VBA EXCEL MINI-GUIA

- ## Mapear columnas de una tabla a rangos y usar los nombres en bucle de tuplas

```
Public Sub mapear_columnas()
    '1º declaramos las cabeceras de las nuevas columnas que creamos
    Range("D1").Value = "Nueva_col_1"

    '2º declaramos y asignamos los rangos por columnas
    Dim rngTabla As Range
    Dim rngCol_1 As Range
    Dim rngCol_2 As Range

    'La tabla principal debe ser "compacta"/"tidy" para que el CurrentRegion funcione
    Set rngTabla = Range("A1").CurrentRegion
    Set rngCol_1 = Range("B:B")
    Set rngCol_2 = Range("C:C")
    Set rngNuevaCol_1 = Range("D:D")

    'Bucle para iterar sobre cada una de las tuplas de la tabla y
    'Rescatar el valor de un campo en concreto segun el rango (con nombre)
    'De esta forma si una columna cambia de lugar solo tenemos que cambiar el range
    Dim fila As Range
    For Each fila In rngTabla.Columns("A:A").Offset(1, 0).Cells 'saltamos las cabeceras
        valor = Cells(fila.row, rngCol_1.Column)
        If valor = "X" Then
            Debug.Print "Row number: " & fila.row
            Debug.Print "rngCol_1 value: " & Cells(fila.row, rngCol_1.Column).Value
        End If
    Next fila
End Sub
```

- ## Ocultar columnas enteras en una sola línea

```
Public Sub ocultarRango()
    '1º Hacemos uniones de rangos
    Dim rangoOculto As Range
    Set rangoOculto = Union(Range("A:A"), Range("B:B"), Range("C:C"))
    'Hay que hacerle un .EntireColumn al Union para poder ocultar la columna entera
    rangoOculto.EntireColumn.Hidden = True
End Sub
```

- ## Filtrado de datos con Autofilter

```
Public Sub filtrarDatos()
    'Filtrado basico de datos que se puede sofisticar cuanto se necesite
    'Si esto no es necesario, bucleamos por todas las tuplas estableciendo las condiciones en una ultima columna
    'Le ponemos un testigo y filtramos segun el valor de dicha ultima columna con el autofilter

    'Establecemos la tabla y los nombres de rango de las columnas para tenerlo todo mapeado
    Dim rngTabla, rngCol_1, rngCol_2, rngCol_3 As Range

    Set rngTabla = Range("A1").CurrentRegion
    Set rngCol_1 = Range("B:B")
    Set rngCol_2 = Range("C:C")
    Set rngCol_3 = Range("D:D")

    'Desactivamos cualquier otro filtro que haya sobre la tabla (necesario)
    'Y activamos el autofilter
    ActiveSheet.AutoFilterMode = False
    rngTabla.AutoFilter

    FilterArray = Array("A", "B", "C", "D")

    With rngTabla
        .AutoFilter field:=rngCol_2.Column, Criteria1:="0" 'Literal
        .AutoFilter field:=rngCol_1.Column, Criteria1:="<1", Operator:=xlOr, Criteria2:=">20" 'Condicional dentro de la misma columna
        .AutoFilter field:=rngCol_3.Column, Criteria1:=FilterArray, Operator:=xlFilterValues 'Este ultimo es importante al hacerlo sobre arrays
        'Todo lo que va aqui dentro se aplica a la vez como cuando aplicamos varios filtros sobre distintas columnas
        'El anterior filtraria todo lo que en la columna de las C's sea igual a 0
        '   y en la columna de las B's sea menor que 1 o mayor que 20
        '   y en la columna de las D's los valores sean cualquiera de los del array indicado anteriormente
    End With
End Sub
```

- ## Asignar Hotkey a un proceso de forma global en el Excel

```
Public Sub AsignarHotkey()
    'Esta funcion habria que crearla en el "ThisWorkbook" en el evento "Workbook_open" para que sea automatico
    'Util si tenemos por ejemplo un form con distintas opciones y le asignamos una hotkey para tener algo tipo menu personal

    'En el Workbook_Open
    Call ejemplos_vba.CrearHotkey

    'En el Workbook_BeforeClose
    Call ejemplos_vba.BorrarHotkey

End Sub
```

```
Public Sub CrearHotkey()
    'Ver ayuda para todas las posibles teclas raras  https://docs.microsoft.com/es-es/office/vba/api/excel.application.onkey
    Application.OnKey "^%{p}", "ProcesoHotkey" 'Lanzar al pulsar : Ctrl + Alt + P    podria ser un form.show   y entre comillas!
End Sub
```

```
Public Sub BorrarHotkey()
    Application.OnKey "^%{p}" 'De-asignar
End Sub
```

```
Public Sub ProcesoHotkey()
    Debug.Print "Hello"
    'lo que sea
End Sub
```

- ## Generar Diccionario (Scripting.Dictionary) – Microsoft Scripting Runtime

```
Public Function getDiccionario()
    'Veremos como declararlo, alimentarlo, y un par de funciones basicas
    'Antes de nada activamos en Herramientas->Referencias
    'La referencia a "Microsoft Scripting Runtime"

    'Declaramos e inicializamos el diccionario
    Dim diccionario As New Scripting.Dictionary 'El new es importante

    'La tabla/dataset en el que nos basamos para alimentar el diccionario
    Dim rngTabla, rngCol_key, rngCol_value As Range
    Set rngTabla = Range("A1").CurrentRegion
    Set rngCol_key = Range("A:A")
    Set rngCol_value = Range("D:D")

    Dim fila As Range
    For Each fila In rngTabla.Offset(1, 0).Resize(rngTabla.Rows.Count - 1, rngTabla.Columns.Count).Columns("A:A").Cells 'Tabla sin cabeceras
        If Not diccionario.Exists(Cells(fila.row, rngCol_key.Column).Value) Then
            diccionario.Add Cells(fila.row, rngCol_key.Column).Value, Cells(fila.row, rngCol_value.Column).Value
        End If 'Si ya existe no lo actualizamos el valor
        '---Si quisieramos crear o actualizar si ya existe el key utilizariamos
        '---diccionario(key) = value  en vez de la condicional esta
    Next fila
    Set getDiccionario = diccionario
End Function
```

- ## Funciones para diccionarios

```
Public Sub funcionesDiccionario()
    'Es util si lanzamos un proceso que accede mucho a database
    'Para no sobrecargarla y meter a un diccionario todos los valores que vamos accediendo
    'O si cada paso del bucle requiere mucho calculo para evitar repetirlo

    Dim x As Scripting.Dictionary
    Set x = getDiccionario()

    'Bucle recorrer keys y valores
    For Each ID In x.Keys
        Debug.Print ID & " - " & x(ID)
    Next ID

    mykey = Val("485")
    'Saber si un valor existe y en caso afirmativo devolver valor
    If x.Exists(mykey) Then
        Debug.Print (x(mykey))
    End If

    'Borrar un par key/value
    If x.Exists(mykey) Then
        x.Remove mykey
    End If

    'Limpiar diccionario
    x.RemoveAll

End Sub
```

- **Acceso a Database Access desde VBA Excel para copiarse una tabla o generar un diccionario según valores de una tabla – Microsoft ActiveX Data Objects 6.1 Library**

```
Public Sub CopiarTablaAccesEnHojaExcel(nombreHoja As String, nombreTabla As String)
    'Se copia la tabla indicada en una nueva hoja excel con el nombre indicado
    'Hay que activar Referencia -> Microsoft ActiveX Data Objects 6.1 Library
    'Esto se podria mejorar metiendo un metodo para acceder a las tablas y pintarlos en un form
    'Incluyendo los campos que queremos rescatar en vez de hacerlo todo a saco


    Application.ScreenUpdating = False

    On Error GoTo Handle:
    Dim conn As New ADODB.Connection
    Dim rs As New ADODB.Recordset
    Dim rutaDb As String
    Dim sSQL As String
    Dim tabla As String
    Dim hojaResultados As Worksheet

    'rutaDb = "C:\adrian\db.accdb" o lo que sea
    'De esta forma indicamos que es en el mismo directorio que el archivo libro
    rutaDb = Application.ActiveWorkbook.Path & "\db.accdb"
    Debug.Print rutaDb
    tabla = nombreTabla
    sSQL = "Select * FROM " & tabla

    conn.Open "Provider=Microsoft.ACE.OLEDB.12.0; Data Source=" & rutaDb & ";"
    rs.Open sSQL, conn

    Set hojaResultados = ActiveWorkbook.Sheets.Add(, Worksheets(Worksheets.Count)) 'Añadir hoja al final del libro
    If Not sheetNameExists(nombreHoja) Then
        Sheets(hojaResultados.Index).Name = nombreHoja
        'En caso contrario lo dejamos como esta
    End If

    hojaResultados.Activate
    'Pintamos cabeceras de la tabla
    For iCol = 1 To rs.Fields.Count
        Cells(1, iCol).Value = rs.Fields(iCol - 1).Name
    Next iCol


    'Recorrerse el recordset hasta el final
    fila = 2
    Do While Not rs.EOF
        For i = 0 To rs.Fields.Count - 1
            Cells(fila, i + 1).Value = rs.Fields.Item(i)
        Next i
        fila = fila + 1
        rs.MoveNext
    Loop

    'Cerrar conexiones
    If Not (rs Is Nothing) Then
        If (rs.State And adStateOpen) = adStateOpen Then
            rs.Close
        End If
        Set rs = Nothing
    End If

    If Not (conn Is Nothing) Then
        If (conn.State And adStateOpen) = adStateOpen Then
            conn.Close
        End If
        Set conn = Nothing
    End If
    Application.ScreenUpdating = True
Exit Sub

Handle:
    Application.ScreenUpdating = True
    On Error Resume Next
    If Not (rs Is Nothing) Then
        If (rs.State And adStateOpen) = adStateOpen Then
            rs.Close
        End If
        Set rs = Nothing
    End If

    If Not (conn Is Nothing) Then
        If (conn.State And adStateOpen) = adStateOpen Then
            conn.Close
        End If
        Set conn = Nothing
    End If
    MsgBox "ERROR EN SUB -CopiarTablaAccesEnHojaExcel-" & vbCrLf & "Algo falla con el access" & vbCrLf & "Abortamos proceso...",
```

```vba
Private Function sheetNameExists(nameSheet As String) As Boolean
    'Comprueba si ya existe una hoja excel con un nombre dado
    'No es elegante pero funciona
    Dim sh As Worksheet
    On Error Resume Next
    Set sh = ThisWorkbook.Sheets(nameSheet)
    On Error GoTo 0
    sheetNameExists = Not sh Is Nothing
End Function
```