Sheet_Ppal - 1

```
'********** Hoja Ppal (Eventos) **********
Option Explicit

Private Sub Combo1_Change()
    HandleCombo1Change
End Sub

Private Sub Combo2_Change()
    HandleCombo2Change
End Sub

Private Sub Combo3_Change()
    HandleCombo3Change
End Sub

Private Sub Combo4_Change()
    HandleCombo4Change
End Sub

Private Sub Combo5_Change()
    HandleCombo5Change
End Sub

Private Sub Combo6_Change()
    HandleCombo6Change
End Sub

Private Sub TextBox7_Change()
    HandleTextBox7Change
End Sub

Private Sub TextBox9_Change()
    HandleTextBox9Change
End Sub

Private Sub TextBox10_Change()
    HandleTextBox10Change
End Sub

Private Sub btn1_Click()
    CargarDatosCombo2
End Sub

Private Sub btn2_Click()
    CargarDatosCombo3
End Sub

Private Sub btn3_Click()
    InitializeForm
    MsgBox "El formulario ha sido actualizado correctamente.", vbInformation
End Sub

Private Sub btn4_Click()
    PrepararReciboParaImpresion
End Sub

Private Sub btn5_Click()
    EliminarSeleccionados
End Sub

Private Sub btn6_Click()
    ConvertirMontoDivisa
End Sub

' KeyPress
Private Sub TextBox7_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
    HandleKeyPress Me.OLEObjects("TextBox7"), KeyAscii
End Sub

Private Sub TextBox9_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
    HandleKeyPress Me.OLEObjects("TextBox9"), KeyAscii
End Sub
```

Sheet_Ppal - 2

```
Private Sub TextBox10_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
    HandleKeyPress Me.OLEObjects("TextBox10"), KeyAscii
End Sub

' LostFocus y formateos
Private Sub TextBox5_LostFocus()
    FormatearDecimalesTextBox "TextBox5", 3
    UpdatePaymentState
End Sub

Private Sub TextBox6_LostFocus()
    FormatearDecimalesTextBox "TextBox6", 2
    UpdatePaymentState
End Sub

Private Sub TextBox7_LostFocus()
    If IsNumeric(Me.OLEObjects("TextBox7").Object.value) Then
        Me.OLEObjects("TextBox7").Object.value = Format(Me.OLEObjects("TextBox7").Object.value, "0.00"
)
    Else
        Me.OLEObjects("TextBox7").Object.value = ""
    End If
    FormatearDecimalesTextBox "TextBox7", 2
    UpdatePaymentState
End Sub

Private Sub TextBox9_LostFocus()
    If IsNumeric(Me.OLEObjects("TextBox9").Object.value) Then
        Me.OLEObjects("TextBox9").Object.value = Format(Me.OLEObjects("TextBox9").Object.value, "0.00"
)
    Else
        Me.OLEObjects("TextBox9").Object.value = ""
    End If
    FormatearDecimalesTextBox "TextBox9", 2
    UpdatePaymentState
End Sub

Private Sub TextBox10_LostFocus()
    If IsNumeric(Me.OLEObjects("TextBox10").Object.value) Then
        Me.OLEObjects("TextBox10").Object.value = Format(Me.OLEObjects("TextBox10").Object.value, "0.0
0")
    Else
        Me.OLEObjects("TextBox10").Object.value = ""
    End If
    FormatearDecimalesTextBox "TextBox10", 2
    UpdatePaymentState
End Sub

Private Sub TextBox12_LostFocus()
    FormatearDecimalesTextBox "TextBox12", 2
    UpdatePaymentState
End Sub
```

ThisWorkbook - 1

```
'********** ThisWorkbook **********
Option Explicit

Private Sub Workbook_Open()
    On Error GoTo ErrorHandler

    frmLogin.Show

    If usuarioAutenticado = "" Then
        MsgBox "Debe autenticarse para usar la aplicación.", vbExclamation
        ThisWorkbook.Close SaveChanges:=False
        Exit Sub
    End If

    InitializeForm
    MsgBox "El formulario ha sido actualizado correctamente.", vbInformation

    Exit Sub

ErrorHandler:
    LogError "Workbook_Open", Err.description
    MsgBox "Error en Workbook_Open: " & Err.description, vbCritical
End Sub

Private Sub Workbook_BeforeClose(Cancel As Boolean)
    ' Código opcional antes de cerrar el libro
End Sub
```

frmLogin - 1

```
'********** frmLogin **********
Option Explicit

Private Sub btnAceptar_Click()
    If txtUsuario.Text = "" Or txtContrasena.Text = "" Then
        MsgBox "Debe ingresar el usuario y la contraseña.", vbExclamation
        Exit Sub
    End If

    Application.ScreenUpdating = False
    If ValidarUsuario(txtUsuario.Text, txtContrasena.Text) Then
        usuarioAutenticado = txtUsuario.Text
        Me.Hide
    Else
        MsgBox "Usuario o contraseña incorrectos.", vbCritical
        txtContrasena.Text = ""
        txtContrasena.SetFocus
    End If
    Application.ScreenUpdating = True
End Sub

Private Sub btnCancelar_Click()
    Me.Hide
    Application.ScreenUpdating = True
End Sub
```

frmPrintOptions - 1

```
'************* frmPrintOptions ******************
Option Explicit

Public opcionImpresion As String
Public impresoraSeleccionada As String
Public numeroCopias As Integer

Private Sub btnAceptar_Click()
    ' Validar número de copias
    If Not IsNumeric(txtCopias.Text) Or Val(txtCopias.Text) < 1 Then
        MsgBox "Por favor, ingrese un número válido de copias.", vbExclamation
        Exit Sub
    End If

    ' Almacenar opciones seleccionadas
    If optImprimirDirecto.value Then
        opcionImpresion = "ImprimirDirecto"
    ElseIf optPrevisualizar.value Then
        opcionImpresion = "Previsualizar"
    ElseIf optExportarPDF.value Then
        opcionImpresion = "ExportarPDF"
    End If

    impresoraSeleccionada = cmbImpresoras.Text
    numeroCopias = Val(txtCopias.Text)

    ' Cerrar el formulario y retornar al código que lo llamó
    Me.Hide
End Sub

Private Sub btnCancelar_Click()
    ' Cerrar el formulario sin hacer nada
    Me.Hide
End Sub

Private Sub UserForm_Initialize()
    ' Cargar impresoras disponibles en el ComboBox
    Dim impresoraActual As String
    impresoraActual = Application.ActivePrinter
    Dim impresoras() As String
    impresoras = ObtenerListaImpresoras()

    Dim i As Integer
    For i = LBound(impresoras) To UBound(impresoras)
        cmbImpresoras.AddItem impresoras(i)
    Next i

    ' Seleccionar la impresora actual
    cmbImpresoras.Text = impresoraActual

    ' Establecer número de copias predeterminado
    txtCopias.Text = "1"

    ' Establecer opción predeterminada
    optImprimirDirecto.value = True
End Sub

Private Function ObtenerListaImpresoras() As String()
    Dim objNetwork As Object
    Set objNetwork = CreateObject("WScript.Network")

    Dim colPrinters As Object
    Set colPrinters = objNetwork.EnumPrinterConnections

    Dim printers() As String
    Dim i As Integer
    Dim count As Integer
    count = colPrinters.count / 2 - 1
    ReDim printers(0 To count)

    Dim index As Integer
    index = 0
    For i = 0 To colPrinters.count - 1 Step 2
```

```
        printers(index) = colPrinters.Item(i + 1)
        index = index + 1
    Next i

    ObtenerListaImpresoras = printers
End Function
```

Module_Config - 1

```
'********** Module_Config **********
Option Explicit

Public ConfigData As Object

Public Sub LoadConfigData()
    Set ConfigData = CreateObject("Scripting.Dictionary")

    Dim configFilePath As String
    configFilePath = ThisWorkbook.Path & "\Config v.1.0.xlsm"

    If Dir(configFilePath) = "" Then
        MsgBox "No se encontró el archivo de configuración en la ruta: " & configFilePath, vbCritical
        Exit Sub
    End If

    Dim wbConfig As Workbook
    Dim wsConfig As Worksheet
    Dim tblConfig As ListObject
    Dim clave As String
    Dim valor As String
    Dim row As ListRow

    Set wbConfig = Workbooks.Open(Filename:=configFilePath, ReadOnly:=True)
    wbConfig.Windows(1).Visible = False

    Set wsConfig = wbConfig.Sheets("Rutas")
    Set tblConfig = wsConfig.ListObjects("Config")

    If tblConfig Is Nothing Then
        MsgBox "No se encontró la tabla 'Config' en la hoja 'Rutas'.", vbCritical
        wbConfig.Close SaveChanges:=False
        Exit Sub
    End If

    For Each row In tblConfig.ListRows
        clave = CStr(row.Range.Cells(1, tblConfig.ListColumns("Ref").index).value)
        valor = CStr(row.Range.Cells(1, tblConfig.ListColumns("Filename").index).value)
        ConfigData.Add clave, valor
    Next row

    wbConfig.Close SaveChanges:=False
End Sub

Public Function GetFilePath(referenceName As String) As String
    If ConfigData Is Nothing Then
        LoadConfigData
    End If

    If ConfigData.Exists(referenceName) Then
        GetFilePath = ConfigData(referenceName)
    Else
        MsgBox "No se encontró la referencia '" & referenceName & "' en el archivo de configuración.",
 vbCritical
        GetFilePath = ""
    End If
End Function
```

Module_GlobalVariables - 1

```
'********** Module_GlobalVariables **********
Option Explicit

Public usuarioAutenticado As String
Public opcionImpresion As String
Public impresoraSeleccionada As String
Public numeroCopias As Integer
Public isInitializing As Boolean
Public previousClient As String

' Diccionario para configuración
Public ConfigData As Object

' Colores personalizados (Valores Numéricos)
Public Const colorOk As Long = 5287936           ' Verde claro (RGB(80, 200, 120))
Public Const colorNegado As Long = 5197823       ' Rojo (RGB(255, 79, 79))
Public Const colorBlanco As Long = 16777215      ' Blanco (RGB(255, 255, 255))
Public Const colorEditable As Long = 15987699    ' Amarillo claro (RGB(255, 255, 183))
Public Const colorPropuesta As Long = 14277081   ' Gris claro (RGB(217, 217, 217))
Public Const colorNegro As Long = 0              ' Negro (RGB(0, 0, 0))
Public Const ESTADO_OK As String = "Ok"
Public Const ESTADO_NEGADO As String = "Negado"
Public Const ESTADO_VACIO As String = ""

' Variables de control
Public isHandlingTextBox7Change As Boolean
Public isHandlingCombo4Change As Boolean
Public isHandlingTextBox9Change As Boolean
Public isHandlingTextBox10Change As Boolean
Public isUpdatingTextBox9 As Boolean
```

```
Module_Initialize - 1

'********** Module_Initialize **********
Option Explicit

Public Sub InitializeForm()
    On Error GoTo ErrorHandler

    isInitializing = True
    previousClient = ""

    Dim wsPpal As Worksheet
    Set wsPpal = ThisWorkbook.Sheets("Ppal")

    wsPpal.OLEObjects("TextBox1").Object.value = Format(Date, "dddd, dd \de mmmm \de yyyy")

    ClearControls wsPpal

    Application.ScreenUpdating = False

    ' Cargar config
    LoadConfigData

    ' Cargar ComboBoxes
    LoadCombo1
    LoadCombo2
    LoadCombo4
    LoadCombo6

    ' Inicializar el Bloque de Pago
    InitializePaymentBlock

    Application.ScreenUpdating = True
    isInitializing = False
    Exit Sub

ErrorHandler:
    LogError "InitializeForm", Err.description
    MsgBox "Error en InitializeForm: " & Err.description, vbCritical
    Application.ScreenUpdating = True
    isInitializing = False
End Sub

Private Sub ClearControls(ByVal ws As Worksheet)
    Dim controlNames As Variant
    controlNames = Array("Combo1", "Combo2", "Combo3", "Combo4", "Combo5", "Combo6", _
                         "TextBox2", "TextBox3", "TextBox4", "TextBox5", "TextBox6", _
                         "TextBox7", "TextBox8", "TextBox9", "TextBox10", "TextBox11", "TextBox12")

    Dim ctrlName As Variant
    For Each ctrlName In controlNames
        With ws.OLEObjects(ctrlName).Object
            .value = ""
            .BackColor = colorEditable
            .Locked = False
        End With
    Next ctrlName

    ws.OLEObjects("TextBox1").Object.Locked = True
    ws.OLEObjects("TextBox2").Object.Locked = True
    ws.OLEObjects("TextBox3").Object.Locked = True
    ws.OLEObjects("TextBox6").Object.Locked = True
    ws.OLEObjects("TextBox8").Object.Locked = True
    ws.OLEObjects("TextBox11").Object.Locked = True

    ' Limpiar nuevos rangos
    ws.Range("AR11:AR70").ClearContents
    ws.Range("AS11:BU70").ClearContents
    ws.Range("CE11:CE70").ClearContents

    ws.OLEObjects("btn4").Object.Enabled = False
End Sub
```

Module_Main - 1

```
'********** Module_Main **********
Option Explicit

Private isHandlingTextBox7Change As Boolean
Private isHandlingCombo4Change As Boolean
Private isHandlingTextBox9Change As Boolean
Private isHandlingTextBox10Change As Boolean

' Procedimiento para cargar Combo1 (Lista de Clientes)
Public Sub LoadCombo1()
    On Error GoTo ErrorHandler

    Dim filePath As String
    Dim wbClte As Workbook
    Dim wsClte As Worksheet
    Dim arrClients() As String
    Dim i As Long
    Dim lastRow As Long

    ' Obtener la ruta del archivo Clte
    filePath = GetFilePath("ClteFilePath")
    If filePath = "" Then Exit Sub

    ' Abrir el libro Clte en modo lectura y oculto
    Set wbClte = Workbooks.Open(Filename:=filePath, ReadOnly:=True)
    wbClte.Windows(1).Visible = False ' Ocultar el libro

    ' Establecer la hoja Cltes
    Set wsClte = wbClte.Sheets("Cltes")

    ' Obtener los datos de los clientes
    arrClients = ObtenerListaClientes(wsClte)

    ' Asignar al ComboBox1
    With ThisWorkbook.Sheets("Ppal").OLEObjects("Combo1").Object
        .Clear
        .List = arrClients
        .MatchEntry = fmMatchEntryComplete
    End With

    ' Cerrar el libro Clte
    wbClte.Close SaveChanges:=False
    Exit Sub

ErrorHandler:
    LogError "LoadCombo1", Err.description
    If Not wbClte Is Nothing Then wbClte.Close SaveChanges:=False
End Sub


' Cargar Combo2 (Propósitos de Uso)
Public Sub LoadCombo2()
    On Error GoTo ErrorHandler

    Dim filePath As String
    filePath = GetFilePath("ParamFilePath")
    If filePath = "" Then Exit Sub

    ' Aquí implementar la lógica para cargar Combo2 según la hoja Param y tabla CRD, etc.
    ' Dependiendo del código original que hayas tenido, repetiré la lógica básica:

    Dim wbParam As Workbook
    Dim wsParam As Worksheet
    Dim lastRow As Long
    Dim arrPurposes() As String
    Dim i As Long

    Set wbParam = Workbooks.Open(Filename:=filePath, ReadOnly:=True)
    wbParam.Windows(1).Visible = False

    ' Asumiendo que la info de Propósitos se encuentra en la hoja "Param" en alguna columna
    Set wsParam = wbParam.Sheets("Param")
    ' Buscar y cargar la lista CRD
```

```
Module_Main - 2

    Dim IDCrd_col As Variant, CRD_col As Variant
    IDCrd_col = Application.Match("IDCrd", wsParam.Rows(1), 0)
    CRD_col = Application.Match("CRD", wsParam.Rows(1), 0)

    If IsError(IDCrd_col) Or IsError(CRD_col) Then
        MsgBox "No se encontraron las columnas 'IDCrd' o 'CRD' en la hoja 'Param'.", vbCritical
        wbParam.Close SaveChanges:=False
        Exit Sub
    End If

    lastRow = wsParam.Cells(wsParam.Rows.count, IDCrd_col).End(xlUp).row
    If lastRow < 2 Then
        MsgBox "No se encontraron registros en la hoja 'Param'.", vbInformation
        wbParam.Close SaveChanges:=False
        Exit Sub
    End If

    ReDim arrPurposes(lastRow - 1)
    For i = 2 To lastRow
        If Trim(CStr(wsParam.Cells(i, CRD_col).value)) <> "" Then
            arrPurposes(i - 1) = wsParam.Cells(i, CRD_col).value
        End If
    Next i

    With ThisWorkbook.Sheets("Ppal").OLEObjects("Combo2").Object
        .Clear
        Dim k As Long
        For k = LBound(arrPurposes) To UBound(arrPurposes)
            If arrPurposes(k) <> "" Then
                .AddItem arrPurposes(k)
            End If
        Next k
        .MatchEntry = fmMatchEntryComplete
    End With

    wbParam.Close SaveChanges:=False
    Exit Sub

ErrorHandler:
    LogError "LoadCombo2", Err.description
    If Not wbParam Is Nothing Then wbParam.Close SaveChanges:=False
End Sub


Public Sub LoadCombo3(clientID As String)
    On Error GoTo ErrorHandler

    Dim filePath As String
    Dim wbPpto As Workbook
    Dim wsPpto As Worksheet
    Dim dataRange As Range
    Dim arrData As Variant
    Dim arrList() As Variant
    Dim i As Long
    Dim itemCount As Long
    Dim colIDClte As Variant, colFecha As Variant, colTipoProd As Variant
    Dim colDescripcion As Variant, colMonto As Variant, colIDPpto As Variant, colHechox As Variant, co
lValidez As Variant
    Dim Fecha As Date, Validez As Variant, FechaVencimiento As Date, FechaLimite As Date

    ' Inicializar contador de items
    itemCount = 0

    ' Obtener la ruta del archivo Ppto
    filePath = GetFilePath("PptoFilePath")
    If filePath = "" Then Exit Sub

    ' Abrir el libro Ppto en modo lectura y oculto
    Application.ScreenUpdating = False
    Set wbPpto = Workbooks.Open(Filename:=filePath, ReadOnly:=True)
    wbPpto.Windows(1).Visible = False                                    ' Ocultar el libro

    ' Establecer la hoja Ppto
```

```
    Set wsPpto = wbPpto.Sheets("Ppto")

    ' Encontrar las columnas necesarias
    colIDPpto = Application.Match("IDPpto", wsPpto.Rows(1), 0)
    colIDClte = Application.Match("IDClte", wsPpto.Rows(1), 0)
    colFecha = Application.Match("Fecha", wsPpto.Rows(1), 0)
    colTipoProd = Application.Match("TipoProd", wsPpto.Rows(1), 0)
    colDescripcion = Application.Match("Descripción", wsPpto.Rows(1), 0)
    colMonto = Application.Match("Monto", wsPpto.Rows(1), 0)
    colHechox = Application.Match("Hechox", wsPpto.Rows(1), 0)
    colValidez = Application.Match("Validez", wsPpto.Rows(1), 0)

    If IsError(colIDPpto) Or IsError(colIDClte) Or IsError(colFecha) Or IsError(colTipoProd) Or _
       IsError(colDescripcion) Or IsError(colMonto) Or IsError(colHechox) Or IsError(colValidez) Then
        MsgBox "No se encontraron todas las columnas necesarias en la hoja 'Ppto'.", vbCritical
        wbPpto.Close SaveChanges:=False
        Application.ScreenUpdating = True
        Exit Sub
    End If

    ' Leer los datos
    Dim lastRow As Long
    lastRow = wsPpto.Cells(wsPpto.Rows.count, colIDClte).End(xlUp).row
    If lastRow < 2 Then
        ' No hay datos en la hoja Ppto
        wbPpto.Close SaveChanges:=False
        Application.ScreenUpdating = True
        Exit Sub
    End If

    ' Leer el rango de datos
    Set dataRange = wsPpto.Range(wsPpto.Cells(2, 1), wsPpto.Cells(lastRow, 9))
    arrData = dataRange.value

    ' Inicializar arrList con una capacidad inicial
    ReDim arrList(1 To 2, 1 To 10) ' Capacidad inicial para 10 presupuestos

    FechaLimite = DateAdd("m", -6, Date) ' Fecha límite de 6 meses atrás
    For i = 1 To UBound(arrData, 1)
        If Trim(CStr(arrData(i, colIDClte))) = clientID Then
            ' Verificar si el presupuesto es válido
            If IsDate(arrData(i, colFecha)) And IsNumeric(arrData(i, colValidez)) Then       ' Fecha y
 Validez
                Fecha = CDate(arrData(i, colFecha))
                Validez = arrData(i, colValidez)
                FechaVencimiento = DateAdd("d", Validez, Fecha)
                If FechaVencimiento >= FechaLimite Then
                    ' Presupuesto válido o expirado hace menos de 6 meses
                    itemCount = itemCount + 1

                    ' Redimensionar arrList si es necesario
                    If itemCount > UBound(arrList, 2) Then
                        ReDim Preserve arrList(1 To 2, 1 To UBound(arrList, 2) + 10)
                    End If

                    Dim displayText As String
                    displayText = CStr(arrData(i, colIDPpto)) & " | " & Format(arrData(i, colFecha), "
dd/mm/yyyy") & " | " & _
                                  CStr(arrData(i, colTipoProd)) & " | " & CStr(arrData(i, colDescripci
on)) & " | " & CStr(arrData(i, colMonto))
                    If FechaVencimiento < Date Then
                        ' Presupuesto expirado
                        displayText = "(Expirado) " & displayText
                    End If
                    arrList(1, itemCount) = displayText
                    arrList(2, itemCount) = Fecha ' Para ordenar
                End If
            End If
        End If
    Next i

    If itemCount = 0 Then
        ' No se encontraron presupuestos, limpiar Combo3 y salir
```

```
        ThisWorkbook.Sheets("Ppal").OLEObjects("Combo3").Object.Clear
        wbPpto.Close SaveChanges:=False
        Application.ScreenUpdating = True
        Exit Sub
    End If

    ' Redimensionar arrList para ajustarlo al número exacto de elementos
    ReDim Preserve arrList(1 To 2, 1 To itemCount)
    Call QuickSort2D(arrList, itemCount)

    With ThisWorkbook.Sheets("Ppal").OLEObjects("Combo3").Object
        .Clear
        For i = 1 To itemCount
            If arrList(1, i) <> "" Then
                .AddItem arrList(1, i)
            End If
        Next i
        .MatchEntry = fmMatchEntryComplete
    End With

    wbPpto.Close SaveChanges:=False
    Application.ScreenUpdating = True
    Exit Sub

ErrorHandler:
    Call LogError("LoadCombo3", Err.description)
    MsgBox "Error en LoadCombo3: " & Err.description, vbCritical
    If Not wbPpto Is Nothing Then wbPpto.Close SaveChanges:=False
    Application.ScreenUpdating = True
End Sub


' Cargar Combo4 (Cuentas desde la columna 4 en adelante)
Public Sub LoadCombo4()
    On Error GoTo ErrorHandler

    Dim filePath As String
    filePath = GetFilePath("BalFilePath")
    If filePath = "" Then Exit Sub

    Dim wbCaja As Workbook
    Dim wsBal As Worksheet
    Dim startCol As Long, lastCol As Long
    Dim numCuentas As Long
    Dim arrCuentas() As String
    Dim i As Long, arrIndex As Long

    Set wbCaja = Workbooks.Open(Filename:=filePath, ReadOnly:=True)
    wbCaja.Windows(1).Visible = False

    Set wsBal = wbCaja.Sheets("Bal")

    startCol = 4
    lastCol = wsBal.Cells(1, wsBal.Columns.count).End(xlToLeft).Column
    numCuentas = lastCol - startCol + 1

    ReDim arrCuentas(1 To numCuentas)
    arrIndex = 1
    For i = startCol To lastCol
        arrCuentas(arrIndex) = wsBal.Cells(1, i).value
        arrIndex = arrIndex + 1
    Next i

    With ThisWorkbook.Sheets("Ppal").OLEObjects("Combo4").Object
        .Clear
        .List = arrCuentas
        .MatchEntry = fmMatchEntryComplete
    End With

    ' Se puede asignar lo mismo a Combo5 si se usa el mismo set de cuentas
    With ThisWorkbook.Sheets("Ppal").OLEObjects("Combo5").Object
        .Clear
        .List = arrCuentas
```

```
        .MatchEntry = fmMatchEntryComplete
    End With

    wbCaja.Close SaveChanges:=False
    Exit Sub

ErrorHandler:
    LogError "LoadCombo4", Err.description
    If Not wbCaja Is Nothing Then wbCaja.Close SaveChanges:=False
End Sub

' Cargar Combo6 (Divisas)
Public Sub LoadCombo6()
    On Error GoTo ErrorHandler

    Dim filePath As String
    filePath = GetFilePath("DivFilePath")
    If filePath = "" Then Exit Sub

    Dim wbParam As Workbook
    Dim wsDiv As Worksheet
    Dim lastRow As Long
    Dim arrDivisas() As String
    Dim i As Long

    Set wbParam = Workbooks.Open(Filename:=filePath, ReadOnly:=True)
    wbParam.Windows(1).Visible = False

    Set wsDiv = wbParam.Sheets("Div")
    lastRow = wsDiv.Cells(wsDiv.Rows.count, 1).End(xlUp).row
    If lastRow < 2 Then
        MsgBox "No se encontraron registros en la hoja 'Div'.", vbInformation
        wbParam.Close SaveChanges:=False
        Exit Sub
    End If

    ReDim arrDivisas(1 To lastRow - 1)
    Dim pais As String, divisa As String, codigo As String, simbolo As String
    Dim itemTexto As String
    Dim arrIndex As Long: arrIndex = 1

    For i = 2 To lastRow
        pais = wsDiv.Cells(i, 1).value
        divisa = wsDiv.Cells(i, 2).value
        codigo = wsDiv.Cells(i, 3).value
        simbolo = wsDiv.Cells(i, 5).value
        itemTexto = pais & " - " & divisa & " (" & codigo & " - " & simbolo & ")"
        arrDivisas(arrIndex) = itemTexto
        arrIndex = arrIndex + 1
    Next i

    With ThisWorkbook.Sheets("Ppal").OLEObjects("Combo6").Object
        .Clear
        .List = arrDivisas
        .MatchEntry = fmMatchEntryComplete
    End With

    wbParam.Close SaveChanges:=False
    Exit Sub

ErrorHandler:
    LogError "LoadCombo6", Err.description
    If Not wbParam Is Nothing Then wbParam.Close SaveChanges:=False
End Sub

Public Sub CargarDatosCombo2()
    ' Implementar la lógica para insertar el registro en AR11:CE70 según lo necesites,
    ' antes estaba en C26:AG..., ahora usar las nuevas referencias.
    ' Aquí se insertan datos asociados a Propósito de Uso (Combo2) y TextBox4
    ' Logica original ajustada a los nuevos rangos.

    Dim wsPpal As Worksheet
    Set wsPpal = ThisWorkbook.Sheets("Ppal")
```

```
    If Trim(wsPpal.OLEObjects("Combo1").Object.value) = "" Then
        MsgBox "Debe seleccionar un Cliente antes de continuar.", vbExclamation
        Exit Sub
    End If

    With wsPpal.OLEObjects("Combo2").Object
        If .ListIndex = -1 Then
            MsgBox "Debe seleccionar un Propósito de Uso.", vbExclamation
            Exit Sub
        End If

        Dim idCrd As String, crd As String
        Dim montoValue As Variant
        Dim firstEmptyRow As Long
        Dim textToInsert As String

        ' Asumiendo que ID y CRD se obtienen de la selección
        ' (Ajustar según la lógica original)
        idCrd = CStr(.List(.ListIndex, 0))
        crd = CStr(.List(.ListIndex, 1))

        montoValue = GetMontoCRD(idCrd)
        If Not IsNumeric(montoValue) Or montoValue = 0 Then
            MsgBox "Busque en Presupuesto la cotización que le elaboraron al Cliente para este Propósi
to de Uso.", vbInformation
            .value = ""
            Exit Sub
        End If

        If Trim(wsPpal.OLEObjects("TextBox4").Object.value) <> "" Then
            textToInsert = Trim(idCrd) & " " & Trim(crd) & " - " & Trim(wsPpal.OLEObjects("TextBox4").
Object.value)
        Else
            textToInsert = Trim(idCrd) & " " & Trim(crd)
        End If

        ' Determinar la primera fila vacía en AR11:AR70 para insertar el registro
        firstEmptyRow = GetFirstEmptyRow(wsPpal, "AR", 11, 70)
        If firstEmptyRow = 0 Then
            MsgBox "El formulario está lleno, por favor imprima o limpie la información antes de conti
nuar.", vbExclamation
            Exit Sub
        End If

        InsertarRegistro wsPpal, firstEmptyRow, textToInsert, montoValue

        ' Limpiar Combo2 y TextBox4
        wsPpal.OLEObjects("Combo2").Object.value = ""
        wsPpal.OLEObjects("TextBox4").Object.value = ""
    End With
End Sub

Public Sub CargarDatosCombo3()
    ' Similar a CargarDatosCombo2, pero para el Presupuesto (Combo3)
    Dim wsPpal As Worksheet
    Set wsPpal = ThisWorkbook.Sheets("Ppal")

    If Trim(wsPpal.OLEObjects("Combo1").Object.value) = "" Then
        MsgBox "Debe seleccionar un Cliente antes de continuar.", vbExclamation
        Exit Sub
    End If

    Dim lookupValue As String
    lookupValue = Trim(wsPpal.OLEObjects("Combo3").Object.value)
    If lookupValue = "" Then
        MsgBox "Debe seleccionar un Presupuesto en Combo3.", vbExclamation
        Exit Sub
    End If

    Dim selectedIDPpto As String
    If InStr(lookupValue, " | ") > 0 Then
        selectedIDPpto = Split(lookupValue, " | ")(0)
```

```
        selectedIDPpto = Replace(selectedIDPpto, "(Expirado) ", "")
    Else
        selectedIDPpto = lookupValue
    End If

    Dim montoValue As Variant
    montoValue = GetMontoPpto(selectedIDPpto)
    If Not IsNumeric(montoValue) Or montoValue = 0 Then
        MsgBox "El monto asociado al Presupuesto seleccionado no es válido.", vbCritical
        wsPpal.OLEObjects("Combo3").Object.value = ""
        Exit Sub
    End If

    Dim textToInsert As String
    If Trim(wsPpal.OLEObjects("TextBox4").Object.value) <> "" Then
        textToInsert = Trim(lookupValue) & " - " & Trim(wsPpal.OLEObjects("TextBox4").Object.value)
    Else
        textToInsert = Trim(lookupValue)
    End If

    Dim firstEmptyRow As Long
    firstEmptyRow = GetFirstEmptyRow(wsPpal, "AR", 11, 70)
    If firstEmptyRow = 0 Then
        MsgBox "El formulario está lleno, por favor imprima o limpie la información antes de continuar
.", vbExclamation
        Exit Sub
    End If

    InsertarRegistro wsPpal, firstEmptyRow, textToInsert, montoValue

    wsPpal.OLEObjects("Combo3").Object.value = ""
    wsPpal.OLEObjects("TextBox4").Object.value = ""
End Sub

' Obtener el monto CRD
Function GetMontoCRD(idCrd As String) As Variant
    On Error GoTo ErrorHandler

    Dim filePath As String
    Dim wbParam As Workbook
    Dim wsParam As Worksheet
    Dim IDCrd_col As Variant
    Dim costCol As Variant
    Dim foundCell As Range
    Dim costValue As Variant

    GetMontoCRD = Null

    filePath = GetFilePath("ParamFilePath")
    If filePath = "" Then Exit Function

    Application.ScreenUpdating = False
    Set wbParam = Workbooks.Open(Filename:=filePath, ReadOnly:=True)
    wbParam.Windows(1).Visible = False
    Application.ScreenUpdating = True

    Set wsParam = wbParam.Sheets("Param")

    IDCrd_col = Application.Match("IDCrd", wsParam.Rows(1), 0)
    costCol = Application.Match("Cost", wsParam.Rows(1), 0)

    If IsError(IDCrd_col) Or IsError(costCol) Then
        MsgBox "No se encontraron las columnas 'IDCrd' o 'Cost' en la hoja 'Param'.", vbCritical
        wbParam.Close SaveChanges:=False
        Exit Function
    End If

    Set foundCell = wsParam.Columns(IDCrd_col).Find(What:=idCrd, LookIn:=xlValues, LookAt:=xlWhole)
    If foundCell Is Nothing Then
        GetMontoCRD = Null
        wbParam.Close SaveChanges:=False
        Exit Function
    End If
```

```
    costValue = wsParam.Cells(foundCell.row, costCol).value
    wbParam.Close SaveChanges:=False

    If Not IsNull(costValue) Then
        costValue = CleanCurrencyValue(costValue)
    Else
        costValue = Null
    End If

    GetMontoCRD = costValue
    Exit Function
ErrorHandler:
    LogError "GetMontoCRD", Err.description
    MsgBox "Error en GetMontoCRD: " & Err.description, vbCritical
    If Not wbParam Is Nothing Then wbParam.Close SaveChanges:=False
End Function

' Insertar Registro en la hoja "Ppal" a partir de AR11:CE70
Public Sub InsertarRegistro(ws As Worksheet, rowNumber As Long, description As String, amount As Varia
nt)
    If rowNumber < 11 Or rowNumber > 70 Then
        MsgBox "La fila para insertar está fuera del rango permitido (AR11:CE70).", vbExclamation
        Exit Sub
    End If

    ' Ejemplo: descripción en columna AS (antes C) y monto en BW (antes AI)
    ' Ajustar según el mapeo dado:
    ' Antes se insertaba descripción y monto en algún lugar, supongamos que la descripción iba en C y
el monto en AG,
    ' Ahora descripción en AS y monto en BW.
    ' O según tu lógica original.

    ' Suponiendo que la descripción iba antes en "C", ahora va en "AS"
    ' Suponiendo que el monto iba antes en "AG", ahora va en "BU", etc.
    ' Ajustar las columnas exactas según el mapeo dado anteriormente.

    ' Por ejemplo, si antes era:
    ' ws.Cells(rowNumber, "C").Value = description
    ' ws.Cells(rowNumber, "AG").Value = amount
    ' Ahora:
    ' Descripción: AS11:AS70 corresponde a la descripción, por lo tanto columna AS = "AS" = columna 45
 (AR=44, AS=45)
    ' Monto: si el monto antes iba en "AG" (col 33), ahora según el mapeo C26:AG85 ? AS11:BU70,
    ' AG estaba dentro del rango C:AG, ahora todo C:AG ? AS:BU
    ' AG era la última col del rango C:AG, que contaba 1(C),2(D),...hasta AG(?, sumamos col)
    ' Para simplificar, colocaré la descripción en AS y el monto en BU (ajustar si es necesario)

    ws.Cells(rowNumber, "AS").value = description
    ws.Cells(rowNumber, "BU").value = amount

    ' Actualizar la serie en AR (antes B)
    ' Si antes contaba items en la col B, ahora en AR
    Dim itemCount As Long
    itemCount = WorksheetFunction.CountA(ws.Range("AS11:AS" & rowNumber)) ' Contar descripciones
    ws.Cells(rowNumber, "AR").value = itemCount

    ' Actualizar TextBox6 (suma de montos)
    Dim totalAmount As Double
    totalAmount = WorksheetFunction.Sum(ws.Range("BU11:BU70"))
    ws.OLEObjects("TextBox6").Object.value = Format(totalAmount, "0.00")
    FormatearDecimalesTextBox "TextBox6", 2

    ' Actualizar TextBox7 para reflejar el mismo valor que TextBox6
    ws.OLEObjects("TextBox7").Object.value = ws.OLEObjects("TextBox6").Object.value
    FormatearDecimalesTextBox "TextBox7", 2
End Sub

Public Function GetMontoPpto(idPpto As String) As Variant
    On Error GoTo ErrorHandler

    Dim filePath As String
    filePath = GetFilePath("PptoFilePath")
```

Module_Main - 9

```
    If filePath = "" Then Exit Function

    Dim wbPpto As Workbook
    Dim wsPpto As Worksheet
    Dim IDPpto_col As Variant, Monto_col As Variant
    Dim foundCell As Range
    Dim montoValue As Variant

    Set wbPpto = Workbooks.Open(Filename:=filePath, ReadOnly:=True)
    wbPpto.Windows(1).Visible = False

    Set wsPpto = wbPpto.Sheets("Ppto")
    IDPpto_col = Application.Match("IDPpto", wsPpto.Rows(1), 0)
    Monto_col = Application.Match("Monto", wsPpto.Rows(1), 0)
    If IsError(IDPpto_col) Or IsError(Monto_col) Then
        MsgBox "No se encontraron las columnas 'IDPpto' o 'Monto' en la hoja 'Ppto'.", vbCritical
        wbPpto.Close SaveChanges:=False
        Exit Function
    End If

    Set foundCell = wsPpto.Columns(IDPpto_col).Find(What:=idPpto, LookIn:=xlValues, LookAt:=xlWhole)
    If foundCell Is Nothing Then
        GetMontoPpto = Null
        wbPpto.Close SaveChanges:=False
        Exit Function
    End If

    montoValue = wsPpto.Cells(foundCell.row, Monto_col).value
    wbPpto.Close SaveChanges:=False

    If Not IsNull(montoValue) Then
        montoValue = CleanCurrencyValue(montoValue)
    Else
        montoValue = Null
    End If

    GetMontoPpto = montoValue
    Exit Function
ErrorHandler:
    LogError "GetMontoPpto", Err.description
    MsgBox "Error en GetMontoPpto: " & Err.description, vbCritical
    If Not wbPpto Is Nothing Then wbPpto.Close SaveChanges:=False
End Function

Public Function CleanCurrencyValue(value As Variant) As Double
    If IsNull(value) Or value = "" Then
        CleanCurrencyValue = 0
        Exit Function
    End If

    Dim valueString As String
    valueString = CStr(value)
    valueString = Replace(valueString, "$", "")
    valueString = Replace(valueString, "€", "")
    valueString = Replace(valueString, "£", "")
    valueString = Replace(valueString, "R$", "")
    valueString = Replace(valueString, "¥", "")
    valueString = Replace(valueString, "ƒ", "")
    valueString = Replace(valueString, "Bs.", "")
    valueString = Replace(valueString, "$U", "")
    valueString = Replace(valueString, ",", "")
    valueString = Replace(valueString, " ", "")

    If Application.decimalSeparator = "," Then
        valueString = Replace(valueString, ".", ",")
    Else
        valueString = Replace(valueString, ",", "")
    End If

    If IsNumeric(valueString) Then
        CleanCurrencyValue = Round(CDbl(valueString), 2)
    Else
        CleanCurrencyValue = 0
```

```
    End If
End Function

Public Function GetFirstEmptyRow(ws As Worksheet, columnLetter As String, startRow As Long, endRow As
Long) As Long
    Dim cell As Range
    For Each cell In ws.Range(columnLetter & startRow & ":" & columnLetter & endRow)
        If Trim(CStr(cell.value)) = "" Then
            GetFirstEmptyRow = cell.row
            Exit Function
        End If
    Next cell
    GetFirstEmptyRow = 0
End Function

' Procedimiento para preparar el recibo para impresión, autenticación y guardar
Public Sub PrepararReciboParaImpresion()
    On Error GoTo ErrorHandler

    ' Cargar datos en hoja Recibo
    CargarRecibo

    ' Autenticación del usuario con frmLogin
    Dim loginForm As New frmLogin
    loginForm.Show

    If usuarioAutenticado = "" Then
        ' Si el usuario no se autenticó, salir
        Exit Sub
    End If

    Dim wsRecibo As Worksheet
    Set wsRecibo = ThisWorkbook.Sheets("Recibo")

    ' Asignar el usuario autenticado en T52 y T92
    wsRecibo.OLEObjects("T52").Object.value = usuarioAutenticado
    wsRecibo.OLEObjects("T92").Object.value = usuarioAutenticado

    ' Mostrar las opciones de impresión usando frmPrintOptions
    Dim printOptionsForm As New frmPrintOptions
    printOptionsForm.Show

    If printOptionsForm.opcionImpresion = "" Then
        ' Si el usuario canceló o no seleccionó una opción, salir
        Exit Sub
    End If

    If printOptionsForm.impresoraSeleccionada <> "" Then
        Application.ActivePrinter = printOptionsForm.impresoraSeleccionada
    End If

    Dim numCopias As Integer
    numCopias = printOptionsForm.numeroCopias

    Select Case printOptionsForm.opcionImpresion
        Case "ImprimirDirecto"
            wsRecibo.PrintOut Copies:=numCopias
        Case "Previsualizar"
            wsRecibo.PrintPreview
        Case "ExportarPDF"
            Dim pdfPath As String
            pdfPath = Application.GetSaveAsFilename(InitialFileName:="Recibo.pdf", FileFilter:="PDF Fi
les (*.pdf), *.pdf")
            If pdfPath <> "False" Then
                ExportarHojaComoPDF "Recibo", pdfPath
            End If
    End Select

    GuardarRecibo

    Exit Sub

ErrorHandler:
```

```
    MsgBox "Error en PrepararReciboParaImpresion: " & Err.description, vbCritical
End Sub


' Exportar hoja como PDF
Public Sub ExportarHojaComoPDF(nombreHoja As String, filePath As Variant)
    On Error GoTo ErrorHandler

    Dim ws As Worksheet
    Set ws = ThisWorkbook.Sheets(nombreHoja)

    ws.ExportAsFixedFormat Type:=xlTypePDF, Filename:=filePath, Quality:=xlQualityStandard, IncludeDoc
Properties:=True, IgnorePrintAreas:=False, OpenAfterPublish:=False

    Exit Sub
ErrorHandler:
    MsgBox "Error al exportar la hoja " & nombreHoja & " como PDF: " & Err.description, vbCritical
End Sub

Public Sub GuardarRecibo()
    On Error GoTo ErrorHandler

    Dim filePath As String
    Dim wbRCAD As Workbook
    Dim wsRCAD As Worksheet
    Dim wsRecibo As Worksheet

    Set wsRecibo = ThisWorkbook.Sheets("Recibo")

    filePath = GetFilePath("RCADFilePath")
    If filePath = "" Then
        MsgBox "No se encontró la ruta de RCADFilePath en el archivo de configuración.", vbExclamation
        Exit Sub
    End If

    Set wbRCAD = Workbooks.Open(Filename:=filePath, ReadOnly:=False)
    wbRCAD.Windows(1).Visible = False

    Set wsRCAD = wbRCAD.Sheets("RCAD")

    Dim lastRow As Long
    lastRow = wsRCAD.Cells(wsRCAD.Rows.count, 1).End(xlUp).row + 1

    ' Copiar datos necesarios desde Recibo a RCAD (ajustar según lógica original)
    ' Ejemplo:
    wsRCAD.Cells(lastRow, 1).value = wsRecibo.OLEObjects("T16").Object.value ' Fecha
    wsRCAD.Cells(lastRow, 2).value = wsRecibo.OLEObjects("T12").Object.value ' IDClte
    wsRCAD.Cells(lastRow, 7).value = wsRecibo.OLEObjects("T14").Object.value ' Monto_Tot
    wsRCAD.Cells(lastRow, 5).value = wsRecibo.OLEObjects("T52").Object.value ' Elabx

    wbRCAD.Close SaveChanges:=True
    Exit Sub

ErrorHandler:
    LogError "GuardarRecibo", Err.description
    If Not wbRCAD Is Nothing Then wbRCAD.Close SaveChanges:=False
    MsgBox "Error en GuardarRecibo: " & Err.description, vbCritical
End Sub

' Función para ordenar arrays (QuickSort)
Public Sub QuickSort(arr() As String, ByVal first As Long, ByVal last As Long)
    Dim low As Long, high As Long
    Dim pivot As String, temp As String

    low = first
    high = last
    pivot = arr((first + last) \ 2)

    Do While low <= high
        Do While arr(low) < pivot
            low = low + 1
        Loop
        Do While arr(high) > pivot
```

```
            high = high - 1
        Loop
        If low <= high Then
            temp = arr(low)
            arr(low) = arr(high)
            arr(high) = temp
            low = low + 1
            high = high - 1
        End If
    Loop

    If first < high Then QuickSort arr, first, high
    If low < last Then QuickSort arr, low, last
End Sub

' Procedimiento para ordenar un array 2D basado en la segunda columna (Fecha) descendente
Public Sub QuickSort2D(ByRef arr() As Variant, ByVal itemCount As Long)
    QuickSortHelper arr, 1, itemCount
End Sub

Private Sub QuickSortHelper(ByRef arr() As Variant, ByVal first As Long, ByVal last As Long)
    Dim low As Long, high As Long
    Dim pivot As Date, temp1 As Variant, temp2 As Variant

    low = first
    high = last
    pivot = arr(2, (first + last) \ 2) ' Pivot based on Fecha

    Do While low <= high
        Do While arr(2, low) > pivot
            low = low + 1
        Loop
        Do While arr(2, high) < pivot
            high = high - 1
        Loop
        If low <= high Then
            ' Swap DisplayText
            temp1 = arr(1, low)
            arr(1, low) = arr(1, high)
            arr(1, high) = temp1
            ' Swap Fecha
            temp2 = arr(2, low)
            arr(2, low) = arr(2, high)
            arr(2, high) = temp2
            low = low + 1
            high = high - 1
        End If
    Loop

    If first < high Then QuickSortHelper arr, first, high
    If low < last Then QuickSortHelper arr, low, last
End Sub

' Procedimiento para Ordenar el Array de Presupuestos por Fecha Descendente
Public Sub QuickSortPresupuestos(ByRef arrList() As String, ByVal itemCount As Long)
    Dim i As Long, j As Long
    Dim temp As String
    Dim fecha1 As Date, fecha2 As Date

    ' Implementar un simple algoritmo de ordenación (Bubble Sort para simplicidad)
    For i = 1 To itemCount - 1
        For j = i + 1 To itemCount
            ' Extraer la fecha del displayText
            fecha1 = CDate(Split(Split(arrList(i), " | ")(1), " | ")(0))
            fecha2 = CDate(Split(Split(arrList(j), " | ")(1), " | ")(0))
            If fecha1 < fecha2 Then
                ' Intercambiar displayText
                temp = arrList(i)
                arrList(i) = arrList(j)
                arrList(j) = temp
            End If
        Next j
    Next i
```

```
Module_Main - 13

End Sub

Public Sub HandleCombo1Change()
    If isInitializing Then Exit Sub

    Dim wsPpal As Worksheet
    Set wsPpal = ThisWorkbook.Sheets("Ppal")

    Dim nombreCliente As String
    nombreCliente = wsPpal.OLEObjects("Combo1").Object.value

    If nombreCliente = "" Then
        wsPpal.OLEObjects("TextBox2").Object.value = ""
        wsPpal.OLEObjects("Combo3").Object.Clear
        Exit Sub
    End If

    Dim idCliente As String
    idCliente = GetClientID(nombreCliente)

    If idCliente <> "" Then
        wsPpal.OLEObjects("TextBox2").Object.value = idCliente
        ' Cargar Combo3 (Presupuestos) para el cliente seleccionado
        LoadCombo3 idCliente
    Else
        wsPpal.OLEObjects("TextBox2").Object.value = ""
        wsPpal.OLEObjects("Combo3").Object.Clear
    End If

End Sub

Public Sub HandleCombo2Change()
    ' Lógica para Combo2 Change
End Sub

Public Sub HandleCombo3Change()
    If isInitializing Then Exit Sub

    Dim wsPpal As Worksheet
    Set wsPpal = ThisWorkbook.Sheets("Ppal")

    Dim idPpto As String
    idPpto = wsPpal.OLEObjects("Combo3").Object.value

    If idPpto = "" Then
        wsPpal.OLEObjects("TextBox6").Object.value = ""
        Exit Sub
    End If

    Dim montoPpto As Double
    montoPpto = GetMontoPpto(idPpto)

    If montoPpto > 0 Then
        wsPpal.OLEObjects("TextBox6").Object.value = FormatNumber(montoPpto, 2)
    Else
        wsPpal.OLEObjects("TextBox6").Object.value = ""
    End If

End Sub

' Manejar el evento Change de Combo4 (Cuenta1)
Public Sub HandleCombo4Change()
    If isInitializing Then Exit Sub
    HandleTextBox7Change
End Sub

' Manejar el evento Change de Combo5 (Cuenta2)
Public Sub HandleCombo5Change()
    If isInitializing Then Exit Sub
    HandleTextBox10Change

End Sub
```

```
' Manejar el evento Change de Combo6 (Divisas)
Public Sub HandleCombo6Change()
    If isInitializing Then Exit Sub

End Sub


Public Sub HandleTextBox9Change()
    UpdatePaymentState
End Sub


' Función para obtener la lista de clientes
Private Function ObtenerListaClientes(ByVal ws As Worksheet) As Variant
    Dim firstName_col As Long
    Dim lastName_col As Long
    Dim lastRow As Long
    Dim arrClients() As String
    Dim i As Long

    firstName_col = 2 ' Columna Nombre
    lastName_col = 3  ' Columna Apellido

    lastRow = ws.Cells(ws.Rows.count, firstName_col).End(xlUp).row
    ReDim arrClients(1 To lastRow - 1)

    For i = 2 To lastRow
        arrClients(i - 1) = ws.Cells(i, firstName_col).value & " " & ws.Cells(i, lastName_col).value
    Next i

    ' Ordenar la lista
    QuickSort arrClients, LBound(arrClients), UBound(arrClients)

    ObtenerListaClientes = arrClients
End Function


' Función para obtener el ID del cliente dado su nombre completo
Private Function GetClientID(ByVal nombreCompleto As String) As String
    On Error GoTo ErrorHandler

    Dim filePath As String
    Dim wbClte As Workbook
    Dim wsClte As Worksheet
    Dim lastRow As Long
    Dim i As Long
    Dim foundCell As Range
    Dim IDClte_col As Variant
    Dim firstName_col As Variant, secondName_col As Variant
    Dim firstLastName_col As Variant, secondLastName_col As Variant
    Dim nombreCliente, fullName As String

    GetClientID = ""

    ' Obtener la ruta del archivo Clte
    filePath = GetFilePath("ClteFilePath")
    If filePath = "" Then Exit Function

    ' Abrir el libro Clte en modo lectura y oculto
    Application.ScreenUpdating = False
    Set wbClte = Workbooks.Open(Filename:=filePath, ReadOnly:=True)
    wbClte.Windows(1).Visible = False ' Ocultar el libro

    ' Establecer la hoja Cltes
    Set wsClte = wbClte.Sheets("Cltes")

    ' Encontrar las columnas necesarias
    IDClte_col = Application.Match("IDClte", wsClte.Rows(1), 0)
    firstName_col = Application.Match("1° Nombre", wsClte.Rows(1), 0)
    secondName_col = Application.Match("2° Nombre", wsClte.Rows(1), 0)
    firstLastName_col = Application.Match("1° Apellido", wsClte.Rows(1), 0)
    secondLastName_col = Application.Match("2° Apellido", wsClte.Rows(1), 0)
```

```
    If IsError(IDClte_col) Or IsError(firstName_col) Or IsError(secondName_col) Or IsError(firstLastNa
me_col) Or IsError(secondLastName_col) Then
        MsgBox "No se encontraron las columnas necesarias en la hoja 'Cltes'.", vbCritical
        wbClte.Close SaveChanges:=False
        Exit Function
    End If

    ' Leer los datos
    lastRow = wsClte.Cells(wsClte.Rows.count, IDClte_col).End(xlUp).row


    ' Buscar el cliente por nombre completo
    lastRow = wsClte.Cells(wsClte.Rows.count, 1).End(xlUp).row

    For i = 2 To lastRow
        nombreCliente = BuildFullName(wsClte.Cells(i, firstName_col).value, _
                        wsClte.Cells(i, secondName_col).value, _
                        wsClte.Cells(i, firstLastName_col).value, _
                        wsClte.Cells(i, secondLastName_col).value)
        If nombreCliente = fullName Then
            GetClientID = Trim(CStr(wsClte.Cells(i, IDClte_col).value))
            Exit For
        End If
    Next i
    wbClte.Close SaveChanges:=False
    Exit Function

ErrorHandler:
    LogError "GetClientID", Err.description
    MsgBox "Error en GetClientID: " & Err.description, vbCritical
    If Not wbClte Is Nothing Then wbClte.Close SaveChanges:=False
End Function

' Función para Construir el Nombre Completo Según el Criterio Especificado
Public Function BuildFullName(firstName As String, secondName As String, firstLastName As String, seco
ndLastName As String) As String
    Dim fullName As String

    fullName = Trim(firstName)

    If Trim(secondName) <> "" Then
        fullName = fullName & " " & Trim(secondName)
    End If

    fullName = fullName & ", " & Trim(firstLastName)

    If Trim(secondLastName) <> "" Then
        fullName = fullName & " " & Trim(secondLastName)
    End If

    BuildFullName = fullName
End Function
```

Module_Payment - 1

```
'********** Module_Payment **********
Option Explicit

' Añadido Bloque de Variables para controlar la recursión de eventos, así como la rutina
' InitializePaymentBlock

' Variables privadas para controlar la recursión en eventos
Private isHandlingEvent As Boolean
Private isUpdatingTextBox9 As Boolean
Private isHandlingTextBox9Change As Boolean


Public Sub UpdatePaymentState()
    ' Lógica para actualizar estado del bloque de pago
    ' Antes se llamaba ActualizarEstado, ahora todo se llama desde aquí.
    ' Si antes la lógica estaba en la hoja Ppal, movemos acá su contenido.
    ' Simplemente llama al evento en la hoja si todavía está ahí, o copiar su lógica aquí.
    ' Suponiendo que ya hemos integrado la lógica en la hoja, copie aquí el contenido original de Actu
alizarEstado:

    ' Por simplicidad, asumiendo que UpdatePaymentState es llamado y la lógica original
    ' estaba en la hoja Ppal, ahora la integras aquí directamente.
    ' Ajustar la lógica según las variables definidas.

    ' Debido a la complejidad, se recomienda mantener la lógica ya definida en la hoja Ppal (Actualiza
doEstado)
    ' y desde los eventos del sheet llamar a esta función. Pero ya fue integrado el código en la respu
esta previa.
    ' Por ahora, si el código ya está en la hoja Ppal en el evento, se puede dejarlo allí o copiar acá
.

    ' Nota: Se ha asumido que UpdatePaymentState es la encargada de recalcular y habilitar btn4.
    ' Ya que en el código previo estaba la lógica en un sub privado en la hoja, lo replicamos aquí:

    Dim wsPpal As Worksheet
    Set wsPpal = ThisWorkbook.Sheets("Ppal")

    ' Obtener valores
    Dim t6 As Double, t7 As Double, t9 As Double, t10 As Double
    Dim t8 As String, t11 As String
    t6 = ConvertirTextoANumero(wsPpal.OLEObjects("TextBox6").Object.value, 2)
    t7 = ConvertirTextoANumero(wsPpal.OLEObjects("TextBox7").Object.value, 2)
    t9 = ConvertirTextoANumero(wsPpal.OLEObjects("TextBox9").Object.value, 2)
    t10 = ConvertirTextoANumero(wsPpal.OLEObjects("TextBox10").Object.value, 2)
    t8 = Trim(wsPpal.OLEObjects("TextBox8").Object.value)
    t11 = Trim(wsPpal.OLEObjects("TextBox11").Object.value)

    Dim totalPagado As Double
    totalPagado = CalcularTotalPagado(t7, t9, t10, t8, t11)

    wsPpal.OLEObjects("btn4").Object.Enabled = (Round(totalPagado, 2) = Round(t6, 2))
End Sub

' Procedimiento para inicializar el bloque de pago
Public Sub InitializePaymentBlock()
    Dim wsPpal As Worksheet
    Set wsPpal = ThisWorkbook.Sheets("Ppal")

    ' Desactivar botón Imprimir al iniciar
    wsPpal.OLEObjects("btn4").Object.Enabled = False

    ' Inicializar controles
    With wsPpal
        ' TextBox5 (Tasa de Cambio)
        .OLEObjects("TextBox5").Object.value = ""
        .OLEObjects("TextBox5").Object.BackColor = colorEditable
        .OLEObjects("TextBox5").Object.Locked = False

        ' TextBox6 (Monto Acumulado)
        .OLEObjects("TextBox6").Object.value = ""
        .OLEObjects("TextBox6").Object.BackColor = colorBlanco
        .OLEObjects("TextBox6").Object.Locked = True
```

```
        ' TextBox7 (Cargo1)
        .OLEObjects("TextBox7").Object.value = ""
        .OLEObjects("TextBox7").Object.BackColor = colorEditable
        .OLEObjects("TextBox7").Object.Locked = False

        ' TextBox8 (Estado1)
        .OLEObjects("TextBox8").Object.value = ""
        .OLEObjects("TextBox8").Object.BackColor = colorBlanco
        .OLEObjects("TextBox8").Object.Locked = True

        ' TextBox9 (Efectivo)
        .OLEObjects("TextBox9").Object.value = ""
        .OLEObjects("TextBox9").Object.BackColor = colorEditable
        .OLEObjects("TextBox9").Object.Locked = False

        ' TextBox10 (Cargo2)
        .OLEObjects("TextBox10").Object.value = ""
        .OLEObjects("TextBox10").Object.BackColor = colorEditable
        .OLEObjects("TextBox10").Object.Locked = False

        ' TextBox11 (Estado2)
        .OLEObjects("TextBox11").Object.value = ""
        .OLEObjects("TextBox11").Object.BackColor = colorBlanco
        .OLEObjects("TextBox11").Object.Locked = True

        ' TextBox12 (Monto Divisa)
        .OLEObjects("TextBox12").Object.value = ""
        .OLEObjects("TextBox12").Object.BackColor = colorBlanco
        .OLEObjects("TextBox12").Object.Locked = True

        ' Combo4 y Combo5
        .OLEObjects("Combo4").Object.value = ""
        .OLEObjects("Combo4").Object.BackColor = colorEditable
        .OLEObjects("Combo4").Object.Enabled = True

        .OLEObjects("Combo5").Object.value = ""
        .OLEObjects("Combo5").Object.BackColor = colorEditable
        .OLEObjects("Combo5").Object.Enabled = True

        ' Combo6
        .OLEObjects("Combo6").Object.value = ""
        .OLEObjects("Combo6").Object.BackColor = colorEditable
        .OLEObjects("Combo6").Object.Enabled = True
    End With
End Sub

' Manejar el evento Change de TextBox7 (Cargo1)
Public Sub HandleTextBox7Change()
    If isHandlingEvent Then Exit Sub
    isHandlingEvent = True

    ValidateCargo "TextBox7", "Combo4", "TextBox8"
    Call FormatearDecimalesTextBox("TextBox7", 2)
    UpdatePaymentState
    isHandlingEvent = False
End Sub

' Manejar el evento Change de TextBox10 (Cargo2)
Public Sub HandleTextBox10Change()
    If isHandlingEvent Then Exit Sub
    isHandlingEvent = True

    ValidateCargo "TextBox10", "Combo5", "TextBox11"
    Call FormatearDecimalesTextBox("TextBox10", 2)
    UpdatePaymentState
    isHandlingEvent = False
End Sub


Private Sub ValidateCargo(ByVal txtCargo As String, ByVal cmbCuenta As String, ByVal txtEstado As Stri
ng)
    Dim wsPpal As Worksheet
    Set wsPpal = ThisWorkbook.Sheets("Ppal")
```

```
    Dim t6 As Double, tCargo As Double
    Dim idCliente As String
    Dim cuenta As String
    Dim saldoCuenta As Double

    ' Obtener el ID del cliente
    idCliente = Trim(wsPpal.OLEObjects("TextBox2").Object.value)
    If idCliente = "" Then Exit Sub

    ' Obtener valores
    t6 = ConvertirTextoANumero(wsPpal.OLEObjects("TextBox6").Object.value, 2)
    tCargo = ConvertirTextoANumero(wsPpal.OLEObjects(txtCargo).Object.value, 2)
    cuenta = Trim(wsPpal.OLEObjects(cmbCuenta).Object.value)

    ' Validar que el cargo no sea mayor que t6
    If tCargo > t6 Then
        MsgBox "El monto no puede ser mayor que el Monto Acumulado.", vbExclamation
        wsPpal.OLEObjects(txtCargo).Object.value = ""
        tCargo = 0
    End If

    ' Manejo de la cuenta asociada
    If tCargo = 0 Then
        wsPpal.OLEObjects(cmbCuenta).Object.value = ""
        wsPpal.OLEObjects(cmbCuenta).Object.Enabled = False
        wsPpal.OLEObjects(txtEstado).Object.value = ""
        wsPpal.OLEObjects(txtEstado).Object.BackColor = colorBlanco
    Else
        wsPpal.OLEObjects(cmbCuenta).Object.Enabled = True
        If cuenta <> "" Then
            saldoCuenta = GetSaldoCuenta(idCliente, cuenta)
            If tCargo <= saldoCuenta Then
                wsPpal.OLEObjects(txtEstado).Object.value = ESTADO_OK
                wsPpal.OLEObjects(txtEstado).Object.BackColor = colorOk
            Else
                wsPpal.OLEObjects(txtEstado).Object.value = ESTADO_NEGADO
                wsPpal.OLEObjects(txtEstado).Object.BackColor = colorNegado
            End If
        Else
            wsPpal.OLEObjects(txtEstado).Object.value = ""
            wsPpal.OLEObjects(txtEstado).Object.BackColor = colorBlanco
        End If
    End If

    ' Verificar que las cuentas en Combo4 y Combo5 sean diferentes
    ValidateCombos

    ' Actualizar el estado general
    UpdatePaymentState
End Sub

' Función para obtener el saldo de una cuenta específica para un cliente
Public Function GetSaldoCuenta(ByVal idCliente As String, ByVal cuenta As String) As Double
    On Error GoTo ErrorHandler

    Dim filePath As String
    Dim wbCaja As Workbook
    Dim wsBal As Worksheet
    Dim tblBal As ListObject
    Dim foundRow As Range
    Dim saldo As Double
    Dim IDClte_col As Long
    Dim cuenta_col As Long

    ' Inicializar el valor de retorno
    GetSaldoCuenta = 0

    ' Obtener la ruta del archivo BalFilePath desde Config
    filePath = GetFilePath("BalFilePath")
    If filePath = "" Then Exit Function

    ' Abrir el libro si no está ya abierto
```

```
    If Not IsWorkbookOpen(filePath) Then
        Set wbCaja = Workbooks.Open(Filename:=filePath, ReadOnly:=True)
        wbCaja.Windows(1).Visible = False ' Ocultar el libro
    Else
        Set wbCaja = Workbooks(Dir(filePath))
    End If

    ' Establecer la hoja Bal
    Set wsBal = wbCaja.Sheets("Bal")

    ' Establecer la tabla estructurada 'Bal'
    Set tblBal = wsBal.ListObjects("Bal")

    ' Encontrar las columnas 'IDClte' y la cuenta especificada
    IDClte_col = tblBal.ListColumns("IDClte").index
    cuenta_col = tblBal.ListColumns(cuenta).index

    ' Buscar el cliente y obtener el saldo
    With tblBal.DataBodyRange
        Set foundRow = .Columns(IDClte_col).Find(What:=idCliente, LookIn:=xlValues, LookAt:=xlWhole)
    End With

    If Not foundRow Is Nothing Then
        ' Obtener el saldo de la cuenta
        If IsNumeric(foundRow.Offset(0, cuenta_col - IDClte_col).value) Then
            saldo = CDbl(foundRow.Offset(0, cuenta_col - IDClte_col).value)
            GetSaldoCuenta = saldo
        Else
            GetSaldoCuenta = 0
        End If
    Else
        ' Cliente no encontrado
        GetSaldoCuenta = 0
    End If

    ' Cerrar el libro Bal si fue abierto por esta función
    If wbCaja.ReadOnly Then
        wbCaja.Close SaveChanges:=False
    End If

    Exit Function

ErrorHandler:
    ' Manejo de errores
    If Not wbCaja Is Nothing Then
        If wbCaja.ReadOnly Then
            wbCaja.Close SaveChanges:=False
        End If
    End If
    GetSaldoCuenta = 0
End Function

' Procedimiento para asegurar que Combo4 y Combo5 sean diferentes
Public Sub ValidateCombos()
    Dim wsPpal As Worksheet
    Set wsPpal = ThisWorkbook.Sheets("Ppal")

    Dim cuenta1 As String
    Dim cuenta2 As String

    cuenta1 = Trim(wsPpal.OLEObjects("Combo4").Object.value)
    cuenta2 = Trim(wsPpal.OLEObjects("Combo5").Object.value)

    If cuenta1 <> "" And cuenta2 <> "" Then
        If cuenta1 = cuenta2 Then
            MsgBox "Las cuentas seleccionadas en Cuenta1 y Cuenta2 deben ser diferentes.", vbExclamati
on
            ' Limpiar solo Combo5
            wsPpal.OLEObjects("Combo5").Object.value = ""
            ' No se borra t10 ni t11
            ' wsPpal.OLEObjects("TextBox10").Object.Value = ""
            ' wsPpal.OLEObjects("TextBox10").Object.ForeColor = colorNegro
             wsPpal.OLEObjects("TextBox11").Object.value = ""
```

```
            wsPpal.OLEObjects("TextBox11").Object.BackColor = colorBlanco
        End If
    End If
End Sub

' Procedimiento para convertir monto de divisa
Public Sub ConvertirMontoDivisa()
    On Error GoTo ErrorHandler

    Dim wsPpal As Worksheet
    Set wsPpal = ThisWorkbook.Sheets("Ppal")

    Dim tasaCambio As Double
    Dim montoGuaranies As Double
    Dim montoDivisa As Double
    Dim divisaSeleccionada As String

    ' Obtener valores
    tasaCambio = ConvertirTextoANumero(wsPpal.OLEObjects("TextBox5").Object.value, 3)
    montoGuaranies = ConvertirTextoANumero(wsPpal.OLEObjects("TextBox9").Object.value, 2)
    divisaSeleccionada = wsPpal.OLEObjects("Combo6").Object.value

    ' Validar que los valores sean válidos
    If tasaCambio <= 0 Then
        MsgBox "Debe ingresar una tasa de cambio válida.", vbExclamation
        Exit Sub
    End If

    If montoGuaranies <= 0 Then
        MsgBox "El monto en efectivo en guaraníes debe ser mayor a cero.", vbExclamation
        Exit Sub
    End If

    If divisaSeleccionada = "" Then
        MsgBox "Debe seleccionar una divisa.", vbExclamation
        Exit Sub
    End If

    ' Realizar el cálculo
    montoDivisa = montoGuaranies / tasaCambio

    ' Mostrar el resultado en TextBox12
    wsPpal.OLEObjects("TextBox12").Object.value = FormatNumber(montoDivisa, 2)

    ' Formatear TextBox12 con separadores de miles
    ' (Se asume que ya está formateado correctamente)

    Exit Sub

ErrorHandler:
    LogError "ConvertirMontoDivisa", Err.description
    MsgBox "Error en ConvertirMontoDivisa: " & Err.description, vbCritical
End Sub
```

Module_PrintSave - 1

```
'********** Module_PrintSave **********
Option Explicit

Public Sub CargarRecibo()
    ' Lógica para cargar datos en la hoja "Recibo"
    ' Ajustar según la lógica original
    ' Si se había dado código antes en la conversación, integrarlo aquí:
    ' Ejemplo:
    Dim wsPpal As Worksheet
    Dim wsRecibo As Worksheet

    Set wsPpal = ThisWorkbook.Sheets("Ppal")
    Set wsRecibo = ThisWorkbook.Sheets("Recibo")

    ' Limpiar y preparar Recibo
    ' Copiar datos desde Ppal a Recibo según la lógica original
    ' Ajustar las columnas y rangos que se han cambiado.
    ' Este es un ejemplo, debes adecuarlo a la lógica original que tenías:
    wsRecibo.Range("C21:AG50").ClearContents
    wsRecibo.Range("C94:AG123").ClearContents
    wsRecibo.Range("AP73").ClearContents

    ' Asignar valores a T41, T81, etc., según lógica original
    ' Ajustar con el IDClte, Trat, etc.

    ' Esta parte depende totalmente de la lógica original que no se detalló completamente aquí.
    ' Insertar el código original de CargarRecibo que tenías y adaptar rangos si es necesario.

    ' Debido a la longitud y a que no se ha proporcionado la lógica exacta final de CargarRecibo en es
te último pedido,
    ' incluyo un placeholder. Debes pegar aquí el código final de CargarRecibo que tenías inicialmente
.
End Sub

Public Sub GuardarRecibo()
    ' Ya implementado en Module_Main (donde se colocó GuardarRecibo)
    ' Si prefieres moverlo aquí, corta y pega la lógica desde Module_Main a aquí y borrar de Module_Ma
in.
End Sub

Public Sub ExportarHojaComoPDF(nombreHoja As String, filePath As Variant)
    ' Ya implementado en Module_Main, igual que GuardarRecibo.
    ' Si prefieres tenerlo aquí, corta y pega desde Module_Main.
End Sub
```

Module_Utils - 1

```
'********** Module_Utils **********
Option Explicit

Public Sub LogError(ByVal routineName As String, ByVal errorMessage As String)
    Dim logFilePath As String
    Dim fileNum As Integer

    logFilePath = ThisWorkbook.Path & "\ErrorLog.txt"
    fileNum = FreeFile
    Open logFilePath For Append As #fileNum
    Print #fileNum, Now & " - " & routineName & ": " & errorMessage
    Close #fileNum
End Sub

Public Function BuildFullName(firstName As String, secondName As String, firstLastName As String, seco
ndLastName As String) As String
    Dim fullName As String
    fullName = Trim(firstName)
    If Trim(secondName) <> "" Then
        fullName = fullName & " " & Trim(secondName)
    End If
    fullName = fullName & ", " & Trim(firstLastName)
    If Trim(secondLastName) <> "" Then
        fullName = fullName & " " & Trim(secondLastName)
    End If
    BuildFullName = fullName
End Function

Public Sub FormatearDecimalesTextBox(ByVal nombreTextBox As String, ByVal numeroDecimales As Integer)
    Dim ws As Worksheet
    Set ws = ThisWorkbook.Sheets("Ppal")

    Dim tb As Object
    Set tb = ws.OLEObjects(nombreTextBox).Object

    If IsNumeric(tb.value) Then
        tb.value = FormatNumber(CDbl(tb.value), numeroDecimales, , , vbTrue)
    End If
End Sub

Public Function ConvertirTextoANumero(ByVal texto As String, Optional ByVal decimales As Integer = 2)
As Double
    Dim textoLimpio As String
    textoLimpio = Replace(texto, Application.ThousandsSeparator, "")
    textoLimpio = Replace(textoLimpio, Application.decimalSeparator, ".")
    If IsNumeric(textoLimpio) Then
        ConvertirTextoANumero = Round(CDbl(textoLimpio), decimales)
    Else
        ConvertirTextoANumero = 0
    End If
End Function

Public Sub HandleKeyPress(tb As OLEObject, ByRef KeyAscii As MSForms.ReturnInteger)
    If KeyAscii < 32 Then Exit Sub

    Dim decimalSeparator As String
    decimalSeparator = Application.decimalSeparator

    If Not (Chr(KeyAscii) Like "[0-9]" Or Chr(KeyAscii) = decimalSeparator) Then
        KeyAscii = 0
        Exit Sub
    End If

    If Chr(KeyAscii) = decimalSeparator Then
        If InStr(tb.Object.Text, decimalSeparator) > 0 Then
            If tb.Object.SelText <> decimalSeparator Then
                KeyAscii = 0
                Exit Sub
            End If
        End If
    End If
End Sub
```

Module_Utils - 2

```
Public Function CalcularTotalPagado(t7 As Double, t9 As Double, t10 As Double, t8 As String, t11 As St
ring) As Double
    If t9 >= 0 Then
        Select Case True
            Case t8 = ESTADO_OK And t11 = ESTADO_OK
                CalcularTotalPagado = t7 + t9 + t10
            Case t8 = ESTADO_OK And (t11 = ESTADO_NEGADO Or t11 = ESTADO_VACIO)
                CalcularTotalPagado = t7 + t9
            Case t8 = ESTADO_NEGADO And t11 = ESTADO_OK
                CalcularTotalPagado = t9 + t10
            Case t8 = ESTADO_NEGADO And (t11 = ESTADO_NEGADO Or t11 = ESTADO_VACIO)
                CalcularTotalPagado = t9
            Case t8 = ESTADO_VACIO And t11 = ESTADO_OK
                CalcularTotalPagado = t9 + t10
            Case t8 = ESTADO_VACIO And (t11 = ESTADO_NEGADO Or t11 = ESTADO_VACIO)
                CalcularTotalPagado = t9
        End Select
    Else
        Select Case True
            Case t8 = ESTADO_OK And t11 = ESTADO_OK
                CalcularTotalPagado = t7 + t10
            Case t8 = ESTADO_OK And (t11 = ESTADO_NEGADO Or t11 = ESTADO_VACIO)
                CalcularTotalPagado = t7
            Case t8 = ESTADO_NEGADO And t11 = ESTADO_OK
                CalcularTotalPagado = t10
            Case t8 = ESTADO_NEGADO And (t11 = ESTADO_NEGADO Or t11 = ESTADO_VACIO)
                CalcularTotalPagado = 0
            Case t8 = ESTADO_VACIO And t11 = ESTADO_OK
                CalcularTotalPagado = t10
            Case t8 = ESTADO_VACIO And (t11 = ESTADO_NEGADO Or t11 = ESTADO_VACIO)
                CalcularTotalPagado = 0
        End Select
    End If
End Function

Public Function ValidarUsuario(usuario As String, contrasena As String) As Boolean
    On Error GoTo ErrorHandler

    Dim filePath As String
    filePath = GetFilePath("UsersFilePath")
    If filePath = "" Then
        MsgBox "No se encontró la ruta de UsersFilePath en el archivo de configuración.", vbExclamatio
n
        ValidarUsuario = False
        Exit Function
    End If

    If Dir(filePath) = "" Then
        MsgBox "El archivo de usuarios no existe en la ruta especificada: " & filePath, vbExclamation
        ValidarUsuario = False
        Exit Function
    End If

    Dim wbUsers As Workbook
    Dim wsUsers As Worksheet
    Dim users_col As Long
    Dim password_col As Long
    Dim lastRow As Long
    Dim i As Long

    Set wbUsers = Workbooks.Open(Filename:=filePath, ReadOnly:=True)
    wbUsers.Windows(1).Visible = False
    Set wsUsers = wbUsers.Sheets("Users")

    users_col = Application.Match("Users", wsUsers.Rows(1), 0)
    password_col = Application.Match("Password", wsUsers.Rows(1), 0)

    If users_col > 0 And password_col > 0 Then
        lastRow = wsUsers.Cells(wsUsers.Rows.count, users_col).End(xlUp).row
        For i = 2 To lastRow
            If wsUsers.Cells(i, users_col).value = usuario Then
                If wsUsers.Cells(i, password_col).value = contrasena Then
                    ValidarUsuario = True
```

```
                    Exit For
                End If
            End If
        Next i
    Else
        MsgBox "No se encontraron las columnas 'Users' y 'Password' en la hoja 'Users'.", vbExclamatio
n
    End If

    wbUsers.Close SaveChanges:=False
    Exit Function

ErrorHandler:
    LogError "ValidarUsuario", Err.description
    MsgBox "Error en ValidarUsuario: " & Err.description, vbCritical
    If Not wbUsers Is Nothing Then wbUsers.Close SaveChanges:=False
    ValidarUsuario = False
End Function

Public Sub EliminarSeleccionados()
    On Error GoTo ErrorHandler

    Dim wsPpal As Worksheet
    Set wsPpal = ThisWorkbook.Sheets("Ppal")

    Application.ScreenUpdating = False
    Application.Calculation = xlCalculationManual

    ' Antes se usaba B26:... ahora AR11:CE70.
    ' Suponiendo que AO era un marcador, ahora es CE. Si AO26:AO85 era un indicador, ahora es CE11:CE7
0.
    ' Si antes se marcaban para eliminar con AO = True, ahora CE

    Dim i As Long
    For i = 70 To 11 Step -1
        If wsPpal.Cells(i, "CE").value = True Then
            wsPpal.Range(wsPpal.Cells(i, "AR"), wsPpal.Cells(i, "CE")).ClearContents
        End If
    Next i

    ' Reorganizar la numeración en AR (antes B)
    Dim itemCount As Long
    itemCount = 0
    Dim rowIndex As Long
    For rowIndex = 11 To 70
        If wsPpal.Cells(rowIndex, "AS").value <> "" Then
            itemCount = itemCount + 1
            wsPpal.Cells(rowIndex, "AR").value = itemCount
        Else
            wsPpal.Cells(rowIndex, "AR").ClearContents
        End If
    Next rowIndex

    ' Actualizar TextBox6 y TextBox7
    wsPpal.OLEObjects("TextBox6").Object.value = Format(Application.WorksheetFunction.Sum(wsPpal.Range
("BU11:BU70")), "0.00")
    wsPpal.OLEObjects("TextBox7").Object.value = wsPpal.OLEObjects("TextBox6").Object.value
    FormatearDecimalesTextBox "TextBox6", 2
    FormatearDecimalesTextBox "TextBox7", 2

    Application.Calculation = xlCalculationAutomatic
    Application.ScreenUpdating = True

    MsgBox "Los registros seleccionados han sido procesados.", vbInformation
    Exit Sub

ErrorHandler:
    Application.Calculation = xlCalculationAutomatic
    Application.ScreenUpdating = True
    LogError "EliminarSeleccionados", Err.description
    MsgBox "Error en EliminarSeleccionados: " & Err.description, vbCritical
End Sub
```

```
' Función para verificar si un workbook está abierto
Public Function IsWorkbookOpen(ByVal name As String) As Boolean
    Dim wb As Workbook
    On Error Resume Next
    Set wb = Workbooks(Dir(name))
    On Error GoTo 0
    IsWorkbookOpen = Not wb Is Nothing
End Function
```