos numéricos

Los valores enteros

El tipo `Byte` permite almacenar un entero comprendido entre 0 y 255 (almacenado en 8 bits, es decir, 1 byte).

```
Dim b As Byte
b = 10
```

El tipo `Integer` contiene un entero comprendido entre -32.768 y 32.767 (en 16 bits).

```
Dim i As Integer
i = 315
```

El tipo `Long` contiene un entero comprendido entre -2 147 483 648 y 2 147 483 647 (en 32 bits).

```
Dim l As Long
l = 226
```

Los valores reales (decimales)

El tipo `Single` contiene un valor real comprendido entre -3,4028823 $\cdot 10^{38}$ y -1,401298 $\cdot 10^{-45}$ para los números negativos y entre 1,401298 $\cdot 10^{-45}$ y 3,402823 $\cdot 10^{38}$ para los números positivos.

```
Dim s As Single
s= 1.8
```

El tipo `Double` contiene un valor real comprendido entre -1,797693134 86231 $\cdot 10^{308}$ y -4,94065645841247 $\cdot 10^{-324}$ para los números negativos y entre 4,94065645841247 $\cdot 10^{-324}$ y 1,79769313486231 $\cdot 10^{308}$ para los números positivos.

```
Dim d As Double
d = 1.23456
```

El tipo `Currency` contiene un valor comprendido entre -922 337 203 685 477,5808 y 922 337 203 685 447,5807 (64 bits). Este tipo de datos se utiliza fundamentalmente para los cálculos monetarios o los cálculos con coma fija.

```
Dim c As Currency
c = 3.142
```

Observe que el separador utilizado en VBA para separar la parte entera de la parte decimal es el punto ".".

b. Los otros tipos de datos

Los valores booleanos

El tipo `Boolean` puede contener `True` (verdadero) o `False` (falso).

```
Dim b As Boolean
b = False
```

Las cadenas de caracteres

El tipo `String` permite almacenar cadenas de caracteres (de texto). Es posible declarar cadenas de longitud fija o variable. Una cadena de longitud fija puede tener hasta 65.536 caracteres, mientras que una cadena de longitud variable puede contener más de de 2 millones (2^{31} = 2.147.483.648). Los valores de texto están encerados entre comillas «"».

```
Dim str1 As String
Dim str2 As String * 10
str1 = "Hello"
```

Aquí se ha declarado una cadena de caracteres `str1` de longitud variable, y una cadena de caracteres `str2` de una longitud de 10 caracteres.

Las fechas y horas

El tipo `Date` permite almacenar fechas y horas, así como duraciones. La sintaxis de las fechas se escribe entre almohadillas "#", en formato #MM/DD/AAAA [hh:mm:ss AM/PM]#, como en el siguiente ejemplo:

```
Dim dt As Date
dt = #7/16/1969 1/32/00 PM#
dt = #7/14/1789#
```

El tipo Variant

El tipo `Variant` es un tipo de datos que agrupa las características del resto de los tipos de datos. Está codificado con 16 bits, aunque también puede contener una cadena de caracteres con 22 bytes. Cuando no se ha definido ningún tipo de variable durante la declaración de una variable, se da como tipo por defecto en VBA `Variant`.

```
Dim v as Variant
v = 1.5 / V = "Neil Amstrong"
```

3. Las declaraciones múltiples de variables

Si lo desea, es posible declarar varias variables en la misma línea. Tiene dos soluciones. La primera consiste en utilizar la sintaxis con ":".

```
Dim a As Integer: Dim b As String: Dim c As Date
```

La segunda consiste en separar las variables y sus tipos, usando comas.

```
Dim a As Integer, b As String, c As Date
```


Atención, la siguiente sintaxis declara dos variables, `a` y `b`, de tipo `Variant`, y una variable de tipo `String`.

```
Dim a, b, c As String
```

4. Asignación de un valor a una variable

En VBA, para asignar un valor a una variable, la sintaxis es la siguiente:

```
Variable = valor_asignado
```

Como sigue:

```
Dim a As Integer
a = 1
```

La variable `a` toma el valor asignado, 1.

Las constantes

1. Las constantes de usuario

Las constantes de usuario permiten asignar una etiqueta a un valor fijo. El uso de constantes permite facilitar la programación, fundamentalmente en el marco de una eventual actualización de valores. Por ejemplo, podemos utilizar una constante para almacenar el número de días por semana. La sintaxis genérica es la siguiente:

```
Const NombreConstante As TipoConstante = ValorConstante
```

Ejemplo:

```
Const NUMERO_DE_DIAS_POR_SEMANA As iNTEGER = 7
```

La constante `NUMERO_DE_DIAS_POR_SEMANA`, de tipo número entero, tiene el valor 7 en todo momento, mientras se utilice en el programa.

Una vez que se ha definido el valor de la constante, no se puede modificar.

2. Las constantes de Office

En VBA, hay un determinado número de constantes de Office que están directamente relacionadas con las diferentes aplicaciones (Access, Excel, etc.). Estas constantes tienen valores predefinidos y sus nombres respetan una nomenclatura: `AplicacionNombre` (`ac` para Access, `xl` para Excel, `vb` para VB, etc.). A continuación se muestran algunos ejemplos de constantes de VBA:

Herramienta/aplicación	Ejemplos de constantes
Access	`acGreaterThan`, `acPage`, `acTextBox`
Excel	`xlUp`, `xlLabel`, `xlPasteFormats`
VB	`vbBlue`, `vbMinimizedFocus`, `vbFriday`

Las matrices

Hasta ahora, hemos visto la manera de almacenar un valor en una variable. También es posible almacenar varios valores en varias variables. Finalmente, es posible almacenar varios valores dentro de una única variable usando una matriz. Crear una matriz de n elementos implica crear n variables diferentes al mismo tiempo.

Las variables de tipo matriz sirven para almacenar grupos de datos del mismo tipo. Se podrá acceder a cada elemento de la matriz utilizando un número secuencial. Una modificación de uno de los elementos de la matriz no modifica el resto de los elementos de la matriz.

Una matriz puede estar compuesta por una o varias dimensiones, cada dimensión se define por medio de unos límites inferior y superior. Estos límites pueden ser fijos o dinámicos. Las matrices solo pueden cambiar de dimensión durante la ejecución del programa, cuando son dinámicas.

Una matriz siempre necesitará 20 bytes de memoria, más 4 bytes por dimensión y el número de bytes necesarios según el tipo de valores almacenados.

1. Las matrices de tamaño fijo

Se determina una matriz de tamaño fijo cuando se define durante la declaración de la variable.

```
Option Base 0
Dim Matriz(4) As Integer
```

El índice 4 especifica el tamaño de la matriz. La numeración de los elementos puede comenzar en 0 o 1, según la definición de la instrucción `Option Base`. Si no se ha indicado ninguna información, el valor por defecto es `Option Base 0`. `As Integer` determina el tipo de datos que contiene la matriz.

Por tanto, se tratará de una matriz de 5 valores enteros (de 0 a 4).

Se accede en modo lectura/escritura a las variables que se indican mediante su identificador en el código.

```
Matriz(2) = 3
```

Observación: atención, las dos variables siguientes no almacenarán la misma cantidad de información.

```
Dim Mat1(5) As String
Dim Mat2 As String * 5
```

`Mat1` es una matriz de 5 cadenas, mientras que `Mat2` es una cadena de 5 caracteres.

También es posible definir los límites inferior y superior, que servirán para identificar los elementos.

```
Dim Matriz3(-3 To 5) As String
```

2. Las matrices dinámicas

Durante la declaración de la variable, si no se ha indicado ninguna dimensión, se dice que la matriz es dinámica. Podemos utilizar matrices dinámicas si no sabemos a priori la dimensión que alcanzará la matriz. Es posible volver a definir más adelante las dimensiones de la matriz con la instrucción `ReDim`.

```
Dim Matriz1() As String
Redim Matriz1(5)
```

La reasignación de dimensiones con `ReDim` elimina los datos ya existentes en la matriz. Para conservarlos, puede utilizar la palabra clave `Preserve`, que conservará los datos en la matriz.

```
Redim Preserve Matriz1(6)
```

3. Las matrices multidimensionales

Las variables que se han visto hasta ahora, solo tenía una dimensión. Por supuesto, es posible trabajar con matrices multidimensionales, en las que las dimensiones se separan por comas durante su declaración o su reajuste. Las dimensiones siguen la misma sintaxis que en caso de las matrices monodimensionales.
El número de dimensiones se limita a 60.

```
Dim Matriz3D(1, 2 To 8, 3) As String
```

Por tanto, el número de elementos contenidos en una matriz se corresponde con el producto de los tamaños de cada dimensión.

4. Los Array

La función **Array** sirve para crear una lista de datos de tipos idénticos o diferentes, separados por comas. La sintaxis general es la siguiente:

```
Option Base 1
Sub Macro_01
    Dim v As Variant
    v = Array(1, "Dos", 3.0)
    MsgBox v(2) 'muestra "Dos"
End Sub
```

Si quiere que Option Base no tenga ningún efecto sobre la función Array, puede utilizar **VBA.Array** con la siguiente sintaxis:

```
Option Base 1
Sub Macro_02()
    Dim MiArray As Variant
    MiArray = VBA.Array("lunes", "martes", "miércoles")
    MsgBox MiArray(0) 'Muestra "lunes"
```

```
    MsgBox MiArray (2) ' Muestra "miércoles"
End Sub
```

5. Vaciar o liberar memoria de una matriz - Erase

Para vaciar una matriz de tamaño fijo o dinámico, se utiliza el procedimiento **Erase**.

```
Option Base 1
Sub Macro_03()
    Dim MiArray(2) As String
    MiArray(1)="Hombre"
    MiArray(2)="Mujer"
    Erase MiArray
    MsgBox MiArray(1) 'Muestra una cadena vacía’
End Sub
```

Erase no libera memoria de una matriz de tamaño fijo, sino de una matriz dinámica. Sobre una matriz de tamaño fijo, solo se vacían sus valores.

Los operadores

Existen varios operadores en VBA que se pueden utilizar, según los tipos de variables. Los resultados de los operadores se representan en los siguientes ejemplos, a la derecha del símbolo =.

1. Los operadores numéricos

+, -, *, /: operadores matemáticos de las operaciones básicas

+: suma

```
1 + 4 = 5
```

-: resta

```
6 – 2 = 4
```

*: multiplicación

```
3 * 6 = 18
```

/: división

```
35 / 5 = 7
```

\: para la división entera

```
8 \ 3 = 2
```

^: potencia

```
3 ^ 4 = 3 * 3 * 3 * 3 = 81
```

mod: módulo

El módulo devuelve el resto de la división entera de un número entre otro.

```
25 mod 3 = 1’ (car 25 = 3 * 8 + 1)
```

2. Los operadores de cadenas de caracteres

&: concatenación

Para concatenar dos cadenas de caracteres entre ellas, se usa el símbolo *ampersand* " & ". Observe que el operador + también puede funcionar, pero se pueden producir errores si utiliza este operador, ya que se usa para las variables numéricas.

```
"Hello " & "world" = "Hello world"
```

3. Los operadores de fechas

+: suma

```
#01/19/2018# + 365 = #01/19/2019#
```

-: resta

```
#02/15/2016# - 3 = #02/12/2016#
```

4. Los operadores booleanos, operadores lógicos

And: operador Y

```
True And False = False
```

Or: operador O

```
True Or False = True
```

Not: operador NO

```
Not True = False
```

Xor: operador O EXCLUSIVO

Xor permite determinar si es una u otra, pero no las dos al mismo tiempo.

```
True Xor False = True
False Xor False = False
True Xor True = False
```

Eqv: operador EQUIVALENTE

Eqv devuelve verdadero si dos valores son equivalentes.

```
True Eqv True = True
False Eqv True = False
```

5. Los operadores de comparación

>: estrictamente superior a

```
4 > 2
```

<: estrictamente inferior a

```
3 < 8
```

>=: superior o igual a

```
5 >= 5
```

<=: inferior o igual a

```
3 <= 3
```

=: igualdad

```
(8 = 9) = False
```

<>: diferente de

```
5 <> 6
```

6. Orden de prioridad de los operadores

Cuando aparecen varios operadores en una misma instrucción, se asigna una prioridad a algunos operadores respecto al orden de ejecución en relación con otros operadores. Los operadores numéricos se tratan en primer lugar, después el operador de concatenación, continuando por los operadores de comparación, para terminar con los operadores lógicos.

El orden de los operadores se da partiendo del más importante (ejecutado en primer lugar) al menos importante:

Numérico: ^, *, /, \, mod, +, -

Concatenación: &

Comparación: todos los operadores tienen la misma prioridad, se evalúan en el orden de aparición en la instrucción (por tanto, de izquierda a derecha).

Lógica: `Not`, `And`, `Or`, `Xor`, `Eqv`

Aquí el operador = es solamente el operador de comparación de igualdad, no el operador de asignación de valor.

Los procedimientos

Un procedimiento es una serie de instrucciones que modifica el entorno, pero no devuelve ningún valor al final. Un procedimiento puede estar directamente codificado en un módulo, así como relacionado con un evento sobre un objeto de Access (formulario, botón, etc.). Algunos procedimientos se pueden crear automáticamente a partir de una macro de Access o del asistente de Access.


Para nuestro perro, "Sentarse" o "Ir a la caja" son procedimientos.

1. Declaración de un procedimiento

La declaración de un procedimiento en VBA se hace con la palabra clave `Sub`, según la siguiente sintaxis:

```
[Private o Public] Sub Nombre_de_Procedimiento([argumento_1 As
Tipo_Argumento, ...])
    'Comentarios
    Instrucciones
End Sub
```

`Nombre_de_Procedimiento` es el nombre del procedimiento, y `argumento_1`, el nombre de un argumento de tipo `Tipo_Argumento`. La expresión entre corchetes `[Private o Public]` significa que podemos añadir si queremos una de las palabras clave `Public` o `Private`, utilizadas para definir el carácter público o privado del procedimiento. Si no se utiliza ninguna de estas palabras clave, el procedimiento será público por defecto.

Un procedimiento público se puede llamar desde cualquier otra ubicación del programa. Un procedimiento privado solo se puede llamar desde otro procedimiento (o función), que se sitúa en el mismo módulo del programa.

Para hacer el código más comprensible para un usuario, es posible añadir comentarios. Estos comentarios no se interpretarán por la máquina durante la ejecución del procedimiento, sino que solamente serán visibles por el desarrollador. Los comentarios en VBA se escriben después de un apóstrofo (’), como en la sintaxis del ejemplo anterior. También pueden aparecer después de la palabra clave `Rem`. Además de poder comentar los procedimientos, algunas veces se comentan algunas instrucciones para explicar el funcionamiento y hacer que el programa sea más legible.

A continuación veremos un ejemplo de procedimiento `Hola`, que muestra el mensaje "`HolaWorld!`":

```
Sub Hola()
    'Muestra un mensaje al usuario
    MsgBox "Hola World!"
End Sub
```

El procedimiento solo contiene una única instrucción `MsgBox` "`Hola World!`". Esta instrucción llama a la función VBA `MsgBox`, que muestra el mensaje que se pasa como argumento (aquí "`Hola World!`"), a través de un cuadro de diálogo (ver la sección La función MsgBox, en este capítulo).

El siguiente procedimiento `MuestraPalabra` muestra la cadena de caracteres que se pasa como argumento:

```
Sub MuestraPalabra(palabra As String)
```

```
    'Muestra la palabra
    MsgBox palabra
End Sub
```

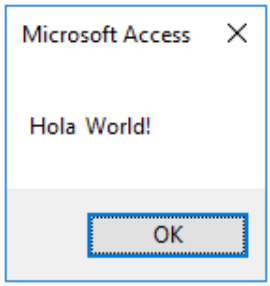
2. Llamada a un procedimiento

Para ejecutar un procedimiento, es suficiente con llamarlo escribiendo su nombre, seguido de los posibles argumentos separados por una coma. Por tanto, la llamada de un procedimiento sin argumentos consiste en escribir su nombre.

Por ejemplo, la llamada del procedimiento `Hola` se realiza con la siguiente instrucción:

```
Hola
```

Esta llamada provoca la visualización de un cuadro de diálogo que contiene el mensaje "`Hola World!`" en la interfaz de Access, como sigue:



3. Macro

Una macro de VBA (no confundir con una macro de Access) es un procedimiento sin argumento. El procedimiento `Hola` es, por ejemplo, una macro. Las macros son los únicos procedimientos de VBA que se pueden ejecutar en Access.

Las funciones

Una función es una serie de instrucciones que devuelve un único valor, a saber, el resultado de una expresión.

Nuestro perro va a traernos un periódico si le pedimos "Ve a buscar un periódico".

1. Declaración de una función

La declaración de una función en VBA se hace con la palabra clave `Function`, según la siguiente sintaxis:

```
[Private o Public] Function Nombre_Funcion([argumento_1
As Tipo_Argumento, ...]) As Tipo_Retorno
    'Comentarios
    Instrucciones
    Nombre_Funcion = expresión
End Function
```

`Nombre_Funcion` es el nombre de la función y `argumento_1` el nombre de un argumento de tipo `Tipo_Argumento`. Una función se puede declarar con cero, uno o varios argumentos. `As Tipo_Retorno` permite determinar el tipo de valor que la función devolverá. La expresión entre corchetes `[Private o Public]` significa que podemos añadir si queremos, una de las palabras clave `Public` o `Private`, utilizadas para definir el carácter público o privado de la función. Si no se utiliza ninguna de estas palabras clave, la función será pública por defecto.

A continuación se muestra un ejemplo de función `El_Doble`, que recibe un entero x como argumento y devuelve el doble de su valor (x*2).

```
Function El_Doble (x As Long) As Long
    'Función que devuelve el valor de x 2 veces
    El_Doble = x * 2
End Function
```

De esta manera, dado un entero `x`, la función `El_Doble` devuelve el valor `x * 2`, lo que se corresponde con el doble del valor de `x`.

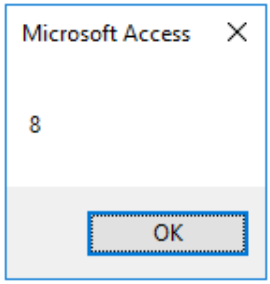
2. Llamada a una función

Para ejecutar una función, es suficiente con llamarla escribiendo su nombre, seguido de los posibles argumentos separados por una coma. Por tanto, la llamada a una función sin argumentos consiste en escribir su nombre, seguido de paréntesis vacíos.

Por ejemplo, para visualizar el doble del valor de 4, podemos escribir una macro `Ver_El_Doble_De_4`, que muestra el resultado de la llamada a la función `El_Doble`, recibiendo el valor 4 como argumento:

```
Sub Ver El_Doble_De_4()
    'Muestra el valor del doble de 4 (4 * 2)
    MsgBox El_Doble(4)
End Sub
```

Esta llamada provoca la visualización de un cuadro de diálogo que contiene el mensaje 8 en la interfaz de Access, como sigue:



Pasar argumentos por valor y por referencia

1. Pasar argumentos

Durante la llamada a una función o procedimiento, algunas veces el programa necesita información para ejecutarse; es el caso, por ejemplo, de la función `El_Doble`, que necesita tener el valor `x` para devolver el doble.

Nuestro perro necesita saber qué buscar: "busca…la pelota".

La información proporcionada al procedimiento o función, se llama parámetros (también se utiliza la palabra argumento).

La pelota es el argumento para el perro.

La sintaxis general del paso de argumentos es la siguiente:

```
[Optional] [ByRef o ByVal] [ParamArray] <nombre_argumento>
[As Tipo_Argumento]
```

Cada una de las palabras clave tiene un impacto en la naturaleza del argumento.

a. Los argumentos obligatorios

Cuando un argumento es obligatorio para que el programa cumpla su función, la sintaxis es la siguiente:

```
Nombre_argumento As Tipo_Argumento
```

Por ejemplo, si se retoma la función `El_Doble` para que la función se ejecute, se debe proporcionar obligatoriamente el argumento `x`:

```
x as Long
```

b. Los argumentos opcionales

Si el argumento se puede proporcionar opcionalmente, se precede de la palabra clave `Optional`. Por ejemplo, un procedimiento que debe mostrar el nombre y apellidos de una persona, así como el nombre de pila, se escribirá como sigue:

```
Nombre As String, Apellidos As String, Opcional Nombre_De_Pila As String
```

`Nombre_De_Pila` es opcional, ya que puede que no tenga.

El perro sabe que estamos "fuera" cuando le decimos "salimos".

Es posible saber si el argumento se ha recibido o no durante la llamada a la función o procedimiento, con la función VBA `IsEmpty`, cuya sintaxis es la siguiente:

```
IsEmpty(Nombre_Argumento)
```

Esta instrucción devuelve `True` si se omite el argumento `Nombre_Argumento`, y `False` en caso contrario.

c. Pasar por referencia

Usando la palabra clave `ByRef`, la función o procedimiento al que se pasa el argumento, recibe «realmente» la variable y, si se modifica en el cuerpo de la función o del procedimiento, su valor también se modificará en la salida del procedimiento llamador.

Si damos un hueso a nuestro perro, es de esperar que el hueso sufra algunos desperfectos.

Por ejemplo, en el marco del siguiente programa:

```
Sub Prueba()
    Dim Valor As Integer
    Valor = 10
    MsgBox Valor
    Proc_5_Veces_ByRef Valor '10
    MsgBox Valor '50
End Sub

Sub Proc_5_Veces_ByRef(ByRef x As Integer)
    x= x * 5
End Sub
```

La variable `Valor` mostrada tendrá los valores sucesivos siguientes: `10` y `50`.

d. Pasar por valor

Usando la palabra clave `ByVal`, la función o procedimiento al que se pasa el argumento, recibe una «copia» de la variable. Por tanto, el valor real de la variable no se puede modificar.

El hueso de goma no se arriesga demasiado.

Por ejemplo, en el marco del siguiente programa:

```
Sub Prueba2()
    Dim Valor As Integer
    Valor = 10
    MsgBox Valor '10
    Proc_5_Veces_ByVal Valor
    MsgBox Valor '10
End Sub

Sub Proc_5_Veces_ByVal(ByVal x As Integer)
    x= x * 5
End Sub
```

La variable `Valor` mostrada tendrá los valores sucesivos siguientes: `10` y `10`.

- El uso de las palabras clave `ByRef` y `ByVal` es opcional en VBA. Si no se concreta ninguna palabra clave `ByRef/ByVal`, el paso por defecto es por referencia `ByRef`.

- Caso particular con `ByRef`: si la variable se encierra entre paréntesis, se tratará como si el paso fuera por valor, y la variable no se cambiará en el procedimiento llamador.

```
Sub Prueba3()
    Dim x As Integer
    x = 1
    Mas_Uno x  '2
    MsgBox x
    Mas_Uno (x)
    MsgBox x'todavía 2
End Sub

Sub Mas_Uno(ByRef y As Integer)
    y = y + 1
End Sub
```

e. Los valores por defecto

Cuando un argumento es opcional, es posible asignarle un valor por defecto en la declaración de la función o procedimiento. La sintaxis utilizada es la siguiente:

```
Sub Prueba4()
    Muestra_Me "El periódico" 'El periódico
    Muestra_Me "Las zapatillas" 'Las zapatillas
    Muestra_Me 'Mis dientes
End Sub
Sub Muestra_Me(Optional TeMuestro As String = " Mis dientes")
    MsgBox TeMuestro
End Sub
```

La macro `prueba` mostrará sucesivamente `" El periódico"`, `"Las zapatillas"` y `" Mis dientes"`.

f. Los argumentos nombrados

Cuando se llamada a una función o procedimiento, los argumentos se deben pasar en el orden de su aparición en la declaración de la función o del procedimiento.

```
Sub Prueba5()
    Ver_Detalle " Mogwaï", "Pastor alemán", #07/01/2016#
End Sub
Sub Ver_Detalle (Apellido As String, Opcional Raza As String,
Opcional Fecha_Nacimiento As Date)
    'Instrucciones
```

```
End Sub
```

Si no se indica un argumento, será necesario marcar su ubicación vacía, como en el siguiente ejemplo:

```
Sub Prueba6()
    Ver_Detalle "Mogwaï", , #07/01/2016#
End Sub
```

Esta no es la única solución para pasar estos argumentos. También es posible pasar los indicando su nombre, seguido de los caracteres ":= ".

```
Sub Prueba7()
    Ver_Detalle Apellido:=" Mogwaï",Raza:= "Pastor alemán", _
    Fec_Nacimiento:=#07/01/2016#
End Sub
```

El hecho de pasar por el nombre del argumento permite mencionar solo los argumentos que se quiere introducir, así como indicar los argumentos en el orden que queramos, como en el siguiente ejemplo:

```
Sub Prueba8()
    Ver_Detalle Fecha_Nacimiento:=# 07/01/2016#, _
    Apellido:="Mogwaï "
End Sub
```

g. ParamArray

La palabra clave `ParamArray` permite especificar si un argumento de procedimiento o función espera una matriz de elementos de un tipo especificado. El tipo `ParamArray` solo se puede utilizar como último argumento de una lista. Permite pasar un nombre arbitrario de argumentos.

Tomemos como ejemplo un procedimiento `Familia_Al_Completo`, que mostrará en la ventana el nombre de una familia, así como el número de animales de la familia que la componen. Este procedimiento recibe como argumento el nombre de la familia, seguido de la lista de los animales que la componen, y se declarará de la siguiente manera:

```
Sub Familia_Al_Completo(ByVal NombreFamilia As String, ByVal ParamArray
Animales())
Dim i As Integer
Debug.Print "La familia " & NombreFamilia & " está compuesta
por los siguientes animales "
For i = 0 To Ubound(Animales)
    Debug.Print Animales(i)
Next i
End Sub
```

Y se podrá llamar de varias maneras:

```
Sub Llamadas()
    Call Familia_Al_Completo("ANDRÉ", "Mogwaï", "Michoko",
"Snoopy", "Shamalow", "Sushi", "Shere-Khan", "Suricate")
    Call Familia_Al_Completo ("GAUTHIER", "Dali")
End Sub
```

h. Cálculo y retorno de varios valores

Pasar argumentos por referencia es la oportunidad para que el desarrollador pueda obtener varios valores al mismo tiempo. De hecho, el paso por referencia ByRef modifica el valor real de la variable en la función o procedimiento llamador y, por tanto, permite acumular el valor de retorno de la función. Por ejemplo, si queremos programar una función `Division`, que devuelva el cociente de la división de un número `x` por `y`, pero que también devuelva el resto por medio del argumento que se pasa por referencia:

```
Public Function Division(ByVal x As Integer, ByVal y As Integer,
ByRef r As Integer) As Integer
    'Calcula el resto y el cociente de la división de x por y
    r = x mod y 'módulo
    Division = x \ y 'división entera de x por y
End Function
Sub Prueba9()
    Dim r As Integer, q As Integer
    q = Division(125, 8, r)
    MsgBox "el cociente de la división de 125 por 8 es " & q & " y
el resto " & r
End Sub
```

Ámbito y ciclo de vida de las variables

1. El ámbito

Cuando se declaran y se utilizan variables en un programa, se plantea la cuestión de la accesibilidad de estas variables respecto al conjunto del programa. Hay varias maneras de declarar variables/constantes. Según la palabra clave utilizada durante la declaración, la variable será accesible solamente desde algunos puntos del programa. De la misma manera, una variable solo se puede conservar durante la ejecución de un procedimiento o durante toda la ejecución de un programa.

Se definen tres niveles de acceso posibles:

- A nivel de un procedimiento: se trata de todas las variables declaradas dentro del procedimiento, no son visibles "fuera" del procedimiento. Se utilizan las palabras clave `Dim` y `Static`.

  Para el perro, se tratará de una habitación de la casa.

- A nivel de un módulo: todos los procedimientos y funciones que se declaren dentro del módulo tendrán acceso a esta variable. Se utilizan las palabras clave `Dim` y `Private`.

  Se tratará de la casa completa.

- A nivel del conjunto de los módulos del proyecto: se usa la palabra clave `Public`.

Es el interior y el exterior de la casa.

En el siguiente ejemplo, la variable `varProc` tiene un ámbito de nivel procedimiento, la variable `varModuloPrivado` tiene un ámbito de nivel módulo y la variable `varModuloPublico` tiene un ámbito de nivel proyecto.

```
Public varModuloPublico As Integer
Dim varModuloPrivado as Integer
Sub Init()
varModuloPublico = 50
varModuloPrivado = 100
End Sub
Sub Proc_Ejemplo()
    Init
    Dim varProc As Integer
    varProc = 10
    MsgBox varProc 'Muestra 10
    MsgBox varModuloPublico + varModulePriv 'Muestra 'Muestra 150
End Sub
Sub Proc_Ejemplo_2()
    Init
    Msgbox varModuloPublico + varModuloPrivado 'muestra 150
    MsgBox varProc 'Generará un error, la variable VarProc
no está declarada en Proc_Ejemplo_2
End Sub
```

2. El ciclo de vida

De la misma manera, la declaración de una variable va a tener una influencia en el tiempo durante el que se conserva su valor.

```
Sub Incrementa()
    Dim i As Long
    i = i + 1'incrementa i en 1
    MsgBox i 'muestra el valor de i
End Sub
Sub Incrementa_2()
    Static i As Long
    i = i + 1           'incrementa i en 1
        MsgBox i  'muestra el valor de i
End Sub
```

Si se ejecuta tres veces el procedimiento `Incrementa`, mostrará tres veces 1. Si se ejecuta tres veces el procedimiento `Incrementa_2`, mostrará 1, después 2, por último 3.

A continuación se muestra una tabla resumen del ámbito de las variables:

Ámbito/accesibilidad	En el procedimiento	En la zona de declaración
Dim	Procedimiento	Módulo
Private	No aplicable	Módulo
Public	No aplicable	Proyecto
Static	Procedimiento	No aplicable

Y aquí se muestra el ciclo de vida de las variables:

Ciclo de vida	En el procedimiento	En la zona de declaración
Dim	Procedimiento	Proyecto
Private	No aplicable	Proyecto
Public	No aplicable	Proyecto
Static	Proyecto	No aplicable

Convenciones de nomenclatura y tipos de código VBA

Es muy recomendable, incluso obligatorio, seguir ciertas reglas de nomenclatura cuando se escribe un programa en VBA. Estas convenciones fundamentalmente permiten facilitar la lectura y relectura del código. Las secciones que siguen permiten ayudarle a conocer algunas de estas reglas.

Las reglas que siguen no son las únicas que se pueden utilizar. Cada empresa, equipo de desarrollo o desarrollador puede tener su propia regla de nomenclatura. Sin embargo, las reglas de nomenclatura más frecuentes se indican en las siguientes secciones.

1. Regla general

Los nombres de objetos, variables, constantes, funciones y procedimientos están sujetos a algunas reglas en VBA.

- el nombre comienza por una letra;
- el nombre está limitado a 255 caracteres;
- el nombre puede estar compuesto por letras, cifras y el carácter de subrayado (_);
- el nombre no puede contener el carácter de puntuación ni de espacio;
- el nombre no puede corresponder a una palabra de VBA reservada;
- el nombre debe ser único dentro de un mismo ámbito.

No podemos tener dos Rantanplan en una habitación, ni dos Milou en la casa, ni siquiera dos Snoopy en el exterior.

Ejemplos de nombres válidos

```
Nombre_Perro
NombrePerro
Nombre_1_Perro
```

Aunque no esté prohibido en VBA, se desaconseja utilizar caracteres acentuados, ya que son específicos y solo se pueden gestionar con ciertos teclados, lo que hace complicado su manejo posterior en el programa.

Le llamaremos Idefix en lugar de Idéfix.

2. Convención de nomenclatura de los objetos

De manera general, se deben renombrar los objetos de la interfaz (`Controls`). El nombre de los objetos sigue el modelo `TypeOfControl`, seguido de un número autoincrementado por Access. A continuación se muestran algunos ejemplos de nombres:

- `Texto10`
- `Lista3`
- `Comando7`

Más allá de la dificultad de saber en el momento de codificar qué objeto hace referencia el nombre, es frecuente utilizar un prefijo que determine el tipo de objeto del que se trata, seguido del nombre del objeto. El uso de un

carácter de subrayado entre el prefijo y el nombre del objeto puede ayudar en la lectura, pero no es un requisito previo.

A continuación se listan algunos prefijos para los objetos de control:

Prefijo	Objeto correspondiente	Ejemplo
Cbo	ComboBox: zona de lista desplegable	Cbo_Usuario
Lst	ListBox: zona de lista	Lst_Empresas
Txt	TextBox: zona de texto	Txt_Nombre
Lbl	Label: etiqueta	Lbl_Fecha_del_dia
Btn	CommandButton: botón de comando	Btn_Salir

3. Convención de nomenclatura de los objetos de Access

Los objetos de Access se pueden nombrar según un prefijo de tres caracteres.

A continuación se listan algunos ejemplos de objetos y nombres.

Control	Nombre de objeto VBA	Prefijo	Ejemplo de nombre del objeto
Tabla	TableDef	Tbl	TblAnimales
Consulta	QueryDef	Qdf	QdfAdoptados
Formulario	Form	Frm	FrmInicio
Informe	Report	Rep	RepEstadisticas
Campo	Field	Fld	FldDtNacimiento
Índice	Index	Idx	IdxID

4. Convención de nomenclatura de las variables

Para poder determinar el tipo de variable después de la lectura del nombre, es recomendable utilizar un prefijo, completado por un nombre lo más explícito posible. Con frecuencia se utilizan dos tipos de prefijos: los compuestos por una única letra y los compuestos por tres letras. El uso de uno u otro no tiene impacto en la legibilidad solamente si el desarrollador ha respetado esta regla en todo su programa.

A continuación se muestran algunos prefijos según el tipo de variable:

Tipo de variable	Prefijo 1 letra	Prefijo 3 letras	Ejemplo prefijo 1 letra	Ejemplo prefijo 3 letras
Boolean	b	bln	bEsterilizado	blnEsterilizado
Double	d	dbl	dPeso	dblPeso
Integer	i	int	iCaja	intCaja
Long	l	lng	lAnimales	lngAnimales
Object	o	obj	oMail	objMail
String	s	str	sNombre	strNombre
Variant	v	vnt	vTabla	vntTabla

5. Convención de nomenclatura para las constantes

Para no tener conflicto con las variables, la convención universal entre los programadores es utilizar nombres escritos en mayúsculas con un nombre explícito, en el que cada palabra que lo compone está separada del resto por un carácter subrayado.

A continuación se muestran algunos ejemplos de constantes:

```
Const CST_PORCENTAJE_IVA As Single = 20
Const COLOR_PERSO_BROWN = &3366
```

6. Convención de nomenclatura para los argumentos

Las variables declaradas como argumentos normalmente no tienen ningún prefijo.

7. La indentación

Para que el código sea lo más legible posible, se recomienda indentarlo. La indentación se hace con la tecla tabulación (Tab). Las líneas de código que forman parte de un mismo bloque de instrucciones deben estar alineadas para facilitar la lectura.

A continuación se muestra un ejemplo de código no indentado:

```
Sub Prueba()
Dim i As Integer
For i = 1 To 8
Msgbox i * i
Next i
End Sub
```

Y el mismo código indentado:

```
Sub Prueba2()
    Dim i As Integer
    For i = 1 to 8
            Msgbox i * i
    Next i
End Sub
```

El conjunto de ejemplos que se proporcionan en este libro se indenta para permitir la mejor lectura posible.

8. La ofuscación

Al contrario de lo que sucede con los elementos que se han presentado hasta ahora, existe una técnica para hace un código ilegible, transformando el código de manera que sea ininteligible. Todos los identificadores se renombran, todos los comentarios se eliminan, así como la indentación del código y además, no se respetan las reglas de estilo.

Nuestro perro se perderá.

Las estructuras de decisión condicional

Cuando se desea ejecutar código dependiendo de varias alternativas, es necesario llamar a estructuras de decisión condicional.

¿Qué cesta elegir?

1. La estructura de prueba If … Then … End If

a. Condicional único

En VBA, se usa la siguiente sintaxis:

```
If condición_1 Then
    Instrucción_A
[Else
    Instrucción_B]
End If
```

Los corchetes indican que las instrucciones situadas dentro son opcionales. El funcionamiento de esta estructura es el siguiente:

Si la condición `condición_1` se cumple, entonces se ejecutará la instrucción `Instrucción_A`, en caso contrario se ejecutará la instrucción `Instrucción_B`.

Si el dueño está en el sofá, selecciono la cesta roja.

Por ejemplo, consideremos un programa que asigna a una variable `bMayor`, declarada inicialmente, el valor `True` si un valor comprendido entre 0 y 99 se almacena en una variable `Edad` y este valor es estrictamente superior a 17. El programa se escribiría de la siguiente manera:

```
If Edad > 17 Then
    bMayor = True
Else
    bMayor = False
End If
```

De esta manera, si el valor almacenado en la variable `Edad` es estrictamente superior a 17 (es decir, contiene un valor comprendido entre 18 y 99), `bMayor` vale `True` (porque la persona que tiene estrictamente más de 17 años es mayor de edad, - mayoría de edad en España); en caso contrario, `bEsterilizado` vale `False` (porque la persona con una edad entre 0 y 17 años es menor).

Como la instrucción `Else` es opcional, también hubiera sido posible escribir el programa anterior de la siguiente manera:

```
bMayor = False
If Edad > 17 Then
    bMayor = True
End If
```

Los dos programas son equivalentes.

b. Condicional múltiple

También es posible probar varias condiciones, unas detrás de otras. Para esto se usa la palabra clave `ElseIf`, como en la siguiente sintaxis:

```
If condición_1 Then
    Instrucción_A
ElseIf condición_2 Then
    Instrucción_B
ElseIf condición_3 Then
    Instrucción_C
...
[Else
    Instrucción_EnCasoContrario]
End If
```

El funcionamiento de esta estructura es la siguiente:

Si la condición `condición_1` se cumple, entonces se ejecuta la instrucción `Instrucción_A`; en caso contrario, si la condición `condición_2` se cumple, entonces se ejecuta la instrucción `Instrucción_B`; en caso contrario, si la condición `condición_3` se cumple, entonces se ejecuta la instrucción `Instrucción_C`, y así sucesivamente. Si no se cumple ninguna de las condiciones, se ejecuta la instrucción `Instrucción_EnCasoContrario` (si se define).

Si el dueño está en el sofá, la cesta roja. En caso contrario, si el dueño está en la oficina, la cesta azul. Es caso contrario, la amarilla.

A continuación se muestra un ejemplo de programa que va a asignar a la variable `strTamanioDelPerro` un valor, según el valor de la variable `StrRazaDePerro`.

```
If strRazaDePerro = "Chihuahua "Then
    strTamanioDelPerro = "Pequeño"
ElseIf strRazaDePerro=" Leonberg" Then
    strTamanioDelPerro = "Grande"
Else
    strTamanioDelPerro = "Mediano"
End If
```

2. Estructura de prueba Select … Case … End Select

Cuando las condiciones se aplican en diferentes momentos sobre la misma variable, es posible simplificar la serie `ElseIf` con otra estructura condicional: la instrucción `Select Case`. La sintaxis general de esta estructura es la siguiente:

```
Select Case ValorAComprobar
```

```
        Case Valor_1
                Instrucción_A
        Case Valor_2
                Instrucción_B
...
        Case Valor_X
                Instrucción_X
        [Case Else
                Instrucción_EnCasoContrario]
End Select
```

`ValorAComprobar` devuelve un valor que se va a comparar con cada uno de los valores (`Valor_1`, `Valor_2`, …, `Valor_X`). Cuando `ValorAComprobar` se corresponde con uno de estos valores, la instrucción que se devuelve se ejecuta (`Instrucción_A`, `Instrucción_B`, …, `Instrucción_X`). Si no se corresponde con ninguno de los valores, se ejecuta la instrucción `Instrucción_EnCasoContrario`, que corresponde a `Case Else` (que puede ser opcional).

Si retomamos el ejemplo anterior, se podría haber escrito el programa como se muestra a continuación:

```
Select Case strRazaDePerro
        Case "Chihuahua", "Bichon"
               strTamanioDelPerro = "Pequeño"
        Case "Leonberg", "Mastiff"
               strTamanioDelPerro = "Grande"
        Case Else
               strTamanioDelPerro = "Mediano"
End Select
```

3. Estructura de prueba IIf

La estructura condicional `If Then End If` algunas veces se puede simplificar en una única línea, cuando solo se ejecuta una única instrucción en el `Then` y en el `Else`. La función `IIf` permite esto. La sintaxis general es la siguiente:

```
IIf(test, valorRetornosi_comprobacion_verdadero,
valorRetornosi_comprobacion_falso)
```

Los dos valores, `valorRetornosi_comprobacion_verdadero` y `valorRetornosi_comprobacion_falso`, se calculan antes de que una de las dos se devuelva, según si el primer argumento devuelve `True` o `False`. A continuación se muestra un ejemplo que devuelve Positivo o Negativo, según el valor que se le aplica.

```
Dim strValor As String
strValor = IIf(Valor<0, "Negativo","Positivo")
```

Atención, esta estructura es una función, por lo que solo se puede usar devolviendo los valores, y no se puede sustituir con una sintaxis `If Then End If` en todos los casos.

Los bucles

En VBA, cuando queremos ejecutar varias veces la misma serie de instrucciones, en lugar de copiar n veces las mismas instrucciones, como en el siguiente ejemplo:

```
Sub Metodo_Erroneo()
    Msgbox "Hola"
    Msgbox "Hola"
....
    Msgbox "Hola"
End Sub
```

Es posible crear bucles, que indicarán al programa que debe ejecutar una serie de instrucciones varias veces. Hay varios tipos de bucles, que se explican en las siguientes secciones.

1. El bucle Do Loop

Hay varias estructuras de bucles `Do Loop` en VBA. Cada estructura tiene su sintaxis.

a. Do While Loop

La siguiente es la sintaxis:

```
Do While CondicionAComprobar
    Instrucciones
Loop
```

La instrucción dentro del bucle se ejecuta mientras la condición `CondicionAComprobar` se cumpla.

El perro ve al cartero y ladra mientras esté frente a la casa.

b. Do Loop While

La sintaxis es la siguiente:

```
Do
    Instrucciones
Loop While CondicionAComprobar
```

La instrucción dentro del bucle se ejecuta una primera vez, y se ejecuta de nuevo mientras la condición `CondicionAComprobar` se cumpla.

El perro ladra, ve al cartero y ladra mientras él esté frente a la casa.

c. Do Until Loop

La sintaxis es la siguiente:

```
Do Until CondicionAComprobar
    Instrucciones
Loop
```

La instrucción dentro del bucle se ejecuta hasta que la condición `CondicionAComprobar` se cumpla.

Mientras que el cartero esté frente a la casa, el perro ladra.

d. Do Loop Until

La sintaxis es la siguiente:

```
Do
    Instrucciones
Loop Until CondicionAComprobar
```

La instrucción dentro del bucle se ejecuta una primera vez, y se ejecuta de nuevo hasta que la condición `CondicionAComprobar` se cumpla.

El perro ladra, hasta que el cartero no esté frente a la casa.

e. Ejemplos

Por ejemplo, podemos realizar un bucle en el que se incremente una variable `i` (ya declarada) hasta que alcance el valor 10, con la siguiente sintaxis:

```
Do While i <> 5
    i = i + 1
Loop
```

O:

```
Do
    i= i +1
Loop While <> 5
```

O:

```
DO Until i = 5
    i = i +1
Loop
```

O:

```
Do
    i = i + 1
Loop Until i = 5
```

Observe que, cuando i vale 5 antes de entrar en el bucle, los bucles Do Loop While i<> 5 y Do Loop Until i = 5 se ejecutan sin fin, ya que i tomará el valor 6 (5+1) después de la primera instrucción i = i +1.

2. El bucle For Next

Cuando se desea ejecutar ciertas instrucciones un determinado número de veces, siendo este número conocido de antemano, es posible utilizar la estructura de bucle For Next. La sintaxis general es la siguiente:

```
For contador  = Inicial To Final [Step Paso]
    Instrucción_A
Next contador
```

La instrucción Instrucción_A dentro del bucle se ejecutará para la variable contador, que va de Inicial a Final (Inicial, Inicial + Paso, Inicial + 2 * Paso ... Final). Si no se ha definido ningún Paso (Step es opcional), entonces la variable contador se incrementará de 1 en 1. Si Inicial es superior a Final, las instrucciones dentro del bucle nunca se ejecutarán.

Por ejemplo, para visualizar 5 veces el texto "Hola", es posible escribir el siguiente programa:

```
Sub Prueba()
    Dim i As Integer
    For i = 1 To 5
          Msgbox "Hola"
    Next i
End Sub
```

Por ejemplo, si se desea realizar un cálculo sobre la suma de los números impares hasta un número n, el programa podría ser el siguiente:

```
Function Sumapar(n as Integer) As Integer
    Dim Total As Integer, i As Integer
    Total = 0
    For i = 1 To n Step 2
          Total = Total + i
    Next i
    Sumapar = Total
End Function
```

3. El bucle While Wend

De la misma manera que funcionan los bucles Do Loop, la estructura While Wend permite ejecutar instrucciones

mientras una condición se cumpla. La sintaxis general es la siguiente:

```
While Condicion
    Instrucción_1
Wend
```

La instrucción `Instrucción_1` se ejecutará mientras la condición `Condicion` se cumpla.

Retomando el ejemplo del incremento de la variable `i` hasta que `i` valga 10, la sintaxis con el bucle sería la siguiente:

```
While i <> 10
    i =i + 1
Wend
```

En la actualidad, esta sintaxis se utiliza fundamentalmente por razones de compatibilidad hacia atrás, aunque son preferibles los bucles `Do Loop`.

Las entradas-salidas en VBA

Durante la ejecución de un programa, es posible interactuar con el usuario por medio de cuadros de diálogo. Estos cuadros de diálogo permiten al programa dar información al usuario, así como pedírsela. Hay dos funciones principales que muestran estos cuadros de diálogo.

1. La función InputBox

Esta función muestra un cuadro de diálogo en el que el usuario puede escribir texto o hacer clic en un botón, y la función devuelve al programa el contenido de la zona de texto en forma de cadena de caracteres. Esta función VBA se declara de la siguiente manera:

```
Public Function InputBox( _
    ByVal Prompt As String, _
    Optional ByVal Title As String = "", _
    Optional ByVal DefaultResponse As String = "", _
    Optional ByVal Xpos As Integer = -1, _
    Optional ByVal YPos As Integer = -1, _
    Optional ByVal HelpFile As String = "", _
    Opcional Context As Long) As String
```

Como puede comprobar al leer la declaración de esta función, muchos de los argumentos son opcionales. A continuación se muestra a qué corresponden:

Argumento	Descripción
`Prompt`	Corresponde al texto que se mostrará encima de la zona de texto en la que el usuario deberá escribir. Este argumento es obligatorio, limitado a 1024 caracteres. Es posible extender el texto en varias líneas, insertando retornos de carro (`Chr(13)`) y saltos de línea (`Chr(10)`).
`Title`	Corresponde al título que tomará el cuadro de diálogo. Este argumento es opcional y, si no se define ningún título, se mostrará como nombre de la aplicación "Microsoft Access".
`DefaultResponse`	Corresponde al valor por defecto de la zona de texto durante la visualización del cuadro de diálogo. Este argumento es opcional y, si no se define ningún valor por defecto, la zona de texto está vacía durante la visualización.
`Xpos`	Corresponde a la distancia horizontal, en píxeles (unidad relacionada con las pantallas), entre el borde izquierdo de la ventana y el borde izquierdo del cuadro de diálogo. Este argumento es opcional y, si no se indica ningún valor, el cuadro de diálogo estará centrado horizontalmente.
`Ypos`	Corresponde a la distancia vertical, en píxeles, entre la parte superior de la ventana y la parte superior del cuadro de diálogo. Este argumento es opcional y, si no se especifica ningún valor, el cuadro de diálogo se posicionará a un tercio de la ventana, partiendo de la parte superior de la ventana.
`HelpFile`	Corresponde al nombre del archivo (ejemplo: "main.hlp") o a la ubicación del archivo de ayuda (ejemplo: "C:\temp\Help.hlp").
`Context`	Corresponde al número de error asociado a `HelpFile`.

Cuando se definen los argumentos `HelpFile` y `Context`, es accesible la ayuda de la tecla [F1] y apuntará a la información que defina la ayuda.

A continuación se muestra un ejemplo de programa que llama a la función `InputBox`, pidiendo al usuario su nombre y almacenando el resultado en una variable respuesta (declarada con antelación):

```
Nombre = InputBox("¿Cuál es su nombre, por favor?","Mi Aplicación
Access", "escriba aquí")
```

El programa mostrará esto:

La variable `nombre` almacenará lo que escriba el usuario.

Si no se escribe ningún texto, si el usuario hace clic en el botón **Cancelar**, o si cierra el cuadro de diálogo pulsando en la cruz la variable `nombre` almacenará una cadena de caracteres vacía ("").

2. La función MsgBox

La función `MsgBox` (se pronuncia message box) permite mostrar al usuario un texto en un cuadro de diálogo, con uno o varios botones. Ya hemos utilizado esta función en diferentes ejemplos de esta sección. Su declaración en VBA es la siguiente:

```
Function MsgBox (Prompt As Variant, _
    Optional Buttons As Long, _
    Optional Title As String, _
    Optional HelpFile As String
) As Integer
```

Durante su llamada, esta función muestra el texto `Prompt` en un cuadro de diálogo, con `Title` como título y uno o varios botones en los que el usuario podrá hacer clic. Cuando no se define ningún valor de `Buttons`, el cuadro de diálogo solo contendrá un botón **Aceptar**. De la misma manera, si no se define ningún título `Title`, se mostrará el nombre de la aplicación "Microsoft Access".

a. Combinaciones de botones

Según el valor definido en el argumento `Buttons`, se podrán mostrar varias combinaciones de botones en el cuadro de diálogo. Estas combinaciones se asocian a constantes de VB. A continuación se muestra una tabla.

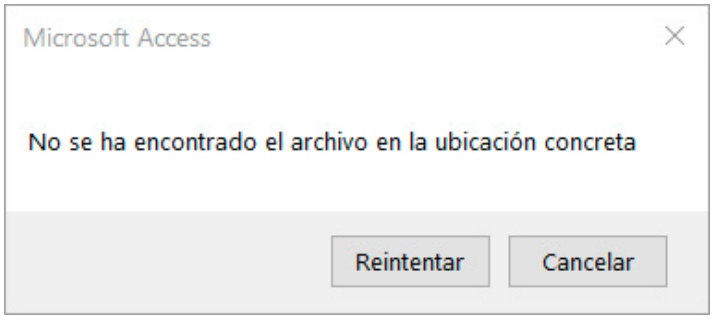
Valor	Descripción	Constante VB
0	Aceptar	vbOkOnly
1	Aceptar y Cancelar	vbOkCancel

2	Abandonar, Comenzar de nuevo e Ignorar	vbAbortRetryIgnore
3	Sí , No y Cancelar	vbYesNoCancel
4	Sí y No	vbYesNo
5	Comenzar de nuevo y Cancelar	vbRetryCancel

Por ejemplo, el siguiente programa:

```
MsgBox "No se ha encontrado el archivo en la ubicación concreta", 5
```

Mostrará el siguiente cuadro de diálogo:



b. Los iconos

Además de los botones mostrados en el cuadro de diálogo, es posible añadir un icono para indicar al usuario el tipo de mensaje al que se refiere. La sintaxis de la función es la siguiente:

```
MsgBox "Su mensaje", combinación_de_botones + código_icono,
"Título del cuadro de diálogo
```

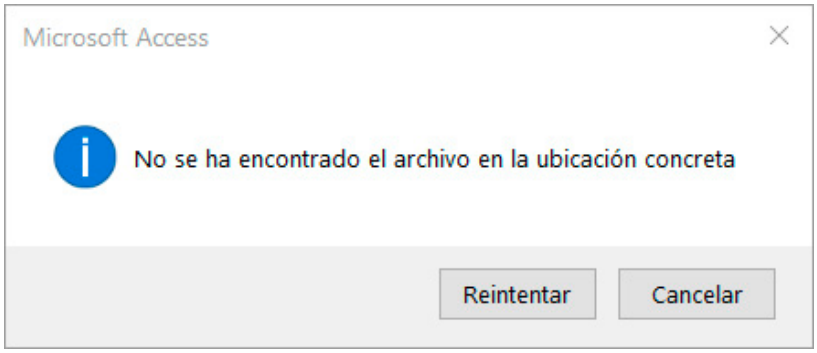
Cada icono tiene un valor y una constante de VB asociados:

Valor	Icono	Imagen	Constante VB
16	Crítico		vbCritical
32	Pregunta		vbQuestion
48	Exclamación		vbExclamation
64	Información		vbInformation

Retomando el ejemplo anterior, el cuadro de diálogo completado con un icono de Información se obtendría usando el siguiente programa:

```
MsgBox "No se ha encontrado el archivo en la ubicación
concreta",5 + 64
```

Lo que mostraría el siguiente cuadro de diálogo:



Observe que el siguiente código, que utiliza constantes, es equivalente al código anterior.

```
MsgBox "No se ha encontrado el archivo en la ubicación
concreta", vbRetryAbort + vbInformacion
```

c. Un botón Ayuda adicional

Además de las combinaciones de botones predefinidos por las constantes de VB, es posible añadir un botón adicional **Help**, que permitirá visualizar un archivo de ayuda (por ejemplo, en el que se explicarán las diferentes etapas de un proceso). Existe una constante para añadir este botón, y se deben añadir los argumentos `HelpFile` y `Context`. La sintaxis en caso de usar este botón sería la siguiente

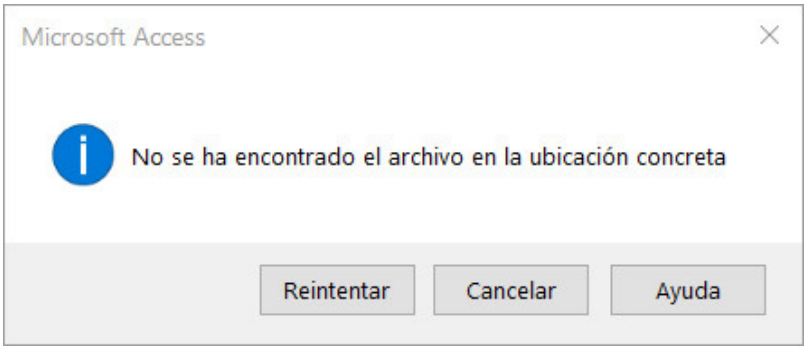
```
MsgBox "Su mensaje", _
    combinación_de_botones + código_icono + vbMsgBoxHelpButton, _
    "Título del cuadro de diálogo", _
    HelpFile, Context
```

Valor	Botón	Imagen	Constante VB
16384	Crítico	Help	vbMsgBoxHelpButton

Retomando el ejemplo anterior, podemos tener con el siguiente programa:

```
MsgBox "No se ha encontrado el archivo en la ubicación concreta",
5 + 64 + 16384
```

El cuadro de diálogo siguiente:



d. Botón seleccionado por defecto

Cuando aparece el cuadro de diálogo en la ventana, es posible definir qué botón está preseleccionado. Para los cuadros de diálogo que muestran varios botones (máximo cuatro: tres botones + **Help**), puede ser adecuado definir inicialmente quién recibirá el foco, permitiendo al usuario validar directamente pulsando la tecla [Intro] o [Espacio].

La sintaxis general se puede convertir en la siguiente:

```
MsgBox "Su mensaje", combinación_de_botones + código_icono
+ botón_por_defecto, "Título"
```

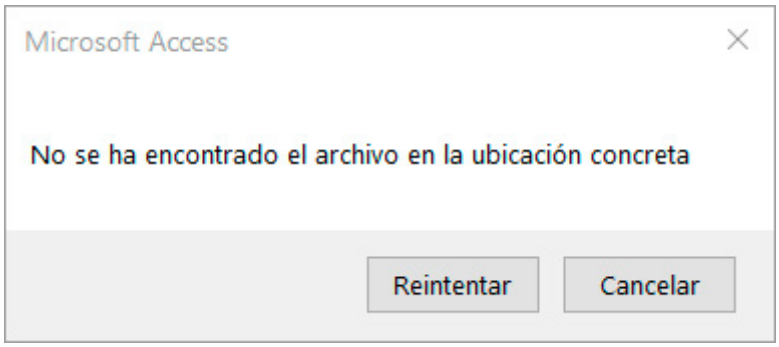
A cada botón por defecto le corresponde una constante de VB, como se indica en la siguiente tabla:

Valor	Botón	Constante VB
0	Primer botón	vbDefaultButton1
256	Segundo botón	vbDefaultButton2
512	Tercer botón	vbDefaultButton3
768	Cuarto botón	vbDefaultButton4

Caso en el que se selecciona por defecto el botón 2 (botón **Cancel**) en nuestro ejemplo:

```
MsgBox "No se ha encontrado el archivo en la ubicación
concreta", vbRetryCancel + vbInformacion + vbDefaultButton2
```

Que mostrará el siguiente cuadro de diálogo:



e. Otros aspectos específicos

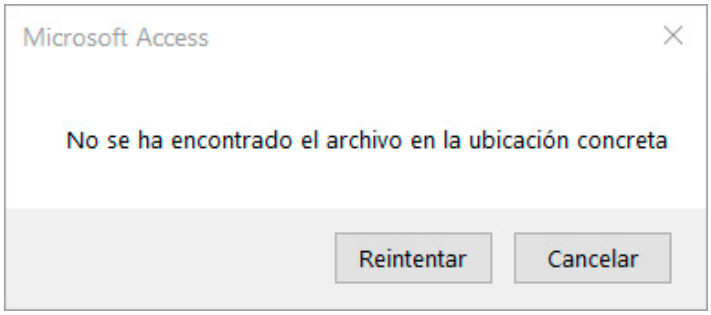
Además de los elementos mencionados, es posible definir algunos aspectos visuales del cuadro de diálogo. Una vez más, estas constantes se añaden al resto.

Valor	Descripción	Constante VB
Modalidad del cuadro de diálogo		
0	Determina, cuando aparece el cuadro de diálogo, si la aplicación se debe detener esperando la respuesta del usuario. Se toma por defecto este valor durante la visualización, si la constante `vbSystemModal` no se indica.	`vbApplicationModal`
4096	Determina, cuando aparece el cuadro de diálogo, si la aplicación debe detener todo el sistema esperando la respuesta del usuario.	`vbSystemModal`
Ver		
65536	Permite visualizar el cuadro de diálogo en primer plano.	`vbMsgBoxSetForeground`
524288	Permite definir una alineación a la derecha del texto que se muestra en el cuadro de diálogo.	`vbMsgBoxRight`
1048576	Permite hacer que se respete el orden de lectura de derecha a izquierda, para los sistemas hebreos y árabes.	`vbMsgBoxRtlReading`

Si retomamos el ejemplo con el siguiente programa:

```
MsgBox "No se ha encontrado el archivo en la ubicación
concreta", vbRetryCancel + vbInformacion + vbDefaultButton2 +
vbMsgBoxRight
```

El cuadro de diálogo mostrado sería el siguiente:



f. Valores de retorno

Una vez que se muestra el cuadro de diálogo, el usuario debe hacer clic en uno de los botones que se le presentan. Según el botón en el que pulse, la función `MsgBox` devolverá un valor particular. Este valor podrá ser objeto de un tratamiento particular (almacenarse en una variable, etc.).

A continuación se muestra la tabla de los valores devueltos según el botón pulsado, acompañados de las constantes de VB asociadas:

Valor	Botón	Constante VB
1	OK	vbOK
2	Cancelar	vbCancel
3	Abandonar	vbAbort
4	Comenzar de nuevo	vbRetry
5	Ignorar	vbIgnore
6	Sí	vbYes
7	No	vbNo

Por lo tanto, es posible determinar la serie de instrucciones que se debe ejecutar según el botón que el usuario pulse, como en el siguiente programa:

```
Dim Respuesta As Integer
Respuesta = Msgbox("¿Desea continuar?",VbYesNo + VbCritical)
Select Case Respuesta
    Case vbYes
          Instrucción_A
    Case vbNo
          Instrucción_B
End Select
```

Las salidas anticipadas: instrucción Exit

Hasta ahora, el código contenido en las funciones y procedimientos se debía ejecutar hasta llegar a la línea de salida del programa (`End Function` o `End Sub`). Sin embargo, puede ser útil poder salir de una función o procedimiento antes del fin de su ejecución. Es posible hacerlo gracias a la palabra clave `Exit`, seguida de la palabra clave que determina la sección de código de la que saldrá el código.

El perro que haya encontrado su hueso, deja de buscar.

Cuando empiece la programación, recomendamos no empezar de manera prematura con su código, porque esto puede causar dificultades en caso de error de diseño o programación.

1. Salida de la función: Exit Function

Es posible salir de una función antes del fin `End Function`. En el ejemplo siguiente, el programa saldrá de la función si el argumento proporcionado es igual a 1:

```
Function SiNoNul(x as Integer) As Boolean
SiNoNul = False
If x = 1 Then
    Exit Function
End If
...
End Function
```

En caso en que la variable `x` que se pase como argumento no sea igual a 1, el código se continúa hasta llegar a la instrucción de fin de la función `End Function`.

2. Salida de un procedimiento: Exit Sub

Es posible salir de un procedimiento antes del fin `End Sub`. En el siguiente ejemplo, el programa saldrá del procedimiento si el argumento proporcionado es una cadena de caracteres vacía.

```
Sub SiNoVacia (x as String)
If x = "" Then
    Exit Sub
End If
...
End Sub
```

En caso en que la variable `x` que se pasa como argumento no esté vacía, el código continuará hasta llegar a la instrucción de fin de procedimiento `End Sub`.

3. Salida del bucle: Exit For y Exit Do

Además de las salidas de las funciones y procedimientos, también es posible salir por anticipado de los bucles en los que el programa pueda haberse metido. Este método permite evitar tener que recorrer todos los elementos de un bucle antes de proseguir con el resto del programa. Según la naturaleza del bucle, se utilizará la palabra clave

Exit seguida de For para los bucles de tipo For ... Next, y la palabra clave Exit seguida de Do para los bucles de tipo Do ... Loop. Los siguientes programas son ejemplos de salida de bucle sobre las condiciones comprobadas:

```
Sub Ejemplo_Exit_For()
...
For i = 1 To 5000
    If i = 100 Then
          Exit For
    End If
Next i
...
End Sub
```

El programa saldrá del bucle cuando la variable contador i alcance el valor 100.

```
Sub Ejemplo_Exit_Do()
...
i = 1
Do Until i = 5000
    If i = 100 Then
          Exit Do
    End If
    i = i +1
Loop
...
End Sub
```

De la misma manera que en el programa anterior, el programa saldrá del bucle Do Loop cuando la variable i alcance el valor 100.

La administración de errores en VBA

Cuando comenzamos en programación, incluiso cuando ya hemos escrito varios miles de líneas de código, puede suceder que el programa no esté perfectamente codificado y que haya errores al intentar ejecutarlo. Hay varios tipos de errores, y vamos a explicar los métodos que nos pueden permitir evitarlos.

1. Los posibles tipos de error

Cuando codificamos, hay varios errores posibles que se pueden cometer a diferentes niveles del código.

a. Los errores de sintaxis

Hablamos de error de sintaxis cuando el programa está escrito de manera incorrecta, desde un punto de vista sintáctico. Por lo general VBE detecta directamente este tipo de error y la línea se colorea en rojo (o en el color elegido en las opciones, ver la sección Las opciones de VBE - Formato del editor, en el capítulo VBE y seguridad en Access 2019). Para que se haga esta detección automática, es necesario marcar la casilla de selección **Comprobación automática de sintaxis** en las opciones de VBE (ver la sección Las opciones de VBE - Editor, en el capítulo VBA y seguridad en Access 2019). Los errores de sintaxis pueden deberse al olvido de palabras o a la incorrecta escritura de palabras.

Ejemplos de errores

El siguiente código genera un error, ya que se ha olvidado la palabra clave `Then`.

```
Sub AusenciaPalabaClave ()
    Dim a As Integer
'la línea siguiente está incompleta,
 se ha olvidado la palabra clave Then
    If a=1
          a = a - 2
    End If
End Sub
```

El siguiente código genera un error, ya que la palabra clave `End` tiene un error ortográfico.

```
Sub ErrorOrtograficoPalabraClave()
    Dim a As Integer
    If a = 1 Then
          a = a + 1
    ' la línea siguiente tiene un error ortográfico,
    ' la palabra clave And se debe sustituir por la palabra clave End
    And if
End Sub
```

El código siguiente genera un error, ya que los paréntesis implican el uso de una asignación de variable.

```
Sub ErrorEnParentesis ()
    'la línea siguiente tiene demasiados paréntesis
```

```
    MsgBox("Ejemplo de código",vbOKOnly+vbCritical,"Capítulo 4")
End Sub
```

Solo se usan paréntesis durante las llamadas a funciones, cuyo resultado se almacena en una variable recipiente.

b. Los errores de nomenclatura (errores de compilación)

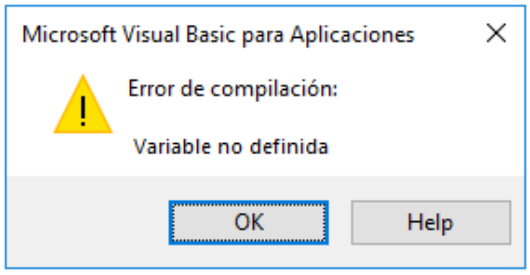
Durante la redacción de un programa, usará nombres de variables más o menos largos o complicados. Si por error cambia de posición dos letras, olvida una o repite alguna demasiadas veces, no será fácil detectarlo inmediatamente entre los miles de líneas de código que haya podido codificar. Pongamos el siguiente ejemplo para explicar lo que pasaría en uno de estos casos:

```
Sub ErrorNomenclatura ()
    Dim NombreDeVariableExtendida As Integer
    NombreDeVariableExtendida = 3
    'La línea siguiente mostrará 0 en lugar de 3,
    'porque el nombre de variable no es el correcto
    MsgBox NombreDeVraiableExtendida
End Sub
```

Si no se detecta automáticamente ningún error de sintaxis, el resultado esperado no sería el correcto, ya que el nombre de variable que escribe no es el mismo que el declarado y utilizado durante la asignación (`NombreDeVariableExtendida` y `NombreDeVraiableExtendida`, las letras `a` y `r` de `Variable` están cambiadas).

Este tipo de error puede ser particularmente lento si no hay ninguna solución automática de detección. Sin embargo, VBE detectará este error durante la compilación del código. Para evitar que no se produzca, es posible añadir en la parte superior de su módulo de código las palabras clave `Option Explicit`, que le obliga a declarar cada una de las variables que se llaman y utilizan durante el programa.

En caso de que no se declare una variable (en nuestro ejemplo anterior, la variable `NombreDeVraiableExtendida`), se muestra un mensaje de error cuando se hace clic en el menú **Depuración - Compilar**.



Para que las palabras clave `Option Explicit` se añadan automáticamente a cada módulo, marque la casilla de selección **Declaración de variables obligatoria** (ver la sección Las opciones de VBE - Editor, en el capítulo VBA y seguridad en Access 2019).

Ejemplos de error

El código siguiente genera un error de compilación, ya que la variable utilizada tiene un error ortográfico.

```
Sub MalNombre()
    Dim Ejemplo_1 As String
    'la variable que se llama no es la misma que la declarada
    Ejemplo1 = "Hello"
End Sub
```

El siguiente código genera un error, ya que la variable no se ha declarado.

```
Sub OlvidarDeclaracion ()
    'no se ha hecho ninguna declaración
i = 1
End Sub
```

VBA considera como diferentes los nombres `VariableExtendida` y `VariableExtendída` (uso de una letra acentuada), pero idénticas `VariableExtendida` y `variableextendida` (solo varían las mayúsculas/minúsculas de las letras).

c. Los errores de ejecución

Cuando escribe un programa, si no hay ningún error sintáctico ni de compilación, puede suceder que aparezcan errores durante su ejecución. Estos errores de ejecución son los más importantes de anticipar, como veremos en la sección La administración de los errores.

Ejemplos de error

El siguiente código generara un error si la tabla `T_PERRO` no existe en la base de datos actual.

```
Sub TablaInexistente()
    Dim tbl As DAO.TableDef
    Set tbl = CurrentDb.TableDefs("T_PERRO")
End Sub
```

No es posible dividir un número por cero.

```
Sub DivisionPorCero()
    Dim a As Integer
    Dim b As Integer
    a = 8
    'no se ha asignado un valor a b, y por tanto vale 0 en la línea siguiente
    'la división por cero no es posible en VBA
    MsgBox a / b
End Sub
```

Intentar multiplicar un número por texto no está permitido.

```
Sub ErrorDeTipo ()
    Dim i As Integer
    Dim j As String
    i = 1
    j="a"
    Msgbox i * j
End Sub
```

d. Los errores de razonamiento y de lógica

Más allá de los errores ya vistos hasta ahora, sigue siendo posible que el error venga del razonamiento del desarrollador durante la redacción de su programa. El resultado que devuelve el programa no es el esperado, como en el ejemplo siguiente:

```
Function AreaCuadrado(Lado As Single) As Single
   AreaCuadrado = 4 * Lado
End Function
```

El perímetro correcto de un cuadrado se da por la fórmula:

```
AreaCuadrado = Lado* Lado
```

2. La administración de los errores

En esta sección se van a mencionar de nuevo los principales medios de los que dispone el desarrollador para garantizar que no hay ningún error que pueda menoscabar la calidad de su aplicación para Microsoft Access 2019.

a. Las Option

En algunos ejemplos, hemos podido ver el uso práctico de `Option Explicit` u `Option Base`. Ahora se nos presenta la ocasión de listar las diferentes `Option` que hay en VBA. Como siempre, en la parte superior se sitúan los módulos de código, que son los siguientes.

Option Explicit

Esta instrucción permite obligar a la declaración de todas las variables situadas en el módulo, donde está la opción.

El interés de esta opción es evitar los errores tipográficos durante la escritura de las líneas de código. Además, esta opción detecta los problemas potenciales que tienen que ver con la nomenclatura, evitando los riesgos de confusión.

Option Base

Esta instrucción permite determinar el límite inferior que se utilizará en los índices de las tablas (ver la sección Las tablas). Sin declaración, el valor por defecto es 0, por lo que solo es útil el cambio de base con:

```
Option Base 1
```

Option Compare

Esta instrucción permite determinar el método que compara las cadenas de caracteres entre ellas. Hay tres soluciones de comparación:

```
Option Compare Text
```

Esta instrucción implica que una letra se considerará como equivalente, independientemente de si está escrita en mayúsculas o minúsculas (y así A=a).

```
Option Compare Binary
```

Esta instrucción implica que los caracteres se toman en su orden de aparición en la tabla ASCII, por lo que una letra mayúscula se considera anterior a su homóloga en minúscula (A=65<a=91).

```
Option Compare Database
```

Esta instrucción, que se aplica únicamente a Microsoft Access, permite comparar las cadenas según los argumentos regionales de la base de datos (también llamada collation).

Option Private Module

Esta instrucción permite hacer Private el módulo y el conjunto de funciones y procedimientos que figuran en él. De esta manera, los elementos no declarados Private están disponibles en el resto del módulo del proyecto, pero no para el resto de aplicaciones o proyectos (caso de la automatización, por ejemplo).

b. La compilación

Para detectar los eventuales errores antes de que se produzcan, es importante realizar frecuentemente compilaciones del proyecto. La compilación se hace con la opción de menú **Depuración - Compilar**.

c. El objeto Err

Los errores tienen su propio tipo de datos en VBA: `ErrObject`. Este objeto ofrece propiedades y métodos que se podrán utilizar, llegado el caso. El objeto que se relaciona automáticamente al error que aparece lleva el nombre `Err`.

Propiedades de Err

Propiedad	Descripción
`Description`	Representa el texto de descripción del error.
`HelpContext`	Representa el identificador del contexto asociado a una regla de un archivo de ayuda.

HelpFile	Representa la ubicación del archivo de ayuda del error.
LasDllError	Representa el código de error de sistema durante la llamada a una dll.
Number	Corresponde al número del error (ver la lista de errores en el anexo de este libro). Por defecto, el valor Err.Number devuelve 0 si no se ha producido ningún error.
Source	Representa el objeto fuente o la aplicación que causa el error.

Método de Err

Método	Descripción
Clear	Permite eliminar los errores.
Raise	Permite desencadenar voluntariamente un error.

3. El comportamiento de VBA en caso de error

a. La aparición de un error

A continuación se muestra el flujo clásico de un error que aparece en su programa.

Objeto `Err` presente

Desde el momento en que se ejecuta un programa, hay un objeto `Err`, con un valor por defecto de 0.

Aparición del error

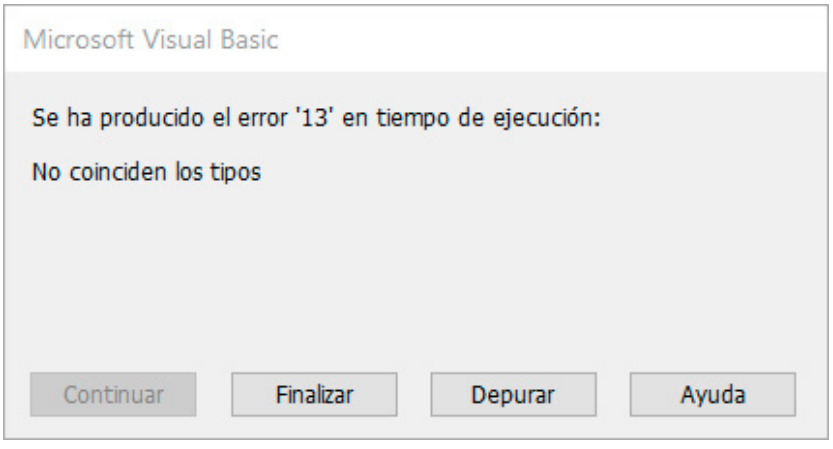
Por alguna razón que se puede solucionar más adelante o no, aparece un error.

Rellenar las propiedades de`Err`

En función del error que acaba de aparecer, las propiedades de este error se alimentan en el objeto `Err`.

El administrador del error se desencadena

Se abre un cuadro de diálogo y muestra la información de la que dispone con relación al error.



Cuando un error aparece en el programa, es muy complicado dejar un cuadro de diálogo con un mensaje de error que se muestra a un usuario, quien no tiene nada que hacer con esta información y seguramente le enviará un mensaje de queja.

Entre las tres soluciones que se ofrecen al desarrollador, se muestra a continuación lo que podemos decir.

b. On Error Resume Next

Muchos desarrolladores principiantes que usan la sintaxis `On Error Resume Next` en su programa se benefician de una ejecución visualmente intrascendente para el usuario. La instrucción `Resume Next` implica que, incluso en caso de aparición de un error, el programa continúa ejecutando las instrucciones siguientes. Por tanto, el error tiene lugar, pero no se asigna ningún tratamiento en este caso.


La palabra clave `Resume` elimina el error (equivalente a `Err.Clear`), cosa que no hace `GoTo`.

Ventajas

No hay ninguna ventaja real; esta sintaxis se ha dejado de utilizar en la inmensa mayoría de los casos.

Inconvenientes

Cuando tiene lugar un error, es preferible tratarlo correctamente, cambiando el código que es susceptible de generarlo, probando los valores que presentarán problemas (valor = 0, valor vacío, longitud nula, etc.) o tratando el código con una de las otras dos soluciones.

Ejemplo

El siguiente código no mostrará error durante la división por cero; en su lugar, aparecerá el segundo mensaje.

```
Sub OnErrorResumeNext()
    On Error Resume Next
    MsgBox 1 / 0
    MsgBox 2
End Sub
```

c. On Error GoTo 0

La función de esta instrucción es anular el soporte del error por `On Error` y restablecer el tratamiento por defecto del error del sistema.

Ventajas

En caso de que el desarrollador administre voluntariamente el error, fuerza el reinicio de la administración de error y devuelve el control al sistema.

Inconvenientes

Volver a una administración del sistema no ayuda al desarrollador a optimizar su aplicación.

Ejemplo

El siguiente código da el control al desarrollador durante el primer error, pero el sistema lo retoma para el segundo.

```
Sub OnErrorGoTo0()
    On Error Resume Next
    MsgBox 1 / 0
    MsgBox 2
    On Error GoTo 0
    MsgBox 1 / 0 ' Muestra un mensaje de
End Sub error
```

d. On Error GoTo Etiqueta

La última solución que podemos utilizar con `On Error GoTo` consiste en hacer que el código continúe desde un punto determinado del código, identificado por medio de una etiqueta. Una etiqueta se determina por una palabra seguida del carácter ":".

```
'Esto es una etiqueta
```

En caso de error detrás de la línea `On Error GoTo`, el código continuará en la línea que se indica después de la palabra clave `GoTo`. La palabra clave `Resume` también sirve para hacer que el código continúe por una ubicación directamente, sin que haya aparecido un error.

Ventajas

Esta instrucción es la que permite controlar mejor el comportamiento del programa.

Esta es la sintaxis que se utiliza automáticamente durante la generación de las macros en VBA.

Inconvenientes

El código se puede convertir rápidamente en difícil de gestionar, si se debe implementar la mayor parte de la administración de errores.

Ejemplo

El siguiente código permite ver varias etiquetas, así como una administración de error que continúa en un lugar determinado.

```
Sub OnErrorGoToEtiqueta()
    On Error GoTo aquí
    MsgBox 3 / 0
    'la línea siguiente nunca se ejecutará
    MsgBox 4
alla:
    Exit Sub
```

```
aquí:
    MsgBox 5
    Resume alla
End Sub
```

Es necesario que la etiqueta esté en el mismo procedimiento que el `On Error Goto` con el que está relacionado. En caso contrario, tiene lugar un error de compilación.

e. El evento Error

Cuando se programa en un formulario, hay un evento "En error", que permite generar un procedimiento de VBA cuya sintaxis general es la siguiente.

```
Private Sub Form_Error(DataErr As Integer, Response As Integer

End Sub
```

Este procedimiento permite gestionar globalmente los errores que puedan aparecer durante la ejecución del código del formulario.

En este procedimiento, se utilizan los siguientes elementos:

Elemento	Descripción
`DataErr`	Representa el número del error que aparece en el formulario.
`Response`	Representa el comportamiento que va a seguir el formulario. Si el sistema ignora el error y se aplica un tratamiento específico, la variable tomará el valor `acDataErrContinue`. Y al contrario, si se debe mostrar el mensaje de error de sistema, tomará el valor `acDataErrDisplay`.

Ejemplo

```
Private Sub Form_Error(DataErr As Integer, Response As Integer)
If DataErr = 440 Then
    'Esto es una error Automation
    Response = acDataErrContinue
    Msgbox "Excel ha encontrado un problema durante la exportación
de datos", vbOkOnly + vbCritical
End If
End Sub
```

Noción de objeto

VBA es un lenguaje que permite hacer programación orientada a objetos (POO): un objeto representa una idea, concepto o entidad del mundo real, como un avión, un individuo e incluso una película. Tiene una estructura interna y un comportamiento, y se puede comunicar con sus homólogos. Los elementos que permiten describir un objeto forman lo que se llama una clase. Cada objeto que proviene de una clase es una instancia de clase. Las clases tienen propiedades, métodos y eventos.

1. Propiedades

El objeto es una entidad que podemos distinguir gracias a sus propiedades (su color o dimensiones, por ejemplo). Si tomamos como ejemplo un libro, este se caracteriza por sus propiedades: número de páginas, título, número de capítulos, editor, contenido, etc. Cada una de sus propiedades se puede especificar en cada libro, pero todos los libros tienen fundamentalmente las mismas propiedades. Algunas propiedades de los objetos se pueden modificar (velocidad de un coche, por ejemplo) y otras no (una marca de coche).

En programación con VBA, la sintaxis general de acceso a las propiedades de un objeto es la siguiente:

```
UnObjeto.SuPropiedad
```

Aquí se accede a la propiedad `SuPropiedad` del objeto `UnObjeto`. La combinación `UnObjeto.SuPropiedad` tendrá el mismo comportamiento que una variable clásica, que puede tomar un valor o devolver un valor con el uso del operador =.

Por ejemplo, es posible leer el contenido SQL de una consulta de Access y visualizarla con el siguiente código:

```
Sub LeerSQLConsulta()
    Dim UnaConsulta As QueryDef
    UnaConsulta.SQL = "SELECT * FROM MiTabla"
    ...
    MsgBox UnaConsulta.SQL
End Sub
```

En el capítulo Los objetos de acceso a los datos DAO y ADO, volveremos al objeto `QueryDef` con más detalle.

Una propiedad de un objeto puede ser un objeto ella misma, por lo que se podría utilizar más adelante en el código.

2. Métodos

Además de los elementos que caracterizan un objeto, también es posible realizar acciones con estos objetos, como por ejemplo "sentarse", "acostarse", "arreglarse", etc. Estas acciones se llaman métodos. Estas acciones se representan en forma de procedimientos y funciones en VBA, según puedan o no devolver valores. Algunas acciones pueden necesitar argumentos para funcionar (número de caracteres de un libro, con o sin espacios).

En programación con VBA, la sintaxis general de acceso a los métodos de un objeto es la siguiente:

```
UnObjeto.SuMetodo [MiArgumento]
```

En el siguiente ejemplo, se trata simplemente de refrescar el vínculo entre una tabla relacionada y la aplicación Access:

```
Sub RefrescarRelacionTabla()
    Dim UnaVariableTemporal As TableDef
    ...
    UnaVariableTemporal.RefreshLink
End Sub
```

Y, en caso de que el método sea una función, podemos almacenar el resultado en una variable, como en el siguiente ejemplo:

```
Sub Número_Campos()
    Dim Nm_Campos As Integer
    Dim MiTabla As TableDef
    ...
Nm = MiTabla.Fields.Count
End Sub
```

De la misma manera que en las llamadas a funciones o procedimientos, los argumentos se separan por comas en las llamadas a métodos del objeto.

3. Eventos

Para cada objeto, es posible detectar y tener en cuenta el resultado de acciones particulares externas, que llamamos eventos. Todos los eventos que tienen lugar durante la ejecución de un programa se detectan y gestionan por la máquina, pero, según la opción del desarrollador, solo algunas se pueden tratar de manera particular, como por ejemplo un clic en un botón, una casilla de selección que se marca, la apertura o cierre de una ventana, etc. Estos eventos se gestionan a través de procedimientos que algunas veces tienen argumentos y otras no.

En VBA, la sintaxis más frecuente para los eventos es la siguiente:

```
Sub MiObjeto_MiEventoDestacado()
    'El código que se ejecutará cuando se detecte el evento
End Sub
```

Por ejemplo, cuando se trata de un clic en un botón llamado `MiBoton`, tendrá el código siguiente:

```
Sub MiBoton_Click()
    'código que se ejecutará
End Sub
```

Los eventos en Access se tratarán más adelante en detalle, en el capítulo Los eventos de Access.

4. Las colecciones

Cuando varios objetos de una misma clase se agrupan, pueden pertenecer a una misma colección de objetos. Para referirse a un elemento en particular de una colección de objetos, hay varias sintaxis, entre las siguientes:

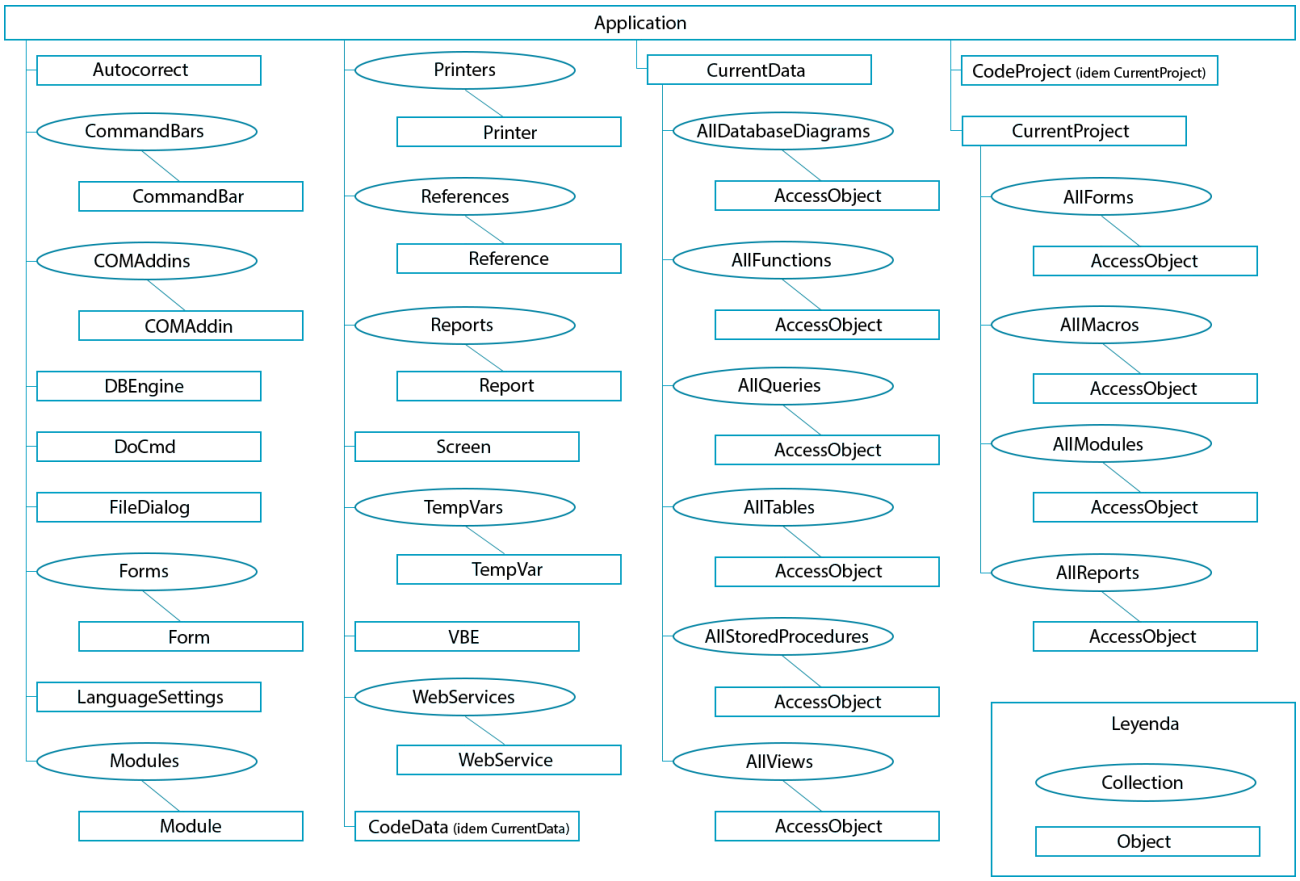
```
Nombre_Colección!Nombre_Objeto
Nombre_Colección![nombreObjeto]
Nombre_Colección("NombreObjeto")
Nombre_Colección(variable_Nombre) 'variable_Nombre es una cadena de
caracteres que contiene el nombre del objeto
Nombre_Colección(variable_Numero) 'variable_Numero es un valor
digital que contiene el número del objeto dentro de la colección.
```

Las dos sintaxis que se utilizan más habitualmente son la tercera y la quinta, ya que permiten el uso de IntelliSense (ver capítulo VBE y seguridad en Access 2019).

Es habitual en las colecciones comenzar a contar en 0 y no en 1. Además, el número de un objeto dentro de una colección dependerá de la presencia de otros elementos antes o después de él, lo que no garantiza localizar el objeto correcto por este medio.

Modelo de objeto de Access

El objetivo de algunas de las secciones siguientes es mostrar el modelo jerárquico utilizado dentro de la aplicación Access, en forma de colecciones de objetos, que se van a explicar a continuación.



Colecciones en Access

A continuación se muestran las principales colecciones que podemos manipular con VBA en Access.

Colección	Contiene una colección de	Descripción
COMAddins	COMAddin	Colección de los **complementos COM**
CommandBars	CommandBar	Colección de las **barras de comando**
Forms	Form	Colección de los **formularios abiertos**. Ver también CurrentProject.AllForms
Modules	Module	Colección de los **módulos**
Printers	Printer	Colección de las **impresoras disponibles**
References	Reference	Colección de las **referencias a librerías**. Ver Menú: **Herramientas\ Referencias**
Reports	Report	Colección de **informes**. Ver también CurrentProject.AllReports
TempVars	TempVar	Colección de las **variables temporales**
WebServices	WebService	Colección de las conexiones a **servicios web**

Objetos de Access

A continuación se muestran los principales objetos que es posible manipular en el modelo Access.

Objeto	Descripción
Application	Representa la aplicación Microsoft Access activa.
AutoCorrect	Representa las opciones de corrección automática de Access.
DBEngine	Representa el motor de base de datos Microsoft Jet. Este objeto permite controlar el resto de los objetos de acceso a los datos.
DoCmd	Permite convertir en VBA las acciones de Macros.
FileDialog	Permite acceder a las funcionalidades de los cuadros de diálogo (Abrir o Guardar, por ejemplo).
LanguageSettings	Permite acceder a los argumentos lingüísticos de la aplicación.
Screen	Permite acceder a la ventana activa (formulario, informe o control).
VBE	Permite acceder al editor Visual Basic Editor.
CurrentProject	Permite acceder a diferentes objetos de Access específicos.
CurrentData	Permite acceder a varios objetos de Access de acceso a los datos.

1. El objeto Application

El objeto Application corresponde a la aplicación Microsoft Access activa.

a. Propiedades

Propiedad	Descripción
AutomationSecurity	Devuelve o define el modo de seguridad utilizado por Access durante la apertura del archivo, por programación. Este valor es una constante msoAutomationSecurity, cuya lista de valores figura en el anexo de este libro.
BrokenReference	Devuelve un booleano que indica si la aplicación tiene una o varias referencias rotas.
Build	Devuelve un valor numérico que presenta el número de copia de Microsoft Access 2019 actualmente instalada.
CodeContextObject	Devuelve un objeto que contiene una macro o el código de VB actual en ejecución.
CurrentObjectName	Devuelve el nombre del objeto activo (formulario, informe, tabla, consulta, macro o módulo).
CurrentObjectType	Devuelve un valor numérico según el tipo del objeto activo. La lista de las constantes acObjectType aparece en el anexo de este libro.
FeatureInstall	Devuelve o define el modo de administración de las llamadas a métodos o las propiedades que necesitan funcionalidades no instaladas en Microsoft Access.
IsCompiled	Devuelve un booleano que indica si el proyecto VB de la aplicación está compilado.
MenuBar	Devuelve o define la barra de menús que se utilizará por la base de datos.
Name	Devuelve el nombre de la base de datos de Microsoft Access.

Parent	Devuelve el objeto padre de la base de datos.
ProductCode	Devuelve el identificador único de la aplicación Microsoft Access.
ShortcutMenuBar	Devuelve o define el menú contextual que se mostrará cuando el usuario haga un clic derecho con el ratón.
UserControl	Devuelve un booleano que indica si la aplicación se ha ejecutado por un usuario o si se trata de un control automático.
Version	Devuelve el número de la versión de Access (17.0).
Visible	Devuelve o define un booleano, que indica si la aplicación es visible o queda oculta por la ventana.

b. Métodos que operan sobre las bases de datos de Access

Método	Descripción
CloseCurrentDatabase	Permite cerrar una base de datos de Access abierta automáticamente en la misma aplicación.
CompactRepair	Compacta y repara la base de datos especificada. Devuelve un booleano que indica el resultado de la compactación.
ConvertAccessProject	Convierte la base de datos en otra versión.
CreateNewWorkgroupFile	Crea un archivo de grupo de trabajo para el acceso seguro a la base de datos.
CurrentDb	Devuelve un objeto Database correspondiente a la base de datos actualmente abierta y activa.
InstantiateTemplate	Crea una nueva base de datos y le aplica el modelo especificado.
NewCurrentDatabase	Crea una nueva base de datos en la ventana Microsoft Access.
OpenCurrentDatabase	Abre una base de datos de Access automáticamente y la convierte en la base de datos activa.
RefreshDatabaseWindow	Refresca la ventana **Base de datos**, permitiendo visualizar las modificaciones añadidas a las tablas, consultas, formularios, informes, macros o módulos.

c. Métodos que operan sobre los proyectos de Access (archivos .adp)

Método	Descripción
CreateAccessProject	Permite crear un nuevo proyecto en un disco.
NewAccessProject	Permite crear y abrir un nuevo proyecto, convirtiéndolo en activo en Access.

d. Métodos que operan sobre los objetos de Access (formularios, informes, etc.)

Método	Descripción
ColumnHistory	Devuelve el histórico de los diferentes valores que se han podido almacenar en un campo Comentario.
CreateControl	Permite la creación de un control en el formulario especificado abierto.
CreateForm	Permite la creación de un formulario y devuelve el objeto creado, de tipo Form.

CreateGroupLevel	Permite determinar un grupo de uno o varios campos sobre los que se podrá realizar una operación de ordenación.
CreateReport	Permite la creación de un informe y devuelve el objeto creado, de tipo Report.
CreateReportControl	Permite la creación de un control en el informe especificado abierto.
DeleteControl	Permite la eliminación de un control en el formulario especificado abierto.
DeleteReportControl	Permite la eliminación de un control en el informe especificado abierto.
DirtyObject	Permite indicar a Microsoft Access que el objeto (formulario o informe) ha sufrido una modificación y se debe guardar para que vuelva a aparecer durante la próxima visualización.
LoadPicture	Permite cargar una imagen en un control ActiveX.
SetHiddenAttribute	Permite activar el atributo caché de un objeto de Access.

e. Métodos que operan en la interfaz de Access

Método	Descripción
AddToFavorites	Permite añadir a los Favoritos el nombre de la base de datos actual.
Echo	Determina si Access refresca la información en la ventana.
GetOption	Devuelve el valor actual de una opción del cuadro de diálogo **Opciones**.
ImportNavigationPane	Importa una configuration de la pestaña de navegación a partir del disco.
Quit	Sale de la aplicación.
RefreshTitleBar	Actualiza la barra de título de la aplicación.
SetOption	Define el valor actual de una opción del cuadro de diálogo **Opciones**.
SysCmd	Permite visualizar una barra de progreso o un texto especificado en la barra de estado, devolver la información de Access y sus archivos asociados, o el estado de un objeto de la base de datos especificada.

f. Métodos que ejecutan una operación

Método	Descripción
Run	Permite la ejecución de un procedimiento o una función de VBA especificada por el usuario o por Microsoft Access.
RunCommand	Permite la ejecución de un comando integrado de Access.
RunDataMacro	Permite la ejecución de una macro de datos.

g. Métodos relacionados con los archivos XML

Método	Descripción
CreateAdditionalData	Permite la creación de un objeto AdditionalData, que se podrá utilizar para añadir tablas y consultas adicionales en la tabla padre que se está exportando por el método ExportXML.
ExportXML	Permite la exportación de datos XML, de información de presentación y de esquemas, a partir de varias fuentes de datos posibles.

ImportXML	Permite la importación de datos XML, de información de presentación y de esquemas, a partir de varias fuentes de datos posibles.
TransformXML	Transforma un archivo de datos XML, aplicándole una hoja de estilo XSL (*eXtensible Stylesheet Language*) y escribiendo el resultado en un archivo XML.

h. Métodos relacionados con los enlaces de hipertexto

Método	Descripción
FollowHyperlink	Permite la apertura de la página web o un documento a partir de la dirección de hipertexto especificada.
HyperLinkPart	Devuelve la información relativa a los datos almacenados en forma de enlace de hipertexto.

i. Métodos relacionados con los usuarios

Método	Descripción
CurrentUser	Devuelve el nombre del usuario actual de la base de datos.
CurrentWebUser	Devuelve la información del usuario actual de la base de datos web en Microsoft SharePoint.
IsCurrentWebUserInGroup	Devuelve un booleano que indica si el usuario actual de una base de datos web, es miembro del grupo de Microsoft SharePoint especificado.

j. Métodos que operan sobre los archivos Application XML (AXL)

Método	Descripción
LoadFromAXL	Importa el objeto definido en un archivo AXL a la base de datos.
SaveAsAXL	Exporta el objeto especificado en un archivo AXL.

Otros métodos

Método	Descripción
AccessError	Devuelve el descriptivo de error Access o del error DAO.
BuildCriteria	Permite la construcción de un criterio de filtro.
DefaultWorkspaceClone	Crea un nuevo espacio de trabajo sin forzar una nueva conexión del usuario.
LoadCustomUI	Carga el código XML para personalizar la barra de opciones.
Nz	Devuelve un valor por defecto si el proporcionado es nula.
SetDefaultWorkgroupFile	Define el archivo especificado como archivo de grupo de trabajo.

Ejemplos

```
Sub Ejemplos()
    Debug.Print Application.’ muestra 1
    Debug.Print Application.’ muestra 2
    Debug.Print
Application.CompactRepair(Replace(CurrentProject.FullName,
"Capítulo_05", "Capítulo_04"), _
    Replace(CurrentProject.FullName, "Capítulo_05",
 "Copia de seguridad_Capítulo_04"))
'cogerá la base del capítulo 04, la compactará y hará una copia
de seguridad, con el nombre 'Copia de seguridad_Capítulo_04’ ,
el nombre de la base de datos de ejemplo del capítulo que contiene
el nombre Capítulo_05. Muestra True si la compilación ha tenido éxito.
    Debug.Print Application.IsCompiled
End Sub
```

k. Las funciones de dominio

Podemos utilizar las funciones de dominio en los formularios e informes; realizan cálculos sobre un conjunto de registros.

La sintaxis general es la siguiente:

```
Application.FuncionDeDominio(campos, fuente, [criterios])
```

Donde `campos` es una etiqueta de campos del origen (tabla o consulta) y donde los criterios permiten filtrar el conjunto de registros del resultado.

Función	Descripción
DAvg	Devuelve la media del conjunto de registros.
DCount	Devuelve el número de registros del conjunto de registros.
DFirst	Devuelve el primer valor del conjunto de registros.
DLast	Devuelve el último valor del conjunto de registros.
DLookUp	Devuelve el valor específico del conjunto de registros.
DMax	Devuelve el valor máximo del conjunto de registros.
DMin	Devuelve el valor mínimo del conjunto de registros.
DStDev	Devuelve la desviación típica para una muestra del conjunto de registros.
DStDevP	Devuelve la desviación típica del conjunto de registros.
DSum	Devuelve la suma de los valores del conjunto de registros.
DVar	Devuelve la varianza para una muestra del conjunto de registros.
DVarP	Devuelve la varianza del conjunto de registros.
Eval	Determina la interpretación de una cadena de caracteres en términos matemáticos y devuelve el resultado.

l. Ejemplos

```
Sub Ejemplos ()
    'Número Total de adopción
    Debug.Print DCount("*", " ENI_ADOPCION_ADO")
    'Fecha de última adopción (fecha mayor)
    Debug.Print DMax("ANI_FECHA _ADOPCION", "ENI_ANIMAL_ANI")
    'Fecha de nacimiento del animal más viejo del refugio
    Debug.Print DMin("ANI_FECHA _NACIMIENTO", "ENI_ANIMAL_ANI",
"ANI_DATE_ADOPCION IS NULL")
    'Fecha de nacimiento del animal llamado Sushi
    Debug.Print DLookup("ANI_FECHA_NACIMIENTO", "ENI_ANIMAL_ANI", _
          "ANI_NOMBRE='Sushi'")
End Sub
```

2. El objeto DoCmd

El objeto `DoCmd` es el objeto "caja de herramientas" que permite replicar prácticamente todos los comandos de la interfaz de Access en VBA.

Cada una de las acciones se obtiene por un método propio, con uno o varios argumentos que se deben indicar cuando se desea ejecutarla.

A continuación se muestra una lista de los principales métodos disponibles con el objeto `DoCmd`.

a. Acciones en la aplicación Access

Método	Descripción	Acción macro equivalente
`CancelEvent`	Anula un evento.	`AnularEvento`
`Quit`	Sale de Access.	`Salir`
`RunCommand`	Ejecuta un comando de menú o una barra de herramientas.	`EjecutarComando`
`RunMacro`	Ejecuta una macro.	`EjecutarMacro`
`RunSQL`	Ejecuta una consulta SQL action.	`EjecutarSql`

b. Acciones sobre los objetos de Access

Método	Descripción	Acción macro equivalente
`ApplyFilter`	Aplica un filtro en una tabla, un formulario o un informe.	**AplicarFiltro**
`BrowseTo`	Permite navegar entre los diferentes objetos de la base de datos. También es posible cambiar la fuente de un control de un subformulario con el argumento `PathToSubFormControl`.	**Examinar**
`Close`	Cierra un objeto en particular.	**Salir**
`CloseDatabase`	Permite cerrar la base de datos abierta en Access. Es posible que aparezcan mensajes en los que se le solicite si desea guardar las modificaciones	**SalirBaseDatos**

	añadidas a los objetos, como durante un cierre normal de la base de datos.	
CopyObject	Permite copiar un objeto en la misma base o en otra.	**CopiarObjeto**
DeleteObject	Permite eliminar un objeto.	**EliminarObjeto**
GotoControl	Permite dar el foco a un control en la interfaz.	**EsperarControl**
GotoPage	Permite dar el foco al primer control de la página del formulario activo.	**EsperarPágina**
OpenData AccessPage	Permite abrir una página de acceso a los datos.	**AbrirPáginaAccesoDatos**
OpenDiagram	Permite abrir un esquema de la base de datos.	**AbrirEsquema**
OpenForm	Permite abrir un formulario.	**AbrirFormulario**
OpenFunction	Permite abrir una función de usuario en una base de datos Microsoft SQL Server, para visualizarla en Microsoft Access.	**AbrirFunción**
OpenModule	Permite abrir un módulo de VBA.	**AbrirMódulo**
OpenQuery	Permite abrir una consulta.	**AbrirConsulta**
OpenReport	Permite abrir un informe.	**AbrirInforme**
OpenStored Procedure	Permite abrir un procedimiento almacenado, en modo Hoja de datos, Creación o Vista preliminar antes de impresión.	**AbrirProcedimientoAlmacenado**
OpenTabla	Permite abrir una tabla.	**AbrirTabla**
OpenView	Permite abrir una página de acceso a los datos.	**AbrirVista**
PrintOut	Permite imprimir el objeto de Access activo.	**ImprimirObjeto**
Rename	Permite renombrar un objeto.	**Renombrar**
RepaintObject	Permite rediseñar un objeto.	**RediseñarObjeto**
Save	Permite guardar el objeto activo o el objeto especificado como argumento.	**Guardar**
SelectObject	Permite modificar la visibilidad de un objeto de Access.	**SeleccionarObjeto**
SetProperty	Permite definir una propiedad del objeto parmi (Activado, Visible, Bloqueado, Izquierdo, Arriba, Longitud, Altura, Color de texto, Color de fondo, Leyenda y Valor)	
SetParameter	Permite definir un argumento, que se utilizará durante las llamadas a métodos BrowseTo, OpenForm, OpenQuery, OpenReport o RunDataMacro.	**DefinirOrdenPor**

c. Métodos sobre los registros

Método	Descripción	Acción macro equivalente

FindNext	Permite continuar la búsqueda.	**BuscarSiguiente**
FindRecord	Permite buscar un registro.	**BuscarRegistro**
GotoRecord	Permite ir a un registro.	**IrARegistro**
RefreshRecord	Permite refrescar los datos de los registros del objeto de Access activo.	**ActualizarRegistro**
Requery	Permite actualizar los datos de la ventana volviendo a ejecutar la consulta fuente fundamentalmente.	**Actualizar**
SearchForRecord	Permite buscar un registro específico en una tabla, un formulario, un informe o una consulta.	**BuscarRegistro**
SetFilter	Permite aplicar un filtro sobre los registros del objeto de Access activo.	**DefinirFiltro**
SetOrderBy	Permite ordenar los registros del objeto de Access activo.	**DefinirOrdenPor**
ShowAllRecords	Permite desactivar los filtros y volver a visualizar todos los registros.	**VerTodosRegistros**

d. Importación y exportación de datos

Método	Descripción	Acción macro equivalente
CopyDatabaseFile	Permite copiar la base de datos conectada al proyecto activo en un archivo de base de datos de Microsoft SQL Server, para exportarla.	**CopiarArchivoBaseDeDatos**
OutputTo	Permite exportar un objeto de Access en otra base de datos de Access o en otro tipo de archivo (Excel, XML, etc.).	**CopiarA**
RunSaved Import Export	Permite ejecutar una especificación de importación o exportación guardada, creada a partir del Asistente de Importación o del Asistente de Exportación.	**EjecutarImportación ExportaciónGuardada**
SendObject	Permite enviar un objeto por correo electrónico	**EnviarObjeto**
Transfer Database	Permite importar o exportar una base de datos.	**TransferirBase**
Transfer SpreadSheet	Permite importar o exportar una hoja de cálculo.	**TransferirHojaCálculo**
Transfer SharePointList	Permite importar o relacionar los datos a partir de un sitio de Microsoft SharePoint Service.	**TransferirListaSharePoint**
TransferSQL Database	Permite transferir toda la base de datos de Microsoft SQL Server especificada a otra base de datos de SQL Server.	**TransferirBaseDeDatosSQL**
TransferText	Permite importar o exportar texto al formato ASCII.	**TransferirTexto**

e. Manipulación de la ventana activa

Método	Descripción	Acción macro equivalente
Maximize	Permite maximizar la ventana activa.	**Maximizar**
Minimize	Permite minimizar la ventana activa.	**Minimizar**
MoveSize	Permite mover o redimensionar la ventana activa.	**MoverRedimensionar**
Restore	Permite hacer que la ventana activa vuelva a su tamaño normal.	**Restaurar**

f. Modificación de la interfaz de Access

Método	Descripción	Acción macro equivalente
AddMenu	Permite crear una barra de menú o un menú contextual.	**AñadirMenú**
DoMenuItem	Permite visualizar el comando de menú apropiado para Microsoft Access.	**VerBarraHerramientas**
HourGlass	Permite visualizar el puntero del ratón como un reloj de arena o no.	**RelojArena**
LockNavigation Pane	Permite impedir que los usuarios eliminen objetos de la base de datos que se muestran en la pestaña de navegación.	**BloquearPestañaNavegación**
NavigateTo	Permite acceder a un grupo y a una categoría Pestaña de navegación, específicos.	**AccederA**
SetDisplayed Categories	Permite indicar las categorías que se muestran bajo la categoría en la barra de título de la pestaña de navegación.	**AccederA**
SetMenuItem	Permite definir el estado (activo o no, marcado o no) de elementos del menú personalizado o global, para la ventana activa.	**DefinirElementoMenú**
SetWarnings	Permite activar y desactivar la visualización de mensajes de alerta estándares.	**Avisos**
ShowToolBar	Permite visualizar y ocultar una barra de herramientas.	**VerBarraHerramientas**

g. Varios

Método	Descripción	Acción macro equivalente
Beep	Permite emitir un bip sonoro.	**Bip**
ClearMacroError	Permite eliminar la información relacionada con un error almacenado en un objeto MacroError.	**EliminarMacroError**
Echo	Permite visualizar u ocultar los resultados de una macro durante su ejecución.	**Echo**
PlainText	Permite eliminar el formateo de texto enriquecido de una cadena	No hay macro equivalente

	y devuelve una cadena de texto sin formato.	
SetDefaultWorkgroupFile	Permite definir el archivo especificado como archivo de grupo de trabajo por defecto.	No hay macro equivalente
SingleStep	Permite suspender la ejecución de la macro activa y abrir el cuadro de diálogo Paso a paso.	**PasoAPaso**

h. Acciones que no se tienen en cuenta por el objeto DoCmd

Hay un determinado número de acciones que no están directamente replicadas por el objeto DoCmd. Sin embargo, algunas acciones se siguen replicando con funciones o procedimientos de VBA.

Acción macro	Equivalencia posible en VBA
CuadroMsg	Podemos utilizar la función MsgBox.
EjecutarAplicación	Podemos utilizar la función Shell.
EjecutarCódigo	Podemos llamar directamente a la función en VBA.
EnviaTecla	Podemos utilizar la instrucción SendKeys.

i. Ejemplos

```
Sub Ejemplos()
    'a probar en modo paso a paso
    'emitir un bip
    DoCmd.Beep
    'se duplica la tabla
    DoCmd.CopyObject , "Ejemplo_Copia", acTabla,
"ENI_ANIMAL_TIPO_ANT"
    'se abre la tabla Ejemplo_Copia
    DoCmd.OpenTabla "Ejemplo_Copia", acViewNormal, acReadOnly
    'se cierra la tabla Ejemplo_Copia
    DoCmd.Close acTabla, "Ejemplo_Copia", acSaveYes
    'se destruye la tabla Ejemplo_Copia
    DoCmd.DeleteObject acTabla, "Ejemplo_Copia"
End Sub
```

3. El objeto Screen

El objeto Screen corresponde al objeto activo en la ventana, tanto si se trata de un formulario, un informe, un control o de una página de acceso a datos. Solo tiene propiedades.

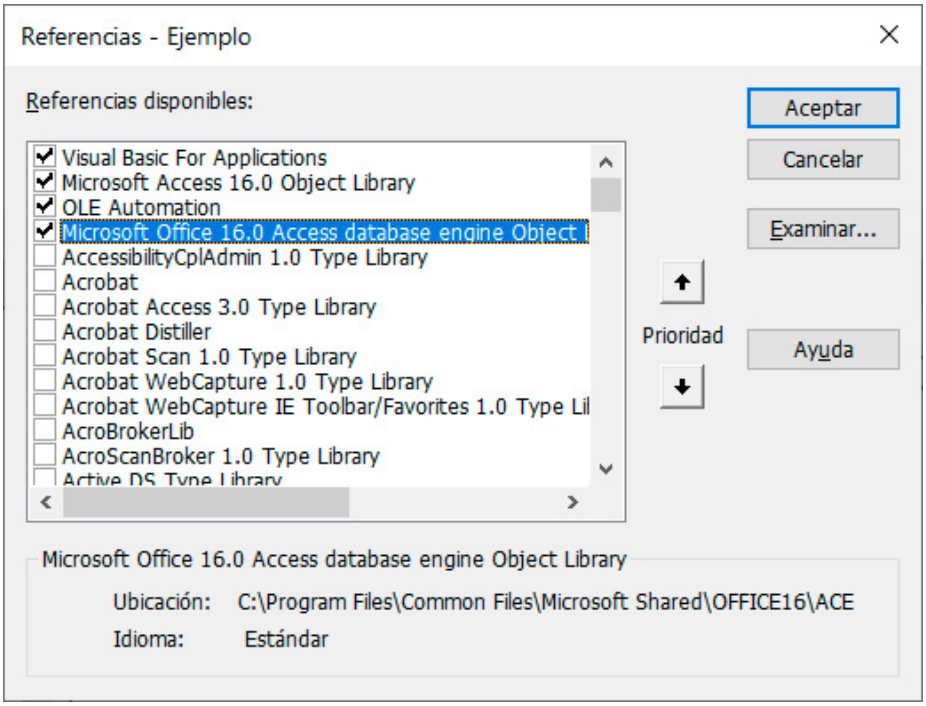
Propiedad	Descripción
ActiveControl	Devuelve el control activo.
ActiveDataAccessPage	Devuelve la página de acceso a los datos activa.
ActiveDataSheet	Devuelve la hoja de datos activa.

ActiveForm	Devuelve el formulario activo.
ActiveReport	Devuelve el informe activo.
Application	Permite el acceso al objeto Application.
MousePointer	Permite determinar el tipo de cursor de ratón.
Parent	Devuelve el objeto padre de un subformulario, control, etc.
PreviousControl	Devuelve el control que estaba activo antes del control activo.

4. La colección References

La colección References corresponde a la lista de librerías (objeto Reference) que provienen de otras aplicaciones.

En la interfaz de Access, podemos visualizar esta lista haciendo clic en el menú **Herramientas - Referencias**.



a. Propiedades

Propiedad	Descripción
Count	Devuelve el número de librerías.
Parent	No se utiliza.

b. Métodos

Método	Descripción
AddFromFile	Permite añadir una librería a partir de la ubicación del archivo.
AddFromGUID	Permite añadir una librería a partir de un identificador global único en el registro de Windows.

Item	Devuelve un objeto Reference en función de su posición en la lista de librerías.
Remove	Permite eliminar una librería en la colección References.

5. El objeto Reference

Todas las propiedades del objeto Reference son de solo lectura.

a. Propiedades

Propiedad	Descripción
BuiltIn	Devuelve un booleano que indica si la librería se presenta por defecto para permitir el buen funcionamiento de Access.
Colección	Devuelve la colección References a la que pertenece el objeto Reference.
FullPath	Devuelve la ubicación del archivo de librería.
Guid	Devuelve el identificador global único de la librería.
IsBroken	Devuelve un booleano que indica si la librería está ausente en el proyecto Access.
Kind	Devuelve el tipo de librería (archivo o proyecto de VB).
Major	Devuelve el número de versión principal de la aplicación asociada a la librería.
Minor	Devuelve el número de versión secundaria de la aplicación asociada a la librería.
Name	Devuelve el nombre de la librería.

b. Ejemplo

El siguiente código recorre el conjunto de librerías y muestra el nombre de las librerías identificadas como ausentes.

```
Sub Lista_AUSENTE()
Dim Ref As Reference
For Each Ref In Application.References
    If Ref.IsBroken Then
          Debug.Print Ref.Name
    End If
Next Ref
End Sub
```

6. La colección Printers

La colección Printers contiene la lista de las impresoras disponibles (objeto Printer) en el sistema actual.

a. Propiedades del objeto Printer

Propiedad	Hace referencia a
BottomMargin	Margen inferior de una página.
ColorMode	Ver en color (acPRCMColor) o en tonos grises (acPRCMMonochrome).

ColumnSpacing	Espaciado vertical de las secciones **Detalle** de un formulario.
Copies	Número de copias que se va a imprimir.
DataOnly	Permite la impresión solo de los datos de una tabla o de una consulta en modo **Hoja de datos**.
DefaultSize	Devuelve o define un booleano que indica si el tamaño de la sección **Detalle** en modo **Creación** se utiliza durante la impresión de un formulario o de un informe.
DeviceName	Nombre de la impresora.
DriverName	Nombre del controlador de la impresora.
Duplex	Modo de administración de la impresión (en una o dos caras del papel).
ItemLayout	Indica si la impresora realiza una impresión de las columnas vertical y horizontalmente o a la inversa.
ItemsAcross	Devuelve o define el nombre de columnas que se han de imprimir (informes o etiquetas).
ItemSizeHeight	Devuelve o define la altura de la sección (en twips).
ItemSizeWidth	Devuelve o define la anchura de la sección (en twips).
LeftMargin	Margen izquierdo de una página.
Orientation	Orientación de la impresión (acPRORPortrait o acPRORLandscape).
PaperBin	Bandeja de impresora utilizada durante la impresión.
PaperSize	Tamaño del papel utilizado durante la impresión.
Port	Puerto de conexión de la impresora.
PrintQuality	Resolución de la impresora durante la impresión.
RightMargin	Margen derecho de una página.
RowSpacing	Devuelve o define el espacio horizontal de las secciones (en twips).
TopMargin	Margen superior de una página.

b. Ejemplo

El siguiente código muestra la lista de las impresoras disponibles:

```
Sub Impresoras()
Dim impresora As Printer
    For Each impresora In Application.Printers
          Debug.Print impresora.DeviceName
        Next
End Sub
```

Manipular los objetos en VBA

1. Instrucción Set

Al contrario que la asignación de un valor a una variable de tipo sencillo (numérico, booleano, fecha, etc.), la asignación de un valor a una variable cuyo tipo es un objeto se realiza con la palabra clave `Set`.

La sintaxis general es la siguiente:

```
Set UnObjeto = [New] InstrucciónGeneraElObjeto
```

La palabra clave `New` permite crear una nueva instancia de un objeto. Si el objeto ya existe, no es necesario utilizar esta palabra clave. `InstrucciónGeneraElObjeto` puede corresponder al nombre de un objeto o a una variable de objeto del mismo tipo.

Por ejemplo aquí, si se desea asignar a la variable `tbl` la tabla `ENI_ANIMAL_ANI`, el código sería el siguiente:

```
Dim tbl As TableDef
Set tbl = CurrentDb.TableDefs("ENI_ANIMAL_ANI")
```

Reinstanciación de la variable objeto

Es posible instanciar de nuevo una variable objeto, gracias a la siguiente instrucción:

```
Set UnObjeto = Nothing
```

2. Instrucción With/End With

A la hora de modificar las diferentes propiedades de un objeto, puede ser largo y pesado para el programador repetir la ruta de acceso a este objeto, para cada propiedad que se desea modificar.

Por ejemplo, con las líneas de operación de un formulario:

```
'El formulario F_ANIMALinicialmente está abierto
Forms("F_ANIMAL").Caption = "Nueva ficha"
Forms("F_ANIMAL").Controls("ANI_FECHA_LLEGADA").Value=Fecha
Forms("F_ANIMAL").Controls("ANI_FECHA_LLEGADA").SetFocus
```

La instrucción `With` permite realizar todas estas operaciones del mismo objeto sin repetir el código. La sintaxis general es la siguiente:

```
With MiObjeto
     'instrucciones utilizando los métodos y propiedades de MiObjeto
End With
```

Por tanto, el ejemplo anterior se podría escribir de la siguiente manera:

```
With Forms("F_ANIMAL")
     .Caption = " Nueva ficha "
     .Controls("ANI_FECHA_LLEGADA").Value = Fecha
     .Controls("ANI_FECHA_LLEGADA").SetFocus
End With
```

Para algunas propiedades de los objetos, que son a su vez objetos, se puede anidar dentro de una estructura `With / End With` otra estructura `With / End With`. Retomando el ejemplo anterior:

```
With Forms("F_ANIMAL")
     .Caption = "Nueva ficha"
     With .Controls("ANI_FECHA_LLEGADA")
            .Value = Fecha
            .SetFocus
     End With
End With
```

3. El bucle For Each

Del mismo tipo que el resto de los bucles `For Next` y `Do Loop`, es posible comprobar los elementos de una colección gracias al bucle `For Each Next`. La sintaxis general de un bucle como este es la siguiente:

```
For Each Elemento In MiColección
    Instrucciones
    [Exit For]
Next Elemento
```

Por ejemplo, para recorrer cada uno de los campos de una tabla y visualizar su nombre en la ventana de ejecución, se podría utilizar una sintaxis como la siguiente:

```
Sub Lista_Campos()
Dim Db As DAO.Database
Dim tbl As DAO.TableDef
Dim Fld As DAO.Field
Set Db = CurrentDb
Set tbl = Db.TableDefs("ENI_ADOPCION_ADO")
For Each Fld In tbl.Fields
    Debug.Print Fld.Name
Next
End Sub
```

4. Instrucción TypeOf

Cuando se manipulan los objetos en VBA, es posible determinar el tipo de objeto que se encuentra en una variable en un momento dado. La función `TypeOf` sirve para probar el tipo de un objeto. Su sintaxis general es la siguiente:

```
If TypeOf UnObjeto Is TypeObject Then
    Instrucciones
End If
```

Por ejemplo, para probar si una variable `MiVariable` contiene un objeto de tipo `ComboBox`, el código sería el siguiente:

```
If TypeOf MiVariable Is ComboBox Then
    Instrucciones
End If
'Otra opción para determinar su tipo mediante enumerados
If MiVariable.ControlType = acComboBox Then
    Instrucciones
End If
```

Las clases de objetos

Una clase es un modelo que utilizará el programa para crear nuevos objetos. Los objetos tienen propiedades similares a su modelo, podrán realizar las mismas acciones que su modelo y podrán actuar ante los mismos eventos que los de su modelo.

En función de las necesidades, cada objeto podrá tener sus características propias (título de un libro, número de páginas, precio, etc.) y un comportamiento específico. Cualquier objeto de Access viene de una clase.

1. Los módulos de clases

Para manipular los objetos que el desarrollador creará, se podrán utilizar los módulos de clases.

Los módulos de clases se definen de la siguiente manera:

- Los módulos de clases tienen el nombre del objeto que definen;
- Tienen procedimientos y funciones públicos `Public Sub` y `Public Function`, que definen los métodos propios del objeto;
- Tienen las funciones `Property Get`, que sirven para leer los valores de sus propiedades, así como los procedimientos `Property Set/Property Let` para definir los valores de estas propiedades.

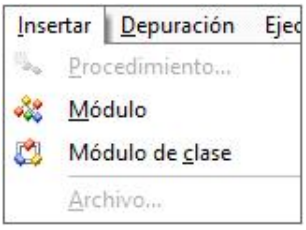
Una vez que se definen estos elementos en un módulo de clase, podemos crear un nuevo objeto correspondiente a esta definición. La sintaxis general para conseguirlo es la siguiente:

```
Dim NombreObjeto As New NombreDeClasePersonalizada
```

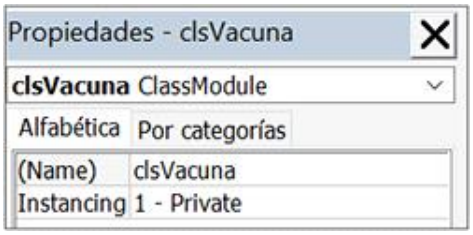
La llamada a los módulos de clase sigue estando restringida solo a los desarrolladores más experimentados.

En los siguientes ejemplos, se define una clase que representa un animal.

Empezamos con una clase llamada `clsVacuna`. En el menú **Insertar**, se selecciona **Módulo de clase**.



En la zona **Propiedades**, se modifica el nombre de la clase como **clsVacuna**.

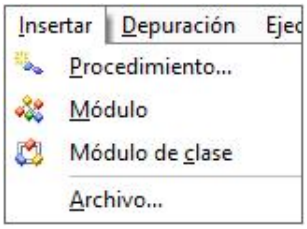


2. Las propiedades

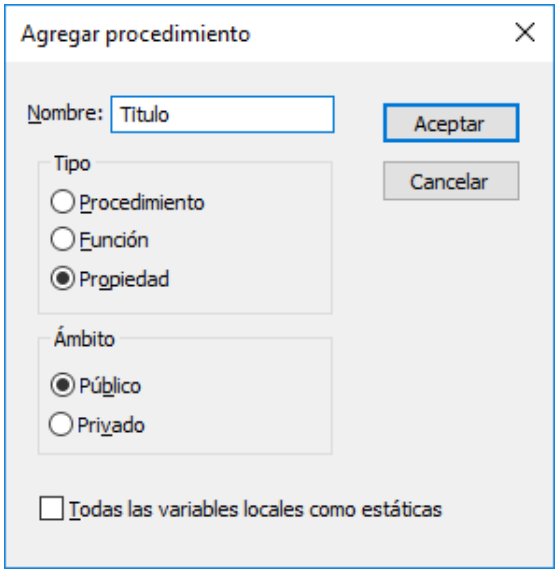
Las propiedades se corresponden con las instrucciones que permiten leer y definir las características de un objeto.

La creación de una propiedad se desarrolla en dos etapas: se define una variable privada dentro de la clase y después se definen, los bloques de propiedades `Property Get` y `Property Let`. La lectura de la propiedad se hace por medio de `Property Get`, y su escritura con `Property Let`.

En la interfaz de VBE, se activa el módulo de clase y seleccione el menú **Insertar - Procedimiento**:



Podemos crear la propiedad `Titulo` indicándolo en la interfaz, como se muestra a continuación:



Adaptando el código, al final se tienen las propiedades `ID`, `Nombre`, `Obligatorio` y `TipoAnimal`, definidas como sigue:

```
'Declaración de las variables privadas
Private pID As Long
Private pNombreVacuna As String
Private pTipoAnimalID As Long
Private pObligatorio As Boolean

'El identificador ID
Public Property Get ID() As Long
    ID = pID
End Property
```

```
Public Property Let ID(newID As Long)
    pID = newID
                                    End Property

'El nombre de la vacuna
Public Property Get Nombre() As String
    Nombre = pNombreVacuna
End Property

Public Property Let Nombre(newNombre As String)
    If Len(newNombre) > 0 Then
        pNombreVacuna = newNombre
    Else
        Err.Raise Number:=vbObjectError + 1, _
            Description:="El nombre de la vacuna no puede ser vacío"
    End If
End Property

'El tipo de animal
Public Property Get TipoAnimal() As Long
    TipoAnimal = pTipoAnimalID
End Property

Public Property Let TipoAnimal(newID As Long)
    pTipoAnimalID = newID
End Property

'La vacuna es obligatoria o no
Public Property Get Obligatorio() As Boolean
    Obligatorio = pObligatorio
End Property

Public Property Let Obligatorio(newObligatorio As Boolean)
    pObligatorio = newObligatorio
End Property
```

Observe que algunas propiedades se pueden definir en modo lectura y escritura si tienen `Property Get` y `Property Let/Set`, así como en modo lectura solo (únicamente una `Property Get`), incluso en modo escritura solo (únicamente una `Property Let/Set`). Por ejemplo, la edad de una persona es una propiedad calculada a partir de su fecha de nacimiento, por lo que no se puede modificar esta información.


Se utilizará `Property Let` para los tipos de datos "básicos" y `Property Set` para los objetos.

Cuando se crea una jerarquía entre los objetos, es posible rastrear dicha jerarquía. Para esto, se usa la propiedad `Parent`. Podrá ver cómo se utiliza en la sección Ejemplo de clase personalizada, en este capítulo.

3. Los métodos

Además de las propiedades, es posible definir acciones que será capaz de realizar nuestro objeto personalizado. Estas acciones corresponden a funciones o procedimientos públicos, que por tanto se podrán llamar.

```
'Función que devuelve si un animal está afectado o no por la vacuna
Public Function AnimalAfectado(IDAnimal As Long) As Boolean
    AnimalAfectado = (DLookup("ANI_ANT_ID", "ENI_ANIMAL_ANI",
"ANI_ID=" & IDAnimal) = TipoAnimal)
End Function
```


Observe que podemos utilizar la palabra clave Me, que hace referencia al objeto que estamos definiendo.

4. Los eventos

Igual que los objetos de Access reaccionan a algunos eventos (carga, clic, etc.), es posible definir una serie de eventos, a los que nuestro objeto será capaz de reaccionar. Un evento es un objeto `Event` en VBA.

No existe constructor o destructor, como sucede en otros lenguajes como C++ o Java. Sin embargo, existen dos eventos implementados en los módulos de clases básicas, `Inicializar` y `Finalizar`.

El evento `Inicializar` se desencadena durante la creación del objeto y el evento `Finalizar`, durante la destrucción del objeto. Más abajo hay ejemplos de instrucciones que se ejecutarán durante la creación y destrucción de los objetos `clsVacuna`.

```
'Eventos de creación y destrucción
Private Sub class_Inicialize()
    Debug.Print "Evento Inicializar - Creación de un objeto Vacuna"
End Sub

Private Sub class_Terminate()
    Debug.Print "Evento Finalizar - Destrucción de un objeto
Vacuna: " & Me.Nombre
End Sub
```

Además de estos dos eventos básicos, es posible personalizar otros eventos para la clase.

La declaración de un evento dentro de una clase se hace de la siguiente manera:

```
Public Event NombreDeEvento()
```

El evento tiene el nombre `NombreDeEvento` y se podrá cargar desde el exterior de la clase, teniendo como ámbito `Public`.

Por ejemplo, podemos crear un evento `AnimalesAVacunar`:

```
Public Event AnimalesAVacunar()
```

Una vez que se declara el evento, solo falta desencadenarlo cuando la condición se rellene. Los eventos se desencadenan con la instrucción `RaiseEvent`. La sintaxis general es la siguiente:

```
RaiseEvent NombreDeEvento([argumento])
```

El evento que se desencadena lleva el nombre `NombreDeEvento` en el módulo de clase y se debe declarar correctamente. También `argumento` se puede proporcionar con el evento.

En nuestro caso, el desencadenamiento del evento `AnimalesAVacunar` se escribirá como se muestra a continuación:

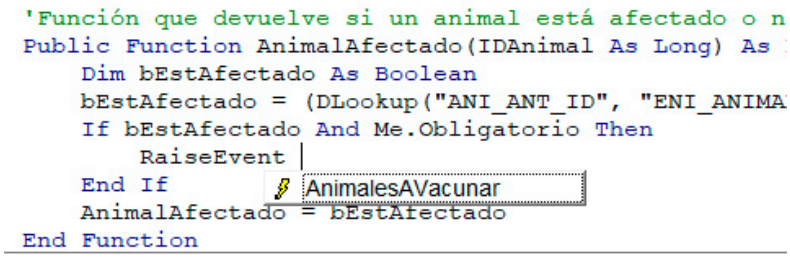
```
RaiseEvent AnimalesAVacuar
```

Para esto, modificamos la función `AnimalesAfectado`, de manera que se pueda probar los animales que se deben vacunar:

```
'Función que devuelve si un animal está afectado o no por
la vacuna
Public Function AnimalAfectado(IDAnimal As Long) As Boolean
Dim bEstaAfectado As Boolean
    bEstaAfectado = (DLookup("ANI_ANT_ID", "ENI_ANIMAL_ANI",
"ANI_ID=" & IDAnimal) = Me.TipoAnimal)
    If bEstaAfectado And Me.Obligatorio Then
         RaiseEvent AnimalesAVacunar
    End If
    AnimalAfectado = bEstaAfectado
End Function
```

Al buscar si un animal está afectado por la vacuna, tendremos el evento `AnimalesaVacunar`, que se desencadenará cuando el animal esté afectado y la vacuna sea obligatoria.

Observe que en la interfaz de VBE, los eventos que se desencadenan están precedidos de un pequeño rayo durante la escritura semiautomática, como se muestra en el siguiente ejemplo:

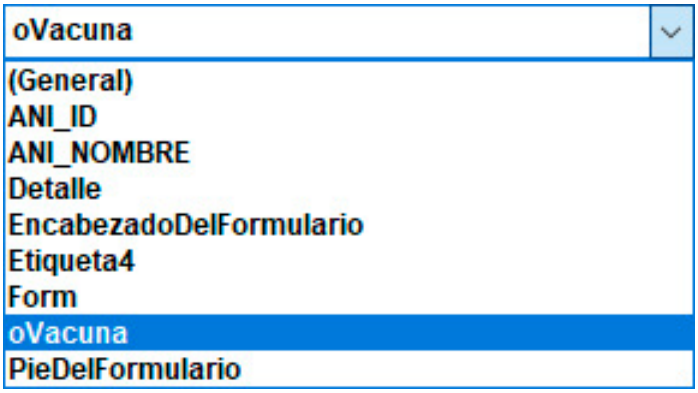


Para gestionar el evento del lado del programa que utilizará este objeto, será necesario usar la palabra clave `WithEvents` durante la declaración de la variable objeto. La sintaxis general es:

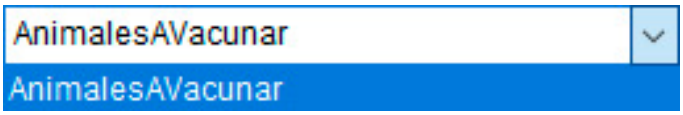
```
Dim WithEvents oVacuna As clsVacuna
```

La palabra clave `WithEvents` no se puede usar al mismo tiempo que la palabra clave `New`.

Se podrá comprobar la presencia de la variable oVacuna en la zona de lista desplegable, en la parte izquierda de la interfaz de VBE.



Si se selecciona el objeto oVacuna, aparecerán los eventos que desencadena el objeto, en la zona de la lista desplegable de la derecha.



Así como la creación del procedimiento correspondiente a este evento:

```
Private Sub oVacuna_AnimalesAVacunar()

End Sub
```

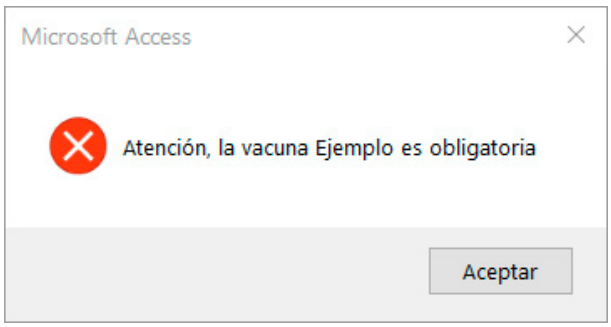
Solo falta definir las instrucciones que se deben ejecutar cuando el evento se desencadene:

```
Private Sub oVacuna_AnimalesAVacunar()
    MsgBox "Atención, la vacuna " & oVacuna.Nombre & _ " es
obligatoria", vbCritical + vbOKOnly
End Sub
```

Si colocamos un botón BtnCompruebaVacuna en nuestro formulario, al hacer el clic en este botón, podremos ejecutar el siguiente código:

```
Private Sub BtnCompruebaVacuna_Click()
    Set oVacuna = New clsVacuna
    With oVacuna
            .ID = 1
            .Nombre = "Ejemplo"
            .TipoAnimal = Me.Controls("ANI_ANT_ID") .Value
            .Obligatorio = True
            .Animalafectado Me.Controls ("ANI_ID") .Value
    End With
End Sub
```

Cuando se hace clic en el botón, aparecerá el siguiente mensaje:



5. Los errores

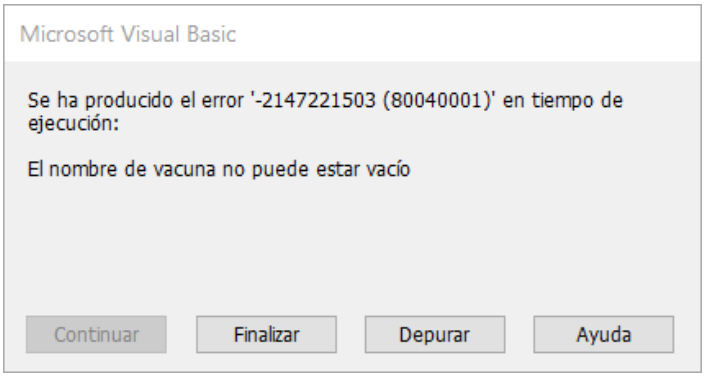
Para que el programa que manipula el objeto esté informado en caso de error, por ejemplo ante un nombre de vacuna vacío, podemos generar un error. Para esto se usa el método `Raise` del objeto error `Err`. La sintaxis general es la siguiente:

```
Err.Raise NumeroError, [OrigenError], [DescripcionError],
[ArchivodeAyuda], [ContextoAyuda]
```

El número del error se puede personalizar, para lo que se utilizará la propiedad `Nombre` en el siguiente código:

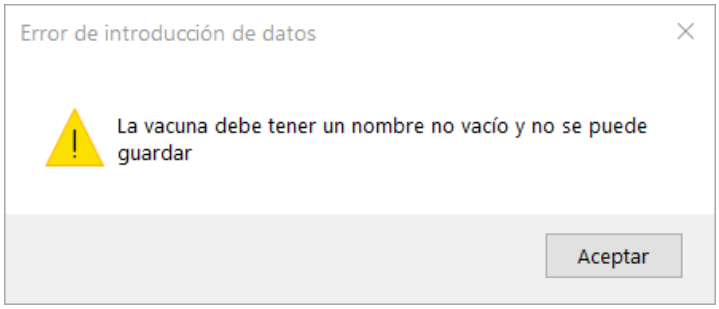
```
Public Property Let Nombre(newNombre As String)
    If Len(newNombre > 0 Then
            pNomVacuna = newNombre
    Else
            Err.Raise Number:=vbObjectError + 1, _
                  Description:="El nombre de la vacuna no puede ser vacío "
    End If
End Property
```

Cuando se intenta poner un nombre de vacuna vacío, se muestra el siguiente mensaje de error:



El error se puede generar por el código llamador, con una administración `On Error GoTo`. Aquí se muestra un mensaje personalizado:

```
Private Sub BtnCompruebaVacunaFalso_Click()
    Set oVacuna = New clsVacuna
    On Error GoTo GestionError
    With oVacuna
           .ID = 0
           .Obligatorio = True
           .Nombre = ""
           .TipoAnimal = 1
    End With
GestionError:
    Select Case Err.Number
           Case -2147221503
                MsgBox "La vacuna debe tener un nombre no vacío
y no se puede guardar", _
                       vbOKOnly + vbExclamation,
"Error de introducción de datos"
               Resume Next
    End Select
End Sub
```



6. Ejemplo de clase personalizada

En el ejemplo siguiente, se crea una clase `clsAnimales`, que representa una colección de animales, y tiene características como la fecha de nacimiento, el nombre del animal e incluso el número de animales total. Desde este objeto se podrá acceder a la lista de los animales, disponible como una colección de objetos `clsAnimal`, llamada `Animales`.

a. La clase clsAnimal

En un primer momento, se crea una clase `clsAnimal`, con un identificador, un nombre y un tipo de animal. La clase devuelve, en modo solo lectura, la edad del animal.

Por tanto, el código del módulo de clase `clsAnimal` será el siguiente:

```
Private pID As Long
Private pTipoAnimalID As Long
Private pNombre As String
Private pFecNacimiento As Date

'El identificador ID
Public Property Get ID() As Long
    ID = pID
```

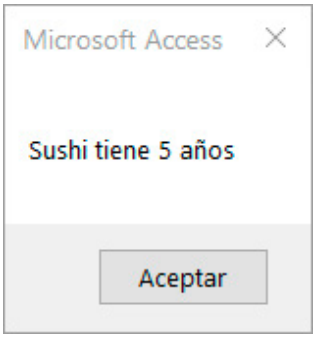
```
End Property

Public Property Let ID(newID As Long)
     pID = newID
                                                    End Property

'Identificador Tipo Animal
Public Property Get TipoAnimalID() As Long
    TipoAnimalID = pTipoAnimalID
End Property

Public Property Let TipoAnimalID(newID As Long)
    pTipoAnimalID = newID
End Property

Public Function TipoAnimal() As String
    TipoAnimal = Nz(DLookup("ANT_TIPO_ANIMAL",
"ENI_ANIMAL_TIPO_ANT", "ANT_ID=" & pTipoAnimalID), "No definido")
End Function

'Nombre del animal
Public Property Get Nombre() As String
    Nombre = pNombre
End Property

Public Property Let Nombre(newNombre As String)
    pNombre = newNombre
End Property

'Fecha de nacimiento
Public Property Get FechaNacimiento() As Date
    FechaNacimiento = pFecNacimiento
End Property

Public Property Let FechaNacimiento(newDate As Date)
    pFecNacimiento = newDate
End Property

'Función en modo solo lectura
Public Function Edad() As Integer
    Edad = DateDiff("yyyy", pFecNacimiento, Date)
    If Date < DateSerial(Year(Now), Month(Me.FechaNacimiento), _
        Day(Me.FechaNacimiento)) Then
         Edad = Edad - 1
    End If
    Edad = CInt(Edad)
End Function
```

Se puede probar la coherencia de la clase, comprobando la creación de un participante:

```
Sub TestAnimal()
    Dim a As clsAnimal
    Set a = New clsAnimal
```

```
    With a
        .Nombre = "Sushi"
        .FechaNacimiento = #1/4/2014#
    End With
    MsgBox a.Nombre & " tiene " & a.Edad & " años."
    Set a = Nothing
End Sub
```


Que mostrará el siguiente mensaje.



b. La colección Animales

Para gestionar la lista de animales de una familia, vamos a crear una colección de objetos `clsAnimal`, que llamaremos `Animales`. Hay que crear las propiedades, funciones y procedimientos clásicos de la colección: `Add`, `Count`, `Item` y `Remove`:

```
Option Compare Database

Private pcolAnimales As Collection

Public Function CrearAnimal(ID As Long, strNombre As String,
dtFechaNacimiento As Date, lTipoAnimal As Long)
    Dim oAnimal As clsAnimal
    Set oAnimal = New clsAnimal
    With oAnimal
        .ID = ID
        .Nombre = strNombre
        .FechaNacimiento = dtFechaNacimiento
        .TipoAnimalID = lTipoAnimal
    End With

    Set CrearAnimal = oAnimal

    If Not (oAnimal Is Nothing) Then Set oAnimal = Nothing

End Function

Public Function Count() As Long
    Count = pcolAnimales.Count
End Function

Public Sub Add(ByRef objAnimal As clsAnimal, _
        Optional ByVal Key As String = "")
```

```
'Si no se proporciona ninguna clave, se genera una automáticamente
    If Len(Key) = 0 Then
        Key = objAnimal.Nombre & _
            Format(objAnimal.FechaNacimiento, "yyyy_mm_dd")
    End If
    pcolAnimales.Add objAnimal, Key
End Sub

Public Sub Remove(ANI_ID As Long)
    Dim Index As Long
    Dim i As Long
    For i = 1 To pcolAnimales.Count
        If pcolAnimales.Item(i).ID = ANI_ID Then
           Index = i
        End If
    Next i
    pcolAnimales.Remove Index
End Sub

Public Function Item(ByVal Index As Variant) As clsAnimal
    Set Item = pcolAnimales.Item(Index)
End Function

Private Sub Class_Initialize()
    Set pcolAnimales = New Collection
End Sub
```

Para probar esta colección, se puede ejecutar este código (las visualizaciones de información se harán en la ventana de ejecución):

```
Sub testColeccion()
Dim colAnimales As clsAnimales
Dim i As Long
   Set colAnimales = New clsAnimales
   With colAnimales
        .Add . CrearAnimal(1, "Chamalow", #7/1/2010#, 2)
       Debug.Print "Número de elemento(s) en la colección: " & .Count
       .Add .CrearAnimal(2, "Sushi", #4/1/2014#, 2)
       Debug.Print "Número de elemento(s) en la colección: " & .Count
       .Add .CrearAnimal(3, "Mogwai", #4/1/2017#, 1)
       Debug.Print "Número de elemento(s) en la colección: " & .Count
       .Add .CrearAnimal(4, "Michoko", #7/1/2017#, 1)
       Debug.Print "Número de elemento(s) en la colección: " & .Count

       Debug.Print "nos desplazamos por la colección: "
       For i = 1 To .Count
           Debug.Print .Item(i).Nombre
       Next i

       Debug.Print "eliminación de un elemento"
        .Remove 2 'lo sinto Sushi
       Debug.Print "Número de elemento(s) en la colección: " & .Count

       For i = 1 To .Count
```

```
            Debug.Print .Item(i).Nombre
        Next i
    End With
    Set colAnimales = Nothing
End Sub
```

Introducción

Cuando queremos manipular los datos en una base de datos, tanto para acceder a la estructura de las tablas, consultas o directamente a los registros, el uso de objetos de acceso a los datos es muy útil. Podemos trabajar con datos locales, así como con datos remotos.

Históricamente hay dos modelos. El objetivo de este capítulo es revisar estos modelos. Antes de la versión Access 2000, solo existía el modelo DAO (*Data Access Objects*). Con la versión Access 2000, aparece un segundo modelo: ADO (*ActiveX Data Objects*).

Es muy recomendable utilizar el modelo más reciente, ADO, que presenta las ventajas respecto a su predecesor: un mejor soporte de Microsoft SQL Server, mejor rendimiento en modo cliente/servidor y un código más corto y sencillo de escribir.

Para la creación de una nueva aplicación, el uso del modelo DAO sigue siendo posible, pero es preferible trabajar con el modelo ADO, que hace permanente la aplicación.

Para terminar, en caso de una recuperación de una aplicación existente, es posible modificar y actualizar el código para hacerla evolucionar desde el modelo DAO al modelo ADO.

DAO

1. Introducción

DAO (*Data Access Objects*) es una librería de VBA que agrupa un conjunto de objetos, permitiendo acceder a una base de datos. Se utilizan estos objetos para acceder, al mismo tiempo, a las estructuras de los datos (tablas y consultas) y a los registros. El modelo tiene en cuenta dos espacios de trabajo posibles o entornos de base de datos:

El espacio de trabajo **Microsoft Jet**, que permite acceder a las bases de datos Access, a los servidores de bases de datos **ODBC** (*Open DataBase Connectivity*) y a las bases de datos externas, como por ejemplo Microsoft Excel, Paradox o dBase, accesibles gracias a un controlador **ISAM** (*Indexed Sequential Access Method*).

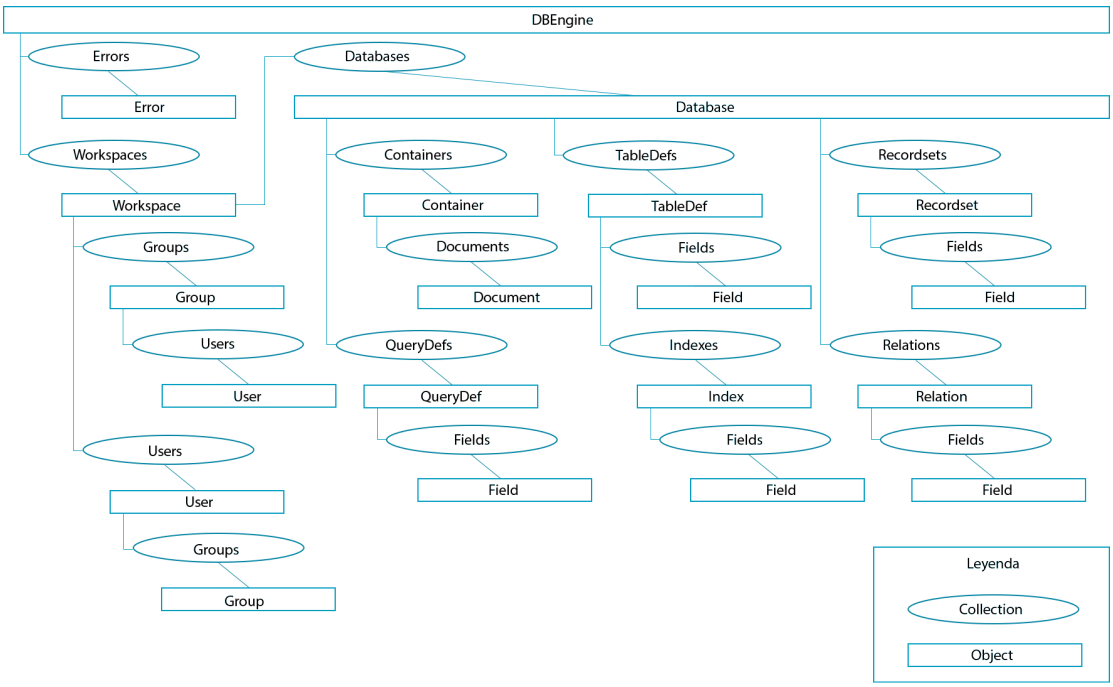
El espacio de trabajo **ODBCDirect**, que permite trabajar directamente con los servidores de datos ODBC sin pasar por el motor de base de datos Microsoft Office. Se recomienda este espacio de trabajo para ejecutar consultas, procedimientos almacenados o funciones específicas de ODBC en un servidor remoto tipo SQL Server.

Para utilizar la librería de objetos DAO, es necesario añadir la referencia **Microsoft DAO 3.6 Object Library** en la lista de las referencias. Puede realizar esta operación seleccionando la pestaña **Referencias** del menú **Herramientas**. Si intenta encontrar directamente esta referencia para marcarla en la lista de las referencias disponibles, puede hacer clic en el botón **Examinar** y buscar su ubicación. El archivo dao360.dll generalmente está en la siguiente carpeta: C:\Program Files (x86)\Common Files\Microsoft Shared\DAO

Adicionalmente, el archivo de ayuda DAO360.chm proporcionado con los ejemplos de este libro, le será útil para encontrar información detallada del uso del modelo de objetos DAO.

2. Modelo de datos DAO

A continuación encontrará el modelo jerárquico de objetos del modelo DAO:



3. Descripción de las colecciones DAO

Para cada uno de los tipos de objeto DAO, podemos comprobar la existencia de una colección, excepto para el

objeto DbEngine.

Colección	Conjunto de los objetos
Containers	Container, que agrupa los siguientes objetos de la base de datos: base de datos, formularios, módulos, relaciones, informes, macros, información interna de las relaciones, tablas y consultas.
Databases	Database, base de datos abierta en el espacio de trabajo.
Documents	Document, nombre genérico dado al contenido de los Container.
Errors	Error, los errores del motor de acceso a los datos DBEngine.
Fields	Field, campo de una tabla (TableDef), consulta (QueryDef), índice (Index), conjunto de registros (Recordset) o relación (Relation).
Groups	Group, grupo de usuarios de un WorkSpace o grupo de usuarios al que pertenece un usuario (User).
Indexes	Index, índice de una tabla (TableDef).
Parameters	Parameter, argumento de una consulta (QueryDef).
Properties	Property, propiedad de un objeto.
QueryDefs	QueryDef, consulta de una base de datos (Database).
Recordsets	Recordset, conjunto de registros abierto de una base de datos (Database).
Relations	Relation, relación de una base de datos (Database).
TableDefs	TableDef, tabla de una base de datos (Database).
Users	User, usuario de un espacio de trabajo (WorkSpace) o miembro de un grupo de usuarios (Group).
Workspaces	Workspace, espacio de trabajo activo.

a. Propiedades

El conjunto de estas colecciones disponen de las siguientes propiedades:

Propiedad	Descripción
Count	Determina el número de elementos presentes en la colección.
Item	Determina un elemento específico de una colección, definida tanto por su nombre como por su identificador dentro de la colección.

Por ejemplo, para visualizar el nombre de cada una de las consultas de la base de datos actual:

```
Sub ListarConsultas()
Dim i As Integer
For i = 0 To CurrentDb.QueryDefs.Count - 1
    Debug.Print CurrentDb.QueryDefs.Item(i).Name
Next i
End Sub
```

b. Métodos

Para las colecciones que no se gestionan directamente por el motor de la base de datos (Containers,

Databases, Documents, Errors, Parameters y Recordsets), es posible utilizar los siguientes métodos:

Método	Descripción
Append	Permite añadir un nuevo elemento a la colección.
Delete	Permite eliminar un elemento de la colección.
Refresh	Permite volver a actualizar el contenido de la colección.

4. El objeto Workspace

El objeto Workspace representa el espacio de trabajo que sirve para definir una sesión de trabajo para un usuario.

a. Propiedades

Propiedad	Descripción
DefaultCursorDriver	Permite modificar los modos de acceso de las conexiones creadas más adelante (devuelve una de las constantes CursorDriverEnum).
IsolateODBCTrans	Define o devuelve un booleano que indica si, durante el uso de varias transacciones, estas están aisladas las unas de las otras.
LoginTimeout	Define o devuelve un valor numérico, correspondiente al tiempo transcurrido (en segundos), antes de que aparezca un mensaje de error durante un intento de conexión a una base de datos ODBC. Su valor por defecto es 20.
Name	Nombre del objeto Workspace (devuelve por ejemplo #Default Workspace#).
Type	Corresponde al tipo de conexión del objeto Workspace. Las dos constantes correspondientes a los entornos son dbUseJet para el motor de base de datos Microsoft Jet y dbUseODBC para la conexión a una fuente de datos ODBC.
UserName	Nombre del propietario del objeto Workspace (devuelve, por ejemplo, "admin").

b. Métodos

Método	Descripción
BeginTrans	Permite iniciar una nueva transacción.
Close	Permite cerrar el espacio de trabajo (Workspace).
CommitTrans	Permite terminar y validar la transacción actual.
CreateDatabase	Permite crear un nuevo objeto Database.
CreateGroup	Permite crear un nuevo objeto Group.
CreateUser	Permite crear un nuevo objeto User.
OpenDatabase	Permite abrir una base de datos (Database).
RollBack	Permite anular y terminar la transacción actual.

c. Colecciones

El objeto Workspace contiene las colecciones Databases, Groups, Properties y Users.

5. El objeto Database

El objeto Database representa una base de datos abierta.

a. Propiedades

Propiedad	Descripción
CollatingOrder	Define el orden del collation utilizado, es decir, el método de comparación de las cadenas de caracteres. Este orden se define por un valor numérico, cuyas constantes se proporcionan en el anexo de este libro. El valor más frecuente para las aplicaciones en castellano hablantes es dbSortGeneral.
Connect	Define la cadena de caracteres correspondiente a la cadena de conexión a la base de datos. La sintaxis de conexión a una base de datos ODBC sería igual que la siguiente: ODBC; DATABASE=base_datos; UID=usuario; PWD=contraseña; DSN=nombre_fuente_datos; [LOGINTIMEOUT=segundas;]
DesignMasterId	Indica el GUID usando 16 bytes de la réplica maestra de un juego de réplicas (solo en un espacio de trabajo de Access).
Name	Devuelve el nombre de la base de datos.
RecordsAffected	Indica el número de registros afectados por la última ejecución de la consulta (método Execute).
Replicable	Indica si la base de datos se puede replicar.
ReplicaID	Indica el ID de un réplica.
Transacciones	Devuelve un booleano que indica si la base de datos soporta las transacciones.
Updatable	Devuelve un booleano que indica si la base de datos se puede modificar.
Version	Devuelve el identificador correspondiente a la versión de la base de datos. Microsoft Access 2019 devuelve el valor 17.

b. Métodos

Método	Descripción
Close	Permite cerrar el objeto Database.
CreateProperty	Permite crear un nuevo objeto propiedad (Property).
CreateQueryDef	Permite crear un nuevo objeto consulta (QueryDef).
CreateRelation	Permite crear un nuevo objeto relación (Relation).
CreateTableDef	Permite crear un nuevo objeto tabla (TableDef).
Execute	Permite ejecutar una consulta de tipo acción o una instrucción SQL.
MakeReplica	Permite crear una réplica de la base de datos.
NewPassword	Permite definir una nueva contraseña para una base de datos Microsoft Jet.
OpenRecordset	Permite crear un nuevo objeto conjunto de registros (Recordset).
PopulatePartial	Permite sincronizar las modificaciones añadidas entre una réplica parcial y una réplica integral.
Synchronize	Permite sincronizar dos réplicas.

c. Colecciones

El objeto `Database` contiene las colecciones `Containers`, `Properties`, `QueryDefs`, `Recordsets`, `Relations` y `TableDefs`.

6. El objeto TableDef

El objeto `TableDef` representa una tabla guardada en local o una tabla adjunta creada en una base externa.

a. Propiedades

Propiedad	Descripción
`Attributes`	Determina o devuelve un valor numérico correspondiente a las diferentes características del objeto `TableDef` (archivos adjuntos, exclusividad, etc.).
`ConflictTable`	Devuelve el nombre de la tabla de conflictos. Esta tabla contiene la lista de registros que entran en conflicto durante una sincronización.
`Connect`	Devuelve una cadena de caracteres correspondiente a la información del origen del objeto `TableDef`.
`DateCreated`	Indica la fecha y la hora de creación del objeto `TableDef`.
`KeepLocal`	Devuelve un booleano que indica si la tabla se puede replicar (`False`) o no (`True`).
`LastUpdated`	Indica la fecha y la hora de la última modificación del objeto `TableDef`.
`Name`	Devuelve el nombre de la tabla.
`RecordCount`	Devuelve el número de registros de la tabla. Si la tabla está vacía, la propiedad devuelve 0. Cuando la tabla está relacionada, la propiedad siempre devuelve -1.
`Replicable`	Devuelve un booleano que indica si la tabla se puede replicar. Es la situación opuesta a la propiedad `KeepLocal`.
`ReplicaFilter`	Define el filtro correspondiente a los registros antes de replicarse.
`SourceTableName`	Devuelve el nombre del origen en la base de datos externa cuando el objeto `TableDef` apunta a una tabla relacionada.
`Updatable`	Devuelve un booleano que indica si la estructura de la tabla es modificable.
`ValidationText`	Devuelve el mensaje de error si la condición de validación no se respeta.
`ValidationRule`	Devuelve la condición que se debe respetar para la validación del objeto.

b. Métodos

Método	Descripción
`CreateField`	Permite crear un nuevo campo (`Field`).
`CreateIndex`	Permite crear un nuevo índice (`Index`).
`CreateProperty`	Permite crear una nueva propiedad (`Property`).
`OpenRecordset`	Permite crear un nuevo conjunto de registros (`Recordset`).
`RefreshLink`	Permite actualizar los datos a partir de una tabla relacionada.

c. Colecciones

El objeto `TableDef` contiene las colecciones `Fields`, `Indexes` y `Properties`.

d. Ejemplos

A continuación se muestra un ejemplo de código que va a indicar el nombre y el número de registros de todas las tablas de la base de datos.

```
Dim tbl As TableDef
For Each tbl In CurrentDb.TableDefs
    Debug.Print tbl.Name & " " & tbl.RecordCount
Next tbl
```

El siguiente código permite crear una nueva tabla de argumentos, con un número autoincremental y dos campos de tipo texto.

```
Sub CrearTabla()
Dim Db As DAO.Database
Dim tbl As DAO.TableDef
Dim fld As DAO.Field
Dim Idx As DAO.Index
'Asigna la base actual a la variable Db
Set Db = CurrentDb
'Crea la nueva tabla
    Set tbl = Db.CreateTableDef("Admin_Params")
'Crea el campo ID_Param
Set fld = tbl.CreateField("ID_Param", dbLong)
'Define el campo como un número autoincremental
    fld.Attributes = dbAutoIncrField
'Añade el campo a la tabla
    tbl.Fields.Append fld
'Crea el campo Argumento_Valor y lo añade
    tbl.Fields.Append tbl.CreateField("Param_Valor", _
                dbText, 55)
'Crea el campo Argumento_Descripción y lo añade
    tbl.Fields.Append tbl.CreateField("Param_Descripcion", _
             dbText, 250)
'define la clave primaria en ID_Param
    Set idx = tbl.CreateIndex("PK_ID_Param")
    Idx.Primary = True
    Idx.Fields.Append Idx.CreateField("Id_Param")
'Añade el índice a la tabla
      tbl.Indexes.Append Idx
'Añade la tabla a la base de datos
    Db.TableDefs.Append tbl
'Libera las variables
Db.Close
Set Idx = Nothing
Set fld = Nothing
Set tbl = Nothing
Set Db = Nothing
```

```
End Sub
```

7. El objeto QueryDef

El objeto `QueryDef` representa una consulta guardada.

a. Propiedades

Propiedad	Descripción
`Connect`	Devuelve una cadena de caracteres correspondiente a la información del origen del objeto `QueryDef`.
`DateCreated`	Indica la fecha y la hora de creación del objeto `QueryDef`.
`KeepLocal`	Devuelve un booleano que indica si la tabla se puede (`False`) o no (`True`) replicar. Es la situación opuesta de la propiedad `Replicable`.
`LastUpdated`	Indica la fecha y la hora de la última modificación del objeto `QueryDef`.
`MaxRecords`	Define el número máximo de registros que la consulta debe devolver.
`Name`	Devuelve el nombre de la consulta.
`RecordsAffected`	Indica el número de registros implicados durante la última ejecución de consulta (método `Execute`).
`Replicable`	Devuelve un booleano que indica si la tabla se puede replicar. Es la situación opuesta de la propiedad `KeepLocal`.
`ReturnsRecords`	Devuelve un booleano que indica si la consulta ha devuelto de registros.
`SQL`	Define o devuelve la instrucción SQL que define la consulta.
`Type`	Devuelve un valor numérico que indica el tipo de consulta (selección, adición, eliminación, etc.).
`Updatable`	Devuelve un booleano que indica si la estructura de la consulta es modificable.

b. Métodos

Método	Descripción
`Close`	Permite cerrar la consulta, por lo que los datos se liberan para el resto de usuarios.
`CreateProperty`	Permite crear una nueva propiedad (`Property`).
`Execute`	Permite ejecutar una consulta de acción o una instrucción SQL.
`OpenRecordset`	Permite crear un nuevo conjunto de registros (`Recordset`).

c. Colecciones

El objeto `QueryDef` contiene las colecciones `Fields`, `Parameters` y `Properties`.

d. Ejemplo

El siguiente código permite recorrer cada consulta y visualizar su nombre, su código SQL y el tipo de consulta del que se trata:

```
Sub RecorrerQueryDefs()
    Dim Qdf As DAO.QueryDef
    For Each Qdf In CurrentDb.QueryDefs
           Debug.Print Qdf.Name & " " & Qdf.SQL & " => " & Qdf.Type
    Next
End Sub
```

8. El objeto Recordset

El objeto Recordset representa un conjunto de registros de una tabla o consulta. Hay cinco tipos de Recordset:

Tipo	Constante de declaración	Descripción	Ventaja
Tabla	dbOpenTabla	Representa una tabla.	Uso posible de los índices.
DynaSet (hoja de respuesta dinámica)	dbOpenDynaset	Representa el resultado de una consulta cuyos registros se pueden actualizar.	Trabaja sobre los datos de varias tablas.
Dynamic (hoja de respuesta dinámica)	dbOpenDynamic	Representa el resultado de una consulta cuyos registros se pueden modificar. Funciona para los espacios de trabajo ODBC Direct.	Trabaja sobre los datos en modo ODBC Direct.
Forward Only (en antes únicamente)	dbOpenForwardOnly	Representa el resultado de una consulta en modo solo lectura, donde solo están permitidos los movimientos (MoveNext y Move) hacia delante.	Rapidez de la operación.
Snapshot (instantaneo)	dbOpenSnapshot	Representa el resultado de una consulta en modo solo lectura.	Rápido con relación a Dynaset.

a. Propiedades

Según el tipo de Recordset (T para dbOpenTable, D para dbOpenDynaset, d para dbOpenDynamic, F para dbOpenForwardOnly y S para dbOpenSnapShot), las propiedades son las siguientes:

Propiedad	Descripción	Tipo
AbsolutePosition	Determina la posición relativa de un registro en un Recordset.	DdS
BOF	Devuelve un booleano que indica si el registro está al inicio del Recordset.	DdSTF
Bookmark	Devuelve o define un marcador que identifica un registro de manera única.	DdST
Bookmarkable	Devuelve un booleano que indica si el Recordset soporta marcadores.	DdST
CacheStart	Define o devuelve el marcador del primer registro que se debe situar en memoria caché.	Dd
DateCreated	Indica la fecha y la hora de creación del objeto TableDef subyacente.	T

EditMode	Indica el estado de modificación del registro actual.	DdT
EOF	Devuelve un booleano que indica si el registro está al final del Recordset.	DdSTF
Filter	Define o devuelve el filtro que se debe aplicar al conjunto de registros.	DdSF
Index	Define o devuelve el índice que se va a utilizar.	T
LockEdits	Define la condición de bloqueo.	DdST
LastModified	Devuelve un marcador que indica el último registro modificado.	DdT
LastUpdated	Indica la fecha y la hora de la última modificación del objeto TableDef subyacente.	T
Name	Devuelve el nombre del Recordset.	DdSTF
NoMatch	Indica si se ha encontrado el registro buscado.	DdST
PercentPosition	Devuelve la posición del registro actual, en forma de porcentaje del número total de registros.	DdST
RecordsCount	Indica el número de registros solicitados durante la ejecución de la consulta o en la tabla.	DdSTF
Restartable	Devuelve un booleano que indica si el Recordset soporta el método Requery que ejecuta de nuevo la consulta subyacente.	DdSTF
Sort	Define el sentido de la ordenación.	DdS
Transacciones	Devuelve un booleano que indica si el Recordset soporta las transacciones.	DdSTF
Type	Devuelve un valor numérico que indica el tipo de Recordset.	DdSTF
Updatable	Devuelve un booleano que indica si se autorizan las actualizaciones.	DdTSF
ValidationText	Devuelve un mensaje de error si la condición de validación no se respeta.	DdSTF
ValidationRule	Devuelve la condición que se debe respetar para validar el objeto.	DdSTF

b. Métodos

Método	Descripción	Tipo
AddNew	Permite crear un nuevo registro.	DdT
CancelUpdate	Anula las actualizaciones en espera.	DdT
Clone	Crea una copia del Recordset.	DdST
Close	Cierra el Recordset.	DdSTF
CopyQueryDef	Crea un objeto QueryDef asociado al Recordset.	DdSF
Delete	Elimina el registro actual.	DdT
Edit	Pone el registro actual en modo Edición.	DdT
FindFirst	Encuentra el primer registro que se corresponde con el criterio y lo convierte en registro actual.	DdS
FindLast	Encuentra el último registro que se corresponde con el criterio y lo convierte en registro actual.	DdS
FindNext	Encuentra el registro siguiente que se corresponde con el criterio y lo convierte en registro actual.	DdS
FindPrevious	Encuentra el registro anterior que se corresponde con el criterio y lo	DdS

	convierte en registro actual.	
GetRows	Extrae varias líneas de un Recordset, y las copia en una tabla.	DdSTF
MoveFirst	Se posiciona en el primer registro y lo convierte en registro actual.	DdST
MoveLast	Se posiciona en el último registro y lo convierte en registro actual.	DdST
MoveNext	Se posiciona en el registro siguiente y lo convierte en registro actual.	DdST
MovePrevious	Se posiciona en el registro anterior y lo convierte en registro actual.	DdST
Move	Se posiciona en un registro concreto y lo convierte en registro actual.	DdST
OpenRecordset	Crea un nuevo conjunto de registros (Recordset).	DdST
Requery	Actualiza los registros de un Recordset ejecutando la consulta subyacente.	DdSF
Seek	Utilizando un índice seleccionado, encuentra el primer registro que se corresponde con los criterios y lo convierte en registro actual.	T
Update	Actualiza el registro.	DdT

c. Abrir

Para abrir un nuevo Recordset, la sintaxis general es la siguiente:

```
Dim RS As DAO.Recordset
Set RS = CurrentDB.OpenRecordset(SQL_o_Nombre, [Tipo_de_Recordset],
[Opción_deApertura], [Bloqueo_RecordSet])
```

La variable SQL_o_Nombre puede ser una instrucción SQL que opera sobre un conjunto de registros, un nombre de tabla o un nombre de consulta ya existente en la base de datos. La variable Tipo_de_Recordset representa el tipo de conjunto de registros, ver la sección El objeto Recordset - Propiedades. La variable Opción_deApertura permite definir el modo de apertura del conjunto de registros. La lista de los valores que puede tomar esta variable está en el anexo de este libro. La variable Bloqueo_Recordset permite determinar el modo de administración de los accesos concurrentes (simultáneos) utilizado por el motor de base de datos. Los posibles valores también están en el anexo de este libro.
Por ejemplo, si se desea abrir un conjunto de registros a partir de la tabla ENI_ANIMAL_ANI,
en modo instantáneo, la sintaxis en VBA será la siguiente:

```
Dim RS As DAO.Recordset
Set RS = CurrentDb.OpenRecordset("ENI_ANIMAL_ANI", dbOpenSnapShot)
```

Otro ejemplo con la apertura de los registros de la tabla ENI_VACUNA_VAC en modo Table, en modo solo lectura.

```
Dim RS As DAO.Recordset
Set RS = CurrentDb.OpenRecorset("ENI_VACUNA_VAC", dbOpenTable,
dbReadOnly)
```

d. Examinar

Una vez que el conjunto de registros está abierto, es posible moverse entre ellos de varias maneras. Por ejemplo, para recorrer todos los registros, podemos pasar por los siguientes bucles:

```
'El conjunto de registros RS se ha declardo y abierto anteriormente
RS.MoveFirst 'Nos posicionamos en el primer registro
Do Until RS.EOF 'se jecuta el bucle hasta llegar al último registro
    'Instrucciones
    RS.MoveNext 'Se pasa al registro siguiente
Loop
```

También es posible recorrer los registros partiendo desde el final hacia el inicio:

```
'El conjunto de registros RS se ha declarado y abierto anteriormente
RS.MoveLast 'Nos posicionamos en el primer registro
Do Until RS.BOF 'se ejecuta el bucle hasta llegar al último registro
    'Instrucciones
    RS.MovePrevious 'Se pasa al registro siguiente
Loop
```

Observe que las nociones de primero y último están directamente relacionadas con el sentido de la ordenación de los datos durante la generación del conjunto de registros.

Son posibles los movimientos registro a registro, pero también podemos movernos directamente a un registro concreto, por ejemplo con Move:

```
'el conjunto de registros RS se ha declarado y abierto
anteriormente RS.Move 3 'nos posicionamos en el tercer registro
del conjunto de registros
```

Además, podemos llegar a algunos registros filtrando e indicando los criterios de correspondencia.

```
' se ha declarado y abierto anteriormente el conjunto de registros
RS RS.FindFirst "ADO_NOMBRE Like 'AND*’"
```

```
' el conjunto de registros se va a posicionar en el
primer registro cuyo campo ADO_NOMBRE comience por 'ÁND’
```

e. Actualizar

Una vez que se modifica el registro como queremos, es posible validar las modificaciones usando las siguientes instrucciones:

```
' Se ha declarado y abierto anteriormente el conjunto de registros
RS RS.Edit
RS("ADO_NOMBRE").value = "ÁNGEL"
```

```
'otra instrucción que modifica el valor de los campos
RS.Update
```

f. Eliminar

Es posible eliminar un registro con la siguiente instrucción:

```
'El conjunto de registros RS se ha declarado y abierto
anteriormente RS.Delete
```

Atención, esta instrucción elimina directamente el registro, sin mensaje de alerta previo. Si intenta acceder a un registro mientras se está eliminando, recibirá un error 3167 "El registro se ha eliminado". En el caso de un `Recordset` abierto en modo `Dynaset`, recibirá un error 3021, "No hay ningún registro".

g. Ejemplo

A continuación se muestra un ejemplo de código que sucesivamente añade una formación, modifica su lugar y después la elimina.

```
Sub Adicion_Modificacion_Eliminación ()
    'se ejecuta en modo paso a paso
    Dim RS As DAO.Recordset
    Set RS = CurrentDb.OpenRecordset("SELECT * FROM ENI_VACUNA_VAC")
    ' Adición
    With RS
           .AddNew
           .Fields("VAC_VACUNA").Value = "Rabia"
           .Fields("VAC_ANT_ID").Value = 1
           .Update
           .Close
    End With
    'Modificación
    Set RS = CurrentDb.OpenRecordset("SELECT * FROM ENI_VACUNA_VAC
WHERE VAC_VACUNA='Rabia'")
    RS.Edit
    RS.Fields("VAC_VACUNA").Value = "Rabia del perro"
    RS.Update
    RS.Close
    'Eliminación
    Set RS = CurrentDb.OpenRecordset("SELECT * FROM ENI_VACUNA_VAC
WHERE VAC_VACUNA='Rabia del perro'")
    RS.Delete
    RS.Close
End Sub
```

9. La colección Relations

El objeto `Relation` representa una relación entre campos de dos tablas `TableDefs`.

a. Propiedades

Propiedad	Descripción
Attributes	Devuelve algunas características de la relación (integridad referencial, relación uno a varios, etc.).
Foreigntable	Devuelve o define el nombre de la tabla externa.
Name	Devuelve el nombre de la relación.
PartialReplica	Indica si la relación se debe tener en cuenta durante la realización de una réplica parcial a partir de una réplica completa.

b. Métodos

Método	Descripción
CreateField	Permite crear un nuevo campo Field.

c. Colecciones

El objeto Relation contiene las colecciones Fields y Properties.

10. La colección Containers

El objeto Container representa el conjunto de información de los objetos de una base de datos, así como la información de la base de datos en sí misma.

a. Propiedades

Propiedad	Descripción
AllPermissions	Devuelve todos los permisos que se aplican a la propiedad UserName del objeto Container actual.
Inherit	Devuelve un booleano que indica si los objetos Documents heredan de un valor para la propiedad Permissions.
Name	Devuelve el nombre del Container.
Owner	Devuelve el nombre del propietario del Container.
Permissions	Define los permisos de un usuario o de un grupo de usuarios, identificados por la propiedad UserName del objeto Container.
UserName	Define el nombre del usuario.

b. Documento

Cada contenido de un objeto Container es una colección de Document.

Propiedades

Propiedad	Descripción
AllPermissions	Devuelve todos los permisos que se aplican a la propiedad UserName del

	objeto Document actual.
Container	Devuelve un objeto Container al que pertenece el objeto Document.
DateCreated	Devuelve la fecha y la hora de creación de la tabla subyacente.
LastUpdated	Devuelve la fecha y la hora de la última modificación de estructura de la tabla subyacente.
Name	Devuelve el nombre del Document.
Owner	Devuelve el nombre del propietario del Document.
Permissions	Define los permisos de un usuario o de un grupo de usuarios, identificados por la propiedad UserName del objeto Document.
UserName	Define el nombre del usuario.

Colección

El objeto Document contiene la colección Properties.

11. Las colecciones Groups y Users

a. Group

Un objeto Group representa un grupo de usuarios con los mismos permisos de acceso a la aplicación. La colección Groups agrupa el conjunto de grupos de usuarios.

Propiedades

Propiedad	Descripción
Name	Devuelve el nombre del grupo.
PID	Devuelve el identificador único de cuenta de grupo.

Métodos

Método	Descripción
CreateUser	Permite crear un nuevo objeto User.

Colecciones

El objeto Group contiene las colecciones Properties y Users.

b. User

Un objeto User representa una cuenta de usuario con los permisos de acceso a la aplicación. La colección Users agrupa el conjunto de usuarios.

Propiedades

Propiedad	Descripción
Name	Devuelve o define el nombre de la cuenta de usuario.
PID	Devuelve el identificador único de la cuenta de usuario.
PassWord	Devuelve la contraseña de la cuenta de usuario.

Métodos

Método	Descripción
CreateGroup	Permite crear un nuevo objeto Group.
NewpassWord	Permite definir una nueva contraseña para la cuenta de usuario.

Colección

El objeto User contiene las colecciones Groups y Properties.

ADO

1. Introducción

ADO (*ActiveX Data Objects*) es una librería de VBA que agrupa un conjunto de objetos que permite acceder a una base de datos. Se utilizan estos objetos para acceder a las estructuras de datos (tablas, consultas), así como para manipular los registros. El modelo tiene en cuenta el entorno de base de datos OLE DB. Esta librería tiene mejor rendimiento que DAO, y permite fundamentalmente realizar aplicaciones cliente/servidor, así como aplicaciones web.

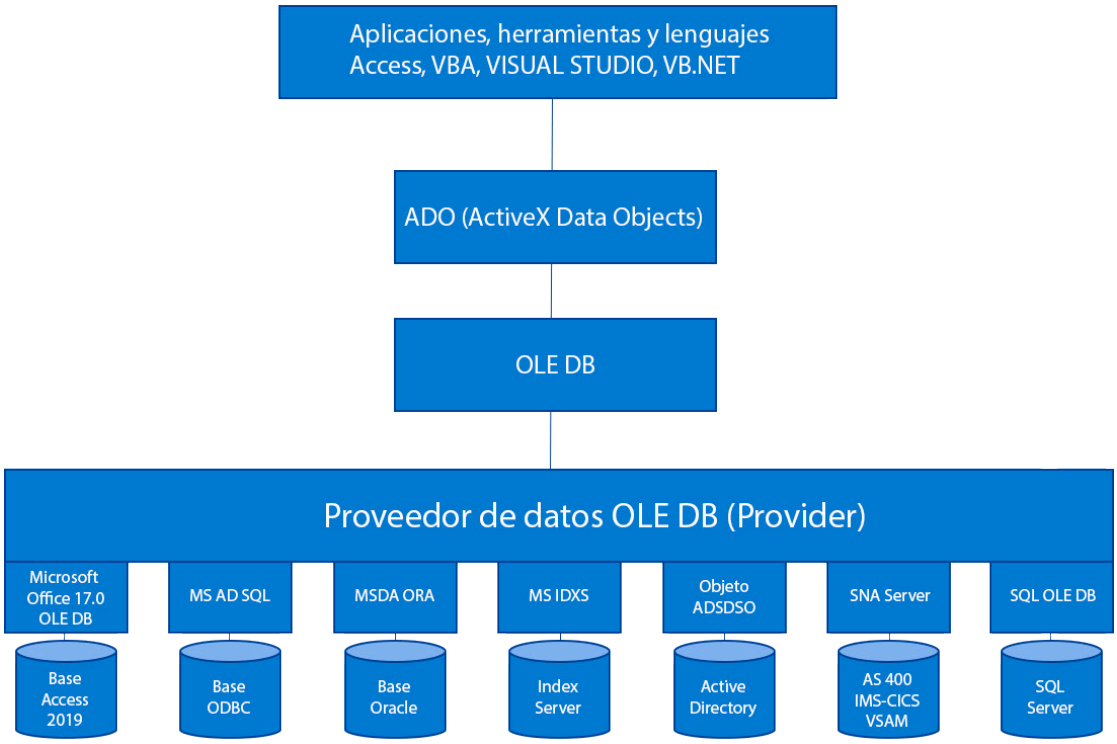
Para utilizar la librería de objetos ADO, es necesario añadir la referencia **Microsoft ActiveX Data Objects 6.0 Library** a la lista de referencias. Puede realizar esta operación seleccionando la pestaña **Referencias** del menú **Herramientas**. Si no encuentra directamente esta referencia en la lista de referencias disponibles, puede hacer clic en el botón **Examinar** y buscar su ubicación. El archivo msado16.dll generalmente está en la siguiente carpeta: C:\Program Files (x86)\Common Files\System\ado

Adicionalmente, el archivo de ayuda ADO210.chm proporcionado con los ejemplos de este libro, le será de utilidad para encontrar información detallada del uso del modelo de objetos DAO.

2. ADO y OLE DB

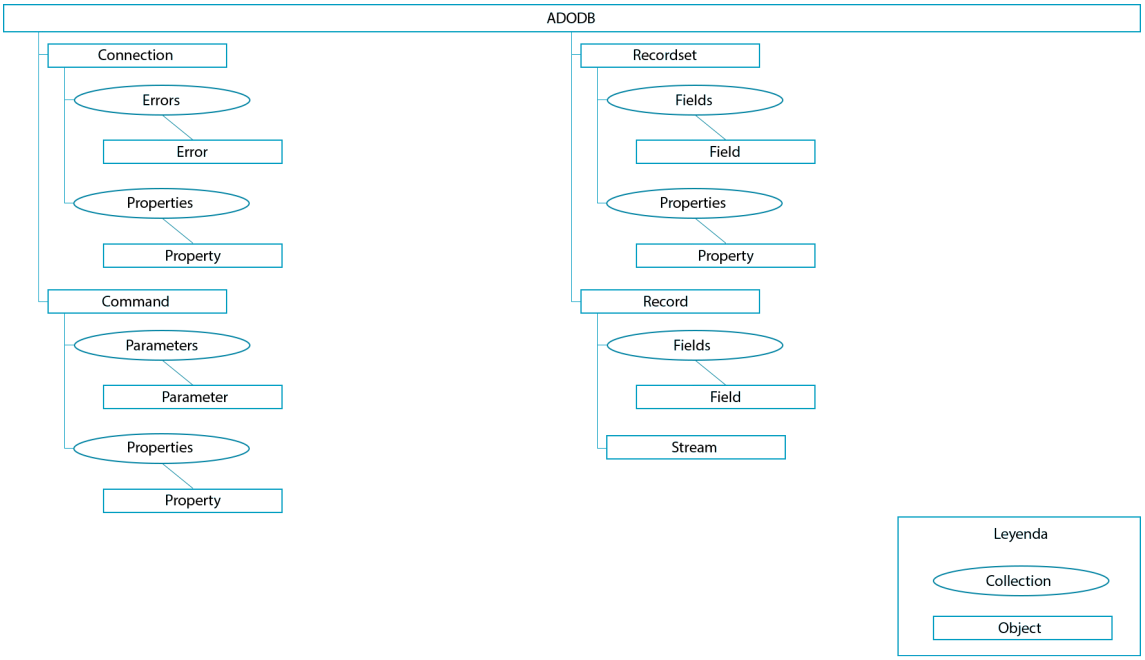
La tecnología OLE DB permite un acceso uniforme a los diferentes datos almacenados en diferentes fuentes de información (base de datos relacionales, mensajería, sistema de archivos, etc.).

Más abajo está la arquitectura tecnológica que relaciona ADO y OLE DB.



3. Modelo de datos ADO

El siguiente esquema representa las diferentes colecciones de objetos contenidos en la jerarquía ADO.



El proceso de un acceso a los datos por medio de la librería ADO es el siguiente:

1. Crear una conexión `Connection` a la fuente que contiene los datos.
2. Escribir el comando SQL `Command`.
3. Enviar el comando a través de la conexión `Connection`. Dependiendo de si el comando devuelve o no datos, llegado el caso estos se almacenan en un objeto `Recordset`.
4. Moverse por el conjunto de registros, añadir un nuevo registro, actualizar y eliminar.
5. Validar y anular la transacción si se inició durante la apertura de la conexión.

4. Descripción de los objetos ADO

Tipo de objeto	Descripción
`Connection`	Representa una conexión a una fuente de datos.
`Command`	Representa una consulta o una instrucción que se ejecutará en una fuente de datos.
`Recordset`	Representa un conjunto de registros que proviene de una consulta o instrucción.
`Record`	Representa un registro único. Su modo de funcionamiento es muy cercano al del `Recordset`, siendo sin embargo su uso pertinente y más eficaz en caso de que el comando `Command` solo devuelva un único registro, en lugar de varios.
`Stream`	Representa un flujo de información legible, en el que se podrá leer, escribir o gestionar texto o datos binarios.

5. Descripción de las colecciones ADO

Colección	Conjunto de objetos
`Parameters`	`Parameter`, argumento del objeto `Command`.
`Properties`	`Property`, una característica de los objetos `Command`, `Connection` y `Recordset`.
`Errors`	`Error`, error durante el intento de conexión a una fuente de datos.

Fields	Field, campo de un conjunto de registros Recordset o de un único registro Record.

El conjunto de estas colecciones dispone de las siguientes propiedades:

Propiedad	Descripción
Count	Determina el número de elementos presentes en la colección.
Item	Determina un elemento específico de una colección, definida por su nombre o su identificador dentro de la colección.

6. Conectarse - el objeto Connection

Un objeto Connection representa una conexión a una fuente de datos.

a. Propiedades

Propiedad	Descripción
Attributes	Devuelve o define una o varias características del objeto Connection.
CommandTimeout	Devuelve o define el lapso de tiempo en segundos más allá del cual el intento de ejecución de un comando se considerará como erróneo y mostrará un mensaje de error. Por defecto, este lapso de tiempo es de 30 segundos.
ConnectionString	Devuelve o define la información utilizada para establecer una conexión con la fuente de datos.
ConnectionTimeout	Devuelve o define el lapso de tiempo en segundos más allá del cual el intento de conexión se considerará erróneo y mostrará un mensaje de error. Por defecto, este lapso de tiempo es de 15 segundos.
CursorLocation	Devuelve o define el tipo de cursor utilizado por defecto (cursor cliente o cursor servidor).
DefaultDatabase	Devuelve o define la base de datos por defecto del objeto Connection.
IsolationLevel	Devuelve o define el nivel de aislamiento del objeto Connection. La lista de las constantes está en el anexo de este libro.
Mode	Devuelve o define el modo de acceso autorizado para la conexión (modo solo lectura, lectura/escritura, etc.). La lista de las constantes está en el anexo de este libro.
Provider	Devuelve o define el nombre del proveedor OLE DB del objeto Connection.
State	Devuelve un valor numérico que indica el estado del objeto Connection si la conexión está abierta o cerrada.
Version	Devuelve el número de versión de la librería ADO.

b. Métodos

Método	Descripción
BeginTrans	Permite iniciar una nueva transacción.
Cancel	Permite anular la ejecución de una consulta ya ejecutada en modo asíncrono (ejecución a partir de los métodos Execute y Open).
Close	Permite cerrar una conexión abierta y todos los objetos relacionados.

CommitTrans	Permite validar la transacción, guardar las modificaciones añadidas y parar la transacción actual.
Execute	Permite ejecutar la instrucción SQL, el procedimiento almacenado o la instrucción del proveedor OLE DB.
Open	Permite abrir una conexión a una fuente de datos y seguidamente ejecutar comandos Command.
OpenSchema	Permite devolver la información del enlace con el esquema de la base de datos, proporcionado por el proveedor OLE DB.
RollbackTrans	Permite anular la transacción, anular las modificaciones añadidas y parar la transacción actual.

c. Ejemplos

El siguiente código permite conectarse a una fuente de datos de tipo MySQL.

```
Sub Abrir_Conexion()
'Declaración de las variables
Dim cnt As ADODB.Connection
'Instanciación de las variables
Set cnt = New ADODB. Connection
'Conexión a la base de datos
cnt.ConnectionString =
"Server=DireccionServidor;Port=3456;DataBase=BaseMySQL;UID=MiLogin;
PWD=Contrasenia;"
cnt.Open 'apertura de la conexión
End Sub
```

7. Realizar una consulta SQL - el objeto Command

Cuando se ha establecido la conexión con la fuente de datos, es posible consultar este origen de datos utilizando consultas. Se usa el objeto Command para hacerlo. El objeto Command representa un comando que se debe ejecutar en la fuente de datos.

a. Propiedades

Propiedad	Descripción
ActiveConnection	Devuelve o define la conexión a la fuente de datos (Connection).
CommandText	Devuelve o define la instrucción SQL o el comando que se debe enviar al proveedor OLE DB.
CommandTimeOut	Devuelve o define el lapso de tiempo en segundos, más allá del cual el intento de ejecución de un comando se considerará como erróneo y mostrará un mensaje de error. Por defecto, este lapso de tiempo es de 30 segundos.
CommandType	Devuelve o define el tipo del objeto Command.
Name	Devuelve o define el nombre del objeto Command.
Prepared	Devuelve o define un booleano que indica si el comando debe tener una versión compilada guardada en el servidor antes de su ejecución.
State	Devuelve un valor numérico que indica el estado del objeto Command.

b. Métodos

Método	Descripción
Cancel	Permite anular la ejecución de una consulta ya lanzada en modo asíncrono (ejecución a partir de los métodos Execute).
CreateParameter	Permite crear un nuevo objeto Parameter e indica sus propiedades.
Execute	Permite ejecutar la instrucción SQL, el procedimiento almacenado o la instrucción del proveedor OLE DB almacenada en la propiedad CommandText.

c. Ejemplos

El siguiente código permite conectarse a una fuente de datos con el objeto cnt ya declarado y abierto, y ejecutar una consulta:

```
Sub EjecutarComando()
   'apertura de la conexión
   Dim cnt As ADODB.Connection
   Set cnt = New ADODB.Connection
   Set cnt = CurrentProject.Connection
   'inútil porque la conexión ya está abierta (base de datos actual)
   'cnt.Open
   'Declaración de las variables
   Dim cmd As ADODB.Command
   'Instanciación de las variables
   Set cmd = New ADODB.Command
   'Conexión a la base de datos
   cmd.ActiveConnection = cnt
   'Preparación del objeto Command
   cmd.CommandText = "SELECT * FROM ENI_VACUNA_VAC WHERE VAC_VACUNA
   LIKE 'Rabia%'"
   'Ejecución de la consulta
   Set rst = cmd.Execute
End Sub
```

8. Recuperar el resultado de una consulta - el objeto Recordset

Para recuperar el resultado del comando SQL ejecutado, es posible almacenar el conjunto de registros en un objeto Recordset.

a. Propiedades

Propiedad	Descripción
AbsolutePage	Devuelve la página del registro actual.
AbsolutePosition	Devuelve la posición numérica del registro actual, dentro del objeto Recordset.
ActiveCommand	Devuelve el objeto Command que ha permitido alimentar el Recordset.
ActiveConnection	Devuelve el objeto Connection al que pertenece el objeto Recordset.

BOF	Devuelve un booleano que indica si el registro actual está en primera posición en el objeto Recordset.
Bookmark	Devuelve o define un marcador que identifica un registro de manera única.
Cachesize	Devuelve un valor numérico que indica el número de registros contenidos en el objeto Recordset, almacenado en memoria caché.
CursorLocation	Devuelve o define el tipo de cursor por defecto utilizado con el objeto Connection (cursor servidor o cursor cliente).
CursorType	Devuelve o define el tipo de cursor utilizado con el objeto Recordset.
DataMember	Devuelve el nombre del miembro de datos objeto de la información DataSource de origen.
DataSource	Devuelve un objeto que se considera como Recordset y puede ser la fuente de datos.
EditMode	Devuelve el estado de edición del registro actual, con una constante EditModeEnum, cuya lista está disponible en el anexo de este libro.
EOF	Devuelve un booleano que indica si el registro actual está después de la última posición en el objeto Recordset.
Filter	Devuelve o define el filtro aplicado al objeto Recordset.
Index	Devuelve el nombre del índice actual en el Recordset.
LockType	Devuelve le tipo de bloqueo utilizado en los registros durante las modificaciones.
MarshalOptions	Devuelve los registros que se deben enviar al servidor.
MaxRecords	Devuelve el número máximo de registros a un objeto Recordset en una consulta. El valor por defecto es 0 (no se fija ningún límite).
PageCount	Devuelve el número de páginas de datos contenidos en el objeto Recordset.
PageSize	Devuelve el número de registros en una página de datos.
Properties	Devuelve la colección de Property del objeto Recordset.
RecordCount	Devuelve el número de registros en el objeto Recordset.
Sort	Devuelve o define el nombre o los nombres de campos que sirven de ordenación al Recordset, con el orden para cada uno de ellos.
Source	Devuelve la fuente de datos del objeto Recordset.
State	Devuelve el estado del Recordset (abierto, cerrado u operación asíncrona).
Status	Devuelve el estado del registro respecto de la actualización por lotes u otras operaciones globales.
StayInSync	Devuelve un booleano que indica si la línea padre, en el registro jerárquico, se debe actualizar en cascada cuando se actualizan registros subyacentes.

b. Métodos

Método	Descripción
AddNew	Permite crear un nuevo registro dentro del Recordset, haciendo posible una actualización.
Cancel	Permite anular una operación Open asíncrona actualmente en ejecución.
CancelBatch	Permite anular una actualización por lotes que está en estado de espera.
CancelUpdate	Permite anular todas las actualizaciones añadidas a registro actual, antes de

	lanzar la ejecución de un Update.
Clone	Permite crear un nuevo Recordset duplicando un Recordset existente.
Close	Permite cerrar el Recordset y todos los objetos que puedan depender de él.
CompareBookmarks	Permite comparar dos marcadores y devolver un valor según sus posiciones relativas.
Delete	Permite eliminar un registro actual o un grupo de registros.
Find	Permite encontrar el primer registro correspondiente al criterio único y convertirlo en registro actual.
GetRows	Permite extraer uno o varios registros de un Recordset y almacenarlos en una tabla.
GetString	Permite devolver un Recordset en forma de cadena de caracteres.
Move	Permite posicionarse en un registro concreto y convertirlo en registro actual.
MoveFirst	Permite posicionarse en el primer registro del Recordset y convertirlo en registro actual.
MoveLast	Permite posicionarse en el último registro del Recordset y convertirlo en registro actual.
MoveNext	Permite posicionarse en el registro que sigue al registro actual y convertirlo en registro actual.
MovePrevious	Permite posicionarse en el registro anterior del registro actual y convertirlo en registro actual.
NextRecordset	En caso de que un comando tenga varias consultas, permite devolver el conjunto de registros siguiente.
Open	Permite abrir un cursor que representa los registros resultado de la consulta o de una tabla origen.
Requery	Permite volver a ejecutar la consulta origen y actualizar los registros del Recordset.
Resync	Permite sincronizar los datos del Recordset a partir de la base de datos origen.
Save	Permite hacer una copia de seguridad del Recordset en un archivo.
Seek	Permite encontrar el primer registro correspondiente a los criterios y convertirlo en registro actual.
Supports	Permite devolver un booleano que indica las posibilidades que ofrece el Recordset (añadir, uso de bookmark, etc.).
Update	Permite guardar las actualizaciones añadidas al registro actual.
UpdateBatch	Permite guardar las actualizaciones por lotes de registros.

c. Ejemplos

El siguiente código permite conectarse a una fuente de datos a través del objeto cnt ya declarado y abierto, y ejecutar una consulta después de actualizar el primer registro que tenga un campo ADO_NOMBRE con Doe, para asignarle el nombre "indeterminado".

```
Sub Prueba_Recordset()
'apertura de la conexión cnt
Dim cnt As ADODB.Connection
Set cnt = New ADODB.Connection
Set cnt = CurrentProject.Connection
```

```
'Declaración de las variables
Dim RS As New ADODB.Recordset
'Instanciación de las variables
RS.Open "SELECT * FROM ENI_ADOPCION_ADO", cnt, adOpenKeyset, adLockOptimistic
With RS
     .Find "ADO_NOMBRE='Doe'", , adSearchForward
     If Not .EOF Then
            .Fields("ADO_APELLIDO").Value = "indeterminada"
     End If
     .Update
     .Close
End With

End Sub
```

9. Los campos - objeto Field

El objeto `Field` se corresponde con un campo, perteneciente a una tabla (`TableDef`), una consulta (`QueryDef`), un conjunto de registros (`Recordset`) o un índice (`Index`). Pertenecen a una colección de campos: `Fields`.

a. Propiedades

Propiedad	Descripción
`ActualSize`	Devuelve la longitud del valor de `Field`.
`Attributes`	Devuelve y define una o varias de las características de `Field`.
`DefinedSize`	Define el tamaño de `Field`.
`Name`	Devuelve el nombre del campo.
`NumericScale`	Define la escala de los valores numéricos de `Field`.
`OriginalValue`	Devuelve el valor de origen de `Field` (antes de la modificación).
`Precision`	Devuelve el grado de precisión para los valores numéricos.
`Properties`	Colección de objetos `Property` de `Field`.
`Type`	Devuelve el tipo de datos de `Field`.
`UnderlyingValue`	Devuelve el valor de `Field` actual en la base de datos.
`Value`	Devuelve o define el valor de `Field`.

b. Métodos

Método	Descripción
`AppendChunk`	Permite añadir datos a un `Field` de tipo binario.
`GetChunk`	Permite devolver todo o parte del contenido de un `Field` de tipo binario.

10. Los argumentos - objeto Parameter

Los argumentos de consulta (`QueryDef`) se manipulan a través de los objetos `Parameter`. El conjunto de

argumentos de una consulta pertenecen a la colección `Parameters` de la consulta.

a. Propiedades

Propiedad	Descripción
`Attributes`	Devuelve y define una o varias de las características de `Parameter`.
`Direction`	Devuelve y define la dirección de `Parameter`, si este es de entrada, salida, ambos tipos o si corresponde a un valor de retorno de un procedimiento almacenado.
`Name`	Devuelve el nombre de `Parameter`.
`NumericScale`	Define la escala de los valores numéricos de `Parameter`.
`Precision`	Devuelve el grado de precisión para los valores numéricos.
`Properties`	Colección de objetos `Property` de `Parameter`.
`Size`	Devuelve o define el tamaño máximo de `Parameter`, en bytes o caracteres.
`Type`	Devuelve el tipo de datos de `Parameter`.
`Value`	Devuelve o define el valor de `Parameter`.

b. Métodos

Método	Descripción
`AppendChunk`	Permite añadir datos a un `Parameter` de tipo binario.

11. Los errores - objeto Error

En caso de error durante la apertura de una conexión, la ejecución de una consulta o la operación de guardado, la librería ADO gestiona los errores a través del objeto `Error`. Si pueden aparecer varios errores, cada uno pertenecerá a la colección `Errors`.

Propiedad	Descripción
`Description`	Devuelve la cadena de caracteres que describe el error.
`HelpContext`	Devuelve el `ContextID` del archivo de ayuda asociado al error.
`HelpFile`	Devuelve el nombre del archivo de ayuda asociado al error.
`NativeError`	Devuelve el código de error del proveedor asociado al error.
`Number`	Devuelve el número de identificación del error.
`Source`	Devuelve el nombre del objeto o de la aplicación que está en el origen del error.
`SQLState`	Devuelve una cadena de cinco caracteres conforme a la norma ANSI SQL, que devuelve el proveedor OLE DB.

El lenguaje SQL

El lenguaje SQL (*Structured Query Language*) es el lenguaje utilizado en Access para extraer, actualizar o eliminar datos que pertenecen a diferentes tablas de la base de datos. El objetivo de las siguientes secciones es explicar las diferentes estructuras y sintaxis que se pueden utilizar para este fin. Aunque se permite usar un amplio abanico de las instrucciones normalizadas SQL, Access y por extensión VBA no respetan la integridad de las funciones llamadas nativas de SQL.

El comando SELECT

La sintaxis general de una consulta SQL es la siguiente:

```
SELECT [DISTINCT o ALL] <* o lista de los Campos>
FROM <Nombre de las Tablas>
[WHERE <Predicados>]
[GROUP BY orden de los grupos]
[HAVING condición]
[ORDER BY] <lista de los Campos>
```

La palabra clave `DISTINCT` permite visualizar solo resultados únicos, al contrario que los resultados duplicados que se mostrarán con la palabra clave `ALL`.

Cuando se desea extraer datos de las tablas, es necesario simplemente listar los campos que se han de extraer.

```
SELECT ANI_NOMBRE, ANI_ESTERILIZADO, ANI_FECHA_NACIMIENTO
FROM ENI_ANIMAL_ANI
```

Los campos `ANI_NOMBRE`, `ANI_ESTERILIZADO` y `ANI_FECHA_NACIMIENTO`, de la tabla `ENI_ANIMAL_ANI`, se mostrarán en el resultado. Cada campo seleccionado se separa del resto por una coma.

El símbolo * permite seleccionar todos los campos disponibles en las tablas que aparecen en la cláusula `FROM`.

El origen FROM

1. Sintaxis general

Las diferentes tablas disponibles en la base de datos de Access pueden servir de fuente de datos. Por lo tanto, cada tabla puede aparecer en la cláusula FROM, separada del resto de las tablas por una coma.

```
SELECT ANI_NOMBRE, ADO_NOMBRE, ADO_CIUDAD
FROM ENI_ANIMAL_ANI, ENI_ADOPTANTE_ADO
```

2. Los joins

Es posible indicar en la consulta si se deben tener en cuenta algunas correspondencias entre las diferentes tablas. Hay tres tipos de correspondencias SQL en Access:

```
SELECT *
FROM Tabla_1 [INNER o LEFT o RIGHT] JOIN Tabla_2
ON <condiciones del join >
```

a. Join interno INNER JOIN

El join INNER JOIN permite tener en cuenta solo los registros para los que existe una correspondencia exacta entre las tablas.

```
SELECT ANI_NOMBRE, ADO_NOMBRE, ADO_CIUDAD
FROM ENI_ANIMAL_ANI INNER JOIN ENI_ADOPTANTE_ADO
ON ENI_ANIMAL_ANI.ID = ENI_ADOPTANTE_ADO.ID
```

b. Join externo LEFT JOIN

El join izquierdo externo LEFT JOIN permite visualizar todos los registros contenidos en la tabla de la izquierda (más abajo ENI_ANIMAL_ANI), incluso si no hay correspondencia en la tabla de la derecha (más abajo ENI_ADOPCION_ADO). De esta manera, se muestran todos los nombres de los animales, incluso si no han sido adoptados por un adoptante (caso de los animales no esterilizados o todavía lactantes, por ejemplo).

```
SELECT ANI_NOMBRE, ADO_NOMBRE, ADO_CIUDAD
FROM ENI_ANIMAL_ANI LEFT JOIN ENI_ADOPCION_ADO
ON ENI_ANIMAL_ANI.ID = ENI_ADOPTANTE_ADO.ID
```

c. Join externo RIGHT JOIN

El join derecho externo RIGHT JOIN permite visualizar todos los registros de la tabla de la derecha (más abajo ENI_ADOPTANTE_ADO), incluso si no hay correspondencia en la tabla de la izquierda (más abajo ENI_ANIMAL_ANI). De esta manera se muestran todos los adoptantes, incluso si todavía no han encontrado lo

que buscan.

```
SELECT ANI_NOMBRE, ADO_NOMBRE, ADO_CIUDAD
FROM ENI_ANIMAL_ANI RIGHT JOIN ENI_ADOPTANTE_ADO
ON ENI_ANIMAL_ANI.ID = ENI_ADOPTANTE_ADO.ID
```

d. Las condiciones de los joins

Las condiciones de los joins se expresan en forma de igualdad o desigualdad entre los campos que utilizamos para hacer la correspondencia.

Los posibles operadores en estas condiciones son los siguientes: =, >, <, >=, <=, <>. La condición de igualdad (=) es la más frecuente.

La cláusula WHERE

En una consulta de extracción de datos, es posible aplicar algunos criterios de valores para filtrar los registros, según el valor de sus campos. Estos criterios se expresan en la cláusula `WHERE` de la consulta SQL.

```
SELECT ANI_NOMBRE, ANI_SEXO
FROM ENI_ANIMAL_ANI
WHERE ANI_DESTETADO = TRUE
```

Esta consulta permite encontrar los nombres y el sexo de los animales para aquellos ya destetados.

1. Los diferentes criterios existentes

Es posible filtrar según varios criterios: igualdad (=), diferencia (<, >, <=, >=, <>), nulidad (`Is Null`), correspondencia de texto (`LIKE`), pertenencia a un intervalo (`BETWEEN`) o lista (`IN`), etc. Los diferentes criterios se pueden combinar entre ellos gracias a los operadores booleanos (`AND`, `OR`, `XOR` y `NOT`).

2. Algunos ejemplos

La siguiente consulta muestra los campos `ANI_NOMBRE` y `ANI_FECHA_NACIMIENTO` de los registros de la tabla `ENI_ANIMAL_ANI`, cuyas fechas de adopción están entre el 1 de enero de 2016 y el 1 de junio de 2018.

```
SELECT ANI_NOMBRE, ANI_FECHA_NACIMIENTO
FROM ENI_ANIMAL_ANI
WHERE ANI_FECHA_ADOPCION BETWEEN #01/01/2016# AND #06/01/2018
```

La siguiente consulta muestra los nombres y apellidos de los registros de la tabla `ENI_ADOPTANTE_ADO` cuyo Nombre empieza por A.

```
SELECT ADO_NOMBRE, ADO_APELLIDO
FROM ENI_ADOPTANTE_ADO
WHERE ADO_NOMBRE LIKE 'A*'
```

Los cálculos en las consultas

Es posible realizar cálculos sobre los datos a partir de las consultas SQL. Por ejemplo, determinar el número total de animales por caja, o incluso el precio total que debe pagar un adoptante por sus adopciones.

Existen varias funciones en SQL (total `COUNT`, suma `SUM`, mínimo `MIN`, máximo `MAX`, etc.), que se completan con otras funciones directamente de VBA (`Left`, `Right`, `Mid`, `DateSerial`, etc.).

Por ejemplo, la siguiente consulta permite visualizar los nombres y las iniciales del apellido de todos los adoptantes.

```
SELECT ADO_NOMBRE, Left(ADO_APELLIDO,1) As Iniciales
FROM ENI_ADOPTANTE_ADO
```

La cláusula GROUP BY

Cuando se hacen cálculos en una consulta, algunas veces es necesario agrupar los campos sobre los que no se hace ningún cálculo. El cálculo solo devolverá una única línea por grupo. Las agrupaciones se hacen con la cláusula GROUP BY.

La siguiente consulta permite sumar los gastos totales de adopción para cada adopción.

```
SELECT ANI_ADO_ID, SUM(ANI_GASTOS_ADOPCION) AS Gastos_Totales
FROM ENI_ANIMAL_ANI
WHERE ANI_ADO_ID IS NOT NULL
GROUP BY ANI_ADO_ID;
```

De manera general, todos los campos sobre los que no se hace ningún cálculo deben aparecer en la agrupación. Si omite un campo en la agrupación, Access se lo indicará cuando se ejecute la consulta.

La cláusula HAVING

Así como es posible filtrar los valores en los registros, también es posible aplicar criterios de filtro a los cálculos de agrupación. Para esto se usa la cláusula HAVING.

Por ejemplo, si solo se desea visualizar las cajas cuyo número de animales ha sido estrictamente inferior a 5:

```
SELECT COUNT(ENI_CAJA_ANIMAL_BOA.BOA_CAJA_ID) AS CuentaDeBOA_CAJA_ID,
ENI_CAJA_CAJA.CAJA_NOMBRE
FROM ENI_CAJA_CAJA INNER JOIN ENI_CAJA_ANIMAL_BOA
ON ENI_CAJA_CAJA.CAJA_ID = ENI_CAJA_ANIMAL_BOA.BOA_CAJA_ID
GROUP BY ENI_CAJA_CAJA.CAJA_NOMBRE
HAVING ((((Count(ENI_CAJA_ANIMAL_BOA.[BOA_CAJA_ID]))<5));
```

La cláusula ORDER BY

Durante la visualización de los datos resultantes de la consulta, es posible ordenarlos. La cláusula `ORDER BY` permite indicar en qué campos se desea realizar una ordenación. Es posible ordenar de dos maneras diferentes: creciente (ascendente), con la palabra clave `ASC`, y decrenciente (descendiente), con la palabra clave `DESC`. Si no se especifica ningún orden, el orden por defecto que se aplica es el creciente (`ASC`). Cada campo puede tener su propio orden.

La sintaxis SQL es la siguiente:

```
ORDER BY Campo_1 [ASC o DESC] [, Campo_2 [ASC o DESC]]
```

Por ejemplo, la siguiente consulta hará que aparezcan los animales por orden de fecha de adopción y los nombres de animales por orden alfabético (creciente).

```
SELECT ANI_NOMBRE, ANI_FECHA_ADOPCION
FROM ENI_ANIMAL_ANI
ORDER BY ANI_FECHA_ADOPCION DESC, ANI_NOMBRE ASC
```

Los alias, el operador AS

Es posible añadir tantas columnas como se quiera en una consulta (con el límite de 255 campos máximo). Para esto se usa la palabra clave AS. Este operador también permite asignar un nombre a cómo queremos ver un campo calculado.

Ejemplo de alias utilizado para el número de vacunas por fecha:

```
SELECT COUNT(*) AS Num_Vacunacion, ANV_FECHA VACUNACION As Fecha_Vacunacion
FROM ENI_ANIMAL_VACUNACION_ANV
GROUP BY ANV_FECHA_VACUNACION
```

El Mundo de la Programación en tus Manos...!

DETODOPROGRAMACION.ORG

DETODOPROGRAMACION.ORG

Material para los amantes de la Programación Java, C/C++/C#,Visual.Net, SQL, Python, Javascript, Oracle, Algoritmos, CSS, Desarrollo Web, Joomla, jquery, Ajax y Mucho Mas…

VISITA

www.detodoprogramacion.org
www.detodopython.com
www.gratiscodigo.com

GRATISCODIGO.COM
Usa, lo que ya se creó...

Detodo Python.com

El comando INSERT INTO

Además de las consultas de selección de datos, es posible realizar consultas, llamadas acciones, que se mencionarán en las siguientes secciones. La primera consulta de tipo acción posible es la de inserción de nuevos registros en las tablas. Este modo corresponde al modo Añadir en la interfaz de Access. Hay varios métodos para insertar nuevos datos.

1. Añadir un registro

En primer lugar, es posible añadir un registro único con la siguiente sintaxis:

```
INSERT INTO Tabla_Destino (<Lista de los Campos>)
VALUES (<Lista de los Valores>)
```

La lista de los campos contiene los campos que se alimentarán, cada uno separado del resto, por una coma. A cada campo se le asignará un valor, que se corresponde con el valor de la lista de valores que aparece en el mismo orden.

Es necesario tener el mismo número de campos y valores asignados, y que los tipos de valores introducidos y su orden sea idéntico. En caso contrario, la consulta se interpretará incorrectamente, incluso Access la rechazará durante su ejecución.

Ejemplo de inserción de una nueva vacuna:

```
INSERT INTO ENI_VACUNA_VAC (VAC_NOMBRE, VAC_ANT_ID)
VALUES ('Parvovirosis del perro', 1)
```

2. Adición como resultado de una consulta

En caso de que quiera insertar registros a partir de una consulta, es posible hacerlo con la siguiente sintaxis:

```
INSERT INTO Tabla_Destino [(<Lista de los Campos>)]
SELECT <Lista de los Campos>
FROM <Tabla_Origen>
```

Aquí, la lista de campos de la tabla `Tabla_Origen` debe corresponder en tipos y orden con los que se desea insertar en `Tabla_Destino`.

Ejemplo de inserción de vacunaciones para todos los perros para la vacuna con identificador 1, el 1 de octubre de 2018:

```
INSERT INTO ENI_ANIMAL_VACUNACION_ANV (ANV_ANI_ID, ANV_VAC_ID,
ANV_FECHA_VACUNACION)
SELECT ANI_ID, 1, #10/01/2018#
FROM ENI_ANIMAL_ANI
WHERE ANI_ANT_ID=1
```

El comando SELECT INTO

De la misma manera que la consulta INSERT INTO va a añadir registros nuevos en una tabla existente, la consulta SELECT INTO va a crear una nueva tabla y añadir en ella los registros. La sintaxis general de este comando es la siguiente:

```
SELECT Campo_1 [, Campo_2] INTO TablaDestino
FROM TablaOrigen
[WHERE <lista de condiciones>]
```

Se va a crear una tabla TablaDestino sobre la marcha (si la tabla TablaDestino ya existe, aparecerá un mensaje de alerta indicando que ya existe y se eliminará). Cada campo de la consulta se creará en la tabla TablaDestino, y después se añadirán los registros del resultado de la consulta SELECT.

Ejemplo de consulta de inserción de las vacunas anteriores a 2018 en una tabla T_VACUNACION_ARCHIVADA:

```
SELECT * INTO T_VACUNACION_ARCHIVADA FROM ENI_ANIMAL_VACUNACION_ANV
WHERE ANV_FECHA_VACUNACION<=#01/01/2018#
```

El comando UPDATE

El comando `UPDATE` se utiliza para actualizar los datos ya presentes en las tablas. Este comando corresponde al tipo de consulta Actualización en Access 2019. También se trata de una consulta de tipo Acción. La sintaxis general de este comando es la siguiente:

```
UPDATE Tabla_ACT
SET Campo_1=valor_1 [, Campo_2=Valor_2]
[WHERE <lista de condiciones de actualización>]
```

Para cada registro que responde a las condiciones propuestas, se actualizará sus campos. Si no hay ninguna condición (no hay cláusula `WHERE`), se actualizan todos los registros.

Ejemplo de una consulta de actualización:

```
UPDATE ENI_ANIMAL_ANI
SET ANI_DESTETADO=TRUE
WHERE ANI_NOMBRE='Maki'
```

Esta consulta actualiza el destete para el animal que se llama Maki.

El comando DELETE

El comando DELETE permite eliminar registros. Corresponde al tipo de consulta Eliminación de Access 2019. La sintaxis general es la siguiente:

```
DELETE * FROM Tabla_A_Vaciar
[WHERE <lista de condiciones>]
```

Se eliminarán todos los registros de la tabla Tabla_A_Vaciar que se corresponden con las condiciones propuestas.

Ejemplo de consulta de eliminación de las vacunas anteriores a 2015:

```
DELETE * FROM ENI_ANIMAL_VACUNACION_ANV
WHERE ANV_FECHA_VACUNACION<=#01/01/2015#
```

Los otros comandos

La lista de comandos no se limita a la selección, inserción, actualización y eliminación. A continuación, se muestra la lista del resto de los comandos SQL añadidos por Access 2019.

1. Consulta de análisis cruzado

La consulta `TRANSFORM` crea una consulta de análisis cruzado. Se utiliza en el asistente de creación de la consulta.TRANSFORM

2. Consulta de tipo Union

La sintaxis `UNION` permite fusionar el resultado de varias consultas cuyas estructuras y campos son idénticos

3. Creación/administración de una tabla

`CREATE TABLE`	Crea una nueva tabla.
`ALTER TABLE`	Modifica la estructura de una tabla.
`DROP TABLE`	Elimina una tabla.
`CREATE INDEX`	Crea un nuevo índice en una tabla existente.
`DROP INDEX`	Elimina un índice.

4. Creación/administración de los usuarios y los grupos

Usuarios

`CREATE USER`	Crea uno o varios usuarios nuevos.
`ADD USER`	Añade un usuario a un grupo de usuarios existente.
`DROP USER`	Elimina uno o varios usuarios.

Grupos

`CREATE GROUP`	Crea uno o varios grupos de usuarios nuevos.
`DROP GROUP`	Elimina uno o varios grupos existentes.

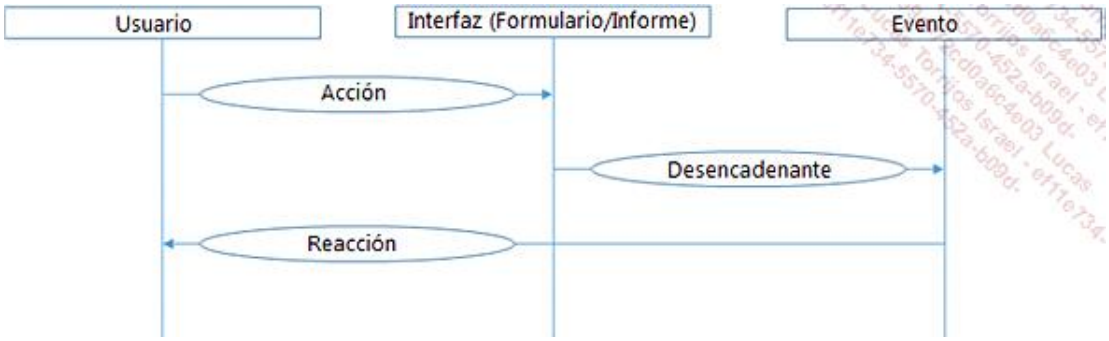
Permisos

`GRANT`	Da permisos concretos a un usuario o a un grupo de usuarios existente.
`REVOKE`	Retira permisos concretos a un usuario o a un grupo de usuarios existente.

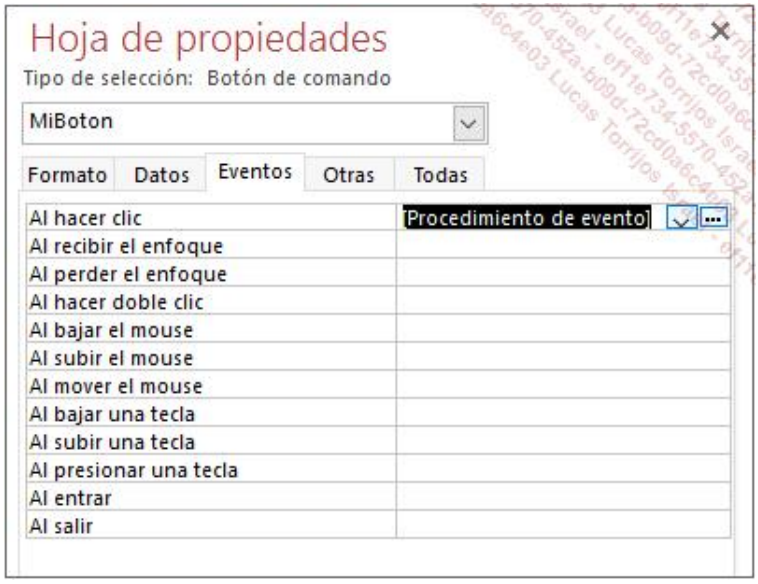
Cada uno de estos comandos se explica en la ayuda de Access (tecla [F1]).

Definición

Un evento permite ejecutar una o varias instrucciones como resultado de una acción por parte de un usuario. La acción desencadena la ejecución del código del procedimiento del evento correspondiente, que está relacionado con el objeto activo.



Para asociar un código VBA durante una acción sobre un objeto, podemos pasar por la pestaña **Evento**, en la pestaña de navegación **Hoja de propiedades**:



También es posible pasar en el editor VBE, por las zonas de listas desplegables (zona izquierda para seleccionar el objeto y zona derecha para seleccionar el evento).

```
BtnAnnuler                                   Click
Option Compare Database
Option Explicit

'variables temporales
Public oAnimal As clsAnimal
Private pANI_ID As Long

'Botones
Private Sub BtnGuardar_Click()
    Guardar True
End Sub

Public Sub Guardar(Salir As Boolean)
    'si el animal tiene todos los elementos necesarios, se guarda la información
    If oAnimal.EsValido Then
        pANI_ID = oAnimal.Guardar
        If Salir Then
            DoCmd.Close acForm, Me.Name
        Else
            SetParam "ANI_ID", pANI_ID
            SetParam "ADO_ID", oAnimal.AdoptanteID
            SetParam "DT_ADOPCION", oAnimal.DateAdopcion
        End If
        'si ele formulario de lista de los animales está abierto, se actualiza la lista de información.
        If CurrentProject.AllForms("F_Animales").IsLoaded Then
            Form_F_Animales.LstAnimales.Requery
        End If
    Else
        MsgBox "La información está incompleta o es incorrecta", vbCritical + vbOKOnly
    End If
End Sub

'se cierra el formulario sin guardar
Private Sub BtnCancelar_Click()
    DoCmd.Close acForm, Me.Name
End Sub
```

El objetivo de este capítulo es revisar los diferentes tipos de eventos, mostrar cómo anular algunos y mencionar el orden en que se desencadenan los eventos durante acciones simples.

Tipos de eventos

Las descripciones también se aplican a los informes.

1. Los eventos durante la apertura

Evento (Propiedad)	Se produce	Anulable
Open (durante la apertura)	Durante la apertura del formulario, pero antes de que se muestre el primer registro.	Sí
Load (durante el cambio)	Durante la apertura del formulario, una vez mostrado el primer registro.	No
Resize (durante el redimensionamiento)	Cuando el formulario cambia de dimensiones, así como durante su primera visualización.	No
Activate (durante la activación)	Cuando el formulario se hace activo.	No
Current (durante la activación)	Durante la apertura del formulario, antes de que el primer registro se convierta en registro actual.	No

2. Los eventos durante el cierre

Evento (Propiedad)	Se produce	Anulable
Unload (durante la liberación)	Durante el cierre del formulario, una vez que se han liberado los registros, pero antes de la desaparición del formulario.	Sí
Deactivate (durante la desactivación)	Durante el cierre del formulario, así como durante la activación de otra ventana, antes de que la otra ventana se active.	No
Close (durante el cierre)	Cuando el formulario desaparece durante el cierre.	No

3. Los eventos de adición, eliminación o modificación de registros

Evento (Propiedad)	Se produce	Anulable
BeforeInsert (antes de la inserción)	Durante la escritura del primer carácter de un nuevo registro, pero antes la adición del nuevo registro.	Sí
AfterInsert (después de la inserción)	Después de la adición de un nuevo registro en la tabla.	No
BeforeUpdate (antes actualización)	Antes de la actualización de un registro o de un control (modificación de los datos).	Sí
Change (durante el cambio)	Durante la modificación del contenido de una lista modificable o de una zona de texto.	No
Updated (durante la actualización)	Cuando los datos de un objeto OLE se han modificado.	No
AfterUpdate (después actualización)	Después de la actualización de un registro o de un control (modificación de las datos).	No
BeforeDelConfirm (antes	Después de que el usuario elimine los registros,	Sí

eliminación)	pero antes de que Access pida confirmación de esta eliminación.	
Delete (durante la eliminación)	Antes de la eliminación efectiva (pulsar la tecla Supr del teclado, por ejemplo).	Sí
AfterDelConfirm (después de la eliminación)	Antes de que el usuario haya respondido a la confirmación de la eliminación de los registros.	No
NotInList (durante la ausencia en la lista)	Durante la introducción de un valor no disponible en una lista y cuando la lista está limitada.	No

4. Los eventos de detección de un error, modificación y anulación

Evento (Propiedad)	Se produce	Anulable
Undo (durante la anulación)	Cuando el usuario anula su modificación (pulsa la tecla [Esc] del teclado).	Sí
Dirty (si hay modificación)	Cuando cambia el contenido de un formulario o una zona de lista desplegable. También tiene lugar cuando el usuario cambia de página en una pestaña.	Sí
Error (durante el error)	Cuando se produce un error.	No

5. Los eventos relacionados con el foco

Evento (Propiedad)	Se produce	Anulable
Enter (durante la entrada)	Antes de que un control reciba el foco.	No
GotFocus (durante la recepción del foco)	Cuando el formulario o un control recibe el foco.	No
Exit (durante la salida)	Antes de que un control pierda el foco.	Sí
LostFocus (durante la pérdida del foco)	Cuando el formulario o un control pierde el foco.	No

6. Los eventos de los periféricos de ratón y teclado

Evento (Propiedad)	Se produce	Anulable
Click (durante el clic)	Durante el clic con el botón izquierdo del ratón, en una zona vacía del formulario o sobre un control.	No
DblClick (durante el doble clic)	Durante dos clics con el botón izquierdo del ratón en una zona vacía del formulario o sobre un control.	Sí
MouseDown (durante la pulsación en el ratón)	Durante el clic con un botón del ratón en un formulario o un control.	Sí
MouseUp (cuando se suelta el botón del ratón)	Cuando el usuario suelta el botón del ratón en un formulario o un control.	No
MouseMove (cuando se mueve el ratón)	Cuando el usuario mueve el ratón en un formulario o un control.	No
KeyDown (con tecla pulsada)	Durante la pulsación en una tecla o durante la ejecución de EnviarTeclas o SendKeys.	No

	Podemos impedir que el objeto reciba la tecla poniendo como argumento Keycode a 0 en el procedimiento KeyDown.	
KeyUp (cuando se suelta la tecla)	Cuando el usuario suelta la tecla o durante la ejecución de EnviarTeclas o SendKeys. Podemos impedir que el objeto reciba la tecla poniendo el argumento Keycode a 0 en el procedimiento KeyUp.	No
KeyPress (con tecla activada)	Durante la pulsación en una tecla que envía un carácter ANSI o durante la ejecución de EnviarTeclas o SendKeys. Podemos impedir que el objeto reciba la tecla poniendo el argumento KeyASCII a 0 en el procedimiento KeyPress.	Sí (únicamente por VBA o por una macro)
MouseWheel (cuando se usa la rueda del ratón)	Cuando el usuario utiliza la rueda del ratón. Podemos controlar el efecto del ratón con las propiedades Page y Count.	Sí

7. Los eventos de filtrado de datos

Evento (Propiedad)	Se produce	Anulable
Filter (durante el filtro)	Cuando el usuario crea un filtro.	Sí
ApplyFilter (cuando se aplica el filtro)	Cuando el usuario aplica un filtro.	Sí

8. Los eventos autónomos

Evento (Propiedad)	Se produce	Anulable
Timer (durante el temporizador)	Cuando se especifica un tiempo de espera y se alcanza.	No

9. Los eventos concretos de los informes

Evento (Propiedad)	Se produce	Anulable
NoData (cuando no hay datos)	Cuando Microsoft Access formatea un informe y este no contiene ningún dato.	Sí
Page (durante la paginación)	Antes de que Microsoft Access formatee una página de un informe para su impresión, pero antes de imprimirse.	No
Formato (durante el formateo)	Cuando Microsoft Access determina qué datos pertenecen a qué secciones.	Sí
Print (durante la impresión)	Durante el formateo de los datos de una sección para impresión.	Sí
Retreat (durante el reformateo)	Cuando Microsoft Access vuelve a una sección anterior durante el formateo del informe.	No

10. Otros eventos

Hay otros eventos que también se desencadenan, como los de renderizado (`BeforeRender`, `AfterRender`), pivotado de una tabla (`BeforeQuery`, `DataChange`, `OnConnect`, `OnDisconnect`, `PivotTablaChange`, `SelectionChange` y `ViewChange`) o incluso los eventos globales de `Command` (`CommandBeforeExecute`, `CommandChecked`, `CommandEnabled` y `CommandExecute`).

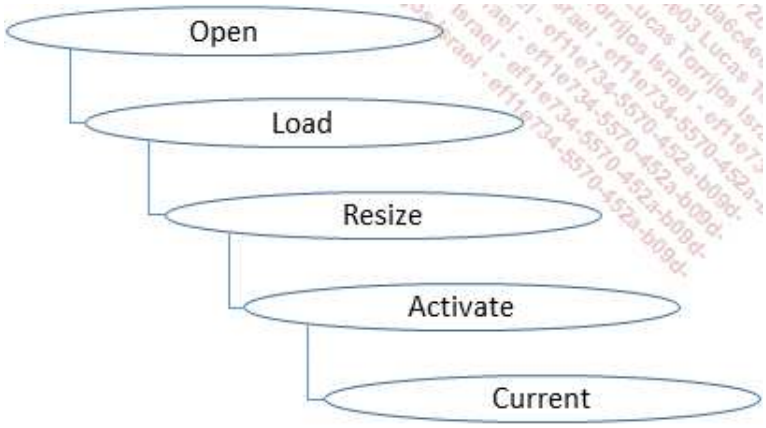
Anulación de un evento

Es posible, y algunas veces muy práctico, anular un evento. Los diferentes eventos listados anteriormente se pueden o no anular. Para anular un evento, es posible pasar como argumento `AnularEvento` a la macro, o poner a `True` la variable `Cancel` en el código VBA.

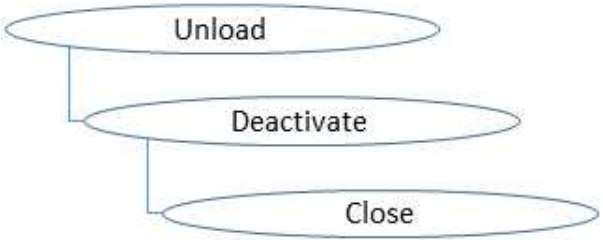
Orden de realización de los eventos

En algunas situaciones, tienen lugar varios eventos unos después de otros. Este capítulo especifica cómo funcionan. Los eventos se representan en orden cronológico de arriba abajo.

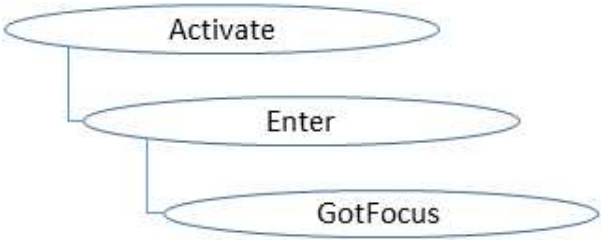
1. Durante la apertura de un formulario



2. Durante el cierre de un formulario

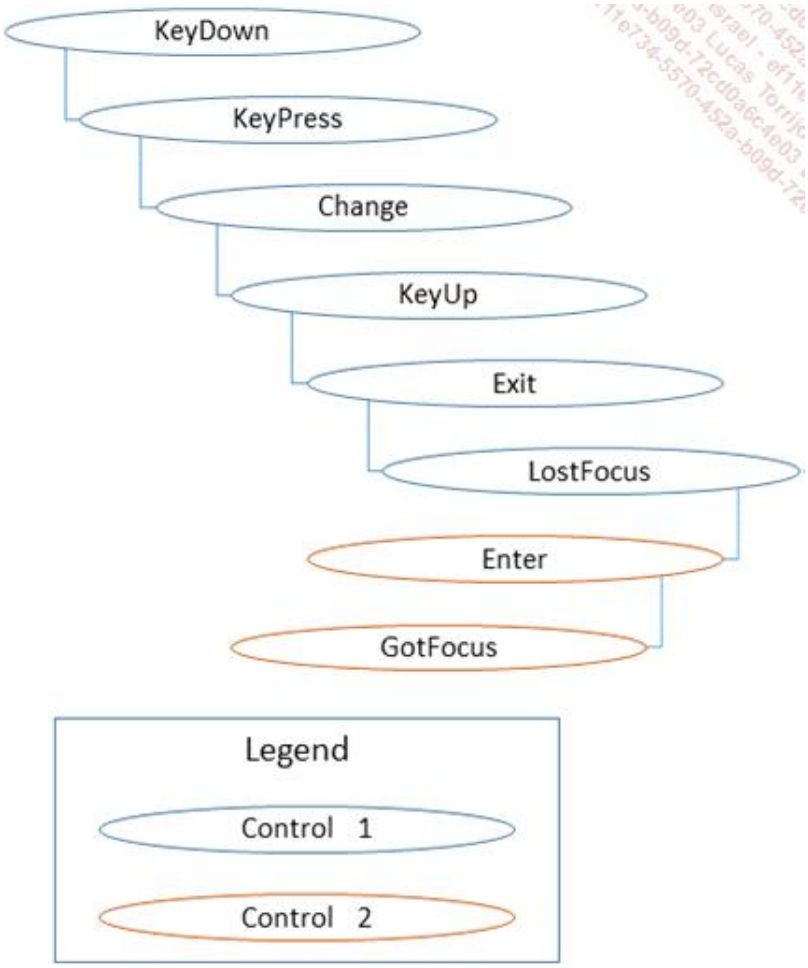


3. Durante la activación de un formulario ya abierto



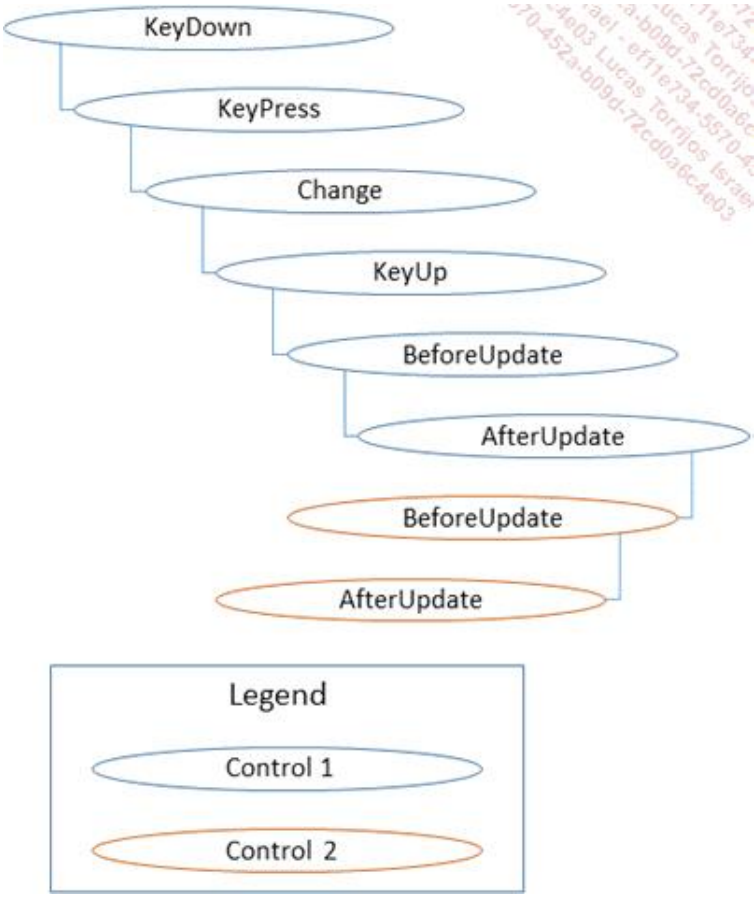
4. Durante la actualización de un control

Tiene lugar cuando el usuario termina de escribir y pasa al siguiente control.

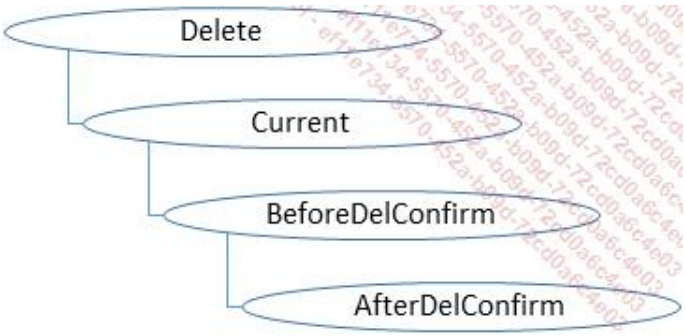


5. Durante la actualización de un registro

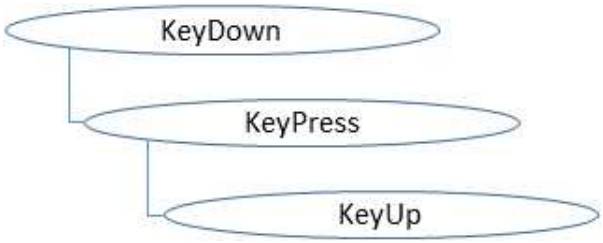
Tiene lugar cuando el usuario termina de escribir en un registro y pasa a otro registro.



6. Durante la eliminación de un registro



7. Durante la pulsación de una tecla



Orden de los eventos durante una actualización

En Microsoft Access, los eventos relacionados con la actualización se desencadenan a dos niveles diferentes:

1. A nivel del control

Durante la actualización de un control, se ejecutan los eventos `BeforeUpdate` y `AfterUpdate` del control. Preceden a los eventos del formulario.

2. A nivel de un registro

Durante la actualización de un registro, se ejecutan los eventos `BeforeUpdate` y `AfterUpdate` del formulario.

Introducción

Es posible personalizar los formularios e informes con el lenguaje VBA. Por ejemplo, es posible:

- modificar la apariencia de un control en un informe o un formulario (tamaño, posición, color, etc.),
- añadir o eliminar controles en un informe o un formulario,
- ver sobre la marcha formularios, haciendo clic en botones,
- imprimir un informe,
- etc.

El objeto Form

El objeto `Form` pertenece a la colección `Forms` y se corresponde con un formulario `Forms` que da la lista del resto de los formularios abiertos. El objeto `Activeform` apunta al formulario activo.

1. Sintaxis

Para referirse a un formulario, hay varias sintaxis posibles, directamente relacionadas con el recorrido de los elementos de la colección `Forms`.

```
Forms!NombreFormulario
    'Ejemplo Forms!Formaciones
Forms![NombreFormulario]
    'Ejemplo: Forms![Formaciones]
Forms("NombreFormulario")
    'Ejemplo: Forms("Formaciones")
Forms(Index)
    'Ejemplo: Forms(2)
```

La sintaxis más frecuente es `Forms![NombreFormulario]`, permitiendo el uso de un nombre de formulario con un espacio, lo que la primera sintaxis no permite hacer.

Para referirse a un subformulario a partir del formulario padre, es necesario pasar por el nombre del objeto que contiene el subformulario, seguido de la palabra clave `Form`.

```
[SubFormulario].Form!ZonaDeTexto
    'Ejemplo: [Participantes].Form.Txt_Nombre
```

2. Equivalencia modo Creación/VBA

a. Pestaña Formato

En la pestaña **Formato** del panel de navegación **Hoja de propiedades**, pueden aparecer los siguientes elementos.

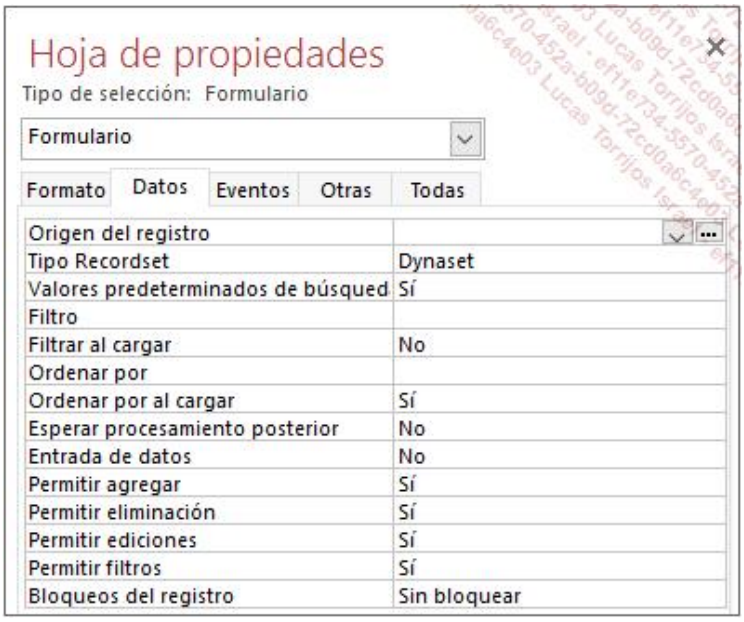
La siguiente tabla lista las propiedades en la interfaz, su equivalente en VBA, así como el tipo de dato de la propiedad.

Etiqueta IHM	Propiedad VBA	Tipo de dato
Título	Caption	String
Vista predeterminada	DefaultView	Byte
Permitir vista Formulario	AllowFormView	Boolean
Permitir vista Hoja de datos	AllowDatasheetView	Boolean
Permitir vista Presentación	AllowLayoutView	Boolean
Tipo de imagen	PictureType	Byte
Imagen	Picture	String
Mosaico de imágenes	PictureTiling	Boolean
Distribución de la imagen	PictureAlignment	Byte
Modo de tamaño de la imagen	PictureSizeMode	Byte
Ancho	Width	Integer
Centrado automático	AutoCenter	Boolean
Ajuste de tamaño automático	AutoResize	Boolean
Ajustar a la pantalla	FitToScreen	Boolean
Estilo de los bordes	BorderStyle	Byte
Selectores de registro	RecordSelector	Boolean
Botones de desplazamiento	NavigationButtons	Boolean
Título de exploración	NavigationCaption	Boolean
Separadores de registros	DividingLines	Boolean
Barras de desplazamiento	ScrollBars	Byte
Cuadro de control	ControlBox	Boolean
Botón Cerrar	CloseButton	Boolean
Botones Minimizar Maximizar	MinMaxButtons	Byte
Movible	Moveable	Boolean
Tamaño del formulario dividido	SplitFormSize	Long
Orientación del formulario dividido	SplitFormOrientation	Byte
Barra divisora del formulario dividido	SplitFormSplitterBar	Boolean
Hoja de datos del formulario dividido	SplitFormDatasheet	Byte
Impresión del formulario dividido	SplitFormPrinting	Byte
Guardar la posición de la barra divisora	SplitFormSplitterBarSave	Boolean
Hoja secundaria de datos expandida	SubDatasheetExpanded	Boolean
Alto de hoja secundaria de datos	SubDatasheetHeight	Integer
Línea X	GridX	Integer
Línea Y	GridY	Integer
Diseño a imprimir	LayoutForPrint	Boolean
Orientación	Orientation	Byte

Origen de la paleta	PaletteSource	String

b. Pestaña Datos

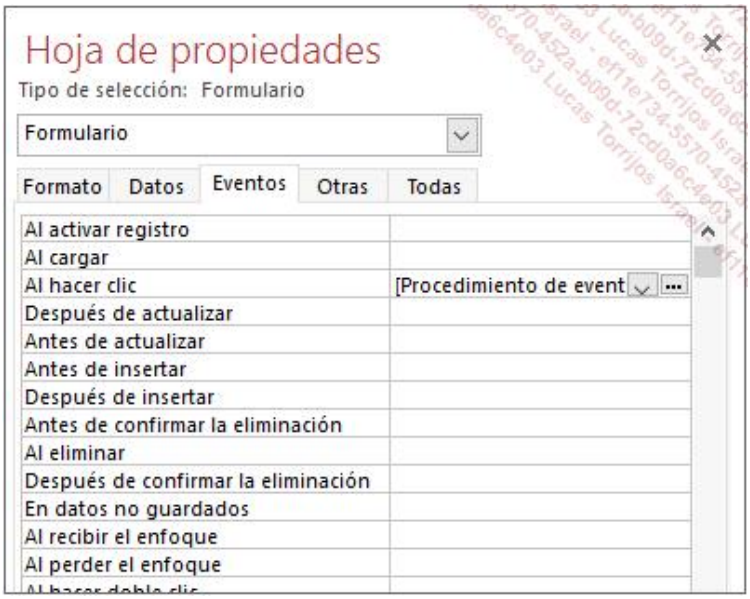
En la pestaña **Datos** del panel de navegación **Hoja de propiedades**, pueden aparecer los siguientes elementos:



Etiqueta IHM	Propiedad VBA	Tipo de dato
Origen del registro	RecordSource	String
Tipo Recordset	TypeRecordset	Byte
Valores predeterminados de búsqueda	FetchDefaults	Boolean
Filtro	Filter	String
Filtrar al cargar	FilterOnLoad	Boolean
Ordenar por	OrderBy	String
Ordenar por al cargar	OrderByOnLoad	Boolean
Esperar procesamiento posterior		
Entrada de datos	DataEntry	Boolean
Permitir agregar	AllowAdditions	Boolean
Permitir eliminar	AllowDeletions	Boolean
Permitir ediciones	AllowEdits	Boolean
Permitir filtros	AllowFilters	Boolean
Bloqueos del registro	RecordLock	Byte

c. Pestaña Eventos

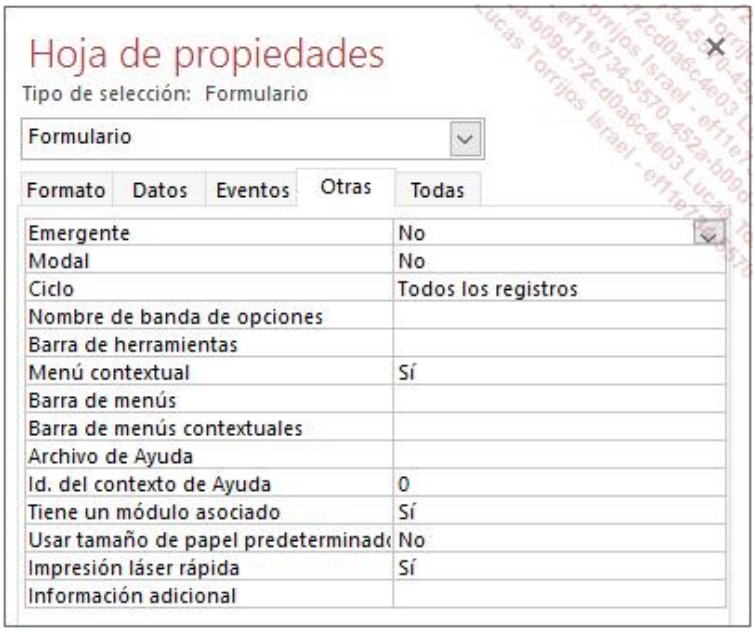
En la pestaña **Eventos** del panel de navegación **Hoja de propiedades**, pueden aparecer los siguientes elementos:



Los diferentes eventos se tratan en el capítulo Los eventos de Access.

d. Pestaña Otras

En la pestaña **Otras** del panel de navegación **Hoja de propiedades**, pueden aparecer los elementos siguientes.



Etiqueta IHM	Propiedad VBA	Tipo de dato
Emergente	`Popup`	`Boolean`
Modal	`Modal`	`Boolean`
Ciclo	`Cycle`	`Byte`
Nombre de banda de opciones	`RibbonName`	`String`
Barra de herramientas	`ToolBar`	`String`
Menú contextual	`ShortcutMenu`	`Boolean`
Barra de menús	`MenuBar`	`String`

Barra de menús contextuales	ShortcutMenuBar	String
Archivo de Ayuda	HelpFile	String
Id. del contexto de Ayuda	HelpContextID	Long
Tiene un módulo asociado	HasModule	Boolean
Usar tamaño de papel predeterminado		
Impresión láser rápida	FastLaserPrinting	Boolean
Información adicional	Tag	String

3. Otras propiedades disponibles en VBA

Algunas propiedades no están disponibles en modo Creación, pero aparecen en VBA.

a. Propiedades relacionadas con los registros

Propiedad	Descripción
Bookmark	Define un marcador identificando de manera única un registro.
CurrentRecord	Devuelve el registro actual.
Dirty	Permite determinar si el registro actual ha sufrido modificaciones.
NewRecord	Permite determinar si el registro actual es un nuevo registro.
RecordSet	Devuelve o define el objeto DAO Recordset, presentando el origen de los datos del formulario.
RecordSourceQualifier	Devuelve o define una cadena de caracteres que indica el nombre del propietario del origen del registro SQL Server para el formulario.

b. Propiedades relacionadas con la visualización

Propiedad	Descripción
CurrentView	Devuelve el modo de visualización del formulario.
OpenArgs	Devuelve el argumento OpenArgs del método OpenForm que ha abierto el formulario.
Page	Devuelve o define el número de página actual.
Section	Identifica una sección y permite acceder a sus propiedades.
SelHeight	Devuelve o define el número de registros seleccionados en el rectángulo de selección actual.
SelLeft	Devuelve o define el campo (columna) más a la izquierda en el rectángulo de selección actual.
SelTop	Devuelve o define el registro que está arriba del todo del rectángulo de selección actual.
SelWidth	Devuelve o define el número de campos (columnas) seleccionadas en el rectángulo de selección actual.

c. Propiedades relacionadas con la presentación del formulario

Propiedad	Descripción
CurrentSectionLeft	Representa la distancia en twips entre la esquina superior izquierda de la sección actual y la esquina superior izquierda del formulario.
CurrentSectionTop	Representa la distancia en twips entre el borde superior de la sección actual y el borde superior del formulario.
DatasheetAlternate BackColor	Representa el código de color mostrado en el resto de las líneas de la hoja de datos de un formulario.
DatasheetBackColor	Representa el código de color de segundo plano de la totalidad de la hoja de datos.
DatasheetBorder LineStyle	Representa el estilo del trazo utilizado para el borde de la hoja de datos.
DatasheetCellsEffect	Indica si de los efectos especiales se aplican a las celdas de una hoja de datos.
DatasheetColumnHeader UnderlineStyle	Representa el estilo de trazo utilizado para el borde inferior de los encabezados de columnas de la hoja de datos.
DatasheetFontHeight	Representa el tamaño en puntos utilizado para visualizar e imprimir los nombres de campos y los datos en modo Hoja de datos.
DatasheetFontItalic	Muestra en itálica los nombres de campos y los datos en modo Hoja de datos.
DatasheetFontName	Determina el nombre del tipo de letra utilizado para visualizar e imprimir los nombres de campos y los datos en modo Hoja de datos.
DataFontUnderline	Muestra en subrayado los nombres de campos y los datos en modo Hoja de datos.
DatasheetFontWeight	Representa el tamaño del tipo de letra utilizado para visualizar e imprimir los nombres de campos y los datos en modo Hoja de datos.
DatasheetForeColor	Representa el color del texto en la hoja de datos.
DatasheetGridLinesBehavior	Representa el tipo de cuadrícula utilizada en la hoja de datos.
DatasheetGridLinesColor	Representa el color de la cuadrícula de la hoja de datos.
HorizontalDatasheet GridLineStyle	Representa la cuadrícula horizontal que se debe utilizar para el borde de la hoja de datos.
InsideHeight	Representa la altura de la ventana que contiene el formulario (en twips).
InsideWidth	Representa la anchura de la ventana que contiene el formulario (en twips).
Pages	Devuelve el número de páginas.
Painting	Especifica si el formulario se debe estirar.
PaintPalette	Representa la paleta que debe utilizar el formulario.
PictureData	Permite la copia de la imagen del formulario en otro objeto que pueda utilizar la imagen (objeto Picture).
VerticalDatasheet GridLineStyle	Representa la cuadrícula vertical que se debe utilizar para el borde de la hoja de datos.
WindowHeight	Representa la altura del formulario (en twips).
WindowLeft	Representa la posición de la ventana respecto al borde

	izquierdo del formulario.
WindowTop	Representa la posición de la ventana respecto al borde superior del formulario.
WindowWidth	Representa la anchura del formulario (en twips).

d. Propiedades relacionadas con la impresión

Propiedad	Descripción
PrtDevMode	Representa la información relativa al modo del periférico de impresión indicado.
PrtDevNames	Representa la información relativa a la impresora indicada.
PrtMip	Representa la información del modo del periférico indicado.
UseDefaultPrinter	Indica si el formulario utilizará la impresora por defecto del sistema a la hora de imprimir.

e. Propiedades que devuelve un objeto

Propiedad	Descripción
Application	Devuelve el objeto Application (Access).
ChartSpace	Devuelve el objeto ChartSpace (espacio gráfico).
Form	Devuelve el formulario (caso con un subformulario).
Module	Devuelve el objeto Module (módulo de código relacionado con el formulario). Se podrá añadir (InsertLines), eliminar (DeleteLines) o modificar (ReplaceLines) líneas de código.
Parent	Devuelve el objeto padre del formulario.
PivotTable	Devuelve el objeto PivotTable (tabla cruzada dinámica).
Printer	Devuelve un objeto Printer (impresora por defecto).
RecordsetClone	Devuelve el objeto Recordset que sirve de origen al formulario.

4. Métodos de los formularios

También es posible realizar un determinado número de acciones con un formulario.

Método	Descripción
GotoPage	Permite activar el primer control de la página indicada.
Move	Permite mover el formulario a la ubicación concreta (coordenadas).
Recalc	Permite recalcular los diferentes controles del formulario.
Refresh	Permite refrescar los datos del formulario. Esto sirve para tener en cuenta las posibles modificaciones añadidas por el resto de los usuarios de la aplicación.
Repaint	Permite realizar todas las actualizaciones en el formulario.
Requery	Permite refrescar los datos resultantes de la consulta origen del formulario.
SetFocus	Permite dar el foco al formulario.
UnDo	Permite anular las modificaciones añadidas al formulario.

5. Ejemplo

El siguiente código permite personalizar los elementos de visualización durante la apertura del formulario F_Animal:

```
Private Sub Form_Open(Cancel as Integer)
    Me.Caption = "Animal '" & Me.txtNombreAni.Value & "' -
Estamos  a " & Date
    'personalización de la visualización
    Me.RecordSelectors = False
    Me.NavigationButtons = True
    Me.FitToScreen = True
    Me.Modal = True
    'activación de la zona de texto Titulo
    Me.txtNombreAni.SetFocus
End Sub
```

El siguiente código permite cerrar el formulario, cuando el usuario hace clic en el botón **Anular** del formulario F_Animal:

```
Private Sub BtnAnular_Click()
    DoCmd.Close acForm, Me.Name
End Sub
```

El objeto Report

El objeto `Report`, que pertenece a la colección `Reports`, corresponde a un informe, `Reports`, que da la lista de todos los informes abiertos. El objeto `ActiveReport` apunta al informe activo.

1. Sintaxis

Para referirse a un informe, hay varias sintaxis posibles, directamente relacionadas con el recorrido de elementos de la colección `Reports`.

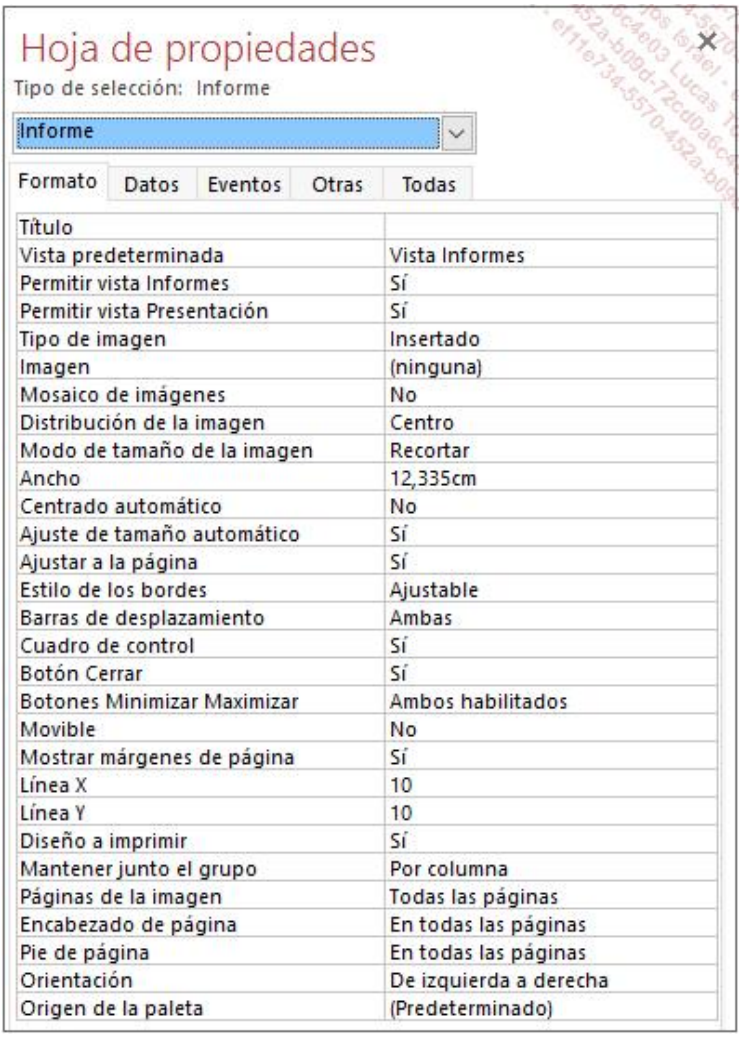
```
Reports!NombreInforme
    'Ejemplo Reports!Formation
Reports![NombreInforme]
    'Ejemplo: Reports![Formation]
Reports("NombreInforme")
    'Ejemplo: Reports("Formaciones")
Reports(Index)
    'Ejemplo: Reports(1)
```

Como para los formularios, la sintaxis más frecuente es `Reports![Nombre Informe]`, permitiendo el uso de un nombre de informe que tiene un espacio, cosa que no es posible con la primera sintaxis.

2. Equivalencia modo Creación/VBA

a. Pestaña Formato

En la pestaña **Formato** del panel de navegación **Hoja de propiedades**, pueden aparecer los siguientes elementos:



La tabla siguiente lista las propiedades en la interfaz, su equivalente en VBA, así como el tipo de dato de la propiedad.

Etiqueta IHM	Propiedad VBA	Tipo de dato
Título	`Caption`	`String`
Vista predeterminada	`DefaultView`	`Byte`
Permitir vista Informes	`AllowReportView`	`Boolean`
Permitir vista Presentación	`AllowLayoutView`	`Boolean`
Tipo de imagen	`PictureType`	`Byte`
Imagen	`Picture`	`String`
Mosaico de imágenes	`PictureTiling`	`Boolean`
Distribución de la imagen	`PictureAlignment`	`Byte`
Modo de tamaño de la imagen	`PictureSizeMode`	`Byte`
Ancho	`Width`	`Integer`
Centrado automático	`AutoCenter`	`Boolean`
Ajuste de tamaño automático	`AutoResize`	`Boolean`
Ajustar a la página	`FitToPage`	`Boolean`
Estilo de los bordes	`BorderStyle`	`Byte`

Barras de desplazamiento	ScrollBars	Long
Cuadro de control	ControlBox	Boolean
Botón Cerrar	CloseButton	Boolean
Botones Minimizar Maximizar	MinMaxButtons	Byte
Movible	Moveable	Boolean
Mostrar márgenes de página	ShowPageMargins	Boolean
Línea X	GridX	Integer
Línea Y	GridY	Integer
Diseño a imprimir	LayoutForPrint	Boolean
Mantener junto el grupo	GrpKeepTogether	Byte
Páginas de la imagen	PicturePages	Byte
Encabezado de página	PageHeader	Byte
Pie de página	PageFooter	Byte
Orientación	Orientation	Byte
Origen de la paleta	PaletteSource	String

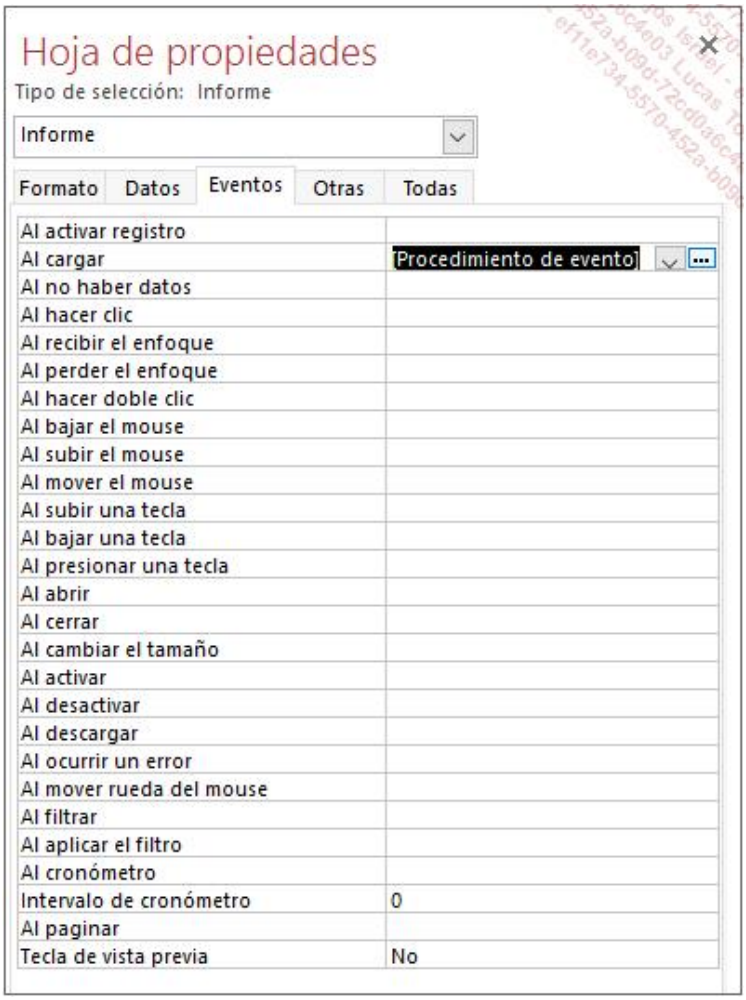
b. Pestaña Datos

En la pestaña **Datos** del panel de navegación **Hoja de propiedades**, pueden aparecer los siguientes elementos:

Etiqueta IHM	Propiedad VBA	Tipo de dato
Origen del registro	RecordSource	String
Filtro	Filter	String
Filtrar al cargar	FilterOnLoad	Boolean
Ordenar por	OrderBy	String
Ordenar por al cargar	OrderByOnLoad	Boolean
Permitir filtros	AllowFilters	Boolean

c. Pestaña Evento

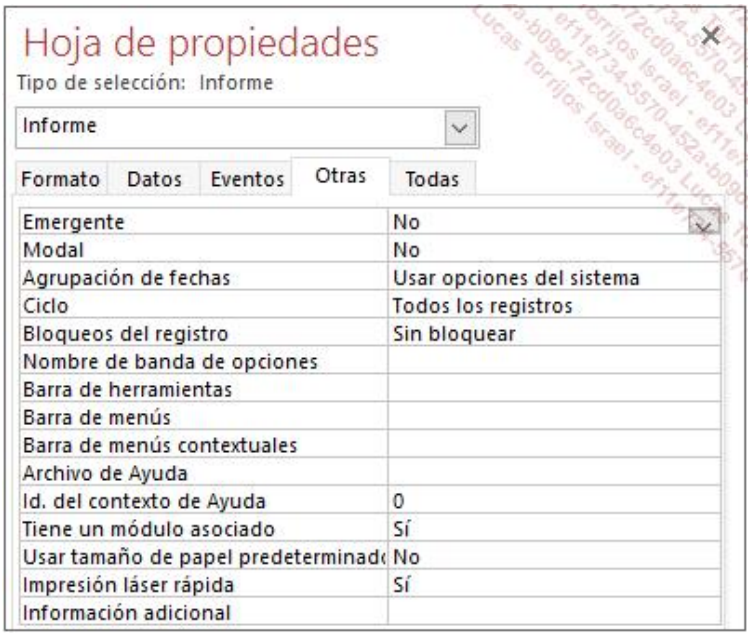
En la pestaña **Evento** del panel de navegación **Hoja de propiedades**, pueden aparecer los siguientes elementos:



Los diferentes eventos se han tratado en el capítulo anterior.

d. Pestaña Otras

En la pestaña **Otras** del panel de navegación **Hoja de propiedades**, pueden aparecer los siguientes elementos:



Etiqueta IHM	Propiedad VBA	Tipo de dato
Emergente	Popup	Boolean
Modal	Modal	Boolean
Agrupación de fechas	DateGrouping	Byte
Ciclo	Cycle	Byte
Bloqueos del registro	RecordLocks	Boolean
Nombre de banda de opciones	RibbonName	String
Barra de herramientas	ToolBar	String
Barra de menús	MenuBar	String
Barra de menús contextuales	ShortcutMenuBar	String
Archivo de Ayuda	HelpFile	String
Id. del contexto de Ayuda	HelpContextID	Long
Tiene un módulo asociado	HasModule	Boolean
Usar tamaño de papel predeterminado		
Impresión láser rápida	FastLaserPrinting	Boolean
Información adicional	Tag	String

3. Otros métodos disponibles en VBA

Algunas propiedades no están disponibles en modo Creación, pero sí lo están en VBA.

a. Propiedades relacionadas con los registros

Propiedad	Descripción
CurrentRecord	Devuelve el registro actual.
Dirty	Permite determinar si el registro actual ha tenido modificaciones.
HasData	Permite determinar si el conjunto de registros origen está vacío.
RecordSourceQualifier	Devuelve o define una cadena de caracteres que indica el nombre del propietario de la fuente del registro SQL Server para el formulario.

b. Propiedades relacionadas con la visualización

Propiedad	Descripción
Moveable	Devuelve un booleano que indica si el informe es movible por el usuario.
Page	Devuelve o define el número de página actual.

c. Propiedades relacionadas con la presentación del informe

Propiedad	Descripción
DrawMode	Representa la manera en que los dibujos (métodos Line, Circle o PSet)

	influyen en los colores del informe.
DrawStyle	Representa el estilo del trazo utilizado con los métodos Line y Circle para dibujar en un informe.
DrawWidth	Representa el grosor del trazo para los métodos Line, Circle y PSet para dibujar en un informe.
FillColor	Representa el color de la trama de los elementos dibujados con los métodos Line y Circle.
FillStyle	Representa el carácter transparente, opaco o con motivos de la trama dibujada con los métodos Line y Circle.
FontBold	Representa la negrita del tipo de letra durante la impresión (método Print) del informe.
Left	Representa la ubicación del informe partiendo de la izquierda.
Painting	Especifica si el informe se debe redibujar.
PaintPalette	Representa la paleta que debe utilizar el informe.
ScaleLeft	Representa la escala de unidad horizontal utilizada para la impresión o la vista previa del informe.
ScaleMode	Representa la unidad de medida de una página utilizada durante la impresión o la vista previa. Los métodos Circle, Line, PSet, Print, TextHeight y TextWidth usan esta información.
ScaleTop	Representa la escala de unidad vertical utilizada para la impresión o la vista previa del informe.
ScaleWidth	Representa el número de unidades utilizadas para la anchura durante la impresión o la vista previa del informe.
Top	Representa la ubicación del informe partiendo de la parte superior.
WindowLeft	Representa la posición de la ventana respecto al borde izquierdo del formulario.
WindowTop	Representa la posición de la ventana respecto al borde superior del formulario.

d. Propiedades relacionadas con la impresión

Propiedad	Descripción
CurrentX	Representa la ubicación horizontal de inicio para la siguiente impresión del informe.
CurrentY	Representa la ubicación vertical de inicio para la siguiente impresión del informe.
MoveLayout	Indica si Access debe pasar a la posición siguiente para la impresión en la página.
NextRecord	Indica si Access debe pasar al registro siguiente para la impresión en la sección.
Pages	Representa el número de páginas total del informe.
PrintCount	Representa el número de veces que la propiedad OnPrint de la sección activa se ejecutará en un informe.
PrintSection	Indica si la sección se debe imprimir o no.
PrtDevMode	Representa la información relativa al modo del periférico de impresión concreto.
PrtDevNames	Representa la información relativa a la impresora indicada.
PrtMip	Representa la información del modo del periférico concreto.
UseDefaultPrinter	Indica si el informe utilizará la impresora por defecto del sistema durante la

	impresión.

e. Propiedades que devuelve un objeto

Propiedad	Descripción
Application	Devuelve el objeto Application (Access).
Module	Devuelve el objeto Module (módulo de código relacionado con el formulario). Se podrá añadir (InsertLines), eliminar (DeleteLines) o modificar (ReplaceLines) líneas de código.
Parent	Devuelve el objeto padre del formulario.
Printer	Devuelve un objeto Printer (impresora por defecto).
Recordset	Devuelve el objeto Recordset que sirve de origen al informe.
Shape	Devuelve un comando Shape que permite la ordenación y la agrupación de los elementos del informe.
Report	Devuelve el informe (caso con un subinforme).

4. Métodos de los informes

También es posible realizar un determinado número de acciones con un informe.

Método	Descripción
Circle	Permite el diseño de un círculo o de una elipse en el informe.
Line	Permite el diseño de una línea o de un rectángulo en el informe.
Move	Permite desplazar el informe a la ubicación concreta (coordenadas).
Print	Permite la escritura de un texto en el informe.
PSet	Permite el cambio de color de un punto a partir de sus coordenadas.
Requery	Permite refrescar los datos del resultado de la consulta origen del informe.
Scale	Permite determinar las escalas de unidad para las coordenadas.
TextHeight	Devuelve la dimensión vertical que tendrá una cadena de caracteres según el tipo de letra actual (dependerá de la propiedad ScaleMode).
TextWidth	Devuelve la dimensión horizontal que tendrá una cadena de caracteres según el tipo de letra actual (dependerá de la propiedad ScaleMode).

El objeto Control

El objeto `Control`, que pertenece a la colección `Controls`, corresponde a un control, `Controls`, que da la lista de todos los controles de un formulario, subformulario, informe o subinforme. El objeto `ActiveControl` apunta al control activo.

1. Sintaxis

Para hacer referencia a un control, hay varias sintaxis posibles, directamente relacionadas con el recorrido de los elementos de la colección `Controls`.

```
FormularioOInforme!NombreControl
   'Ejemplo Me!Txt_Nombre
FormularioOInforme![NombreControl]
   'Ejemplo: F_Animal![Txt_Nombre]
FormularioOInforme("NombreControl")
   'Ejemplo: Me("Txt_Nombre")
FormularioOInforme.NombreControl

    'Ejemplo: Me.Txt_Nombre
```

Podemos también pasar por la colección `Controls` con las sintaxis siguientes:

```
FormularioOInforme.Controls!NombreControl
   'Ejemplo Me.Controls!Txt_Nombre
FormularioOInforme.Controls![NombreControl]
   'Ejemplo: F_Animal.Controls![Txt_Nombre]
FormularioOInforme.Controls("NombreControl")
   'Ejemplo: Me.Controls("Txt_Nombre")
FormularioOInforme.Controls(Indice)
    'Ejemplo: Me.Controls(3)
```

2. Propiedades genéricas comunes de la mayoría de los controles

Cada control tiene propiedades, métodos y eventos que le son propios. Las propiedades que se listan en este capítulo son propiedades genéricas de la mayoría de los controles.

a. Propiedades relacionadas con las dimensiones

Propiedad	Descripción
`Height`	Representa la altura del control.
`Left`	Representa la ubicación del control partiendo de la izquierda.
`Top`	Representa la ubicación del control partiendo de la parte superior.
`Width`	Representa la anchura del control.

b. Propiedades que devuelve un objeto

Propiedad	Descripción
Application	Devuelve el objeto Application (Access).
Hyperlink	Devuelve un hiperenlace hacia un objeto.
Parent	Devuelve el objeto padre del control (formulario, informe, etc.).

c. Otras propiedades

Propiedad	Descripción
Name	Representa el nombre del control.
OldValue	Devuelve el valor de un control cuando este se modifica y es dependiente (alimentado por un campo de una consulta origen).
Value	Representa el valor de un control cuando tiene uno (zona de texto, zona de lista, casilla de selección, etc.).

3. Métodos genéricos comunes de la mayoría de los controles

También es posible realizar un determinado número de acciones sobre un control.

Método	Descripción
Requery	Refresca los datos del resultado de la consulta origen del control.
SetFocus	Da el foco al control.
SizeToFit	Adapta las dimensiones del control a su contenido.
UnDo	Anula las modificaciones añadidas al control.

4. Ejemplo

El siguiente código permite listar los controles que aparecen en el formulario F_Animal abierto:

```
Private Sub Form_Load()
Dim ctl As Control
    For Each ctl In Me.Controls
          Debug.Print ctl.Name & " " & TypeName(ctl)
    Next
End Sub
```

Los controles de Access

Los controles de Access son los elementos que permiten enriquecer una aplicación y vienen a completar las interfaces que el desarrollador pone a disposición del usuario final. Como cualquier aplicación profesional, debe ser intuitiva, y la gran variedad de controles directamente disponibles en el motor de Access, así como la posibilidad de llamar a controles adicionales (llamados controles ActiveX) hace que la realización de formularios e informes sea muy exitosa.

El objetivo de esta sección es permitir al lector tener un inventario de los controles directamente disponibles, asociados a su tipo VBA correspondiente.

Para añadir nuevos controles a un formulario o a un informe, puede abrir el formulario o el informe en modo Creación. Una vez que se activa este modo, aparece en la cinta de opciones una pestaña **Diseño** en la que podemos ver el grupo **Controles**, que se muestra a continuación:

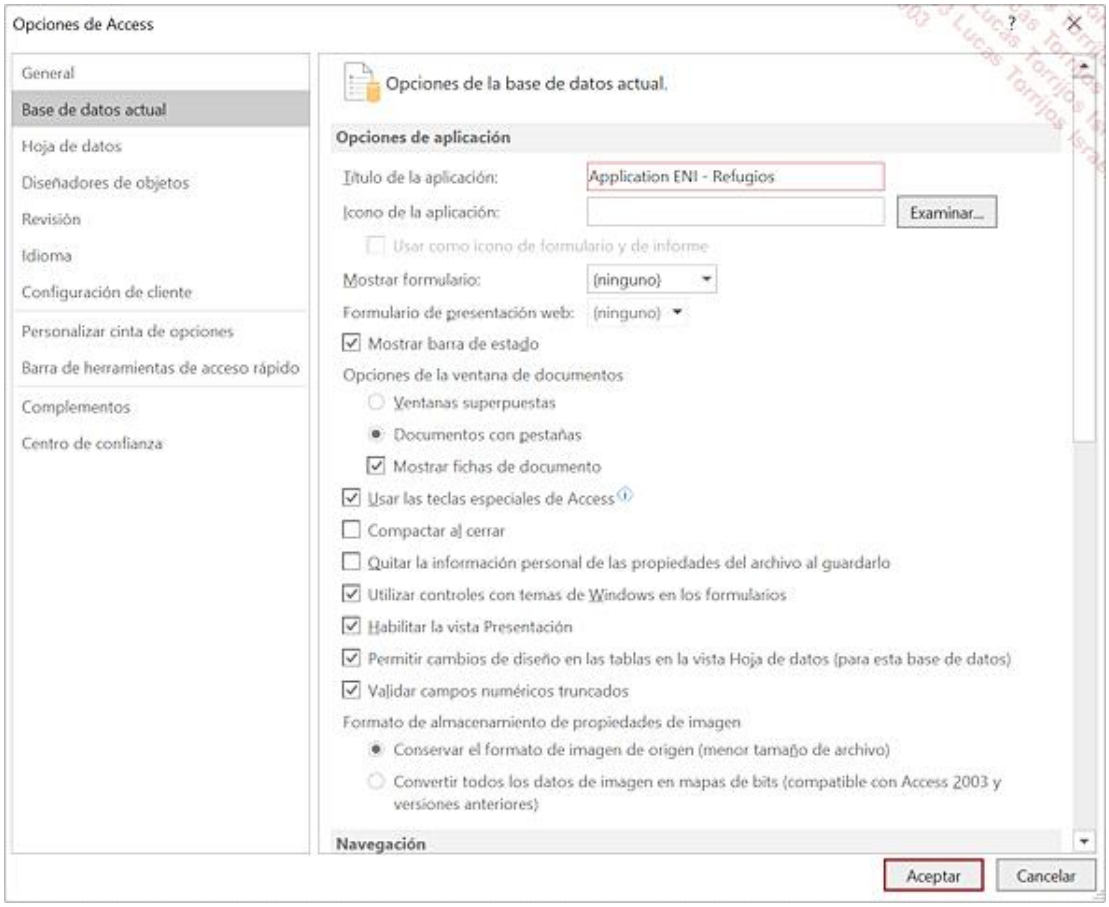
A continuación se muestra la lista de los controles mostrados en el orden de lectura (de izquierda a derecha, de arriba hacia abajo):

Etiqueta del control	Objeto VBA equivalente
Zona de texto	`TextBox`
Etiqueta	`Label`
Botón de comando	`CommandButton`
Control de pestaña	`TabControl`
Enlace hipertexto	`Hyperlink`
Grupo de opciones	`OptionGroup`
Salto de página	`PageBreak`
Zona de lista desplegable	`ComboBox`
Gráfico	`Chart`
Trazo	`Line`
Botón de cambio	`ToggleButton`
Zona de lista	`ListBox`
Rectángulo	`Rectangle`
Casilla de selección	`CheckBox`

Marco del objeto independiente	UnboundObjectFrame
Elemento adjunto	Attachment
Casilla de opción	OptionButton
Subformulario/Subinforme	SubForm/SubReport
Marco del objeto dependiente	BoundObjectFrame
Imagen	Image

Personalizar las opciones de Access

Para hacer que la aplicación tenga el mayor éxito posible después de su inicio, Microsoft Access permite gestionar un gran número de opciones de inicio. Puede acceder a estas opciones en la pestaña **Archivo - Opciones**. Al seleccionar la pestaña **Base de datos actual** en la pestaña de navegación de la izquierda, se muestra la siguiente interfaz.



Algunas de estas opciones solo se pueden manipular en la interfaz, mientras que otras también en VBA.

Se podrán utilizar las propiedades de la base activa (`CurrentDb.Properties`), pero también es posible controlar las opciones con los métodos `SetOption` (nombre de la opción seguida del valor que tomará la opción) y `GetOption` (nombre de la opción) del objeto `Application`.

Las propiedades y métodos tendrán la siguiente sintaxis general:

```
CurrentDb.Properties("NombrePropiedad").Value = ValorPropiedad
Application.SetOption "NombreOption", ValorOption
X = Application.GetOption ("NombreOption")
```

Si la propiedad no está disponible con `CurrentDb.Properties` (error '3270': 'Propiedad no encontrada'), debe crearla con `CurrentDb.CreateProperty` o usando `Application.SetOption`.

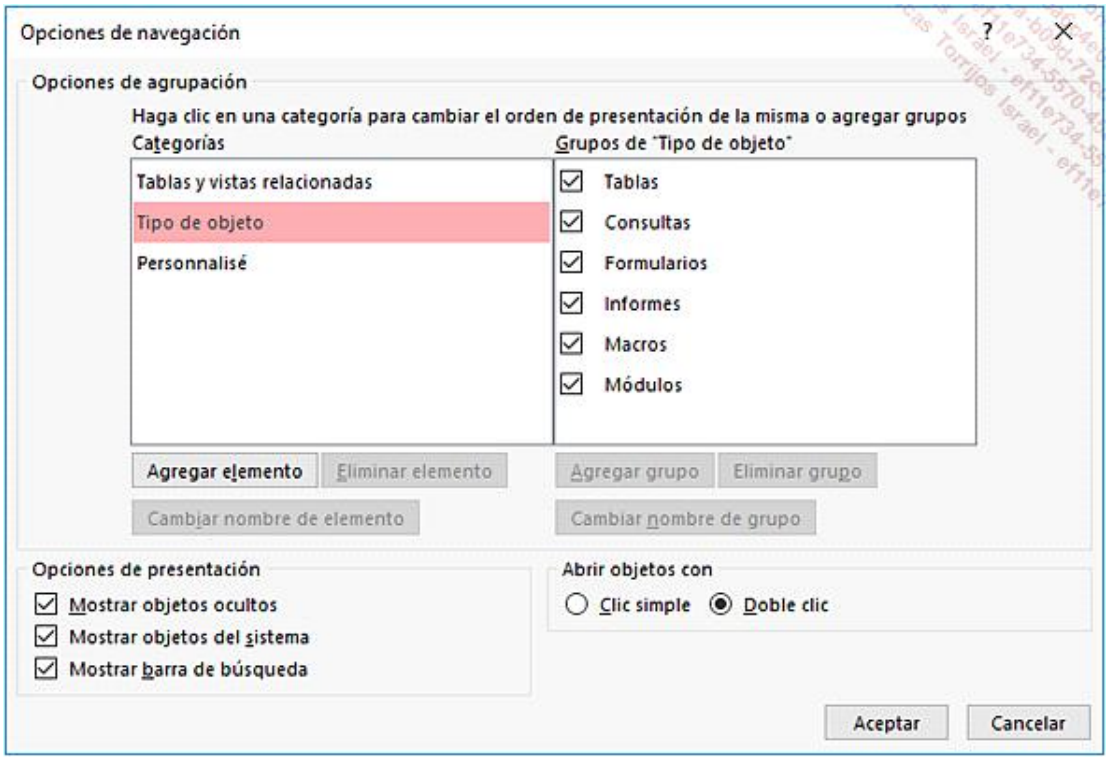
1. Opciones de la aplicación

A cada opción de la interfaz le puede corresponder una propiedad VBA o una opción y un tipo de dato.

Etiqueta en la interfaz Opciones de Access	Propiedad u opción correspondiente	Tipo de dato
Título de la aplicación	AppTitle	String
Icono de la aplicación	AppIcon	String
Usar como icono de formulario y de informe	UseAppIconForFrmRpt	Boolean
Mostrar formulario	StartUpForm	String
Formulario de presentación web (solo funciona para una aplicación web)		String
Mostrar barra de estado	StartUpShowStatusBar	Boolean
Opciones de la ventana de documentos	UseMDIMode	Byte
Mostrar fichas de documento	ShowDocumentoTabs	Boolean
Usar las teclas especiales de Access	AllowSpecialKeys	Boolean
Compactar al cerrar	Auto Compact	Boolean
Quitar la información personal [...] al guardarlo	Remove Personal Information	Boolean
Utilizar controles con temas de Windows en los formularios	Themed Form Controls	Boolean
Habilitar la vista Presentación	DesignWithData	Boolean
Permitir cambios de diseño [...] en la vista Hoja de datos	AllowDatasheetSchema	Boolean
Validar campos numéricos truncados	CheckTruncated NumFields	Boolean
Formato de almacenamiento de propiedades de imagen	Picture Property Storage Format	Byte

2. Opciones de navegación

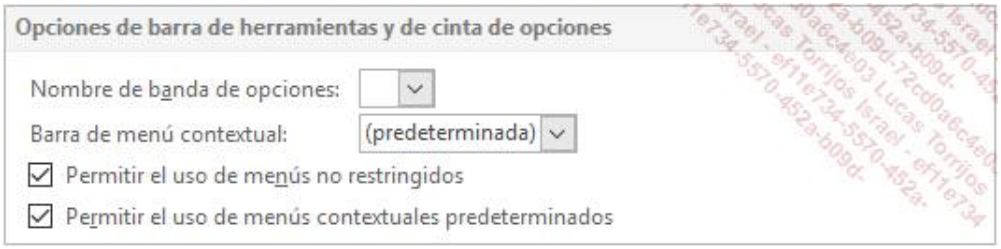
Haciendo clic en el botón **Opciones de navegación** en la interfaz **Opciones de Access**, aparece la siguiente interfaz:



A cada opción de la interfaz le puede corresponder una propiedad VBA o una opción y un tipo de dato.

Etiqueta en la interfaz Opciones de Access	Propiedad u opción correspondiente	Tipo de dato
Ver la pestaña de navegación	`StartUpShowDBWindow`	`Boolean`
Mostrar objetos ocultos	`Show Hidden Objects`	`Boolean`
Mostrar objetos del sistema	`Show System Objects`	`Boolean`
Mostrar barra de búsqueda	`Show Navigation Pane Search Bar`	`Boolean`
Abrir objetos con ...	`Database Explorer Click Behavior`	`Byte`

3. Opciones de la barra de herramientas y de la cinta de opciones

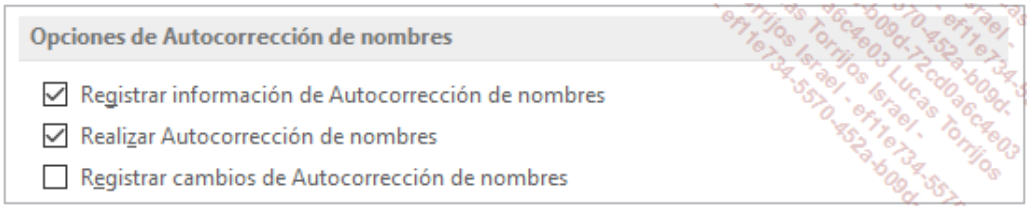


A cada opción de la interfaz le puede corresponder una propiedad VBA o una opción y un tipo de dato.

Etiqueta en la interfaz Opciones de Access	Propiedad u opción correspondiente	Tipo de dato
Nombre de banda de opciones	`RibbonName`	`String`
Barra de menú contextual	`StartupShortcutMenuBar`	`String`

Permitir el uso de menús no restringidos	AllowFullMenus	Boolean
Permitir el uso de menús contextuales predeterminados	AllowShortcutMenus	Boolean

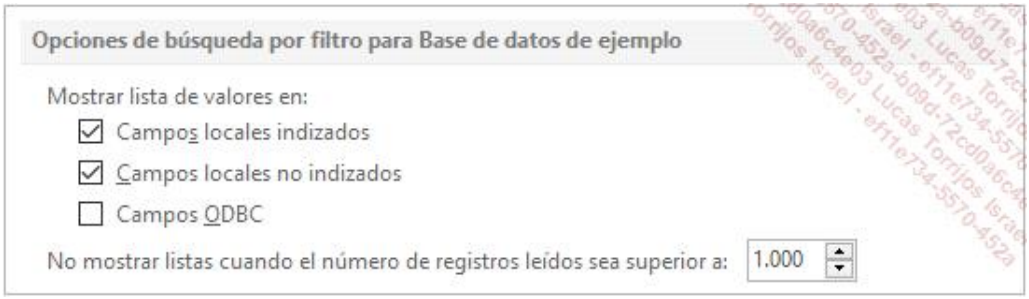
4. Opciones de corrección automática de nombre



A cada opción en la interfaz le puede corresponder una propiedad VBA o una opción y un tipo de datos.

Etiqueta en la interfaz Opciones de Access	Propiedad u opción correspondiente	Tipo de dato
Registrar información de Autocorrección de nombres	Track Name AutoCorrect Info	Boolean
Realizar Autocorrección de nombres	Perform Name AutoCorrect	Boolean
Registrar cambios de Autocorrección de nombres	Log Name AutoCorrect Changes	Boolean

5. Opciones de búsqueda



A cada opción en la interfaz le puede corresponder una propiedad VBA o una opción y un tipo de dato.

Etiqueta en la interfaz Opciones de Access	Propiedad u opción correspondiente	Tipo de dato
Campos locales indizados	Show Values in Indexed	Boolean
Campos locales no indizados	Show Values in No-Indexed	Boolean
Campos ODBC	Show Values in Remote	Boolean
No mostrar listas...	Show Values Limit	Long

6. Opciones de cacheado

A cada opción en la interfaz le puede corresponder una propiedad VBA o una opción y un tipo de dato.

Etiqueta en la interfaz Opciones de Access	Propiedad u opción correspondiente	Tipo de dato
Usar el formato de caché...	`Use Microsoft Access 2010 compatible cache`	`Boolean`
Borrar memoria caché al cerrar	`Clear Cache on Close`	`Boolean`
Nunca copiar en caché	`Never Cache`	`Boolean`

7. Otras opciones

Hay un gran número de otras opciones (**General**, **Hoja de datos**, **Diseñador de objetos**, **Comprobación**, **Idioma** y **Argumentos del cliente**), que permiten ir más lejos en la personalización de las aplicaciones.

8. Ejemplo

Para forzar la compactación durante el cierre, la eliminación de información personal y cachear la barra de estado durante la ejecución de la aplicación, se puede ejecutar el siguiente código:

```
Function AperturaPerso()
'Compactar durante el cierre
    Application.SetOption "Auto Compact", True
'Eliminar la información personal
    Application.SetOption "Remove Personal Information", False
'Cachear la barra de estado
    Application.SetOption "Show Status Bar", False
End Function
```

Personalizar las cintas de opciones

Para no limitarse a la única cinta de opciones por defecto que propone Access durante el uso de la aplicación, además de conseguir el objetivo de hacer algunas acciones lo más intuitivas posible, puede ser importante implementar menús dedicados y, por tanto, nuevas cintas de opciones. El objetivo de esta sección es explicar la cinta de opciones de Access original: cómo se forma, cómo interactuar con ella y para terminar cómo hacerla dinámica en su aplicación, personalizándola.

1. La cinta de opciones de Access

La cinta de opciones de Access es la interfaz que ha sustituido al menú de las versiones 2003 y anteriores. Esta cinta de opciones permite visualizar u ocultar las pestañas según el lugar donde está el usuario. Cada pestaña contendrá un determinado número de grupos en los que estarán accesibles las funcionalidades, haciendo clic en imágenes, iconos, zonas de listas desplegables, etc.

a. Las pestañas en Access

Las pestañas que se muestran de base son las siguientes:

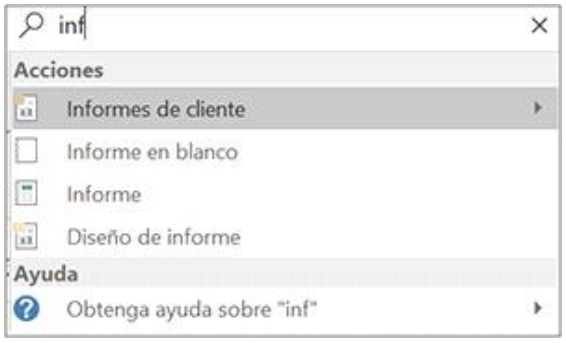
Pestaña	Principales grupos	Descripción
Archivo	**Información, Nuevo, Abrir, Guardar, Imprimir, Opciones**	Permite gestionar los archivos de base de datos y acceder a las opciones de Access.
Inicio	**Vistas, Portapapeles, Ordenar y filtrar, Registros, Buscar, Formato de texto**	Permite gestionar el modo de visualización, copiar/pegar elementos desde el portapapeles, gestionar los registros del objeto que se está usando, buscar y formatear el texto.
Crear	**Plantillas, Tablas, Consultas, Formularios, Informes, Macros y código**	Permite la creación de nuevos objetos de Access, tanto manualmente como pasando por el Asistente de Access.
Datos externos	**Importar y vincular, Exportar**	Permite importar y exportar datos desde/hacia diversas fuentes soportadas por Microsoft Access 2019.
Herramientas de base de datos	**Herramientas, Macro, Relaciones, Analizar, Mover datos, Complementos**	Permite compactar la base de datos, acceder a Visual Basic Editor, gestionar las relaciones entre tablas de la base de datos, analizar la estructura y el rendimiento de la base de datos, crear una estructura front-end/back-end de la base de datos e incluso activar las librerías de Complementos.
Ayuda	**Ayuda**	Permite mostrar la ayuda, dar su opinión y conocer las no-vedades (Office 365)

b. ¿Qué desea hacer ?

Disponible desde la versión 2016 de Access, hay disponible un motor de ayuda en las pestañas. Es suficiente con indicar las palabras clave para las que desea obtener la ayuda.

Cuando se entra en la zona de texto, Microsoft Access propone entradas por defecto, mostrándolas como un menú desplegable.

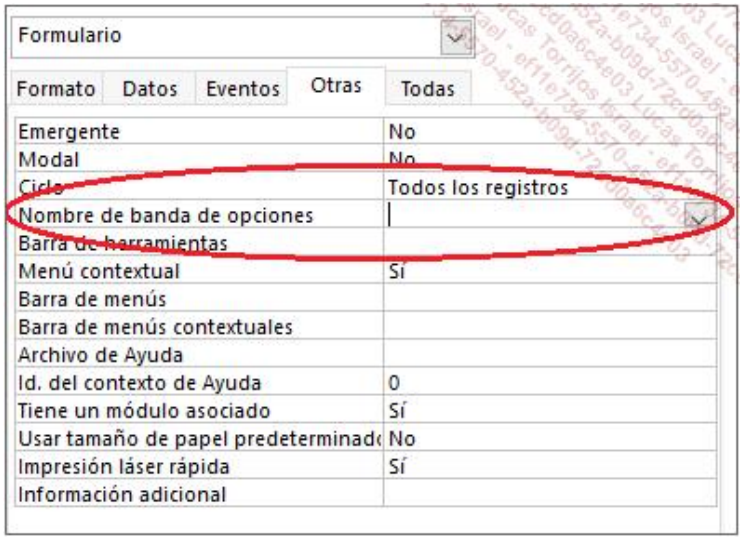
Cuando el usuario escribe una o varias palabras, Microsoft Access le propone directamente un acceso a las funcionalidades disponibles. Por ejemplo, escribiendo la palabra **Inf**, Access 2019 propone lo siguiente:



Esta herramienta es muy práctica y le lleva a la ayuda en línea clásica si hace clic en **Obtenga ayuda**.

c. Asociación de la cinta de opciones

En el modo Creación de formularios e informes, en las propiedades **Otras** (ver los capítulos respectivos), podemos ver una propiedad llamada **Nombre de banda de opciones**.



Por tanto, es posible asociar a un objeto de Access una cinta de opciones que se pueda personalizar. El capítulo Controlar otras aplicaciones de Office 2019, va a permitir entender cómo se estructura una cinta de opciones.

2. La estructura XML de la cinta de opciones

En las versiones anteriores a Microsoft Access 2007, solo podía crear sus propias barras de menú en VBA. Desde esta versión y con la aparición de la cinta de opciones, el lenguaje que se utiliza para modelizar dicha cinta de opciones es XML. XML es la abreviatura de e*Xtensible Markup Language*, y sirve para almacenar datos. Utiliza, como su padre el HTML (*Hyper Text Markup Language*), etiquetas anidadas en cascada. Las etiquetas se delimitan por los caracteres "<" y ">", entre los que figura el nombre de la etiqueta. Pueden aparecer emparejadas con otras etiquetas, que les permiten "cerrarse" y que contienen el mismo nombre de etiqueta, precedido del carácter "/", como en el siguiente ejemplo:

```
<ribbon></ribbon>
```

En la redacción del código XML, es importante que las cintas de opciones respeten la diferencia entre mayúsculas y minúsculas. En caso contrario, el código se considerará inválido y se provocará un error.

a. Estructura elemental de la cinta de opciones

Archivo de ejemplo

En el ámbito de nuestro análisis de la estructura XML, el archivo de ejemplo será el siguiente:

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
    <ribbon startFromScratch="false">
          <tabs>
                <tab id="PestaniaPerso" label="Pestaña personalizada"
visible="true">
                      <group id="G_1" label="Animal">
                            <button id="B_Agregar"
label="Nuevo animal" onAction="Agregar animal"
imageMso="CreateReportFromWizard" />
                      </group>
                </tab>
       </tabs>
     </ribbon>
</customUI>
```

El aspecto visual de esta cinta de opciones, una vez cargada en Microsoft Access (esta operación se verá más adelante), es el siguiente:

Vamos a analizar cada parte de esta estructura.

customUI

```
CustomUI
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/
customui">
</customUI>
```

La etiqueta `customUI` es la raíz invariable que permite la inserción de una cinta de opciones customUI significa *Custom User Interface*, lo que se traduce por interface de usuario personalizada.

Observe que también es posible tener una estructura con una versión 2010 de la cinta de opciones, compatible con Microsoft Access 2019:

```
<customUI
xmlns="http://schemas.microsoft.com/office/2009/07/customui"
onLoad="RibbonLoad">
</customUI>
<customUI
xmlns="http://schemas.microsoft.com/office/2006/01/customui">
    <ribbon startFromScratch="false">
    </ribbon>
</customUI>
```

La etiqueta `ribbon` permite describir la cinta de opciones personalizada.

El atributo `startFromScratch` permite especificar si las otras pestañas predefinidas en Microsoft Access estarán ocultas o no.

tabs y tab

```
<customUI
xmlns="http://schemas.microsoft.com/office/2006/01/customui">
    <ribbon startFromScratch="false">
           <tabs>
                 <tab id="PestaniaPerso" label="Pestaña
Personalizada" visible="true"
                 </tab>
```

```
      </tabs>
    </ribbon>
</customUI>
```

Dentro de la cinta de opciones, se define una colección de pestañas entre las etiquetas `<tabs>` y `</tabs>`.


Observe la administración común de la colección `tabs` de los objetos `tab`, con una sintaxis idéntica a la de VBA.

group

```
<customUI
xmlns="http://schemas.microsoft.com/office/2006/01/customui">
    <ribbon startFromScratch="false">
          <tabs>
                <tab id="PestaniaPerso" label="Pestaña
Personalizada" visible="true">
                      <group id="G_1" label="Animales">
                      </group>
                </tab>
          </tabs>
    </ribbon>
</customUI>
```

Dentro de cada pestaña (`tab`), se pueden definir grupos de controles.

Los atributos

Para las pestañas y los grupos, podemos ver los siguientes atributos:

- `id`: permite dar un identificador al objeto.
- `label`: permite indicar el título mostrado.
- `visible`: permite indicar si la pestaña es visible.

b. Los controles de la cinta de opciones

Dentro de las pestañas y de los grupos, pueden aparecer varios tipos de controles, similares a los que podemos añadir en los formularios e informes, en la aplicación Microsoft Access.

Las etiquetas (labelControl)

El `labelControl` es una zona de etiquetas en la que no hay prevista ninguna acción particular. Sirve para visualizar la información o para determinar un encabezado de grupo, por ejemplo.

Los botones (button)

El `button` es un botón que se podrá pulsar. Podemos utilizar botones predefinidos en Microsoft Access, así como crear nuestros propios botones personalizados.

Los botones de activación (toggleButton)

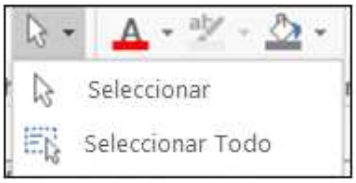
El `toggleButton` es un botón de activación que cambia de aspecto cuando se pulsa. El botón **Filtro** de la pestaña **Inicio - Ordenar y filtrar** es un ejemplo de botón de activación.

El grupo de botones (buttonGroup)

El `buttonGroup` sirve para agrupar varios botones en un mismo conjunto.

Las listas de botones en un menú desplegable (splitButton)

El `splitButton` permite visualizar en un menú desplegable una lista de botones.



Las zonas de texto (editBox)

El `editBox` sirve de zona de texto en la que el usuario puede escribir lo que desea.

Las zonas de listas desplegables (dropDown)

El `dropDown` permite visualizar un menú desplegable que puede contener botones.

Las casillas de selección (checkBox)

La `checkBox` permite visualizar una casilla de selección.

Los menús (menu)

El `menu` permite crear una barra de menús, que puede contener botones o sub-menús anidados en forma jerárquica.

Los menús dinámicos (dynamicMenu)

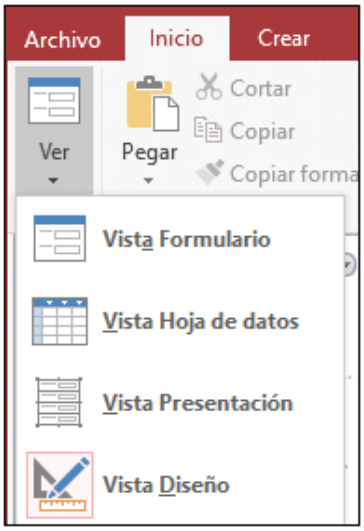
El `dynamicMenu` muestra una barra de menú, cuyo contenido se puede crear y modificar dinámicamente en VBA, al contrario que el resto de los controles con lista de opciones (`comboBox`, `menu`, `dropDown`).

Los títulos de barra de menú (menuSeparator)

El `menuSeparator` sirve para crear un título en las barras de menú. Esto permite agrupar elementos en las listas.

Las galerías (gallery)

La `gallery` es una tabla que incluye varios tipos de controles.

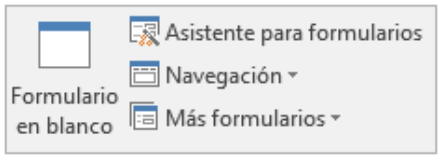


Los botones de ejecución del cuadro de diálogo (dialogBoxLauncher)

El `dialogBoxLauncher` es el botón en la parte inferior derecha de los grupos que permite abrir un cuadro de diálogo.

Las cajas (box)

El `box` permite agregar contenidos en grupos vertical u horizontalmente.



Los comandos (command)

El `command` permite volver a asignar comandos predefinidos en Access a las instrucciones personalizadas.

c. Los atributos de los controles

Los controles se pueden definir y configurar con sus propiedades, también llamadas atributos, para una cinta de opciones, además de aquellas vistas en la cinta de opciones de ejemplo (`id`, `visible`, `label`).

Atributo	Descripción
description	Especifica un texto de descripción que se mostrará en los menús cuando el tamaño del control se defina a large.
enabled	Especifica si el control está disponible o no.
idMso	Especifica el código Mso predefinido que permite asignar a un control una funcionalidad ya implementada por Microsoft Office.
image	Especifica la ubicación de una imagen para el control.
imageMso	Especifica un código predefinido de imagen, que permite asignar a un control una imagen ya implementada por Microsoft Office.
insertAfterMso	Especifica la ubicación del control, situado después del control de Microsoft Office predefinido indicado.
insertBeforeMso	Especifica la ubicación del control, situado antes del control de Microsoft Office predefinido indicado.
itemSize	Especifica el tamaño de los elementos que aparecen en un menú.
keyTip	Especifica un acceso directo de teclado para acceder al control ([Alt] + tecla o combinación de teclas).
maxLength	Especifica el número máximo de caracteres en un control de entrada de datos (editBox, comboBox).
rows	Especifica el número de líneas en una galería (gallery).
screentip	Especifica la tooltip del control.
showImage	Indica si la imagen del control se mostrará o no.
showItemImage	Indica si la imagen de un elemento del control (comboBox, dropDown, gallery) se mostrará o no.
showItemLabel	Indica si la etiqueta de un elemento del control (comboBox, dropDown, gallery) se mostrará o no.
showLabel	Indica si la etiqueta del control se mostrará o no.
size	Especifica el tamaño del control.
sizeString	Permite ajustar la anchura del control (comboBox, editBox, gallery).
supertip	Especifica el tooltip complementario del control, adicional a la información de la screentip.
tag	Especifica información complementaria del control.
title	Especifica un título (menuSeparator).

d. Los separadores

El control `separator` permite añadir barras verticales entre dos grupos de controles. No hay ninguna interacción con el usuario y su único atributo es su identificador (`id`).

e. Los comentarios

Para añadir comentarios dentro del código XML, se puede utilizar el siguiente comando:

```
<!-- -->
```

Como por ejemplo:

```
<!—esto es un comentario -->
```

Los comentarios no se interpretan en el código como una instrucción.

3. Los eventos

Una vez que se definen los controles en la cinta de opciones, es necesario poder detectar si el usuario interactúa con ellos.

a. Los procedimientos Callback

Se utilizan funciones de llamada de VBA, programadas en los módulos estándares, que se desencadenan como consecuencia de algunos eventos que se producen sobre los controles. Estos procedimientos se llaman procedimientos de Callback. Por tanto, será posible desencadenar código a partir de la cinta de opciones, así como modificar los atributos de los controles dinámicamente con las propiedades `getXXX`.

b. Evento OnLoad de la cinta de opciones

El primer evento que se desencadena durante la carga de cinta de opciones es la función `onLoad`. Esta función de llamada se sitúa a nivel del objeto `customUI` del código XML de la cinta de opciones. Solo se desencadena una única vez, después de la apertura del formulario o del informe asociado a la cinta de opciones. La variable objeto que almacena la cinta de opciones será de tipo `IRibbonUI`.

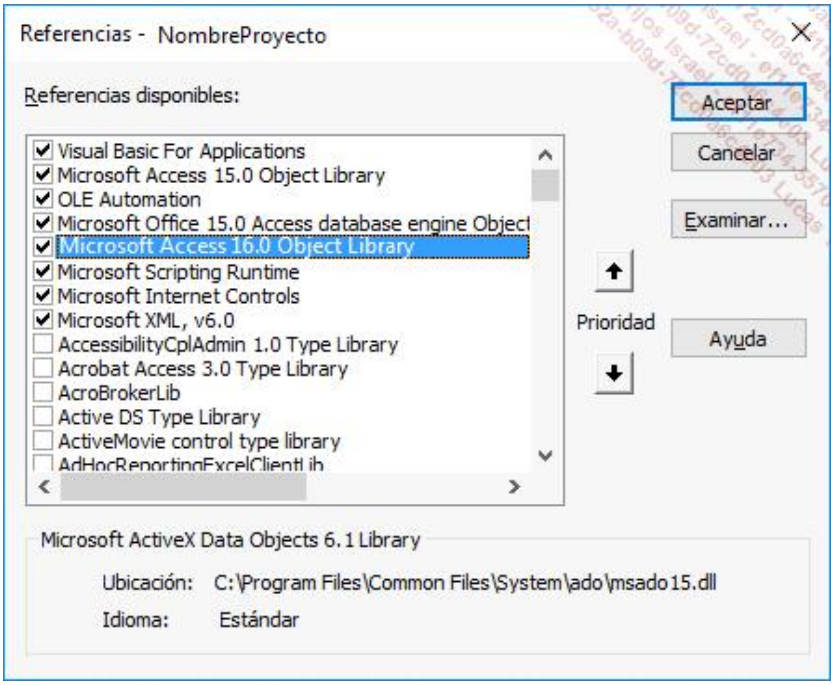
En caso de que la cinta de opciones contenga el siguiente código XML:

```
<customUI
xmlns="http://schemas.microsoft.com/office/2006/01/customui"
onLoad="DuranteCargaCinta">
    <ribbon>
    </ribbon>
</customUI>
```

El procedimiento `DuranteCargaCinta` se ejecutará durante la carga de la cinta de opciones. Por tanto, conviene definir en un módulo estándar el siguiente procedimiento:

```
Sub DuranteCargaCinta(cintaOpciones As IribbonUI)
    Debug.Print "Cinta de opciones cargada"
End Sub
```

Para poder manipular una variable de tipo `IRibbonUI`, es necesario añadir en el proyecto la referencia **Microsoft Access 16.0 Object Library**, cuyo archivo está en la siguiente ubicación: C:\Program Files\Common Files\System\ado\msado15.dll



c. Eventos comunes a todos los controles

El objeto de control en el que tiene lugar el evento se representa por la variable `control` de tipo `IRibbonControl`.

Atributo	Procedimiento	Descripción
`getDescription`	`sub GetDescription (control As IRibbonControl, ByRef description)`	Actúa sobre el atributo `description`
`getEnabled`	`sub GetEnabled (control As IRibbonControl, ByRef enabled)`	Actúa sobre el atributo `enabled`.
`getImage`	`sub GetImage (control As IRibbonControl, ByRef image)`	Actúa sobre el atributo `image`.
`getLabel`	`sub GetLabel (control As IRibbonControl, ByRef label)`	Actúa sobre el atributo `label`.
`getScreentip`	`sub GetScreentip (control As IRibbonControl, ByRef screentip)`	Actúa sobre el atributo `screentip`.
`getSize`	`sub GetSize (control As IRibbonControl, ByRef size)`	Actúa sobre el atributo `size`.
`getSupertip`	`sub GetSupertip (control As IRibbonControl, ByRef screentip)`	Actúa sobre el atributo `supertip`.
`getVisible`	`sub GetVisible (control As IRibbonControl, ByRef visible)`	Actúa sobre el atributo `visible`.

Observe que el paso por referencia `ByRef` permite actualizar si es necesario el atributo del control.

d. Eventos concretos del botón

Atributo	Procedimiento	Descripción

onAction	sub OnAction (control As IRibbonControl)	Se desencadena al hacer clic en el botón.

e. Eventos concretos de la casilla de selección

Atributo	Procedimiento	Descripción
getPressed	sub GetPressed (control As IRibbonControl, ByRef return)	Actúa sobre el atributo marcado.
onAction	sub OnAction (control As IRibbonControl, pressed As Boolean)	Se desencadena durante el clic en la casilla de selección.

f. Eventos concretos de la zona de lista desplegable

Atributo	Procedimiento	Descripción
getItemCount	sub GetItemCount (control As IRibbon Control, ByRef count)	Actúa sobre el número de elementos en la lista desplegable.
getItemImage	sub GetItemImage (control As IRibbonControl, index As Integer, ByRef image)	Actúa sobre la imagen de un elemento de la lista desplegable.
getItemLabel	sub GetItemLabel (control As IRibbon Control, index As Integer, ByRef label)	Actúa sobre la etiqueta de un elemento de la lista desplegable.
getText	sub GetText (control As IRibbonControl, ByRef text)	Actúa sobre el valor por defecto de la lista desplegable.
onChange	sub OnChange (control As IRibbonControl, text As String)	Actúa durante un cambio de valor de la lista desplegable.

g. Eventos concretos de menú desplegable

Atributo	Procedimiento	Descripción
getItemCount	sub GetItemCount (control As IRibbonControl, ByRef count)	Actúa sobre el número de elementos en el menú desplegable.
getItemLabel	sub GetItemLabel (control As IRibbonControl, index As Integer, ByRef label)	Actúa sobre la etiqueta de un elemento del menú desplegable.
getSelected ItemIndex	sub GetSelected ItemIndex (control As IRibbonControl, ByRef index)	Actúa sobre el valor por defecto del menú desplegable (a partir de su índice).
getSelected ItemID	sub GetSelected ItemID (control As IRibbonControl,ByRef index)	Actúa sobre el valor por defecto del menú desplegable (a partir de su id).
onAction	sub OnAction (control As IRibbonControl, selectedId As String, selectedIndex As Integer)	Se desencadena durante el clic en el menú desplegable.
getItemImage	sub GetItemImage (control As	Actúa sobre la imagen de un

	IRibbonControl, index As Integer, ByRef image)	elemento del menú desplegable.
getItemID	sub GetItemID (control As IRibbonControl, index As Integer, ByRef id)	Actúa sobre el id de un elemento del menú desplegable.

h. Eventos concretos de la zona de texto

Atributo	Procedimiento	Descripción
getText	sub GetText (control As IRibbonControl, ByRef text)	Actúa sobre el valor por defecto de la zona de texto.
onChange	sub OnChange (control As IRibbonControl, text As String)	Actúa durante un cambio de valor de la zona de texto.

i. Eventos concretos de la galería

Atributo	Procedimiento	Descripción
getItemLabel	sub GetItemLabel (control As IRibbonControl, index As Integer, ByRef label)	Actúa sobre la etiqueta de un elemento de la galería.
onAction	sub OnAction (control As IRibbonControl, selectedId As String, selectedIndex As Integer)	Se desencadena durante el clic en la galería.
getSelected ItemIndex	sub GetSelectedItemIndex (control As IRibbonControl, ByRef index)	Actúa sobre el valor por defecto de la galería (a partir de su índice).
getSelected ItemID	sub (control As IRibbonControl, ByRef index)	Actúa sobre el valor por defecto de la galería (a partir de su id).
getItemWidth	sub getItemWidth (control As IRibbonControl, ByRef width)	Actúa sobre la anchura (en píxeles) de un elemento de la galería.
getItemHeight	sub getItemHeight (controlAs IRibbonControl, ByRef height)	Actúa sobre la altura (en píxeles) de un elemento de la galería.
getItemImage	sub (control As IRibbonControl, index As Integer, ByRef image)	Actúa sobre la imagen de un elemento de la galería.
getItemCount	sub GetItemCount (control As IRibbonControl, ByRef count)	Actúa sobre el número de elementos de la galería.
getItemID	sub GetItemID (control As IRibbonControl, index As Integer, ByRef id)	Actúa sobre el id de un elemento de la galería.

j. Eventos concretos del botón cambiar

Atributo	Procedimiento	Descripción
getPressed	sub GetPressed (control As IRibbonControl, ByRef return)	Actúa sobre el atributo pulsación del botón.

onAction	sub OnAction (control As IRibbonControl, pressed As Boolean)	Se desencadena durante el clic en el botón.

4. Gestión dinámica de la cinta de opciones

Para actualizar la cinta de opciones en función de las acciones que efectúa el usuario, podemos forzar su refresco con el método `Invalidate` del objeto `IRibbonUI`.

a. Código durante la carga

Durante su carga, es necesario almacenarlo en una variable de tipo `IRibbonUI`, que se utilizará a partir de ese momento para actualizar la cinta de opciones.

```
Public MiCintaOpciones As IribbonUI
'Esto se desencadena durante la carga de la cinta de opciones
personalizada.
Sub DuranteCargaCinta(cintaOpciones As IribbonUI)
    Debug.Print "Cinta de opciones cargada"
    Set MiCintaOpciones = cintaOpciones
End Sub
```

b. Gestión posterior de las actualizaciones

Una vez que la cinta de opciones se puede manipular con la variable `MiCintaOpciones`, podemos llamar a los métodos `Invalidate` e `InvalidateControl`.

La sintaxis general de cada uno de los dos métodos es la siguiente:

```
MiCintaOpciones.Invalidate
MiCintaOpciones.InvalidateControl "NombreDelControlARefrescar"
```

En función de los eventos, y teniendo en cuenta los principales atributos `XXX` por las funciones de llamada callback `getXXX`, se podría actualizar el conjunto de la cinta de opciones.

5. Carga de una cinta de opciones a partir de un archivo XML

a. La escritura del archivo XML

El archivo XML original se puede editar utilizando cualquier editor de texto. Algunos están dedicados específicamente a la redacción del contenido XML, como NotePad++, un editor genérico que integra la coloración sintáctica. Una vez que el archivo está disponible, la siguiente etapa es cargarlo en Microsoft Access.

b. El método LoadCustomUI

El método `LoadCustomUI` del objeto `Application` permite cargar una cinta de opciones a partir de un contenido XML. La sintaxis general de este método es la siguiente:

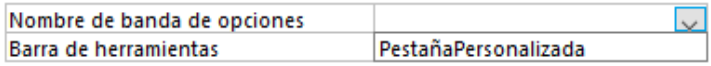
```
Application.LoadCustomUI "NombreDeCintaOpciones", "contenidoXML"
```

Desde ese momento es posible leer el contenido XML de un archivo XML. Se podría utilizar el objeto `FileSystemObject` de la librería **Microsoft Scripting Runtime**.

El siguiente código va a leer el contenido XML del archivo CintaOpciones.xml, situado en la raíz de la carpeta en la que se encuentra la base de datos.

```
Public Function CargarCintaOpciones()
    Dim strContenidoXML As String
    Dim Fso As New FileSystemObject
    Dim Ftxt As TextStream
    'Abre el archivo XML en memoria
    Set Ftxt = Fso.OpenTextFile(CurrentProject.Path & _
           "\CintaOpciones.XML", ForReading)
    'Lee la integridad de su contenido
    strContenidoXML = Ftxt.ReadAll
    'Carga la cinta de opciones correspondiente
    Application.LoadCustomUI "PestañaPersonalizada", strContenidoXML
End Function
```

Una vez que se realiza esta etapa de carga, aparecerá en la propiedad **Nombre de banda de opciones** la cinta de opciones cargada.



c. Inconveniente de la carga

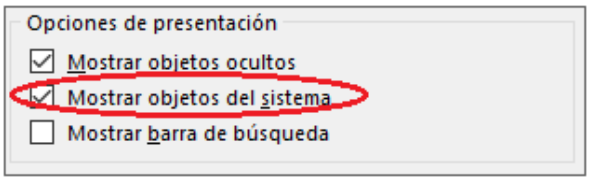
El método `LoadCustomUI` permite cargar una cinta de opciones, pero se descarga durante el cierre de la base de datos. Por tanto, hay que pasar por otro método si se desea hacer permanentes las cintas de opciones personalizadas.

6. Uso de la tabla de sistema USysRibbons

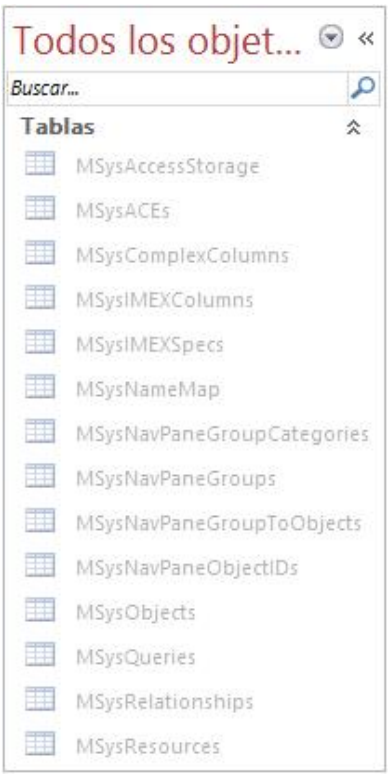
Para evitar la carga y descarga de las diferentes cintas de opciones personalizadas, es posible almacenar sus contenidos XML en una tabla de sistema llamada `USysRibbons`. Para conseguirlo, a continuación se muestran las etapas que es necesario seguir.

a. Ver las tablas de sistema

Para poder visualizar las tablas de sistema, debe ir a las opciones de navegación (**Archivo - Opción - Base de datos activa - Opciones de navegación**) y marcar la casilla de selección **Mostrar objetos del sistema** en las opciones de visualización.



Desde ese momento, las tablas de sistema están visibles en la pestaña de navegación de la interfaz de Microsoft Access.



b. Creación de la tabla

Para crear la tabla, vaya a la pestaña **Crear** y seleccione **Diseño de tabla**.

Añada los siguientes campos:

Nombre del campo	Tipo de campo	Tamaño de campo
ID (clave primaria)	Autonumérico	Entero largo
NombreCintaOpciones	Texto corto	255
XMLCintaOpciones	Texto largo	La teoría sería 1 GB de longitud, pero los controles Access es-tán limitados a 64.000 caracteres

Guarde la tabla con el nombre USysRibbons.

c. Alimentación de la tabla

A partir de ese momento, puede añadir nuevas cintas de opciones personalizadas copiando/pegando el contenido XML.

También es posible pasar por un formulario que mostraría los tres campos.

d. Carga de las cintas de opciones

Una vez que el conjunto de las cintas de opciones está disponible en la tabla `USysRibbons`, podemos ejecutar el tratamiento de la carga de todas las cintas de opciones durante la apertura de la base. Para esto, se crea en un módulo una función como se muestra a continuación:

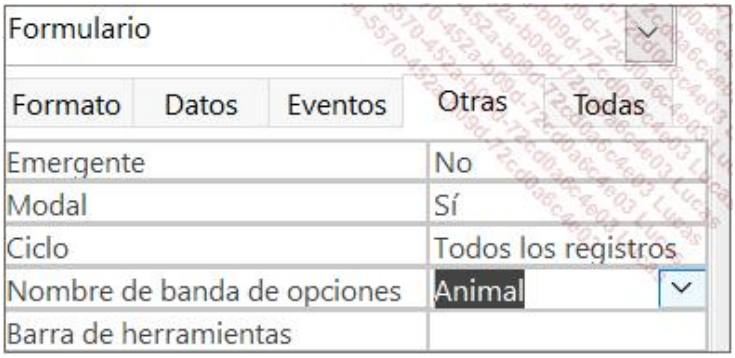
```
Public Function CargaTodasCintasOpciones()
    Dim RS As New ADODB.Recordset
    Dim strSQL As String
    strSQL = "SELECT * FROM UsysRibbons"
    RS.Open strSQL, CurrentProject.Connection, adOpenDynamic,
adLockOptimistic
    Do Until RS.EOF
           Application.LoadCustomUI RS.Fields("NombreCintaOpciones").Value, _
                 RS.Fields("XMLCintaOpciones").Value
           RS.MoveNext
    Loop
    RS.Close
End Function
```

Esta función se debe ejecutar durante la apertura de la base de datos. Por tanto, se crea una macro `AutoExec`.

Se selecciona **Macro** en la pestaña **Crear** y se añade una instrucción `EjecutarCodigo` y se completa el nombre de la función con `CargaTodasCintasOpciones()`. Se guarda la macro con el nombre `AutoExec`.

e. Asociación de las cintas de opciones con los formularios e informes

Una vez que todas las cintas de opciones se han cargado correctamente, es posible asociar los formularios con las cintas de opciones personalizadas.



7. Ejemplo de cinta de opciones personalizada

A continuación, vamos a ver cómo realizar una cinta de opciones personalizada y cómo interactuar con ella. La cinta de opciones elegida para este ejemplo es la siguiente:

a. Realización del contenido XML de la cinta de opciones

Para obtener esta cinta de opciones, el código XML utilizado sería el siguiente:

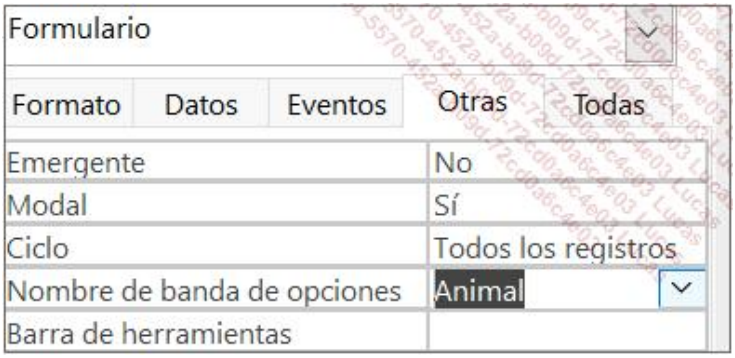
```
<customUI
xmlns="http://schemas.microsoft.com/office/2009/07/customui"
onLoad="DuranteCargaCinta">
    <ribbon startFromScratch="true">
           <tabs>
<!-Bloque Inicio -->
            <tab id="TabAnimales" label="Inicio">
<!-Grupo Animales -->
                <group id="Grupo_Animales" label="Animales">
                    <button id="Btn_Animal_Agregar"
label="Nuevo animal" size="large" onAction="Mostrar _Animal" tag=
"Logo_animales.jpg" getImage="Cargar_Imagen_Cinta" />
                    <button id="Btn_Animal_Busqueda"
label="Lista Animales" size="large" onAction="Buscar _Animal"
tag="vuelta_busqueda.jpg" getImage="Cargar_Imagen_Cinta"/>
                <button id="Btn_Animal_Estadistica" label=
"Estadist.animales" size="large" onAction="Estadistical_Animales"
imageMso="CycleFillColor"/>
                </group>
<!-- Grupo Adoptantes -->
                <group id="Grupo_Adoptante" label="Adoptantes">
                    <button id="Btn_Adoptante_Busqueda"
label="Lista" size="normal" onAction="Buscar_Adoptante"
imageMso="AddOrRemoveAttendees" />
                    <button id="Btn_Adoptante_Agregar"
label="Agregar" size="normal" onAction="Mostrar_Adoptante"
imageMso="DistributionListAddNewMember"/>
                    <button id="Btn_Adoptante_Mail"
label="Enviar hoja informativa" size="normal"
onAction="Mostrar_hojaInformativa" imageMso="CheckNames"/>
                    <button id="Btn_Adoptante_Estadistica"
label="Estadistical Adoptante" size="normal" onAction="Estadistica_Adoptante"
imageMso="CycleFillColor"/>
                </group>
<!-- Grupo Vacunas -->
               <group id="Grupo_Vacuna" label="Vacunas">
                    <button id="Btn_Vacuna_Agregar"
label="Nueva vacuna" size="large" onAction="Mostrar_Vacuna"
```

```
tag="Vacuna.jpg" getImage="Cargar_Imagen_Cinta"/>
                    <button id="Btn_Vacuna_Busqueda"
label="Lista vacunas" size="large" onAction="Buscar_Vacuna"
imageMso="FunctionsLookupReferenceInsertGallery"/>
                    <button id="Btn_Vacunacion"
label="Vacunaciones" size="large" onAction="Mostrar_Vacunación"
imageMso="LookUp"/>
                </group>
<!-- Grupo Box -->
               <group id="Grupo_Box" label="Box">
                    <button id="Btn_Box_Agregar"
label="Nueva caja" size="large" onAction="Mostrar_Caja "
imageMso="BlogHomePage"/>
                    <button id="Btn_Caja_Busqueda"
label="Lista cajas" size="large" onAction="Buscar_Caja "
imageMso="ArrangeByCompany"/>
               </group>
<!-- Grupo Salir -->
               <group id="Grupo_Salir" label="Salir">
                    <button id="Btn_Salir " label=" Salir
aplicación" size="large" onAction="Salir_Aplicacion"
imageMso="CancelRequest"/>
               </group>

          </tab>
       </tabs>
   </ribbon>
</customUI>
```

Podemos comprobar la existencia de tres grupos y 13 botones de comando en la cinta de opciones. Las imágenes utilizadas en la cinta de opciones tienen todas un identificador `imageMso`, pero se hubiera podido hacer referencia a otras imágenes personalizadas, como logos de empresa, por ejemplo.

b. Asociación con el formulario

Una vez determinado el contenido XML, puede añadirlo como registro en su tabla `USysRibbons`, creada durante la sección Realización del contenido XML de la cinta de opciones. Sería suficiente con añadir en la propiedad **Nombre de la cinta de opciones** la cinta de opciones especificada, como en el ejemplo anterior.



c. Redacción de los callbacks

Para poder utilizar los diferentes botones que se han creado en la cinta de opciones, el código VBA de las funciones de llamada callback será el siguiente:

```
'Gestión de la Cinta de opciones
'Común
Public Sub Cerrar_Formulario (control As IRibbonControl)
    DoCmd.Close acForm, Screen.ActiveForm.Name
End Sub

Public Sub Cargar_Imagen_Cinta(control As IRibbonControl,
ByRef image)
    Set image = LoadPicture(CurrentProject.Path &
"\images_cinta\" & control.Tag)
End Sub

'Inicio
'Grupo Animales
Public Sub Mostrar_Animal(control As IRibbonControl)
    SetParam "ANI_ID", 0
DoCmd.OpenForm "F_ANIMAL", acNormal, , , acFormEdit,
acWindowNormal
End Sub

Public Sub Buscar_Animal(control As IRibbonControl)
    DoCmd.OpenForm "F_ANIMALES", acNormal, , , acFormEdit,
acWindowNormal
End Sub

Public Sub Estados_Animal(control As IRibbonControl)
    DoCmd.OpenReport "R_Animales", acViewNormal, , ,
acWindowNormal
End Sub

'Grupo Adoptantes
Public Sub Buscar_Adoptante(control As IRibbonControl)
    DoCmd.OpenForm "F_Adoptantes"
End Sub

Public Sub Mostrar_Adoptante(control As IRibbonControl)
    SetParam "ADO_ID", 0
DoCmd.OpenForm "F_Adoptante"
End Sub

Public Sub Mostrar_BoletinNoticias(control As IRibbonControl)
    DoCmd.OpenForm "F_Infolettre"
End Sub

Public Sub Estados_Adoptante(control As IRibbonControl)

End Sub

'Grupo Vacunas
Public Sub Mostrar_Vacuna(control As IRibbonControl)
    SetParam "VAC_ID", 0
```

```
    DoCmd.OpenForm "F_Vacuna"
End Sub

Public Sub Buscar_Vacuna(control As IRibbonControl)
    DoCmd.OpenForm "F_Vacunas"
End Sub

Public Sub Mostrar_Vacunacion(control As IRibbonControl)
    SetParam "VAC_ID", 0
    SetParam "ANI_ID", 0
    DoCmd.OpenForm "F_Vacunacion"
End Sub


'Grupo Box
Public Sub Mostrar_Caja(control As IRibbonControl)
    SetParam "BOX_ID", 0
    DoCmd.OpenForm "F_Caja"
End Sub

Public Sub Buscar_Caja(control As IRibbonControl)
    DoCmd.OpenForm "F_Caja"
End Sub

'Salir
Public Sub Salir_Aplicacion(control As IRibbonControl)
    If MsgBox("¿Está seguro que desea salir de la aplicación?",
vbYesNoCancel) = vbYes Then
        Application.Quit acQuitSaveAll
    End If
End Sub
```

Automatización

Hasta ahora hemos visto varias maneras de programar en una aplicación Microsoft Access 2019. En este capítulo, vamos a ver cómo es posible controlar desde Access el resto de las aplicaciones adicionales principales de la suite de Office, que son Excel, Word y Outlook.

1. Enlace retrasado y enlace anticipado

Durante el trabajo con variables de tipo `Object`, el compilador VB realiza lo que se llama enlace cuando se asigna una variable a un valor, como en el siguiente ejemplo:

```
Set Tbl = CurrentDb.TableDefs("ENI_VACUNA_VAC")
```

Para poder realizar un enlace entre una variable y un objeto, la variable debe estar declarada con antelación en el programa. Esta declaración desembocará en dos posibles enlaces.

a. Enlace anticipado o Early Binding

Cuando la variable se declara con su tipo concreto, como en el siguiente ejemplo, se dice que el enlace es anticipado:

```
Dim Tbl As TableDef
```

Este tipo de declaración permite al código prever el espacio de memoria necesario para almacenar el tipo de la variable. También tenemos la ventaja de poder hacer referencia a los métodos y propiedades del tipo de objeto utilizando la funcionalidad de autocompletar.

Para que esta sintaxis sea posible, es necesario añadir al proyecto la referencia a la librería implicada (**Herramientas - Referencias**).

Se tiende a dar prioridad a este tipo de declaración cuando se puede hacer.

b. Enlace retrasado o Late Binding

Cuando la variable se declara de tipo `Object`, como en el siguiente ejemplo, se dice que la declaración es retrasada, es decir, que no se define concretamente a priori el tipo de la variable.

```
Dim Tbl As Object
```

Esta sintaxis permite crear una variable de tipo `Object`, pero sin saber más a este nivel del programa. Por tanto, no existen las ventajas del enlace anticipado. La funcionalidad de autocompletar o la ayuda interactiva ([F1]) ya no están disponibles. El código también será ligeramente más lento que con el enlace anticipado.

2. La función CreateObject

Cuando se desea asignar a una variable de tipo `Object` un objeto, podemos utilizar la función `CreateObject`,

que crea una instancia de la clase que se indica como argumento. La sintaxis general de esta función es la siguiente:

```
Public Function CreateObject(Clase As String, [NombreServidor
As String]) As Object
```

El argumento `Clase` contiene el nombre de la clase del objeto que se desea crear, y el argumento opcional `NombreServidor` permite indicar el servidor de red a partir del cual el objeto se creará. Si no se concreta ningún argumento `NombreServidor`, se utilizará el ordenador.

La llamada a esta función se hará como en el siguiente ejemplo:

```
Dim xlApp As Object
Set xlApp = CreateObject("Excel.Application")
```

Esta instrucción permitirá crear una instancia de la aplicación Excel, lo que creará un proceso EXCEL.EXE en la máquina.

Algunos objetos que crea este método deberán estar cerrados, como `xlApp.Quit` en el ejemplo anterior. Además, cualquier tipo de clase se puede referenciar en el código. Si la clase a la que hace referencia no está disponible, como por ejemplo cuando intenta crear un objeto cuya aplicación no está instalada en el puesto, recibirá un error 429 "El componente ActiveX no puede crear el objeto". Por supuesto, este error no aparece en todos los ordenadores.

3. La función GetObject

De la misma manera que la función `CreateObject` puede crear una nueva instancia de una clase en la máquina, la función `GetObject` recupera una instancia de clase que ya existiría en el puesto. La sintaxis general de la función es la siguiente:

```
Public Function GetObject([UbicaciónArchivo As String],[Clase
As String])
```

El argumento `Clase` corresponde al nombre de la clase del objeto que se desea crear, como en la función `CreateObject`. El argumento `Ubicación Archivo` corresponde a la ubicación del archivo que VBA abrirá, antes de asignar su tipo a la variable que servirá de receptor.

Los dos argumentos son opcionales, sin embargo será necesario indicar al menos uno para que la función no genere error. Hay varias sintaxis posibles para la llamada a esta función.

a. Solo se indica la ubicación del archivo

Solo indicando la ubicación del archivo, la función devolverá un objeto del mismo tipo que el archivo especificado. Por ejemplo, si se ejecuta el siguiente código:

```
Sub EjemploGetObject()
Dim a As Object
```

```
Set a = GetObject(CurrentProject.Path & "\ENI.xlsx") 'el archivo
ENI.xlsx está en la misma carpeta que la base de datos actual.
Debug.Print TypeName(a)
End Sub
```

La variable `a` almacena un objeto de tipo `Workbook`.

Se puede comprobar que la función `TypeName` devuelve una cadena de caracteres que contiene el tipo de dato de la variable que se pasa como argumento.

b. Se indica la ubicación del archivo y la clase

Indicando la ubicación del archivo, así como la clase, la función devolverá un objeto del tipo de la clase especificada. Por ejemplo, si se ejecuta el siguiente código:

```
Sub EjemploGetObject2()
Dim a As Object
Set a = GetObject(CurrentProject.Path & "\ENI.docx",
"Word.Document") 'el documento ENI.docx se sitúa en la misma
carpeta que la base de datos actual.
Debug.Print TypeName(a)
End Sub
```

La variable `a` almacena un objeto de tipo `Document`.

c. Solo se indica la clase

Si solo se indica la clase, la función devolverá un objeto de la clase especificada, libre de cualquier modificación.

```
Sub EjemploGetObject3()
Dim a As Object
Set a = GetObject(, "PowerPoint.Application")
Debug.Print TypeName(a)
End Sub
```

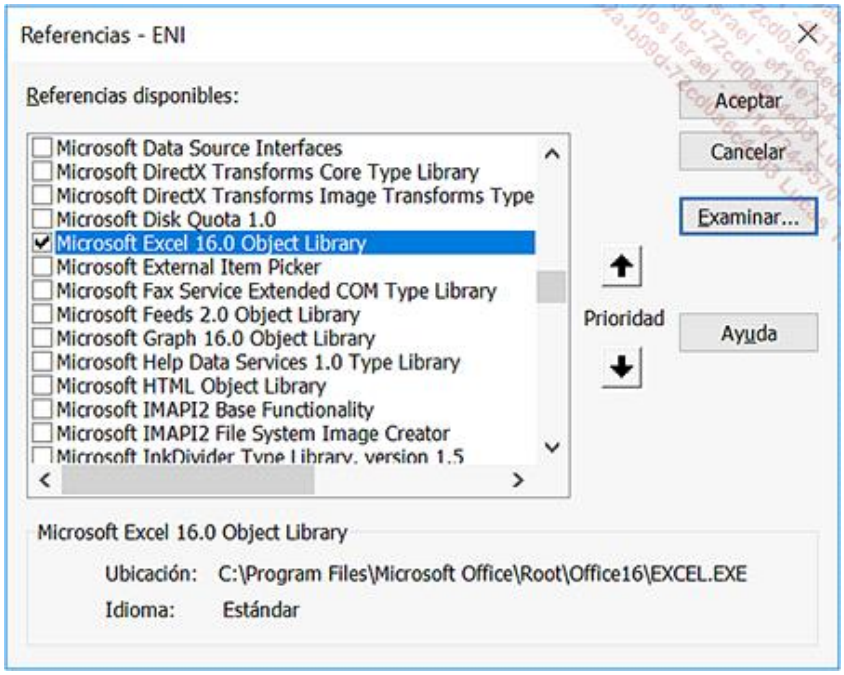
La variable `a` almacena un objeto de tipo `PowerPoint.Application`.

4. Automatización

Para que las aplicaciones se puedan comunicar entre ellas, se llama al mecanismo OLE Automatization, o de manera más simple Automatization, que se basa en la arquitectura COM (*Component Object Model*). La tecnología OLE (*Object Linking and Embedding*) permite a las aplicaciones diferentes formatos de comunicación, y ello fundamentalmente durante las conexiones con programación VBA. El programa a partir del que se desarrolla utilizará como servidor OLE la aplicación controlada desde Microsoft Access. Por ejemplo, si se controla una aplicación Excel 2019 desde Access, Excel hará de servidor OLE. Cuando un programa soporta la automatización, como es el caso de las aplicaciones de Office 2019, puede utilizar VBA para acceder a los objetos que expone. Se pueden manipular estos objetos en VB, invocando los métodos en el objeto u obteniendo y definiendo las propiedades del objeto.

Controlar Excel

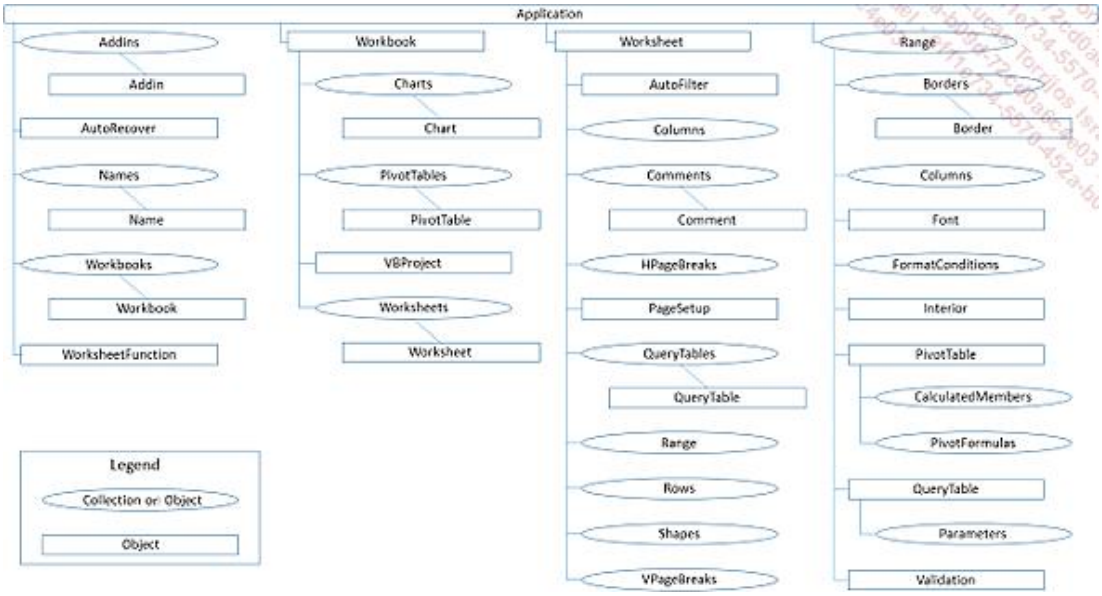
Para controlar la aplicación Excel, es necesario añadir la librería **Microsoft Excel 16.0 Object Library** en las referencias al proyecto (**Herramientas** - **Referencias**).



Si no encuentra esta librería en la lista disponible, puede hacer clic en **Examinar** y buscar el archivo Excel.exe en su máquina, en la que normalmente estará en la siguiente ubicación: C:\Program Files\Microsoft Office\root\Office16 \EXCEL.EXE

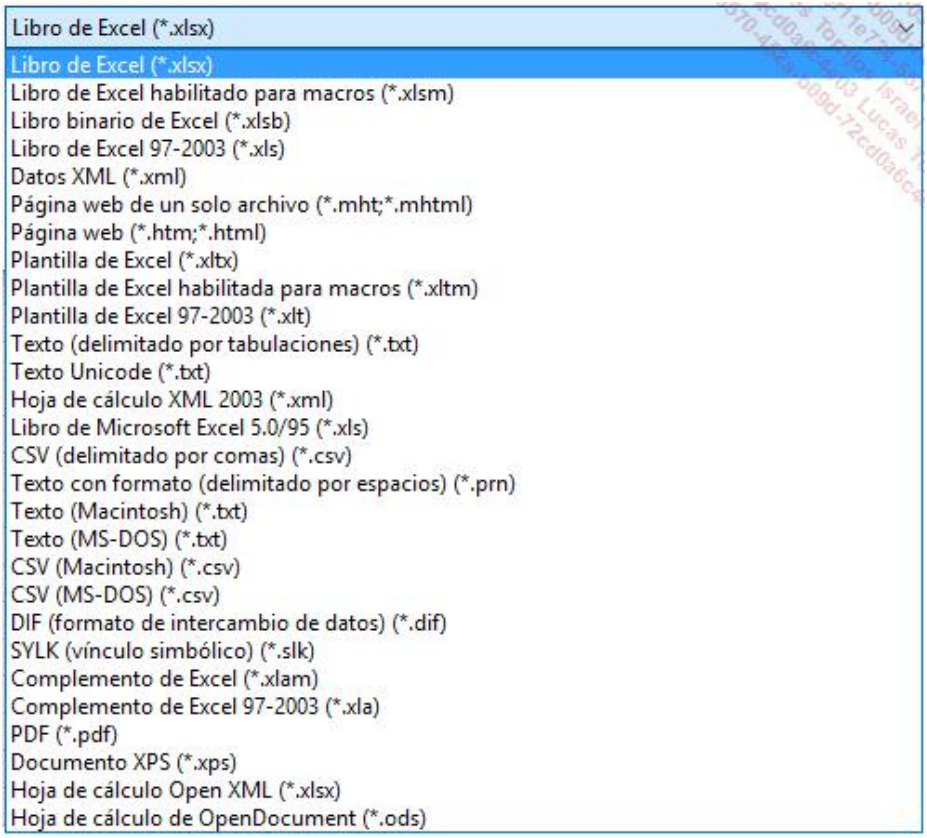
1. Jerarquía de los objetos de Excel

El siguiente esquema representa la jerarquía de los principales objetos y colecciones disponibles en Microsoft Excel.



2. Los formatos de Excel

Los diferentes formatos que se soportan aparecen en la interfaz de Excel 2019 que permite guardar.



Respecto a los archivos que se definen como "archivos de Excel", hay varias extensiones de archivos. En esta sección se va a recordar las diferentes versiones que puede manipular.

a. Antes de Excel 2007 - .xls

La extensión histórica de los archivos Excel es .xls. El nombre de este formato es Libro de Microsoft Excel 97-2003, y contiene entre 16.384 y 65.536 filas, según las versiones, con 256 columnas.

b. Después de Excel 2007

Cuando salió la versión Microsoft Excel 2007, aparecieron varias nuevas extensiones. El soporte del formato XML y el aumento de la capacidad de la almacenamiento de los archivos son las principales evoluciones y mejoras añadidas a estos.

Extensión	Formato	Descripción
xlsx	Libro de Excel	Formato por defecto, archivo sin macro.
xlsm	Libro de Excel que soporta las macros	Formato que permite guardar módulos de código VBA.
xlsb	Libro de Excel binario	Formato que permite almacenar macros, haciendo más compacto el archivo.
xlt, xltx, xltm	Libro plantilla de Excel	Formato de plantilla, xlt es anterior a Microsoft Excel 2007, xltx es una plantilla sin macro, xltm una plantilla con macro.

Las extensiones de archivos que se utilizan más habitualmente son xlsx y xlsm, y en menor medida xlsb, a pesar de que su rendimiento y tamaño reducido que ofrece a contenido equivalente.

También hay extensiones de archivos compatibles, pero que no tienen interés actualmente, vista la evolución de la tecnología y, fundamentalmente, la llegada de la cinta de opciones.

Extensiones compatibles pero anticuadas	Descripción
xlb	Archivo para la administración de las barras de herramientas y comandos, que hoy se sustituyen por la cinta de opciones.
xll	Formato equivalente a las dll de Excel; permite que esté disponible el código VBA del libro.
xlw	Formato que permite la apertura de varios libros de Excel al mismo tiempo.

3. Lista de los principales objetos y colecciones de Excel

Los principales objetos y colecciones que es posible manipular con Excel son los siguientes.

a. La aplicación

El objeto `Application` de Excel es el objeto principal en el modelo de Excel, al mismo nivel que el objeto `Application` de Microsoft Access. Para que no haya conflictos de nombres, debe estar precedido del nombre de la aplicación. Por tanto, la variable que representa el objeto `Application` de Excel se declarará como sigue:

```
Dim xlApp As Excel.Application
```

Observe que a este nivel del programa todavía no se ha creado ninguna instancia de Excel. Solo puede existir cuando la variable `xlApp` tenga valor. La siguiente sintaxis crea una nueva instancia de la aplicación Excel durante la declaración de la variable:

```
Dim xlApp As New Excel.Application
```

Para no crear procesos zombis en la máquina, conviene salir de la aplicación Excel con la instrucción:

```
xlApp.Quit
```

Principales métodos

El objetivo de este libro no es explicar exhaustivamente respecto a la aplicación Microsoft Excel 2019. A continuación se muestran los principales métodos que es interesante conocer para controlarla desde Microsoft Access:

Método	Descripción
`Calculate`	Permite forzar el cálculo de los libros abiertos en la aplicación Excel.

CheckSpelling	Permite la comprobación ortográfica de una palabra.
Evaluate	Permite realizar los cálculos a partir de una fórmula, expresada como cadena de caracteres.
GetOpenFileName	Permite visualizar el cuadro de diálogo **Examinar**, que devuelve como una cadena de caracteres la ubicación del archivo o de los archivos seleccionados.
GetSaveAsFileName	Permite visualizar el cuadro de diálogo **Guardar en**, que devuelve la ubicación deseada, pero no guarda realmente el libro de Excel.
InputBox	Permite visualizar un cuadro de diálogo para una entrada de usuario. Es más completa que la función InputBox nativa de VBA.
OnTime	Permite planificar la ejecución de una macro en un momento dado en el futuro.
Quit	Permite salir de Excel.
Run	Permite ejecutar una macro o llamar a una función VBA.

Principales propiedades

Propiedad	Descripción
Calculation	Permite definir si el cálculo es automático (por cada modificación) o manual (bajo demanda) en los libros de Excel abiertos.
DisplayAlerts	Permite definir si los mensajes de alerta están activados durante la ejecución de una macro.
EnableEvents	Permite definir si los eventos están activados en los libros de Excel abiertos.
International	Permite conocer la información de los argumentos regionales e internacionales del lugar en la máquina y Microsoft Excel.
Path	Permite conocer la ubicación de la aplicación Excel.exe en el puesto.
ScreenUpdating	Permite definir si la ventana se actualiza.
StatusBar	Permite devolver o definir el texto que aparece en la barra de estado.
VBE	Permite acceder al código VBA de Microsoft Excel.

Ejemplo

El siguiente código crea una nueva aplicación de Excel, la hace visible y después la cierra:

```
Dim xlApp As New Excel.Application
xlApp.Visible = True
xlApp.Quit
```

b. El libro de Excel Workbook

El objeto Workbook es el objeto que representa un libro de Excel. El libro ActiveWorkbook es el libro de Excel activo en la aplicación Microsoft Excel. Para declarar una variable de tipo Workbook, la siguiente sintaxis es posible:

```
Dim xlwbk As Excel.Workbook
```

O directamente:

```
Dim xlwbk As Workbook
```

Principales métodos de la colección Workbooks

Método	Descripción
Add	Permite crear un nuevo libro de Excel y determinarlo como libro activo.
Close	Permite cerrar todos los libros de Excel abiertos.
Open	Permite abrir un libro de Excel.

Principales métodos de Workbook

Método	Descripción
Activate	Permite activar un libro de Excel.
Close	Permite cerrar el libro de Excel.
PrintOut	Permite imprimir el libro de Excel.
Protect	Permite proteger el libro de Excel para que no se pueda modificar.
Save	Permite guardar el libro de Excel.
SaveAs	Permite guardar (con un nuevo nombre) el libro de Excel.
SaveCopyAs	Permite guardar una copia del libro de Excel sin modificar el libro de Excel abierto en memoria.
Unprotect	Permite eliminar la protección del libro de Excel.

Principales propiedades

Propiedad	Descripción
FileFormat	Permite conocer el formato del libro de Excel.
FullName	Permite conocer el nombre del libro de Excel, así como su ubicación.
HasPassword	Permite saber si el libro de Excel está protegido mediante contraseña.
HasVBProject	Permite saber si el libro de Excel contiene código VBA.
Name	Permite conocer el nombre del libro de Excel.
Path	Permite conocer la ubicación del libro de Excel.

Principales colecciones

Colección	Descripción
Charts	Contiene todos los gráficos Chart del libro de Excel.
PivotTables	Contiene todas las tablas cruzadas dinámicas PivotTable del libro de Excel.
Sheets	Contiene todas las hojas Sheet del libro de Excel.

Worksheets	Contiene todas las hojas Worksheet del libro de Excel (es idéntico que el anterior).

Ejemplo

El siguiente código permite al usuario seleccionar un archivo de Excel y lo abre desde Access:

```
Dim xlApp As New Excel.Application
Dim xlwbk As Excel.Workbook
Dim Ubicacion As String
Ubicacion = xlApp.GetOpenFileName()
If Ubicacion <>Cstr(False) Then
    Set xlwbk = xlApp.Workbooks.Open(Ubicacion)
End If
```

c. La hoja Worksheet

El objeto Worksheet es el objeto que representa una hoja de Excel. La hoja ActiveWorksheet es la hoja activa en un libro de Excel. Para declarar una variable de tipo Worksheet, es posible usar la siguiente sintaxis:

```
Dim xlwsh As Excel.Worksheet
```

O incluso:

```
Dim xlwsh As Worksheet
```

Principales métodos de la colección Worksheets

Método	Descripción
Add	Permite crear una nueva hoja, hacerla activa.
Copy	Permite copiar la hoja en otra ubicación del libro de Excel.
Delete	Permite eliminar la hoja.
Move	Permite mover una hoja a otro lugar del libro de Excel.
PrintOut	Permite imprimir la hoja.

Principales métodos de Worksheet

Método	Descripción
Activate	Permite activar una hoja.
Copy	Permite copiar la hoja en otra ubicación del libro de Excel.
Delete	Permite eliminar la hoja.
Move	Permite mover una hoja a otro lugar del libro de Excel.

Paste	Permite pegar el contenido del portapapeles en la hoja.
PasteSpecial	Permite pegar el contenido del portapapeles según un formato particular.
PrintOut	Permite imprimir la hoja.
Protect	Permite proteger la hoja para que no se pueda modificar.
SaveAs	Permite guardar la hoja, asignándole un nombre.
Unprotect	Permite eliminar la protección de la hoja.

Principales propiedades

Propiedad	Descripción
Name	Permite conocer el nombre de la hoja.
PageSetup	Permite acceder a la configuración de los elementos de la hoja.
Visible	Permite determinar si la hoja es visible.

Principales colecciones

Colección	Descripción
Cells	Representa todas las celdas de la hoja.
Columns	Representa todas las columnas de la hoja.
Comments	Representa todos los comentarios Comments de la hoja.
Hyperlinks	Representa todos los enlaces de hipertexto de la hoja.
QueryTables	Representa todas las tablas de consulta QueryTable de la hoja.
Range	Permite acceder a una celda o rango de celdas de la hoja.
Rows	Representa todas las filas de la hoja.
Shapes	Representa todas las formas de la hoja.

Ejemplo

El siguiente código permite recorrer el conjunto de las hojas de un libro de Excel xlwbk abierto con antelación y visualizar los nombres de cada una de ellas:

```
Sub Ejemplo5()
Dim xlwsh As Excel.Worksheet
Dim xlApp As New Excel.Application
Dim xlwbk As Excel.Workbook
Dim Ubicacion As String
Ubicacion = xlApp.GetOpenFileName()
If Ubicacion <> CStr(False) Then
    Set xlwbk = xlApp.Workbooks.Open(Ubicacion)
    For Each xlwsh In xlwbk.Worksheets
          Debug.Print xlwsh.Name
    Next
    xlwbk.Close
    xlApp.Quit
```

```
End If
End Sub
```

d. Las celdas Range y Cells

El objeto `Range` es el objeto que representa una celda o un rango de celdas de Excel. La celda `ActiveCell` es la celda activa en una hoja. Para declarar una variable de tipo `Range`, es posible la siguiente sintaxis:

```
Dim xlRng As Excel.Range
```

E incluso:

```
Dim xlRng As Range
```

Principales métodos del objeto Range

Método	Descripción
`Activate`	Permite activar una celda.
`AutoFit`	Permite ajustar la anchura de la columna en función del contenido de la celda.
`Clear`	Permite eliminar el contenido y el formateo de la celda.
`Copy`	Permite copiar la hoja a otra ubicación del libro de Excel.
`Cut`	Permite cortar el contenido de la celda y ubicarlo en el portapapeles.
`Delete`	Permite eliminar la celda.
`Insert`	Permite insertar una celda.
`Merge`	Permite fusionar celdas.
`PasteSpecial`	Permite pegar el contenido del portapapeles según un formato particular.
`PrintOut`	Permite imprimir la celda o el rango de celdas.

Principales propiedades

Propiedad	Descripción
`Column`	Devuelve el número de la columna en la que se encuentra la celda.
`ColumnWidth`	Representa la anchura de la columna en la que se encuentra la celda.
`Count`	Devuelve el número de celdas contenidas en el objeto `Range`.
`EntireColumn`	Representa la columna entera en la que se encuentra la celda.
`EntireRow`	Representa la fila entera en la que se encuentra la celda.
`Font`	Representa el tipo de letra de la celda.
`Formula`	Representa la fórmula contenida en la celda, en forma de cadena de caracteres.
`Height`	Representa la altura de la celda.
`Hidden`	Permite determinar si la celda está oculta o no.

HorizontalAlignment	Permite determinar la alineación horizontal del contenido de la celda.
Interior	Representa el fondo (el interior) de la celda.
Row	Devuelve el número de la fila en la que se encuentra la celda.
RowHeight	Representa la altura de la fila en la que se encuentra la celda.
Value	Representa el valor de la celda.

Principales colecciones

Colección	Descripción
Borders	Representa los bordes de la celda o el rango de celdas.
Cells	Representa la lista de todas las celdas del rango de celdas.
Columns	Representa la lista de las columnas en las que se encuentra el rango de celdas.
Rows	Representa la lista de las filas en las que se encuentra el rango de celdas.

Ejemplo

El siguiente código permite escribir los nombres de los meses del año en las celdas A1 a A12 de la hoja xlwsh, determinada anteriormente en el programa:

```
Sub Ejemplo6()
Dim xlwsh As Excel.Worksheet
Dim xlApp As New Excel.Application
Dim xlwbk As Excel.Workbook

Set xlwbk = xlApp.Workbooks.Add
Set xlwsh = xlwbk.Worksheets(1)
Dim i As Integer
For i = 1 To 12
    xlwsh.Range("A" & i).Value = Format(DateSerial(Year(Now), i,
1), "mmmm")
Next i
xlApp.Visible = True
Set xlApp = Nothing
End Sub
```

Otra manera llegar a las celdas es usar el objeto Cells, con la siguiente sintaxis:

```
Cells(número_fila, número_columna)
```

e. Los gráficos Chart

El objeto Chart es el objeto que representa un gráfico de Excel. El gráfico ActiveChart es el gráfico activo en una hoja. Para declarar una variable de tipo Chart, es posible la siguiente sintaxis:

```
Dim xlCht As Excel.Chart
```

O incluso:

```
Dim xlCht As Chart
```

Principales métodos del objeto Chart

Método	Descripción
CopyPicture	Permite copiar el gráfico y situarlo en el portapapeles como imagen.
Export	Permite exportar el gráfico en un formato de imagen.
Move	Permite mover el gráfico a una nueva ubicación.
PrintOut	Permite imprimir el gráfico.
Refresh	Permite actualizar los datos del gráfico.
SetSourceData	Permite definir el rango de datos origen del gráfico.

Principales propiedades

Propiedad	Descripción
ChartArea	Representa la zona gráfica del gráfico.
ChartColor	Representa el juego de colores del gráfico.
ChartStyle	Representa el estilo gráfico del gráfico.
ChartTitle	Representa el título del gráfico.
ChartType	Representa el tipo gráfico del gráfico.
HasLegend	Indica si el gráfico tiene una leyenda.
HasTitle	Indica si el gráfico tiene un título.
Legend	Representa la leyenda del gráfico.
PageSetup	Representa los elementos de configuración del gráfico en la página.
PlotArea	Representa la superficie de trazado del gráfico.
PlotBy	Representa el soporte de los datos origen en fila o en columna.

Principales colecciones

Colección	Descripción
Shapes	Representa la colección de todas las formas de la hoja del gráfico.
ChartObjects	Representa la colección de todos los gráficos incorporados.
SeriesColección	Representa la colección de todas las series del gráfico.

Ejemplo

El siguiente código permite generar, para los años entre el 2001 y el 2018 (columna A), valores aleatorios entre 0 y 20 (columna B), en el rango de celdas A1:B19 de una hoja `xlwsh` determinada con anterioridad en el programa, y visualizar estos datos en un gráfico de tipo columna:

```
Sub Ejemplo7()
Dim xlwsh As Excel.Worksheet
Dim xlApp As New Excel.Application
Dim xlwbk As Excel.Workbook
Set xlwbk = xlApp.Workbooks.Add
Set xlwsh = xlwbk.Worksheets(1)
Dim i As Integer
xlwsh.Range("A1").Value = "Año"
xlwsh.Range("B1").Value = "CA"
For i = 1 To 18
    xlwsh.Range("A" & i + 1).Value = 2000 + i
    xlwsh.Range("B" & i + 1).Value = Int(Rnd() * 20)
Next i
Dim xlCht As Excel.Chart
xlwsh.Shapes.AddChart2(201, xlColumnClustered).Select
Set xlCht = xlapp.ActiveChart
With xlCht
    .SetSourceData Source:=xlwsh.Range("A1:B19")
    .FullSeriesCollection(1).Delete
    .FullSeriesCollection(1).XValues = "=" & xlwsh.Name & "!A2:A19"
End With
xlApp.Visible = True
Set xlApp = Nothing
End Sub
```

f. Las tablas dinámicas PivotTable

El objeto `PivotTable` es el objeto que representa una tabla dinámica de Excel. Para declarar una variable de tipo `PivotTable`, es posible la siguiente sintaxis:

```
Dim xlPvT As Excel.PivotTable
```

O incluso:

```
Dim xlPvT As PivotTable
```

Principales métodos del objeto PivotTable

Método	Descripción
`GetData`	Permite recuperar el valor de los datos en la tabla dinámica.
`RefreshTable`	Permite actualizar los datos de la tabla dinámica.

Principales propiedades

Propiedad	Descripción
Location	Representa la dirección de la celda en la parte superior izquierda de la tabla dinámica.
SourceData	Representa la fuente de datos para la tabla dinámica.

Principales colecciones

Colección	Descripción
ColumnRange	Representa la lista de columnas en el informe de la tabla dinámica.
DataFields	Representa la colección de campos de la tabla dinámica.
RowRange	Representa la lista de filas en el informe de la tabla dinámica.

g. Otras posibilidades

Además de los diferentes niveles de objetos que es posible controlar mediante la automatización en Excel, la transferencia de datos desde la base de datos Access se puede hacer con las siguientes funciones.

Range.CopyFromRecordSet

Cuando se manipulan los registros a través de los Recordset (ADO o DAO), podemos volver a copiar todos estos datos íntegramente con el método CopyFromRecordset del objeto Range. La sintaxis para copiar es la siguiente:

```
Sub Ejemplo8()
Dim xlwsh As Excel.Worksheet
Dim xlApp As New Excel.Application
Dim xlwbk As Excel.Workbook
Set xlwbk = xlApp.Workbooks.Add
Set xlwsh = xlwbk.Worksheets(1)
Dim RS As New ADODB.Recordset
RS.Open "SELECT * FROM ENI_ANIMAL_ANI", CurrentProject.Connection,
adOpenDynamic, adLockOptimistic
'copia los datos a partir de la celda A2.
xlwsh.Range("A2").CopyFromRecordset RS
RS.Close
xlApp.Visible = True
Set xlApp = Nothing
End Sub
```

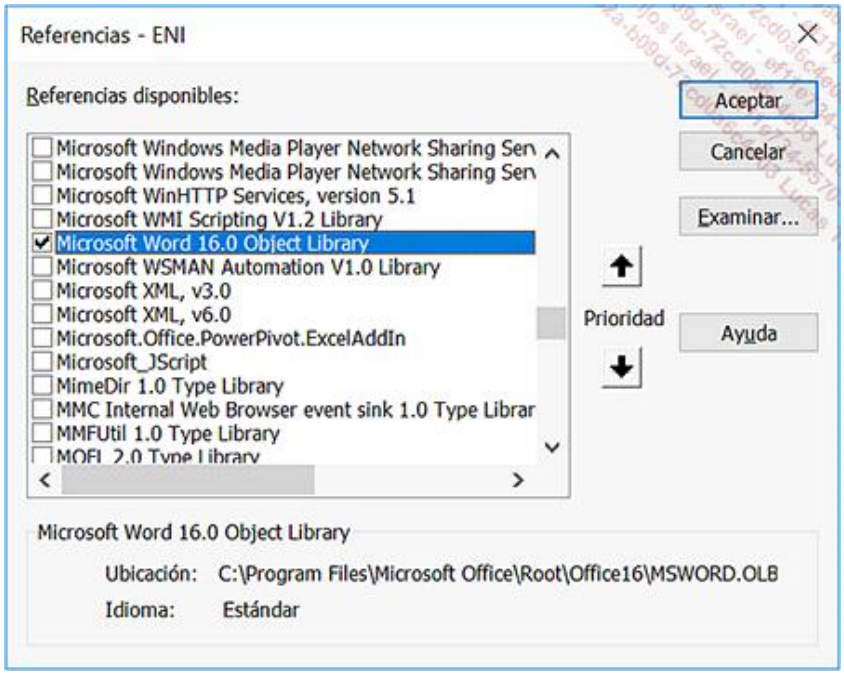
Escritura fila a fila

También es posible recorrer los registros uno por uno y escribir el contenido de los campos en las celdas. El siguiente código permitirá obtener un resultado similar al ejemplo de la Range.CopyFromRecordset, añadiendo etiquetas de columna en la primera fila.

```
Sub Ejemplo9()
Dim xlwsh As Excel.Worksheet
Dim xlApp As New Excel.Application
Dim xlwbk As Excel.Workbook
Set xlwbk = xlApp.Workbooks.Add
Set xlwsh = xlwbk.Worksheets(1)
Dim RS As New ADODB.Recordset
Dim fld As Object
Dim i As Integer, j As Integer
RS.Open "SELECT * FROM ENI_ANIMAL_ANI", CurrentProject.Connection,
adOpenDynamic,
 adLockOptimistic
'copia las etiquetas de columnas en la primera fila
i = 1: j = 1
For Each fld In RS.Fields
    xlwsh.Cells(i, j).Value = fld.Name
    j = j +1
Next
'copia los datos a partir de la celda A2.
I = 2
Do Until RS.EOF
    For j = 0 To RS.Fields.Count-1
           xlwsh.Cells(i, j + 1).Value = RS.Fields(j).Value
    Next j
    i = i +1
    RS.MoveNext
Loop
RS.Close
xlApp.Visible = True
Set xlApp = Nothing
End Sub
```

Controlar Word

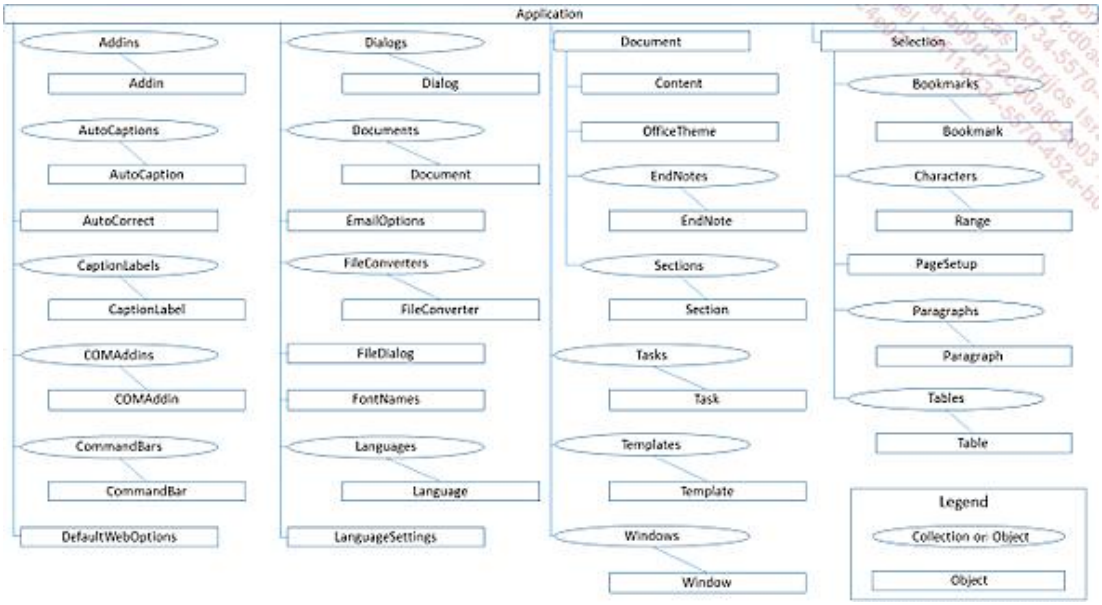
Para controlar la aplicación Word, es necesario añadir la librería **Microsoft Word 16.0 Object Library** en las referencias del proyecto (**Herramientas** - **Referencias**).



Si no encuentra esta librería en la lista disponible, puede hacer clic en **Examinar** y buscar en su máquina el archivo WinWord.exe, que normalmente estará en la ubicación: C:\Program Files\Microsoft Office\Root\Office16 \WINWORD.EXE

1. Jerarquía de los objetos de Word

El siguiente esquema representa la jerarquía de los principales objetos y colecciones en la aplicación.



2. Lista de los principales objetos y colecciones de Word

Los principales objetos y colecciones que es posible manipular en Word son los siguientes.

a. La aplicación

El objeto `Application` de Word es el objeto de más alto nivel en el modelo de Word, al mismo nivel que el objeto `Application` de Microsoft Access. Para no tener conflicto con los nombres, debe estar precedido del nombre de la aplicación. Por tanto, la variable que representa al objeto `Application` de Word, se declarará como sigue:

```
Dim wdApp As Word.Application
```

Observe que a este nivel del programa no se ha creado ninguna instancia de Word. Solo puede existir cuando la variable `wdApp` tenga valor. La siguiente sintaxis crea una nueva instancia de la aplicación Word durante la declaración de la variable:

```
Dim wdApp As New Word.Application
```

Para no crear procesos zombis en la máquina, convendría salir de la aplicación Word con la instrucción:

```
wdApp.Quit
```

Principales métodos

El objetivo del presente libro no es explicar exhaustivamente la aplicación Microsoft Word 2019. A continuación se muestran los principales métodos que es interesante conocer para controlarla desde Microsoft Access.

Método	Descripción
`CheckSpelling`	Permite la comprobación ortográfica de una palabra.
`MergeDocuments`	Permite fusionar dos documentos de Word, los elementos fusionados se marcan usando las marcas de revisión.
`OnTime`	Permite planificar la ejecución de una macro en un momento dado del futuro.
`PrintOut`	Permite la impresión del documento especificado.
`Quit`	Permite salir de Word.
`Run`	Permite ejecutar una macro o llamar a una función VBA.

Principales propiedades

Propiedad	Descripción
`DisplayAlerts`	Permite definir si los mensajes de alerta están activos durante la ejecución de una macro.
`International`	Permite conocer la información de los argumentos regionales e internacionales del lugar en la máquina y Microsoft Word.

Options	Representa las opciones de la aplicación.
Path	Permite conocer la ubicación de la aplicación WinWord.exe en el puesto.
ScreenUpdating	Permite definir si la ventana se actualiza.
Selection	Representa el punto de inserción.
StatusBar	Permite devolver o definir el texto que aparece en la barra de estado.
System	Permite acceder a la información relacionada con el sistema.
VBE	Permite acceder al código VBA de Microsoft Word.

Ejemplo

El siguiente código crea una nueva aplicación de Word, la hace visible, muestra la versión de la aplicación y después la cierra:

```
Sub Ejemplo3()
Dim wdApp As New Word.Application
wdApp.Visible = True
MsgBox wdapp.Version
wdApp.Quit
End Sub
```

b. El documento Document

El objeto Document es el objeto que representa un documento de Word. El documento ActiveDocument es el documento activo en la aplicación Microsoft Word. Para declarar una variable de tipo Document, es posible la siguiente sintaxis:

```
Dim wddoc As Word.Document
```

Principales métodos de la colección Documents

Método	Descripción
Add	Permite crear un nuevo documento y convertirlo en el documento activo.
Close	Permite cerrar todos los documentos abiertos.
Open	Permite abrir un documento.

Principales métodos de Document

Método	Descripción
Activate	Permite activar un documento.
CheckGrammar	Permite ejecutar la comprobación ortográfica y gramatical sobre el documento.
Close	Permite cerrar el documento.
PrintOut	Permite imprimir el documento.

Protect	Permite proteger el documento para que no se pueda modificar.
Repaginate	Permite volver a paginar el conjunto del documento.
Save	Permite guardar todos los documentos de la colección Documents.
SaveAs2	Permite guardar el documento con otro nombre.
Unprotect	Permite eliminar la protección del documento.

Principales propiedades

Propiedad	Descripción
AttachedTemplate	Representa el modelo adjunto (Template) al documento.
FullName	Permite conocer el nombre del documento, así como su ubicación.
HasPassword	Permite saber si el libro está protegido mediante contraseña.
HasVBProject	Permite saber si el documento contiene código VBA.
Kind	Permite determinar el tipo de formato utilizado por Word en el documento.
Name	Permite conocer el nombre del documento.
PageSetup	Permite acceder a los elementos de configuración del documento.
Path	Permite conocer la ubicación del documento.

Principales colecciones

Colección	Descripción
Bookmarks	Representa la colección de marcadores (Bookmark) del documento.
Characters	Representa la colección de caracteres del documento.
EndNotes	Representa la colección de notas al final del documento.
FootNotes	Representa la colección de notas al pie de página del documento.
Frames	Representa la colección de márgenes del documento.
Shapes	Representa la colección de formas del documento.
Tables	Representa la colección de tablas del documento.
Words	Representa la colección de palabras del documento.

Ejemplo

El siguiente código abre un archivo de Word (situado en la misma carpeta que el de la base de datos desde Access), muestra el número de palabras que contiene y después cierra Word:

```
Sub Ejemplo4()
Dim wdApp As New Word.Application
Dim wddoc As Word.Document
Set wddoc = wdApp.Documents.Open(CurrentProject.Path &
"\ENI.docx")
MsgBox wddoc.Words.Count
wWddoc.Close
```

```
wdApp.Quit
End Sub
```

c. El objeto Table

El objeto `Table` es el objeto que representa una tabla en un documento de Word. Para declarar una variable de tipo `Table`, es posible la siguiente sintaxis:

```
Dim wdTabla As Word.Table
```

Principales métodos de la colección Tables

Método	Descripción
Add	Permite crear una nueva tabla vacía.
Item	Permite apuntar a una tabla que pertenece a todas las tablas existentes.

Principales métodos de Table

Método	Descripción
Cell	Permite apuntar a una celda de la tabla.
Delete	Permite eliminar la tabla.
Sort	Permite ordenar el contenido de la tabla
SortAscending	Permite ordenar los párrafos o filas de la tabla por orden alfabético creciente.
SortDescending	Permite ordenar las filas de la tabla por orden alfabético decreciente.

Principales propiedades

Propiedad	Descripción
AllowAutofit	Permite indicar si Word puede redimensionar automáticamente las celdas de la tabla para ajustarse a su contenido.
Shading	Permite manipular el formateo de la trama de fondo de la tabla.
Spacing	Permite manipular el espaciado entre las celdas de la tabla.

Principales colecciones

Colección	Descripción
Borders	Representa la colección de bordes de la tabla.
Columns	Representa la colección de columnas de la tabla.

Ejemplo

El siguiente código permite añadir una nueva tabla de 10 por 10, en un nuevo documento `wdDoc`, a partir de Microsoft Word ya abierto en el puesto y visualizar en cada una de las celdas los números de 1 a 100:

```
Sub Ejemplo5()
Dim wdApp As Word.Application
Dim wdDoc As Word.Document
Dim wdTabla As Word.Table
Dim i As Integer, j As Integer
Set wdApp = GetObject(, "Word.Application")
Set wdDoc = wdApp.Documents.Add
Set wdTabla = wdDoc.Tables.Add (wddoc.Range, numrows:=10,
numcolumns:=10)
With wdTabla
    For i = 0 To 9
           For j = 1 To 10
                  .Cell(i+1, j).Range.Text = i*10 +j
           Next j
    Next i
End With
wdApp.Visible = True
Set wdApp = Nothing
End Sub
```


d. Otras posibilidades

Además de los diferentes niveles de objetos que es posible controlar por la automatización de Word, la transferencia de datos desde la base de datos de Access se puede hacer mediante la publicación por mail.

Document.MailMerge

Cuando se manipulan los registros a través de los `Recordset` (ADO o DAO), podemos volver a copiar todos esos datos con el método `MailMerge` del objeto `Document`. La sintaxis para imprimir un documento para cada registro es la siguiente, pasando por los marcadores (`Bookmark`) del documento:

```
Sub CombinarCorrespondencia()
Dim wdApp As Word.Application
Dim ruta_Plantilla As String
Dim RS As New ADODB.Recordset
RS.Open "SELECT * FROM ENI_ANIMAL_ANI", CurrentProject.Connection,
adOpenDynamic, adLockOptimistic
Set wdApp = New Word.Application
ruta_Plantilla = CurrentProject.Path & "\Plantilla.Docx"
wApp.Visible = True

Do While Not RS.EOF
With wApp
    .Documents.Open (ruta_Plantilla)
    .ActiveDocument.Bookmarks("nombre").Range.Text =
RS.Fields("ANI_NOMBRE").Value
    .ActiveDocument.Bookmarks("id").Range.Text =
 RS.Fields("ANI_ID").Value
```

```
    .ActiveDocument.PrintOut
    .ActiveDocument.Close False
End With
RS.MoveNext
Loop
RS.Close
Set RS = Nothing
wdApp.Quit
Set wdApp = Nothing
End Sub
```

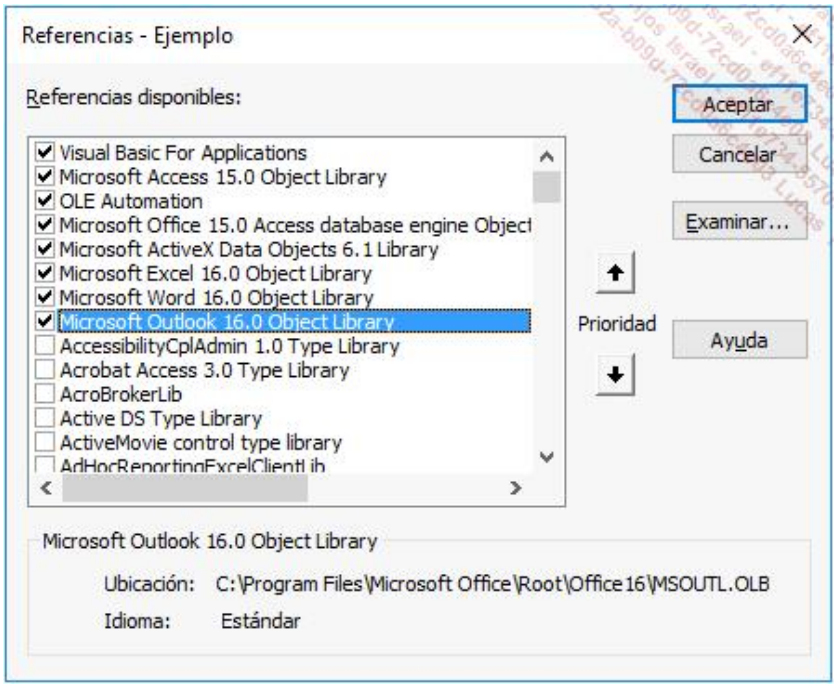
Escritura fila a fila

También es posible recorrer los registros uno por uno, y escribir el contenido de los campos en las celdas de una tabla.

```
Sub AlimentarWord()
Dim wdApp As Word.Application
Dim wddoc As Word.Document
Dim wdtabla As Word.Table
Dim RS As New ADODB.Recordset
Dim fld As Object
Dim i As Integer, j As Integer
RS.Open "SELECT TOP 10 * FROM ENI_ANIMAL_ANI ORDER BY
ANI_FECHA_NACIMIENTO DESC",
CurrentProject.Connection, adOpenDynamic, adLockOptimistic
Set wdApp = GetObject(, "Word.Application")
Set wddoc = wdApp.Documents.Add
Set wdtabla = wddoc.Tables.Add (wddoc.Range, numrows:=11,
numcolumns:=Rs.Fields.Count)
i = 2
With wdtabla
    Do Until RS.EOF
           For j = 1 To 10
                  .Cell(i, j).Range.Text = Nz(RS.Fields(j).Value,""
           Next j
           i = i +1
           RS.MoveNext
    Loop

'copia las etiquetas de columnas en la primera fila
    i = 1: j = 1
    For Each fld In RS.Fields
           .Cell(i, j).Range.Text = fld.Name
           j = j +1
    Next
End With
RS.Close
End Sub
```

Controlar Outlook

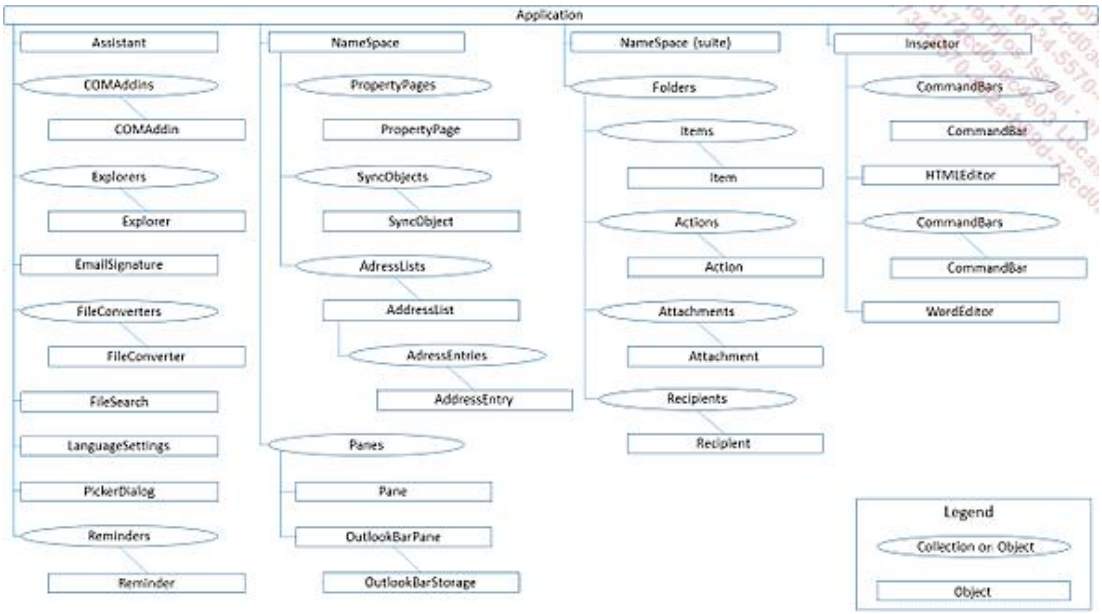
Para controlar la aplicación Outlook, es necesario añadir la librería **Microsoft Outlook 16.0 Object Library** a las referencias del proyecto (**Herramientas** - **Referencias**).



Si no encuentra esta librería en la lista disponible, puede hacer clic en **Examinar** y buscar el archivo MSOUTL.OLB en su máquina, que normalmente está en: C:\Program Files\Microsoft Office\Root\Office16\MSOUTL.OLB

1. Jerarquía de los objetos de Outlook

El siguiente esquema representa la jerarquía de los principales objetos y colecciones de la aplicación Outlook.



2. Lista de los principales objetos y colecciones de Outlook

Los principales objetos y colecciones que es posible manipular en Outlook son los siguientes.

a. La aplicación

El objeto `Application` de Outlook es el objeto en el modelo de Outlook, al mismo nivel que el objeto `Application` de Microsoft Access. Para no tener conflicto con los nombres, debe estar precedido del nombre de la aplicación. Por tanto, la variable que representa el objeto `Application` de Outlook se declarará como sigue:

```
Dim olApp As Outlook.Application
```


Observe que en este nivel del programa no se ha creado ninguna instancia de Outlook. No puede existir hasta que la variable `olApp` tenga valor. La siguiente sintaxis crea una nueva instancia de la aplicación Outlook durante la declaración de la variable:

```
Dim olApp As New Outlook.Application
```

Para no crear procesos zombis en la máquina, conviene salir de la aplicación Outlook con la instrucción:

```
olApp.Quit
```

Principales métodos

El objetivo de este libro no es ser exhaustivo en la explicación de la aplicación Outlook. A continuación se muestran los principales métodos que es interesante conocer para controlarla desde Microsoft Access.

Método	Descripción
`CreateItem`	Permite crear un nuevo objeto de Outlook (Reunión, Contacto, Mail y Tarea, entre de otras).
`Quit`	Permite salir de Outlook.

Principales propiedades

Propiedad	Descripción
`DefaultProfileName`	Permite obtener el nombre del perfil por defecto en Outlook.
`Session`	Permite manipular la información de la sesión actual en Outlook.

Ejemplo

El siguiente código crea una nueva aplicación de Outlook, la hace visible y después la cierra.

```
Dim olApp As New Outlook.Application
```

```
olApp.Visible = True
olApp.Quit
```

Para acceder a las carpetas, e-mails o incluso a los contactos y reuniones, se pasa por un objeto `NameSpace`, que representa la raíz de la herramienta Outlook. El `NameSpace` se recupera usando la función `GetNameSpace("MAPI")`.

b. Los e-mails MailItem

El objeto `MailItem` es el objeto que representa un e-mail de Outlook. Para declarar una variable de tipo `MailItem`, es posible la siguiente sintaxis:

```
Dim olmail As Outlook.MailItem
```

Principales métodos de MailItem

Método	Descripción
`Close`	Permite cerrar el e-mail.
`Delete`	Permite eliminar el e-mail.
`Forward`	Permite reenviar el e-mail.
`PrintOut`	Permite imprimir el e-mail.
`Reply`	Permite crear un mail de respuesta, poniendo como destinataria la persona que envió el e-mail original.
`ReplyAll`	Permite crear un e-mail de respuesta, poniendo como destinatarios todos los destinatarios originales del e-mail.
`Save`	Permite guardar el e-mail.
`SaveAs`	Permite guardar el e-mail como...
`Send`	Permite enviar el e-mail.

Principales propiedades

Propiedad	Descripción
`BCC`	Representa la lista de destinatarios en copia oculta del e-mail.
`Body`	Representa el cuerpo de texto del e-mail.
`CC`	Representa la lista de los destinatarios en copia del e-mail.
`CreationTime`	Representa la fecha y la hora de creación del e-mail.
`HTMLBody`	Representa el cuerpo HTML del e-mail.
`Importance`	Representa el nivel de importancia relativo del e-mail.
`ReadReceiptRequested`	Representa la presencia de confirmación de lectura solicitada por quien envía el e-mail.
`Sender`	Representa la cuenta desde la que se envía el e-mail.
`Subject`	Representa el asunto del e-mail.

To	Representa la lista de destinatarios del e-mail.

Principales colecciones

Colección	Descripción
Attachments	Representa la colección de los elementos adjuntos del e-mail.
ItemProperties	Representa la colección de las propiedades estándares del e-mail.
Recipients	Representa la colección de todos los destinatarios del e-mail.

Ejemplo

El siguiente código permite crear un e-mail vacío desde Access y mostrarlo:

```
Sub Ejemplo2()
Dim olApp As New Outlook.Application
Dim olMail As Outlook.MailItem
Set olMail = olApp.CreateItem(olMailItem)
olMail.Display
Set olApp = Nothing
End Sub
```

c. Los contactos ContactItem

El objeto `ContactItem` es el objeto que representa un contacto de Outlook. Para declarar una variable de tipo `ContactItem`, es posible la siguiente sintaxis:

```
Dim olContact As Outlook.ContactItem
```

Principales métodos de ContactItem

Método	Descripción
AddPicture	Permite añadir una imagen a un contacto.
Close	Permite cerrar y, llegado el caso, guardar las modificaciones añadidas al contacto.
Delete	Permite eliminar un contacto.
Display	Permite visualizar un contacto.
ForwardAsBusinessCard	Permite crear un e-mail y poner en el e-mail la información del contacto.
PrintOut	Permite imprimir la información del contacto.
RemovePicture	Permite eliminar una imagen de un contacto.
Save	Permite guardar la información del contacto.
ShowCheckAddressDialog	Permite visualizar el cuadro de diálogo **Comprobar dirección**, para comprobar los detalles de la dirección del contacto.

Principales propiedades

Propiedad	Descripción
Birthday	Representa la fecha de cumpleaños del contacto.
BusinessHomePage	Representa la dirección de la página web profesional del contacto.
Children	Representa los nombres de los hijos del contacto.
CompanyName	Representa el nombre de la empresa del contacto.
Email1Address	Representa la dirección de e-mail de la primera entrada de dirección de mensajería del contacto.
Email2Address	Representa la dirección de e-mail de la segunda entrada de dirección de mensajería del contacto.
FirstName	Representa el nombre del contacto.
LastName	Representa el apellido del contacto.
NickName	Representa el apodo del contacto.

Principales colecciones

Colección	Descripción
ItemProperties	Representa la colección de las propiedades estándares del contacto.

Ejemplo

El siguiente código permite crear un nuevo contacto, asignándole algunos atributos:

```
Sub CrearContacto()
    Dim olApp As New Outlook.Application
    Dim olContact As Outlook.ContactItem
    Set olContact = olApp.CreateItem(olContactItem)  'Creación
de un nuevo Contacto
    With olContact
           .LastName = "Sánchez"
           .FirstName = "Ángel María "
           .Email1Address = "mail@dominio.com"
           .Email1DisplayName = "Mail ficticio"
           .WebPage = "https://www.ediciones-eni.com//"
    End With
    'Ver el nuevo Objeto Contact
    olContact.Display
    Set olApp = Nothing
End Sub
```

d. Las reuniones AppointmentItem

El objeto AppointmentItem es el objeto que representa una reunión de Outlook. Para declarar una variable de tipo AppointmentItem, es posible la siguiente sintaxis:

```
Dim olReunion As Outlook.AppointmentItem
```

Principales métodos del objeto AppointmentItem

Método	Descripción
Close	Permite cerrar el elemento reunión.
Delete	Permite eliminar el elemento reunión.
Display	Permite visualizar la reunión.
Respond	Permite responder a una petición de reunión.
Save	Permite guardar el elemento reunión.
Send	Permite enviar la petición de reunión.

Principales propiedades

Propiedad	Descripción
AllDayEvent	Devuelve un booleano que indica si la reunión se desarrolla durante todo el día.
BusyStatus	Representa la información de disponibilidad del usuario para la reunión.
Duration	Representa la duración en minutos de la reunión.
End	Representa la fecha y la hora de fin de reunión.
EndUTC	Representa la fecha y la hora de fin de reunión en tiempo universal (*Universal Time Coordinated*).
Location	Representa la ubicación donde tiene lugar la reunión.
ResponseRequested	Devuelve un booleano que indica si el organizador desea recibir una respuesta a la petición de reunión.
Start	Representa la fecha y hora de inicio de la reunión.
StartUTC	Representa la fecha y la hora de inicio de la reunión en tiempo universal.
Subject	Representa el objeto de la reunión.

Principales colecciones

Colección	Descripción
Attachments	Representa la colección de los elementos adjuntos de la reunión.
ItemProperties	Representa la colección de las propiedades estándares de la reunión.
Recipients	Representa la colección de los destinatarios de la reunión.

e. Otras posibilidades

Además de los diferentes niveles de objetos que es posible controlar por la automatización de Outlook, el envío de e-mails automáticos usando programación VBA se puede realizar con `Docmd.SendObject` (tecnología CDO fundamentalmente).

Introducción

En un mundo que se abre cada más a Internet, Microsoft Access 2019 integra los principales estándares de comunicación en este canal, y los formatos XML y HTML están totalmente soportados. El objetivo de este capítulo es abordar el contenido XML como contenedor de datos, los archivos que lo completan, así como el código VBA que va a permitir manipularlo con mayor facilidad, para optimizar el almacenamiento y la transferencia de información dentro de las aplicaciones que desea implementar. También se tratan los archivos HTML y las consultas HTTP.

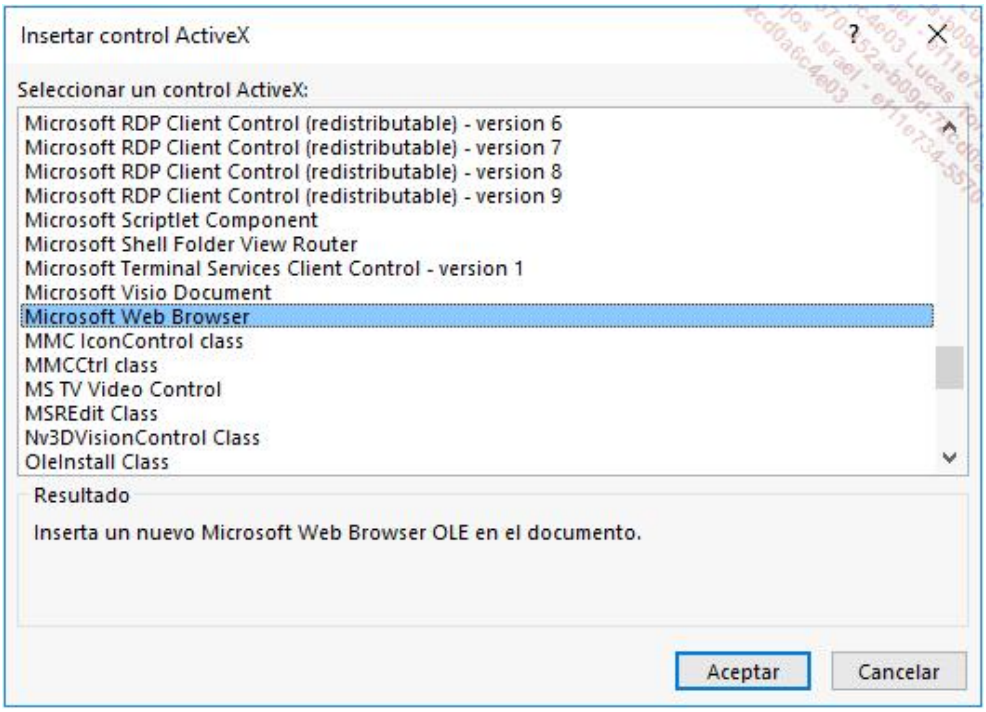
Access 2019 y la navegación web

Aunque las aplicaciones de Office inicialmente no estaban dedicadas a la navegación en la Web, sin embargo es posible establecer interfaces entre su aplicación e Internet para navegar en una red local o en línea.

1. Los controles de Internet

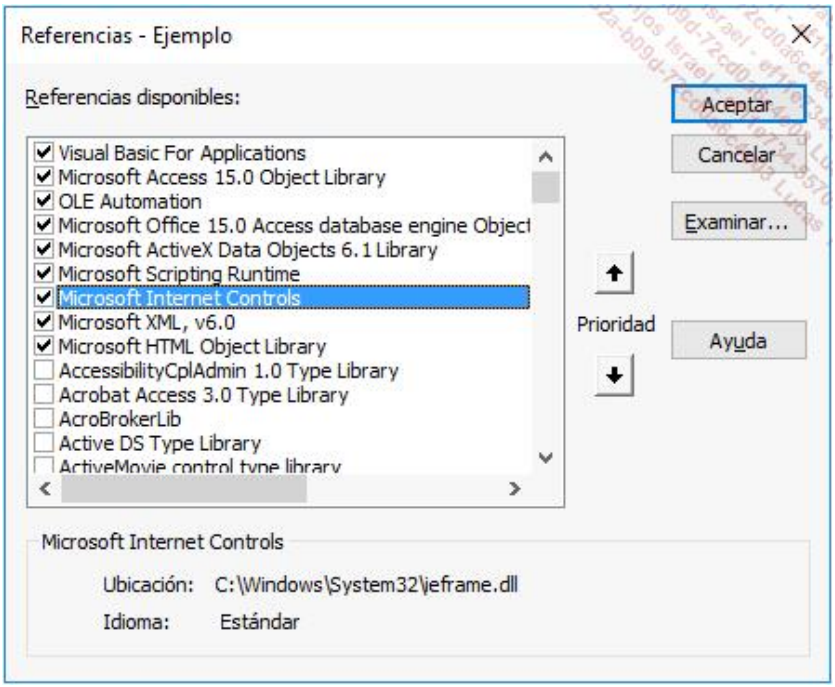
a. Control interno de la aplicación: WebBrowser

Es posible añadir un navegador web en un formulario. Para esto, es suficiente con ir a la pestaña **Diseño** del formulario en el que se desea añadir un navegador, y dentro del grupo Controles, pinche en el botón con punta de flecha inferior. Verá la opción **Controles ActiveX**, haga clic para que se habrá una ventana emergente, con los controles disponibles y seleccione **Microsoft Web Browser** haciendo clic en **OK**.



Se añade un navegador automáticamente al formulario. Observe que esta etapa desencadena dos eventos en la aplicación Microsoft Access 2019:

- La activación de la librería dedicada **Microsoft Internet Controls**:



- Aparece un icono **Control del navegador web** en el grupo **Controles** de la pestaña **Creación**. En la siguiente captura de pantalla, el icono **Control del navegador web** se resalta en el centro de la imagen.

Principales propiedades del control WebBrowser

Propiedad	Descripción
`BorderColor`	Representa el color del borde del control.
`Height`	Representa la altura del control en el formulario.
`Width`	Representa la anchura del control en el formulario.

Principales propiedades del control WebBrowser.Object

Para acceder al control `WebBrowser`, es necesario pasar por su propiedad `Object`. De hecho, el control solo está en el ámbito del objeto ActiveX que contiene el `WebBrowser`.

Propiedad	Descripción
`AddressBar`	Representa la barra de dirección del navegador (presente o no).
`Busy`	Indica si el navegador está ejecutando una tarea particular (carga de una página, etc.).
`Document`	Representa el documento web abierto en el navegador (página HTML u otro).
`LocationName`	Representa el título de la página web.
`LocationURL`	Representa la dirección URL de la página.
`Offline`	Indica si el navegador está en línea o no.

ReadyState	Representa el estado de disponibilidad del navegador.

Principales métodos del control WebBrowser.Object

Método	Descripción
Navigate	Permite navegar hacia una dirección.
GoHome	Permite navegar hacia la página de bienvenida.
GoBack	Permite navegar hacia la página anterior.
GoForward	Permite navegar hacia la página siguiente.
Refresh	Permite refrescar la página actual.

Ejemplos para controlar el WebBrowser

Para conectarse a una dirección durante la carga del formulario (aquí, el control `WebBrowser` con nombre `WB1`), el código será el siguiente:

```
Private Sub Form_Load()
    Me.WB1.Object.Navigate "http://www.ediciones-eni.com//"
End Sub
```

Para añadir en el formulario dos botones de comando, el primero `Btn_Google` y el segundo `Btn_Volver`, que sirvan respectivamente para ir a la dirección http://www.google.es y volver atrás, hay que implementar los eventos clic como sigue:

```
Private Sub Btn_Google_Click()
    Me.WB1.Object.Navigate "http://www.google.es/"
End Sub
Private Sub Btn_Volver_Click()
    Me.WB1.Object.GoBack
End Sub
```

b. Control externo de la aplicación: ejemplo de Internet Explorer

El control Microsoft Web Browser permite integrar directamente un control de Internet en la aplicación Microsoft Access 2019. También es posible proponer una navegación fuera de la aplicación. En la actualidad existen varios navegadores. Nuestro ejemplo se limitará a Internet Explorer, todavía muy extendido en las máquinas de los usuarios.

De la misma manera que se ha podido comprobar la adición automática de la librería **Microsoft Internet Controls** en la sección Control interno de la aplicación: WebBrowser, el uso del objeto `InternetExplorer` se va a hacer a partir de esta misma librería. Por tanto, conviene añadirla al proyecto antes de pasar al resto del programa.

El navegador Internet Explorer se define con la siguiente línea:

```
Dim Navegador As SHDocVw.InternetExplorer
```

La librería **Microsoft Internet Controls** tiene el nombre **SHDocVw** en VBA. Como se ha podido ver en el capítulo sobre Los objetos y colecciones en VBA. la variable no apunta actualmente a nada en concreto. Para crear un nuevo navegador, será necesario ejecutar la siguiente línea:

```
Set Navegador = New SHDocVw.InternetExplorer
```

Y para hacer visible el navegador, se cambia su propiedad `Visible` a `True`:

```
Navegador.Visible = True
```

Para salir del navegador, se usa el método `Quit` del objeto como sigue:

```
Navegador.Quit
```

El objeto `InternetExplorer` cuenta con los mismos métodos que el objeto `WebBrowser`. Por tanto, se podrán crear en un formulario tres botones de comando, llamados respectivamente `Btn_Ejecutar`, `Btn_Google` y `Btn_Salir`, que permitirán ejecutar Internet Explorer, visualizar la página Google y salir de Internet Explorer.

Por ejemplo, el código VBA será el siguiente:

```
Private Navegador As SHDocVw.InternetExplorer
Private Sub Btn_Ejecutar_Click()
    Set Navegador = New SHDocVw.InternetExplorer
    Navegador.Visible = True
    Navegador.Navigate " http://www.ediciones-eni.com/"
End Sub

Private Sub Btn_Google_2_Click()
    Navegador.Navigate " http://www.google.es"
End Sub

Private Sub Btn_Salir_Click()
    Navegador.Quit
End Sub
```

c. Los eventos de los controles

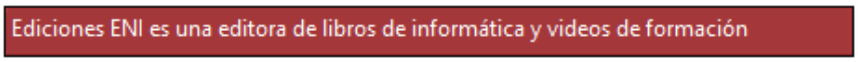
El control WebBrowser

Como el resto de los controles de Microsoft Access 2019, el control `WebBrowser` detecta y reacciona ante un determinado número de eventos. Por tanto, en la interfaz VBE es posible ver cómo aparece el objeto en la zona de la lista desplegable de los objetos.

Esto significa que es posible generar código VBA cuando estos eventos tienen lugar. Por ejemplo, cuando el título de la página del navegador cambia o aparece en la barra de estado.

```
Private Sub WB1_TitleChange(ByVal Text As String)
    Application.SysCmd acSysCmdSetStatus, Text
End Sub
```

Esto permite ver el siguiente mensaje cuando vamos a la dirección: http://www.ediciones-eni.com/.



El control InternetExplorer

Para que aparezca el administrador de los eventos, como es el caso del objeto `WebBrowser`, es suficiente con añadir la palabra clave `WithEvents` durante la declaración de la variable `Navegador` como sigue:

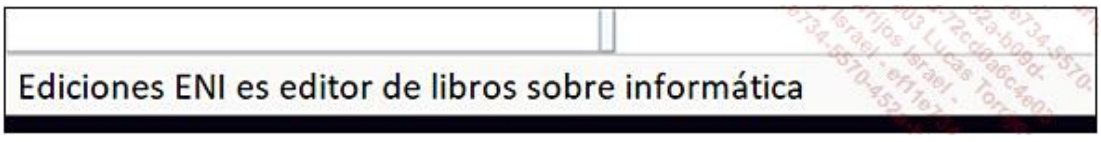
```
Private WithEvents Navegador As SHDocVw.InternetExplorer
```

Una vez que se haya añadido, podemos ver como el objeto `Navegador` aparece en la zona de lista desplegable de los objetos.

Por tanto, es posible generar código VBA cuando tienen lugar estos eventos. Por ejemplo, cuando el título de la página del navegador cambia, aparece en la barra de estado.

```
Private Sub Navegador_TitleChange(ByVal Text As String)
    Application.SysCmd acSysCmdSetStatus, Text
End Sub
```

Esto es lo que permite también ver un mensaje idéntico cuando se va a la dirección http://www.ediciones-eni.com/.



2. Las librerías de VBA que se pueden utilizar

Además de la librería **Microsoft Internet Controls**, que se ha visto en la sección Los controles de Internet, a continuación se muestran las otras dos principales librerías que se pueden utilizar para manipular los objetos Internet.

a. Microsoft XML

La librería **Microsoft XML** está disponible en varias versiones, con el nombre msxml6.dll para la versión 6 por ejemplo, que en la actualidad es la versión más reciente disponible. Está dedicada especialmente al uso de XML.

Los objetos y métodos relacionados con esta librería están accesibles desde el objeto `MSXML2` en VBA.

A continuación se muestra un ejemplo de carga de archivo XML, con una visualización del número de caracteres en el contenido XML del archivo:

```
Sub EjemploXML()
    Dim archivo As MSXML2.DOMDocument60
    Set archivo = New MSXML2.DOMDocument60
    archivo.Load "C:\temp\ENI_VACUNA_VAC.xml"
    Debug.Print Len(archivo.Text)
End Sub
```

b. Microsoft HTML Object Library

La librería **Microsoft HTML Object Library** permite manipular los objetos del modelo HTML (*HyperText Markup Language*), como tablas, formularios, documentos, etc.). Se utiliza adicionalmente a la librería **Microsoft XML** o **Microsoft Internet Controls**.

Se puede acceder a los objetos y métodos relacionados con esta librería, con el objeto `MSHTML` en VBA.

Para acceder a los diferentes elementos de una página HTML, podemos pasar por su ID único. El código de ejemplo va a abrir la página www.twitter.com y escribir en la ventana **Buscar** las palabras clave "VBA Access 2019":

```
#If VBA7 Then
    Public Declare PtrSafe Sub Sleep Lib "kernel32"
(ByVal dwMilliseconds As LongPtr) 'Para los sistemas 64 Bits
#Else
    Public Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds
As Long) 'Para los sistemas 32 Bits
#End If
Sub EjemploTwitter()
'Declaración
Dim IE As New InternetExplorer
Dim IEDoc As HTMLDocument
Dim InputTwitterBusqueda As HTMLInputElement
    'Carga de una página web Twitter
    IE.Navigate "www.twitter.com"
    'Ver la ventana IE
    IE.Visible = True
    Do While IE.ReadyState = 4: Sleep 100: Loop
    Do While IE.ReadyState <> 4: Sleep 100: Loop
    'Loop
    'Se apunta al miembro Document
    Set IEDoc = IE.Document
    'Se apunta a la zona de búsqueda
    Set InputTwitterBusqueda = IEDoc.all("q")
    'Se escribe el texto deseado
```

```
    InputTwitterBusqueda.Value = "VBA Access 2019"
    Set IE = Nothing
    'Se comprueba la zona de búsqueda, se visualiza si no está
visible, se muestra el texto
End Sub
```

3. Las consultas HTTP

Esta parte del capítulo trata de la realización de consultas HTTP (*HyperText Transfer Protocol*) y la recuperación de respuestas en VBA. Se trata de usar un navegador de Internet y ejecutar directamente instrucciones y consultas en VBA. El uso de la librería **MSXML2** va a permitir manipular las consultas HTTP.

a. Abrir una conexión HTTP

Para abrir una conexión, podemos usar la siguiente sintaxis general:

```
Sub EjemploHTTP()
Dim oHTTP As MSXML2.XMLHTTP60
Set oHTTP = New MSXML2.XMLHTTP60
oHTTP.Open "POST", "http://www.ediciones-eni.com/", False
End Sub
```

Observe que, entre las propiedades y métodos del objeto `XMLHTTP60`, podemos citar las siguientes:

Propiedad/método	Descripción
`Abort`	Permite anular la consulta HTTP.
`getAllResponseHeaders`	Permite recuperar el conjunto de las cabeceras de las respuestas de las consultas.
`Open`	Permite abrir una conexión para ejecutar una consulta HTTP.
`send`	Permite ejecutar una consulta HTTP.
`setRequestHeader`	Permite personalizar la cabecera de la consulta HTTP.
`readystate`	Representa el estado de disponibilidad de la consulta.
`responseBody`	Representa el cuerpo de la respuesta de la consulta.
`responseText`	Representa el cuerpo de la respuesta en forma de cadena de caracteres.
`status`	Representa el estado de la consulta.
`timeout`	Representa la duración a partir de la cual la consulta se considerará errónea si no se devuelve ninguna respuesta.
`withCredentials`	Representa la información de los usuarios que se proporciona para realizar la consulta (identificador/contraseña).

b. Descargar un archivo

Si se desea descargar un archivo, se puede abrir una conexión directamente en el archivo para recuperar su contenido. También será necesario utilizar el método `GET` en lugar de `POST` para conseguirlo.

La respuesta incluye el contenido del archivo que se desea descargar, por lo que es necesario prever un sistema para almacenar la información de la respuesta. Por ejemplo, el siguiente código permite almacenar el archivo del logo de ediciones ENI, disponible en la siguiente dirección: http://www.ediciones-eni.com/styles/espagne/images/logo_ENI.png

```
Function DescargarImageHTTP()
Dim oHTTP As MSXML2.XMLHTTP60
Dim archivo As Integer
Dim Matriz_Imagen() As Byte
Set oHTTP = New MSXML2.XMLHTTP60
oHTTP.Open "GET", _
    " http://https://www.ediciones-eni.com/styles/espagne/images/
logo_ENI.png ", False
oHTTP.send
'Status OK?
If oHTTP.Status = 200 Then
    archivo = FreeFile
    Open "c:\temp\mi_logo_ENI.png" For Binary As #archivo
    Matriz_Imagen = oHTTP.responseBody
    Put #archivo, , Matriz_Imagen
    Erase Matriz_Imagen
    Close #archivo
End If
'Ver el archivo
End Function
```

Access 2019 y el formato XML

El formato XML ya se ha mencionado en el marco del capítulo Optimizar las interfaces de Access - Ejemplo de cinta de opciones personalizada. En este capítulo, vamos a abordar su uso centrado en su capacidad de almacenar datos que se utilizarán en Access 2019, pero veremos también cómo va a ser posible saber más de su manejo en VBA.

1. El formato XML y los datos

a. El formato XML

El lenguaje XML se deriva del HTML, pero su objetivo principal es la administración de los datos en la Web. El sistema de etiquetas que utiliza se define por el creador del archivo, lo que simplifica mucho su lectura.

La estructura de un archivo XML siempre empieza de esta manera:

```
<?xml version="1.0" encoding="UTF-8"?>
```

O incluso:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

Podemos leer la versión del XML (1.0), así como el modo de codificación del archivo.

Hay una versión 1.1 de XML, pero todavía no está lo suficientemente soportada por los diferentes actores de la Web para imponerse como versión de referencia.

Más abajo se muestra un ejemplo de archivo XML, resultado de una tabla:

```
<?xml version="1.0" encoding="UTF-8"?>
<dataroot xmlns:od="urn:schemas-microsoft-com:officedata"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="ENI_VACUNA_VAC.xsd">
ENI_VACUNA_VAC>
<VAC_ID>1</VAC_ID>
<VAC_VACUNA>Parvovirosis del perro </VAC_VACUNA>
<VAC_ANT_ID>1</VAC_ANT_ID>
</ENI_VACUNA_VAC>
<ENI_VACUNA_VAC>
<VAC_ID>2</VAC_ID>
<VAC_VACUNA>Moquillo del perro </VAC_VACUNA>
<VAC_ANT_ID>1</VAC_ANT_ID>
</ENI_VACUNA_VAC>
</dataroot>
```

Podemos ver que el conjunto de etiquetas comienza con < y termina con >, así como los diferentes campos de la tabla `ENI_VACUNA_VAC`. Dentro de las etiquetas podemos leer información, como por ejemplo `Parvovirosis del perro` para el campo `VAC_VACUNA`. También podemos ver dentro de una de las etiquetas la información:

```
xsi:noNamespaceSchemaLocation=" ENI_VACUNA_VAC.xsd"
```

Por tanto, ¿a qué hace referencia `ENI_VACUNA_VAC.xsd`?

b. El formato XSD

El formato XSD, de *XML Schema Description*, es un complemento y una necesidad que valida los datos de XML. Relacionado con un archivo XSD, el XML no podrá contener otro tipo de datos diferente al previsto por el diseñador.

A continuación, el archivo XSD que se refiere al archivo XML presentado en el ejemplo anterior:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:od="urn:schemas
-microsoft-com:officedata">
<xsd:element name="dataroot">
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="ENI_VACUNA_VAC" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="generated" type="xsd:dateTime"/>
</xsd:complexType>
</xsd:element>
<xsd:element name="ENI_VACUNA_VAC">
<xsd:annotation>
<xsd:appinfo>
<od:index index-name="PrimaryKey" index-key="VAC_ID " primary="yes"
unique="yes" clustered="no" order="asc"/>
<od:index index-name="VAC_ANT_ID" index-key="VAC_ANT_ID " primary="no"
unique="no" clustered="no" order="asc"/>
<od:index index-name="VAC_ID" index-key="VAC_ID " primary="no"
unique="no"
clustered="no" order="asc"/>
<od:tableProperty name="Orientation" type="2" value="0"/>
<od:tableProperty name="OrderByOn" type="1" value="0"/>
<od:tableProperty name="NameMap" type="11"
value="CswOVQAAAACP3ZOVRAolTJCz9fzCHAq7AAAAAGmoAjSaL+VAAAAAAAAAAABFAE4A
SQBfAFYAQQBDAEMASQBOAF8AVgBBAEMAAAAAAAAAjKmJn5KvK06OVHa9FbksDQcA
AACP3ZOVRAolTJCz9fzCHAq7VgBBAEMAXwBJAEQAAAAAAAAAuJSeK35VlkOsrw2N
ukoyngcAAACP3ZOVRAolTJCz9fzCHAq7VgBBAEMAXwBWAEEAQwBDAEkATgAAAAAA
AAAJFAtOO/QsSa2VWA+5MS5qBwAAAI/dk5VECiVMkLP1/MIcCrtWAEEAQwBfAEEA
TgBUAF8ASQBEAAAAAAAAAOg33+rbMIhPjQJbHBCB92wHAAAAj92TlUQKJUyQs/X8
whwKu1YAQQBDAF8ATwBCAEwASQBHAEEAVABPAEkAUgBFAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAMAAAABQAAAAAAAAAAAAAAAAAAAAA
"/>
<od:tableProperty name="DefaultView" type="2" value="2"/>
<od:tableProperty name="DisplayViewsOnSharePointSite" type="2"
value="1"/>
<od:tableProperty name="TotalsRow" type="1" value="0"/>
<od:tableProperty name="FilterOnLoad" type="1" value="0"/>
```

```
<od:tableProperty name="OrderByOnLoad" type="1" value="1"/>
<od:tableProperty name="HideNewField" type="1" value="0"/>
<od:tableProperty name="BackTint" type="6" value="100"/>
<od:tableProperty name="BackShade" type="6" value="100"/>
<od:tableProperty name="ThemeFontIndex" type="4" value="1"/>
<od:tableProperty name="AlternateBackThemeColorIndex" type="4"
value="1"/>
<od:tableProperty name="AlternateBackTint" type="6" value="100"/>
<od:tableProperty name="AlternateBackShade" type="6" value="95"/>
<od:tableProperty name="ReadOnlyWhenDisconnected" type="1" value="0"/>
<od:tableProperty name="DatasheetGridlinesThemeColorIndex" type="4"
value="3"/>
<od:tableProperty name="DatasheetForeThemeColorIndex" type="4"
value="0"/>
<od:tableProperty name="PublishToWeb" type="2" value="1"/>
<od:tableProperty name="GUID" type="9" value="j92TlUQKJUyQs/X8whwKuw==">/>
</xsd:appinfo>
</xsd:annotation>
<xsd:complexType>
<xsd:sequence>
<xsd:element name="VAC_ID" minOccurs="1" od:jetType="autonumber"
od:sqlSType="int" od:autoUnique="yes" od:nonNullable="yes"
type="xsd:int">
<xsd:annotation>
<xsd:appinfo>
<od:fieldProperty name="ColumnWidth" type="3" value="-1"/>
<od:fieldProperty name="ColumnOrder" type="3" value="0"/>
<od:fieldProperty name="ColumnHidden" type="1" value="0"/>
<od:fieldProperty name="TextAlign" type="2" value="0"/>
<od:fieldProperty name="AggregateType" type="4" value="-1"/>
<od:fieldProperty name="ResultType" type="2" value="0"/>
<od:fieldProperty name="CurrencyLCID" type="4" value="0"/>
<od:fieldProperty name="GUID" type="9" value="jKmJn5KvK06OVHa9FbksDQ==">/>
</xsd:appinfo>
</xsd:annotation>
</xsd:element>
<xsd:element name="VAC_VACUNA" minOccurs="0" od:jetType="text"
od:sqlSType="nvarchar">
<xsd:annotation>
<xsd:appinfo>
<od:fieldProperty name="ColumnWidth" type="3" value="2115"/>
<od:fieldProperty name="ColumnOrder" type="3" value="0"/>
<od:fieldProperty name="ColumnHidden" type="1" value="0"/>
<od:fieldProperty name="Required" type="1" value="0"/>
<od:fieldProperty name="AllowZeroLength" type="1" value="1"/>
<od:fieldProperty name="DisplayControl" type="3" value="109"/>
<od:fieldProperty name="IMEMode" type="2" value="0"/>
<od:fieldProperty name="IMESentenceMode" type="2" value="3"/>
<od:fieldProperty name="UnicodeCompression" type="1" value="1"/>
<od:fieldProperty name="TextAlign" type="2" value="0"/>
<od:fieldProperty name="AggregateType" type="4" value="-1"/>
<od:fieldProperty name="ResultType" type="2" value="0"/>
<od:fieldProperty name="CurrencyLCID" type="4" value="0"/>
<od:fieldProperty name="GUID" type="9" value="uJSeK35VlkOsrw2Nukoyng==">/>
</xsd:appinfo>
```

```
</xsd:annotation>
<xsd:simpleType>
:restriction base="xsd:string">
<xsd:maxLength value="255"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="VAC_ANT_ID" minOccurs="0" od:jetType="longinteger"
od:sqlSType="int" type="xsd:int">
<xsd:annotation>
<xsd:appinfo>
<od:fieldProperty name="ColumnWidth" type="3" value="-1"/>
<od:fieldProperty name="ColumnOrder" type="3" value="0"/>
<od:fieldProperty name="ColumnHidden" type="1" value="0"/>
<od:fieldProperty name="DecimalPlaces" type="2" value="255"/>
<od:fieldProperty name="DefaultValue" type="12" value="0"/>
<od:fieldProperty name="Required" type="1" value="0"/>
<od:fieldProperty name="DisplayControl" type="3" value="109"/>
<od:fieldProperty name="TextAlign" type="2" value="0"/>
<od:fieldProperty name="AggregateType" type="4" value="-1"/>
<od:fieldProperty name="ResultType" type="2" value="0"/>
<od:fieldProperty name="CurrencyLCID" type="4" value="0"/>
<od:fieldProperty name="GUID" type="9" value="CRQLTjv0LEmtlVgPuTEuag==">
</xsd:appinfo>
</xsd:annotation>
</xsd:element>
<xsd:element name="VAC_OBLIGATORIA" minOccurs="1" od:jetType="yesno"
od:sqlSType="bit" od:nonNullable="yes" type="xsd:boolean">
<xsd:annotation>
<xsd:appinfo>
<od:fieldProperty name="ColumnWidth" type="3" value="-1"/>
<od:fieldProperty name="ColumnOrder" type="3" value="0"/>
<od:fieldProperty name="ColumnHidden" type="1" value="0"/>
<od:fieldProperty name="Format" type="10" value="Yes/No"/>
<od:fieldProperty name="DefaultValue" type="12" value="No"/>
<od:fieldProperty name="DisplayControl" type="3" value="106"/>
<od:fieldProperty name="TextAlign" type="2" value="0"/>
<od:fieldProperty name="AggregateType" type="4" value="-1"/>
<od:fieldProperty name="ResultType" type="2" value="0"/>
<od:fieldProperty name="CurrencyLCID" type="4" value="0"/>
<od:fieldProperty name="GUID" type="9" value="6Dff6tswiE+NAlscEIH3bA==">
</xsd:appinfo>
</xsd:annotation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

Para cada uno de los elementos, podemos ver sus detalles, como por ejemplo el campo `VAC_VACUNA`, de tipo Texto, correspondiente al tipo SQL `nvarchar`. Podemos ver la estructura de la tabla que se ha exportado.

c. El formato XSL

El formato XSL, de *eXtensible Stylesheet Language*, sirve para formatear los datos tal y como aparecen en el navegador web. Tenemos hojas de estilo típicas de los formatos recientes de páginas web.

Por ejemplo, el contenido del XSL podría ser el siguiente:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:msxsl="urn:schemas-microsoft-com:xslt" xmlns:fx="#fx-
functions" exclude-result-prefixes="msxsl fx">
    <xsl:output method="html" version="4.0" indent="yes"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"/>
    <xsl:template match="//dataroot"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
          <html>
                <head>
                      <META HTTP-EQUIV="Content-Type"
CONTENT="text/html;charset=UTF-8"/>
                      <title>ENI_VACUNA_VAC</title>
                      <style type="text/css"></style>
                </head>
                <body link="#0c0000" vlink="#050000">
                      <table border="1" bgcolor="#ffffff"
cellspacing="0" cellpadding="0" id="CTRL1">
                            <colgroup>
                                  <col style="TEXT-ALIGN:
right; WIDTH: 2.38cm"/>
                                  <col style=" WIDTH: 3.73cm"/>
                                  <col style="TEXT-ALIGN:
right; WIDTH: 2.38cm"/>
                                  <col style=" TEXT-ALIGN:
right;WIDTH: 2.38cm"/>
                            </colgroup>
                            <tbody>
                                  <tr>
                                        <td>
                                             <div align="center">
    <strong>VAC_ID</strong>
                                             </div>
                                        </td>
                                        <td>
                                             <div align="center">
    <strong>VAC_VACUNA</strong>
                                             </div>
                                        </td>
                                        <td>
                                             <div align="center">
    <strong>VAC_ANT_ID</strong>
                                             </div>
                                        </td>
                                        <td>
                                             <div align="center">
    <strong>VAC_OBLIGATORIA</strong>
                                             </div>
```

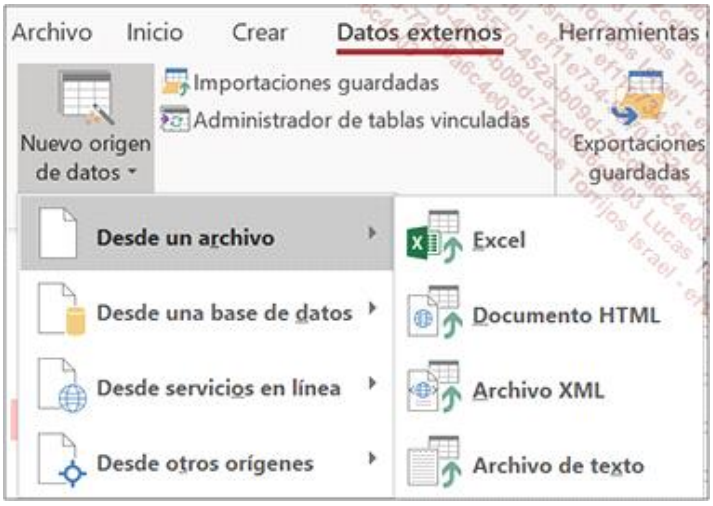
```
                                        </td>
                                    </tr>
                                </tbody>
                                <tbody id="CTRL2">
                                <xsl:for-each se-lect="ENI_VACUNA_VAC">
                                        <!-- Cache the current
node incase the a field is formatted -->
                                        <xsl:value-of
select="fx:CacheCurrentNode(.)"/>
                                         <tr>
                                                <td>
    <xsl:value-of select="VAC_ID"/>
                                                </td>
                                                <td>
    <xsl:value-of select="VAC_VACUNA"/>
                                                </td>
                                                <td>
    <xsl:value-of select="VAC_ANT_ID"/>
                                                     </td>
                                                      <td>
    <xsl:value-of select="fx:FormatFromXSL('VAC_OBLIGATORIA',
                                    'Yes/No', '', '', 11)"/>
                                                </td>
                                        </tr>
                                    </xsl:for-each>
                                </tbody>
                            </table>
                    </body>
            </html>
```

2. Access y los datos XML

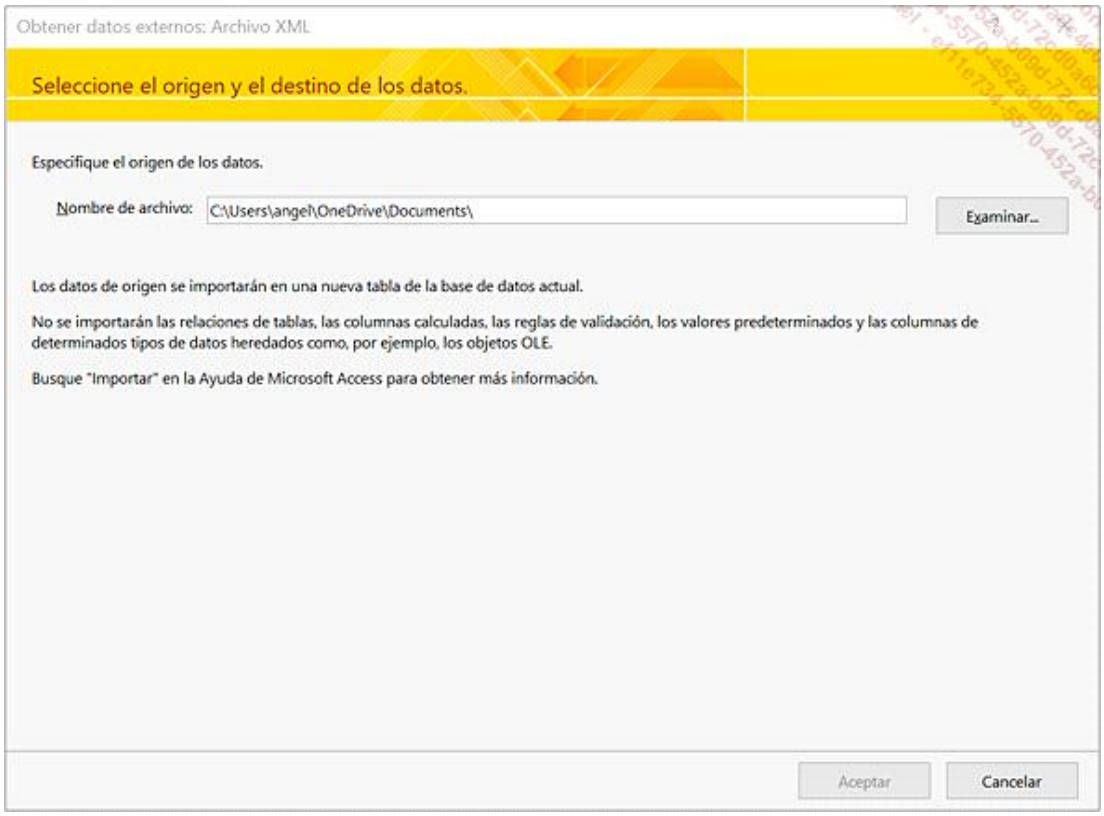
En la interfaz de Microsoft Access 2019, es posible importar y exportar de manera muy sencilla los archivos que se han mencionado en este capítulo.

a. Importar los datos en la interfaz

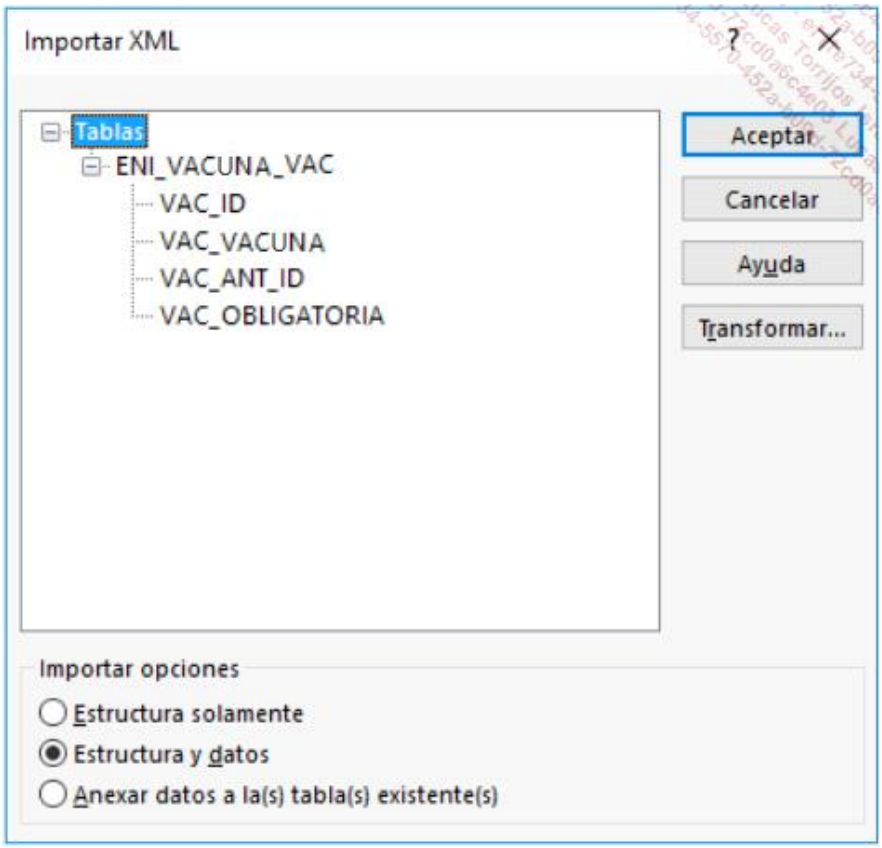
Para importar los datos al formato XML, podemos pasar por la pestaña **Datos externos** de Microsoft Access 2019 y seleccionar el icono **Archivo XML** en el menú **Datos externos - Nuevo origen de datos - Desde un archivo - Documento XML**.



Se abre la interfaz de selección del archivo.



Pulse en el botón **Examinar** para abrir el cuadro de diálogo de búsqueda del archivo. Selecciona el archivo XML que desea importar. Haciendo clic en **Aceptar**, se abre la siguiente ventana:

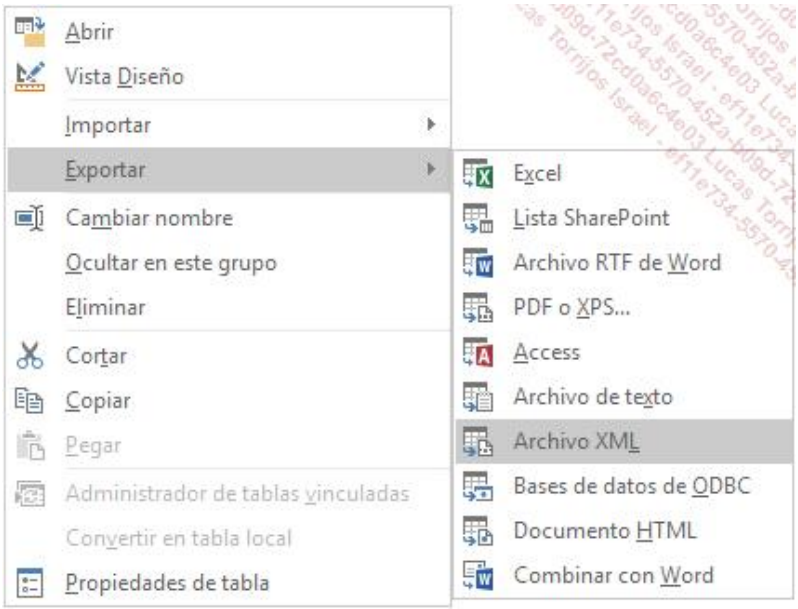


Se puede indicar lo que se desea importar (solo la estructura, la estructura y los datos), o indicar que deseamos añadir los datos en una tabla ya existente en la base de datos). Haciendo clic en **Aceptar**, se importan los datos.

b. Exportar los datos en la interfaz

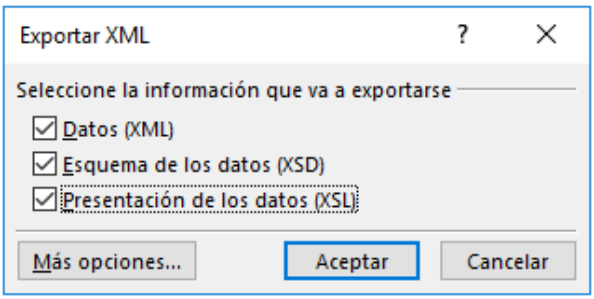
De la misma manera que es posible importar los datos en formato XML, es posible exportarlos.

Haciendo clic derecho en el objeto que se desea exportar, seleccione **Exportar - Archivo XML**.

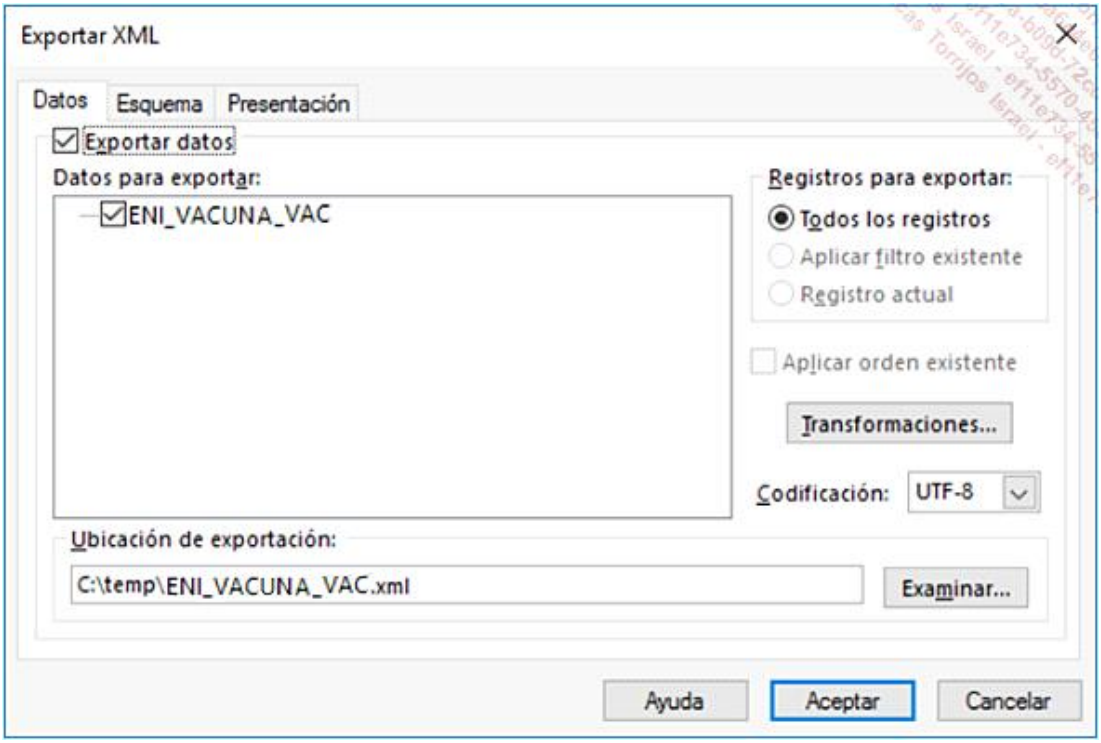


Seleccione la ubicación del archivo y pulse en el botón **Aceptar**.

Se abre la ventana de especificación de exportación de XML, que nos invita a seleccionar la información a exportar (XML, XSD o XSL).



Haciendo clic en el botón **Más opciones**, aparece la siguiente ventana de opciones:



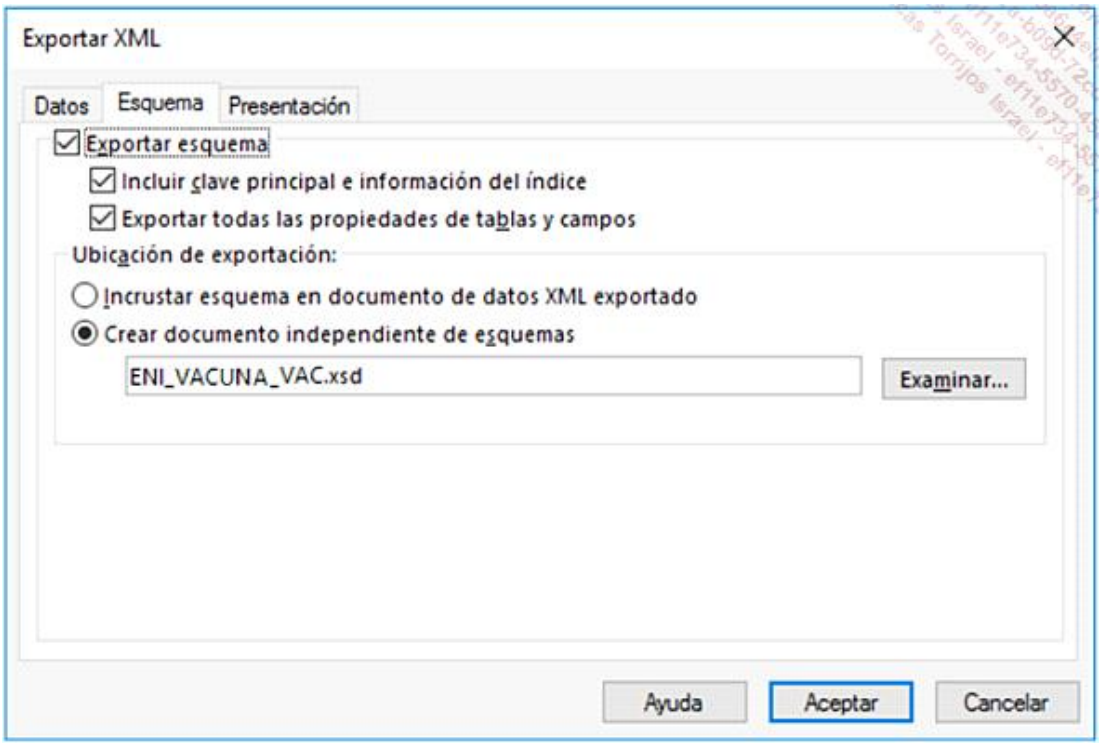
Se puede acceder a los siguientes elementos:

Pestaña Datos (del archivo XML)

Elemento	Descripción
Exportar datos	Permite indicar si se exportan los datos o solo la estructura.
Datos para exportar	Permite definir la lista de los datos que se han de exportar.
Registros para exportar	Permite indicar si se exportan todos los registros de la tabla o si se debe guardar el filtro aplicado durante la exportación.
Aplicar orden existente	Permite indicar si se debe guardar el orden aplicado en la tabla durante la exportación.
Transformaciones	Permite especificar si se debe aplicar un formateo ya existente (hoja XSL).

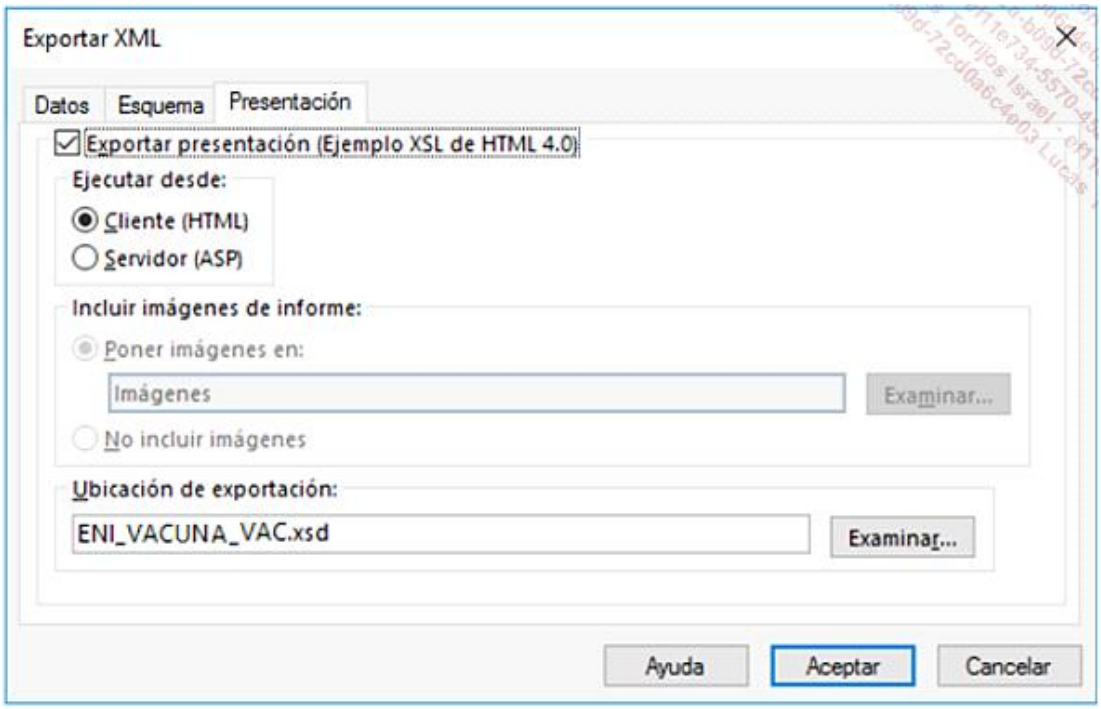
Codificación	Permite indicar el encoding que se utilizará (UTF-8 o UTF-16).
Ubicación de exportación	Permite definir la ubicación del archivo XML que se exportará.

Pestaña Esquema (del archivo XSD)



Elemento	Descripción
Exportar esquema	Permite indicar si se exporta el esquema de los datos.
Incluir clave principal	Permite especificar si la clave primaria y los datos de indexación deben aparecer en el archivo XSD.
Exportar todas las propiedades	Permite especificar si todas las propiedades deben aparecer en el archivo XSD.
Ubicación de exportación	Permite indicar si los datos del esquema se incorporan al archivo XML o si deben aparecer en un archivo XSD aparte.

Pestaña Presentación (del archivo XSL)



Elemento	Descripción
Exportar presentación	Permite indicar si se ha creado un archivo XSL durante la exportación.
Ejecutar desde	Permite seleccionar el modo de ejecución del archivo XSL durante la apertura del archivo XML.
Incluir imágenes de informe	Permite indicar la ubicación de las imágenes (se aplica para la exportación de un informe).
Ubicación de exportación	Permite indicar la ubicación del archivo XSL que se genera durante la exportación.

Si no se ha añadido ninguna modificación, pulse en **Cancelar** o en **Aceptar** para validar las modificaciones. La ventana se cierra y se regresa a la ventana **Exportar XML**.

Una vez que se hace clic en **Aceptar**, los datos se exportan a la ubicación indicada.

VBA y el formato XML

Los aspectos de la operación que se han visto en la sección Access y los datos XML se pueden tratar directamente en VBA.

1. La importación de datos XML

Para importar los datos XML, se usa el método `ImportXML` disponible con el objeto `Application`. La sintaxis general de este método es la siguiente:

```
Application.ImportXML OrigenDatosXML, [OpcionesdeImportación]
```

La información `OrigenDatosXML` es una cadena de caracteres que indica la ubicación del archivo XML que se debe importar. El argumento opcional `OpcionesImportación` permite indicar si la importación solo afecta a la estructura (`acStructureOnly`), la estructura y los datos (`acStructureAndData`) o la adición en una tabla existente (`acAppendData`). En caso de que el nombre de tabla ya se esté utilizando, Microsoft Access 2019 genera automáticamente un nuevo nombre de tabla.

Por ejemplo, para importar datos a una nueva tabla a partir del archivo XML ubicado en la carpeta C:\temp de la máquina, el código VBA deberá ser el siguiente:

```
Sub Ejemplo1()
Application.ImportXML "C:\temp\ENI_VACUNA_VAC.xml", _
    acStructureAndData
End Sub
```

Permite importar los datos en una tabla `ENI_VACUNA_VAC`:

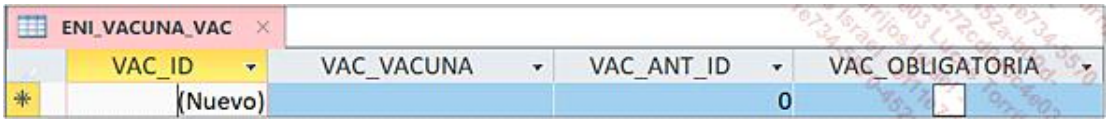
ENI_VACUNA_VAC

VAC_ID	VAC_VACUNA	VAC_ANT_ID	VAC_OBLIGATORIA
1	Parvovirosis del perro	1	☑
2	Moquillo del chien	1	☑
3	Rabia	2	☑
(Nuevo)		0	☐

Si solo se quiere la estructura de la tabla, el código será el siguiente:

```
Sub Ejemplo2()
Application.ImportXML "C:\temp\ENI_VACUNA_VAC.xml", acStructureOnly
End Sub
```

Y el resultado sería el siguiente:



ENI_VACUNA_VAC

VAC_ID	VAC_VACUNA	VAC_ANT_ID	VAC_OBLIGATORIA
(Nuevo)		0	☐

2. La exportación de datos XML

Para exportar datos XML en VBA, podemos llamar al método `ExportXML` del objeto `Application`. La sintaxis general de este método es la siguiente:

```
Application.ExportXML(ObjectType, SourceData, [DataTarget],
[SchemaTarget], [PresentationTarget], [ImageTarget], [Encoding],
[OtherFlags], [WhereCondition], [AdditionalData])
```

Los argumentos son los siguientes:

Argumento	Tipo	Descripción
`ObjectType`	Constante Access (`AcExportXMLObjectType`)	Representa el tipo de objeto que se va a exportar, ver la tabla de las constantes en el anexo de este libro.
`SourceData`	`String`	Representa el nombre del objeto que se va a exportar. Por defecto, apunta al objeto activo de tipo `ObjectType` especificado.
`DataTarget`	`String`	Representa la ubicación del archivo XML que se va a generar (se debe indicar para que la exportación tenga lugar).
`SchemaTarget`	`String`	Representa la ubicación del archivo XSD que se va a generar.
`PresentationTarget`	`String`	Representa la ubicación del archivo XSL que se va a generar.
`ImageTarget`	`String`	Representa la ubicación a la que se van a exportar las imágenes. Si no se rellena, las imágenes no se exportarán.
`Encoding`	Constante Access (`AcExportXMLEncoding`)	Representa el tipo de encoding del archivo (`acUTF8` - es el valor por defecto - o `acUTF16`).
`OtherFlags`	Constante Access (`AcExportXMLOtherFlags`)	Representa el resto de las opciones de exportación, vistas en la sección Exportar los datos HTML en la interfaz. Ver la lista de las constantes en el anexo de este libro.
`WhereCondition`	`String`	Representa las cláusulas que permiten filtrar los datos que se van a exportar a partir del objeto principal.
`AdditionalData`	`Variant`	Representa la lista del resto de los objetos que se pueden exportar al mismo tiempo que el objeto SourceData especificado.

Por ejemplo, para exportar el conjunto de archivos desde la tabla `ENI_ VACUNA_VAC`, el código VBA podría ser el siguiente:

```
Public Sub Export ENI_VACUNA_VAC XML()
Application.ExportXML acExportTable, "ENI_VACUNA_VAC", _
    "C:\temp\ENI_VACUNA_VAC.xml", "C:\temp\ ENI_VACUNA_VAC.xsd", _
    "C:\temp\ ENI_VACUNA_VAC.xsl"
End Sub
```

Si se desea exportar solo las vacunas para perros que ya se han tenido lugar, se añade esta información al argumento `WhereCondition` como sigue:

```
Public Sub ExportENI_VACUNA_VACXML2()
Application.ExportXML acExportTable, "ENI_VACUNA_VAC", _
    "C:\temp\ENI_VACUNA_VAC.xml", "C:\temp\ ENI_VACUNA_VAC.xsd", _
    "C:\temp\ ENI_VACUNA_VAC.xsl", _
WhereCondition:="VAC_ANI_ID=1"
End Sub
```

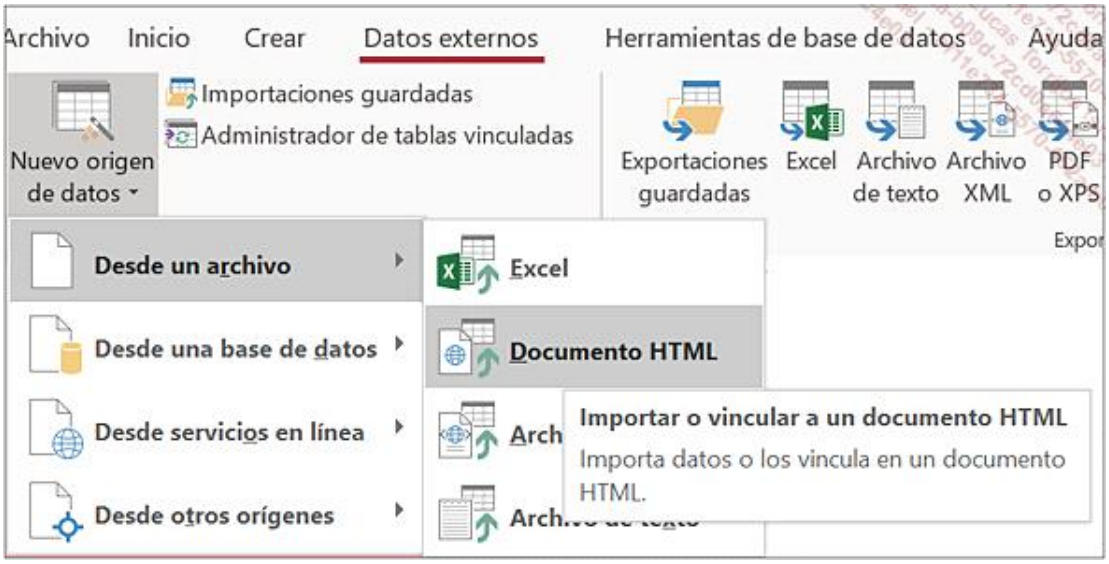
Si se desea exportar varias tablas (caso más frecuente), es posible añadir elementos en el argumento `AdditionalData` como sigue:

```
Public Sub ExportTablasXML()
Dim colAD As AdditionalData
'creación de AdditionalData
Set colAD = Application.CreateAdditionalData
'Añadir la tabla ENI_ANIMAL_ANI
colAD.Add "ENI_ANIMAL_ANI"
Application.ExportXML acExportTable, "ENI_VACUNA_VAC", _
    "C:\temp\ ENI_VACUNA_VAC.xml", "C:\temp\ ENI_VACUNA_VAC.xsd", _
    "C:\temp\ ENI_VACUNA_VAC.xsl", AdditionalData:=colAD
End Sub
```

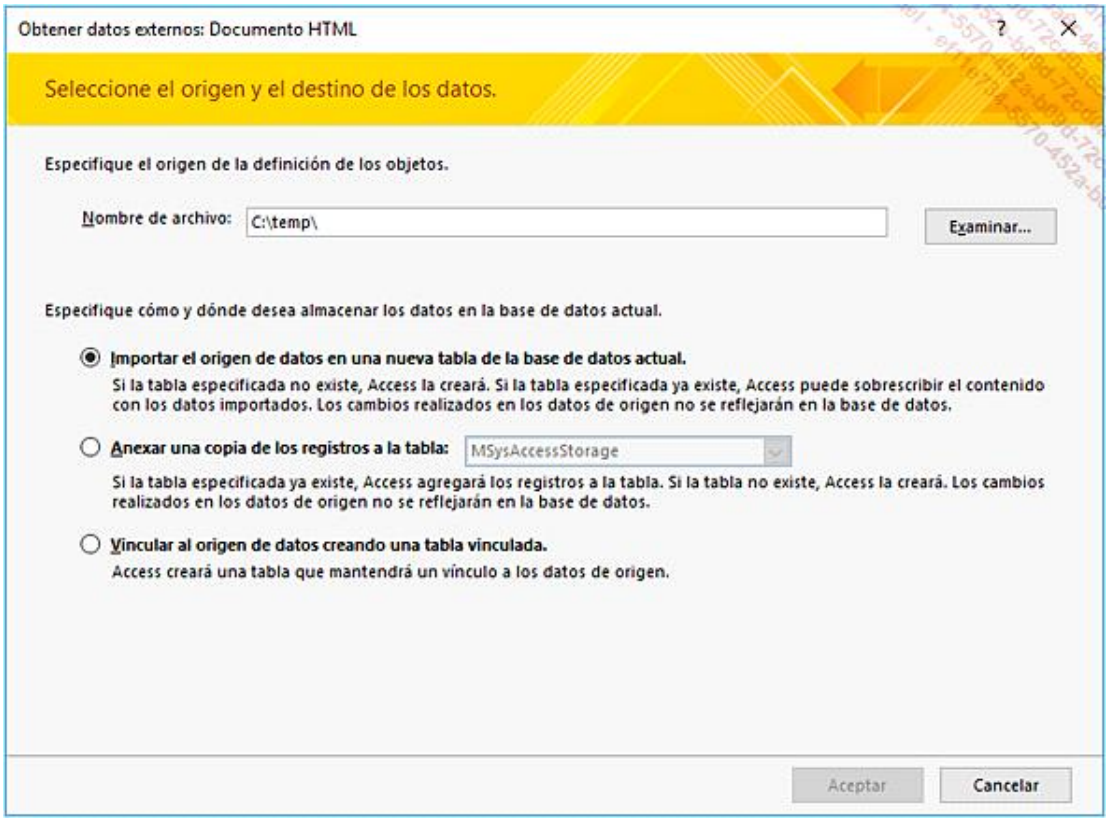
VBA y el formato HTML

1. Importar los datos HTML en la interfaz

Para importar datos en formato HTML, podemos pasar por la pestaña**Datos externos** de Microsoft Access 2019. Seleccione el icono **Documento HTML** haciendo clic en el menú **Nuevo origen de datos - Desde un documento HTML**.



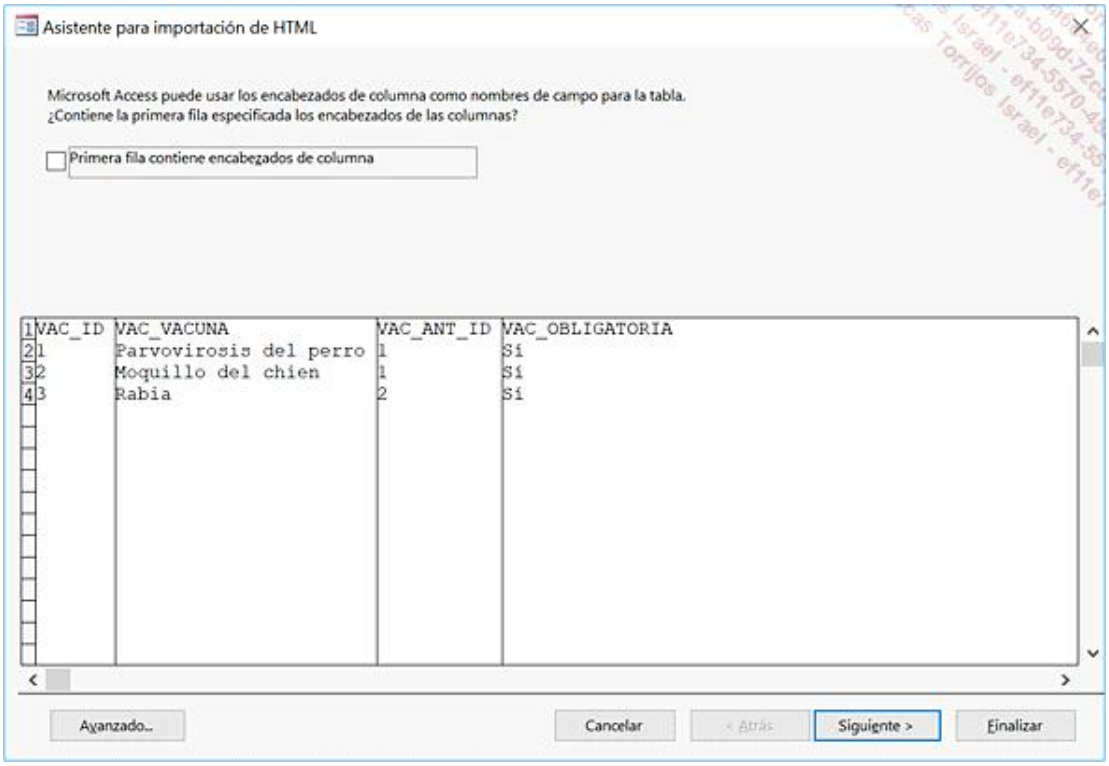
Se abre la interfaz de selección del archivo.



Pulse en el botón **Examinar** para que se abra el cuadro de diálogo de búsqueda del archivo. Seleccione el archivo

HTML que desea importar.

Haciendo clic en **Aceptar**, se abre la siguiente ventana:

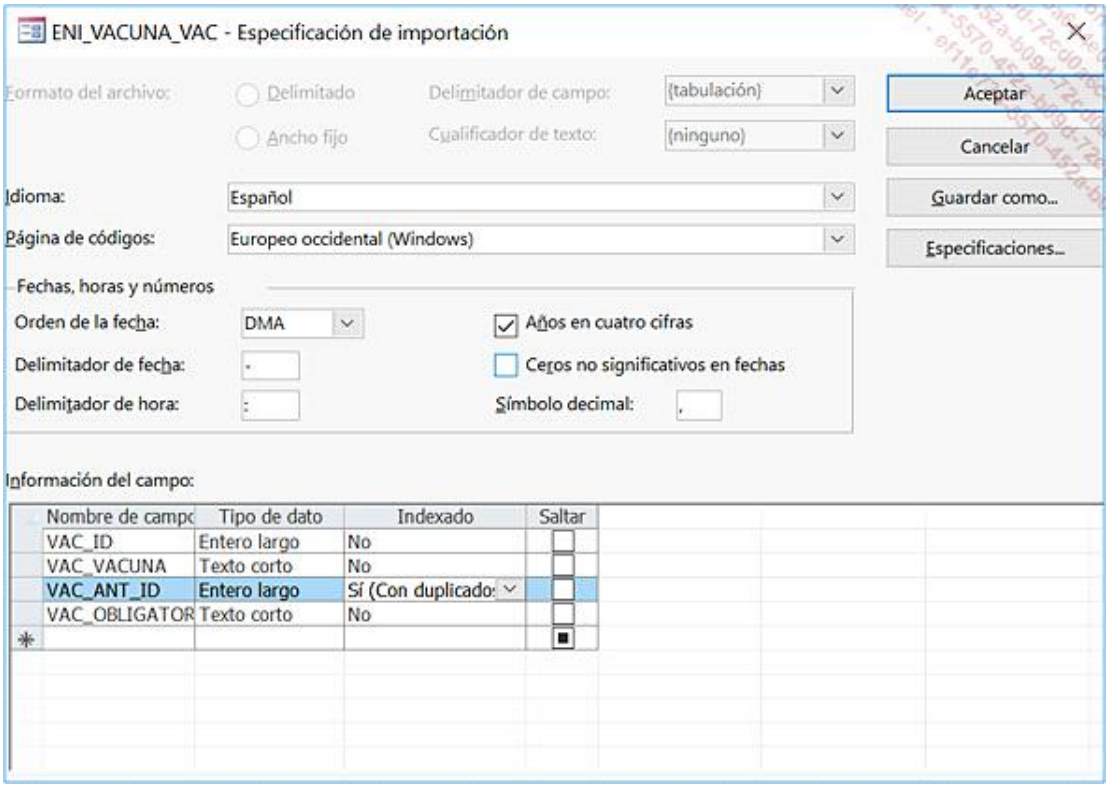


El proceso de importación se desarrolla de la manera siguiente:

- Indique si la primera fila contiene los encabezados de columnas,
- Especifique el tipo de dato que contiene cada una de las columnas.

Aquí, el proceso de importación es idéntico al de los archivos planos, como los archivos CSV.

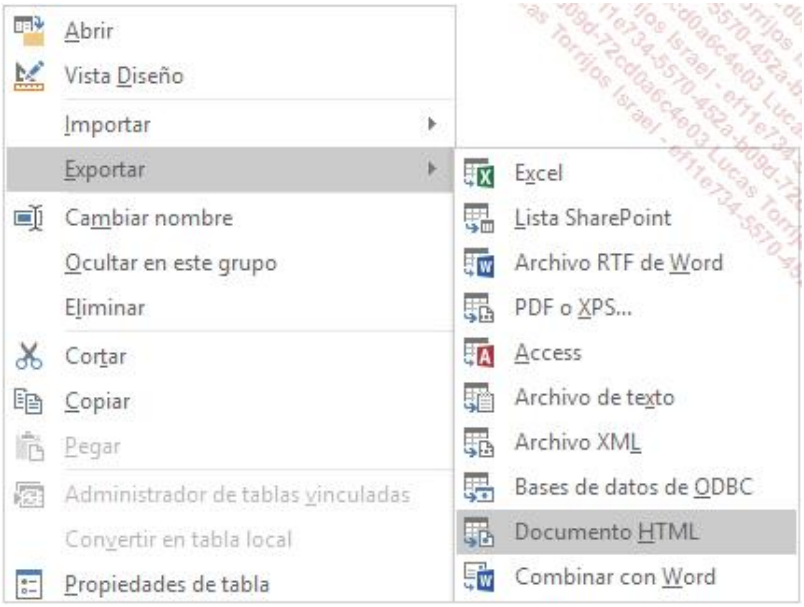
Haciendo clic en el botón **Avanzado**, se abre la interfaz con las opciones concretas de la importación.



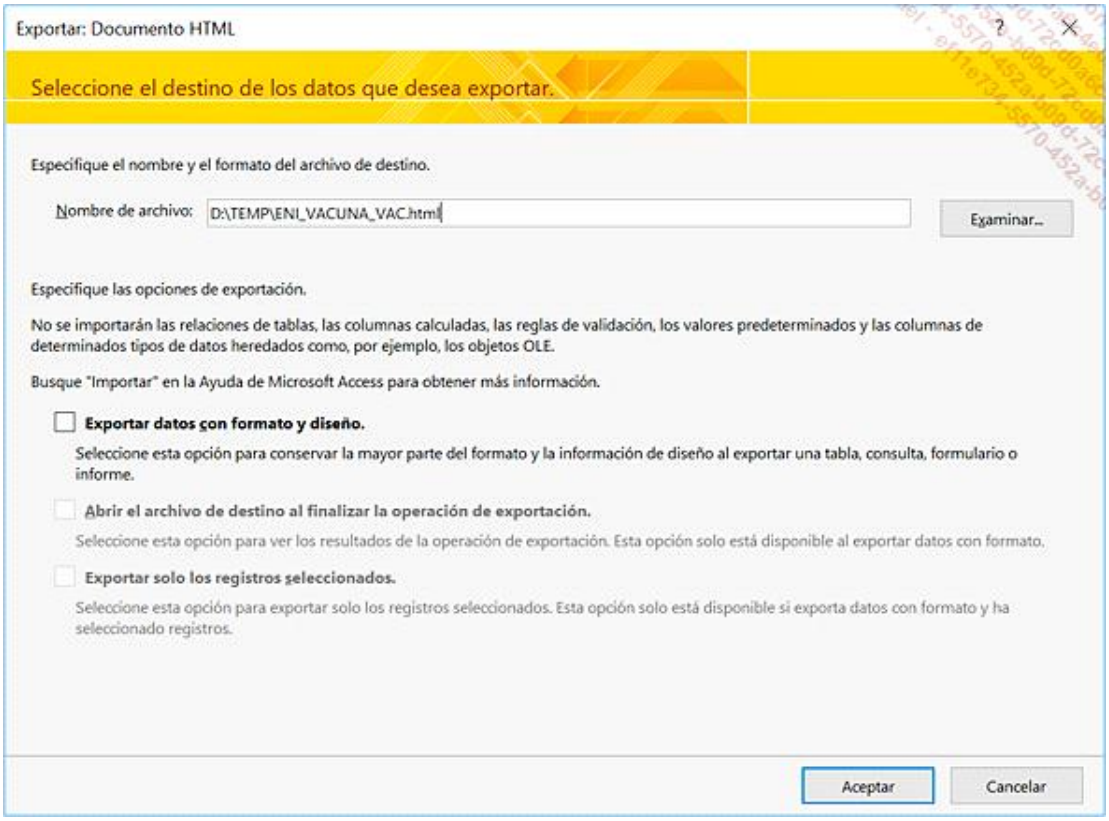
2. Exportar los datos HTML en la interfaz

De la misma manera que es posible importar datos a formato HTML, es posible exportarlos.

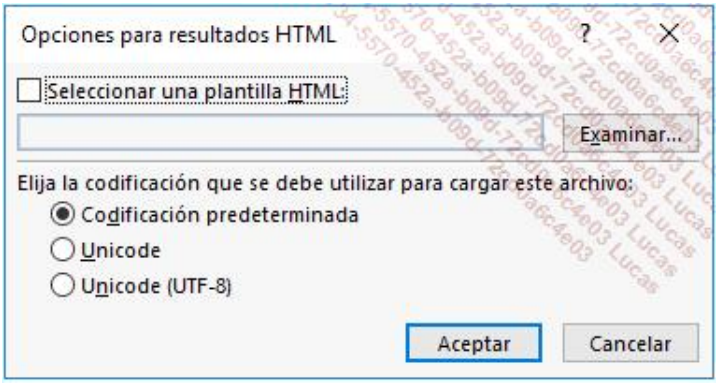
Haciendo clic derecho en el objeto a exportar, seleccione **Exportar - Documento HTML**.



Seleccione la ubicación del archivo. Ahora, la interfaz es la siguiente:



Es posible especificar si se desea exportar los datos, junto al formato y el diseño, y si el archivo de destino se debe abrir cuando termine la exportación. Para terminar, si se ha aplicado un filtro a la tabla, por ejemplo, será accesible la casilla **Exportar solo los registros seleccionados** y se podrá marcar. Una vez que se seleccionan estas opciones, pulse en **Aceptar** y se muestra la ventana que aparece a continuación:



Entonces es posible utilizar un modelo HTML para exportar los datos e indicar el formato de codificación del archivo. Después de haber pulsado en **Aceptar**, se muestra la ventana de resumen de la exportación que indica, llegado el caso, si ha tenido éxito o no.

VBA y el formato HTML

1. La importación de datos HTML

Para importar datos HTML, se usa el método `TransferText` disponible con el objeto `DoCmd`. La sintaxis general de este método es la siguiente:

```
DoCmd.TransferText acImportHTML, [SpecificationName], [TableName],
[FileName], [HasFieldNames], [HTMLTableName], [CodePage]
```

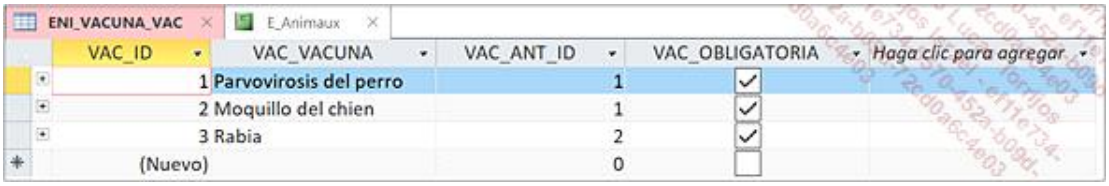
Los argumentos opcionales que pueden utilizar son los siguientes.

Argumento	Descripción
`SpecificationName`	Representa las especificaciones que se pueden guardar en Access.
`TableName`	Representa el nombre de la tabla que se importa.
`FileName`	Representa la ubicación del archivo HTML que sirve de fuente.
`HasFieldNames`	Indica si la primera línea del archivo HTML contiene el nombre de los campos o no.
`HTMLTableName`	Representa el nombre de la tabla en el archivo HTML, contenido entre las etiquetas `<CAPTION>` y `</CAPTION>`.
`CodePage`	Representa la página de código, es decir, el alfabeto utilizado para la visualización de los datos.

Por ejemplo, para importar los datos en una nueva tabla a partir del archivo XML que está en la carpeta C:\temp de la máquina, el código VBA deberá ser el siguiente:

```
Sub Ejemplo1()
DoCmd.TransferText acImportHTML, , "ENI_VACUNA_VAC", _
    "C:\temp\ENI_VACUNA_VAC.html", True
End Sub
```

Permite importar los datos en una tabla `ENI_VACUNA_VAC`:



2. La exportación de datos HTML

Para exportar los datos HTML en VBA, podemos llamar al mismo método `TransferText` del objeto `DoCmd`. La sintaxis general de este método es la siguiente:

```
DoCmd.TransferText acExportHTML, [SpecificationName], [TableName],
[FileName], [HasFieldNames], [HTMLTableName], [CodePage]
```

Por ejemplo, para exportar el conjunto de archivos desde la tabla ENI_VACUNA_VAC, el código VBA podrá ser el siguiente:

```
Public Sub Export ENI_VACUNA_VAC HTML()
DoCmd.TransferText acExportHTML, , "ENI_VACUNA_VAC",
"C:\temp\ENI_VACUNA_VAC.html", True
End Sub
```

Las API de Windows

1. Definición

Una API (*Application Programming Interface*) es una serie de funciones de sistema del sistema operativo, que se pueden llamar a partir de VBA. Estas funciones son accesibles desde los archivos DLL que hay en las carpetas de sistema.

En algunos casos, el uso de API es preferible, fundamentalmente:

- el uso de información de sistema que no se explota de manera nativa con VBA,
- la posible mejora del rendimiento del código. VBA no es el lenguaje más rápido, pero se pueden utilizar las API desarrolladas en otros lenguajes más rápidos, para optimizar el rendimiento,
- la optimización de la aplicación en términos de peso. El uso de API es menos costoso que el de ActiveX.

Es frecuente que el uso de las API provoque algunas reservas entre los desarrolladores, fundamentalmente por la baja calidad de la documentación disponible, la falta de información sobre la programación interna de las API, así como porque manipular las funciones del sistema es una fuente de riesgos de inestabilidad de las aplicaciones y del sistema operativo dentro del entorno informático.

2. Declaración de una API

La llamada a las funciones API en el cuerpo del programa se hace igual que para cualquier otra función o procedimiento. Sin embargo, hay una diferencia principal durante la declaración.

a. Sintaxis general

La sintaxis general de la declaración de una API es la siguiente:

```
[Private o Public] Declare Function o Sub Nombre Lib "NombreLibrería"
[Alias "NombreAlias"] [([[ByVal] variable [As type] [,[ByRef] variable
[As type]]...])] [As Type]
```

Instrucción	Descripción
`Private` o `Public`	Permite definir el ámbito de la función, como cualquier otra función o procedimiento en VBA.
`Declare`	`Declare` es la palabra clave obligatoria que se usa para la llamada a un procedimiento externo al programa.
`Function` o `Sub`	Permite definir si el recurso externo es una función o un procedimiento, como es el caso en VBA.
`Nombre`	Representa el nombre del procedimiento que se llama. Se debe respetar la norma de nomenclatura de VBA para que el nombre sea válido.
`Lib`	`Lib` es la palabra clave obligatoria que indica la ubicación de la librería `Nombrelibrería`.
`NombreLibrería`	Representa el nombre de la librería de sistema o la ubicación del archivo DLL. Debe estar entre comillas.
`Alias`	`Alias` es la palabra clave que sirve para introducir el `NombreAlias`. Esta palabra clave permite nombrar de manera diferente el procedimiento respeto a

	su nombre en la DLL, así como evitar el uso de nombres reservados del lenguaje VBA (Dim, As, etc.).
As Type	Permite determinar el tipo del valor de retorno de la función, si devuelve alguno.
Argumentos	Representa los posibles argumentos (nombre y tipo de dato) que se pueden proporcionar al código para que funcione.

b. Ejemplos

La siguiente función permite devolver el nombre del ordenador en el que se ejecuta el código. El resultado se almacena en la variable `lpBuffer`:

```
Public Declare Function GetComputerName Lib "kernel32" _
    Alias "GetComputerNameA" ( _
                ByVal lpBuffer As String, nSize As Long) As Long
```

La siguiente función permite generar un bip en los altavoces:

```
Private Declare Function Beep Lib "kernel32" _
    Alias "Beep" ( _
          ByVal dwFreq As Long, ByVal dwDuration As Long) As Long
```

3. Argumentos y punteros

Algunas funciones y procedimientos necesitan argumentos en la llamada, por lo que conviene saber qué tipos de datos se definen en el lenguaje que se utiliza para programar las DLL. A continuación se muestra una tabla de correspondencia entre los tipos de datos de VBA y los de C.

Un puntero en C corresponde a pasar un argumento por referencia en VBA.

Tipo C	Declaración equivalente VBA	Descripción
BYTE, CHAR	ByVal variable As Byte	Representa un único byte en memoria.
BOOL	ByVal variable As Long	Representa un valor numérico entero (Long) igual a 0 o 1.
ATOM	ByVal variable As Integer	Representa un valor numérico entero (Integer).
SHORT	ByVal variable As Integer	Representa un valor numérico entero en 16 bits, como el tipo Integer en VBA.
INT	ByVal variable As Long	Representa un valor numérico entero en 32 bits, como el tipo Long en VBA.
LONG	ByVal variable As Long	Igual que INT.
WORD	ByVal variable As Integer	Representa un valor numérico entero (Integer) o también dos bytes concatenados.
DWORD	ByVal variable As Long	Representa un valor numérico entero (Long) o también dos WORD.

UINT	ByVal variable As Long	Representa un valor numérico entero positivo (Long).
LPARAM, WPARAM, LRESULT	ByVal variable As Long	Igual que INT, utilizados algunas veces para describir el valor esperado.
COLORREF	ByVal variable As Long	Igual que INT, representa un código de color RGB.
HWND, HDC, HMENU, etc. (variables de administración de las ventanas, menús, etc.)	ByVal variable As Long	Iguales que INT, utilizados algunas veces para describir el valor esperado.
LPDWORD, LPINT, LPUINT	variable As Long	Punteros a un DWORD, INT o UINT.
LPWORD	variable As Integer	Puntero a un WORD.
LPRECT	variable As RECT	Puntero a una estructura de tipo RECT.
LP*	variable As (type)	Puntero a una variable, estructura o función.
LPSTR, LPCSTR	ByVal variable As String	Representa una cadena de caracteres (String); Visual Basic efectúa la conversión de valores.
LPVOID	variable As Any	Representa cualquier tipo de variable (utilizar la palabra clave ByVal al pasar una cadena de caracteres String).
NULL	As Any o ByVal variable As Long	Solo se puede utilizar con las sintaxis ByVal [Nothingo 0& o vbNull String] como valores.

4. Ejemplos

a. Abrir un archivo con la aplicación por defecto

La API más utilizada para abrir un archivo con la aplicación por defecto en la máquina es ShellExecute. Su sintaxis general es la siguiente:

```
Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" _
          (ByVal hwnd As Long, ByVal lpszOp As String, _
           ByVal lpszFile As String, ByVal lpszParams As String, _
           ByVal LpszDir As String, ByVal FsShowCmd As Long) _
                  As Long
```

Para abrir un archivo PDF con la herramienta por defecto, se realizará la llamada de esta función como sigue:

```
ShellExecute Application.hWndAccessApp, "open", "C:\temp\archivo_pdf.pdf", "",
CurrentProject.Path, 1
```

Es posible utilizar el valor de retorno de la función para saber si la ejecución se ha hecho correctamente. En caso contrario, la función devuelve un número de error, por lo que a continuación se muestran los posibles valores:

Valor error	Descripción
0	Al sistema le falta memoria o recursos, el ejecutable está corrupto o las reasignaciones no son válidas.
2	Archivo no encontrado.
3	Ruta no encontrada.
5	Se ha hecho un intento para relacionarse dinámicamente con una tarea, o hay un error de compartición o de protección de red.
6	La librería necesita segmentos de datos separados para cada tarea.
8	No hay suficiente memoria disponible para ejecutar la aplicación.
10	Versión de Windows incorrecta.
11	El archivo ejecutable no es correcto, puede que no se trate de una aplicación de Windows, o que haya un error en el archivo .EXE.
12	La aplicación se ha diseñado para otro sistema operativo.
13	La aplicación se ha diseñado para MS-DOS 4.0.
14	El tipo de archivo ejecutable es desconocido.
15	Intento de carga de una aplicación en modo real.
16	Intento de carga de una segunda instancia de un archivo ejecutable que contiene varios segmentos de datos que no están marcados en modo solo lectura.
19	Intento de carga de un archivo ejecutable comprimido, el archivo se debe descomprimir antes de cargarse.
20	Archivo de librería relacionada dinámicamente (DLL) incorrecta; una de las DLL que se necesita para ejecutar esta aplicación está corrupta.
21	La aplicación requiere extensiones Microsoft Windows 32-bits.
31	No hay asociación con el tipo de archivo especificado, o no hay asociación para la acción elegida y para el tipo de archivo seleccionado.

b. Acceder a la base de registro

Hay varias API que permiten manipular las claves de la base de registro:

API	Descripción
RegCreateKeyEx	Permite crear una clave o abrirla si ya existe.
RegDeleteKey	Permite eliminar una clave y todas sus subclaves posibles en Windows 9x, pero no puede eliminar una clave que contiene subclaves en Windows NT.
RegOpenKeyEx	Permite abrir una clave.
RegCloseKey	Permite cerrar una clave.
RegSetValueEx	Permite modificar o crear un valor.
RegQueryValueEx	Permite leer el contenido de un valor.
RegDeleteValue	Permite eliminar un valor.
RegEnumKeyEx	Permite enumerar las subclaves de una clave.
RegEnumValue	Permite enumerar los valores de una clave.

c. Crear una carpeta

La API `CreateDirectory` permite crear una nueva carpeta, indicando su ubicación. La sintaxis de declaración es la siguiente:

```
Private Declare Function CreateDirectory Lib "kernel32" _
                         Alias "CreateDirectoryA" ( _
    ByVal lpPathName As String, _
    lpSecurityAttributes As SECURITY_ATTRIBUTES) _
                                          As Long
```

Se debe completar con el tipo `SECURITY_ATTRIBUTES`:

```
' Declaración del tipo SECURITY_ATTRIBUTES
Private Type SECURITY_ATTRIBUTES
    nLength As Long
    lpSecurityDescriptor As Long
    bInheritHandle As Long
End Type
```

Para crear una estructura completa, podemos utilizar la API `MakeSureDirectoryPathExists`, que utiliza la siguiente declaración:

```
Public Declare Function MakeSureDirectoryPathExists _
        Lib "imagehlp.dll" (ByVal lpPath As String) As Long

Sub Ejemplo_Creacion_Carpeta()
'Comprueba que la carpeta existe y la crea en caso de no existir
MakeSureDirectoryPathExists "C:\temp\carpeta_inexistente\"
End Sub
```

d. Recuperar el nombre de la máquina o modificarlo

Las API `GetComputerName` y `SetComputerName` permiten manipular el nombre de la máquina. Sus declaraciones son las siguientes:

```
Private Declare Function GetComputerName Lib "kernel32" _
                         Alias "GetComputerNameA" _
                   (ByVal lpBuffer As String, _
                   nSize As Long) As Long
```

y:

```
Private Declare Function SetComputerName Lib "kernel32" _
                         Alias "SetComputerNameA" _
                   (ByVal lpComputerName As _
                               String) As Long
```

```
Sub Ejemplo_Nombre_Maquina()
   'Se declaran las dos variables necesarias
   Dim lngResponse As Long
   Dim strNombreMaquina As String * 32
   lngResponse = GetComputerName(strNombreMaquina, 32)
   'strNombreMaquina toma el valor deseado
   'seguido de varios caracteres ' ' para llegar a 32 _
   caracteres en total
   'se muestra el resultado
     MsgBox Trim(strNombreMaquina)
End Sub
```

e. Las API relacionadas con el portapapeles de Windows

Hay varias API que permiten manipular el portapapeles de Windows.

API	Descripción
OpenClipboard	Permite abrir el portapapeles.
CloseClipboard	Permite cerrar el portapapeles.
GetClipboardData	Permite recuperar el contenido del portapapeles.
SetClipboardData	Permite escribir en el portapapeles.
EmptyClipboard	Permite vaciar el portapapeles.

La sintaxis de las API es la siguiente:

```
Declare Function OpenClipboard Lib "user32" (ByVal hWnd As Long)
As Long
Declare Function CloseClipboard Lib "user32" () As Long
Declare Function GetClipboardData Lib "user32" (ByVal wFormato As
Long) As Long
Declare Function SetClipboardData Lib "user32" (ByVal wFormato As
Long, ByVal hMem As Long) As Long
Declare Function EmptyClipboard Lib "user32" () As Long

Sub Ejemplo_VaciarPortaPapeles()
   OpenClipboard 0
   EmptyClipboard
   CloseClipboard
End Sub
```

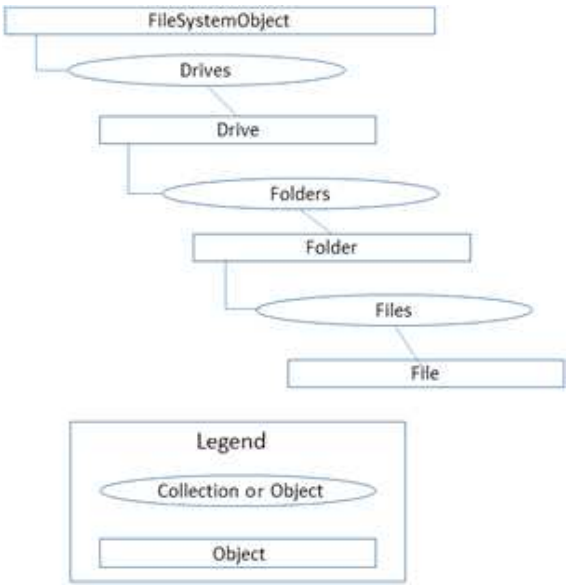
El objeto FileSystemObject

1. Introducción

Más allá de las funciones básicas que ofrece VBA para acceder y manipular los archivos y carpetas, hay un objeto muy práctico. El objeto `FileSystemObject` está disponible desde la librería **Microsoft Scripting Runtime**.

2. Jerarquía de objeto

El siguiente esquema representa la jerarquía de los principales objetos y colecciones del tipo FileSystemObject.



3. Gestión de los discos

Es posible recorrer la colección de los discos `Drives`. Un disco es un objeto `Scripting.Drive`.

a. Propiedades del objeto Drive

Propiedad	Descripción
`AvailableSpace`	Representa el espacio disponible en el disco, expresado en bytes.
`DriveLetter`	Representa la letra utilizada por el sistema para acceder al disco.
`DriveType`	Representa el tipo de disco (`CDRom`, `Fixed`, `RamDisk`, `Remote`, `Removable` o `UnknownType`).
`FileSystem`	Representa el tipo de sistema del disco (NTFS, FAT).
`FreeSpace`	Representa el espacio libre en el disco, expresado en bytes.
`IsReady`	Representa un valor booleano que indica si el disco está disponible (por ejemplo, presencia de una llave USB en el puerto USB frontal).
`Path`	Representa la ruta de acceso al disco (`I:` por ejemplo).
`RootFolder`	Representa la carpeta raíz (`I:\` por ejemplo).
`SerialNumber`	Representa el número de serie del disco.

ShareName	Representa el nombre compartido del disco (devolverá una cadena de caracteres vacía si el disco no está compartido).
TotalSize	Representa el tamaño total del disco, expresado en bytes.
VolumeName	Representa el nombre de la unidad del disco (ejemplo DATA).

b. Ejemplo

A continuación se muestra un procedimiento que lista los discos disponibles, indica para cada uno su letra, espacio disponible, tipo y sistema:

```
Sub Lista_de_discos()
    Dim FSO As New Scripting.FileSystemObject
    Dim drv As Scripting.Drive
    For Each drv In FSO.Drives
           If drv.IsReady Then
                 Debug.Print drv.DriveLetter & " " & _
                        drv.AvailableSpace & " " & _
                        drv.DriveType & " " & _
                        drv.FileSystem
           End If
    Next
End Sub
```

4. Gestión de carpetas

Es posible recorrer la colección de las carpetas `Folders`. Una carpeta es un objeto `Scripting.Folder`.

a. Propiedades del objeto Folder

Propiedad	Descripción
DateCreated	Fecha de creación de la carpeta.
DateLastAccessed	Fecha del último acceso a la carpeta.
DateLastModified	Fecha de la última modificación de la carpeta.
Drive	Objeto `Drive` correspondiente a la unidad de disco en la que se encuentra la carpeta.
Files	Colección que agrupa los archivos de la carpeta, ver la sección Gestión de archivos.
IsRootFolder	Representa un valor booleano que indica si la carpeta es la carpeta raíz de su unidad de disco.
Name	Nombre de la carpeta. Ejemplo: `Imágenes`.
ParentFolder	Objeto `Folder` correspondiente a la carpeta padre. Si la carpeta es una carpeta `RootFolder`, esta propiedad devuelve `Nothing`.
Path	Ruta completa de acceso a la carpeta. Ejemplo: `D:\Imágenes`
ShortName	Nombre corto, con ocho caracteres máximo. Ejemplo: `PRUEBA~1`
ShortPath	Ruta completa de acceso a la carpeta donde cada componente respeta la norma definida en `ShortName`. Ejemplo: `D:\ABCDEF1~1\PRUEBA~1`

Size	Tamaño total de la carpeta en bytes. Se trata de la suma de los tamaños de todos los archivos presentes en la carpeta y sus subdirectorios.
SubFolders	Colección de objetos Folder que agrupa los subdirectorios.
Type	Tipo de la carpeta. En todos los casos probados, se trata de FileFolder.

b. La propiedad Attributes de una carpeta

Como hemos podido ver en la sección ADO del capítulo Los objetos de acceso a los datos DAO y ADO, las carpetas pueden tener atributos, cuyos valores se pueden se acumular (carpeta cacheada, comprimido, etc.). Los diferentes valores Attributes son los siguientes:

Atributo	Valor	Descripción
Archive	32	Carpeta archivada.
Compressed	2048	Carpeta comprimida.
Hidden	2	Carpeta cacheada.
Normal	0	Carpeta normal.
ReadOnly	1	Carpeta en modo solo lectura.
System	4	Carpeta de sistema.

c. Métodos del objeto Folder

Método	Descripción
Copy	Permite copiar una carpeta, así como su contenido, en otra carpeta (este último creado sobre la marcha, si no existe).
CreateTextFile	Permite crear un archivo en la carpeta. Ver la sección Gestión de archivos.
Delete	Permite eliminar una carpeta, indicando si los archivos en modo solo lectura también se verán eliminados o no.
Move	Permite mover una carpeta a otro lugar en la máquina.

d. Ejemplos

A continuación se muestra un procedimiento que lista los subdirectorios a partir de una carpeta raíz (aquí D:\ENI):

```
Sub Lista_de_carpetas()
    Dim FSO As Scripting.FileSystemObject
    Dim Carpeta_Raiz As Scripting.Folder
    Dim fld As Scripting.Folder
    Set FSO = New Scripting.FileSystemObject
    Set Carpeta_Raiz = FSO.GetFolder("D:\ENI")
    For Each fld In Carpeta_Raiz.SubFolders
           Debug.Print fld.Name
    Next
End Sub
```

El siguiente código comprueba la existencia de una carpeta, crea otro, copia los elementos del primero en el segundo y elimina la carpeta.

```
Sub Creacion_Movimiento_Eliminacion_Carpeta()
    Dim FSO As Scripting.FileSystemObject
    Dim Carpeta_Raiz As Scripting.Folder
    Dim Carpeta_Destino As Scripting.Folder
    Dim strDirectorio As String
    Dim strDestino As String
    Set FSO = New Scripting.FileSystemObject
    strDirectorio = "D:\ENI\"
    strDestino = "D:\ENI2\"
    If FSO.FolderExists(strDirectorio) Then
           Set Carpeta_Raiz = FSO.GetFolder(strDirectorio)
           FSO.CreateFolder strDestino
           Carpeta_Raiz.Copy strDestino
           Set Carpeta_Destino = FSO.GetFolder(strDestino)
           Carpeta_Destino.Delete True
    End If
End Sub
```

5. Gestión de archivos

Es posible recorrer la colección de archivos `Files`. Un archivo es un objeto `Scripting.File`. Se trata del elemento más bajo en la jerarquía de objetos `FileSystemObject`.

a. Propiedades del objeto File

Propiedad	Descripción
`DateCreated`	Fecha de creación del archivo.
`DateLastAccessed`	Fecha del último acceso al archivo.
`DateLastModified`	Fecha de la última modificación del archivo.
`Drive`	Objeto `Drive` correspondiente a la unidad de disco en la que se encuentra el archivo.
`Name`	Nombre de la carpeta. Ejemplo: `Imagen.bmp`
`ParentFolder`	Objeto `Folder` correspondiente a la carpeta padre.
`Path`	Ruta completa de acceso al archivo. Ejemplo: `D:\Imágenes\imagen.bmp`
`ShortName`	Nombre corto, con ocho caracteres máximo. Ejemplo: `PRUEBA~1`
`ShortPath`	Ruta completa de acceso al archivo, dónde cada componente respeta la norma fijada en `ShortName`. Ejemplo: `D:\ABCDEF1~1\ PRUEBA~1`
`Size`	Tamaño total del archivo en bytes.
`Type`	Tipo del archivo.

b. Métodos del objeto File

Método	Descripción
Copy	Permite copiar un archivo en otra carpeta (ya existente, porque en caso contrario se produce un error 76).
Delete	Permite eliminar un archivo, indicando si se puede borrar incluso si el archivo está en modo solo lectura.
Move	Permite mover una carpeta a otro lugar en la máquina.
OpenAsTextStream	Permite abrir un archivo de texto, ver la sección Los archivos de texto.

c. Ejemplos

El siguiente código muestra todos los archivos de una carpeta:

```
Sub Lista_de_Archivos()
    Dim FSO As Scripting.FileSystemObject
    Dim Carpeta_Raiz As Scripting.Folder
    Dim arc As Scripting.File
    Set FSO = New Scripting.FileSystemObject
    Set Carpeta_Raiz = FSO.GetFolder("D:\ENI")
    For Each arc In Carpeta_Raiz.Files
           Debug.Print arc.Name
    Next
End Sub
```

6. Métodos del objeto FileSystemObject

Método	Descripción
BuildPath	Permite crear una ruta válida a partir de una ubicación y de un nombre de archivo.
CopyFile	Permite copiar un archivo.
CopyFolder	Permite copiar una carpeta.
CreateFolder	Permite crear un nueva carpeta.
CreateTextFile	Permite crear un nuevo archivo de texto.
DeleteFile	Permite eliminar un archivo.
DeleteFolder	Permite eliminar una carpeta.
DriveExists	Permite probar la existencia de un disco en la máquina.
FileExists	Permite probar la existencia de un archivo a partir de su ubicación.
FolderExists	Permite probar la existencia de una carpeta a partir de su ubicación.
GetAbsolutePathName	Permite recuperar una ruta completa sin ambigüedad, a partir de una ruta.
GetBaseName	Permite recuperar la ruta de base a partir de una ruta.
GetDrive	Permite acceder a un disco.
GetDriveName	Permite extraer el disco a partir de una ubicación de archivo.
GetExtensionName	Permite extraer la extensión de un archivo a partir de su ubicación.
GetFile	Permite acceder a un archivo.

GetFileName	Permite extraer el nombre del archivo a partir de su ubicación.
GetFileVersion	Permite recuperar la versión de un archivo a partir de su ubicación.
GetFolder	Permite acceder a una carpeta.
GetParentFolderName	Permite recuperar el nombre de la carpeta padre a partir de una ubicación.
GetSpecialFolder	Permite acceder a las carpetas especiales (directorio Windows, directorio de sistema o directorio Temporal).
GetStandardStream	Permite recuperar un objeto de tipo TextStream, siguiendo tres posibles modos (StdIn para un flujo entrante, StdOut para un flujo saliente y StdErr para un flujo de error estándar).
GetTempName	Permite recuperar un nombre de archivo o de directorio temporal generado aleatoriamente, para realizar operaciones que necesitan el uso de un archivo o de una carpeta temporal.
MoveFile	Permite mover un archivo.
MoveFolder	Permite mover una carpeta.
OpenTextFile	Permite abrir un archivo y devolver un objeto de tipo TextStream, pudiendo leer el archivo o realizar una adición.

El siguiente código permite recuperar la diferente información de la base de datos actual:

```
Sub Details_FSO()
    Dim FSO As Scripting.FileSystemObject
    Dim strUbicacionBDD As String
    Set FSO = New Scripting.FileSystemObject
    strUbicacionBDD = CurrentProject.FullName
    Debug.Print FSO.GetDriveName(strUbicacionBDD)
    Debug.Print FSO.GetBaseName(strUbicacionBDD)
    Debug.Print FSO.GetParentFolderName(strUbicacionBDD)
    Debug.Print FSO.GetFileName(strUbicacionBDD)
    Debug.Print FSO.GetExtensionName(strUbicacionBDD)
End Sub
```

Los archivos de texto

Los archivos de texto están entre los elementos más manipulados en programación, ya sea para enviar o para recibir información. Por tanto, es importante tratar su manejo en esta sección. Como recordatorio, un archivo de texto es un archivo compuesto por una o varias líneas, separadas por una combinación de caracteres de retorno de carro y cambio de línea: `vbCr` & `vbLf` e incluso `vbCrLf`.

1. Acceso secuencial

Históricamente, el primer medio de acceder y leer un archivo en VBA es el acceso secuencial: en primer lugar, se abre el archivo con el método `Open` y se le asigna un número hasta su cierre.

a. Sintaxis general

La sintaxis general de apertura de un archivo es la siguiente:

```
Open Ubicación For Input|Output|Append As Número
```

El archivo se puede abrir en modo lectura, en modo escritura o en modo adición.

Para cerrar el archivo cuando se ha terminado de usar, la sintaxis general es la siguiente:

```
Close Número
```

b. Lectura

Para abrir un archivo en modo solo lectura, la sintaxis es la siguiente:

```
Open Ubicación For Input As Número
```

La variable `Número` no se puede utilizar varias veces mientras esté abierto el archivo al que se ha asignado dicho número. Para no tener conflicto con la numeración, la función `FreeFile` devuelve un valor correcto comprendido entre 1 y 255.

```
Dim NumArc As Integer
NumArc = FreeFile
Open "D:\ENI\ejemplo.txt" For Input As NumArc
```

Si la numeración entre 1 y 255 no es suficiente, queda la posibilidad de generar un número válido entre 256 y 511, gracias a la llamada a la función `FreeFile` con el argumento 1:

```
NumArc = FreeFile(1)
```

Una vez abierto el archivo, el método que permite leer su contenido línea a línea se llama `Line Input`, con la

siguiente sintaxis general:

```
Line Input #Numero, CadenaReceptora
```

En el siguiente ejemplo, se muestra la primera línea del archivo `ejemplo.txt`:

```
Sub Leer_Primera_Linea ()
Dim NumArc As Integer
Dim strLinea As String
NumArc = FreeFile
Open "D:\ENI\ejemplo.txt" For Input As NumArc
Line Input #NumArc, strLinea
Debug.Print strLinea
Close NumArc
End Sub
```

Antes de la lectura de la línea, el cursor pasará a la siguiente línea, sin posibilidad de volver a la línea anterior. De esta manera, para recorrer la integridad de un archivo, se utilizará la función `EOF()`, que indica si se alcanza el fin del archivo.

La siguiente sintaxis permite recorrer cada una de las líneas del archivo:

```
Sub Leer_Todo_Archivo()
Dim NumArc As Integer
Dim strLinea As String
NumArc = FreeFile
Open "D:\ENI\ejemplo.txt" For Input As NumArc
While Not EOF(NumArc)
    Line Input #NumArc, strLinea
    Debug.Print strLinea
Wend
Close NumArc
End Sub
```

c. Escritura

Para escribir en un archivo de texto, es suficiente con abrirlo en modo escritura (`Output`) o en modo adición (`Append`).

```
Sub Abrir_Archivo_En_Escritura()
Dim NumArc As Integer
NumArc = FreeFile
Open "D:\ENI\ejemplo_2.txt" For Output As NumArc
Close numArc
End Sub
```

Para escribir, se utiliza la instrucción `Print` con la siguiente sintaxis general:

```
Print #Numero, TextoAEscribir
```

Una vez que el texto que se debe escribir `TextoAEscribir` se ha añadido al archivo, se añaden un retorno de carro y un salto de línea automáticamente, para terminar el archivo.

El siguiente código permite escribir la fecha de hoy en un archivo vacío:

```
Sub Escribir_Fecha_Hoy ()
Dim NumArc As Integer
NumArc = FreeFile
Open "D:\ENI\ejemplo_3.txt" For Output As NumArc
Print #NumArc, Date
Close NumArc
End Sub
```

La apertura con `Output` va a eliminar el archivo si ya existe, mientras que la apertura con `Append` pone automáticamente el cursor al final del archivo para añadir texto a partir de ahí y conservar intacto el inicio del archivo.

2. Acceso directo

Adicionalmente a los modos de apertura `Input`, `Output` y `Append`, también hay un modo `Random`, que permite abrir el archivo de texto tanto en modo lectura como escritura. Este modo de apertura es particularmente útil cuando desea manipular datos exportados desde Access con un formato fijo de datos. Cada línea del archivo corresponderá a un registro, que se intentará cortar de manera correcta.

a. Lectura

La función `Get` permite leer una línea del archivo y almacenar el resultado en una variable. En primer lugar, será necesario determinar una variable que tiene la misma estructura que los datos que se desea recuperar.

Por ejemplo, la estructura del archivo Individuo está formada por un número de 4 caracteres, un nombre de 5 caracteres y un apellido de 25 caracteres.

```
Type Individuo
    Num As String * 4
    Nombre As String * 5
    Apellido As String * 25
End Type
```

El código también necesita conocer la estructura que se debe recuperar durante la lectura; para esto se usa la palabra clave `Len`, a la que se asigna el tamaño global de un registro.

Para terminar, una vez que se conoce esta información, podemos indicar a la función `Get` el número de registros que se va a leer durante la llamada del método.

El código para recuperar el contenido de la primera y segunda línea y visualizarlo será el siguiente:

```
Sub Leer_Random()
Dim ind As Individuo
Dim NumArc As Integer
NumArc = FreeFile
Open "D:\ENI\ejemplo_4.txt" For Random As NumArc Len = Len(ind)
Get NumArc, 1, ind
Debug.Print ind.Num & " " & ind.Nombre & " " & ind.Apellido
Debug.Print ind.Apellido
Get NumArc, 2, ind
Debug.Print ind.Num & " " & ind.Nombre & " " & ind.Apellido
Close #NumArc
End Sub
```

b. Escritura

De la misma manera que es posible leer un registro con la función `Get`, podemos escribir un registro en un archivo con el método `Put`. El número de registros también se pasará al método, pero, si este ya está asignado, los datos se eliminarán por la escritura.

```
Sub Escribir _Random()
Dim NumArc As Integer
Dim ind As Individuo
NumArc = FreeFile
Open "D:\ENI\ejemplo_5.txt" For Random As NumArc Len = Len(ind)
With ind
    .Num = "003"
    .Nombre = " Ángel María "
    .Apellido = "Sánchez"
End With
Put NumArc, 3, ind
Close NumArc
End Sub
```

Problemática

1. Contexto

La asociación del Refugio para Animales desea poder tener un mejor seguimiento de los animales que acoge, de su estado de salud y de los adoptantes junto a los que los pequeños animales encuentran su felicidad. El refugio se ocupa actualmente de los perros, gatos y conejos, pero en el futuro se podría ocupar de otras razas. Para poder gestionar el conjunto de diferentes elementos, el equipo desea implementar una base de datos Access.

2. Objetivos

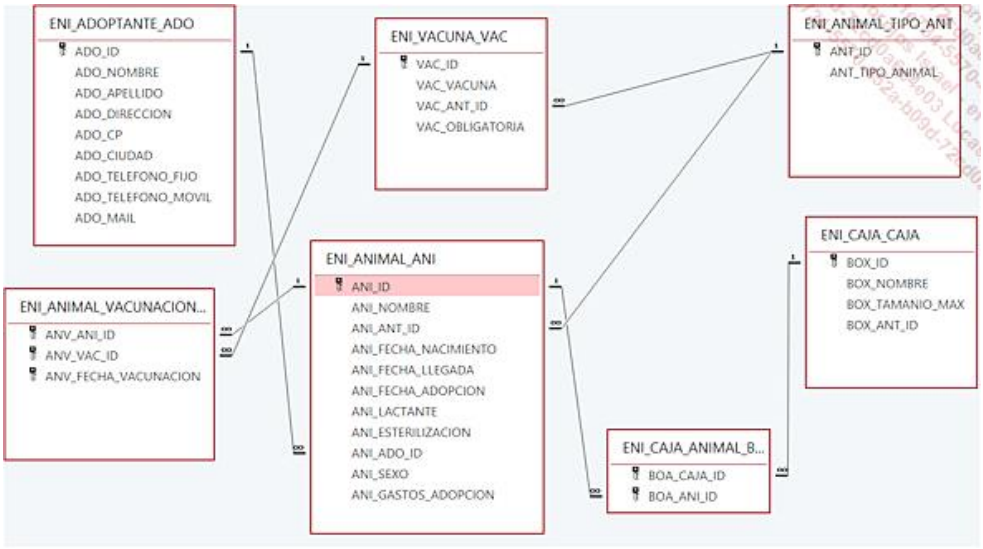
La base de datos Access que se debe implementar, debe permitir rellenar la información de los animales, así como sus vacunas y adoptantes. Aquí solo se muestra una parte de la aplicación, pero puede encontrar la totalidad en las bases de datos de ejemplo. La mini aplicación aspira a servir de punto de partida, y evidentemente algunos elementos quedan pendientes de implementación.

3. Arquitectura de la base

La información Animal se almacena en la tabla ENI_ANIMAL_ANI. Aquí encontramos:

- el ID del animal;
- su nombre;
- su raza;
- su fecha de nacimiento;
- su fecha de llegada al refugio;
- su fecha de adopción;
- si está destetado;
- su esterilización;
- la información del adoptante;
- su sexo;
- los gastos de adopción.

Las razas de los animales se almacenan en la tabla ENI_ANIMAL_TIPO_ANT.



4. Formulario Inicio

Todos los formularios tienen un tamaño automático de 15 cm de lado y no tiene selector.

a. Interfaz

El formulario no contiene ningún elemento para el usuario, obligándole a pasar por los elementos de la cinta de opciones.

b. Cinta de opciones dedicada

Aspecto visual de la cinta de opciones

Código XML de la cinta de opciones

```
<customUI xmlns="http://schemas.microsoft.com/office/2009/07/customui"
onLoad="SobreCargaCintaOpciones">
   <ribbon startFromScratch="true">
       <tabs>
<!-- Bloque Inicio -->
          <tab id="TabAnimales" label="Inicio">
<!-- Grupo Animales -->
              <group id="Grupo_Animales" label="Animales">
                  <button id="Btn_Animal_Agrega" label="Nuevo Animal"
size="large" onAction="Mostrar_Animal" tag="Logo_Animales.jpg"
getImage="CintaOpciones_Recupera_Imagen" />
                  <button id="Btn_Animal_Busqueda" label="Lista Animales"
size="large" onAction="Buscar_Animal" tag="lupa_busqueda.jpg"
getImage="CintaOpciones_Recupera_Imagen"/>
                  <button id="Btn_Animal_Estadisticas"
label="Estadísticas Animal"
size="large" onAction="Estadisticas_Animal" imageMso="CycleFillColor"/>
```

```
            </group>
<!-- Grupo Adoptantes -->
            <group id="Grupo_Adoptante" label="Adoptantes">
                <button id="Btn_Adoptante_Busqueda" label="Lista"
size="normal" onAction="Buscar_Adoptante"
imageMso="AddOrRemoveAttendees" />
                <button id="Btn_Adoptante_Agrega" label="Agregar"
size="normal" onAction="Mostrar_Adoptante"
imageMso="DistributionListAddNewMember"/>
                <button id="Btn_Adoptante_Mail"
label="Enviar Noticias"
size="normal" onAction="Mostrar_Boletin" imageMso="CheckNames"/>
                <button id="Btn_Adoptante_Estadisticas"
label="Estadísticas Adoptante"
size="normal" onAction="Estadisticas_Adoptante" imageMso="CycleFillColor"/>
            </group>
<!-- Grupo Vacunas -->
            <group id="Grupo_Vacuna" label="Vacunas">
                <button id="Btn_Vacuna_Agrega" label="Nueva Vacuna"
size="large" onAction="Mostrar_Vacuna" tag="Vacuna.jpg"
getImage="CintaOpciones_Recupera_Imagen"/>
                <button id="Btn_Vacuna_Busqueda" label="Lista Vacunas"
size="large" onAction="Buscar_Vacuna"
imageMso="FunctionsLookupReferenceInsertGallery"/>
                <button id="Btn_Vacunacion" label="Vacunaciones"
size="large" onAction="Mostrar_Vacunacion" imageMso="LookUp"/>
            </group>
<!-- Grupo Caja -->
            <group id="Grupo_Caja" label="Caja">
                <button id="Btn_Caja_Agrega" label="Nueva Caja"
size="large" onAction="Mostrar_Caja" imageMso="BlogHomePage"/>
                <button id="Btn_Caja_Busqueda" label="Lista Caja"
size="large" onAction="Buscar_Caja" imageMso="ArrangeByCompany"/>
            </group>
<!-- Grupo Salir -->
            <group id="Grupo_Salir" label="Salir">
                <button id="Btn_Salir" label="Salir Aplicación"
size="large" onAction="Salir_Aplicacion " imageMso="CancelRequest"/>
            </group>

        </tab>
    </tabs>
  </ribbon>
</customUI>
```

Código VBA de la cinta de opciones

```
'Común
Public Sub Cerrar_Formulario(control As IRibbonControl)
   DoCmd.Close acForm, Screen.ActiveForm.Name
End Sub

Public Sub CintaOpciones_Recupera_Imagen(control As IRibbonControl, ByRef image)
   Set image = LoadPicture(CurrentProject.Path & "\imagenes_cintaOpciones\" &
control.Tag)
End Sub


'Inicio
'Grupo Animales
```

```
Public Sub Mostrar_Animal(control As IRibbonControl)
   SetParam "ANI_ID", 0
   DoCmd.OpenForm "F_ANIMAL", acNormal, , , acFormEdit, acWindowNormal
End Sub

Public Sub Buscar_Animal(control As IRibbonControl)
   DoCmd.OpenForm "F_ANIMALES", acNormal, , , acFormEdit, acWindowNormal
End Sub


Public Sub Estadisticas_Animal(control As IRibbonControl)
   DoCmd.OpenReport "I_Animales", acViewReport, , , acWindowNormal
End Sub

'Grupo Adoptantes
Public Sub Buscar_Adoptante(control As IRibbonControl)
   DoCmd.OpenForm "F_Adoptantes"
End Sub

Public Sub Mostrar_Adoptante(control As IRibbonControl)
   SetParam "ADO_ID", 0
   DoCmd.OpenForm "F_Adoptante"
End Sub

Public Sub Mostrar_Boletin(control As IRibbonControl)
   DoCmd.OpenForm "F_BoletinNoticias"
End Sub

Public Sub Estadisticas_Adoptante(control As IRibbonControl)
   MsgCaja "Cree su propio informe;)", vbOKOnly + vbInformation
End Sub

'Grupo Vacunas
Public Sub Mostrar_Vacuna(control As IRibbonControl)
   SetParam "VAC_ID", 0
   DoCmd.OpenForm "F_Vacuna"
End Sub

Public Sub Buscar_Vacuna(control As IRibbonControl)
   DoCmd.OpenForm "F_Vacunas"
End Sub

Public Sub Mostrar_Vacunacion(control As IRibbonControl)
   SetParam "VAC_ID", 0
   SetParam "ANI_ID", 0
   DoCmd.OpenForm "F_Vacunacion"
End Sub

'Grupo Caja
Public Sub Mostrar_Caja(control As IRibbonControl)
   SetParam "BOX_ID", 0
   DoCmd.OpenForm "F_Caja"
End Sub

Public Sub Buscar_Caja(control As IRibbonControl)
   DoCmd.OpenForm "F_Cajas"
End Sub

'Salir
Public Sub Salir_Aplicacion(control As IRibbonControl)
   If MsgCaja("¿Está seguro de querer salir de la aplicación?",
vbYesNoCancel) = vbYes Then
```

```
        Application.Quit acQuitSaveAll
    End If
End Sub
```

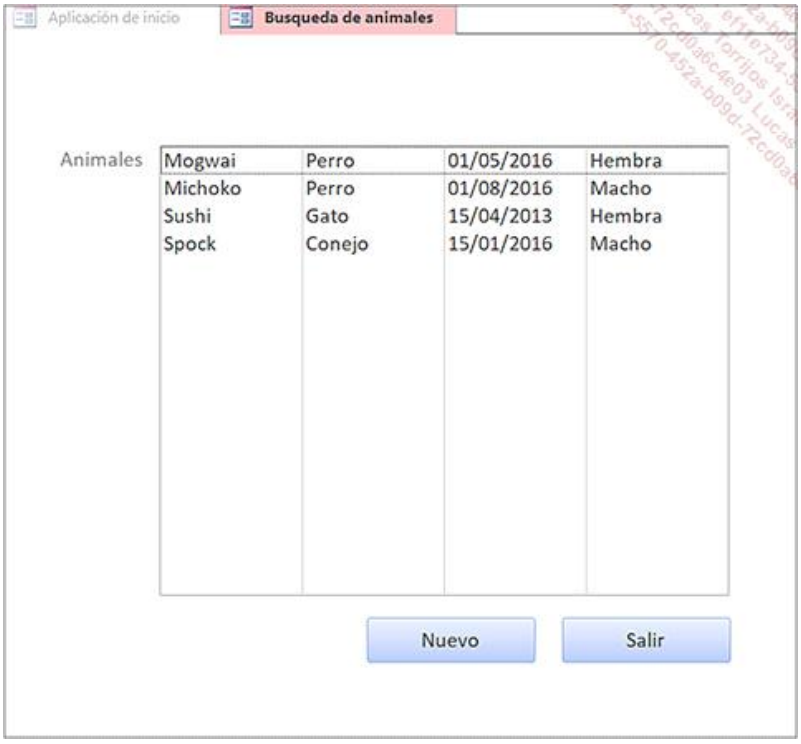
c. Código VBA

La idea es abrir el formulario y ocultar los avisos (en caso de adición, eliminación o actualización de datos en las tablas de la base de datos).

```
Option Compare Database
Option Explicit

Private Sub Form_Load()
    DoCmd.SetWarnings False
```

5. Formulario Lista Animales

a. Interfaz



Los elementos visibles son los siguientes:

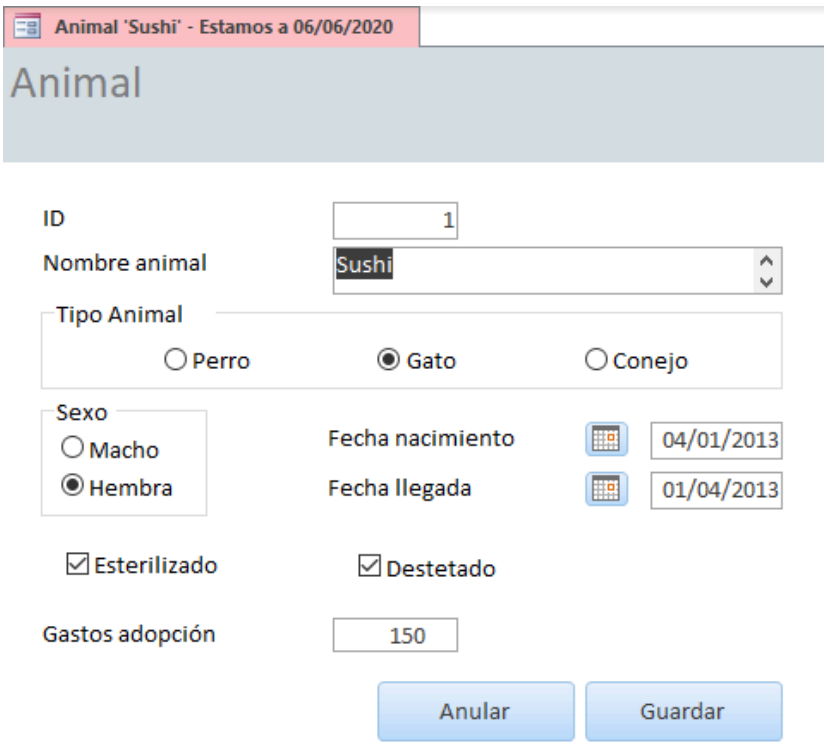
Control	Descripción
Lst_Animales	Zona de lista que representa los animales clasificados por orden de identificador.
Btn_Nuevo	Botón que permite crear un nuevo animal en la base de datos, abriendo el formulario F_Animal.
Btn_Cerrar	Botón que permite cerrar el formulario.

b. Código VBA

```
Option Compare Database
Option Explicit
Private Sub BtnCerrar_Click()
   DoCmd.Close acForm, Me.Name
End Sub

Private Sub BtnNuevo_Click()
   SetParam "ANI_ID", 0
   DoCmd.OpenForm "F_Animal", acNormal, , , , acWindowNormal
End Sub

Private Sub LstAnimales_DblClick(Cancel As Integer)
If Not IsNull(Me.LstAnimales) Then
   SetParam "ANI_ID", Me.LstAnimales.Value
   DoCmd.OpenForm "F_Animal", acNormal, , , , acWindowNormal
End If
End Sub
```

6. Formulario Animal

a. Interfaz



Los elementos visibles son los siguientes (no exhaustivo):

Control	Descripción
ANI_ID	Zona de texto utilizada por el identificador del animal (en modo solo lectura).
CadSexo	Grupo de Opción para el sexo del animal.
CadTipo	Grupo de Opción para la raza del animal.
ChkDestetado	Casilla de selección para indicar si el animal está destetado.
ChkEsterilizacion	Casilla de selección para la esterización del animal.

Txt_Nombre	Zona de texto para el nombre del animal.
Txt_Fec_llegada	Zona de texto utilizada para la fecha de llegada al refugio.
Txt_Fec_Nacimiento	Zona de texto utilizada para la fecha de nacimiento del animal.
TxtGastos	Zona de texto utilizada para los gastos de adopción a pagar para el animal.
BtnCancelar	Botón que permite cerrar el formulario sin guardar.
BtnGuardar	Botón que permite guardar las modificaciones y el caso de cerrar el formulario.

b. CintaOpciones dedicada

Aspecto visual de la cinta de opciones

Podemos ver que la cinta de opciones permite elegir el tipo de animal entre los tres que acoge el refugio (perro, gato y conejo), el sexo del animal, las fechas de nacimiento y llegada al refugio. También permite indicar que el animal ha encontrado un adoptante.

Código XML de la cinta de opciones

```
<customUI xmlns="http://schemas.microsoft.com/office/2009/07/customui"
onLoad="SobreCargaCintaOpciones">
   <ribbon startFromScratch="true">
       <tabs>
<!-- Bloque Animal  -->
           <tab id="TabAnimal" label="Animal">
<!-- Grupo Tipo animal -->
               <group id="Grupo_Tipo" label=" Tipo de animal">
                   <button id="Btn_Perro" label="Perro" size="large"
onAction="Elegir_Perro" tag="Logo_Perro.jpg"
getImage="CintaOpciones_Recupera_Imagen" />
                   <button id="Btn_Gato" label="Gato" size="large"
onAction="Elegir_Gato" tag="Logo_Gato.jpg"
getImage="CintaOpciones_Recupera_Imagen"/>
                   <button id="Btn_Conejo" label="Conejo" size="large"
onAction="Elegir_Conejo" tag="Logo_Conejo.jpg"
getImage="CintaOpciones_Recupera_Imagen"/>
               </group>
<!-- Grupo Sexo -->
               <group id="Grupo_Adoptante" label="Sexo">
                   <button id="Btn_Hembra" label="Hembra" size="large"
onAction="Elegir_Hembra" tag="Logo_Hembra.jpg"
getImage="CintaOpciones_Recupera_Imagen"/>
                   <button id="Btn_Macho" label="Macho" size="large"
onAction="Elegir_Macho" tag="Logo_Macho.jpg"
getImage="CintaOpciones_Recupera_Imagen"/>
               </group>
<!-- Grupo Dates -->
               <group id="Grupo_Fecha" label="fechas">
```

```
                    <button id="Btn_Fecha_Nacimiento" label="Nacimiento"
size="large" onAction="Elegir_Nacimiento" imageMso="AccessListEvents"/>
                    <button id="Btn_Fecha_Llegada" label="Llegada" size="large"
onAction="Elegir_Llegada" imageMso="AccessListEvents"/>
                </group>
<!-- Grupo Adopción -->
                <group id="Grupo_Adopcion" label="Adopcion">
                    <button id="Btn_Caja_Agrega" label="Adoptante encontrado"
size="large" onAction="Activar_Adopcion" imageMso="HappyFace"/>
                </group>
<!-- Grupo Guardar -->
                <group id="Grupo_Guardar" label="Guardar">
                    <button id="Btn_Guardar" label="Guardar"
size="large" onAction="Guardar_Animal" imageMso="CreateWorkflow"/>
                </group>

<!-- Grupo Cerrar -->
                <group id="Grupo_Cerrar" label="Cerrar">
                    <button id="Btn_Cerrar" label="Cerrar Formulario"
size="large" onAction="Cerrar_Formulario" imageMso="CancelRequest"/>
                </group>

            </tab>
        </tabs>
    </ribbon>
</customUI>
```

Código VBA de la cinta de opciones

```
'Animal
Public Sub Elegir_Perro(control As IRibbonControl)
    Form_F_Animal.CadTipo = 1
    Form_F_Animal.CadTipo_AfterUpdate
End Sub

Public Sub Elegir_Gato(control As IRibbonControl)
    Form_F_Animal.CadTipo = 2
    Form_F_Animal.CadTipo_AfterUpdate
End Sub

Public Sub Elegir_Conejo(control As IRibbonControl)
    Form_F_Animal.CadTipo = 3
    Form_F_Animal.CadTipo_AfterUpdate
End Sub

Public Sub Elegir_Macho(control As IRibbonControl)
    Form_F_Animal.CadSexo = 2
    Form_F_Animal.CadSexo_AfterUpdate
End Sub

Public Sub Elegir_Hembra(control As IRibbonControl)
    Form_F_Animal.CadSexo = 1
    Form_F_Animal.CadSexo_AfterUpdate
End Sub

Public Sub Elegir_Nacimiento(control As IRibbonControl)
    Form_F_Animal.BtnNacimiento_Click
End Sub

Public Sub Elegir_Llegada(control As IRibbonControl)
```

```
   Form_F_Animal.BtnLlegada_Click
End Sub

Public Sub Guardar_Animal(control As IRibbonControl)
   Form_F_Animal.Guardar True
End Sub

Public Sub Activar_Adopcion(control As IRibbonControl)
   Form_F_Animal.Guardar False
   DoCmd.OpenForm "F_Adopcion"
End Sub
```

c. Código VBA

```
Option Compare Database
Option Explicit

'variables temporales
Public oAnimal As clsAnimal
Private pANI_ID As Long

'Botones
Private Sub BtnGuardar_Click()
   Guardar True
End Sub

Public Sub Guardar(Cerrar As Boolean)
   'si el animal tiene bien todos los elementos necesarios, se registra
la información
   If oAnimal.EsValido Then
       pANI_ID = oAnimal.Guardar
       If Cerrar Then
           DoCmd.Close acForm, Me.Name
       Else
           SetParam "ANI_ID", pANI_ID
           SetParam "ADO_ID", oAnimal.AdoptanteID
           SetParam "DT_ADOPCION", oAnimal.FechaAdopcion
       End If
'si el formulario de lista de los animales está abierto,
actualizamos la lista de la información.
       If CurrentProject.AllForms("F_Animales").IsLoaded Then
           Form_F_Animales.LstAnimales.Requery
       End If
   Else
       MsgCaja "La información está incompleta o es incorrecta",
vbCritical + vbOKOnly
   End If
                                                          End Sub

'se cierra el formulario sin guardar
Private Sub BtnCancelar_Click()
   DoCmd.Close acForm, Me.Name
End Sub

'se muestra el calendario para la fecha de llegada del animal
Public Sub BtnLlegada_Click()
   Me.Txt_Fc_Llegada.Value = DisplayCalendar(Me.BtnLlegada, "Por favor, rellene una fecha de llegada", Date, , , True)
   If IsDate(Me.Txt_Fc_Llegada.Value) Then
       oAnimal.FechaLlegada = CDate(Me.Txt_Fc_Llegada.Value)
   Else
```

```
       oAnimal.FechaLlegada = DT_DEFAULT
   End If
End Sub

'se muestra el calendario para la fecha de nacimiento del animal
Public Sub BtnNacimiento_Click()
   Me.Txt_Fec_Nacimiento.Value = DisplayCalendar(Me.BtnNacimiento,
" Por favor, rellene una fecha de nacimiento ", Date, , , True)
   If IsDate(Me.Txt_Fec_Nacimiento.Value) Then
       oAnimal.FechaNacimiento = CDate(Me.Txt_Fec_Nacimiento.Value)
   Else
       oAnimal.FechaNacimiento = DT_DEFAULT
   End If
End Sub

'marco para la selección macho/hembra
Public Sub CadSexo_AfterUpdate()
   oAnimal.Sexo = Me.CadSexo
End Sub

'marco para el tipo de animal perro/gato/conejo
Public Sub CadTipo_AfterUpdate()
  oAnimal.TipoAnimalID = Me.CadTipo
End Sub

'casilla para el lactante del animal
Private Sub ChkDestetado_AfterUpdate()
   oAnimal.Destetado = Me.ChkDestetado.Value
End Sub

'casilla para la esterilización del animal
Private Sub ChkEsterilizacion_AfterUpdate()
   oAnimal.Esterilizado = Me.ChkEsterilizacion.Value
End Sub


'carga del formulario
Private Sub Form_Load()
   pANI_ID = GetParam("ANI_ID")
   Set oAnimal = New clsAnimal
   oAnimal.CargarDesdeID pANI_ID
   With oAnimal
       Me.ANI_ID.Value = pANI_ID
       Me.Txt_NOMBRE.Value = .Nombre
       Me.Txt_Fec_Nacimiento.Value = IIf(.FechaNacimiento =
DT_DEFAULT, "", .FechaNacimiento)
       Me.Txt_Fc_Llegada.Value = IIf(.FechaLlegada =
DT_DEFAULT, "", .FechaLlegada)
       Me.CadSexo.Value = .Sexo
       Me.CadTipo.Value = .TipoAnimalID
       Me.ChkDestetado.Value = .Destetado
       Me.ChkEsterilizacion.Value = .Esterilizado
       Me.TxtGastos.Value = .GastosAdopcion
   End With
End Sub

'durante la carga de nombre
Private Sub Txt_NOMBRE_AfterUpdate()
   oAnimal.Nombre = Me.Txt_NOMBRE.Value
End Sub

'durante la carga de la cantidad de gastos
```

```
Private Sub TxtGastos_AfterUpdate()
   If IsNumeric(Me.TxtGastos.Value) Then
       oAnimal.GastosAdopcion = CDbl(Me.TxtGastos.Value)
   Else
       MsgCaja "Los gastos deben ser un valor numérico ",
vbCritical + vbOKOnly
       Me.TxtGastos.Value = 0
   End If
End Sub
```

Funciones e instrucciones de VBA

1. Declaración

Lista

Instrucción	Descripción
`Const`	Permite definir una constante simbólica.
`Declare`	Permite declarar una subrutina de DLL.
`Deftype`	Permite definir un tipo por defecto.
`Dim`	Permite definir una variable.
`Let`	Permite asignar un valor a una variable.
`Option Base`	Permite definir el valor más pequeño del índice de una tabla.
`Option Compare`	Permite definir el modo de comparación de los archivos de texto.
`Option Explicit`	Permite imponer la declaración de las variables.
`Option Private Module`	Permite prohibir que el contenido de un módulo sea referenciado desde el exterior de la aplicación.
`Private`	Permite definir una variable o un procedimiento como privado.
`Public`	Permite definir una variable global.
`Redim`	Permite redefinir las dimensiones de una tabla dinámica.
`Set`	Permite asignar un objeto a una variable.
`Static`	Permite definir una variable estática.
`Type`	Permite definir las variables estructuradas.

2. Funciones lógicas

Lista

Función	Descripción
`Choose()`	Devuelve un valor en función de una lista de argumentos.
`Iif()`	Devuelve unos u otros argumentos, según la evaluación de la expresión que entra como argumento.
`IsDate()`	Devuelve un booleano que indica si el valor se puede convertir en fecha.
`IsEmpty()`	Devuelve un booleano que indica si la variable se ha inicializado.
`IsError()`	Devuelve un booleano que indica si el argumento es un valor erróneo.
`IsMissing()`	Devuelve un booleano que indica si se ha pasado un argumento opcional a un procedimiento o a una función.
`IsNull()`	Devuelve un booleano que indica si el argumento es `Null` (no si contiene un dato válido).
`IsNumeric()`	Devuelve un booleano que indica si el valor se puede convertir en un valor numérico.
`IsObject()`	Devuelve un booleano que indica si una variable representa una variable de

	tipo objeto.
Switch()	Devuelve un valor en función de la expresión proporcionada como argumento y cuyo primer valor es True.

Ejemplo

```
Sub Funciones_Logicas()
Dim Edad As String
Dim FcNacimiento As String
Edad = "18" 'después probar con "y" por ejemplo
FcNacimiento = "1973/29/07"
If IsNumeric(Edad) Then
    MsgBox Iif(CInt(Edad) > 18, "Mayor", "Menor")
Else
    If IsDate(FcNacimiento) Then
           MsgBox "No se puede leer su edad, pero nació el día " &
FcNacimiento
    End If
End If
End Sub
```

3. Funciones de cadena

Lista

Función	Descripción
Asc()	Devuelve un valor entero (Integer) correspondiente al código ASCII (norma ANSI) de la primera letra de una cadena de caracteres.
AscB()	Devuelve un valor entero (Byte) correspondiente al código ASCII de la primera letra de una cadena de caracteres.
AscW()	Devuelve un valor entero (Integer) correspondiente al código ASCII (norma Unicode) de la primera letra de una cadena de caracteres.
Chr()	Devuelve una cadena de caracteres (String) que representa el carácter asociado al código ASCII (norma ANSI) del carácter indicado.
ChrB()	Devuelve una cadena de caracteres (String) que representa el carácter asociado al código ASCII (norma ANSI) del carácter indicado.
ChrW()	Devuelve una cadena de caracteres (String) que representa el carácter asociado al código ASCII (norma Unicode) del carácter indicado.
InStr()	Devuelve un valor numérico (Long) que indica la posición de la primera ocurrencia de una cadena de caracteres dentro de otra cadena de caracteres.
InStrRev()	Devuelve un valor numérico (Long) que indica la posición de una ocurrencia de una cadena de caracteres en otra, partiendo del fin de la cadena.
LCase()	Devuelve una cadena de caracteres (String) convertida a minúsculas.
Left()	Devuelve una cadena de caracteres (String) que contiene el número de caracteres indicado de una cadena de caracteres, partiendo de la izquierda.
Len()	Devuelve un valor numérico (Long) que indica el número de caracteres de una cadena de caracteres o el número de bytes necesarios para almacenar una

	variable.
LTrim()	Devuelve una cadena de caracteres (String) que contiene una copia de una cadena de caracteres sin los espacios a la izquierda.
Mid()	Devuelve una cadena de caracteres (String) que contiene un número indicado de caracteres extraídos de una cadena de caracteres.
Replace()	Devuelve una cadena de caracteres en la que una cadena de caracteres especificada se ha sustituido totalmente por otra.
Right()	Devuelve una cadena de caracteres (String) que contiene el número de caracteres indicado de una cadena de caracteres, partiendo de la derecha.
RTrim()	Devuelve una cadena de caracteres (String) que contiene una copia de una cadena de caracteres sin espacios a la derecha.
Space()	Devuelve una cadena de caracteres (String) que se compone del número de espacios indicado.
Str()	Devuelve una cadena de caracteres (String) que representa un valor numérico.
StrComp()	Devuelve un valor numérico (Integer) que indica el resultado de una comparación de dos cadenas de caracteres.
StrConv()	Devuelve una cadena de caracteres (String) convertida al formato indicado.
String()	Devuelve una cadena de caracteres (String) que contiene una cadena formada por un único carácter repetido a lo largo de la longitud indicada.
StrReverse()	Devuelve una cadena de caracteres que contiene los caracteres cuyo orden se ha invertido respecto a una cadena dada.
Trim()	Devuelve una cadena de caracteres (String) que contiene una copia de una cadena de caracteres sin espacios ni a izquierda ni a derecha.
UCase()	Devuelve una cadena de caracteres (String) que contiene la cadena indicada, convertida en mayúsculas.

Las funciones Asc, AscB, AscW y Chr, ChrB, ChrW están relacionadas con las diferentes normas establecidas a lo largo de su implementación por Microsoft (ANSI y después Unicode).

Ejemplo

```
Sub Funciones_Cadena()
'el programa prueba si una cadena de caracteres
'es un palíndromo cuando se eliminan los espacios
   Dim strEjemplo As String
   strEjemplo = "    Anita lava la tina"
   Debug.Print strEjemplo
   Debug.Print StrReverse(strEjemplo)
   Debug.Print strEjemplo = StrReverse(strEjemplo)
   Debug.Print Ltrim(strEjemplo)
   Debug.Print Rtrim(strEjemplo)
   Debug.Print Trim(strEjemplo)
   Debug.Print Trim(strEjemplo) = Trim(StrReverse(strEjemplo))
   Debug.Print Replace(Trim(LCase(strEjemplo)), " ", "") = _
         Replace(Trim(StrReverse(LCase(strEjemplo))), " ", "")
End Sub
```

4. Funciones de fecha

Lista

Función	Descripción
Date()	Devuelve una fecha (Date) que contiene la fecha actual del sistema.
DateAdd()	Devuelve una fecha (Date) a la que se añade o quita un periodo de tiempo especificado.
DateDiff()	Devuelve un valor numérico entero (Long) que indica el tiempo transcurrido entre dos fechas dadas.
DatePart()	Devuelve un valor numérico entero (Integer) que representa el elemento especificado de una fecha dada.
DateSerial()	Devuelve una fecha (Date) correspondiente a un año, mes y día determinados.
DateValue()	Devuelve una fecha (Date) correspondiente a una cadena de caracteres convertible en fecha.
Day()	Devuelve un valor numérico entero (Integer) que indica un nombre comprendido entre 1 y 31 incluidos, que representa el día del mes.
Hour()	Devuelve un valor numérico (Integer) que indica un número comprendido entre 0 y 23 incluidos, que representa la hora del día.
Minute()	Devuelve un valor numérico entero (Integer) que indica un nombre comprendido entre 0 y 59 incluidos, que representa los minutos de la hora actual.
Month()	Devuelve un valor numérico entero (Integer) que indica un número comprendido entre 1 y 12 incluidos, que representa el mes del año.
MonthName()	Devuelve una cadena de caracteres que indica el mes especificado en el idioma de la aplicación.
Now()	Devuelve una fecha (Date) que indica la fecha y la hora de sistema del ordenador.
Second()	Devuelve un valor numérico entero (Integer) que indica un número comprendido entre 0 y 59 incluidos, que representa los segundos del minuto actual.
Time()	Devuelve una fecha (Date) que indica la hora de sistema actual.
Timer()	Devuelve un valor numérico decimal (Single) que representa el número de segundos transcurridos desde medianoche.
TimeSerial()	Devuelve una fecha (Date) que contiene una hora concreta (hora, minutos y segundos).
TimeValue()	Devuelve una fecha (Date) que contiene una hora.
Weekday()	Devuelve un valor numérico entero (Integer) que contiene un número que representa el día de la semana.
WeekdayName()	Devuelve una cadena de caracteres que indica el día de la semana especificado en el idioma de la aplicación.
Year()	Devuelve un valor numérico entero (Integer) que contiene un número que representa el año.

Ejemplo

```
Sub Funciones_Fecha()
'el programa determina si el cumpleaños
'de una persona, cuya fecha de nacimiento se conoce,
'ya ha pasado este año o no, deduciendo
```

```
'su edad, así como la fecha de su próximo cumpleaños
'puede modificar la fecha de nacimiento para verificación
Dim fc_nacimiento As Date
Dim fc_siguiente_cumpleanio As Date
Dim strMensaje As String
fc_nacimiento = #1973/07/29#
strMensaje = "La persona nació el " & fc_nacimiento & " su edad es "
If Month(fc_nacimiento) > Month(Date) Then
    'cumpleaños todavía no ha pasado
    fc_siguiente_cumpleanio = DateSerial(Year(Date), _
Month(fc_nacimiento), _
          Day(fc_nacimiento))
    Debug.Print strMensaje & Year(Date) - Year(fc_nacimiento) - 1 & " años"
ElseIf Month(fc_nacimiento) = Month(Date) Then
    If Day(fc_nacimiento) > Day(Date) Then
           'cumpleaños todavía no ha pasado
           fc_siguiente_cumpleanio = DateSerial(Year(Date), _
Month(fc_nacimiento), _
                  Day(fc_nacimiento))
           Debug.Print strMensaje & Year(Date) - Year(fc_nacimiento) -
1 & " años”
    Else
           'cumpleaños ha pasado
           fc_siguiente_cumpleanio = DateSerial(Year(Date) + 1, _
Month(fc_nacimiento), _
                  Day(fc_nacimiento))
           Debug.Print strMensaje & Year(Date) - Year(fc_nacimiento)
& _
" años”
    End If
Else 'Month(fc_nacimiento) < Month(Date)
    'cumpleaños ha pasado
    fc_siguiente_cumpleanio = DateSerial(Year(Date) + 1, _
Month(fc_nacimiento), _
          Day(fc_nacimiento))
    Debug.Print strMensaje & Year(Date) - Year(fc_nacimiento) & " años"
End If
    Debug.Print "Falta " & DateDiff("d", Date, _
fc_siguiente_cumpleanio) & _
          " días antes del próximo cumpleaños"
End Sub
```

5. Funciones matemáticas

Lista

Función	Descripción
Abs()	Devuelve el valor absoluto de un número, del mismo tipo que el recibido por parámetro.
Atn()	Devuelve un valor numérico decimal (Double) que indica la arcotangente de un número.

Cos()	Devuelve un valor numérico decimal (Double) que indica el coseno de un ángulo.
Exp()	Devuelve un valor numérico decimal (Double) que indica el valor del exponente en base e (base de los logaritmos neperianos), elevado a una potencia.
Fix()	Devuelve la parte entera de un número.
Hex()	Devuelve una cadena de caracteres (String) que representa un número en forma hexadecimal.
Int()	Devuelve la parte entera de un número.
Log()	Devuelve un valor numérico decimal (Double) que indica el logaritmo neperiano de un número.
Oct()	Devuelve una cadena de caracteres (String) que representa el valor octal de un número.
Partition()	Devuelve una cadena de caracteres (String) que indica el lugar en el que un número aparece dentro de una serie calculada de rangos de valores.
Rnd()	Devuelve un valor numérico decimal (Single) que contiene un número aleatorio.
Round()	Devuelve un número redondeado a un número especificado de cifras decimales.
Sgn()	Devuelve un valor numérico entero (Integer) que indica el signo de un número.
Sin()	Devuelve un valor numérico decimal (Double) que indica el seno de un número.
Sqr()	Devuelve un valor numérico decimal (Double) que indica la raíz cuadrada de un número.
Tan()	Devuelve un valor numérico decimal (Double) que indica la tangente de un ángulo.
Val()	Devuelve el número contenido en una cadena de caracteres, en forma de un valor numérico del tipo apropiado.

La diferencia entre `Int` y `Fix` se centra en los números negativos, donde `Fix` devuelve la parte entera superior (`Fix(-5.2)=-5`), mientras que `Int` devuelve la parte entera inferior (`Int(-5.2)=-6`).

Hay funciones trigonométricas más elaboradas, que se pueden formular partiendo de las funciones básicas que VBA proporciona.

Ejemplo

```
Sub Funciones_Math()
'Se trabaja en la ecuación ax²+bx+c=0, para determinar los valores
de las soluciones
Dim a As Single
Dim b As Single
Dim c As Single
Dim delta As Single
Dim x1 As Single
Dim x2 As Single
a = 8: b = 10: c = 2  'y = 8x²+10x+2
delta = b * b - 4 * a * c
If delta > 0 Then
    x1 = (b - Sqr(delta)) / (2 * a)
    x2 = (b + Sqr(delta)) / (2 * a)
    Debug.Print "2 soluciones: "
    Debug.Print "x1=" & x1
    Debug.Print "x2=" & x2
```

```
ElseIf delta = 0 Then
    x1 = b / (2 * a)
    Debug.Print "1 solución: "
    Debug.Print "x1=" & x1
Else
    Debug.Print "ninguna solución"
End If
End Sub
```

6. Funciones financieras

Lista

Función	Descripción
DDB()	Devuelve un valor numérico decimal (Double) que indica la amortización de un bien durante un período específico, usando el método de la amortización regresiva de doble ritmo o cualquier otro método indicado.
FV()	Devuelve un valor numérico decimal (Double) que indica la cantidad futura de una anualidad basada en pagos constantes y periódicos y en una tasa de interés fija.
IPmt()	Devuelve un valor numérico decimal (Double) que indica la cantidad, en un período dado, de una anualidad basada en pagos constantes y periódicos y en una tasa de interés fija.
IRR()	Devuelve un valor numérico decimal (Double) que indica la tasa de rendimiento interno de una serie de movimientos de tesorería periódicos (pagos e ingresos).
MIRR()	Devuelve un valor numérico decimal (Double) que indica la tasa de rendimiento interna modificada de una serie de movimientos de tesorería periódicos (pagos e ingresos).
Nper()	Devuelve un valor numérico decimal (Double) que indica el número de plazos de una anualidad basada en los pagos constantes y periódicos y en una tasa de interés fijo.
NPV()	Devuelve un valor numérico decimal (Double) que indica el valor neto actual de una inversión, calculada en función de una serie de movimientos de tesorería periódicos (pagos e ingresos) y de una tasa de descuento.
Pmt()	Devuelve un valor numérico decimal (Double) que indica la cantidad de una anualidad basada en pagos constantes y periódicos y en una tasa de interés fija.
PPmt()	Devuelve un valor numérico decimal (Double) que indica el reembolso del capital, en un plazo dado, de una anualidad basada en pagos constantes y periódicos y en una tasa de interés fija.
PV()	Devuelve un valor numérico decimal (Double) que indica la cantidad actual de una anualidad basada en plazos futuros constantes y periódicos y en una tasa de interés fija.
Rate()	Devuelve un valor numérico decimal (Double) que indica la tasa de interés en un plazo para una anualidad.
SLN()	Devuelve un valor numérico decimal (Double) que indica la amortización lineal de un bien en un período dado.
SYD()	Devuelve un valor numérico decimal (Double) que indica la amortización global de un bien en un período dado.

Ejemplo

```
Sub Funciones_Financieras()
'Cálculo de la cantidad de reembolso de un
préstamo
```

```
Dim Cantidad_Prestada As Double, Tasa_Anual As Double,
Reembolso As Double, Numero_Reembolsos As Double
Cantidad_Prestada = CDbl(InputBox("¿Cuál es la cantidad prestada?"))
Tasa_Anual = CDbl(InputBox("¿Cuál es la cantidad anual
del préstamo?"))
If Tasa_Anual > 1 Then Tasa_Anual = Tasa_Anual / 100
    ' Nos aseguramos de tener una tasa en porcentaje.
    Numero_Reembolsos = CDbl(InputBox("¿Cuántos
reembolsos se realizarán?"))
    Reembolso = Pmt(Tasa_Anual / 12, Numero_Reembolsos,
-Cantidad_Prestada, 0, 0)
    MsgBox ("Su reembolso será de " & Round(Reembolso, 2)
& "€ por mes.")
End Sub
```

7. Funciones de archivo

Lista

Función	Descripción
CurDir()	Devuelve una cadena de caracteres (String) que indica la ruta actual.
Dir()	Devuelve una cadena de caracteres (String) que representa el nombre de un archivo, de una carpeta o de una carpeta correspondiente a una cadena de caracteres de búsqueda, a un atributo de archivo o al nombre de unidad de un lector.
EOF()	Devuelve un valor numérico entero (Integer) que contiene el valor booleano True cuando se llega al fin de un archivo abierto en modo Random o Input secuencial.
FileAttr()	Devuelve un valor numérico entero (Long) que representa el modo de apertura de los archivos con la instrucción Open.
FileDateTime()	Devuelve una fecha (Date) que indica la fecha y hora de creación o de última modificación de un archivo.
FileLen()	Devuelve un valor numérico entero (Long) que indica la longitud en bytes de un archivo.
FreeFile()	Devuelve un valor numérico entero (Integer) que representa el siguiente número de archivo que se puede utilizar por la instrucción Open.
GetAttr()	Devuelve un valor numérico entero (Integer) que indica los atributos del archivo o de la carpeta.
Input()	Devuelve una cadena de caracteres (String) que contiene los caracteres que se leen de un archivo abierto en modo Input o Binary.
Loc()	Devuelve un valor numérico entero (Long) que indica la posición de lectura/escritura actual en un archivo abierto.
LOF()	Devuelve un valor numérico entero (Long) que representa el tamaño, expresado en bytes, de un archivo abierto con la instrucción Open.
Seek()	Devuelve un valor numérico entero (Long) que indica la posición de lectura/escritura actual en un archivo abierto con la instrucción Open.

Ejemplo

```
Sub Funciones_Archivo()
'el siguiente programa abre el archivo
'y muestra cada una de las líneas
Dim strArchivo As String
Dim strLinea As String
strArchivo = CurrentProject.Path & "\ejemplo_anexo.txt"
    If Dir(strArchivo) <> "" Then
          Open strArchivo For Input As #1
          Do Until EOF(1)
                Line Input #1, strLinea
                Debug.Print strLinea
          Loop
          Close #1
    End If
End Sub
```

8. Funciones de conversión

Lista

Función	Descripción
CBool()	Permite convertir un valor numérico cualquiera en booleano (Boolean).
CByte()	Permite convertir un valor numérico entero o una cadena de caracteres que representa un número comprendido entre 0 y 255 en bytes (Byte).
CCur()	Permite convertir un valor numérico o una cadena de caracteres que representa un número incluido en el rango del tipo Currency, en tipo Currency.
CDate()	Permite convertir un valor numérico o una cadena de caracteres que representa una fecha válida, en fecha (Date).
CDbl()	Permite convertir un valor numérico o una cadena de caracteres que representa un número incluido en el rango del tipo Double, en valor numérico decimal (Double).
CInt()	Permite convertir un valor numérico o una cadena de caracteres que representa un número comprendido entre -32768 y 32767 en valor numérico entero (Integer).
CLng()	Permite convertir un valor numérico o una cadena de caracteres que representa un número incluido en el rango del tipo Long en un valor numérico entero (Long).
CSng()	Permite convertir un valor numérico o una cadena de caracteres que representa un número incluido en el rango del tipo Single, en un valor numérico decimal (Single).
CStr()	Permite convertir un valor cualquiera en cadena de caracteres (String).

Ejemplo

```
Sub Funcion_Conversion()
    Dim strEjemplo As String
    strEjemplo = "12345,6789"
    Debug.Print CByte(CInt(strEjemplo) \ 100)
    Debug.Print CCur(strEjemplo)
    Debug.Print CDbl(strEjemplo)
```

```
    Debug.Print CInt(strEjemplo)
    Debug.Print CLng(strEjemplo)
End Sub
```

9. Funciones de sistema

Lista

Función	Descripción
Command()	Devuelve una cadena de caracteres (String) que representa la parte de argumentos de la línea de comandos, utilizada para ejecutar Microsoft Visual Basic o un programa ejecutable desarrollado con Visual Basic.
DoEvents()	Permite detener momentáneamente la ejecución para que el sistema operativo pueda tratar otros eventos.
Environ()	Devuelve la cadena de caracteres (String) asociada a una variable de entorno del sistema operativo. No disponible en Macintosh.
GetAllSettings()	Devuelve una lista de claves y sus valores respectivos (creados originalmente en la ayuda de la instrucción SaveSetting), que figura en una entrada de aplicación de la base de registro de Windows.
GetSetting()	Devuelve un valor de clave de una entrada de aplicación de la base de registro de Windows.
IMEStatus()	Devuelve un valor numérico entero (Integer) que indica el modo IME (*Input Method Editor*) actual de Microsoft Windows; disponible únicamente en las versiones destinadas a los países asiáticos.
QBColor()	Devuelve un valor numérico entero (Long) que indica el código de color RGB correspondiente al número de color indicado.
RGB()	Devuelve un valor numérico entero (Long) que representa el código RGB.
Shell()	Permite ejecutar un programa ejecutable y enviar un valor de tipo numérico real (Double) que representa el identificador (ID) de la tarea ejecutada en caso de éxito, o un cero en caso de error.
TypeName()	Devuelve una cadena de caracteres (String) que proporciona información sobre una variable.
VarType()	Devuelve un valor numérico entero (Integer) que indica el subtipo de una variable.

10. Funciones de tabla

Lista

Función	Descripción
Array()	Devuelve una variable de tipo Variant que contiene una tabla.
Filter()	Devuelve una tabla de base cero, que contiene un subconjunto de una tabla de cadenas basada en los criterios de filtro especificados.
IsArray()	Devuelve un valor booleano (Boolean) que indica si una variable es una tabla.
Join()	Devuelve una cadena de caracteres (String) creada para la adición de varias subcadenas contenidas en una tabla.

LBound()	Devuelve un valor numérico entero (Long) que contiene el índice más pequeño disponible para la dimensión indicada de una tabla.
Split()	Devuelve una tabla de base cero a una dimensión que contiene el número especificado de subcadenas.
UBound()	Devuelve un valor numérico entero (Long) que contiene el índice más grande disponible para la dimensión indicada de una tabla.

Ejemplo

Ver el capítulo El lenguage VBA - Las tablas.

11. Funciones de administración de objetos

Lista

Función	Descripción
CallByName()	Ejecuta un método de un objeto o define o devuelve una propiedad de un objeto.
CreateObject()	Crea y devuelve una referencia a un objeto ActiveX.
GetObject()	Devuelve una referencia a un objeto proporcionado por un componente ActiveX.

Ejemplo

Ver el capítulo Los objetos y colecciones en VBA - Objetos de Access.

12. Funciones e instrucciones de administración de errores

Funciones

Función	Descripción
CVErr()	Devuelve un dato de tipo Variant y de subtipo Error que contiene un número de error especificado por el usuario.
Err()	Devuelve un dato entero (Integer) correspondiente al código de error.
Error()	Devuelve el mensaje de error correspondiente a un número de error dado.
IsError()	Devuelve un valor booleano (Boolean) que indica si la variable es un valor de error.

Instrucciones

Instrucción	Descripción
Error	Permite simular un error.
On Error	Permite desencadenar una acción en caso de que aparezca un error.
Resume	Permite continuar con el código después de tratar un error.

Ejemplo

Ver el capítulo El lenguaje VBA - El comportamiento de VBA en caso de error.

13. Funciones de formateo

Lista

Función	Descripción
Format()	Devuelve una cadena de caracteres (String) que contiene una expresión formateada en función de las instrucciones contenidas en la expresión de formateo.
FormatCurrency()	Devuelve una expresión formateada en forma de valor numérico real (Currency) utilizando el símbolo monetario definido en el Panel de configuración del sistema.
FormatDateTime()	Devuelve una expresión formateada en forma de fecha u hora.
FormatNumber()	Devuelve una expresión formateada en forma de número.
FormatPercent()	Devuelve una expresión formateada en forma de porcentaje (multiplicado por 100) con un carácter % de final.

Ejemplo

```
Sub Funciones_Format()
Dim valor_num As Double
Dim valor_date As Date
valor_numero = 12345.6789
valor_fecha = Now
Debug.Print Format(valor_numero, "000 000 000.00")
Debug.Print FormatCurrency(valor_numero)
Debug.Print FormatDateTime(valor_fecha, vbGeneralDate)
Debug.Print FormatDateTime(valor_fecha, vbShortDate)
Debug.Print FormatNumber(valor_numero, 1)
Debug.Print FormatPercent(valor_numero, 1)
End Sub
```

14. Funciones de interfaz de usuario

Lista

Función	Descripción
InputBox	Permite visualizar una línea de comandos en un cuadro de diálogo, esperar que el usuario escriba el texto o haga clic en un botón y después envíe el contenido de la zona de texto en forma de una cadena de caracteres (String).
MsgBox	Permite visualizar un mensaje en un cuadro de diálogo, esperar que el usuario haga clic en un botón y después enviar un valor numérico entero (Integer) que indique el botón elegido por el usuario.

Ejemplo

Ver el capítulo El lenguaje VBA - Las entradas-salidas en VBA.

Funciones y procedimientos de VBA Access

1. Funciones SQL

Lista

Función	Descripción
`Avg()`	Devuelve el valor medio de un campo de un conjunto de registros.
`Count()`	Devuelve el número de registros de un conjunto de registros.
`First()`	Devuelve el primer valor de un campo de un conjunto de registros.
`Last()`	Devuelve el último valor de un campo de un conjunto de registros.
`Max()`	Devuelve el valor máximo de un campo de un conjunto de registros.
`Min()`	Devuelve el valor mínimo de un campo de un conjunto de registros.
`StDev()`	Devuelve una estimación de la desviación típica de un campo de una población de un conjunto de registros.
`StDevP()`	Devuelve una estimación de la desviación típica de un campo de una muestra de una población de un conjunto de registros.
`Sum()`	Devuelve la suma de los valores de un campo de un conjunto de registros.
`Var()`	Devuelve una estimación de la varianza de un campo de una población de un conjunto de registros.
`VarP()`	Devuelve una estimación de la varianza de un campo de una muestra de población de un conjunto de registros.

Ejemplo

Ver el capítulo El lenguaje SQL aplicado a Access.

2. Métodos Docmd

Métodos	Descripción
`AddMenu`	Ejecuta la acción **AñadirMenu** en Visual Basic.
`ApplyFilter`	Ejecuta la acción **AplicarFiltro** en Visual Basic.
`Beep`	Ejecuta la acción **Bip** en Visual Basic.
`BrowseTo`	Realiza la acción **Examinar** en Visual Basic.
`CancelEvent`	Ejecuta la acción **AnularEvento** en Visual Basic.
`ClearMacroError`	Elimina la información relativa a un error que se almacena en el objeto `MacroError`.
`Close`	Ejecuta la acción **SalirEn** en Visual Basic.
`CloseDatabase`	Cierra la base de datos activa.
`CopyDatabaseFile`	Copia la base de datos conectada al proyecto activo de un archivo de base de datos de Microsoft SQL Server para exportarlo.
`CopyObject`	Ejecuta la acción **CopiarObjeto** en Visual Basic.
`DeleteObject`	Ejecuta la acción **EliminarObjeto** en Visual Basic.

DoMenuItem	Microsoft Access muestra el comando de menú o de barra de herramientas apropiado.
Echo	Ejecuta la acción **Echo** en Visual Basic.
FindNext	Ejecuta la acción **BuscarSiguiente** en Visual Basic.
FindRecord	Ejecuta la acción **BuscarRegistro** en Visual Basic.
GoToControl	Ejecuta la acción **IrAControl** en Visual Basic.
GoToPage	Ejecuta la acción **IrAPagina** en Visual Basic.
GoToRecord	Ejecuta la acción **IrARegistro** en Visual Basic.
Hourglass	Ejecuta la acción **RelojArena** en Visual Basic.
Lock NavigationPane	Utiliza la acción LockNavigationPane para impedir a los usuarios que puedan eliminar los objetos de base de datos que se muestran en la pestaña de navegación.
Maximize	Ejecuta la acción **Maximizar** en Visual Basic.
Minimize	Ejecuta la acción **Minimizar** en Visual Basic.
MoveSize	Ejecuta la acción **MoverDimensionar** en Visual Basic.
NavigateTo	Puede utilizar el método NavigateTo para controlar la visualización de los objetos de base de datos en la pestaña de navegación.
OpenDataAccessPage	Ejecuta la acción **AbrirPaginaAccesoDatos** en Visual Basic.
OpenDiagram	Ejecuta la acción **AbrirEsquema** en Visual Basic.
OpenForm	Ejecuta la acción **AbrirFormulario** en Visual Basic.
OpenFunction	Abre una función de usuario en una base de datos de Microsoft SQL Server para visualizarla en Microsoft Access.
OpenModule	Ejecuta la acción **AbrirModulo** en Visual Basic.
OpenQuery	Ejecuta la acción **AbrirConsulta** en Visual Basic.
OpenReport	Ejecuta la acción **AbrirInforme** en Visual Basic.
OpenStored Procedure	Ejecuta la acción **AbrirProcedimientoAlmacenado** en Visual Basic.
OpenTable	Ejecuta la acción **AbrirTabla** en Visual Basic.
OpenView	Ejecuta la acción **AbrirVista** en Visual Basic.
OutputTo	Ejecuta la acción **CopiarEn** en Visual Basic.
PrintOut	Ejecuta la acción **Imprimir** en Visual Basic.
Quit	Sale de Microsoft Access. Puede seleccionar una de las opciones de registro de un objeto de base de datos antes de salir.
RefreshRecord	Realiza la operación de macro **ActualizarRegistro** desde Visual Basic.
Rename	Ejecuta la acción **Renombrar** en Visual Basic.
RepaintObject	Ejecuta la acción **RepintarObjeto** en Visual Basic.
Requery	Ejecuta la acción **Actualizar** en Visual Basic.
Restore	Ejecuta la acción **Restaurar** en Visual Basic.
RunCommand	Ejecuta un comando integrado.
RunDataMacro	Use el método RunDataMacro para ejecutar una macro de datos, llamada desde Visual Basic.
RunMacro	Ejecuta la acción **EjecutarMacro** en Visual Basic.
RunSaved ImportExport	Permite ejecutar una especificación de importación o exportación guardada.

RunSQL	Ejecuta la acción **EjecutarSQL** en Visual Basic.
Guardar	Ejecuta la acción **Guardar** en Visual Basic.
SearchForRecord	Puede utilizar el método SearchForRecord para buscar un registro específico en una tabla, una consulta, un formulario o un informe.
SelectObject	Ejecuta la acción **SeleccionarObjeto** en Visual Basic.
SendObject	Ejecuta la acción **EnviarObjetoBaseDeDatos** en Visual Basic.
SetDisplayed Categories	Indica las categorías que se muestran en **Ir a la categoría**, en la barra de título de la pestaña de navegación.
SetFilter	Usa el método SetFilter para aplicar un filtro a los registros de la hoja de datos activa (un formulario, un informe o una tabla).
SetMenuItem	Ejecuta la acción **DefinirElementoMenu** en Visual Basic.
SetOrderBy	Usa el método SetOrderBy para aplicar una ordenación en la hoja de datos activa (un formulario, un informe o una tabla).
SetParameter	Usa el método SetParameter para crear un argumento para un uso por los métodos BrowseTo, OpenForm, OpenQuery, OpenReport o RunDataMacro.
SetProperty	Ejecuta la acción **DefinirPropiedad** en Visual Basic.
SetWarnings	Ejecuta la acción **Advertencias** en Visual Basic.
ShowAllRecords	Ejecuta la acción **VerTodosRegistros** en Visual Basic.
ShowToolbar	Ejecuta la acción **VerBarraHerramientas** en Visual Basic.
SingleStep	Suspende la ejecución de la macro activa y abre el cuadro de diálogo **Paso a paso**.
TransferDatabase	Ejecuta la acción **TransferirBase** en Visual Basic.
Transfer SharePointList	Puede utilizar el método TransferSharePointList para importar o relacionar datos de SharePoint Foundation.
Transfer Spreadsheet	Ejecuta la acción **TransferirHojaCalculo** en Visual Basic.
TransferSQL Database	Transfiere la integridad de la base de datos de Microsoft SQL Server especificada a otra base de datos de SQL Server.
TransferText	Ejecuta la acción **ImportarExportarTexto** en Visual Basic.

3. Funciones de dominio

Función	Descripción
DAvg()	Devuelve la media de un dominio.
DCount()	Devuelve el número de elementos de un dominio.
DFirst()	Devuelve el primer valor de un dominio.
DLast()	Devuelve el último valor de un dominio.
DLookup()	Devuelve el valor buscado de un dominio.
DMax()	Devuelve el valor máximo de un dominio.
DMin()	Devuelve el valor mínimo de un dominio.
DStdev()	Devuelve una estimación de la desviación típica de la población de un dominio.
DStdevP()	Devuelve una estimación de la desviación típica de una muestra de la población de un dominio.

DSum()	Devuelve la suma de los valores de un dominio.
DVar()	Devuelve una estimación de la varianza de una población de un dominio.
DVarP	Devuelve una estimación de la varianza de una muestra de la población de un dominio.

4. Funciones e instrucciones de intercambio dinámico de datos

Funciones

Función	Descripción
DDE()	Permite crear un canal DDE y solicitar una información.
DDEInitiate()	Permite enviar un comando usando un canal DDE.
DDERequest()	Permite solicitar una información usando un canal DDE.
DDESend()	Permite enviar una información usando un canal DDE.

Instrucciones

Función	Descripción
DDEExecute	Permite utilizar un canal DDE abierto para enviar un comando a una aplicación.
DDEPoke	Permite utilizar un canal DDE abierto para enviar datos a una aplicación.
DDETerminate	Permite cerrar un canal DDE.
DDETerminateAll	Permite cerrar todos los canales DDE abiertos.

Constantes

1. Constantes de VBA - constantes vb

a. Constantes de textos

Constante	Valor	Descripción
vbBack	Chr(8)	Corresponde al un retroceso del teclado.
vbCr	Chr(13)	Corresponde a un retorno de carro (Intro).
vbCrLf	Chr(13) & Chr(10)	Corresponde al retorno de carro (Intro) y salto de línea.
vbFormFeed	Chr(12)	Corresponde al salto de página.
vbLf	Chr(10)	Corresponde al salto de línea.
vbNewLine	Chr(13) & Chr(10) & Chr(10)	Corresponde a una nueva línea.
vbNullChar	Chr(0)	Corresponde a un carácter Null.
vbNullString	""	Corresponde a una cadena de caracteres vacía.
vbTab	Chr(9)	Corresponde a una tabulación horizontal.
vbVerticalTab	Chr(11)	Corresponde a una tabulación vertical.

b. Constantes de fechas

Constante	Valor	Descripción
vbMonday	2	Lunes
vbTuesday	3	Martes
vbWednesday	4	Miércoles
vbThursday	5	Jueves
vbFriday	6	Viernes
vbSaturday	7	Sábado
vbSunday	1	Domingo

c. Constantes de colores

Constante	Valor hexadecimal	Descripción
vbBlack	&h00	Negro
vbRed	&hFF	Rojo
vbGreen	&hFF00	Verde
vbYellow	&hFFFF	Amarillo
vbBlue	&hFF0000	Azul
vbMagenta	&hFF00FF	Magenta
vbCyan	&hFFFF00	Cian

vbWhite	&hFFFFFF	Blanco

d. Constantes de botones

Constante	Valor	Descripción
vbOK	1	OK
vbCancel	2	Anular
vbAbort	3	Cancelar
vbRetry	4	Volver a intentar
vbIgnore	5	Ignorar
vbYes	6	Sí
vbNo	7	No

e. Constantes de teclas de teclado

Teclas alfanuméricas

Constante	Valor	Descripción
vbKeyA	65	Tecla A
vbKeyB	66	Tecla B
vbKeyC	67	Tecla C
vbKeyD	68	Tecla D
vbKeyE	69	Tecla E
vbKeyF	70	Tecla F
vbKeyG	71	Tecla G
vbKeyH	72	Tecla H
vbKeyI	73	Tecla I
vbKeyJ	74	Tecla J
vbKeyK	75	Tecla K
vbKeyL	76	Tecla L
vbKeyM	77	Tecla M
vbKeyN	78	Tecla N
vbKeyO	79	Tecla O
vbKeyP	80	Tecla P
vbKeyQ	81	Tecla Q
vbKeyR	82	Tecla R
vbKeyS	83	Tecla S
vbKeyT	84	Tecla T
vbKeyU	85	Tecla U
vbKeyV	86	Tecla V

vbKeyW	87	Tecla W
vbKeyX	88	Tecla X
vbKeyY	89	Tecla Y
vbKeyZ	90	Tecla Z

Teclas numéricas

Constante	Valor	Descripción
vbKey0	48	Tecla 0
vbKey1	49	Tecla 1
vbKey2	50	Tecla 2
vbKey3	51	Tecla 3
vbKey4	52	Tecla 4
vbKey5	53	Tecla 5
vbKey6	54	Tecla 6
vbKey7	55	Tecla 7
vbKey8	56	Tecla 8
vbKey9	57	Tecla 9

Teclas del teclado numérico

Constante	Valor	Descripción
vbKeyNumpad0	96	Tecla 0
vbKeyNumpad1	97	Tecla 1
vbKeyNumpad2	98	Tecla 2
vbKeyNumpad3	99	Tecla 3
vbKeyNumpad4	100	Tecla 4
vbKeyNumpad5	101	Tecla 5
vbKeyNumpad6	102	Tecla 6
vbKeyNumpad7	103	Tecla 7
vbKeyNumpad8	104	Tecla 8
vbKeyNumpad9	105	Tecla 9
vbKeyMultiply	106	Tecla "*"
vbKeyAdd	107	Tecla "+"
vbKeySeparator	108	Tecla "Intro"
vbKeySubtract	109	Tecla "-"
vbKeyDecimal	110	Tecla "."
vbKeyDivide	111	Tecla "/"

Teclas de funciones

Constante	Valor	Descripción
vbKeyF1	112	Tecla F1
vbKeyF2	113	Tecla F2
vbKeyF3	114	Tecla F3
vbKeyF4	115	Tecla F4
vbKeyF5	116	Tecla F5
vbKeyF6	117	Tecla F6
vbKeyF7	118	Tecla F7
vbKeyF8	119	Tecla F8
vbKeyF9	120	Tecla F9
vbKeyF10	121	Tecla F10
vbKeyF11	122	Tecla F11
vbKeyF12	123	Tecla F12
vbKeyF13	124	Tecla F13 (de teclado Mac)
vbKeyF14	125	Tecla F14 (de teclado Mac)
vbKeyF15	126	Tecla F15 (de teclado Mac)
vbKeyF16	127	Tecla F16 (de teclado Mac)

Otras teclas

Constante	Valor	Descripción
vbKeyLButton	1	Botón izquierdo del ratón
vbKeyRButton	2	Botón derecho del ratón
vbKeyCancel	3	Tecla Cancelar
vbKeyMButton	4	Botón central del ratón
vbKeyBack	8	Tecla Retorno
vbKeyTab	9	Tecla Tab
vbKeyClear	12	Tecla Eliminar
vbKeyReturn	13	Tecla Intro
vbKeyShift	16	Tecla Shift
vbKeyControl	17	Tecla Ctrl
vbKeyMenu	18	Tecla Menú
vbKeyPause	19	Tecla Pausa
vbKeyCapital	20	Tecla Bloq Mayús
vbKeyEscape	27	Tecla Esc
vbKeySpace	32	Tecla Espacio
vbKeyPageUp	33	Tecla RePág
vbKeyPageDown	34	Tecla AvPág
vbKeyEnd	35	Tecla Fin

vbKeyHome	36	Tecla Inicio
vbKeyLeft	37	Tecla Flecha izquierda
vbKeyUp	38	Tecla Flecha arriba
vbKeyRight	39	Tecla Flecha derecha
vbKeyDown	40	Tecla Flecha abajo
vbKeySelect	41	Tecla Selección
vbKeyPrint	42	Tecla ImpPant
vbKeyExecute	43	Tecla Ejecutar
vbKeySnapshot	44	Tecla Snapshot
vbKeyInsert	45	Tecla Insert
vbKeyDelete	46	Tecla Supr
vbKeyHelp	47	Tecla Help
vbKeyNumLock	144	Tecla Bloq Num

2. Constantes de Access - constantes ac

a. AcCloseSave - constantes de copia de seguridad durante el cierre

Constante	Valor	Descripción
acCloseNo	2	No guarda nada durante el cierre.
acSavePrompt	0	Pregunta al usuario si quiere o no guardar las modificaciones.
acSaveYes	1	Guarda automáticamente durante el cierre.

Ejemplo

```
Sub Salir_Libro_Activo()
    DoCmd.Close acForm, Screen.ActiveForm.Name, acSaveYes
End Sub
```

b. AcColorIndex - constantes de colores

Constante	Valor	Descripción
acColorIndexAqua	14	Verde agua
acColorIndexBlack	0	Negro
acColorIndexBlue	12	Azul
acColorIndexBrightGreen	10	Verde claro
acColorIndexDarkBlue	4	Azul oscuro
acColorIndexFuschia	13	Magenta
acColorIndexGray	7	Gris
acColorIndexGreen	2	Verde

acColorIndexMaroon	1	Rojo oscuro
acColorIndexOlive	3	Marrón claro
acColorIndexRed	9	Rojo
acColorIndexSilver	8	Gris claro
acColorIndexTeal	6	Azul verdoso
acColorIndexViolet	5	Violeta
acColorIndexWhite	15	Blanco
acColorIndexYellow	11	Amarillo

c. AcCommand - constantes de comandos

La ejecución de estos comandos puede hacerse con:

```
Application.RunCommand Constante
RunCommand Constante
Docmd.RunCommand Constante
```

Pestaña Archivo

Constante	Valor	Descripción
acCmdCompactDatabase	4	Información - Compactar y reparar
acCmdEncryptDecrypt Database	5	Seguridad - Cifrar/descifrar una base de datos...
acCmdSetDatabasePassword	275	Seguridad - Definir la contraseña de la base de datos...
acCmdNewDatabase	26	Nuevo - Nueva base de datos...
acCmdOpenDatabase	25	Abrir...
acCmdSave	20	Guardar
acCmdSaveAs	21	Guardar como...
acCmdPrint	340	Imprimir...
acCmdClose	58	Salir
acCmdOptions	49	Opciones...
acCmdPageSetup	32	Configuración...
acCmdPrintPreview	54	Vista previa antes de la impresión
acCmdDatabaseProperties	256	Propiedades de la base
acCmdExit	3	Salir

Pestaña Inicio

Constante	Valor	Descripción
acCmdRefresh	18	Actualizar
acCmdLineUpIcons	213	Alinear los iconos

acCmdViewCode	170	Código
acCmdViewDetails	210	Detalles
acCmdViewLargeIcons	209	Grandes iconos
acCmdViewList	212	Lista
acCmdViewForms	112	Objetos de la base - Formularios
acCmdViewMacros	114	Objetos de la base - Macros
acCmdViewMódulos	115	Objetos de la base - Módulos
acCmdViewReports	113	Objetos de la base - Informes
acCmdViewQueries	111	Objetos de la base - Consultas
acCmdViewTables	110	Objetos de la base - Tablas
acCmdPreviewOnePage	246	Páginas - Una página
acCmdPreviewTwoPages	247	Páginas - Dos páginas
acCmdPreviewFourPages	248	Páginas - Cuatro páginas
acCmdPreviewEightPages	249	Páginas - Ocho páginas
acCmdPreviewTwelvePages	250	Páginas - Doce páginas
acCmdViewSmallIcons	211	Iconos pequeños
acCmdProperties	287	Propiedades
acCmdArrangeIconsBy Created	216	Reorganizar los iconos - Por fecha de creación
acCmdArrangeIconsBy Modified	217	Reorganizar los iconos - Por fecha de modificación
acCmdArrangeIconsByName	214	Reorganizar los iconos - Por nombre
acCmdArrangeIconsByType	215	Reorganizar los iconos - Por tipo
acCmdZoom10	244	Zoom - 10 %
acCmdZoom25	243	Zoom - 25 %
acCmdZoom50	242	Zoom - 50 %
acCmdZoom75	241	Zoom - 75 %
acCmdZoom100	240	Zoom - 100 %
acCmdZoom150	239	Zoom - 150 %
acCmdZoom200	238	Zoom - 200 %
acCmdZoom500	481	Zoom - 500 %
acCmdFitToWindow	245	Zoom - Anchura de la página
acCmdZoom1000	482	Zoom - Máximo 1000 %

Pestaña Crear

Constante	Valor	Descripción
acCmdNewObjectReport	137	informe
acCmdNewObjectAutoReport	194	informe instantáneo
acCmdNewObjectForm	136	Formulario
acCmdNewObjectAutoForm	193	Formulario instantáneo
acCmdNewObjectMacro	138	Macro

acCmdNewObjectModule	139	Módulo
acCmdNewObjectClassModule	140	Módulo de clase
acCmdNewObjectQuery	135	Consulta
acCmdNewObjectTable	134	Tabla

Pestaña Datos externos

Constante	Valor	Descripción
acCmdeImport	257	Datos externos - Importar...
acCmdLinkTables	102	Datos externos - Relacionar tablas...
acCmdExport	487	Exportar...

Pestaña Herramientas de base de datos

Constante	Valor	Descripción
acCmdRunMacro	31	Macro - Ejecutar una macro...
acCmdDocumentar	285	Análisis - Documentación
acCmdAnalyzePerformance	283	Análisis - Rendimiento
acCmdAnalyzeTable	284	Análisis - Tabla
acCmdRelationships	133	Relaciones

Barra de sistema

Constante	Valor	Descripción
acCmdAppMaximize	10	Maximizar
acCmdAppMove	12	Mover
acCmdExit	3	Salir
acCmdAppMinimize	11	Minimizar
acCmdAppRestore	9	Restaurar
acCmdAppSize	13	Tamaño

Pestaña del objeto actual

Constante	Valor	Descripción
acCmdDocMaximize	15	Maximizar
acCmdDocMove	16	Mover
acCmdClose	58	Salir
acCmdDocMinimize	60	Minimizar
acCmdDocRestore	14	Restaurar
acCmdDocSize	17	Tamaño

Menu Ayuda

Constante	Valor	Descripción
acCmdAnswerWizard	235	Ayuda sobre Microsoft Access
acCmdAboutMicrosoft Access	35	Sobre Microsoft Access
acCmdMicrosoftOnTheWeb	236	Microsoft Office en la Web

Pestaña Inicio/Tabla

Constante	Valor	Descripción
acCmdUndo	292	Anular campo/registro actual (entrada)
acCmdRecordsGoToLast	68	Ir a - Último
acCmdRecordsGoToNew	28	Ir a - Nuevo registro
acCmdRecordsGoTo Previous	66	Ir a - Anterior
acCmdRecordsGoToFirst	67	Ir a - Primero
acCmdRecordsGoToNext	65	Ir a - Siguiente
acCmdPasteSpecial	64	Pegado especial...
acCmdPaste	191	Pegar
acCmdPasteAppend	38	Pegar y añadir
acCmdCopy	190	Copiar
acCmdCut	189	Cortar
acCmdFind	30	Buscar...
acCmdReplace	29	Sustituir...
acCmdRename	143	Renombrar
acCmdSelectRecord	50	Seleccionar guardar
acCmdSelectAllRecords	109	Seleccionar todos los registros
acCmdDelete	337	Eliminar
acCmdDeleteRecord	223	Eliminar guardar

Grupo Registros

Constante	Valor	Descripción
acCmdRefresh	18	Actualizar
acCmdRemoveFilterSort	144	Ver todos los registros
acCmdApplyFilterSort	93	Aplicar filtro/ordenación
acCmdAdvancedFilterSort	99	Filtrar - Filtro/ordenación avanzado...
acCmdFilterExcluding Selection	277	Filtrar - Filtrar fuera de la selección
acCmdFilterByForm	207	Filtrar - Filtrar mediante formulario
acCmdFilterBySelection	208	Filtrar - Filtrar mediante selección

acCmdDataEntry	78	Rellenar datos
acCmdSaveRecord	97	Guardar copia de seguridad
acCmdSortAscending	163	Ordenar - Orden creciente
acCmdSortDescending	164	Ordenar - Orden decreciente

Menú Ventana

Constante	Valor	Descripción
acCmdWindowUnhide	1	Ver...
acCmdSizeToFitForm	69	Ajustar al tamaño del formulario
acCmdWindowCascade	22	Cascada
acCmdWindowHide	2	Ocultar
acCmdTileHorizontally	286	Mosaico horizontal
acCmdTileVertically	23	Mosaico vertical
acCmdWindowArrangeIcons	24	Reorganizar iconos

Grupo Formato

Constante	Valor	Descripción
acCmdUnhideColumns	80	Ver las columnas...
acCmdDatasheetView	282	Hoja de datos...
acCmdFreezeColumn	105	Fijar las columnas
acCmdRowHeight	116	Altura de fila...
acCmdColumnWidth	117	Anchura de columna...
acCmdUnfreezeAllColumns	106	Liberar todas las columnas
acCmdHideColumns	79	Ocultar las columnas
acCmdFont	19	Fuente de letra...

Otros comandos

Constante	Valor	Descripción
acCmdRegister ActiveXControls	254	Controles ActiveX...
acCmdStartupProperties	224	Iniciar...
acCmdOutputToExcel	175	Enlaces Office - Análisis con Microsoft Excel
acCmdWordMailMerge	195	Enlaces Office - Fusión con Microsoft Word
acCmdOutputToRTF	176	Enlaces Office - Publicación con Microsoft Word
acCmdConvertMacrosTo VisualBasic	279	Macro - Convertir las macros en Visual Basic
acCmdCreateShortcut MenuFromMacro	336	Macro - Crear un menú contextual a partir de una macro
acCmdCreateMenu FromMacro	334	Macro - Crear un menú a partir de una macro

acCmdAutoCorrect	261	Opciones de corrección automática...
acCmdSpelling	269	Ortografía...
acCmdToolbarsCustomize	165	Personalizar...
acCmdCreateReplica	263	Replicación - Crear una réplica...
acCmdRecoverDesign Master	265	Replicación - Recuperar un documento de creación maestro...
acCmdResolveConflicts	266	Replicación - Resolver conflictos...
acCmdSynchronizeNow	264	Replicación - Sincronizar ahora...
acCmdUserLevel SecurityWizard	276	Seguridad - Asistente Seguridad a nivel del usuario...
acCmdConvertDatabase	171	Herramientas... - Convertir una base de datos
acCmdMakeMDEFile	7	Herramientas... - Crear un archivo MDE...
acCmdDatabaseSplitter	520	Herramientas... - Fraccionar una base de datos

d. AcControlType - constantes de tipos de controles

Constante	Valor	Descripción
acAttachment	126	Control de elemento adjunto
acBoundObjectFrame	108	Control Marco de objeto dependiente
acCheckBox	106	Control Casilla de selección
acComboBox	111	Control Zona de lista desplegable
acCommandButton	104	Control Botón de comando
acCustomControl	119	Control ActiveX
acEmptyCell	127	Control EmptyCell
acImage	103	Control Imagen
acLabel	100	Control Etiqueta
acLine	102	Control Trazo
acListBox	110	Control Zona de lista
acNavigationButton	130	Control Botón de navegación
acNavigationControl	129	Control de navegación
acObjectFrame	114	Control Marco de objeto independiente
acOptionButton	105	Control Botón de opción
acOptionGroup	107	Control Grupo de opciones
acPage	124	Control de página
acPageBreak	118	Control Salto de página
acRectangle	101	Control Rectángulo
acSubform	112	Control de subformulario
acTabCtl	123	Control Pestaña
acTextBox	109	Control Zona de texto
acToggleButton	122	Control Botón de activación
acWebBrowser	128	Control de navegador web

e. AcCurrentView - constantes de vistas actuales

Constante	Valor	Descripción
acCurViewDatasheet	2	El objeto se muestra en modo Hoja de datos.
acCurViewDesign	0	El objeto se muestra en modo Creación.
acCurViewFormBrowse	1	El objeto se muestra en modo Formulario.
acCurViewLayout	7	El objeto se muestra en modo Página.
acCurViewPivotChart	4	El objeto se muestra en modo Gráfico dinámico.
acCurViewPivotTable	3	El objeto se muestra en modo Tabla dinámica.
acCurViewPreview	5	El objeto se muestra en modo Vista previa de impresión.
acCurView ReportBrowse	6	El objeto se muestra en modo Informe.

f. AcDataObjectType - constantes de tipos de objetos de Access

Constante	Valor	Descripción
acActiveDataObject	-1	El objeto activo contiene registros.
acDataForm	2	Un formulario contiene el registro.
acDataFunction	10	Una función definida por el usuario contiene registros (proyecto de Microsoft Access únicamente).
acDataQuery	1	Una consulta contiene registros.
acDataReport	3	Un informe contiene registros.
acDataServerView	7	Una vista de servidor contiene registros (proyecto de Microsoft Access únicamente).
acDataStoredProcedure	9	Un procedimiento almacenado contiene registros (proyecto de Microsoft Access únicamente).
acDataTable	0	Una tabla contiene registros.

g. acExportXMLObjectType - constantes de los tipos de objetos para exportar

Constante	Valor	Descripción
acExportForm	2	Formulario
acExportFunction	10	Función definida por el usuario (proyecto de Microsoft Access únicamente).
acExportQuery	1	Consulta
acExportReport	3	Informe
acExportServerView	7	Vista servidor (proyecto de Microsoft Access únicamente).
acExportStored Procedure	9	Procedimiento almacenado (proyecto de Microsoft Access únicamente).
acExportTable	0	Tabla

h. acExportXMLOtherFlags - constantes posexportación XML

Constante	Valor	Descripción
acEmbedSchema	1	Escribe la información del esquema en el documento especificado por el argumento DataTarget; este valor tiene prioridad sobre el argumento SchemaTarget.
acExcludePrimary KeyAndIndexes	2	No exporta las propiedades de la clave primaria ni del esquema del índice.
acExportAllTable AndFieldProperties	32	El esquema exportado contiene las propiedades de la tabla y de sus campos.
acLiveReportSource	8	Crea un enlace activo hacia una base de datos Microsoft SQL Server 2000 remota. Solo vale cuando exporta los informes relacionados con una base de datos de Microsoft SQL Server 2000.
acPersistReportML	16	Conserva la información ReportML del objeto exportado.
acRunFromServer	4	Crea un wrapper ASP en lugar del wrapper HTML por defecto. Solo se aplica cuando exporta los informes.

i. AcFileFormat - constantes de formatos de archivo de Microsoft Access

Constante	Valor	Descripción
acFileFormatAccess12	12	Formato Microsoft Access 2007 (por tanto también en 2019)
acFileFormatAccess2	2	Formato Microsoft Access 2.0
acFileFormatAccess2000	9	Formato Microsoft Access 2000
acFileFormatAccess2002	10	Formato Microsoft Access 2002
acFileFormatAccess95	7	Formato Microsoft Access 95
acFileFormatAccess97	8	Formato Microsoft Access 97

j. AcObjectType - constantes de tipos de objetos

Constante	Valor	Descripción
acDatabaseProperties	11	Propiedades de base de datos
acDefault	-1	
acDiagram	8	Esquema de base de datos (proyecto de Microsoft Access)
acForm	2	Formulario
acFunction	10	Función
acMacro	4	Macro
acModule	5	Módulo
acQuery	1	Consulta
acReport	3	Informe
acServerView	7	Ver los servidores

acStoredProcedure	9	Procedimiento almacenado (proyecto de Microsoft Access)
acTable	0	Tabla
acTableDataMacro	12	Macro de datos

k. AcProperty - constantes de propiedades

Constante	Valor	Descripción
acPropertyBackColor	8	Permite definir la propiedad BackColor.
acPropertyCaption	9	Permite definir la propiedad Caption.
acPropertyEnabled	0	Permite definir la propiedad Enabled.
acPropertyForeColor	7	Permite definir la propiedad ForeColor.
acPropertyHeight	6	Permite definir la propiedad Height.
acPropertyLeft	3	Permite definir la propiedad Left.
acPropertyLocked	2	Permite definir la propiedad Locked.
acPropertyTop	4	Permite definir la propiedad Top.
acPropertyValue	10	Permite definir la propiedad Value.
acPropertyVisible	1	Permite definir la propiedad Visible.
acPropertyWidth	5	Permite definir la propiedad Width.

l. AcWindowMode - constantes de visualización de ventana

Constante	Valor	Descripción
acDialog	3	Abre el formulario o informe en modo Pop-up y modal.
acHidden	1	Abre el formulario o informe en modo cacheado.
acIcon	2	Abre el formulario o informe minimizado en la barra de tareas de Windows.
acWindowNormal	0	Abre el formulario o informe en el modo definido por sus propiedades (Valor por defecto).

3. Constantes DAO - constantes db

a. CollatingOrderEnum - constantes de Collating Order

Constante	Valor	Descripción
dbSortArabic	1025	Collating Order árabe
dbSortChineseSimplified	2052	Collating Order chino simplificado
dbSortChinese Traditional	1028	Collating Order chino tradicional
dbSortCyrillic	1049	Collating Order ruso
dbSortCzech	1029	Collating Order checo
dbSortDutch	1043	Collating Order belga

dbSortGeneral	1033	Collating Order inglés, alemán, francés y portugués
dbSortGreek	1032	Collating Order griego
dbSortHebrew	1037	Collating Order hebreo
dbSortHungarian	1038	Collating Order húngaro
dbSortIcelandic	1039	Collating Order islandés
dbSortJapanese	1041	Collating Order japonés
dbSortKorean	1042	Collating Order coreano
dbSortNeutral	1024	Collating Order neutro
dbSortNorwdan	1030	Collating Order nórdico y danés
dbSortPDXIntl	1033	Collating Order paradox internacional
dbSortPDXNor	1030	Collating Order noruego y danés paradox
dbSortPDXSwe	1053	Collating Order sueco y finés paradox
dbSortPolish	1045	Collating Order polaco
dbSortSlovenian	1060	Collating Order esloveno
dbSortSpanish	1034	Collating Order español
dbSortSwedFin	1053	Collating Order sueco y finés
dbSortThai	1054	Collating Order thai
dbSortTurkish	1055	Collating Order turco
dbSortUndefined	-1	Collating Order no definido o desconocido

b. DataTypeEnum - constantes de tipos de datos

Constante	Valor	Descripción
dbAttachment	101	Tipo de dato Elemento adjunto
dbBigInt	16	Tipo de dato Entero muy grande
dbBinary	9	Tipo de datos Binario
dbBoolean	1	Tipo de dato booleano (Verdadero/falso)
dbByte	2	Tipo de dato byte (8 bits)
dbChar	18	Tipo de dato Texto (anchura fija)
dbComplexByte	102	Tipo de dato de valores múltiples Byte
dbComplexDecimal	108	Tipo de dato de valores múltiples Decimal
dbComplexDouble	106	Tipo de dato de valores múltiples Coma flotante doble precisión
dbComplexGUID	107	Tipo de dato de valores múltiples GUID
dbComplexInteger	103	Tipo de dato de valores múltiples Entero
dbComplexLong	104	Tipo de dato de valores múltiples Entero long
dbComplexSingle	105	Tipo de dato de valores múltiples Coma flotante simple precisión
dbComplexText	109	Tipo de dato de valores múltiples de Texto (anchura variable)
dbCurrency	5	Tipo de dato Moneda

dbDate	8	Tipo de dato Fecha
dbDecimal	20	Tipo de dato Decimal (ODBCDirect únicamente)
dbDouble	7	Tipo de dato Coma flotante doble precisión
dbFloat	21	Tipo de dato Coma flotante (ODBCDirect únicamente)
dbGUID	15	Tipo de dato GUID
dbInteger	3	Tipo de dato Entero
dbLong	4	Tipo de dato Entero long
dbLongBinary	11	Tipo de dato Binario (imagen bitmap)
dbMemo	12	Tipo de dato Memo (texto extendido)
dbNumeric	19	Tipo de dato Numérico (ODBCDirect únicamente)
dbSingle	6	Tipo de dato Coma flotante precisión simple
dbText	10	Tipo de dato Texto (anchura variable)
dbTime	22	Datos al formato horario (ODBCDirect únicamente)
dbTimeStamp	23	Datos al formato fecha y hora (ODBCDirect únicamente)
dbVarBinary	17	Tipo de dato Binario de longitud variable (ODBCDirect únicamente)

c. LockTypeEnum - constantes de modos de bloqueo

Constante	Valor	Descripción
dbOptimistic	3	Acceso concurrente optimista basado en el ID de registro. El cursor compara los ID del registro antiguo y el nuevo para determinar si hay modificaciones desde el último acceso al registro.
dbOptimisticBatch	5	Permite las actualizaciones por lotes optimistas (espacios de trabajo ODBCDirect únicamente).
dbOptimisticValue	1	Acceso concurrente optimista basado en los valores de registro. El cursor compara los valores de los datos en el registro antiguo y el nuevo para determinar si se han añadido modificaciones desde el último acceso al registro (espacios de trabajo ODBCDirect únicamente).
dbPessimistic	2	Acceso concurrente pesimista. El cursor utiliza el nivel de bloqueo más bajo permitiendo garantiza la actualización del registro.

d. QueryDefTypeEnum - constantes de tipos de consultas

Constante	Valor	Descripción
dbQAction	240	Consulta Acción
dbQAppend	64	Consulta Adición
dbQCompound	160	No documentada
dbQCrosstab	16	Consulta de análisis cruzado

dbQDDL	96	Consulta Definición de datos
dbQDelete	32	Consulta Eliminación
dbQMakeTable	80	Consulta Creación de tabla
dbQProcedure	224	Consulta de procedimiento almacenado
dbQSelect	0	Consulta Selección
dbQSetOperation	128	Consulta Unión
dbQSPTBulk	144	Consulta que no devuelve ningún registro
dbQSQLPassThrough	112	Consulta SQL Directa
dbQUpdate	48	Consulta Actualización

e. RecordsetTypeEnum - constantes de tipos de conjunto de registros

Constante	Valor	Descripción
dbOpenDynamic	16	Abre un conjunto de registros de tipo hoja de respuesta dinámica (ODBC Direct).
dbOpenDynaset	2	Abre un conjunto de registros de tipo hoja de respuesta dinámica.
dbOpenForwardOnly	8	Abre un conjunto de registros de tipo transferencia únicamente.
dbOpenSnapshot	4	Abre un conjunto de registros de tipo captura instantánea.
dbOpenTable	1	Abre un conjunto de registros de tipo tabla.

4. Constantes ADO - constantes ad

a. DataTypeEnum - constantes de tipos de datos

Constante	Valor	Descripción
adArray	0x2000	Un valor de indicador, siempre combinado con otra constante de tipo de dato, que indica una tabla de este otro tipo de dato.
adBigInt	20	Indica un número entero firmado de 8 bytes (DBTYPE_I8).
adBinary	128	Indica un valor binario (DBTYPE_BYTES).
adBoolean	11	Indica un valor de tipo booleano (DBTYPE_BOOL).
adBSTR	8	Indica una cadena de caracteres que termina con un carácter Null (Unicode) (DBTYPE_BSTR).
adChapter	136	Indica un valor de capítulo de 4 bytes que identifica las líneas en un juego de líneas hijo (DBTYPE_HCHAPTER).
adChar	129	Indica un valor de cadena (DBTYPE_STR).
adCurrency	6	Indica un valor moneda (DBTYPE_CY). Se trata de un número en coma fija de 4 cifras a la derecha de la coma decimal. Se almacena en un número entero firmado de 8 bytes en una escala de 10 000.
adDate	7	Indica un valor de fecha (DBTYPE_DATE). Una fecha

		se almacena como un número doble: la parte entera tiene el número de días desde el 30 de diciembre de 1899, la parte decimal representa la fracción de un día.
adDBDate	133	Indica un valor de fecha (aaaammdd (DBTYPE_DBDATE).
adDBTime	134	Indica un valor de hora (hhmmss) (DBTYPE_DBTIME).
adDBTimeStamp	135	Indica un valor fecha y hora completo (aaaammddhhmmss, más una fracción en milisegundos) (DBTYPE_DBTIMESTAMP).
adDecimal	14	Indica un valor numérico exacto con una precisión y una escala fijas (DBTYPE_DECIMAL).
adDouble	5	Indica un valor en coma flotante de doble precisión (DBTYPE_R8).
adEmpty	0	No especifica ningún valor (DBTYPE_EMPTY).
adError	10	Indica un código de error de 32 bits (DBTYPE_ERROR).
adFileTime	64	Indica un valor de 64 bits, que representa el número de intervalos de 100 nanosegundos, desde el 1 de enero de 1601 (DBTYPE_FILETIME).
adGUID	72	Indica un identificador universal único (GUID) (DBTYPE_GUID).
adIDispatch	9	Indica un puntero a una interfaz IDispatch en un objeto COM (DBTYPE_IDISPATCH).
adInteger	3	Indica un número entero firmado de 4 bytes (DBTYPE_I4).
adIUnknown	13	Indica un puntero a una interfaz IUnknown en un objeto COM (DBTYPE_IUNKNOWN).
adLongVarBinary	205	Indica un valor binario largo.
adLongVarChar	201	Indica un valor de cadena larga.
adLongVarWChar	203	Indica un valor de cadena larga terminada por un carácter Null (Unicode).
adNumeric	131	Indica un valor numérico exacto, con una precisión y una escala fijas (DBTYPE_NUMERIC).
adPropVariant	138	Indica un PROPVARIANT de automatización (DBTYPE_PROP_VARIANT).
adSingle	4	Indica un valor en coma flotante en precisión simple (DBTYPE_R4).
adSmallInt	2	Indica un número entero firmado de 2 bytes (DBTYPE_I2).
adTinyInt	16	Indica un número entero firmado de 1 byte (DBTYPE_I1).
adUnsigned-BigInt	21	Indica un número entero no firmado de 8 bytes (DBTYPE_UI8).
adUnsignedInt	19	Indica un número entero no firmado de 8 bytes (DBTYPE_UI4).
adUnsigned-SmallInt	18	Indica un número entero no firmado de 2 bytes (DBTYPE_UI2).
adUnsigned-TinyInt	17	Indica un número entero no firmado de 1 byte (DBTYPE_UI1).
adUserDefined	132	Indica una variable definida por el usuario (DBTYPE_UDT).

adVarBinary	204	Indica un valor binario (objeto Parameter únicamente).
adVarChar	200	Indica un valor de cadena.
adVariant	12	Indica un objeto Variant de automatización (DBTYPE_VARIANT).
adVarNumeric	139	Indica un valor numérico (objeto Parameter únicamente).
adVarWChar	202	Indica una cadena de caracteres terminada por un carácter Null (Unicode).
adWChar	130	Indica una cadena de caracteres terminada por un carácter Null (Unicode) (DBTYPE_WSTR).

b. LockTypeEnum - constantes de modos de bloqueo

Constante	Valor	Descripción
adLockBatchOptimistic	4	Indica las actualizaciones por lotes optimistas. Obligatorio para el modo de actualización por lotes.
adLockOptimistic	3	Indica un bloqueo optimista, registro por registro. El proveedor aplica este tipo de bloqueo cuando llama al método Update.
adLockPessimistic	2	Indica un bloqueo pesimista, registro por registro. El proveedor debe garantizar que la edición de los registros tiene éxito, en general bloqueando los registros a la fuente de datos inmediatamente después de la edición.
adLockReadOnly	1	Indica los registros en modo solo lectura. No puede modificar los datos.
adLockUnspecified	-1	No especifica el tipo de bloqueo. Se crea un clon con el mismo tipo de bloqueo que el original.

5. Constantes de Microsoft - constantes mso

a. msoAutomatizationSecurity - constantes de seguridad durante la apertura automática

Constante	Valor	Descripción
msoAutomation SecurityByUI	2	Utiliza el argumento de seguridad especificado en el cuadro de diálogo **Seguridad**.
msoAutomation	3	Desactiva todas las macros en todos los archivos abiertos por programación, sin que se muestre ninguna alerta de seguridad.
msoAutomation SecurityLow	1	Activa todas las macros. Corresponde al valor por defecto al inicio de la aplicación.

Lista de errores

A continuación se muestra la lista de los principales errores que pueden tener lugar en VBA.

Número de error	Descripción
3	Retorno sin GoSub
5	Argumento o llamada de procedimiento incorrecto
6	Se sobrepasa la capacidad
7	Memoria insuficiente
9	El índice no pertenece a la selección
10	Esta tabla es fija o está temporalmente bloqueada
11	División por cero
13	Incompatibilidad de tipos
14	Espacio de cadena insuficiente
16	Expresión demasiado compleja
17	Imposible realizar la operación necesaria
18	Interrupción por el usuario
20	Recuperación sin error
28	Espacio de pila insuficiente
35	Sub o Function no definida
47	Demasiados clientes de aplicación para la DLL
48	Error de carga de la DLL
49	Convención de llamada de DLL incorrecta
51	Error interno
52	Nombre o número de archivo incorrecto
53	Archivo no se encuentra
54	Modo de acceso al archivo incorrecto
55	Archivo ya abierto
57	Error de entrada/salida de periférico
58	Este archivo ya existe
59	Longitud de registro incorrecta
61	Disco lleno
62	La entrada sobrepasa el fin del archivo
63	Número de registros incorrectos
67	Demasiados archivos
68	Periférico no disponible
70	Permisos denegados
71	Disco no preparado
74	Imposible renombrar con un lector diferente
75	Error de acceso Ruta/Archivo
76	Ruta de acceso no encontrada

91	Variable de objeto o variable de bloque With no definida
92	Bucle For no inicializado
93	Formato de cadena incorrecto
94	Uso incorrecto de Null
96	Imposible recibir eventos del objeto; este genera ya el número máximo de eventos gestionados a los destinatarios
97	Imposible llamar a un procedimiento Friend para un objeto que no es una instancia de la clase de definición
98	Una llamada a una propiedad o a un método no se puede utilizar referencia a un objeto privado, aunque sea como argumento o como valor de retorno
321	Formato de archivo incorrecto
322	Imposible crear el archivo temporal necesario
325	Formato incorrecto en el archivo de recurso
380	Valor de propiedad incorrecta
381	Índice de la tabla de propiedades incorrecto
382	Set no gestionado en la ejecución
383	Set no gestionado (propiedad en modo solo lectura)
385	Índice de tabla de propiedad requerida
387	Set no permitido
393	Get no gestionado en la ejecución
394	Get no gestionado (propiedad en modo solo escritura)
422	Propiedad no encontrada
423	Propiedad o método encontrado
424	Objeto requerido
429	Un componente ActiveX no puede crear objetos
430	La clase no gestiona Automatización o la interfaz esperada
432	Nombre del archivo o de la clase no encontrado durante la operación de automatización
438	Propiedad o método no gestionado por este objeto
440	Error Automatization
442	La conexión a la librería de tipos o de objetos para el tratamiento remoto se ha perdido. Pulse en **OK** para eliminar la referencia en el cuadro de diálogo.
443	El objeto Automatization no tiene valor por defecto
445	Este objeto no gestiona esta acción
446	Este objeto no gestiona los argumentos nombrados
447	Este objeto no gestiona los argumentos regionales actuales
448	Argumento llamado no encontrado
449	Argumento no opcional
450	Número de argumentos incorrecto o asignación de propiedad incorrecta
451	El procedimiento Property Let no está definido y el procedimiento Property Get no ha devuelto ningún objeto
452	Número incorrecto
453	Función de DLL especificada no encontrada

454	Recurso de código no encontrado
455	Error de bloqueo del recurso de code
457	Esta clave ya está asociada a un elemento de esta colección
458	Esta variable utiliza un tipo Automatization no gestionado por Visual Basic
459	El objeto o la clase no gestiona el juego de eventos
460	Formato de portapapeles incorrecto
461	Miembro de método o de datos no encontrado
462	El servidor remoto no existe o no está disponible
463	La clase no se ha guardado en la máquina local
481	Imagen incorrecta
482	Error de impresora
735	Imposible guardar el archivo en TEMP
744	El texto buscado no se ha encontrado
746	Sustituciones demasiado largas

Accesos directos de teclado

Tecla Ctrl

Acceso directo	Descripción
[Ctrl] **C**	Copia un texto, imagen o archivo seleccionado.
[Ctrl] **F**	Abre el cuadro de diálogo **Buscar y Reemplazar**.
[Ctrl] **H**	Abre el cuadro de diálogo **Reemplazar**.
[Ctrl] **N**	Crea una nueva base de datos.
[Ctrl] **O**	Abre el cuadro de diálogo **Abrir** (para abrir una base de datos existente).
[Ctrl] **P**	Imprime el objeto seleccionado automáticamente.
[Ctrl] **S**	Guarda el proyecto de base de datos.
[Ctrl] **V**	Pega lo que ha sido copiado.
[Ctrl] **W**	Cierra la ventana activa.
[Ctrl] **X**	Corta lo que está seleccionado.
[Ctrl] **Z**	Anula la/las última(s) acciones realizadas.
[Ctrl][Intro]	Inserta un retorno de carro en un campo de tipo memo o texto (Enter).
[Ctrl][Flecha a la derecha]	Mueve el control seleccionado a la derecha.
[Ctrl][Flecha a la izquierda]	Mueve el control seleccionado a la izquierda.
[Ctrl][Flecha arriba]	Mueve el control seleccionado hacia arriba.
[Ctrl][Flecha abajo]	Mueve el control seleccionado hacia abajo.
[Ctrl][F2]	Llama a un generador.
[Ctrl] [F4]	Cierra la ventana activa.
[Ctrl] [F6]	Recorre las ventanas abiertas.
[Ctrl] [F8]	Activa el modo Redimensionar de la ventana activa cuando no está maximizada.
[Ctrl][F11]	Cambia entre una barra de menús personalizada y una barra de menús integrada.
[Ctrl] **;**	Inserta la fecha del día actual.
[Ctrl][Shift] **,**	Inserta los datos del mismo campo en el registro anterior.
[Ctrl][Shift] **;**	Inserta la hora actual.

Tecla Shift

Acceso directo	Descripción
[Shift][Intro]	Para añadir un control a una sección.
[Shift][F1]	Para visualizar los tooltips.
[Shift][F2]	Para abrir la zona **Zoom** para facilitar la entrada de expresiones y otros textos.
[Shift][F4]	Se utiliza para encontrar la siguiente ocurrencia del texto especificado en el cuadro de diálogo **Buscar y Reemplazar** cuando se cierra el cuadro.

[Shift][F10]	Para visualizar el menú contextual.
[Shift][F12]	Para guardar un objeto de base de datos.
[Shift][Flecha hacia abajo]	Para aumentar la altura del control seleccionado.
[Shift][Flecha a la derecha]	Para aumentar la anchura del control seleccionado.
[Shift][Flecha hacia arriba]	Para reducir la altura del control seleccionado.
[Shift][Flecha a la izquierda]	Para reducir la anchura del control seleccionado.

Otros accesos directos

Acceso directo	Descripción
[Esc]	Anular las modificaciones añadidas al campo actual.
[Esc][Esc] (2 veces)	Anular las modificaciones añadidas al registro actual.
[Intro]	Restablece el tamaño de la ventana reducida seleccionada cuando todas las ventanas se reducen.
[F1]	Ver la ayuda del asistente de Office y de Microsoft Access.
[F2] (selección)	Ver la dirección completa del enlace de hipertexto seleccionado.
[F2] (creación)	Cambiar entre el modo Modificación y el modo Movimiento.
[F4]	Abrir una zona de lista modificable.
[F5]	Cambiar del modo Creación de formulario al modo Formulario.
[F6]	Cambiar entre las partes inferior y superior de una ventana.
[F7]	Verificar la ortografía.
[F9]	Actualizar el contenido de un campo Lista de opciones.
[F11]	Llevar la ventana **Base de datos** a primer plano.
[F12]	Abrir el cuadro de diálogo **Guardar en**.
[Tab]	Salir de la zona de lista modificable o de la zona de lista.

Recursosinformáticos

VBA Access (versiones 2019 y Office 365)

Programar en Access

Este libro sobre **VBA Access (versión 2019 y Office 365)**, se dirige tanto a las personas con conocimientos de Access, como a los **desarrolladores principiantes** o **aquellos con más experiencia**. Cada uno de ellos encontrará la información necesaria para transformar sus herramientas «caseras», en una aplicación robusta o para **personalizar y optimizar de manera gráfica** las soluciones existentes. Conocer el funcionamiento de Access y utilizarlo de manera habitual, es un requisito previo imprescindible para obtener el máximo beneficio de este libro.

En este libro se tratan tanto los aspectos **básicos de VBA, como el uso de APIs externos**. El autor cubre los diferentes aspectos de la programación en Visual Basic. Este desarrollo, que tiene una dificultad progresiva, permite a los lectores reconocer la necesidad de usar este lenguaje, para entregar soluciones potentes y eficaces. Después de la **sintaxis básica en VBA**, el autor trata la noción de **programación orientada a objetos**, para más adelante, seguir con los modelos **de acceso a los datos DAO y ADO**, el **lenguaje SQL** aplicado a Access, los **eventos Access**, las **interfaces** de usuario y cómo optimizarlas con la **cinta de opciones de Access**, el control de otras aplicaciones Office con la **automatización**, las **interacciones con Internet** o la **programación con Windows**. Al final del libro, se ofrece una mini-aplicación.

Las bases de datos de ejemplo para implementar los diferentes aspectos que se tratan en el libro, están disponibles para su descarga en el sitio web de Ediciones ENI www.editions-eni.com.

Jean-Philippe ANDRÉ ha sido desarrollador y consultor senior de las herramientas de la suite de Microsoft durante muchos años. Jean-Philippe ANDRÉ interviene en la actualidad en el apoyo a los desarrolladores en Quebec, especializado en tecnologías Microsoft. Ha sido profesor durante 10 años en las escuelas de ingenieros y en la universidad, y en este libro transmite toda su experiencia técnica y pedagógica, para permitir al lector dominar la potencia del lenguaje VBA, aplicado a las soluciones de gestión de bases de datos Access 2019.

En **www.ediciones-eni.com**:
➜ Las bases de datos de ejemplo.

Para más información:

ISBN: 978-2-409-02357-6

www.ediciones-eni.com

El Mundo de la Programación en tus Manos...!

DETODOPROGRAMACION.ORG

DETODOPROGRAMACION.ORG

Material para los amantes de la Programación Java, C/C++/C#,Visual.Net, SQL, Python, Javascript, Oracle, Algoritmos, CSS, Desarrollo Web, Joomla, jquery, Ajax y Mucho Mas…

VISITA

www.detodoprogramacion.org
www.detodopython.com
www.gratiscodigo.com

GRATISCODIGO.COM
Usa, lo que ya se creó...

Detodo Python.com