



Manual de uso de la herramienta MapBD. Versión 0.0.3



Autor: Julio Sánchez Berro
Licencia GPL 3.
Software libre
<http://jsbsan.blogspot.com/>

4 de Enero de 2012
versión del manual 01



Índice del usuario

1. Introducción.....	3
2. Ejecutar el programa MapBD.....	4
1º Paso:.....	4
2º Paso:.....	7
3. Módulos.....	7
4. Clases.....	8
.Abrir().....	9
.Cerrar().....	9
.Total().....	9
.Insertar().....	9
.Modificar{nombre de campo}(id).....	10
.Borrar{nombre de campo}(id).....	10
.BuscarContenido{nombre del campo}(valor, opcional orden de campo).....	10
.BuscarIgual{nombre del campo}(valor, opcional orden de campo).....	11
.BuscarMenorQue{nombre del campo}(valor, opcional orden de campo).....	11
.BuscarMayorQue{nombre del campo}(valor, opcional orden de campo).....	11
.BuscarEntre{nombre del campo}(valormin, valormax, opcional orden de campo).....	12
.SQL(sentencia_sql)	12
.informe().....	13
.contenido().....	13
.mostrarRegistro(numero AS Integer, grid AS GridView, OPTIONAL sqlcadena AS String) AS Result.....	13
4. Formateo Automático de Gridviews.....	14
.gridFormatearColumnas(grid AS GridView) AS gridview.....	14
.gridFormatearFilas(grid AS GridView) AS gridview	14
.gridResultanteSQL(res AS result, grid AS GridView) AS gridview	15
5. Otros Funciones no incluidas.....	16
Rellenar un gridviews con el evento Data:.....	16
Código del grid_Data.....	16
Barra de botones.....	16
6. ¿Como añadir más funciones a las clase2?.....	19
Heredar las clases2 existentes ¿por qué a aconsejo hacerlo así? ¿que ventajas tiene?.....	20
7. Programas de ejemplo.....	21
8. Link de descargas:.....	21

1. Introducción

El programa MapBD, se ha realizado para simplificar el trabajo del programador, creando automáticamente el código fuente necesario para agilizar y facilitar las tareas más comunes que hacemos con la base de datos (B.D a partir de ahora en este texto), sus tablas y campos que la componen.

Esta versión funciona para mapear B.D del tipo SQLite3. (ver nota 1 para uso con MySQL)

El programa lee una B.D dada por el usuario, y crea en el directorio que le indiquemos, una serie de módulos y clases para manejarla.

Nota:

1. Si tenemos una base de datos tipo MySQL, podemos hacer que este programa cree también el código fuente para manejarla, de la siguiente manera:

1.1) Tenemos que crear una base de datos tipo SQLite3 con la misma estructura de tablas y campos, que tenga la base de datos MySQL.

1.2) Luego usar el programa con esta base. (MapBD)

1.3) Tenemos que editar el módulo generado (“conectar”), irnos al método abrir(), y reescribirlo, dándole las opciones (nombre, hosting, password, etc), para que se pueda conectar. El resto de código es válido ya que se usan sentencias SQL.

Ejemplo:

SQLite3:

Este es el método Abrir() que abre una base de datos llamada “ejemplo”, situada en la carpeta de usuario de usuario (por ejemplo: “/home/usuario/ejemplo”)

```
PUBLIC FUNCTION abrir(conexion AS
Connection) AS Connection
' hace la conexion a la base de datos
IF NOT Exist(user.home & "/ControlConsulta")
THEN
  COPY "ControlConsulta" TO user.home &
"/ControlConsulta"
ENDIF
conexion.Name = user.home & "/ControlConsulta"
conexion.host = user.home & "/ControlConsulta"
conexion.Type = "SQLite3"
conexion.open
RETURN conexion
END
```

MySQL:

Asi seria el metodo Abrir() para una base de datos tipo MySQL, llamada “ejemplo”, que esta en nuestro ordenador (“localhost”)

```
PUBLIC FUNCTION abrir(conexion AS
Connection) AS Connection
' hace la conexion a la base de datos
conexión.Login = “administrador”
conexión.Password = “clave”
conexión.Name = "ejemplo"
conexión.Type = "mysql"
conexión.Host = "localhost"
conexión.Open
Return conexión
End
```

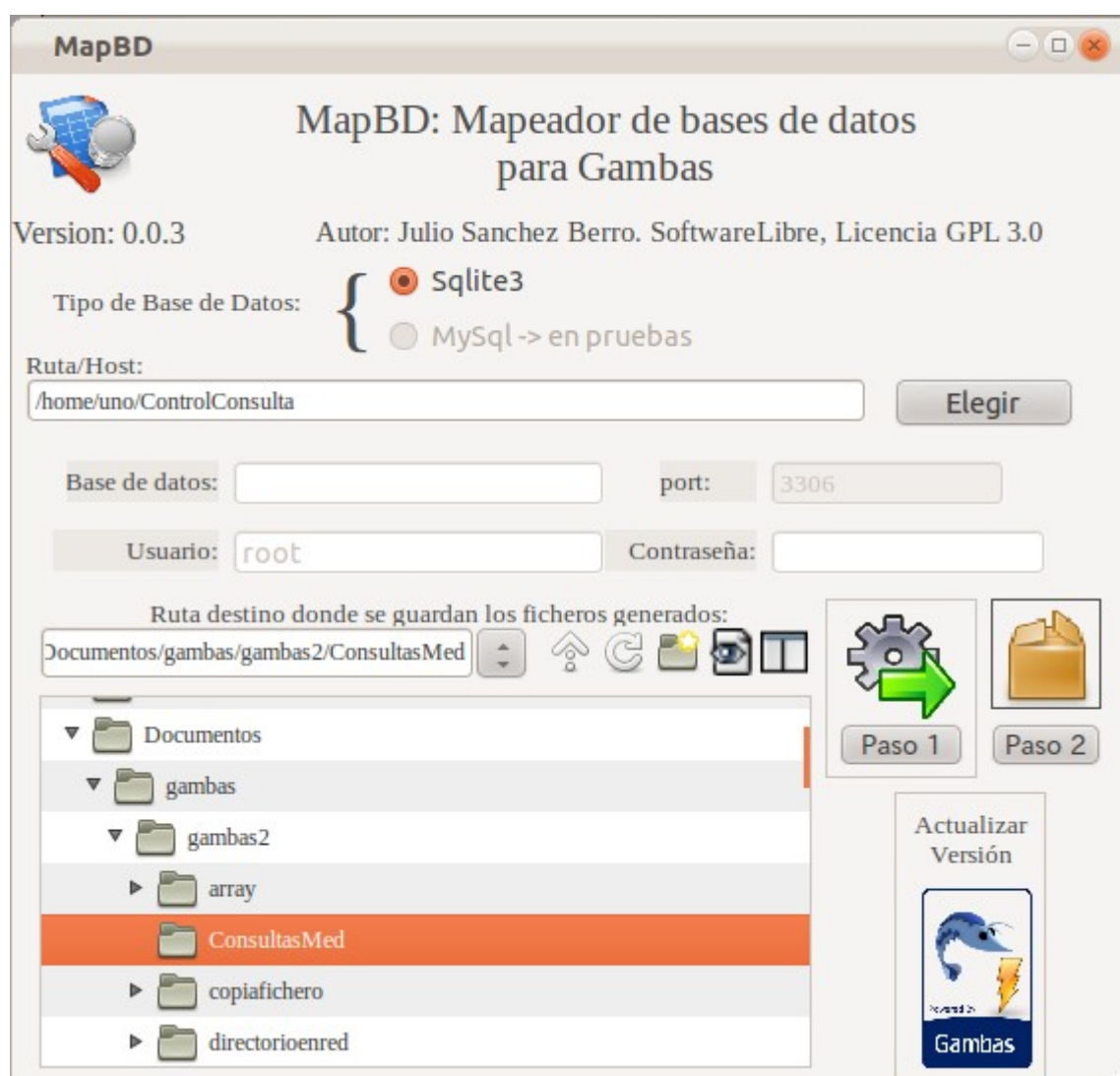
2. Ejecutar el programa MapBD

El programa MapBD, va a leer la base de datos que le indiquemos y va a crearnos las clases con las que trabajaremos en nuestro programa.

El programa funciona en dos pasos:

1º Paso:

Le decimos que base de datos vamos a leer (ejemplo: /home/uno/ControlConsulta) y donde se van a añadir las clases creadas (ejemplo: /home/uno/Documentos/gambas2)

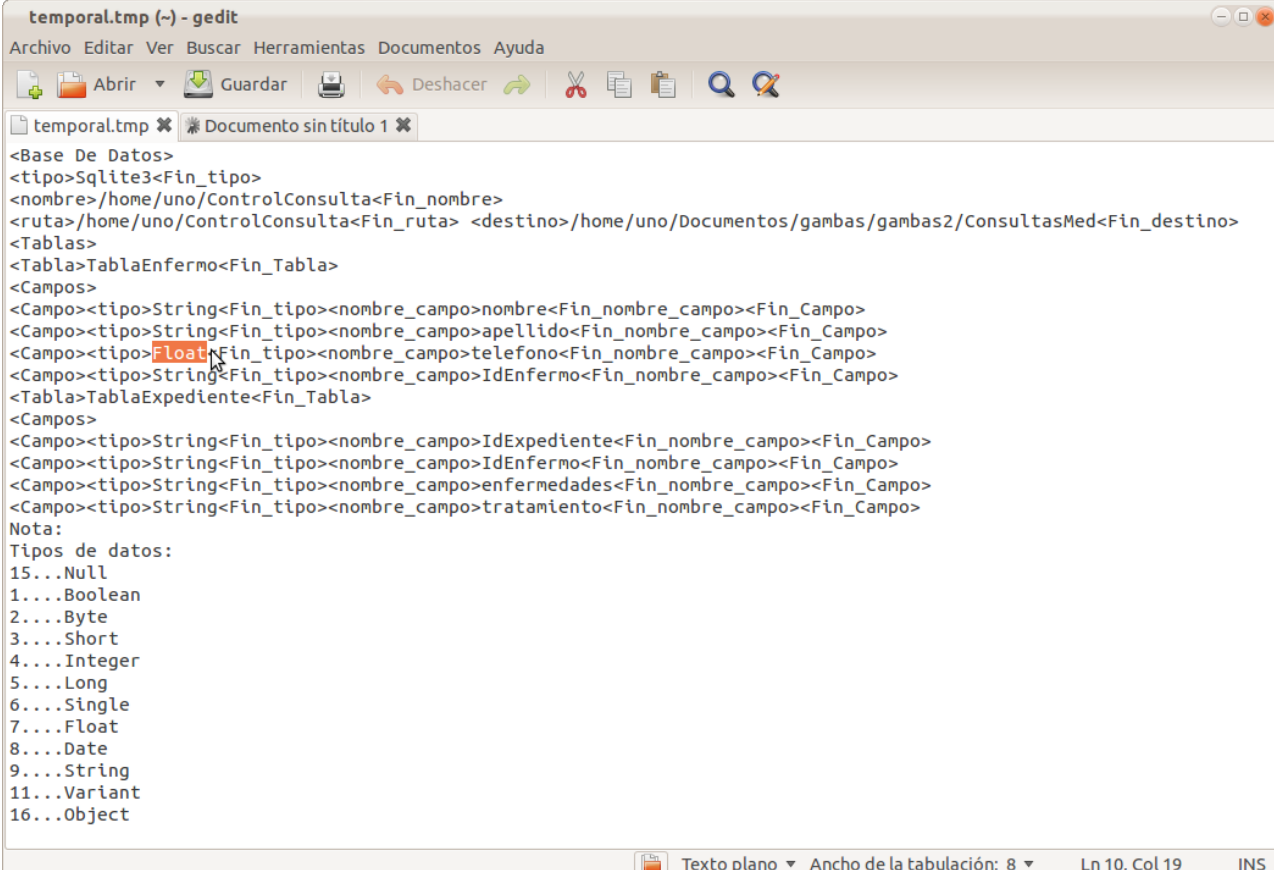


Nos va a generar un fichero “temporal.tmp”, el cual contiene los datos de las tablas y campos de la base de datos, y nos lo abre en el editor gedit, para que lo revisemos.

Cuando digo **“revisión”**, es para comprobar que el tipo de dato del campo es el correcto.

Tendrás que rectificar el tipo en los **datos numéricos y fechas**, ya que los pone (algunas veces) como si fueran **String**.

En la captura de la imagen siguiente se ve la zona donde he cambiado y escrito **Float**, el resto del archivo esta correcto.



```
<Base De Datos>
<tipo>Sqlite3<Fin_tipo>
<nombre>/home/uno/ControlConsulta<Fin_nombre>
<ruta>/home/uno/ControlConsulta<Fin_ruta> <destino>/home/uno/Documentos/gambas/gambas2/ConsultasMed<Fin_destino>
<Tablas>
<Tabla>TablaEnfermo<Fin_Tabla>
<Campos>
<Campo><tipo>String<Fin_tipo><nombre_campo>nombre<Fin_nombre_campo><Fin_Campo>
<Campo><tipo>String<Fin_tipo><nombre_campo>apellido<Fin_nombre_campo><Fin_Campo>
<Campo><tipo>Float<Fin_tipo><nombre_campo>telefono<Fin_nombre_campo><Fin_Campo>
<Campo><tipo>String<Fin_tipo><nombre_campo>IdEnfermo<Fin_nombre_campo><Fin_Campo>
<Tabla>TablaExpediente<Fin_Tabla>
<Campos>
<Campo><tipo>String<Fin_tipo><nombre_campo>IdExpediente<Fin_nombre_campo><Fin_Campo>
<Campo><tipo>String<Fin_tipo><nombre_campo>IdEnfermo<Fin_nombre_campo><Fin_Campo>
<Campo><tipo>String<Fin_tipo><nombre_campo>enfermedades<Fin_nombre_campo><Fin_Campo>
<Campo><tipo>String<Fin_tipo><nombre_campo>tratamiento<Fin_nombre_campo><Fin_Campo>
Nota:
Tipos de datos:
15...Null
1...Boolean
2...Byte
3...Short
4...Integer
5...Long
6...Single
7...Float
8...Date
9...String
11...Variant
16...Object
```

Nota:

Para más detalles e información sobre este, a título informativo:

La estructura de este archivo es muy simple:

1º: Indica que es bases de datos: <Base de Datos>

2º: Indica el tipo de la base de datos, entre la etiquetas “<tipo>” y “<Fin_tipo>”: Sqlite3

3º: Escribe el nombre, entre las etiquetas “<nombre>” y “<Fin_nombre>”:

/home/uno/ControlConsulta

4º: Escribe la ruta, entre las etiquetas “<ruta>” y “<Fin_ruta>”: /home/uno/ControlConsulta

Ademas en esa misma linea se indica entre las etiquetas “<destino>” y “<Fin_destino>” donde se guardarán las clases creadas (normalmente elegiremos el directorio del proyecto)

Luego empieza a definir las tablas <Tablas>

Pone entre las etiquetas <Tabla> y <Fin_Tabla>, el nombre de la tabla

Y entre las etiquetas “<Campo>” y “<Fin_campo>” define el **tipo de datos** (entre las etiquetas “<Tipo>” y “<Fin_tipo>”) y el **nombre del campo** (entre las etiquetas “<nombre_campo>” y “<Fin_Nombre_campo>”)

Lo importante es que los tipos de datos sean los correctos.

El mismo archivo informa de los tipos de datos admisibles:

Null , Boolean , Byte , Short , Integer , Long , Single , Float , Date , String , Variant , Object

Para dudas ver: <http://gambasdoc.org/help/cat/datatypes?es&v3>

Tipo de dato	Descripción	Valor por defecto	Tamaño en memoria
Boolean	Verdadero o falso.	FALSE	1 byte
Byte	0...255	0	1 byte
Short	-32.768...+32.767	0	2 bytes
Integer	-2.147.483.648...+2.147.483.647	0	4 bytes
Long	- 9.223.372.036.854.775.808...+9.223.372.036.854.775.807	0	8 bytes
Single	Como el tipo <i>float</i> de C.	0.0	4 bytes
Float	Como el tipo <i>double</i> de C.	0.0	8 bytes
Date	Fecha y hora, cada una almacenada en un entero.	NULL	8 bytes
String	Una cadena con un número variable de caracteres.	NULL	4 bytes
Variant	Cualquier tipo de dato.	NULL	12 bytes
Object	Referencia anónima a un objeto.	NULL	4 bytes
Pointer	Una dirección de memoria.	0	4 bytes

2º Paso:

Ahora pulsamos el botón del paso 2:



Se van a crear las clases en la ruta definida anteriormente en el programa (en mi caso elegí: /home/uno/Documentos/gambas2/ConsultasMed)

Nota importante:

Comentaros también que tanto la base de datos como el archivo temporal.tmp (este con el nombre de “esquema.MpBd”), se van a copiar dentro de la carpeta que hemos indicado.

3. Módulos

Crea un modulo llamado “conectar” que contiene un procedimiento para abrir la B.D (ver nota 1), que lo van a usar las clases que a continuación describo.

La definición de este procedimiento es la siguiente

PUBLIC FUNCTION abrir(conexion **AS Connection**) **AS Connection**

La llamada a esta función la podemos hacer como sigue:

hcon=**conectar.abrir**(hcon)

Os lo comento de modo informativo, ya que las clases son las que se van a encargar de abrir la base de dato directamente.

4. Clases

Por cada *tabla* que contenga la B.D, va a crear dos clases.

Aprovecha el nombre de la tabla para nombrar las clases.

Clase1: {nombre_tabla}Registro

Clase2: {nombre_tabla}

Las **clase1**, tiene unas propiedades que coinciden con el nombre de cada campo de tabla.

Tabla: TEnfermo	Campo	Clase1: TEnfermoRegistro	Propiedades
	nombre		nombre
	apellido		apellido
	idDNI		idDNI

Las clase2, hereda a la clase 1, por lo tanto, contiene todas las propiedades de esta.

Además en las clase2, tiene los métodos más habituales para trabajar con los datos de la B.D.

En los programas, lo que haremos es hacer instancias de esta clase2, para trabajar con la B.D.

Ejemplo de instancia:

DIM enfer **AS NEW** TEnfermos

Ejemplo de los nombres que se crean:

Nombre de la BD	Nombre de la Tabla	Clase1	Clase2 (hereda de Clase1)
ConsultasMedicas	TEnfermos	TEnfermosRegistro	TEnfermos
	TExpediente	TExpedienteRegistro	TExpediente
	TCitas	TCitasRegistro	TCitas

Los métodos de las Clase2.

.Abrir()

Se encarga de abrir la base de datos. Como ves no hay que introducir nombre, ni ruta, ni tipo. Todo lo se ha hecho automáticamente, por el programa MapBD.

Este método lo usaremos cuando abramos la bd, para poder empezar a trabajar con ella.

Ejemplo:

```
DIM enfer AS NEW TEnfermos
enfer.abrir()
```

.Cerrar()

Cierra la base de datos.

Ejemplo:

```
DIM enfer AS NEW Tenfermos
....
enfer.cerrar()
```

.Total()

Devuelve el numero de registros que contiene la tabla. Devuelve un integer

Ejemplo:

```
Dim numero as integer
....
numero=enfer.total()
```

.Insertar()

Devuelve el resultado de la inserción en la tabla de los propiedades de la clase2

Ejemplo:

```
DIM enfer AS NEW Tenfermos
enfer.nombre="Julio"
enfer.apellido="Sanchez"
enfer.idDNI="34000233"
```

```
enfer.abrir()  
enfer.insertar() 'inserta los datos de “Julio” y “Sanchez” en los campos “nombre” y  
“apellido” de la tabla “Tenfermos”
```

.Modificar{nombre de campo}(id)

Este procedimiento lo que devuelve es el result de la modificación de todas las propiedades menos la que indique el {nombre de campo} que es la que nos sirve para localizar el registro o registros (clausula WHERE de SQL)

Se van a crear tantos procedimientos .Modificar{nombre de campo}, como campos haya en la tabla.

Ejemplo:
DIM enfer **AS NEW** Tenfermos
enfer.nombre=”Julio”
enfer.apellido=”Gomez” 'anteriormente se defino como “Sanchez”
enfer.ModificaridDNI(”34000233”)
'modifica el registro que tenga como id = “34000233”, poniendo el campo nombre como “Julio” y apellido como “Gomez”. (esta cambiando el apellido “Sanchez” por “Gomez”)

.Borrar{nombre de campo}(id)

Devuelve el resultado del borrado de los registros de la Tabla, que sean igual al id dado.

Se van a crear tantos procedimientos .Borrar{nombre de campo}, como campos haya en la tabla.

Ejemplo:
DIM enfer **AS NEW** Tenfermos
enfer.Borrannombre(”Julio”)
'borra todos los registro de la B.D, cuyo campo nombre sea igual a “Julio”

.BuscarContenido{nombre del campo}(valor, opcional orden de campo)

Devuelve el resultado de los registros de la tabla, que en el campo {nombre del campo} **contengan** el valor indicado. También se puede poner opcionalmente el campo para que se ordene la salida.

Se van a crear tantos procedimientos .BuscarContenido{nombre de campo}, como campos haya en la tabla.

Ejemplo:

DIM enfer AS NEW Tenfermos

DIM hres AS Result

hres=enfer.BuscarContenido**nombre**("ul") ' devuelve todos los registros que tiene en el campo nombre la palabra "ul".

.BuscarIgual{nombre del campo}(valor, opcional orden de campo)

Devuelve el resultado de los registros de la tabla, que en el campo {nombre del campo} **igual** el valor indicado. También se puede poner opcionalmente el campo para que se ordene la salida.

Se van a crear tantos procedimientos .BuscarIgual{nombre de campo}, como campos haya en la tabla.

Ejemplo:

DIM enfer AS NEW Tenfermos

DIM hres AS Result

hres=enfer.BuscarIgual**nombre**("Julio") ' devuelve todos los registros que tiene en el campo nombre la palabra "Julio".

.BuscarMenorQue{nombre del campo}(valor, opcional orden de campo)

Devuelve el resultado de los registros de la tabla, que en el campo {nombre del campo} **menor que** el valor indicado. También se puede poner opcionalmente el campo para que se ordene la salida.

Se van a crear tantos procedimientos .BuscarMenorQue{nombre de campo}, como campos haya en la tabla.

Ejemplo:

DIM enfer AS NEW Tenfermos

DIM hres AS Result

hres=enfer.BuscarMenorQue**idDNI**(55) ' devuelve todos los registros que tiene en el campo idDNI, sea menor que 55

.BuscarMayorQue{nombre del campo}(valor, opcional orden de campo)

Devuelve el resultado de los registros de la tabla, que en el campo {nombre del campo} **mayor que** el valor indicado. También se puede poner opcionalmente el

campo para que se ordene la salida.

Se van a crear tantos procedimientos `.BuscarMayorQue{nombre de campo}`, como campos haya en la tabla.

Ejemplo:

```
DIM enfer AS NEW Tenfermos
```

```
DIM hres AS Result
```

```
hres=enfer.BuscarMayorQueidDNI(55) ' devuelve todos los registros que cuyo  
campo idDNI, sea mayor que 55
```

.BuscarEntre{nombre del campo}(valormin, valormax, opcional orden de campo)

Devuelve el resultado de los registros de la tabla, que en el campo {nombre del campo} este **entre el valor mínimo y el valor máximo**. También se puede poner opcionalmente el campo para que se ordene la salida.

Se van a crear tantos procedimientos `.BuscarEntre{nombre de campo}`, como campos haya en la tabla.

Ejemplo:

```
DIM enfer AS NEW Tenfermos
```

```
DIM hres AS Result
```

```
hres=enfer.BuscarEntreidDNI(25,55) ' devuelve todos los registros que cuyo campo  
idDNI, sea este entre 25 y 55
```

.SQL(sentencia_sql)

Devuelve el resultado de la consulta realizada.

Ejemplo:

```
Dim enfer as New Tenfermos
```

```
Dim hres as Result
```

```
Dim sentencia_sql as string
```

```
sentencia_sql="Select * From TablaEnfermo WHERE nombre like '%u%' "
```

```
hres=enfer.sql(sentencia_sql)
```

```
' devuelve todos los registros que cumplan la sentencia_sql
```

.informe()

Devuelve un texto con la información de la tabla, campos, y tipos, y también lo escribe en la consola.

Ejemplo:

Dim enfer as New Tenfermos

enfer.informe()

'Se escribe la consola:

Base de datos: ControlConsulta

Tabla: TablaEnfermo

Campo: nombre Tipo: String

Campo: apellido Tipo: String

Campo: telefono Tipo: Float

Campo: IdEnfermo Tipo: String

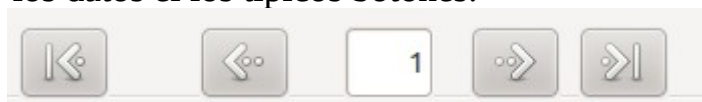
.contenido()

Devuelve el contenido de la tabla (sentencia SQL="Select * From Tabla")

.mostrarRegistro(numero AS Integer, grid AS GridView, OPTIONAL sqlcadena AS String) **AS Result**

Devuelve el registro nº numero, escribiendo en el grid dado, aplicando la sentencia .contenido() o opcionalmente la que se introduzca (sqlcadena)

Esta función se usa conjuntamente con el grid formateado por **.gridFormatearFilas**, para ir presentando los datos en los típicos botones:



4. Formateo Automático de Gridviews

.gridFormatearColumnas(grid AS GridView) AS gridview

Recibe un gridview (grid), y devuelve un gridview, formateado, con los títulos de las **columnas** igual a los nombres de los campos, con anchos fijos

Ejemplo:

```
gridviewColumnas = registrohola.gridFormatearColumnas(gridviewColumnas)
```

	id	texto	fecha	datos
1	230	eva	21/06/1212	0
2	2344	23422	20/01/1902	2
3	11123	22	21/06/1212	0
4	22	adiosAdios	18/07/7005	2230
5	22332	texto	21/06/1212	120

.gridFormatearFilas(grid AS GridView) AS gridview

Recibe un gridview (grid), y devuelve un gridview, formateado, con los títulos de las **filas** igual a los nombres de los campos.

Ejemplo:

```
gridviewFilas = registrohola.gridFormatearFilas(gridviewFilas)
```

Frame1: GridFilas		
	Campos	Registro
1	id	230
2	texto	eva
3	fecha	26/05/12415 23:00:00
4	datos	0

.gridResultanteSQL(res AS result, grid AS GridView) AS gridview

Con el resultado de una consulta (res), se va a formatear (pone las columnas con el nombre de los campos) y rellena el gridview dado (grid)

Ejemplo de uso:

```
DIM SQLString AS String
```

```
SQLString = "Select * From TablaEnfermo WHERE nombre like '%u%' "
```

'aquí es donde puedo redefinir la sentencia SQL

```
SQLString = InputBox("Introduce la sentencia SQL: ", SQLString)
```

```
TextLabelSentenciaSQL.text = SQLString
```

```
GridView2 = RegistroEnfermo.GridResultanteSQL(RegistroEnfermo.sql(SQLString), GridView2)
```

Resultado de una sentencia SQL:

```
Select apellido,nombre,telefono From TablaEnfermo  
WHERE nombre like '%u%'
```

	apellido	nombre	telefono
1	sanchez	julio	0

5. Otros Funciones no incluidas

Como comentario final, os dejo varias funciones que son muy útiles, que hay que configurar en los propios formulario que usemos.

Rellenar un gridviews con el evento Data:

Para que se produzca el evento Data podemos variar el numero de filas (insertamos, borramos datos), y salta automáticamente. Para editar, yo lo que hago, es variar “artificialmente” el numero de filas, y luego vuelvo a ponerlas en el numero real.

```
resultado = registrohola.contenido()  
gridviewColumnas.Rows.Count = resultado.Count
```

Para editar, yo lo que hago, es variar “artificialmente” el numero de filas, y luego vuelvo a ponerlas en el numero real.

```
gridviewColumnas.Rows.Count = 0  
resultado = registrohola.contenido()  
gridviewColumnas.Rows.Count = resultado.Count  
'se produce el evento DATA
```

Código del grid_Data

```
PUBLIC SUB gridviewColumnas_Data(Row AS Integer, Column AS Integer)  
    resultado.moveTo(row)  
    gridviewColumnas.Data.text = Str(resultado[gridviewColumnas.Columns[column].text])  
    IF row MOD 2 = 0 THEN gridviewColumnas.Data.Background = Color.LightBackground  
END
```

Barra de botones

Estos códigos no se incluyen automáticamente por MapBD, pero son muy usados, y creo conveniente exponerlo:



'Este es un código de ejemplo.

Public contador as integer

Public registrohola as Clase2

```
PUBLIC SUB ToolButtonPrimero_Click()  
contador = 0  
registrohola.mostrarRegistro(contador, gridviewFilas)  
gridviewFilas.Refresh  
ValueBoxcontador.value = contador + 1  
END
```

```
PUBLIC SUB ToolButtonAtras_Click()  
contador -= 1  
IF contador > -1 THEN  
registrohola.mostrarRegistro(contador, gridviewFilas)  
gridviewFilas.Refresh  
ValueBoxcontador.value = contador + 1  
ELSE  
contador += 1  
ENDIF
```

```
PUBLIC SUB ToolButtonAdelante_Click()  
contador += 1  
IF contador < (registrohola.Total()) THEN  
registrohola.mostrarRegistro(contador, gridviewFilas)  
gridviewFilas.Refresh  
ValueBoxcontador.value = contador + 1  
ELSE  
contador -= 1  
ENDIF  
END  
END
```

```
PUBLIC SUB ToolButtonUltimo_Click()  
contador = registrohola.Total() - 1  
registrohola.mostrarRegistro(contador, gridviewFilas)  
gridviewFilas.Refresh  
ValueBoxcontador.value = contador + 1  
END
```

'para cuando se escribe el numero del registro

```
PUBLIC SUB ValueBoxcontador_KeyPress()
```

DIM numero AS Integer

IF Key.Code = Key.enter OR Key.code = Key.return THEN

numero = Val(ValueBoxcontador.text)

IF numero > 0 OR numero <= (registrohola.Total() - 1 - 1) THEN

registrohola.mostrarRegistro(contador, gridviewFilas)

gridviewFilas.Refresh

contador = numero - 1

ENDIF

ENDIF

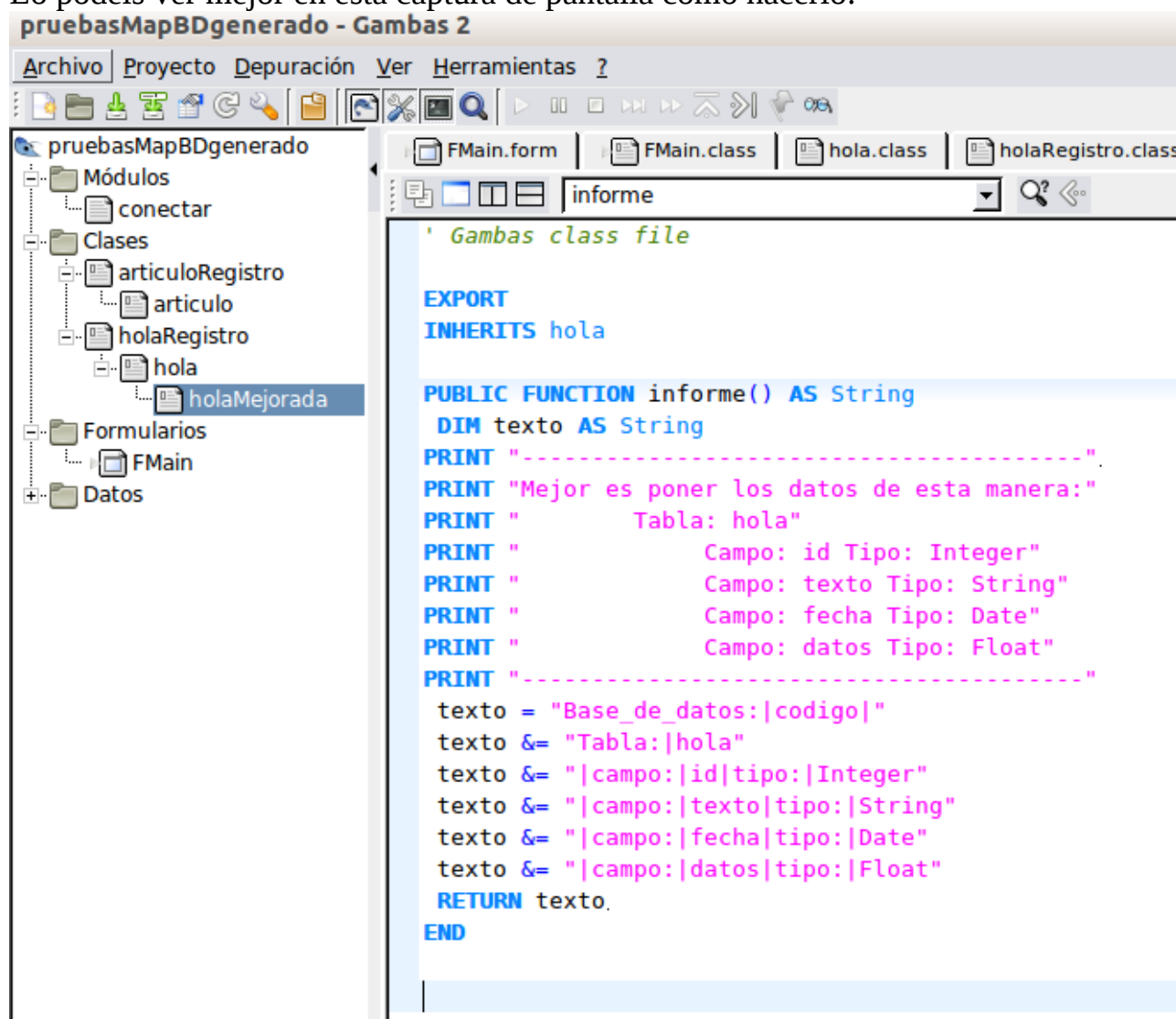
END

6. ¿Como añadir más funciones a las clase2?

Evidentemente, las clases que se crear automáticamente, a la larga se quedan cortas o hay que modificarlas por algún motivo (un bug, una mejora, adaptarlas a nuestro programa en concreto).

La mejor forma de hacer esto, es aprovechar las opciones que nos da la programación orientada a objetos (POO): crear otra clase (llamemo la “Mejorada” que herede la clase2), donde se incluyan las mejoras o se redefinan los métodos.

Lo podéis ver mejor en esta captura de pantalla como hacerlo:



Como veis, la clase “holaMejorada”, hereda de la clase “hola”, y redefinimos la función informe().

Ahora en el código de Fmain, cambiamos la definición, de la siguiente línea:

```
PUBLIC registrohola AS NEW hola
```

por

```
PUBLIC registrohola AS NEW holaMejorada
```

Y ya los informes se escribirán como define la clase “hola Mejorada” !!!

Ejemplo: Al ponemos el siguiente código:

```
registrohola.informe()
```

escribirá en la consola:

Mejor es poner los datos de esta manera:

Tabla: hola

Campo: id Tipo: Integer

Campo: texto Tipo: String

Campo: fecha Tipo: Date

Campo: datos Tipo: Float

Heredar las clases2 existentes ¿por qué a consejo hacerlo así? ¿que ventajas tiene?

→ Si modificamos la clase2 inicial, y alguna vez se vuelve a utilizar el programa MapBD, (por ejemplo: porque hayamos ampliado la base de datos con más tablas o cambiado algo de las existentes), este programa “machacará” los archivos existentes, perdiendo los cambios realizados por ti.

→ Si se actualiza el programa MapBD, y se arreglan bug, o se ampliarán las métodos, estos cambios los puedes seguir aprovechando para el desarrollo de tu programa.

7. Programas de ejemplo

He creado dos programas de ejemplo práctico para que veáis como se usan estas clases. Lo podeis descargar desde mi blog.

<http://jsbsan.blogspot.com/>

8. Link de descargas:

[MapBD-0.0.3.tar.gz](#)

ejemplos:

[ConsultasMed-0.0.1.tar.gz](#)

[pruebasMapBDgenerado-0.0.3.tar.gz](#)

Espero que os sea útil, y cualquier duda, bug, mejoras que encontréis, hacérmela llegar por el blog (<http://jsbsan.blogspot.com/>) o el foro de gambas en español/ <http://www.gambas-es.org/>

Saludos

El autor
Julio Sánchez Berro
4 de Enero de 2012

