gambas-extra-functions

Autor: **Martín Belmonte**

Proveedor: **Belmotek**

Versión: **0.1.3**

Componentes

- gb.image
- gb.gui.qt
- gb.form
- gb.db
- gb.desktop.x11
- gb.desktop
- gb.eval
- gb.eval.highlight
- gb.form.editor
- gb.settings
- gb.form.mdi
- gb.form.terminal
- gb.gui.qt.webkit
- gb.markdown
- gb.pcre
- gb.pdf
- gb.report2
- gb.sdl2.audio
- gb.xml

gambas-extra-functions consta de **108** métdos

GEFAbout

Form_Open

```
If Exist("logo.png") Then
  pioLogo.Picture = Picture.Load("logo.png")
Else
  pioLogo.Picture = Null
Endif
Me.Move(FMain.X + 50, FMain.Y + 70)
stxInfo = GEFSys.ProjInfo()
stxComp = Split(stxInfo[5], ":")
txa = New TextLabel(pnlAbout)
txa.Name = "txa-" & "Title"
txa.Text = GEFValidator.Capital(Application.Title)
txa.Font.Grade = 4
txa.Alignment = Align.Center
txa.Height = 35
txa = New TextLabel(pnlAbout)
txa.Name = "txa-" & "Description"
txa.Text = stxInfo[1]
txa.Font.Grade = 0
txa.Alignment = Align.Center
txa.Height = 28
txa = New TextLabel(pnlAbout)
txa.Name = "txa-" & "Version"
txa.Text = Application.Version
txa.Font.Grade = 0
txa.Alignment = Align.Center
txa.Height = 28
txa = New TextLabel(pnlAbout)
txa.Name = "txa-" & "Copy"
txa.Text = "Copyright (c)" & " 2016-" & Year(Now)
txa.Font.Grade = 0
txa.Alignment = Align.Center
txa.Height = 28
txa = New TextLabel(pnlAbout)
txa.Name = "txa-" & "Authors"
txa.Text = stxInfo[2]
txa.Font.Grade = 0
txa.Alignment = Align.Center
txa.Height = 28
tob.Name = "tob-" & "Web"
tob.Text = ("Pagina web") & ": https://github.com/" & String.LCase(stxInfo[3]) &/ String.LCase(Application
tob.Tag = "https://github.com/" & String.LCase(stxInfo[3]) &/ String.LCase(Application.Name)
tob.Font.Grade = 0
tob.Height = 28
tob.Border = False
tob.Tooltip = ("Código fuente")
tob.Name = "tob-" & "Email"
tob.Text = ("Correo de contacto") & ": info@belmotek.net"
tob.Tag = "info@belmotek.net"
tob.Font.Grade = 0
tob.Height = 28
tob.Border = False
tob.Tooltip = ("Enviar un correo al desarrollador")
txa = New TextLabel(pnlAbout)
txa.Name = "txa-" & "Components"
txa.Text = ("Componentes utilizados en este programa") & ":"
txa.Font.Grade = 0
txa.Alignment = Align.Center
txa.Height = 28
pnl = New Panel(pnlAbout)
pnl.Name = "pnl-" & "Components"
pnl.Arrangement = Arrange.Row
pnl.Height = 56
For intComp = 0 To stxComp.Max
  tob.Name = "tob-" & stxComp[intComp]
  tob.Text = stxComp[intComp]
  tob.Tag = "http://gambaswiki.org/wiki/comp/" & stxComp[intComp]
  tob.Tooltip = ("Clic para ver detalles en") & " " & "GambasForge/Wiki"
  tob.Font.Grade = 0
  tob.Height = 28
  tob.Width = 7 * String.Len(stxComp[intComp])
  tob.Border = False
Next
txa = New TextLabel(pnlAbout)
txa.Name = "txa-" & "Warranty"
txa.Text = ("Este programa se entrega sin ningún tipo de garantía")
txa.Font.Grade = 0
```

```
  txa.Alignment = Align.Center
  txa.Height = 56
  tob.Name = "tob-" & "License"
  tob.Text = ("Licencia") & " GPL v3. "
  tob.Tooltip = ("Clic para ver la licencia en") & " gnu.org"
  tob.Tag = "http://www.gnu.org/licenses/old-licenses/gpl-2.0.html"
  tob.Font.Grade = 0
  tob.Height = 28
  tob.Border = False
  Me.Caption = ("Acerca de") & " " & Application.Title
```

Url_Click

```
  tob = Last
  strUrl = tob.Tag
  If InStr(strUrl, "@") > 0 Then
    strType = "email"
    stxAddress.Add(strUrl)
    strSubject = ("Hola desarrolladores de") & " " & Application.Name & " [" & GEFUtility.Timestamp(Now()) &
  Else
    If InStr(strUrl, "http") > 0 Then
      strType = "web"
    Else
      strType = "Unknown"
    Endif
  Endif
  Select strType
    Case "email"
      Desktop.SendMail(stxAddress,,, strSubject)
    Case "web"
      Desktop.Open(strUrl)
  End Select
```

GEFConfig

Form_Open

```
  stxType.Clear
  inxTypeKey.Clear
  stxLang.Clear
  intCounter = 0
  If stxType.Count > 0 Then
    TabPanel1.Count = stxType.Count
    If stxType.Count > 0 Then
      Me.Height = FMain.Height
      Me.Width = FMain.Width
      Me.X = FMain.X
      Me.Y = FMain.Y
    Endif
    For intTab = 0 To stxType.Max
      TabPanel1[intTab].Text = stxType[intTab]
      With scw
        .Name = "scw" & CStr(intTab)
        .Arrangement = Arrange.Row
        .Expand = True
        .Width = TabPanel1.Width
        .Border = True
        For intLang = 0 To GEFStarter.stxLanguage.Max
        If GEFStarter.stxLanguage[intLang][7] = stxType[intTab] Then
          With chk
            .Name = "chk" & CStr(intLang)
            .Tooltip = GEFStarter.stxLanguage[intLang][6]
            strText = GEFStarter.stxLanguage[intLang][2]
            .Text = strText
            .Tag = GEFStarter.stxLanguage[intLang][0]
            .Width = TabPanel1.Width / 2
            .Height = 28
            Select GEFStarter.stxLanguage[intLang][9]
              Case "False"
                .Value = 0
              Case "True"
                .Value = -1
            End Select
          End With
        Endif
      Next
    End With
  Next
  TabPanel1.Index = 0
Endif
MakeConfigControls()
```

cmdSaveConf

```
For Each obj In scwConfig.Children
  If Object.Type(obj) = "Panel" Then
    Select Object.Type(obj.Children[1]) ' El primero "0" es el label, el segundo "1" es el combo o text
      Case "TextBox", "ComboBox", "ValueBox"
        GEFStarter.stxProgVal[obj.Children[1].Tag] = obj.Children[1].Text
        Print Format(obj.Children[1].Tag, "0#") & ":" & Object.Type(obj.Children[1]) & ": " & obj.Children
      Case "ValueBox", "ColorButton", "SpinBox"
        GEFStarter.stxProgVal[obj.Children[1].Tag] = CStr(obj.Children[1].Value)
        Print Format(obj.Children[1].Tag, "0#") & ":" & Object.Type(obj.Children[1]) & ": " & CStr(obj.Chil
      Case "SwitchButton"
        Select obj.Children[1].Value
          Case True
            GEFStarter.stxProgVal[obj.Children[1].Tag] = "True"
          Case Else
            GEFStarter.stxProgVal[obj.Children[1].Tag] = ""
        End Select
        Print Format(obj.Children[1].Tag, "0#") & ":" & Object.Type(obj.Children[1]) & ": " & obj.Children
    End Select
  Endif
Next
If GEFStarter.Terminator() = 1 Then
  Print ("Configuración guardada con exito")
Endif
Me.Close
```

tobClear_Click

```
For intTab = 0 To TabPanel1.Count - 1
  For Each obj In TabPanel1[intTab].Children
    If Object.Type(obj) = "CheckBox" Then
      obj.Value = 0
    Endif
  Next
Next
```

MakeConfigControls

Esta funcion crea los controles para editar la configuración del programa

```
  intState = 0
  scwConfig.Children.Clear
  stxParam.Add(["0", ("Motor de bases de datos"), "cmo", "postgresql:mysql:sqlite3:odbc", GEFStarter.stxProgV
  stxParam.Add(["1", ("Ruta"), "txo", "", GEFStarter.stxProgVal[1]])
  stxParam.Add(["2", ("Nombre"), "txo", "", GEFStarter.stxProgVal[2]])
  stxParam.Add(["3", ("Puerto"), "txo", "", GEFStarter.stxProgVal[3]])
  stxParam.Add(["4", ("Usuario"), "txo", "", GEFStarter.stxProgVal[4]])
  stxParam.Add(["5", ("Contraseña"), "txow", "", GEFStarter.stxProgVal[5]])
  stxParam.Add(["6", ("Archivo de bitácora") & " 1", "cmo", "pandoc:wkhtmltopdf", GEFStarter.stxProgVal[6]])
  stxParam.Add(["7", ("Verificar dependencias al inicio"), "swb", "", GEFStarter.stxProgVal[7]])
  stxParam.Add(["8", ("Líneas vacías"), "swb", "", GEFStarter.stxProgVal[8]])
  stxParam.Add(["9", ("Entorno de ejecucion") & " 9", "txor", "", GEFStarter.stxProgVal[9]])
  stxParam.Add(["10", ("Método del informe") & " 1", "cmo", "pandoc:wkhtmltopdf", GEFStarter.stxProgVal[10]]
  stxParam.Add(["11", ("Método del informe") & " 2", "cmo", "pandoc:wkhtmltopdf", GEFStarter.stxProgVal[11]]
  stxParam.Add(["12", ("Método del informe") & " 3", "cmo", "pandoc:wkhtmltopdf", GEFStarter.stxProgVal[12]]
  stxParam.Add(["13", ("Método del informe") & " 4", "cmo", "pandoc:wkhtmltopdf", GEFStarter.stxProgVal[13]]
  stxParam.Add(["14", ("Método del informe") & " 5", "cmo", "pandoc:wkhtmltopdf", GEFStarter.stxProgVal[14]]
  stxParam.Add(["15", ("Tema del editor"), "cmo", "gef:amber:blues:gambas:obsidian:quest:ruby:visual:amethys
  stxParam.Add(["16", ("Mostrar") & " GEF" & " 16", "swb", "", GEFStarter.stxProgVal[16]])
  stxParam.Add(["17", ("Nombre alternativo"), "txo", "", GEFStarter.stxProgVal[17]])
  stxParam.Add(["18", ("Libre") & " 18", "txo", "", GEFStarter.stxProgVal[18]])
  stxParam.Add(["19", ("Libre") & " 18", "txo", "", GEFStarter.stxProgVal[18]])
  stxParam.Add(["20", ("Nombre de la fuente"), "txo", "", GEFStarter.stxProgVal[20]])
  stxParam.Add(["21", ("Tamaño de la fuente"), "txo", "", GEFStarter.stxProgVal[21]])
  stxParam.Add(["22", ("Color del fondo"), "cob", "", GEFStarter.stxProgVal[22]])
  stxParam.Add(["23", ("Color del frente"), "cob", "", GEFStarter.stxProgVal[23]])
  stxParam.Add(["24", ("Orientación"), "txo", "", GEFStarter.stxProgVal[24]])
  stxParam.Add(["25", ("Tema"), "txo", "", GEFStarter.stxProgVal[25]])
  stxParam.Add(["26", ("Barra de menú"), "txo", "", GEFStarter.stxProgVal[26]])
  stxParam.Add(["27", ("Opacidad"), "sio", "100", GEFStarter.stxProgVal[27]])
  stxParam.Add(["28", ("Transparencia"), "sio", "0", GEFStarter.stxProgVal[28]])
  stxParam.Add(["29", ("Nueva ventana"), "txo", "", GEFStarter.stxProgVal[29]])
  stxParam.Add(["30", ("Cerrar ventana"), "txo", "", GEFStarter.stxProgVal[30]])
  stxParam.Add(["31", ("Nueva pestaña"), "txo", "", GEFStarter.stxProgVal[31]])
  stxParam.Add(["32", ("Cerrar pestaña"), "txo", "", GEFStarter.stxProgVal[32]])
  stxParam.Add(["33", ("Copiar"), "txo", "", GEFStarter.stxProgVal[33]])
  stxParam.Add(["34", ("Pegar"), "txo", "", GEFStarter.stxProgVal[34]])
  For intCfg = 0 To stxParam.Max
    intWidt = 6 * String.Len(stxParam[intCfg][1]) + 21
    pnl = New Panel(scwConfig)
    pnl.Name = "pnl-" & CStr(intCfg)
    pnl.Width = intWidt
    pnl.Height = 49
    pnl.Arrangement = Arrange.Vertical
    pnl.Border = Border.Plain
    lbl = New Label(pnl)
    lbl.Name = "lbl-" & CStr(intCfg)
    lbl.Text = " " & stxParam[intCfg][1]
    lbl.Width = intWidt
    lbl.Height = 21
    lbl.Expand = True
    Select stxParam[intCfg][2]
      Case "txo", "txow", "txor"
      txo.Name = "txo-" & CStr(intCfg)
      txo.Tag = intCfg
      txo.Text = stxParam[intCfg][4]
      Select stxParam[intCfg][2]
        Case "txow"
          txo.Password = True
        Case "txor"
          txo.ReadOnly = True
      End Select
      txo.Width = intWidt
      txo.Height = 28
      txo.Expand = True
    Case "cmo"
      cmo.Name = "cmo-" & CStr(intCfg)
      cmo.Text = stxParam[intCfg][4]
      cmo.Tag = intCfg
      cmo.Width = intWidt + 14
      cmo.Height = 28
      cmo.ReadOnly = True
      cmo.Expand = True
      stxTmp = Split(stxParam[intCfg][3], ":")
      For intM = 0 To stxTmp.Max
        cmo.Add(stxTmp[intM])
      Next
      If stxTmp.Find(stxParam[intCfg][4]) > -1 Then
        cmo.Text = stxParam[intCfg][4]
      Endif
    Case "sio"
      sio.Name = "sio-" & CStr(intCfg)
      sio.Tag = intCfg
      sio.Value = CInt(stxParam[intCfg][4])
      sio.MaxValue = 300
      sio.MinValue = 60
      sio.Width = intWidt
      sio.Height = 28
      sio.Expand = True
    Case "swb"
      swb.Name = "swb-" & CStr(intCfg)
      swb.Tag = intCfg
      swb.Value = stxParam[intCfg][4]
      swb.Width = intWidt
      swb.Height = 28
      swb.Expand = True
    Case "cob" ' ColorButton
      cob.Name = "cob-" & CStr(intCfg)
      cob.Tag = intCfg
      cob.Value = CInt(stxParam[intCfg][4])
      cob.Width = intWidt
      cob.Height = 28
      cob.Expand = True
    Case "vao"
      vao.Name = "vao-" & CStr(intCfg)
      vao.Tag = intCfg
      vao.Value = CInt(stxParam[intCfg][4])
      vao.Width = intWidt
      vao.Height = 28
      vao.Expand = True
  End Select
Next
intState = 1
```

```
  Return intState
```

WriteConfig

```
  stxFields.Clear
  For Each objp In scwConfig.Children
    If Object.Type(objp) = "Panel" Then
      For Each objx In objp.Children
        Select Object.Type(objx)
          Case "TextBox", "ComboBox"
            stxValues.Add(objx.Tag & ":" & objx.Text)
        End Select
      Next
    Endif
  Next
  Return intState
```

GEFDataEdit

btnOK_Click

```
  stxFields.Clear
  For Each objp In pnlData.Children
    If Object.Type(objp) = "Panel" Then
      For Each objx In objp.Children
        Select Object.Type(objx)
          Case "TextBox", "ComboBox"
            stxValues.Add(objx.Tag & "\t" & objx.Text)
        End Select
      Next
    Endif
  Next
  If intKey Then ' Editando registro existente
    GEFData.RecordEdit(conData, strTab, stxValues)
  Else ' Nuevo registro
    GEFData.RecordNew(conData, strTab, GEFStarter.stxTableFields, stxValues)
  Endif
  FMain.UpdateGrid()
  Me.Close
```

MakeControls

strTable As String/Optional intKey As Integer

```
  intState = 0
  pnlData.Children.Clear
  pnlData.Arrangement = 3
  If IsNull(intKey) = False Then
    For intFld = 0 To GEFStarter.stxTableFields.Max
      If GEFStarter.stxTableFields[intFld][0] = strTable Then
        If GEFStarter.stxTableFields[intFld][5] = "YK" Then
          strSQLEdit = "select * from " & GEFStarter.stxTableFields[intFld][0]
          strSQLEdit &= " where " & GEFStarter.stxTableFields[intFld][1]
          strSQLEdit &= "='" & CStr(intKey) & "'"
          resEdit = conData.Exec(strSQLEdit)
        Endif
      Endif
    Next
  For intFld = 0 To GEFStarter.stxTableFields.Max
    If GEFStarter.stxTableFields[intFld][0] = strTable Then
      Select GEFStarter.stxTableFields[intFld][5] ' Si el campo es Primary Key o no
        Case "YK"
          bolReadonly = True
          intWidt = 60
        Case "NK"
          bolReadonly = False
          intWidt = 120
      End Select
      pnl = New Panel(pnlData)
      pnl.Name = "pnl-" & GEFStarter.stxTableFields[intFld][1] 'strFieldName
      pnl.Width = intWidt
      pnl.Height = 58
      pnl.Arrangement = 2
      pnl.Padding = 2
      lbl = New Label(pnl)
      lbl.Name = "lbl-" & GEFStarter.stxTableFields[intFld][1] 'strFieldName
      lbl.Text = GEFStarter.stxTableFields[intFld][10] 'strFieldName
      lbl.Width = intWidt
      lbl.Height = 28
      Select GEFStarter.stxTableFields[intFld][7]
        Case ""
          txo = New TextBox(pnl)
          txo.Name = "txo-" & GEFStarter.stxTableFields[intFld][1] 'strFieldName
          txo.Tag = GEFStarter.stxTableFields[intFld][1] 'strFieldName
          txo.Width = intWidt
          txo.Height = 28
          txo.ReadOnly = bolReadonly
          If resEdit.Available Then
            txo.Text = resEdit[GEFStarter.stxTableFields[intFld][1]]
          Endif
        Case Else
          strComboSql = "select " & GEFStarter.stxTableFields[intFld][8] & ", "
          strComboSql &= GEFStarter.stxTableFields[intFld][9] & " from "
          strComboSql &= GEFStarter.stxTableFields[intFld][7] & " order by " & GEFStarter.stxTableFields[i
          resCombo = conData.Exec(strComboSql)
          cmo = New ComboBox(pnl)
          cmo.Name = "cmo-" & GEFStarter.stxTableFields[intFld][1]
          cmo.Tag = GEFStarter.stxTableFields[intFld][1]
          cmo.Width = pnl.Width
          cmo.Height = 28
          While resCombo.Available
            cmo.Add(resCombo[GEFStarter.stxTableFields[intFld][9]])
            If resEdit.Available Then
              If resCombo[GEFStarter.stxTableFields[intFld][8]] = resEdit[GEFStarter.stxTableFields[intFld
                cmo.Text = resCombo[GEFStarter.stxTableFields[intFld][9]]
              Endif
            Endif
            resCombo.MoveNext
          Wend
      End Select
    Endif
  Next
Endif
intState = 1
Return intState
```

GEFPrint

tobPrint_Click

```
If Exist("/tmp/pdf") Then
  prsRM = Shell "rm -r -f /tmp/pdf"
  While
    prsRM.State = prsRM.Running
    Wait 0.1
  Wend
Endif
prsMd = Shell "mkdir -p /tmp/pdf"
While
  prsMd.State = prsMd.Running
  Wait 0.1
Wend
File.Save("/tmp/pdf/help.html", strHtml)
Copy "logo.png" To "/tmp/pdf/logo.png"
Wait 0.5
strPDF = GEFBatch.HTMLPDF("/tmp/pdf/help.html", "pandoc")
If Exist(strPDF) Then
  Desktop.Open(strPDF)
Endif
```

GEFBatch

FileDelibery

Funcion que reparte el archivo a una funcion coincidente con el nombre de la herramienta "Tool" en el segundo parametro

strTool As String/strPath As String/Optional intPage As Integer/Optional strMod As String

```
str = "0"
Select strTool
  Case "File-Rename" 'Renombrador de archivos
  Case "File-Check" 'Verificador de nombre de archivo
  Case "Image-Convert" ' Convesion de formatos de imagenes
    str = ImageConvert(strPath, strMod)
  Case "JPEG>GIFEmail" 'Crea un archivo GIF y lo adjunta en un nuevo email.
    str = JPEGGIFEmail(strPath)
  Case "JPEG>OCR-Text" 'Reconocimiento optico de caracteres
    str = JPEGOCRText(strPath, strMod)
  Case "JPEG>Copy-Reduced-Color" 'Crea una copia reducida en la misma ubicación que la original
    str = JPEGCopyRC(strPath, strMod)
  Case "JPEG>Copy-Reduced-Gray" 'Crea una copia reducida en escala de grices en la misma ubicación que la o
    str = JPEGCopyRG(strPath, strMod)
  Case "JPEG>Copy-Gray" 'Crea una copia en escala de grices en la misma ubicación que la original
  Case "JPEG>Square" 'Crea una copia reducida de proporción cuadrada en la misma ubicación que la original
  Case "JPEG>PDF" 'Crea un archivo PDF con todas las imagenes que se le pase.
    str = JPEGPDF(strPath, intPage)
  Case "PNG>Copy-Reduced-Color" 'Crea una copia reducida en la misma ubicación que la original
  Case "PNG>JPEG-Copy-Reduced-Color" 'Crea una copia reducida en la misma ubicación que la original
    str = PNGJPEGReduced(strPath)
  Case "TIF>JPEG" 'Crea una copia de una imagen TIFF en otra JPEG
    str = TIFJPEG(strPath)
  Case "PNG>OCR-Text" 'Reconocimiento optico de caracteres
    str = PNGOCRText(strPath, strMod)
  Case "ODT-Thumbnail" 'Extractor de miniatura del documento
  Case "ODT>EPUB" 'Convertir documentos ODT en documentos EPUB
  Case "PDF>Decrypt" 'Crea una copia desencriptada del archivo.
```

```
      str = PDFDecrypt(strPath)
    Case "PDF>Image" 'Extractor de paginas en formato de imagen jpeg una imagen por página.
      str = PDFImage(strPath, intPage, strMod)
    Case "PDF>PDF-R90" 'Rota las paginas 90 grados.
      str = PDFR90(strPath)
    Case "PDF>Pages" 'Extractor de paginas en formato pdf
    Case "PDF>OCR-Text" 'Extractor de paginas en formato de imagen TIF y reconocimiento optico de caracteres
      str = PDFOCRText(strPath, intPage, strMod)
    Case "TXT>OGG" 'Crea un archivo de sonido a partir de uno de texto
      str = TXT2OGG(strPath, strMod)
    Case "LATEXPDF" 'Crea un archivo de sonido a partir de uno de texto
      str = LATEXPDF(strPath, strMod)
    Case "HTML>PDF" 'Crea un archivo de sonido a partir de uno de texto
      str = HTMLPDF(strPath, strMod)
    Case "Play-Sound" 'Reproductor de archivos de sonido.
      str = PlaySound(strPath)
    Case "Video>Frame" 'Extractor de fotograma de un video
    Case "Audio-Extractor" 'Extractor del audio de un video
      str = AudioExtractor(strPath)
    Case "Video-Extractor" 'Extractor del audio de un video
      str = VideoExtractor(strPath)
    Case "Video-Mixer" 'Crea un video copia pero sin audio y le agrega una pista de audio que se llama igual
      str = VideoMixer(strPath)
    Case "Media>Arrange" 'Mueve las fotografias organizandolas según su fecha de captura.
      str = MediaArrange(strPath, strMod)
    Case "JET>SQL" 'Crea los archivos SQL necesarios para reconstruir una base de datos Jet en SQLite, Postg
  End Select
  Return str
```

ImageConvert

Convierte una imágenes JPEG a otra formato PNG.

strPath As String/strFormat As String

```
  If GEFStarter.strState = "running" Then
    strRenPath = File.Dir(strPath) &/ String.LCase(File.BaseName(strPath)) & "." & strFormat
    strCommand2 = "convert -quality 80 '" & strPath & "' '" & strRenPath & "'"
    prs2 = Shell strCommand2
    While
      prs2.State = prs2.Running
      Wait 0.05
    Wend
  Endif
  If Exist(strRenPath) Then
    If Stat(strRenPath).Type = gb.File Then
      Return strRenPath
    Else
      Return "0"
    Endif
  Endif
```

WAV2OGG

Esta función convierte un archivo WAV un UNO OGG. Como parametro de entrada requiere una ruta del archivo WAV. El archivo OGG sera creado en la misma ubicación que el WAV y se devolvera la ruta de este en caso que todo haya ido bien.

strPath As String

```
  strDir = File.Dir(strPath)
  strDirBase = File.BaseName(strPath)
  strAudioFile = strDir &/ strDirBase & ".ogg"
  strCommand = "avconv -i '" & strPath & "' -vn "
  strCommand &= " '" & strAudioFile & "'"
  Print strCommand
  prsWAVE = Shell strCommand For Read As "ExtAudio"
  While
    prsWAVE.State = prsWAVE.Running
    Wait 2
  Wend
  If Exist(strAudioFile) Then
    Print "Audio file exist: " & strAudioFile
    If Stat(strAudioFile).Type = gb.File Then
      Return strAudioFile
    Else
      Return "0"
    Endif
  Else
    Return "0"
  Endif
```

LATEXPDF

Convierte un arcivo tex en pdf usando el prgrama pdflatex

strPath As String/strMod As String

```
  strDir = File.Dir(strPath)
  strDirBase = File.BaseName(strPath)
  strOutputFile = strDir &/ strDirBase & ".pdf"
  Select strMod
    Case "pdflatex"
      strCommand = "pdflatex -output-format=pdf '" & strPath & "'"
  End Select
  prsAE = Shell strCommand ' For Read As "HmlToPdf"
  While
    prsAE.State = prsAE.Running
    Wait 0.5
  Wend
  Wait 0.5
  If Stat(strOutputFile).Type = gb.File Then
    Return strOutputFile
  Else
    Return "0"
  Endif
```

AudioExtractor

Esta funcion extrae solamente la pista de audio de un archivo de video. Devuelve la rutade destino del archivo extraido

strPath As String

```
  strDir = File.Dir(strPath)
  strDirBase = File.BaseName(strPath)
  strAudioFile = strDir &/ strDirBase & ".ogg"
  strCommand = "avconv -i '" & strPath & "' -vn "
  strCommand &= "-af volume=7dB:precision=fixed "
  strCommand &= "-acodec libvorbis '" & strAudioFile & "'"
  Print strCommand
  prsAE = Shell strCommand For Read As "ExtAudio"
  While
    prsAE.State = prsAE.Running
    Wait 2
  Wend
  If Exist(strAudioFile) Then
    Print "Audio file exist: " & strAudioFile
    If Stat(strAudioFile).Type = gb.File Then
      Return strAudioFile
    Else
      Return "0"
    Endif
  Else
    Return "0"
  Endif
```

VideoExtractor

Esta funcion extrae solamante la pista de video de un archivo de video. Devuelve la rutade destino del archivo extraido

strPath As String

```
  strExt = File.Ext(strPath)
  strDir = File.Dir(strPath)
  strDirBase = File.BaseName(strPath)
  strOutputFile = strDir &/ strDirBase & "-muted." & strExt
  strCommand = "avconv -i '" & strPath & "' -an "
  strCommand &= "'" & strOutputFile & "'"
  Print strCommand
  prsVE = Shell strCommand For Read As "ExtVideo"
  While
    prsVE.State = prsVE.Running
    Wait 1
  Wend
  If Exist(strOutputFile) Then
    If Stat(strOutputFile).Type = gb.File Then
      Return strOutputFile
    Endif
  Endif
```

VideoMixer

Devuelve la ruta de destino del archivo remasterzado creado. avconv -i output.mkv -an -i konvertilo-02-media-arrange.ogg output-unido.mkv

strPath As String

```
  strExt = File.Ext(strPath)
  strDir = File.Dir(strPath)
  strDirBase = File.BaseName(strPath)
  strAudioFile = strDir &/ strDirBase & ".ogg"
  strInputFile = strDir &/ strDirBase & "-muted." & strExt
  strOutputFile = strDir &/ strDirBase & "-remaster." & strExt
  If Exist(strOutputFile) Then
    Kill strOutputFile
  Endif
  strCommand = "avconv -i '" & strInputFile & "' "
  strCommand &= "-i '" & strAudioFile & "' "
  strCommand &= "'" & strOutputFile & "'"
  Print strCommand
  prsVM = Shell strCommand For Read As "MixVideo"
```

```
While
  prsVM.State = prsVM.Running
  Wait 1
Wend
If Exist(strOutputFile) Then
  If Stat(strOutputFile).Type = gb.File Then
    Return strOutputFile
  Else
    Return "0"
  Endif
  Return "0"
Endif
```

MediaArrange

Devuelve la rutade destino del archivo

strPath As String/strLog As String

```
strCRC = GEFUtility.CRC32(strPath)
strExt = String.UCase(File.Ext(strPath)) ' Extensión del archivo en mayúsculas
Select strLog
  Case "T"
    strAcct = "mv"
  Case "F", ""
    strAcct = "cp"
End Select
If GEFStarter.strState = "running" Then
  stxExif = GEFUtility.ExifRaw(strPath)
  For intRep = 0 To stxExif.Max
    intKey = GEFStarter.stxFileExifUgly.Find(Split(stxExif[intRep], "\t")[0])
    If intKey > -1 Then
      stxExif[intRep] = GEFStarter.stxFileExifGood[intKey] & "\t" & Split(stxExif[intRep], "\t")[1]
    Endif
  Next
  For intEx = 0 To stxExif.Max
    Select Split(stxExif[intEx], "\t")[0]
      Case "MediaCreateDate", "DateTimeOriginal"
        strTimeStamp = Split(stxExif[intEx], "\t")[1]
        strTimeStamp = GEFValidator.OnlyNumbers(strTimeStamp)
        strTimeStamp = String.Mid(strTimeStamp, 1, 14)
      Case "Make" ' Fabricante de la cámara
        strMnfr = Split(stxExif[intEx], "\t")[1]
        strMnfr = String.UCase(strMnfr)
      Case "Model" ' Modelo de cámara sin caracteres extraños.
        strModel = Split(stxExif[intEx], "\t")[1]
        strModel = GEFValidator.NoSymbols(strModel)
        strModel = String.UCase(strModel)
      Case "ImageSize"
        strImageSize = String.UCase(Split(stxExif[intEx], "\t")[1])
      Case "FileType"
        strExt = Split(stxExif[intEx], "\t")[1]
    End Select
  Next
  Select strTimeStamp
    Case "00000000000000", ""
      For int2 = 0 To stxExif.Max
        Select Split(stxExif[int2], "\t")[0]
          Case "FileModifyDate"
            strTimeStamp = Split(stxExif[int2], "\t")[1]
            strTimeStamp = GEFValidator.OnlyNumbers(strTimeStamp)
            strTimeStamp = String.Mid(strTimeStamp, 1, 14)
        End Select
      Next
  End Select
  strYear = String.Mid(strTimeStamp, 1, 4)
  strMonth = String.Mid(strTimeStamp, 5, 2)
  strDay = String.Mid(strTimeStamp, 7, 2)
  strTime = String.Mid(strTimeStamp, 9, 6)
  If strMnfr = "" Then
    strMnfr = "MNFR"
  Endif
  If strModel = "" Then
    strModel = "MOD"
  Endif
  Select strExt
    Case "JPG", "JPEG"
      strDirBase = User.Home &/ strYear & "F" &/ strMonth
      strDirDup = User.Home &/ strYear & "F-DUP" &/ strMonth
      strFileName = strTimeStamp & "-" & strMnfr & "-" & strModel & "-" & strCRC & "." & strExt
    Case "AVI", "MOV", "MTS", "M2TS", "MP4", "WEBM", "OGV"
      strDirBase = User.Home &/ strYear & "V"
      strDirDup = User.Home &/ strYear & "V-DUP"
      If strImageSize <> "" Then
        strFileName = strTimeStamp & "-" & strImageSize & "-" & strCRC & "." & strExt
      Else
        strFileName = strTimeStamp & "-" & strCRC & "." & strExt
      Endif
    Case Else
      strDirBase = User.Home &/ strYear & "X"
      strDirDup = User.Home &/ strYear & "X-DUP"
      strFileName = strTimeStamp & "-" & strCRC & "." & strExt
  End Select
  If String.Len(strTimeStamp) = 14 Then
    If String.Len(strCRC) = 8 Then
      strRenPath = strDirBase &/ strFileName
      If Exist(strRenPath) = False Then
        Print "Bucle 1"
        While
          Exist(strDirBase) = False
          Shell "mkdir -p " & strDirBase
          Wait 0.05
        Wend
        prs = Shell strAcct & " '" & strPath & "' " & strRenPath
        While
          prs.State = prs.Running
          Wait 0.05
        Wend
      Else
        While
          Exist(strDirDup) = False
          Shell "mkdir -p " & strDirDup
          Wait 0.05
        Wend
        strRenPath = strDirDup &/ strFileName
        Wait 0.01
        prs = Shell strAcct & " '" & strPath & "' " & strRenPath
        While
          prs.State = prs.Running
          Wait 0.05
```

```
          Wend
        Endif
      Endif
    Endif
  Endif
  If Exist(strRenPath) Then
    If Stat(strRenPath).Type = gb.File Then
      Return strRenPath
    Else
      Return "0"
    Endif
  Endif
```

TIFJPEG

Convierte una imágenes TIF a JPEG.

strPath As String

```
  If GEFStarter.strState = "running" Then
    strRenPath = File.Dir(strPath) &/ String.LCase(File.BaseName(strPath)) & ".jpeg"
    strCommand2 = "convert -quality 80 '" & strPath & "' '" & strRenPath & "'"
    prs2 = Shell strCommand2
    While
      prs2.State = prs2.Running
      Wait 0.05
    Wend
  Endif
  If Exist(strRenPath) Then
    If Stat(strRenPath).Type = gb.File Then
      Return strRenPath
    Else
      Return "0"
    Endif
  Endif
```

JPEGPDF

Convierte todas las imágenes JPEG de la lista en un archivo PDF. Si el archivo PDF es creado satisfactoriamente se devuelve la ruta del mismo.

strPath As String

```
  If GEFStarter.strState = "running" Then
    strRenPath = File.Dir(strPath) &/ String.LCase(File.BaseName(strPath)) & ".pdf"
    strCommand2 = "convert -quality 30 '" & strPath & "' '" & strRenPath & "'"
    prs2 = Shell strCommand2
    While
      prs2.State = prs2.Running
      Wait 0.05
    Wend
  Endif
  If Exist(strRenPath) Then
    If Stat(strRenPath).Type = gb.File Then
      Return strRenPath
    Else
      Return "0"
    Endif
  Endif
```

JPEGCopyRG

Devuelve la ruta al archivo jpeg creado

strPath As String/Optional strCompress As String

```
If GEFStarter.strState = "running" Then
  If Exist(strRenPath) = False Then
    If strCompress = "" Then
      strCompress = "75"
    Else
      If IsNumber(strCompress) = True Then
        If CInt(strCompress) < 1 Or CInt(strCompress) > 100 Then
          strCompress = "75"
        Endif
      Endif
      Wait 0.01
      strRenPath = File.Dir(strPath) &/ String.LCase(File.BaseName(strPath)) & "-" & strCompress & "-rg.jp
      strCommand = "convert -type Grayscale -depth 8 -normalize -level 0%,75%,0.8"
      strCommand &= " -compress jpeg -quality " & strCompress & " -enhance"
      strCommand &= " '" & strPath & "' '" & strRenPath & "'"
      Print strCommand
      prs = Shell strCommand
      While
        prs.State = prs.Running
        Wait 0.05
      Wend
    Endif
  Endif
  If Exist(strRenPath) Then
    If Stat(strRenPath).Type = gb.File Then
      Return strRenPath
    Else
      Return "0"
    Endif
  Endif
Endif
```

PDFDecrypt

Devuelve la ruta al archivo desencriptado

strPath As String

```
strName = File.Dir(strPath) &/ File.BaseName(strPath) & "-dcr.pdf"
strCommand = "gs -q -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -sOutputFile='"
strCommand &= strName & "' -c .setpdfwrite -f '" & strPath & "'"
prsCommand = Shell strCommand
While prsCommand.State = prsCommand.Running
  Wait 0.05
Wend
If Exist(strName) Then
  If Stat(strName).Type = gb.File Then
    Return strName
  Else
    Return "0"
  Endif
Endif
```

PNGOCRText

Devuelve la ruta al archivo .txt con el texto extraido o “0” si no hay texto.

strPath As String/Optional strLang As String

```
strFileOutput = File.Dir(strPath) &/ File.BaseName(strPath) & ".txt"
If GEFStarter.stxOCRLang.Find(strLang) > -1 Then
  If GEFStarter.strState = "running" Then
    If Exist(strPath) Then
      If Stat(strPath).Type = gb.File Then
        strTxtTemp = ""
        strCommand = "tesseract '" & strPath & "' stdout -l " & strLang & " 2>&1"
        Shell strCommand To strTxtTemp
        If strTxtTemp <> "" Then
          File.Save(strFileOutput, strTxtTemp)
        Endif
        Wait 0.05
      Endif
    Endif
  Endif
Endif
If strTxtTemp <> "" Then
  Return strFileOutput
Else
  Return "0"
Endif
```

PDFOCRText

Devuelve el texto extraido de la pagina. Como parametros de entrada requiere la ruta del archivo PDF la página y el idioma.

strPath As String/intPage As Integer/Optional strLang As String

```
strPageText = ""
strFilePNG = PDFImage(strPath, intPage, "png")
If Exist(strFilePNG) Then
  strPageText = PNGOCRText(strFilePNG, strLang)
Endif
Return strPageText
```

PDFR90

Rota las paginas 90 grados

strPath As String

```
strName = File.Dir(strPath) &/ File.BaseName(strPath) & "-r90.pdf"
strCommand = "pdftk A=" & strPath & " cat A1-endeast output " & strName
prsCommand = Shell strCommand
While prsCommand.State = prsCommand.Running
  Wait 0.05
Wend
If Exist(strName) Then
  If Stat(strName).Type = gb.File Then
    Return strName
  Else
    Return "0"
  Endif
Endif
```

GEFData

DBTemplate

Crea una plantilla de la base de datos que se le pasa como parametros.

stxDB As String[]

```
conCreate.Type = stxDB[0]
conCreate.Host = stxDB[1]
conCreate.Name = stxDB[2]
conCreate.Open
strTemplate = conCreate.GetTemplate()
conCreate.Close
Return strTemplate
```

DBSqlite

Inicia una base de datos o la crea y la inicia. Devuelve una conexion y como parametro de entrada requiere una matriz con los parametros de la base. Si la base de datos no existe, entonces crea una y la inicia. ' Si la base de datos si existe, entonces puede hacer dos cosas, iniciarla o crear una copia de respaldo y crear una base nueva. stxDB contiene los paramentros de la base. 0 - DBHost. 1 - DBName. 2 - DBPath

stxDB As String[]/Optional strMod As String

```
If conCreate.Opened Then
  conCreate.Close
Endif
strTimeStamp = GEFUtility.Timestamp(Now())
strDBZip = File.Dir(stxDB[3]) &/ File.BaseName(stxDB[3]) & "-" & strTimeStamp & ".zip"
conCreate.Type = stxDB[0]
conCreate.Host = stxDB[1]
conCreate.Name = ""
conCreate.Open
Select strMod
  Case "reset"
    If Exist(stxDB[3]) = True Then
      Shell "zip -j " & strDBZip & " " & stxDB[3]
      Wait 0.5
      Kill stxDB[3]
    Endif
End Select
Wait 0.2
Select strMod
  Case "empty"
    If Not conCreate.Databases.Exist(stxDB[2]) Then
      conCreate.Databases.Add(stxDB[2])
    Endif
  Case "new"
    If Not conCreate.Databases.Exist(stxDB[2]) Then
      conCreate.Databases.Add(stxDB[2])
    Endif
    If Exist("db.template") Then
      strSQLCreate = File.Load("db.template")
      conCreate.ApplyTemplate(strSQLCreate)
    Else
      Message.Warning(("No existe el archivo") & " " & "new.sql" & gb.NewLine & ("La base de datos no sera
    Endif
End Select
If conCreate.Opened Then
  conCreate.Close
Endif
Return conCreate
```

RecordKey

Devuelve el nombre del campo clave de la tabla. connDB

strTable As String/stxDBFields As String[][]

```
Return: Connection es laq conxión a la base de  datos. strTable
```

RecordNewRef

Inserta un registro nuevo en la base de datos, ctnVal es una coleccion opcional del pares de campo:valor.

connDB As Connection/strTable As String/stxDBFields As String[][]/stxValues As String[]

```
  If stxValues.Count > 0 Then
    For intVal = 0 To stxValues.Max
      stxTag.Add(Split(stxValues[intVal], "\t")[0])
      stxVal.Add(Split(stxValues[intVal], "\t")[1])
    Next
  Endif
  For int = 0 To stxDBFields.Max
    If stxTables.Find(stxDBFields[int][0]) = -1 Then
      stxTables.Add(stxDBFields[int][0])
    Endif
  Next
  resIns = connDB.Create(strTable)
  For intField = 0 To stxDBFields.Max
    If stxDBFields[intField][0] = strTable Then
      If stxDBFields[intField][6] <> "YA" Then
        strTag = stxDBFields[intField][1]
        intVal = stxTag.Find(strTag)
        If intVal > -1 Then
          strVal = stxVal[intVal]
        Else
          strVal = ""
        Endif
        Select stxDBFields[intField][7]
          Case ""
            resIns[strTag] = strVal
        End Select
      Endif
    Endif
  Next
  resIns.Update
```

GetForeignKey

Devuelve la clave del registro referenciado en otra tabla

strValue As String/conRef As Connection/strTable As String/strFieldKey As String/strFieldName As String

```
  intKey = -1
  intCounter = 0
  Repeat
    stxForeignKey.Clear
    stxForeignName.Clear
    strSQLForeign = "select " & strFieldKey & ", " & strFieldName & " from " & strTable
    resFeoreign = conRef.Exec(strSQLForeign)
    While resFeoreign.Available
      stxForeignKey.Add(resFeoreign[strFieldKey])
      stxForeignName.Add(resFeoreign[strFieldName])
      If strValue = resFeoreign[strFieldName] Then
        intKey = resFeoreign[strFieldKey]
        Break
      Endif
      resFeoreign.MoveNext
    Wend
    If intKey = -1 Then ' Quiere decir que ese nombre no esta en la tabla de referencia, entonces hay que ins
      resInsert = conRef.Create(strTable)
      resInsert[strFieldName] = strValue
      resInsert.update
    Endif
  Until intKey <> -1
  Return intKey
```

Chek4SQLscript

Devuelve un texto apto para consulatas SQL, quita los saltos del línea y los caracteres no compatibles con sentencias SQL.

strInput As String

```
  strOutput = Replace(strInput, "\n", "")
  strOutput = Replace(strOutput, "\r", "")
  strOutput = Replace(strOutput, "\t", " ")
  strOutput = Replace(strOutput, "\x00", "")
  strOutput = Replace(strOutput, Chr(96), Chr(39)) ' ' > .
  'strOutput = Replace(strOutput, Chr(39), Chr(46)) ' ' > .
  Return strOutput
```

getTables

Extraccion de la lista de tablas de la conexión. s ele pasan dos parametros, la conexión a la base de datos y el tipo view|table

connDB As Connection/Optional strMod As String

```
  strEngine = connDB.Type
  Select String.LCase(strMod)
    Case "", "table", "tables", "tabla", "tablas"
      strMod = "table"
    Case "view", "views", "vista", "vistas"
      strMod = "view"
  End Select
  If strEngine <> "" Then
    Select strEngine
      Case "sqlite3"
        strSQL = "SELECT name FROM sqlite_master WHERE type='" & strMod & "'"
        resSQL = connDB.Exec(strSQL)
        While resSQL.Available
          strTable = resSQL["name"]
          Select strTable
            Case "sqlite_sequence"
            Case Else
              stxSQL.Add(strTable)
          End Select
          resSQL.MoveNext
        Wend
    End Select
  Endif
  stxSQL.Sort
  Return stxSQL
```

getIndex

connDB As Connection/strTab As String

```
  Select connDB.Type
    Case "sqlite3"
      strSQLk = "SELECT * FROM sqlite_master WHERE type='index' AND tbl_name='" & strTab & "'  AND sql<>''"
      resDBk = connDB.Exec(strSQLk)
      If resDBk.Available Then
        resDBk.MoveFirst
        stxIndex.Clear
        stxIndex = ["", ""]
        For intSnt = 0 To stxSentence.Max
          If InStr(stxSentence[intSnt], "CREATE UNIQUE INDEX") > 0 And InStr(stxSentence[intSnt], strTab) > (
            intPos1 = InStr(stxSentence[intSnt], "(", 1)
            intPos2 = InStr(stxSentence[intSnt], ")", intPos1)
            strSentence = String.Mid(stxSentence[intSnt], intPos1, intPos2 - intPos1)
            strSentence = Replace(stxSentence, "ASC", "")
            strSentence = Replace(strSentence, "DESC", "")
            strSentence = Replace(strSentence, " ", "")
            stxTmp = Split(strSentence, ",")
            For intFi = 0 To stxTmp.Max
            Next
          Endif
        Next
      Endif
  End Select
  Return stxIndex
```

getViewFields

Extraccion de la informacion de la estructura de una vista.

connDB As Connection/strView As String

```
  Select connDB.Type
    Case "sqlite3"
      strSQLk = "SELECT * FROM " & strView
      resDBk = connDB.Exec(strSQLk)
      If connDB.Opened Then
        If resDBk.Available Then
          resDBk.MoveFirst
          intOrd = 0
          For Each fld In resDBk.Fields
            stxFieldInfo.Clear
            stxFieldInfo = ["0", "1", "2", "3", "4", "5", "6"]
            stxFieldInfo[0] = strView
            stxFieldInfo[1] = fld.Name
            stxFieldInfo[2] = String.LCase(GEFUtility.TypeVar(fld.Type))
            stxFieldInfo[3] = CStr(intOrd) ' Numero de columna
            strOrderTmp = Settings[strView & "-order/" & CStr(intOrd), ""]
            intOrder = ""
            strOrder = ""
            stxFieldInfo[4] = CStr(intOrder) ' Tipo de filtro
            stxTableInfo.Add(stxFieldInfo)
            Inc intOrd
          Next
      Endif
    Endif
  End Select
  Return stxTableInfo
```

RecordDelete

Devuelve -1 si no existe o un numero

connDB As Connection/strTable As String/stxDBFields As String[][]/intKey As Integer

```
For intField = 0 To stxDBFields.Max
  If stxDBFields[intField][0] = strTable Then
    If stxDBFields[intField][5] = "YK" Then
      strFieldKey = stxDBFields[intField][1]
      Break
    Endif
  Endif
Next
connDB.Delete(strTable, strFieldKey & "=&1", intKey)
strSQLDelete = "select *"
strSQLDelete &= " from " & strTable
strSQLDelete &= " where " & strFieldKey & "='"
strSQLDelete &= CStr(intKey) & "'"
resCheck = connDB.Exec(strSQLDelete)
If resCheck.Available Then
  If resCheck.Count > 0 Then
    intKey = resCheck[strFieldKey]
  Else
    intKey = -1
  Endif
Else
  intKey = -1
Endif
Return intKey
```

TableMake1

Crea una tabla en la conexion que se le pasa como parametro. Donde el orden de los paramentros dentro de la matriz debe ser el siguiente:

- 0 Nombre de la tabla
- 1 Nombre del campo clave
- 2 Nombre del resto de los campos
- N Nombre del ultimo campo

cnx As Connection/stxParam As String[]

```
If cnx.Tables.Exist(stxParam[0]) = False Then
  cnx.Tables.Add(stxParam[0])
  tbl = cnx.Tables[stxParam[0]]
  With tbl
    For int = 1 To stxParam.Max
      Select int
        Case 1
          strFKey = stxParam[0] & stxParam[int]
          tbl.Fields.Add(strFKey, db.Serial)
        Case 2
          strFIdx = stxParam[0] & stxParam[int]
          tbl.Fields.Add(strFIdx, db.String)
        Case Else
```

```
          strFNme = stxParam[0] & stxParam[int]
          tbl.Fields.Add(strFNme, db.String)
        End Select
      Next
      tbl.PrimaryKey = [strFKey]
      tbl.Update()
      tbl.Indexes.Add(stxParam[0] & "_idx", [strFIdx], True)
    End With
  Endif
  If cnx.Tables.Exist(stxParam[0]) = True Then
    Return 1
  Else
    Return 0
  Endif
```

ViewMake1

Crea una Vista en la conexion que se le pasa como parametro. La funcion trabaja con campos que se llaman TABLA+i y TABLA+n donde n es un numero correlativo. Todos los vinculas seran left join y el orden sera por la segunda columna. Por ejemplo para una tabla de productos donde hay un campo color y otro clase y ambos son tablas relacionadas. Donde el orden de los paramentros dentro de la matriz debe ser el siguiente:

- 0 Nombre de la tabla base
- 1 Este y en adelante, nombre las tablas secundarias

Para el ejemplo

- 0 productos
- 1 color
- 2 clase

Donde color tendra los campos colori, color1 y clase tendrá los campos clasei, clase1. La tabla principal tendra los campos productosi, productos1, productos2

cnx As Connection/stxParam As String[]

```
  strType = cnx.Type
  ptbl = cnx.Tables[stxParam[0]]
  stxHeader.Clear
  stxFields.Clear
  stxJoints.Clear
  stxOrders.Clear
  stxHeader.Add("CREATE VIEW `" & "v" & stxParam[0] & "` AS SELECT\n")
  With ptbl
    intfld = 0
    For Each fld In .Fields
      Select .PrimaryKey.Find(fld.Name)
        Case 0 'es la primera clave el resto, si las hubiera, se destartan
          stxFields.Add(fld.Name & strEnd)
          stxTb.Add(String.Mid(fld.Name, 1, 2))
          stxTb.Add(String.Mid(fld.Name, 3, 2))
        Case Else
          Select fld.Type
            Case db.Integer
              strTb = stxTb[intX]
              strFx = strTb & "i"
              strFs = strTb & "1"
              Inc intX
              stxFinfo = ["", "", "", "", strTb, strFx, strFs]
              If stxFinfo[4] <> "" Then
                stxFields.Add(stxFinfo[6] & " AS " & fld.Name)
                stxFields.Add(stxFinfo[5])
                stxJoints.Add("LEFT JOIN " & stxFinfo[4] & " ON " & fld.Name & "=" & stxFinfo[5])
                stxOrders.Add(fld.Name)
              Else
                stxFields.Add(fld.Name)
              Endif
            Case Else
              stxFields.Add(fld.Name)
          End Select
      End Select
      Inc intfld
    Next
  End With
  strCreate = stxHeader.Join("\n")
  strCreate &= stxFields.Join(",\n")
  strCreate &= "\n"
  strCreate &= "FROM " & stxParam[0] & "\n"
  strCreate &= stxJoints.Join("\n")
  strCreate &= "\n"
  strCreate &= "ORDER BY " & stxOrders.Join(", ")
  cnx.Exec(strCreate)
```

RecordNewRefTest

Inserta un registro nuevo en la base de datos.

cnx As Connection/stxTbl As String[]/stxIns As String[][]

```
strSQL = "select * " &
  " from " & stxTbl[1] &
  " where " & stxTbl[2] &
  "='" & stxIns[int][1] & "'"
res = cnx.Exec(strSQL)
For Each fld In res.Fields
  Select fld.Type
    Case db.Integer, db.Serial
      strRef = fld.Name
  End Select
Next
strChk = "select * " &
  " from " & stxTbl[0] &
  " where " & stxIns[int][0] &
  "='" & res[strRef] & "'"
resChk = cnx.Exec(strChk)
If resChk.Count = 0 Then ' Esto es para evitar insertar un registro que ya existe
  If stxIns.Count > 0 Then
    resIns = cnx.Create(stxTbl[0])
    For int = 0 To stxIns.Max
      resIns[stxIns[int][0]] = res[strRef]
      Print stxIns[int][0] & ": " & res[strRef]
    Next
  Endif
  resIns.Update
Endif
```

MDBtoSQL

Estadisticas de bases de datos .mdb usando mdbtools. Como parametro de entrada precisa la ruta completa del archivo .mdb Dependencias: mdbtools DB.V.T.Bytes TB.R.C.Bytes

strFilePath As String

```
stxStatTablesTmp.Clear
stxStatTables.Clear
stxStatFieldsTmp.Clear
stxStatFields.Clear
stxDatabaseInfo.Clear
strPkg = "mdbtools"
strTools = GEFSys.PkgStat(strPkg)
Print "mdbtools " & ("instalado correctamente")
strJobName = String.LCase(File.BaseName(strFilePath))
strJobName = Replace(strJobName, " ", "-")
strJobName = Replace(strJobName, "--", "-")
strFilePathPerm = Stat(File.Dir(strFilePath)).Perm[User.Name]
strJobPath = User.Home &/ ".databases" &/ strJobName
Select InStr(strFilePathPerm, "w") ' Verificacion de que existe permisos de escritura
  Case 0 ' No se tienen permisos de escritura > se escoge el directorio home del usuario
    strFinalPath = strJobPath
  Case Else
    strFinalPath = File.Dir(strFilePath)
End Select
Print strJobPath
If Exist(strJobPath) = False Then
  Shell "mkdir -p '" & strJobPath & "'"
Endif
Shell "mdb-ver '" & strFilePath & "' 2>&1" To strStatVersion
strStatVersion = Replace(strStatVersion, "\n", "")
Print strStatVersion
Shell "mdb-tables -S -1 '" & strFilePath & "' 2>&1" To strStatTables
prsQ1 = Shell "mdb-schema '" & strFilePath & "' postgres > '" & strJobPath &/ strJobName & "-sch-postgres.s
While prsQ1.State = prsQ1.Running
  Wait 0.1
Wend
prsQ2 = Shell "mdb-schema '" & strFilePath & "' mysql > '" & strJobPath &/ strJobName & "-sch-mysql.sql'"
While prsQ2.State = prsQ2.Running
  Wait 0.1
Wend
prsQ3 = Shell "mdb-schema '" & strFilePath & "' sqlite > '" & strJobPath &/ strJobName & "-sch-sqlite.sql'
While prsQ3.State = prsQ3.Running
  Wait 0.1
Wend
Print ("Esquemas extraídos")
stxStatFieldsTmp = Split(File.Load(strJobPath &/ strJobName & "-sch-postgres.sql"), "\n")
For Each strStatFieldLine In stxStatFieldsTmp
  Select String.Mid(strStatFieldLine, 1, 2)
    Case "CR" ' Comienzo de la tabla
      strStatFieldsTable = Split(strStatFieldLine, Chr(34))[1]
    Case "\t" & Chr(34) ' Nombre y tipo de campo
      strStatFieldsTitle = Split(strStatFieldLine, Chr(34))[1]
      strStatFieldsType = Split(strStatFieldLine, Chr(34))[2]
      strStatFieldsType = Replace(strStatFieldsType, ",", "")
      strStatFieldsType = Replace(strStatFieldsType, "\t", "")
      stxStatFields.Add(strStatFieldsTable & "." & strStatFieldsTitle & "." & strStatFieldsType)
  End Select
Next
Print ("Nombres de campos cargados")
File.Save(strJobPath &/ "fields.txt", stxStatFields.Join("\n"))
Print stxStatFields.Join("\n")
prsA = Shell "mdb-tables -S -1 '" & strFilePath & "' > '" & strJobPath &/ "tables.txt'"
While prsA.State = prsA.Running
  Wait 0.1
Wend
```

```
Print ("Nombres de tablas cargados")
If strStatTables <> "" Then
  stxStatTablesTmp = Split(strStatTables, "\n")
  stxStatTablesTmp.Sort
  stxDatabaseInfo.Add(strStatVersion)
  For intW = 0 To stxStatTablesTmp.Max
    strStatTable = stxStatTablesTmp[intW]
    If Mid(strStatTable, 1, 4) <> "MSys" Then
      If strStatTable <> "" Then
        If InStr(strStatTable, " ") = 0 Then
          stxStatTables.Add(strStatTable)
        Endif
      Endif
    Endif
  Next
Endif
intQtyTb = stxStatTables.Count
stxDatabaseInfo.Add(Str(intQtyTb))
intBytes = Stat(strFilePath).Size
stxDatabaseInfo.Add(Str(intBytes))
stxDatabaseInfo.Add(strJobPath)
stxDatabaseInfo.Add(strJobName & ".sqlite")
stxDatabaseInfo.Add(strFinalPath)
For Each strStatTable In stxStatTables
  stxDatabaseInfo.Add(strStatTable)
  strFileExt = strJobPath &/ strStatTable & ".tmp"
  strFileSQL = strJobPath &/ strStatTable & "-data.sql"
  prsB = Shell "mdb-export -D %Y%m%d%H%M%S -H -b strip -R '::rrr::' -d '::ccc::' " & strFilePath & " " & st
  Exec ["notify-send", "-t", "2000", ("Sistema"), ("Extrayendo datos de") & " " & strStatTable]
  While prsB.State = prsB.Running
    Print ("Extrayendo datos de") & " " & strStatTable
    Wait 0.1
  Wend
  prsC = Shell "tr -cd '[:print:]' < " & strFileExt & " | tr -s ' ' | sed 's/\\d96//g' | sed 's/\\d39//g'
  Exec ["notify-send", "-t", "2000", ("Sistema"), ("Formateando datos de") & " " & strStatTable]
  While prsC.State = prsC.Running
    Print ("Formateando datos de") & " " & strStatTable
    Wait 0.1
  Wend
  Print "Ha finalizado exitosamente la extracción de datos de la tabla: " & strStatTable
Next
Exec ["notify-send", "-t", "2000", ("Sistema"), ("Base de datos completada")]
File.Save(strJobPath &/ "tables.txt", stxStatTables.Join("\n"))
Print stxDatabaseInfo.Join("\n")
Print ("Conversion terminada satisfactoriamente")
Return stxDatabaseInfo
```

GEFDesk

FileChooser

Selecciona la ruta completa de un archivo, con el nombre y las extensiones. Como opcional se puede pasar un directorio que es a donde se dirigira el filechooser cuando se abra. También como opcional se pued epasar un filtro de tipos de archivos separados por :, por ejemolo “txt:csv”

Optional strInputPath As String/Optional strFilter As String

```
If strInputPath = "" Then
  strInputPath = User.Home
Endif
```

```
  Dialog.Title = ("Seleccionar archivo")
  If strFilter <> "" Then
    stxExtensions = Split(strFilter, ":")
    strFilterB &= ("Filtro por") & ":"
    For Each strExtension In stxExtensions
      strFilterA &= "*." & String.LCase(strExtension) & ";"
      strFilterA &= "*." & String.UCase(strExtension)
      strFilterB &= " *." & String.LCase(strExtension)
    Next
    Dialog.Filter = [strFilterA, strFilterB]
  Endif
  If Not Dialog.OpenFile(True) Then
    stxFilepaths = Dialog.Paths
  Endif
  Return stxFilepaths
```

GEFStarter

Main

```
  strAppPath = User.Home &/ "." & Application.Name
  stxDBEngines = GEFUtility.FileLoad("engines.txt") ' Motores de bases de datos soportados por gambas
  If Exist(strAppPath) = False Then
    Mkdir strAppPath
  Endif
  Wait 0.05
  If Exist(strAppPath &/ "README.md") = False Then
    Copy "README.md" To strAppPath &/ "README.md"
  Endif
  If Exist(strAppPath &/ "logo.png") = False Then
    Copy "logo.png" To strAppPath &/ "logo.png"
  Endif
  strDBName = Replace(Application.Name & ".db", "-", "")
  If Initiator() = 1 Then
    strTemplate = GEFDataEdit.DBTemplate(stxProgVal)
    If strTemplate <> "" Then
      Print strTemplate
      File.Save(User.Home &/ stxProgVal[2] & ".template", strTemplate)
    Endif
    For intKey = 0 To stxProgKey.Max
      stxProgVal[intKey] = Settings[stxProgKey[intKey], stxProgVal[intKey]]
    Next
  Endif
  Select stxProgVal[7]
    Case "T", "True"
      Select Dependences() ' Estado de las dependencias.///////////////////////////////////////////////
        Case 0
          Message.Info("Algunas caracteristicas no funcionaran hasta que no instale las dependencias")
        Case 1
      End Select
  End Select
  conProgram.Type = stxProgVal[0]
  conProgram.Host = stxProgVal[1]
  conProgram.Name = stxProgVal[2]
  conProgram.Port = stxProgVal[3]
  conProgram.User = stxProgVal[4]
  conProgram.Password = stxProgVal[5]
  conProgram = GEFData.DBOpen(stxProgVal)
  If LoadTitles() = 1 Then
```

```
    stxTables.Clear
    stxTables.Insert(GEFData.getTables(conProgram, "table")) ' Lista de TABLAS
    stxTableFields.Clear
    For intTable = 0 To stxTables.Max
      strTableTmp = stxTables[intTable]
      strSQL = "select * from " & strTableTmp
      resSQL = conProgram.Exec(strSQL)
      stxTmp.Clear
      stxTmp.Insert(GEFData.getFields(conProgram, strTableTmp, "table", stxTitles))
      For intTmp = 0 To stxTmp.Max
        stxTableFields.Add(stxTmp[intTmp])
      Next
    Next
    Print "###Tablas###"
    For intP = 0 To stxTableFields.Max
      Print stxTableFields[intP].Join(":")
    Next
    stxViews.Clear
    stxViews.Insert(GEFData.getTables(conProgram, "view")) ' Lista de VISTAS
    For intView = 0 To stxViews.Max
      strViewTmp = stxViews[intView]
      strSQL = "select * from " & strViewTmp
      resSQL = conProgram.Exec(strSQL)
      stxViewFields.Insert(GEFData.getFields(conProgram, strViewTmp, "view", stxTitles))
    Next
    Print "###Vistas ###"
    For intP = 0 To stxViewFields.Max
      Print stxViewFields[intP].Join(":")
    Next
    If conProgram.Opened Then
      FMain.Show()
    Else
      Message.Warning(("No se pudo abrir la conexión a la base de datos") &
        "\n" & ("Por favor verifique la configuración"))
      FMain.Show()
    Endif
  Endif
```

LoadTitles

La carga de títulos de los campos de las tablas y de las vistas. De esta manera permite luego traducir la aplicacion mas facilmente. La funcion trabaja con tres parametros, el nombre de la tabla o vista el nombre del campo, y el título de este campo, que sera el que se traducira. Esta funcion permite controlar en un solo sitio todos los titulos.

```
  stxTitles.Add(["view_jobs", "jidx", ("Índice")])
  stxTitles.Add(["view_jobs", "jname", ("Trabajo")])
  stxTitles.Add(["view_jobs", "jdate", ("Fecha")])
  stxTitles.Add(["view_jobs", "jowner", ("Cliente")])
  stxTitles.Add(["boms", "midx", ("Índice")])
  stxTitles.Add(["boms", "mjob", ("Trabajo")])
  stxTitles.Add(["boms", "mcod", ("Código")])
  stxTitles.Add(["boms", "mqty", ("Cantidad")])
  stxTitles.Add(["view_boms", "midx", ("Índice")])
  stxTitles.Add(["view_boms", "mjob", ("Trabajo")])
  stxTitles.Add(["view_boms", "mcod", ("Código")])
  stxTitles.Add(["view_boms", "mclass", ("Clase")])
  stxTitles.Add(["view_boms", "mqty", ("Cantidad")])
  stxTitles.Add(["view_boms", "mprice", ("Precio")])
  stxTitles.Add(["view_boms", "muom", ("Unidad")])
  stxTitles.Add(["view_boms", "msupplier", ("Suministrador")])
```

```
stxTitles.Add(["codcls", "sidx", ("Índice")])
stxTitles.Add(["codcls", "sname", ("Clase")])
stxTitles.Add(["codes", "cidx", ("Índice")])
stxTitles.Add(["codes", "cname", ("Insumo")])
stxTitles.Add(["codes", "cclass", ("Clase")])
stxTitles.Add(["codes", "cqty", ("Cantidad")])
stxTitles.Add(["codes", "cuom", ("Unidad")])
stxTitles.Add(["codes", "cprice", ("Precio")])
stxTitles.Add(["view_codes", "cidx", ("Índice")])
stxTitles.Add(["view_codes", "cname", ("Insumo")])
stxTitles.Add(["view_codes", "cclass", ("Clase")])
stxTitles.Add(["view_codes", "cqty", ("Cantidad")])
stxTitles.Add(["view_codes", "cuom", ("Unidad")])
stxTitles.Add(["view_codes", "cprice", ("Precio")])
stxTitles.Add(["composites", "aidx", ("Índice")])
stxTitles.Add(["composites", "aname", ("Compuesto")])
stxTitles.Add(["composites", "adesc", ("Descripción")])
stxTitles.Add(["deliverables", "didx", ("Índice")])
stxTitles.Add(["deliverables", "dname", ("Informe")])
stxTitles.Add(["jobs", "jidx", ("Índice")])
stxTitles.Add(["jobs", "jname", ("Trabajo")])
stxTitles.Add(["jobs", "jdate", ("Fecha")])
stxTitles.Add(["jobs", "jowner", ("Cliente")])
stxTitles.Add(["jobs", "jlogistic", ("Logística")])
stxTitles.Add(["logistics", "lidx", ("Índice")])
stxTitles.Add(["logistics", "lname", ("Logística")])
stxTitles.Add(["owners", "oidx", ("Índice")])
stxTitles.Add(["owners", "oname", ("Cliente")])
stxTitles.Add(["suppliers", "fidx", ("Índice")])
stxTitles.Add(["suppliers", "fname", ("Suministrador")])
stxTitles.Add(["uoms", "uidx", ("Índice")])
stxTitles.Add(["uoms", "uname", ("Simbolo")])
stxTitles.Add(["uoms", "udesc", ("Unidad")])
stxViewsEx.Add(["view_jobs", ("Trabajos"), "icon:/16/add"])
stxViewsEx.Add(["view_boms", ("Materiales y tareas"), "icon:/16/add"])
stxViewsEx.Add(["view_codes", ("Códigos"), "icon:/16/add"])
stxViewsEx.Add(["view_composites", ("Compuestos"), "icon:/16/delete"])
stxViewsEx.Add(["view_deliverables", ("Informes"), "icon:/16/delete"])
stxViewsEx.Add(["view_logistic", ("Logística"), "icon:/16/delete"])
stxViewsEx.Add(["view_owners", ("Clientes"), "icon:/16/delete"])
stxViewsEx.Add(["view_suppliers", ("Suministradores"), "icon:/16/delete"])
stxViewsEx.Add(["view_uoms", ("Unidades de medida"), "icon:/16/delete"])
If stxViewsEx.Count > 0 Then
  Return 1
Else
  Return 0
Endif
```

Dependences

Analisis de dependencias, si hay paquetes que falta instalar se procede a instalarlos y la funcion retorna la cantidad remanente de paquetes, siendo cero si se instalaron todos.

```
str &= ("Debes instalar los siguientes paquetes") & ":\n"
Select GEFSys.DistroShort()
  Case "debian", "ubuntu", "mint"
    strCmm = "sudo apt-get install "
    If Exist("deb.txt") = True Then
      stxPackages = GEFUtility.FileLoad("deb.txt")
    Endif
  Case "manjaro", "arch"
    strCmm = "sudo pacman -S "
    If Exist("arc.txt") = True Then
      stxPackages = GEFUtility.FileLoad("arc.txt")
    Else
      If Exist("deb.txt") = True Then
        stxPackages = GEFUtility.FileLoad("deb.txt")
      Endif
    Endif
  Case "fedora", "redhat"
    strCmm = "sudo dnf install "
    If Exist("rht.txt") = True Then
      stxPackages = GEFUtility.FileLoad("rht.txt")
    Else
      If Exist("deb.txt") = True Then
        stxPackages = GEFUtility.FileLoad("deb.txt")
      Endif
    Endif
  Case "gentoo"
    strCmm = "sudo emerge -a "
    If Exist("gto.txt") = True Then
      stxPackages = GEFUtility.FileLoad("gto.txt")
    Else
      If Exist("deb.txt") = True Then
        stxPackages = GEFUtility.FileLoad("deb.txt")
      Endif
    Endif
  Case "suse", "opensuse"
    strCmm = "sudo zipper install "
    If Exist("osu.txt") = True Then
      stxPackages = GEFUtility.FileLoad("osu.txt")
    Else
      If Exist("deb.txt") = True Then
        stxPackages = GEFUtility.FileLoad("deb.txt")
      Endif
    Endif
End Select
If stxPackages.Count > 0 Then
  stxPackages = GEFSys.PkgDep(stxPackages)
  If stxPackages.Count > 0 Then ' Existen paquete que no estan instalados
    File.Save("/tmp/apt.txt", str & strCmm & stxPackages.Join(" "))
    Wait 1
    Desktop.Open("/tmp/apt.txt")
  Endif
Endif
Return stxPackages.Count
```

GEFSys

Resume

```
  strResume = "So: " & Distro()
  strResume &= " Arquitectura.So: " & ArqSO()
  strResume &= " Arquitectura.Proc: " & ArqMicro()
  strResume &= " Procesador: " & MicroType()
  strResume &= " Ram: " & Ram()
  strResume &= " Nombre.PC: " & ComputerName()
  strResume &= " Usuario: " & CurrentUser()
  strResume &= " Usuarios: " & AllUsers()
```

```
strResume &= " Grupo: " & WGroup()
strResume &= " Gambas: " & Vgambas()
strResume &= " Actualizado: " & LastUpgrade()
Return strResume
```

Distro

Devuelve la distribución instalada xmi

```
Shell "lsb_release -d | cut -d':' -f2" To sDis
Replace(sDis, gb.Tab, "")
sDis = Trim(sDis)
Return sDis
```

ArqSO

Devuelve la Arquitecura del Sistema Operativo

```
Return System.Architecture
```

MicroType

Devuelve el tipo de Procesador

```
Shell "cat /proc/cpuinfo | grep -i ghz | uniq | cut -f2 -d" & ":" To sPro
Return LTrim(Replace(sPro, "\n", ""))
```

ComputerName

Devuelve el nombre del pc

```
Return System.Host
```

GetSystemUsers

Devuelve el una lista de usuarios del sistema Linux.

```
stxTUsr = GEFUtility.FileLoad("/etc/passwd")
If stxTUsr.Count > 0 Then
  For Each strUser In stxTUsr
    If Split(strUser, ":")[5] = "/home" &/ Split(strUser, ":")[0] Then
      If InStr(Split(strUser, ":")[6], "false") = 0 Then
        stxUsr.Add(Split(strUser, ":")[0])
      Endif
    Endif
  Next
Endif
Return stxUsr
```

WGroup

Devuelve el Grupo de trabajo del pc

```
Return System.Domain
```

Vgambas

Devuelve la versión de gambas intalada en el pc

```
Return System.FullVersion
```

Ls

Devuelve un listado del directorio pasado en ruta

Ruta As String

```
Shell "ls -a " & Ruta To sListado 'Almacenamos listado directorio
Return RTrim(Replace(sListado, "\n", ":"))
' dpkg --get-selections es igual a dpkg -l | cut -d ' ' -f3
```

LANIP

Devuelve las IP v4 de la red local, como parametro de entrada requiere la direccion IP base, por ejemplo “192.168.1” pero si no se le pasa el parametro entonces usa la ip de la computadora donde se este ejecutando el programa quitandole el último número. El formato de salida de cada item de la matriz es host-name[tab]8.8.8.8

Optional strBase As String

```
If InStr(strBase, ".") = 0 Then
  strBase = Split(AddressIP(), ".")[0] & "."
  strBase &= Split(AddressIP(), ".")[1] & "."
  strBase &= Split(AddressIP(), ".")[2]
Endif
Shell "nmap -sP " & strBase & ".1-254" To strAddr
If Len(strAddr) > 0 Then
  stxTmp = Split(strAddr, "\n")
  For Each strPart In stxTmp
    If InStr(strPart, "Nmap scan report for ") > 0 Then
      strPart = Replace(strPart, "Nmap scan report for ", "")
      strPart = Replace(strPart, " ", "\t")
      strPart = Replace(strPart, "(", "")
      strPart = Replace(strPart, ")", "")
      stxAddr.Add(strPart)
    Endif
  Next
Endif
Return stxAddr
```

LastNIP

Devuelve el último digito de la Ip

```
Shell "ifconfig | grep inet: | grep Difus.|cut -d '.' -f5|cut  -d ' ' -f1" To sIPs
Return RTrim(Replace(sIPs, "\n", " "))
```

UUIDswap

```
''Devuelve UUID de la swap para utilizarlo como PK de la BDD
Shell "blkid | grep swap | cut -d ' ' -f2 |cut -d '=' -f2" To sUID
Return Left(Right(sUID, -1), -2)
```

PkgStat

Devuelve el estado respecto a la instalacion de un paquete. Requiere como parametro de entrada el nombre exacto del paquete.

strPkg As String

```
strDistroShort = DistroShort()
Select strDistroShort
  Case "arch", "manjaro"
    strCommand = "pacman -Qs " & strPkg
    Shell strCommand & " 2>&1" To strPkgStatus
    If strPkgStatus <> "" Then
      strPkgStatus = Split(strPkgStatus, "\n")[0]
      strPkgStatus = Split(strPkgStatus, " ")[0]
    Endif
    Select strPkgStatus
      Case "local/" & strPkg
        bolPkgStatus = True
      Case Else
        bolPkgStatus = False
    End Select
  Case "debian", "ubuntu", "mint"
    strCommand = "dpkg-query -W -f='${Status}\n' " & strPkg
    Shell strCommand & " 2>&1" To strPkgStatus
    If strPkgStatus = "install ok installed\n" Then
      bolPkgStatus = True
    Else
      bolPkgStatus = False
    Endif
End Select
Return bolPkgStatus
```

PkgDep

Verifica si los paqutes que se le pasan como parametros en una matriz, estan instalados en el sistema, devuelve una matriz con los paquetes que no estan instalados, si todo lo estuviera la matriz devuelta estara vacia.

stxPackages As String[]

```
For intPkg = 0 To stxPackages.Max
  If PkgStat(stxPackages[intPkg]) = False Then
    stxMissing.Add(stxPackages[intPkg])
  Endif
Next
Return stxMissing
```

GEFUtility

DirParent

Devuelve el directorio padre de otro que se pasa como parámetro.

strPath As String

```
If strPath <> "" Then
  intLast = String.RInStr(strPath, "/")
  strParent = String.Mid(strPath, 1, intLast)
  If String.Len(strParent) > 1 Then
    If String.Right(strParent) = "/" Then
      strParent = String.Mid(strParent, 1, String.Len(strParent) - 1)
    Endif
  Endif
Endif
If Exist(strParent) = False Then
  strParent = "-1"
Else
  If Stat(strParent).Type <> gb.Directory Then
    strParent = "-1"
  Endif
Endif
Return strParent
```

CodeTag

Analiza de una cadena de texto que se le pasa como parámetro y en el contexto de un fragmento de código, devuelve que es esa frase.

str As String

```
stxStruc = CodeStructure()
stxCoin.Clear
strJob = Replace(str, "  ", " ")
strJob = Replace(strJob, " '", "'")
Select strJob
  Case ""
    For int1 = 0 To stxStruc.Max
      If stxStruc[int1] = "##Blank Line" Then
        inxCoin.Add(1)
      Else
        inxCoin.Add(0)
      Endif
    Next
  Case Else
    If String.Mid(strJob, 1, 1) = " " Then
      stxSplitText = GEFValidator.SplitText(strJob)
      strIsComment = "no"
      For intCh = 0 To stxSplitText.Max
        If stxSplitText[intCh] = " " Then
          If stxSplitText[intCh + 1] = "'" Then
            strIsComment = "yes"
            Break
          Endif
        Endif
      Next
      Select strIsComment
        Case "yes"
          strJob = "'c"
      End Select
    Endif
    For int1 = 0 To stxStruc.Max ' alrededor de 50 frases
      stxSTmp = Split(stxStruc[int1], "#")
      int3 = 0
      If String.Mid(strJob, 1, 1) = stxSTmp[0] Then
        For int2 = 1 To stxSTmp.Max - 1
          If InStr(strJob, stxSTmp[int2]) > 0 Then
            Inc int3
          Endif
        Next
        inxCoin.Add(int3)
      Else 'si la 1º letra no coincide > es 0, es decir no hay coincidencia
        inxCoin.Add(int3) ' en esta instancia int3 vale 0
      Endif
    Next
End Select
For intCoin = 0 To stxStruc.Max
  If inxCoin[intCoin] > 0 Then
    If inxCoin[intCoin] = ArrayMax(inxCoin) Then
      strType = Split(stxStruc[intCoin], "#")[Split(stxStruc[intCoin], "#").Max]
    Endif
  Endif
Next
Select strType
  Case ""
    strType = "Code"
End Select
strOutput = strType & "\t" & str
Return strOutput
```

RelationProj

Lee las matrices de métodos y codigo del proyecto para luego analizar las relaciones entre estos generando una matrix con estas relaciones.

stxMet As String[]/stxCod As String[]

```
  For intCod = 0 To stxCod.Max
    For intMet = 0 To stxMet.Max
      If InStr(stxCod[intCod], stxMet[intMet]) > 0 Then
        stxReltn.Add(stxMet[intCod] & "=" & stxMet[intMet])
      Endif
    Next
  Next
  Return stxReltn
```

CodeComment

str As String

DokuHtm2

Devuelve un html con las funciones de un módulo y todos los datos de estas, comoparametro de entrada requiere el directorio raiz a partir del cual buscar los módulos.

```
  stxInfo = GEFSys.ProjInfo()
  strHtml &= ""
  strHtml &= "" ""
  strHtml &= ""
  strHtml &= " " " ""
  strHtml &= ""
  strHtml &= ""
  strHtml &= "" & Application.Name & ""
  strHtml &= ""
  strHtml &= "" & ("Autor") & ": " & stxInfo[2] & ""
  strHtml &= "" & ("Proveedor") & ": " & stxInfo[3] & ""
  strHtml &= "" & ("Versión") & ": " & stxInfo[4] & ""
  strHtml &= "" & ("Componentes") & ""
  strHtml &= "" & GEFWeb.ListHtml(stxInfo[5], ":") & ""
  strHtml &= "" & Application.Name & " " & ("consta de") & " " & CStr(FMain.stxClass.Count) & " " & ("métdos"
  For intFun = 0 To FMain.stxClass.Max
```

```
    If FMain.stxClass[intFun] <> strCurrClass Then
      strHtml &= "" & FMain.stxClass[intFun] & ""
      strCurrClass = FMain.stxClass[intFun]
    Endif
    strHtml &= "" & FMain.stxName[intFun] & ""
    strHtml &= "" & FMain.stxDesc[intFun] & ""
    strHtml &= "" & FMain.stxArgs[intFun] & ""
    strHtml &= "" & FMain.stxCode[intFun] & "
```

" Next ‘ ’FMain.stxClass ’ ‘FMain.stxName ’ ‘FMain.stxArgs ’ ‘FMain.stxDesc ’ ‘FMain.stxCode ’ ’FMain.stxVars Return strHtml

FilesNew

Devuelve una lista de archivos de un directorio que se pasa como parametro. Opcionalmente se puede pasar como parametro una lista de archivos existentes los cuales seran omitidos de la lista de salida si es que son encontrados y un filtro de extensiones de archivo de l estilo mp3:ods:txt

strDirectory As String/Optional stxFilesOpt As String[]/Optional strFilterOpt As String

```
  stxFiles = GEFUtility.ScanFolder(strDirectory, strFilterOpt)
  If stxFilesOpt.Count > 0 Then
    For intFile = 0 To stxFiles.Max
      If stxFilesOpt.Find(stxFiles[intFile]) = -1 Then
        stxFilesNew.Add(stxFiles[intFile])
      Endif
    Next
  Else ' Todos los archivos que se encuentren seran nuevos
    stxFilesNew = stxFiles
  Endif
  Return stxFilesNew
```

FilesNone

Devuelve una lista de archivos que no existen en el directorio, es necesario parar una lista de archivos para contrastar.

strDirectory As String/Optional stxFilesOpt As String[]/Optional strFilterOpt As String

```
  stxFiles = GEFUtility.ScanFolder(strDirectory, strFilterOpt)
  If stxFilesOpt.Count > 0 Then
    For intFile = 0 To stxFilesOpt.Max
      If stxFiles.Find(stxFilesOpt[intFile]) = -1 Then
        stxFilesNone.Add(stxFilesOpt[intFile])
      Endif
    Next
  Else
    stxFilesNone.Clear
  Endif
  Return stxFilesNone
```

FileNospace

Devuelve un texto, nombre de arcivo concatenando todos los fragmentos que se le pase y pone todo en minusculas quita los caracteres fuera del rango 97-122 de ascci.

stxParam As String[]/Optional strDelim As String/Optional strExt As String

```
  If strDelim = "" Then
    strDelim = "#"
  Endif
  If strExt <> "" Then
```

```
    strExt = String.LCase(strExt)
    If InStr(strExt, ".") Then
      strExt = Replace(strExt, ".", "")
    Else
      strExt = "." & strExt
    Endif
  Endif
  If stxParam.Count > 0 Then
    For intPar = 0 To stxParam.Max
      strTemp = stxParam[intPar]
      strTemp = String.LCase(strTemp)
      strTemp = String.RemoveDiacritics(strTemp)
      strTemp = Replace(strTemp, "  ", Chr(45))
      strTemp = Replace(strTemp, "_", Chr(45))
      strTemp = Replace(strTemp, "--", Chr(45))
      strOutTmp = ""
      For intLeter = 1 To String.Len(strTemp)
        strLeter = String.Mid(strTemp, intLeter, 1)
        Select Asc(strLeter)
          Case Chr(32), Chr(45) ' Espacio o Gión
            strOutTmp &= "-"
          Case Else
            If Asc(strLeter) > 96 And Asc(strLeter) < 123 Then
              strOutTmp &= strLeter
            Endif
        End Select
      Next
      stxOutput.Add(strOutTmp)
    Next
  Endif
  strOutput = stxOutput.Join(strDelim) & strExt
  Return strOutput
```

FileLoad

Devuelve una matriz de texto con los valores listados en un archivo de texto desl cual se pasa su ruta como parametro.

strPath As String

```
  stxList.Clear
  If Exist(strPath) Then
    strList = File.Load(strPath)
    If InStr(strList, "\n") > -1 Then
      stxListTmp = Split(strList, "\n")
    Else
      If String.Len(strList) > 0 Then
        stxListTmp.Add(strList)
      Endif
    Endif
    For intList = 0 To stxListTmp.Max
      If stxListTmp[intList] <> "" Then
        stxList.Add(stxListTmp[intList])
      Endif
    Next
  Endif
  Return stxList
```

FileTemplate

Tomando un archivo template reemplaza las etiquetas por valores. Retorna una matriz con una lista de archivos, primero el producto y luego el pdf, en cas que alguno de estos no exista en la posicion de la matriz hara una cadena vacia.

strFileSeed As String/strFileProduct As String/stxTag As String[]/stxDat As String[]

```
strFilePdf = File.Dir(strFileProduct) &/ File.BaseName(strFileProduct) & ".pdf"
If Exist(strFileSeed) = True Then
  strTextProduct = File.Load(strFileSeed)
  If strTextProduct <> "" Then
    For intN = 0 To stxTag.Max
      strTextProduct = Replace(strTextProduct, stxTag[intN], stxDat[intN])
    Next
    File.Save(strFileProduct, strTextProduct)
    Wait 0.1
    prsTemp = Shell "dia " & strFileProduct & " -e " & strFilePdf
    While prsTemp.State = prsTemp.Running
      Wait 0.1
    Wend
  Endif
Endif
If Exist(strFileProduct) = True Then
  stxFilesOutput.Add(strFileProduct)
Else
  stxFilesOutput.Add("")
Endif
If Exist(strFilePdf) = True Then
  stxFilesOutput.Add(strFilePdf)
Else
  stxFilesOutput.Add("")
Endif
Return stxFilesOutput
```

ArrangePath

Devuelve una ruta sin los saltos del línea ni caracteres problemáticos

strPathRaw As String

```
strPath = Replace(strPathRaw, "\n", "")
strPath = Replace(strPath, "\r", "")
strPath = Replace(strPath, "\x00", "")
Return strPath
```

FileExifPages

Devuelve la cantidad de páginas del archivo si no tiene el tag entonces se devuelve 1. Para la extraccion de esta informacion se usa ExifTool.

strPath As String

```
strPath = ArrangePath(strPath)
Shell "exiftool -f -s -s '" & strPath & "' 2>&1" To strExifBruto ' Toma todos los tags del archivo
stxExifBruto = Split(strExifBruto, "\n")
For intLin = 0 To stxExifBruto.Max
  If stxExifBruto[intLin] <> "" Then
    intCur = InStr(stxExifBruto[intLin], ": ")
    intLen = String.Len(stxExifBruto[intLin])
    If String.Mid(stxExifBruto[intLin], 1, intCur - 1) = "PageCount" Then
      intPages = CInt(String.Mid(stxExifBruto[intLin], intCur + 2, intLen - intCur - 1))
      Break
    Else
      intPages = 1
    Endif
  Endif
Next
Return intPages
```

Timestamp

Retorna una cadena de texto con el tiempo en formato “yyyymmddhhnnss”.

datTime As Date

```
Return Format(datTime, "yyyymmddhhnnss")
```

MouseButton

Funcion que retorna el nombre en ingés del boton del ratón que se ha presionado.

intKey As Integer

```
Select intKey
  Case 1
    strMouseButton = "Left"
  Case 2
    strMouseButton = "Right"
  Case 4
    strMouseButton = "Center"
  Case 16
    strMouseButton = "Function-1"
  Case 8
    strMouseButton = "Function-2"
End Select
Return strMouseButton
```

ArrayMax

Devuelve el maximo valor de lalista de numeros enteros.

inx As Integer[]

```
For Each int In inx
  If int > intRet Then
    intRet = int
  Endif
Next
Return intRet
```

FileInfo

Devuelve una matriz con datos del archivo que se le pasa como ruta.

strFilePath As String

```
strFilePath = ArrangePath(strFilePath)
strSep = "\t"
If Stat(strFilePath).Type = gb.File Then
  stxFileMeta.Add("FilePath" & strSep & strFilePath)
  strFileDir = File.Dir(strFilePath)
  stxFileMeta.Add("FileDirectory" & strSep & strFileDir)
```

```
    strFileName = File.Name(strFilePath)
    stxFileMeta.Add("FileName" & strSep & strFileName)
    strFileExt = File.Ext(strFilePath)
    stxFileMeta.Add("FileExt" & strSep & strFileExt)
    strFileBase = File.BaseName(strFilePath)
    stxFileMeta.Add("FileBase" & strSep & strFileBase)
    strFileSize = Stat(strFilePath).Size
    stxFileMeta.Add("FileSize" & strSep & strFileSize)
    strFileTime = GEFUtility.Timestamp(Stat(strFilePath).Time)
    strFileTime = Replace(strFileTime, ":", ".")
    stxFileMeta.Add("FileTime" & strSep & strFileTime)
    strFileVersion = GEFutility.FileVersion(strFilePath)
    strFileVersion = Replace(strFileVersion, ":", ".")
    stxFileMeta.Add("FileVersion" & strSep & strFileVersion)
  Endif
  Return stxFileMeta
```

HMStoSeconds

Devuelve el tiempo en segundos de una cadena que se le pase con el formato HH:MM:SS HORAS:MINUTOS:SEGUNDOS.

strTime As String

```
  stxTime = Split(strTime, ":")
  intTime = (stxTime[0] * 3600) + (stxTime[1] * 60) + stxTime[2]
  Return intTime
```

MkConfXml

Creacion de archivo de configuracion inicial xml.

strXmlPath As String

```
  stxParameters.Add("MediaFolder:Path")
  stxParameters.Add("CapitalMode:Mode")
  stxParameters.Add("LangCurr:Name")
  stxParameters.Add("Languages:Name")
  stxParameters.Add("Software:Name:Seed:Prod")
  writer.Open(strXmlPath, True) 'True es para que le ponga los saltos de linea
  writer.StartElement(Application.Name)
  For intE = 0 To stxParameters.Max
    stxAtrib.Clear
    stxAtrib = Split(stxParameters[intE], ":")
    writer.StartElement(stxAtrib[0])
    If stxAtrib.Count > 1 Then
      For intA = 1 To stxAtrib.Max
        writer.StartElement(stxAtrib[intA])
        writer.Text("")
        writer.EndElement
      Next
    Endif
    writer.EndElement
  Next
  writer.EndElement
  writer.EndDocument
  Return 1
```

WhereRun

Indica si el programa se esta ejecutando desde el IDE o desde un ejecutable solo utilizando código de gambas.

```
strProcess = File.Load("/proc" &/ CStr(Application.Id) &/ "comm")
If Left(strProcess, 4) = "gbx3" Then
  intRun = 1
Else
  intRun = 0
Endif
Return intRun
```

GEFValidator

VEmail

Validación de una direccion de correo electrónico

strAddress As String

```
If regex.Match(strAddress, strPattern, regex.Caseless) = True Then
  strChecked = strAddress
Else
  strChecked = ""
Endif
Return strChecked
```

OnlyTextParenthesis

Validación de solo texto, espacio, punto y coma entre parentesis.

strInput As String

```
If regex.Match(strInput, strPattern, regex.Caseless) = True Then
  strChecked = strInput
Else
  strChecked = ""
Endif
Return strChecked
```

CaptionCheck

Validación del texto de un control en KDE el texto de los botones por ejemplo tiene un simbolo & delante del texto.

strInput As String

```
If String.Left(strInput) = "&" Then
  strOut = String.Right(strInput, -1)
Endif
Return strOut
```

OnlyNumbers

Devuelve un texto solo con numeros.

strInput As String

```
btxLeters = Byte[].FromString(strInput)
stxOut.Clear
stxLeters.Clear
stxLeters = Split("0:1:2:3:4:5:6:7:8:9", ":")
For int = 1 To String.Len(strInput)
  strSymbol = String.Mid(strInput, int, 1)
  intKey = stxLeters.Find(strSymbol)
  If intKey > -1 Then
```

```
      stxOut.Add(strSymbol)
    End If
  Next
  strOut = stxOut.Join("")
  Return strOut
```

OnlyText

Validación de solo texto, Numeros NO, Doble espacio NO, Espacio Al principio y/o al final NO.

strInput As String

```
  strError = ""
  stxSpaces.Clear
  stxRepated.Clear
  stxExcluded.Clear
  stxEx.Clear
  inxExN.Clear
  stxLeters.Clear
  stxLower.Clear
  stxUpper.Clear
  stxUpper = Split("A:B:C:D:E:F:G:H:I:J:K:L:M:N:O:P:Q:R:S:T:U:V:W:X:Y:Z:Á:À:Â:Ã:É:È:Í:Ï:Ó:Ô:Õ:Ö:Ú:Ç:Ñ", ":")
  stxLower = Split("a:b:c:d:e:f:g:h:i:j:k:l:m:n:o:p:q:r:s:t:u:v:w:x:y:z:à:â:ã:é:è:ê:í:ï:ó:ô:õ:ö:ú:ç:ñ", ":")
  stxLeters.Insert(stxLower)
  stxLeters.Insert(stxUpper)
  intM = 0
  For int = 1 To String.Len(strInput)
    strSymbol = String.Mid(strInput, int, 1)
    intKey = stxLeters.Find(strSymbol)
    Select intKey
      Case -1
        Select strSymbol
          Case " ", ",", ".", "¿", "?", "!", "¡"
            If String.Right(strChecked) <> strSymbol Then
              strChecked &= strSymbol
            Else
              stxRepated.Add("'" & strSymbol & "'")
            Endif
          Case Else
            Select Asc(strSymbol)
              Case 9
                stxExcluded.Add("'TB'")
              Case 10
                stxExcluded.Add("'LF'")
              Case 13
                stxExcluded.Add("'CR'")
              Case Else
                stxExcluded.Add("'" & strSymbol & "'")
            End Select
        End Select
      Case Else
        If stxUpper.Find(strSymbol) > -1 Then
          Inc intM
        Endif
        strChecked &= strSymbol
    End Select
  Next
  If String.Right(strChecked) = " " Then
    strChecked = String.Mid(strChecked, 1, String.Len(strChecked) - 1)
    stxSpaces.Add("end")
  Endif
  If String.Left(strChecked) = " " Then
    strChecked = String.Mid(strChecked, 2)
    stxSpaces.Add("ini")
  Endif
  If stxSpaces.Find("ini") > -1 Then
    If stxSpaces.Find("end") > -1 Then
      strError &= "[" & ("Espacios al inicio y al final") & "]"
    Else
      strError &= "[" & ("Espacio al inicio") & "]"
    Endif
  Else
    If stxSpaces.Find("end") > -1 Then
      strError &= "[" & ("Espacio al final") & "]"
    Endif
  Endif
  If stxRepated.Count > 0 Then
    strError &= "[" & ("Repetidos") & ": " & stxRepated.Join(",") & "]"
  Endif
  If stxExcluded.Count > 0 Then
    For int = 0 To stxExcluded.Max
      intKx = stxEx.Find(stxExcluded[int])
      If intKx = -1 Then
        stxEx.Add(stxExcluded[int])
        inxExN.Add(1)
      Else
        inxExN[intKx] = inxExN[intKx] + 1
      Endif
    Next
  Endif
  stxExcluded.Clear
  If stxEx.Count > 0 Then
    For int = 0 To stxEx.Max
      Select inxExN[int]
        Case 1
          stxExcluded.Add(stxEx[int])
        Case Else
          stxExcluded.Add(stxEx[int] & "#" & CStr(inxExN[int]))
      End Select
    Next
    strError &= "[" & ("Excluidos") & ": " & stxExcluded.Join(",") & "]"
  Endif
  If intM > 1 Then
    strError &= "[" & ("Mas de una letra mayúscula") & "]"
  Endif
  If strError = "" Then
    Return strChecked
  Else
    Return strInput & "\t" & strError
  Endif
```

ConvertPath

Descodifica los caracteres hexadecimales en las URI's recorriendo la cadena dada Params: strInput la URintPos a descodificar Return: la URintPos descodificada

strInput As String

```
  strOutput = ""
  intLen = Len(strInput)
```

```
intPos = 1
Do While intPos <= intLen
  strChar = Mid$(strInput, intPos, 1)
  If strChar = "+" Then
    strOutput = strOutput & strChar
  Else If strChar <> "%" Then
    strOutput = strOutput & strChar
  Else If intPos > intLen - 2 Then
    strOutput = strOutput & strChar
  Else
    strDigits = Mid$(strInput, intPos + 1, 2)
    strOutput = strOutput & Chr$(CInt(Val("&" & strDigits)))
    intPos = intPos + 2
  Endif
  intPos = intPos + 1
Loop
Return strOutput
```

SplitText

Particiona un texto dado como parametro, si el segundo argumento, que es la palabra o letra de corte es nula, cada item de la matriz sera un caracter de la cadena de texto, si , por el contrario, se pasa un parametro de corte y este existe en la cadena, esta sera dividida por este parametro. Pero en el caso que se pase una frase de corte y esta no exista se devolvera la misma fras eoriginal sin alterar como item cero de la matriz.

strText As String/Optional strCut As String

```
stx.Clear
Select strCut
  Case ""
    For int = 1 To String.Len(strText)
      stx.Add(String.Mid(strText, int, 1))
    Next
  Case Else
    If InStr(strText, strCut) > 0 Then
      stx = Split(strText, strCut)
    Else
      stx.Add(strText)
    Endif
End Select
Return stx
```

VRUTChile

strRutIn As String/strDigRut As String

```
strDigit = ""
intConstant = 1
intLen = Len(Trim(strRutIn))
strRutTmp = Val(Trim(strRutIn))
Do Until intLen = 0
  intConstant = intConstant + 1
  intPlus = intPlus + Mid(strRutTmp, intLen, 1) * intConstant
  If intConstant = 7 Then
    intConstant = 1
  End If
  intLen = intLen - 1
Loop
intDigit = intPlus Mod 11
strDigit = Str(11 - intDigit)
```

```
If Val(strDigit) = 11 Then
  strDigit = "0"
End If
If Val(strDigit) = 10 Then
  strDigit = "K"
End If
If Trim(strDigit) <> Trim(strDigRut) Then
  bolValid = False
Else
  bolValid = True
End If
Return bolValid
```

Capital

Devuelve un texto con la primera letra en mayusculas y todas las siguientes en minúsculas.

strInput As String

```
If strInput <> "" Then
  strOutput = String.UCase(String.Mid(strInput, 1, 1))
  strOutput &= String.LCase(String.Mid(strInput, 2, String.len(strInput) - 1))
Else
  strOutput = ""
Endif
Return strOutput
```

GEFWeb

ListHtml

strList As String/strSep As String

```
If InStr(strList, strSep) > 0 Then
  stx = Split(strList, strSep)
Else
  If strList <> "" Then
    stx.Add(strList)
  Endif
Endif
str = ""
For int = 0 To stx.Max
  str &= "" & stx[int] & ""
Next
str &= ""
Return str
```

FMain

Form_Open

```
HSplit2.Layout = [1, 4, 1]
txeCode.View.Highlight = "gambas"
If LoadModel(Application.Path) > 0 Then
  ArrangeMethods()
Else
  Message.Info(("El proyecto no pudo ser cargado"))
Endif
ShowData()
```

tobAbout_Click

```
GEFAbout.ShowModal()
```

tobConfig_Click

```
GEFConfig.ShowModal()
```

tobHelp_Click

```
strHtml = GEFUtility.DokuHtm2()
If Exist("/tmp/pdf") Then
  prsRM = Shell "rm -r -f /tmp/pdf"
  While
    prsRM.State = prsRM.Running
    Wait 0.1
  Wend
Endif
prsMd = Shell "mkdir -p /tmp/pdf"
While
  prsMd.State = prsMd.Running
  Wait 0.1
Wend
File.Save("/tmp/pdf/help.html", strHtml)
Copy "logo.png" To "/tmp/pdf/logo.png"
Wait 0.5
strPDF = GEFBatch.HTMLPDF("/tmp/pdf/help.html", "pandoc")
If Exist(strPDF) Then
  Desktop.Open(strPDF)
Endif
```

mnuDevHelp_Click

```
DevDocument()
```

mnuPrint_Click

```
GEFPrint.ShowModal()
```

DevDocument

```
strInput = GEFUtility.DokuHtml(Application.Path &/ ".src")
strPath = Application.Path &/ "devdoc.html"
File.Save(strPath, strInput)
Wait 2
Select GEFStarter.stxProgVal[9]
  Case "ide"
    strOutput = GEFBatch.HTMLPDF(strPath, GEFStarter.stxProgVal[10])
    Desktop.Open(strOutput, True)
  Case Else
    If Exist(strOutput) Then
      Desktop.Open(strOutput, True)
    Else
      Message.Info("La documentacion para el desarrollador no esta disponible")
    Endif
End Select
```

ArrangeMethods

```
If bolLoaded = False Then
  trvMethods.Clear
  strClassCurrent = ""
  trvMethods.Add(Application.Name, Application.Name, Picture["icon:/16/linux"])
  For int = 0 To stxMethod.Max
    TreeIndoLoad(int)
    Select inx.Count
      Case 0
        If strFilterText = "" Then
          If trvMethods.Exist(strClass) = False Then
            trvMethods.Add(strClass, strClass, Picture["icon:/16/add"], Application.Name)
          Endif
          trvMethods.Add(strClass & "." & strName, strName, Picture["icon:/16/apply"], strClass)
        Else
        Endif
      Case Else
        If inx.Find(int) > -1 Then
          If trvMethods.Exist(strClass) = False Then
            trvMethods.Add(strClass, strClass, Picture["icon:/16/add"], Application.Name)
          Endif
          trvMethods.Add(strClass & "." & strName, strName, Picture["icon:/16/apply"], strClass)
        Endif
    End Select
    strClassCurrent = strClass
  Next
  trvMethods[Application.Name].Expanded = True
  txeCode.Text = ""
Endif
Select GEFStarter.stxProgVal[16]
  Case "True", "T"
    TabPanel1[0].Visible = True
  Case Else
    TabPanel1[0].Visible = False
End Select
```

trvMethods_Select

```
If InStr(trvMethods.Key, ".") > 0 Then
  strMeClass = Split(trvMethods.Key, ".")[0]
  strMeName = Split(trvMethods.Key, ".")[1] ' NOmbre del método
  For intKeyMet = 0 To stxName.Max
    If stxName[intKeyMet] = strMeName Then
      inxKey.Add(intKeyMet)
    Endif
  Next
  Wait 0.01
  For intKeyCls = 0 To inxKey.Max
    If stxClass[inxKey[intKeyCls]] = strMeClass Then
      intKey = inxKey[intKeyCls]
      Break
    Endif
  Next
  txeCode.Text = stxCode[intKey]
  lblInformation.Text = stxDesc[intKey]
  lblInformation.Refresh
Endif
```

Search_Change

```
bto = Last
strFilterText = bto.Text
Print bto.Tag & ":" & bto.Text
Strainer(bto.Tag, bto.Text)
```

tobDevHelp_Click

```
strOutput = User.Home &/ "tmp.tex"
strFilePdf = User.Home &/ "tmp.pdf"
stx = GEFUtility.RelationProj(stxName, stxCode)
stxColor.Add("blue!30")
stxColor.Add("green!40")
stxColor.Add("red!30")
stxColor.Add("purple!50")
stxColor.Add("teal!40")
stxColor.Add("yellow!30")
If stx.Count > 0 Then
  strData &= "\\documentclass[landscape]{article}\n"
  strData &= "\\usepackage[utf8]{inputenc}\n"
  strData &= "\\usepackage{tikz}\n"
  strData &= "\\usepackage[a2paper]{geometry}\n"
  strData &= "\\usetikzlibrary{mindmap}\n"
  strData &= "\\pagestyle{empty}\n"
  strData &= "\\begin{document}\n"
  strData &= "\\begin{tikzpicture}[mindmap, grow cyclic, every node/.style=concept, concept color=orange!4
  strData &= "    level 1/.append style={level distance=8cm,sibling angle=25},\n"
  strData &= "    level 2/.append style={level distance=6cm,sibling angle=25}]\n"
  strData &= "\\node{" & Application.Name & "}\n"
  For int = 0 To stx.Max
    strNode = Split(stx[int], "=")[0]
    strNode = Replace(strNode, "_", "")
    strChild = Split(stx[int], "=")[1]
    strChild = Replace(strChild, "_", "")
    If strNode <> strNodeCurrent Then
      Select int
        Case 0
        Case stx.Max
          strData &= "}\n"
        Case Else
          strData &= "}\n"
      End Select
      strData &= "  child [concept color=" & stxColor[intColor] & "] { node {" & strNode & "}\n"
      strData &= "    child { node {" & strChild & "}}\n"
      If intColor < 5 Then
        Inc intColor
      Else
        intColor = 0
      Endif
      strNodeCurrent = strNode
    Else
      strData &= "    child { node {" & strChild & "}}\n"
    Endif
    Select int
      Case stx.Max
        strData &= "}\n"
    End Select
  Next
```

```
    strData &= ";\n"
    strData &= "\\end{tikzpicture}\n"
    strData &= "\\end{document}"
    File.Save(strOutput, strData)
    Wait 1
    If Exist(strOutput) = True Then
      If GEFBatch.LATEXPDF(strOutput, "pdflatex") = strFilePdf Then
        Desktop.Open(strFilePdf)
      Endif
    Endif
    Desktop.Open(strOutput)
  Endif
```

mnuExit_Click

```
  Me.Close
```

ShowData

Muestra los datos de la base de datos, suas vistas y consultas SQL que den un resultado en el gridview y en el Treeview

```
  If GEFStarter.stxViews.Count > 0 Then
    strRoot = GEFStarter.stxProgVal[2] ' Nombre de la conexion a BBDD
    strRootAlt = GEFStarter.stxProgVal[17] ' Nombre alternativo de la aplicación
    picTab = Picture["icon:/16/sun"]
    trvData.Add(strRoot, strRootAlt, Picture["icon:/16/sun"])
    For intTree = 0 To GEFStarter.stxViewsEx.Max
      strKey = GEFStarter.stxViewsEx[intTree][0]
      strText = GEFStarter.stxViewsEx[intTree][1]
      picTab = Picture[GEFStarter.stxViewsEx[intTree][2]]
      trvData.Add(strKey, strText, picTab, strRoot)
    Next
  Endif
```

UpdateGrid

```
  grwData.Header = 1 ' Muestra solo el encabezado
  grwData.ScrollBar = 3 ' Muestra los scrol vertical y horizontal
  grwData.Mode = Select.Multiple
  GEFStarter.strSQLCurrent = GEFData.SqlMake(GEFStarter.strViewNameSqlCurrent, GEFStarter.stxViewFields)
  GEFStarter.resProgram = GEFStarter.conProgram.Exec(GEFStarter.strSQLCurrent)
  grwData.Rows.Count = 0
  If GEFStarter.resProgram.Available Then
    If GEFStarter.resProgram.Count > 0 Then
      grwData.Rows.Count = 0
      grwData.Rows.Count = GEFStarter.resProgram.Count
    Endif
  Endif
  grwData.Columns.Count = GEFStarter.resProgram.Fields.Count
  intFld = 0
  For int = 0 To GEFStarter.stxViewFields.Max
    If GEFStarter.stxViewFields[int][0] = GEFStarter.strViewNameSqlCurrent Then
      grwData.Columns[intFld].Title = GEFStarter.stxViewFields[int][10]
      Select GEFStarter.stxViewFields[int][6]
        Case ""
          GEFStarter.stxViewFields[int][6] = 75
      End Select
      grwData.Columns[intFld].Width = GEFStarter.stxViewFields[int][6]
      strGridState = "loading"
      Inc intFld
    Endif
  Next
  If strGridState = "loading" Then
    strGridState = "loaded"
  Endif
  Print GEFStarter.strSQLCurrent
  Return 1
```

tobNewRecord_Click

```
  cmdRecordNew()
```

tobEditRecord_Click

```
  cmdRecordEdit()
```

cmdArrangeButtonFilters

```
  pnlDataFilter.Children.Clear
  pnlDataFilter.Arrangement = Arrange.Fill
  With btnFilter
    .Name = "btnFilter~" & "OnWorking" '& stxButtonsFields[intA]
    .Border = False
    .Picture = Picture["icon:/16/filter"]
    .Text = ("Los filtros todavía no están implementados")
    .Tag = 0
    .Expand = True
    .Width = pnlDataFilter.Width
    .Height = pnlDataFilter.Height
  End With
```

GOrderType_Click

```
  btn = Last
  intColumn = CInt(btn.Tag)
  strFieldName = GEFStarter.stxFieldsViewCurrent[intColumn]
  For int = 0 To GEFStarter.stxViewFields.Max
    If GEFStarter.stxViewFields[int][0] = GEFStarter.strViewNameSqlCurrent Then
      If GEFStarter.stxViewFields[int][1] = strFieldName Then
        Select btn.Picture
          Case Null
            btn.Picture = Picture["icon:/16/sort-ascent"]
            GEFStarter.stxViewFields[int][3] = "asc"
          Case Picture["icon:/16/sort-ascent"]
            btn.Picture = Picture["icon:/16/sort-descent"]
            GEFStarter.stxViewFields[int][3] = "desc"
          Case Picture["icon:/16/sort-descent"]
            btn.Picture = Null
            GEFStarter.stxViewFields[int][3] = ""
        End Select
      Endif
    Endif
  Next
  UpdateGrid()
```

grwData_DblClick

```
cmdRecordEdit()
```

cmdRecordNew

```
strTable = Replace(GEFStarter.strViewNameSqlCurrent, "view_", "")
GEFDataEdit.RunEditor(GEFStarter.conProgram, strTable)
```

cmdRecordDelete

```
strTable = Replace(GEFStarter.strViewNameSqlCurrent, "view_", "")
For intR = 0 To grwData.Rows.Max
  If grwData.Rows[intR].Selected = True Then
    Inc intCount
    intKey = CInt(grwData[intR, 0].Text)
    If GEFData.RecordDelete(GEFStarter.conProgram, strTable, GEFStarter.stxTableFields, intKey) = -1 Then
      Inc intOk
    Endif
  Endif
Next
If intCount > 0 Then
  If intOk = intCount Then
    UpdateGrid()
  Else
    Select intOk
      Case 0
        Message.Info("No se pudo borrar ningun registro")
      Case Else
        Message.Info("Hay registros que no se pudieron borrar")
    End Select
  Endif
  UpdateGrid()
Else
  Message.Info("Debe seleccionar registros de la lista")
Endif
```

FTest

btnAddressIP_Click

```
strIP = GEFSys.AddressIP()
Message.Info(strIP)
```