

A. Formulario de gestión de ventas: descripción del ejemplo

1. Presentación del ejemplo

Somos la empresa **BolEni**, un distribuidor que vende bolsas deportivas en dos tallas, L y XL. La empresa comienza su actividad y desea disponer de un archivo simple para hacer un seguimiento de sus ventas y su *stock*.

La herramienta que va a configurar permitirá al vendedor de esta pequeña empresa crear, almacenar y descargar facturas.

La herramienta se presentará como un formulario para ser rellenado por el vendedor. Estará accesible desde un archivo Excel y el botón **Ir a la herramienta de gestión de ventas**, que se colocará en la hoja de apertura del libro.

y conviértase en un experto en Excel (versiones 2019, 2021 y Microsoft 365)

He aquí la herramienta finalizada:

Software de gestión de ventas

Cantidad: 5 Producto: Bolsa tamaño X

Añadir a la línea de pedido

4*Bolsa tamaño L=80€
5*Bolsa tamaño XL=150€

Eliminar las líneas seleccionadas

Total sin IVA 230 € Descuento 10 % pro

Total con descuento 230 € IVA 46 €

Total con IVA 276 €

Guardar e imprimir el pedido

2. Presentación del archivo

Para realizar este ejemplo, utilizará el archivo de Excel **Enunciado_3-ABC.xlsm**. El formato **XLSM** significa que es un archivo de Excel que admite macros (lo que permite el uso de VBA, a menudo deshabilitado de forma predeterminada).

Este archivo contiene tres hojas de Excel (denominadas *sheets* en Visual Basic).

Hoja Inicio

Esta hoja está diseñada para contener solo el botón **Ir a la herramienta de administración de ventas** que facilita el acceso a la herramienta de gestión de ventas.

Hoja Productos

La hoja **Productos** enumera los productos vendidos por la empresa. Contiene tres columnas:

	Columna A	Columna B	Columna C
Fila 1	Nombre del producto	Precio sin IVA	Cantidad disponible
Fila 2	Bolsa tamaño L	20 €	90
Fila 3	Bolso tamaño XL	30 €	50

Hoja Facturas

Esta hoja contendrá las facturas creadas con la herramienta de gestión de ventas. Estarán referenciadas por número y también encontrará la fecha/hora de edición, así como el importe.

	Columna A	Columna B	Columna C
Fila 1	Número de factura	Fecha y hora	Importe de la factura

3. Funciones

El objetivo es disponer de una herramienta que permita gestionar las ventas de las bolsas de la empresa. He aquí la transcripción de las necesidades en forma de requisitos.

Requisitos operativos

Los requisitos operativos corresponden a la descripción de las funcionalidades generales de la aplicación; este es el nivel más bajo de detalle:

- ▶ Crear una factura
- ▶ Hacer el seguimiento de la factura
- ▶ Actualizar el *stock* de productos

Traducir estos requisitos operativos en funciones

Se trata de detallar los requisitos operativos como funciones que corresponden a las acciones del usuario y los procesos del sistema. Esta lista debe ser exhaustiva para realizar estas funciones en forma de aplicación.

- ▶ Agregar una fila de pedido.
 - ▶ Elegir la cantidad.
 - ▶ Elegir el producto.
 - ▶ Validar la fila de pedido.
- ▶ Ver el pedido.
- ▶ Eliminar una fila de pedido.

- ▶ Calcular el total y permitir la aplicación de un 10 % de descuento.
- ▶ Actualizar los *stocks*.

B. Formulario de gestión de ventas: conceptos del curso

Este ejemplo contiene muchos conceptos nuevos relacionados con la programación en lenguaje Visual Basic. Se trata, por lo tanto, de algunos puntos de referencia útiles para poder comenzar con tranquilidad.

1. Concepto de programación

He aquí una descripción simplificada de algunos conceptos básicos de programación que permiten un mejor enfoque de los ejemplos propuestos.

Objeto y clase

Un **objeto** es una entidad informática; puede tener cualquier forma y cada objeto es único. Se caracteriza según su tipo.

La **clase** corresponde a la definición del objeto; servirá como lienzo para la creación de objetos nuevos. Esto significa que todos los objetos de la misma clase tendrán las mismas propiedades y se diferenciarán por los valores de sus propiedades.

Ejemplo:

Clase	Objetos
Cell (celda en una hoja de Excel)	<ul style="list-style-type: none"> ▶ Cells("A1"): Celda A1 de la hoja actual; ▶ Cells("C4"): Celda C4 de la hoja actual.
Sheet (hoja de libro)	<ul style="list-style-type: none"> ▶ Sheets(0): primera hoja del libro actual; ▶ Sheets(1): segunda hoja del libro actual.
Textbox (cuadro de texto que se puede introducir en un formulario)	<ul style="list-style-type: none"> ▶ Textbox1: cuadro de texto que se puede introducir denominado Textbox1 por el usuario; ▶ Textbox2: cuadro de texto que se puede introducir denominado Textbox2 por el usuario.

Propiedades

Una **propiedad** es un atributo de una clase. Cuando se crea un objeto, tiene valores asignados a sus propiedades.

Ejemplo: la celda de una hoja tiene muchas propiedades, como el valor: `Cells("A1").value`

Métodos

Un **método** corresponde a una acción que puede realizar un objeto. Por ejemplo, el objeto hoja de cálculo (**Sheets**) proporciona un método **Add** para agregar una hoja.

Ejemplo: `Sheets.Add`

Colecciones

Una **colección** es una lista de objetos de la misma clase. Por ejemplo, la colección `sheets` corresponde al conjunto de hojas. En Visual Basic, las colecciones son objetos completos, con sus propios métodos y propiedades.

Variables

Una **variable** es una entidad informática que almacena información dentro de la aplicación. Se declara de la siguiente manera:

- ▶ `Dim`: permite definir la variable (`Public` para una variable pública);
- ▶ `Nombre_variable`: da un nombre a la variable;
- ▶ `As TypeVariable`: permite dar un tipo a la variable.

Ejemplo:

```
Dim MiVariable As String
```

Esto significa que la variable `MiVariable` se ha declarado como cadena de caracteres.

Las variables son las siguientes:

- ▶ **Públicas**: son accesibles en toda la aplicación. Se declaran fuera de cualquier procedimiento de código.
- ▶ **Privadas**: son accesibles solo en el procedimiento donde se declaran (en un procedimiento determinado).

Las variables se tipan principalmente por las tres razones siguientes:

- ▶ Para tener métodos (ver con anterioridad) adaptados a la variable: una adición de cadenas corresponde a la concatenación, mientras que una adición de números corresponde a la suma de los valores:

Operaciones	Valor de la variable cadena de caracteres	Valor de la variable número entero
<code>MiVar = "A" + "E"</code>	"AE"	Error
<code>MiVar = 1 + 2</code>	"12"	3

- ▶ Para facilitar el desarrollo y el uso de la variable; se espera el contenido de la variable.
- ▶ Cada tipo de variable tiene una cantidad asignada de memoria; por lo tanto, usar el tipo correcto de variable ahorra memoria.

He aquí los tipos de variables y los detalles de cada una:

Nombre	Tipo	Detalles
Byte	Numérico	Entero de 0 a 255
Integer	Numérico	Entero de -32'768 a 32'767
Long	Numérico	Entero de - 2'147'483'648 a 2'147'483'647
Currency	Numérico	Número decimal fijo de -922'337'203'685'477.5808 a 922'337'203'685'477.5807
Single	Numérico	Número en coma flotante de -3.402823E38 a 3.402823E38
Double	Numérico	Número en coma flotante de -1.79769313486232D308 a 1.79769313486232D308
String	Texto	Texto
Date	Fecha	Fecha y hora
Boolean	Booleano	True (verdadero) o False (falso)
Object	Objeto	Objeto de Microsoft (por ejemplo: celda, rango de celdas u hoja)
Variant	Todos	Valor por defecto si no se declara

2. Concepto de formulario

Formulario

Un **formulario** es una ventana de interacción entre el usuario y el sistema. Es una interfaz visual para reproducir o recopilar información con fines de procesamiento.

El objeto formulario es el contenedor de los otros objetos visuales. Esto significa que incorpora otros objetos visuales, también propuestos en la caja de herramientas.

Una misma aplicación puede contener varios formularios.

Los controles

Un formulario es un contenedor de controles. Estos **controles** son objetos visuales que permiten la interacción con el usuario.

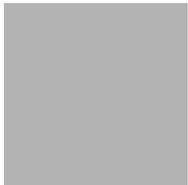
De forma predeterminada, se proponen unos quince controles en la caja de herramientas; para los objetivos de este ejemplo, se detallarán siete de ellos. Sin embargo, es posible importar controles nuevos, pero se trata de un uso más avanzado de la aplicación VBA.

Un control se crea cuando se coloca en un formulario. Un formulario puede contener varios controles del mismo tipo porque los controles son objetos únicos. La propiedad *Name* (nombre) de cada control debe ser única dentro del mismo formulario. En contraposición, es posible encontrar un control con el mismo valor para la propiedad *Name* en otro formulario:

```
' Un objeto Control1 colocado en el Formulario1
Formulario1.Control1
' Un objeto Control1 colocado en el Formulario2
Formulario2.Control1
```

He aquí algunos tipos de controles y su uso:

Nombre del objeto	Icono en la caja de herramientas	Visualización en un formulario	Uso
CommandButton			Botón de formulario que normalmente desencadena acciones.
Label			Cuadro de texto no editable por el usuario. Puede ser editado por el sistema.
TextBox			Cuadro de texto donde puede escribir el usuario.
ComboBox			Lista desplegable con la posibilidad de elegir un solo valor.
OptionButton			Botón de opción para seleccionar un valor en un grupo: normalmente, no se usa solo (ejemplo: hombre o mujer).
CheckBox			Casilla de verificación para seleccionar uno o más valores en un grupo (ejemplo: selección de aficiones).

Nombre del objeto	Icono en la caja de herramientas	Visualización en un formulario	Uso
ListBox			Lista con la posibilidad de ver todos los elementos que la componen, a diferencia de un control ComboBox, donde solo se muestra un valor. Además, esta lista permite seleccionar varios valores (se puede configurar en los atributos del objeto).

3. Escribir código

Esta subparte le enseñará las instrucciones básicas para la codificación. ¿Dónde, cuándo y cómo escribir código?

Módulo

Un **módulo** es una hoja donde es posible escribir código. Los módulos pueden contener varios procedimientos.

Procedimiento

Un **procedimiento** representa un fragmento de código que cuenta con un título. Se puede llamar a un procedimiento en cualquier momento dentro de la aplicación mediante la instrucción `Call`.

Puede tener parámetros de entrada llamados **argumentos**, que pueden ser opcionales.

Un procedimiento comienza por la instrucción `Sub` con el nombre del procedimiento y termina con la instrucción `End Sub`. Por defecto, el ámbito del procedimiento es público, lo que significa que es visible (y, por tanto, es posible llamarlo) en cualquier parte de la aplicación. No obstante, puede especificarse el alcance del procedimiento escribiendo `Public` o `Private` (privado) antes de la instrucción `Sub`. La instrucción `Private Sub` limitará la visibilidad del procedimiento al módulo donde se declara.

He aquí un ejemplo de procedimiento que muestra el producto de un cálculo dentro de un módulo.

```
Public Sub MiProcedimiento()
    'Definición de mis variables A y B como números enteros
    Dim A As Integer
    Dim B As Integer
    'Asignación de valores a las variables
    A = 3
```