



Haciendo un programa y usando
la herramienta MapBD:
CONSULTA MEDICA



Autor: Julio Sánchez Berro
web: <http://jsbsan.blogspot.com/>

versión del documento: 0.0.1
7 de enero de 2012

Índice

1. Introducción.....	3
2. Definir cual es el problema que tiene que solucionar nuestra aplicación.....	4
3. Creando la base de datos.....	7
4. Creando un nuevo proyecto en Gambas: ConsultasMed.....	9
5. Usando el MapBD.....	10
1º Paso:.....	10
2º Paso:.....	13
6. Añadiendo las clases generadas en nuestro proyecto ConsultasMed.....	14
7. Creando Formularios.....	16
8. Codificando.....	19
8.1 El código de Fmain:.....	19
8.2. El código de FormEnfermos.....	20
8.3. El código del Formulario Elige Enfermo.....	23
8.4. El código del Formulario Expediente.....	24
9. Mejoras que podrías introducir en el programa.....	28
10 Anexos: Métodos que crea automáticamente MapBd 0.0.3.....	29
.Abrir().....	29
.Cerrar().....	29
.Total().....	29
.Insertar().....	29
.Modificar{nombre de campo}(id).....	29
.Borrar{nombre de campo}(id).....	30
.BuscarContenido{nombre del campo}(valor, opcional orden de campo).....	30
.BuscarIgual{nombre del campo}(valor, opcional orden de campo).....	30
.BuscarMenorQue{nombre del campo}(valor, opcional orden de campo).....	30
.BuscarMayorQue{nombre del campo}(valor, opcional orden de campo).....	30
.BuscarEntre{nombre del campo}(valormin, valormax, opcional orden de campo).....	31
.SQL(sentencia_sql)	31
.informe().....	31
.contenido().....	31
.mostrarRegistro(numero AS Integer, grid AS GridView, OPTIONAL sqlcadena AS String)	31
Formateo Automático de Gridviews.....	32
.gridFormatearColumnas(grid AS GridView) AS gridview.....	32
.gridFormatearFilas(grid AS GridView) AS gridview	32
.gridResultanteSQL(res AS result, grid AS GridView) AS gridview	32
11. Enlaces interesantes y Donde conseguir los Programas:.....	34

1. Introducción.

La mayoría de los programas que se realizan, trabajan con datos. Estos datos normalmente se gestionan a través de los gestores de bases de datos (MySQL, Sqlite, etc).

El programa MapBD, se ha realizado para simplificar el trabajo del programador, creando automáticamente el código fuente necesario para agilizar y facilitar las tareas comunes que hacemos con la base de datos, tablas y registros.

Dada una base de datos, en esta versión tiene que ser del tipo: Sqlite3 (Ver nota 1 para usarlo con MySQL) nos va a generar 2 clases por cada tabla que contenga la base de datos. Osea si una base de datos tiene 4 tablas, nos creará 8 clases. Solamente 4 de esas clases son las que vamos a utilizar en el programa que hagamos.

Vamos a ir explicando tanto el uso del programa, como del código que se genera, a través de un ejemplo.

Se van a desarrollar varias fases o etapas:

1. Definición y análisis del problema: Requisitos y requerimientos.
2. Creación de la B.D (base de datos)
3. Uso de MapBd, para generar las clases que manejen la B.D
4. Creación de los distintos formularios
5. Creación del código fuente de los formularios

Hacer una definición y análisis exhaustivo del problema, y obteniendo así los distintos diagramas, facilitará la generación de código fuente, haciendo que podamos llevar a “buen puerto” nuestro proyecto, y consiguiendo finalmente un “buen producto de software”.

Para más información:

http://es.wikipedia.org/wiki/Ingeniería_de_software

2. Definir cual es el problema que tiene que solucionar nuestra aplicación.

Esta es la definición de nuestro problema:

“En el ambulatorio ANDALUCIA, los médicos llevan el control de los pacientes mediante un archivo físico en papel. Para cualquier consulta, tienen que ir y ver el expediente de cada enfermo en el archivador. Se quiere hacer un programa que facilite el trabajo, llevándolo, por un sistema informatizado.”

Haciendo una entrevista los médicos del ambulatorio, nos informamos de “TODO” lo hacen.
Por ejemplo:

- Cada vez que entra un enfermo le dice “Buenos Dias”
 - Cada vez que hay un nuevo enfermo le hacen una ficha, con todos sus datos (nombre, apellidos, teléfono, enfermedades pasadas, tratamientos que siguen,...)
 - Cuando entra un enfermo a su consulta, miran su expediente, apuntan las nuevas observaciones, anulan o añaden tratamientos.
 - Cuando se va el enfermo, le estrecha la mano, y le dice “Que se mejore...”
- etc....

toda esta información debemos “abstraerla”, osea quitar las cosas no importantes y dejar solo lo que es importante, por ejemplo, lo de decir “Buenos Dias”, y “Que se mejore...” no lo tenemos que tener en cuenta en nuestro programa :)

Al final tendremos unos datos a manejar y unas acciones a realizar.

Por ejemplo:

Datos a manejar

De cada **enfermo** tenemos que tener su:

nombre
apellido
teléfono

Para cada enfermo, tendremos su **expediente**, donde apuntaremos:
enfermedades pasadas.
tratamiento que sigue.

Acciones a realizar:

Dar de alta a un enfermo

Dar de baja a un enfermo (esperemos que porque han sanado... :))

Modificar la ficha de un enfermo, por ejemplo, porque cambia de teléfono

Modificar un expediente, añadiendo un tratamiento.

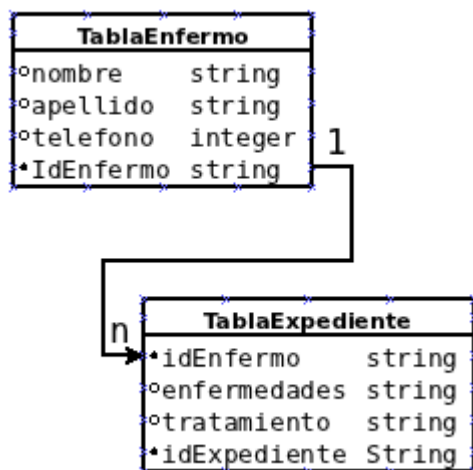
Etc...

Consulta Médica: Haciendo un programa desde cero y uso de la herramienta MapBD

Con “**Datos a manejar**”, podemos crear un primer esbozo, de como va ha ser la estructura de la base de datos:

Dos tablas (enfermo y expediente) y los campos (para enfermo: nombre, apellido, teléfono, y para expediente: enfermedades, tratamiento, los dos relacionados con un enfermo en concreto).

Podemos usar alguna herramienta gráfica ([Dia](#)), para crearnos un esquema de la base de datos, sus tablas, campos y las **relaciones** entre las tablas.



Como veis, he creado un nuevo campo “IdEnfermo”, que nos sirve para dos cosas:

- Identifica como “único” a cada enfermo (por ejemplo el nº de DNI o Cédula Ciudadana)
- Nos sirve para relacionar cada expediente con su enfermo. La relación “1 a n”, nos indica que un enfermo puede tener varios expedientes, pero un expediente solo pertenece a un enfermo.

También he creado un nuevo campo “idExpediente”, que nos sirve para identificar como “único” cada expediente. Esto nos va a facilitar la edición y borrado de un determinado expediente.

Consejo:

Crea siempre un campo en todas las tablas para que identifique como “único” cada registro. Facilitará mucho el trabajo en las rejillas/gridviews, como veremos a lo largo de este manual.

Nota:

Este problema y su análisis son muy simples, y también se puede ver de muchas formas distintas, pero nos servirá para nuestro propósito.

Los problemas “reales”, y sus análisis son muchos más complicados, y te llevarán mucho tiempo hacerlos y “optimizarlos”.

Pero sin duda, cuanto más tiempo le dediques a esta “primera etapa de desarrollo” de tu software,

vas a conseguir que las posteriores etapas sean más rápidas y que el acabado final sea útil y válido para tu cliente/usuario, evitando tener que volver hacia atrás porque te des cuenta que falta alguna opción o detalle, no analizado adecuadamente.

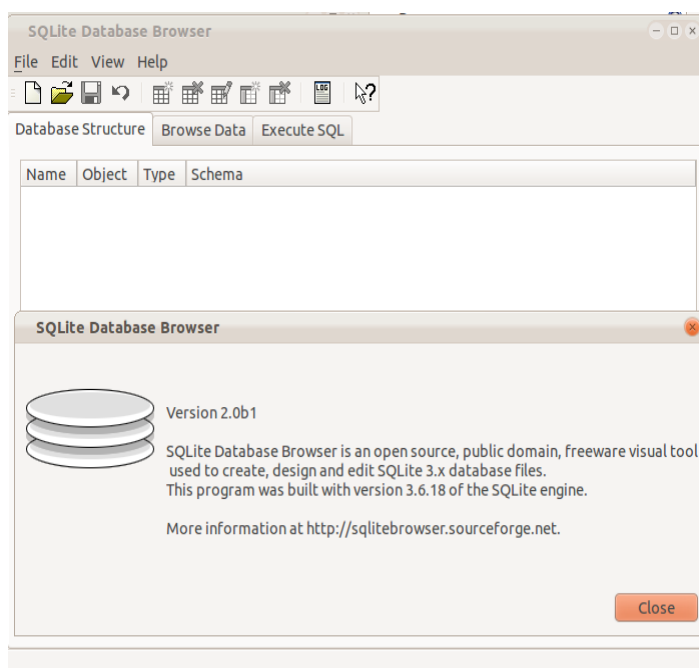
Si quieres aprender más sobre “análisis funcional”, visita los apuntes que hice:

<https://sites.google.com/site/cursofpeanalistafuncional/>

3. Creando la base de datos.

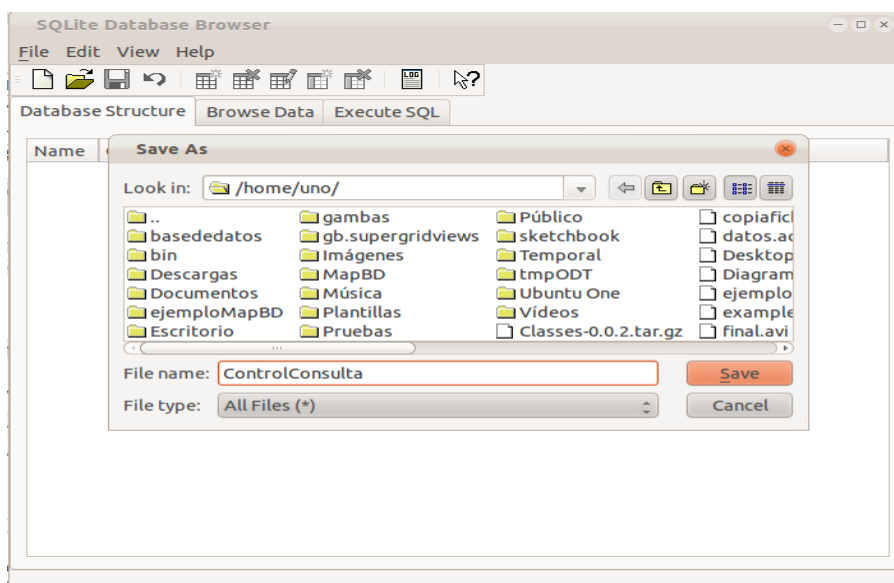
Bien, una vez que tenemos claro como es la estructura de nuestra base de datos, vamos a crearla en nuestro ordenador.

Para ello voy a utilizar el programa [SQLite Database Browser](http://sqlitebrowser.sourceforge.net).



Con el Menu File/New Database, creamos la base de datos, le ponemos de nombre “ControlConsulta”, y lo guardamos en el directorio de usuario (/home/usuario),

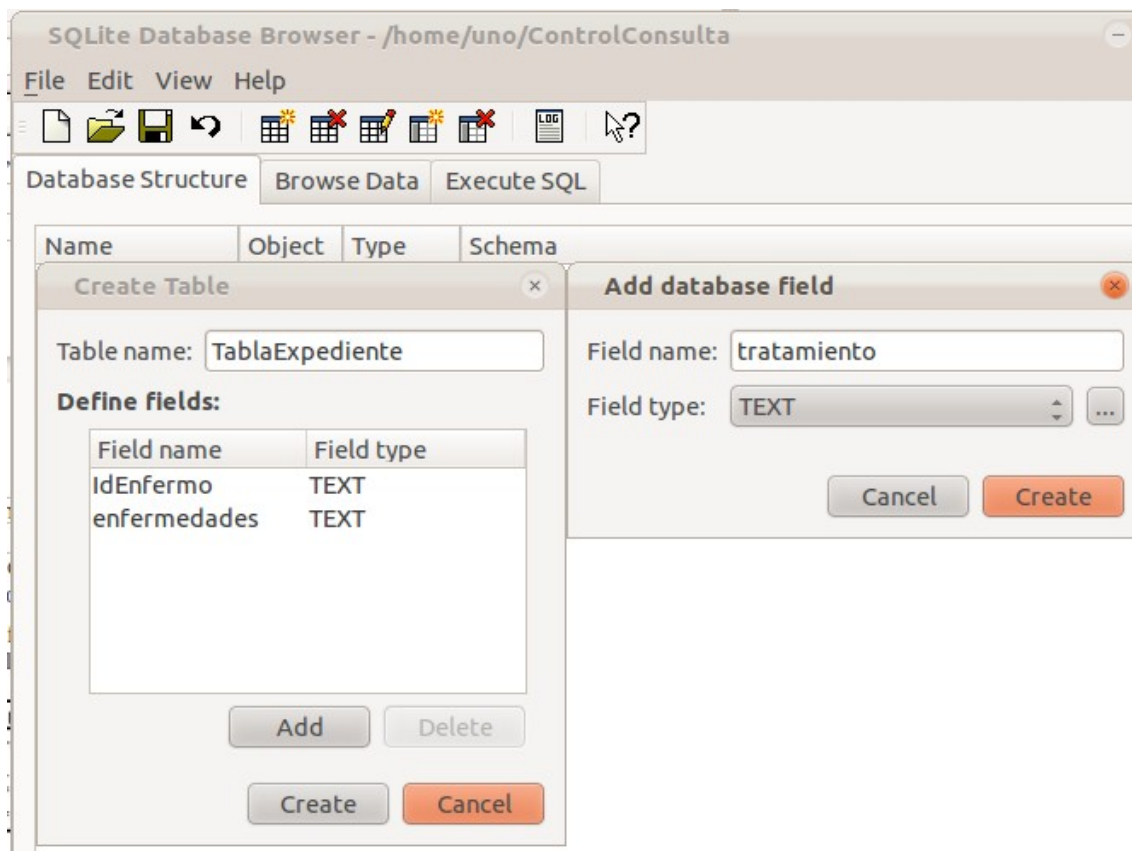
En mi caso la ruta es /home/uno:



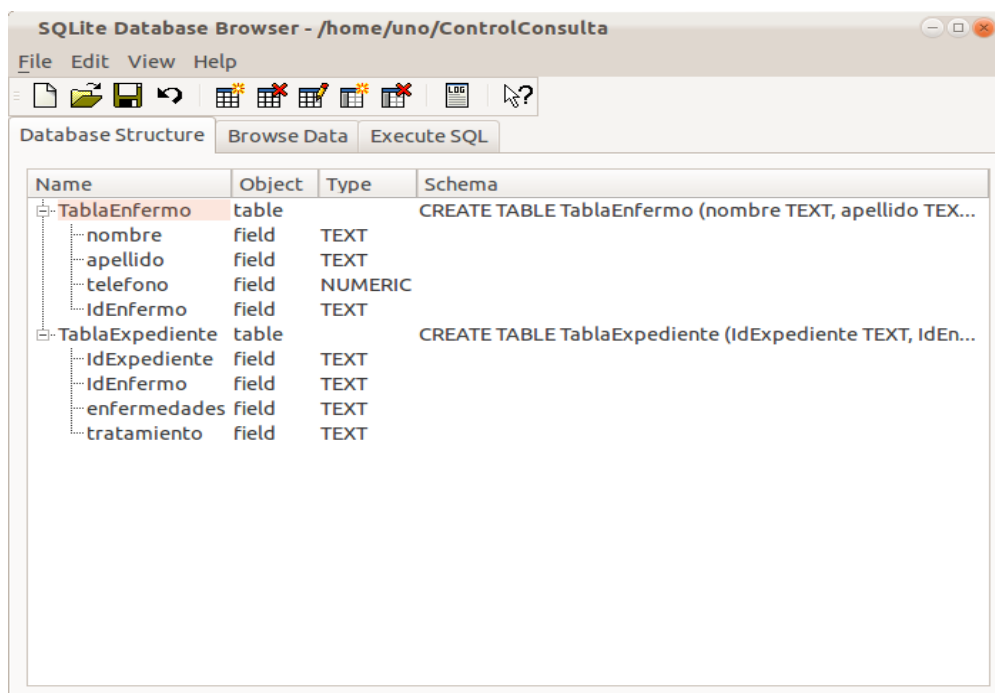
Donde se guardara la base de datos es: /home/uno/ControlConsulta

Consulta Médica: Haciendo un programa desde cero y uso de la herramienta MapBD

Una vez creada la base de datos, crearemos las tablas con sus campos, usando los botones “Create”, “Add” y eligiendo el tipo de campo (Field Type:) con el combo



Captura mientras creo la tabla de “TablaExpediente”. Del mismo modo, crearemos la tabla “TablaEnfermo”. El resultado final es:



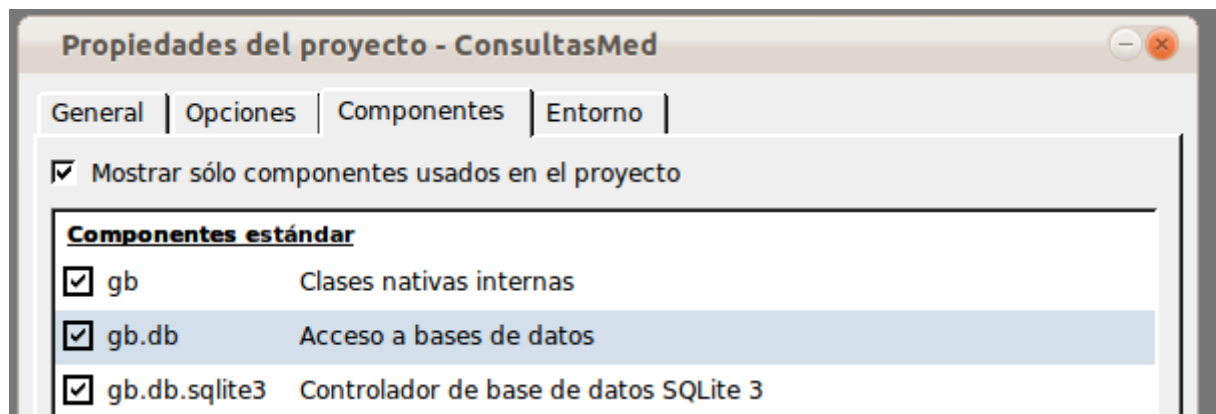
Ahora grabamos la base de datos (File / Save DataBase) y salimos del programa (File / Exit)

4. Creando un nuevo proyecto en Gambas: ConsultasMed

En este manual uso la versión de Gambas2.23 , (aunque el código fuente que crea el MapBD también es válida para el Gambas3)

Creo un proyecto con gambas con el nombre “ConsultasMed”, en la ruta (por ejemplo: /home/uno/Documentos/gambas2)

Y en Proyecto/Propiedades/Componentes, le tengo que añadir los componentes gb.db y gb.sqlite3



Ahora pasamos a ejecutar el MapBD.

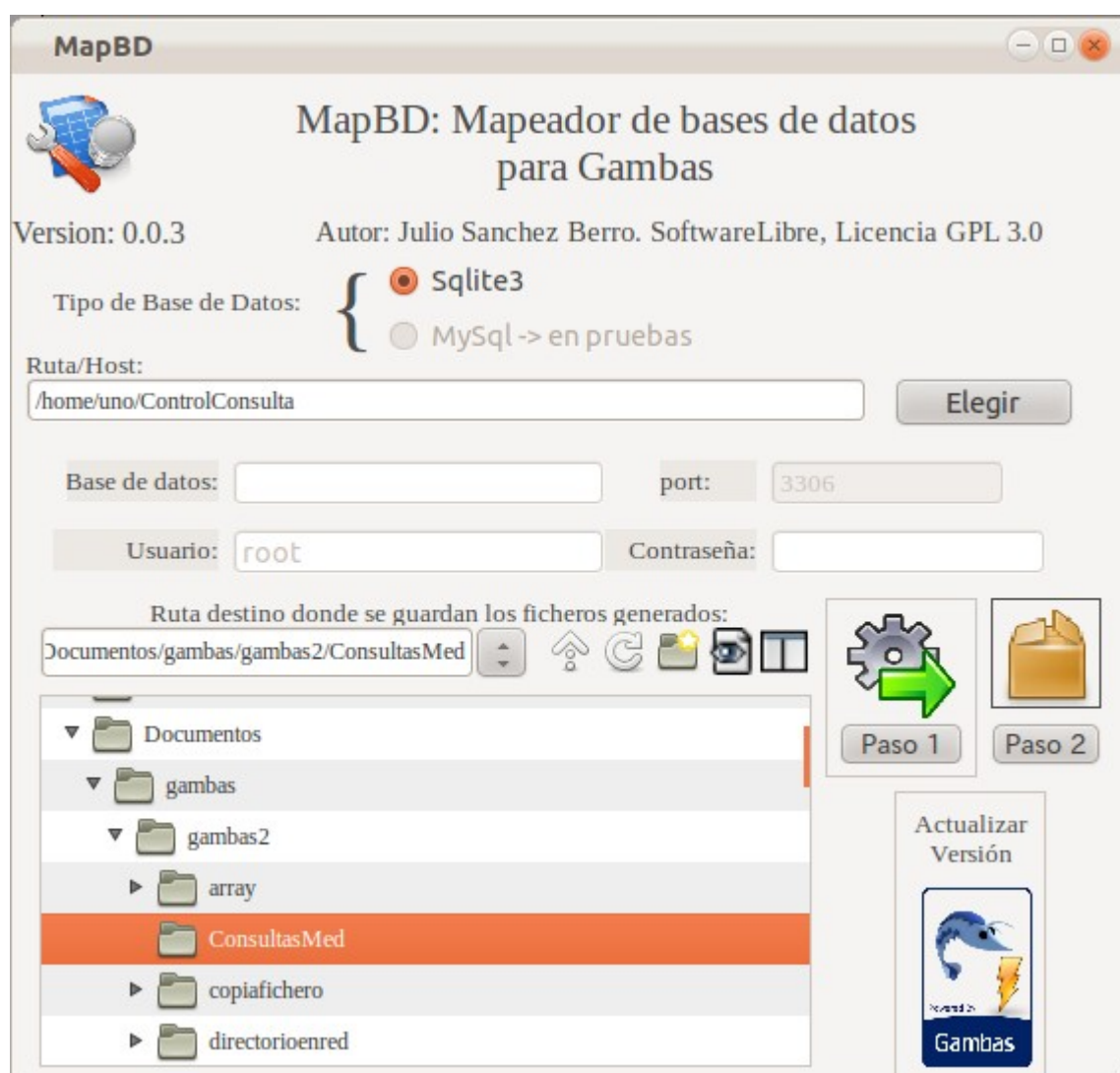
5. Usando el MapBD

Ahora vamos usar el programa MapBD, este va a leer la base de datos que hemos hecho, y va a crearnos las clases con las que trabajaremos en el programa “ConsultasMed”.

El programa funciona en dos pasos:

1º Paso:

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



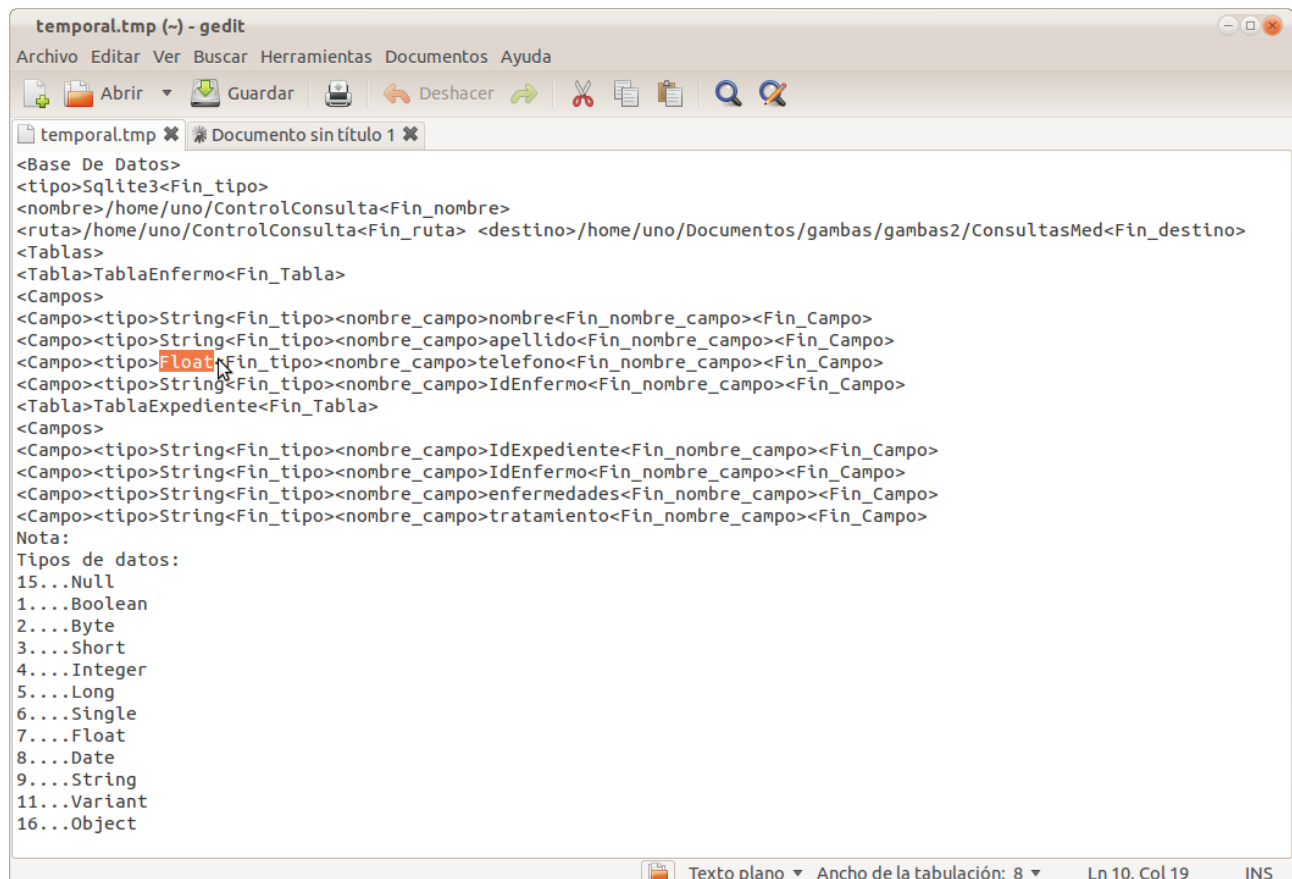
Consulta Médica: Haciendo un programa desde cero y uso de la herramienta MapBD

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 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
```

Para más detalles e información sobre este:

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

Consulta Médica: Haciendo un programa desde cero y uso de la herramienta MapBD

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:



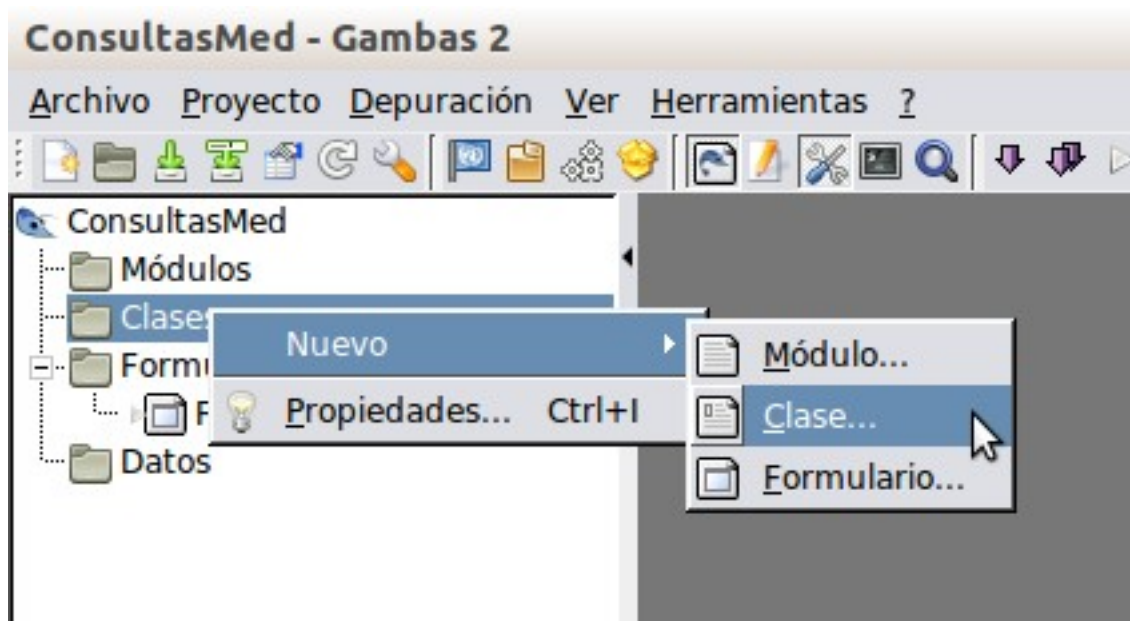
Y se van a crear las clases en la ruta definida anteriormente en el programa (en mi caso elegí: /home/uno/Documentos/gambas2/ConsultasMed)

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.**

6. Añadiendo las clases generadas en nuestro proyecto ConsultasMed

Ahora abrimos nuestro proyecto y vamos a añadir las clases creadas por el programa MapBD, para ello, hacemos click en el icono de carpeta “clases”, eliges “Nuevo” y luego “Clase...”



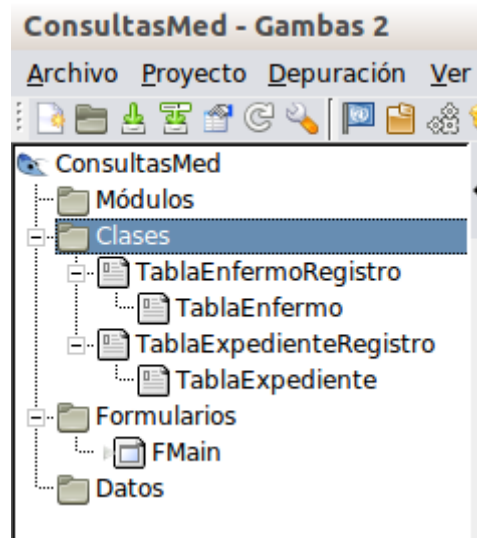
Y elegimos “existente” y nos vamos al directorio donde se hayan creado, para ir las añadiendo una a una.

Truco:

Hay también un pequeño truco, para cuando son muchos archivos, la forma anterior es muy tediosa, puedes hacerlo también con tu administrador de archivos Nautilus: copias desde el directorio donde se hayan creado las clases al directorio de tu aplicación, y luego le das a “Refrescar el Proyecto”, (o cierras y abres el proyecto de gambas), para actualizar el contenido del proyecto.

Consulta Médica: Haciendo un programa desde cero y uso de la herramienta MapBD

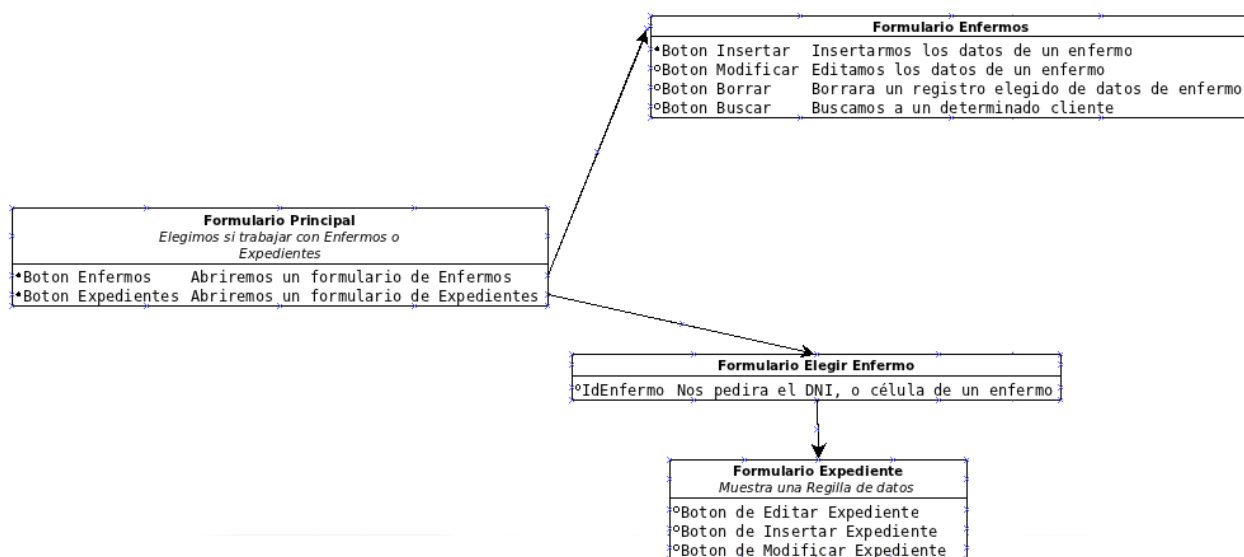
El resultado final será:



7. Creando Formularios.

7.1 Diseño de formularios

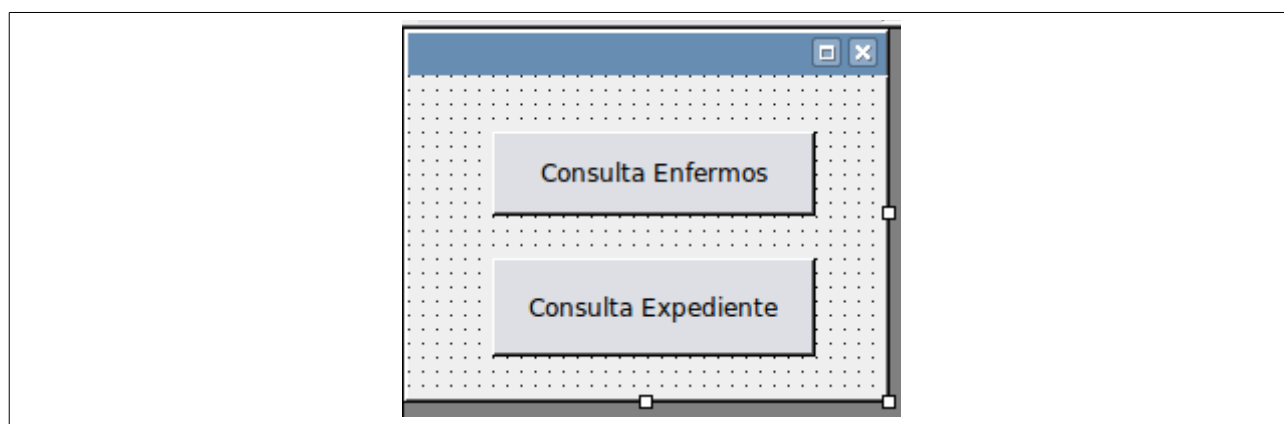
Antes de empezar a trabajar con Gambas, debemos crear un esquema para definir los formularios necesarios y que cosas deben de hacer cada uno de ellos. Esto nos ahorrará mucho trabajo de “rediseño” cuando estemos ya “codificando” (creando el código fuente) para evitar tener que volvernos atrás cuando nos demos cuenta de que nos faltan detalles.



7.2. Creando los formularios en Gambas

Ahora vamos a crear los distintos formularios:

- 1) Un formulario Fmain Principal, le añadimos dos botones, uno nos servirá para ver el formulario Enfermos y otro para ver los Expedientes.:



Consulta Médica: Haciendo un programa desde cero y uso de la herramienta MapBD

2) Creamos un formulario llamado FormEnfermos

Para hacer las altas (insertar), bajas (borrar), o modificar los datos de los Enfermos (TablaEnfermo)

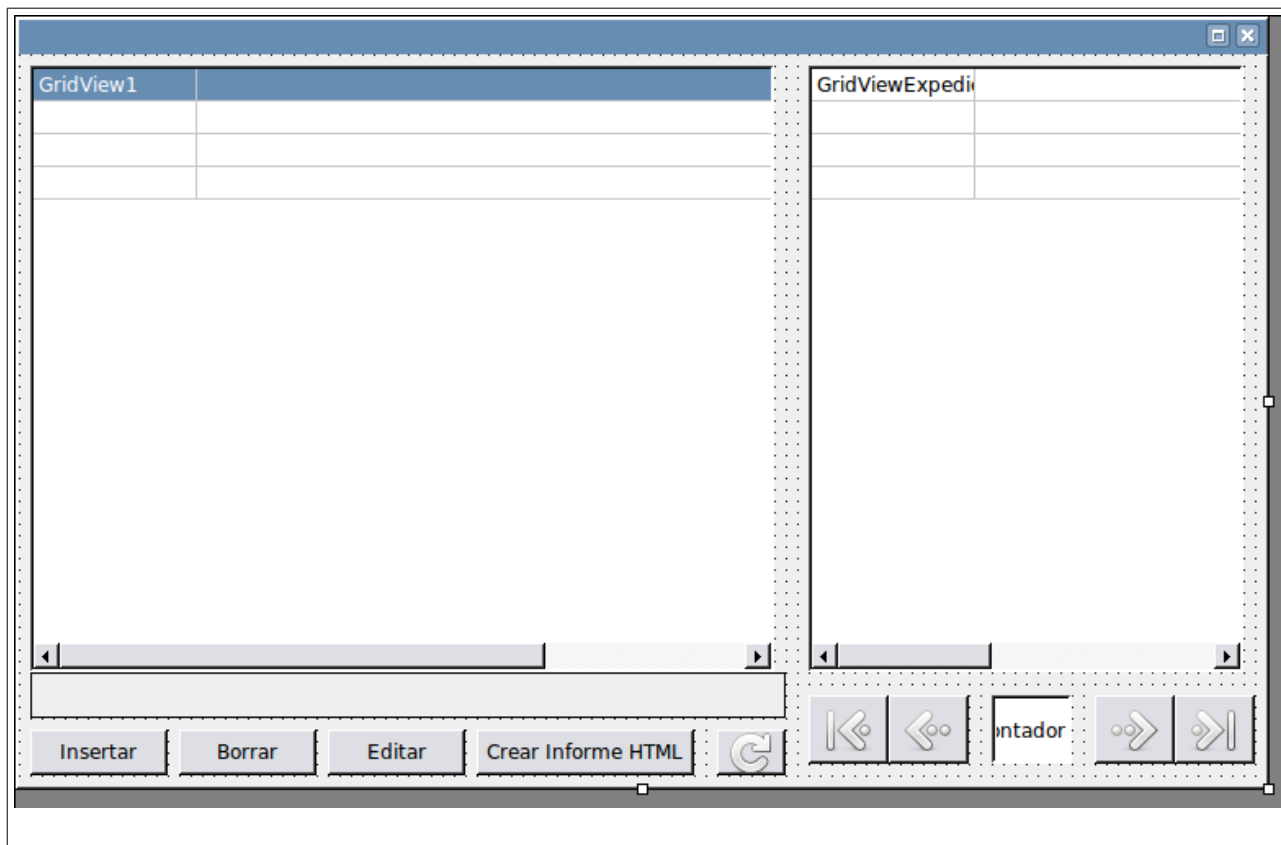
The screenshot shows a Windows Forms application window titled "FormEnfermos". The window is divided into two main sections. On the left is a large grid labeled "GridView1". On the right is a smaller grid labeled "GridView2". Above "GridView2" is a text box labeled "Resultado de una sentencia SQL:". Below the grids is a horizontal bar containing five buttons: "Insertar", "Borrar", "Editar", "Buscar", and "SQL". A circular arrow icon is positioned between the "Buscar" and "SQL" buttons. The window has a standard Windows title bar with minimize, maximize, and close buttons.

3) Creamos un formulario donde elijo al enfermo cuyo expedientes quiero ver.

The screenshot shows a Windows Forms application window titled "FormEnfermos". The window contains a single grid labeled "GridView1". Below the grid is a text box with the instruction "Haga click en la fila de la regilla, para ver el expediente del enfermo". At the bottom center is a button labeled "Cancelar". The window has a standard Windows title bar with minimize, maximize, and close buttons.

- 4) Creamos un formulario para editar el expediente de un enfermo (elegido en el formulario anterior)

En este formulario vamos a insertar, borrar, editar y crear informes (que lo podemos ver el en navegador de internet, ya que genera una salida de un archivo tipo HTML),



Nota Importante:

Cuando creamos los controles (botones, gridviews, label, etc), gambas pone nombres (propiedad **.Name**) seguidos de un número (por ejemplo: textbox1, gridview1, como podéis ver en las capturas anteriores).

Es fundamental nombrar los controles con nombres descriptivos que indiquen cual es su función. Por ejemplo el boton1, si se va a encargar de Insertar datos, su nombre adecuado seria BotonInsertar. Esto nos va a evitar mucho confusiones al leer el código, siendo la única manera de poder hacer un mantenimiento de la aplicación con garantías de éxito. ¿dentro de 3 meses, quien se va a acordar que hacia el botón1?.

Lo mismo ocurre con los nombres de variables, funciones, métodos y procedimiento, cuanto más descriptivo sea el nombre, mejor será la comprensión del código.

Además es muy importante realizar comentarios al código fuente.

Son reglas que hay que aplicar para mejorar la calidad del código. tanto como, para que todos se puedan beneficiar de nuestros programas.

8. Codificando

Vamos a empezar a realizar el código de nuestro programa.

Recuerda que gran parte del código fuente ya esta realizado (sobre el 80%) , gracias a las clases que ha creado el MapBD, que incluyen la mayoría de las funciones que nos van a hacer falta para trabajar con los registros de la Base de Datos.

Para ver más sobre que funciones y propiedades tienen las clases creadas por MapBD, revisa el documento [“Manual del MapBD”](#)

8.1 El código de *Fmain*:

Este es el código más sencillo, se trata de dos botones que nos van a abrir respectivamente los formularios de enfermos y expedientes:

```
' Gambas class file

PUBLIC SUB _new()

END

PUBLIC SUB Form_Open()
'centramos el formulario en la pantalla
ME.Center
'ponemos el titulo de Aplicacion Consultas Medicas en el titulo del formulario
ME.caption = "Aplicacion Consultas Medicas"

END

PUBLIC SUB ButtonConsultaEnfermos_Click()
'al pulsar el boton ConsultaEnfermos, muestra el formulario FormEnfermos
FormEnfermos.Showmodal

END

PUBLIC SUB ButtonConsultaExpediente_Click()
'al pulsar el boton ConsultaEnfermos, muestra el formulario FormElegirEnfermo
FormElegirEnfermo.Showmodal

END
```

8.2. El código de FormEnfermos

En este formulario vamos a trabajar con la Tabla de Enfermos, para insertar, editar, borrar, hacer búsquedas y hacer consultas de filtros con SQL, y vamos a usar las clases que ha generado el MapBD automáticamente.

```
' Gambas class file
' Creamos una instancia de la clase TablaEnfermo, con esta instancia trabajaremos en todo el
' formulario al estar declarada como "Private"
Private RegistroEnfermo AS NEW TablaEnfermo
Private resultado AS Result
Private contador AS Integer

PUBLIC SUB Form_Open()
' abrimos la conexión de RegistroEnfermo con la base de datos
RegistroEnfermo.Abrir()
' El gridview existente en el formulario lo formateamos con los datos la instancia RegistroEnfermo,
' consiguiendo así que por ejemplo los títulos de las columnas van a tener el nombre de los campos
GridViewEnfermos = RegistroEnfermo.gridFormatearColumnas(GridViewEnfermos)

' rellenamos el gridview con datos actualizado.
Actualizartabla

' escritor en la consola un pequeño informe con los datos de la tabla (nombre, tipo de campos)
RegistroEnfermo.informe()
END

PUBLIC SUB actualizartabla()
' rellenamos el gridview con datos actualizado.
' para ello:
' 1º hacemos una consulta de todo su contenido.
resultado = RegistroEnfermo.contenido()
' 2º igualamos el número de filas con el número de registros que tenga la consulta anterior. Esto
' hace que se produzca el evento DATA del gridviews, que se va a encargarse de dibujar los datos
' necesarios.
GridViewEnfermos.Rows.count = resultado.Count
END

PUBLIC SUB GridViewEnfermos_Data(row AS Integer, column AS Integer)
' este evento se encarga de dibujar los datos en el gridviews, para que aparezcan en la pantalla.
resultado.MoveTo(row)
GridViewEnfermos.Data.text = Str(resultado[GridViewEnfermos.Columns[column].text])

' colorea la fila, según sea par o impar
IF row MOD 2 = 0 THEN GridViewEnfermos.Data.Background = Color.LightBackground
END

PUBLIC SUB ButtonInsertar_Click()
' Vamos a insertar un registro. Para ello 1º instanciamos de nuevo la clase (borrando todos los
' datos anteriores que tuviera)
RegistroEnfermo = NEW TablaEnfermo
' Vamos a pedir al usuario los nuevos datos,
' para ello usamos un simple "inputBox", que nos muestra un pequeño formulario donde se nos pide
' con un texto que introduzcamos un nuevo valor.
' El valor introducido por el usuario, se lo asignamos a cada propiedad
RegistroEnfermo.nombre = InputBox("Introduce el nombre del enfermo: ")
RegistroEnfermo.apellido = InputBox("Introduce el apellido del enfermo: ")
RegistroEnfermo.telefono = InputBox("Introduce el nº del telefono del enfermo: ")
RegistroEnfermo.IdEnfermo = InputBox("Introduce el DNI del enfermo: ")

' Hacemos la comprobación del que el valor de "IdEnfermo" no está vacío (ya que si lo está y lo
' insertamos, nos dará problemas al manipular posteriormente la B.D)
IF RegistroEnfermo.IdEnfermo <> "" THEN
' nos conectamos a la base de datos
RegistroEnfermo.Abrir()
```

Consulta Médica: Haciendo un programa desde cero y uso de la herramienta MapBD

```
'insertamos el nuevo enfermo en la base de datos
RegistroEnfermo.Insertar()
'actualizamos el gridviews, para que se muestren los datos..
actualizartabla
ENDIF
END

' Para borrar y editar, vamos a hacer que el usuario primero tenga primero pulsar el botón de la
opción y luego pulsar en la fila del gridviews que quiera realizar la edición o borrado del
registro de datos.
' Esto es una forma de hacerlo, hay muchas más, incluso mejores. ;;; Tu puedes mejorarla !!!
' Para ello, vamos a usar el valor de una "label", que además de mostrar al usuario la opción que
ha elegido, nos va a servir para cuando haga click el usuario en el gridviews, sepamos que opcion
ejecutar (borrar o editar).

PUBLIC SUB ButtonBorrar_Click()
'informamos que vamos a borrar la fila del gridviews que pulse a continuación el usuario.
Label1.text = "Pulse la fila a borrar"
END

PUBLIC SUB ButtonEditar_Click()
'informamos que vamos a editar la fila del gridviews que pulse a continuación el usuario.
Label1.text = "Pulse la fila a editar"
END

PUBLIC SUB ButtonBuscar_Click()
' El usuario ha hecho click en el boton buscar
'1° con inputbox(), pedimos el "Idenfermo" o parte de él, que queremos buscar
'2° con el método "BuscarContenidoIdEnfermo()", busca todos los IdEnfermos que contengan los
caracteres que hemos introducido en el inputbox
    resultado = RegistroEnfermo.BuscarContenidoIdEnfermo(InputBox("Puede introducir Id del enfermo
(aunque no es completo):"))
'actualizamos las filas de gridviews, generando un evento DATA, que se va a encargar de dibujar los
resultados
    GridViewEnfermos.Rows.count = resultado.count
END

PUBLIC SUB GridViewEnfermos_Click()
'El usuario ha hecho click en alguna fila del gridviews, según el valor que contenga el label1
vamos a actuar.

    IF Label1.text = "Pulse la fila a borrar" THEN
' la columna nº 3 del gridview contiene los IdEnfermos. Con GridViewEnfermos.row sabemos cual es
la fila que se ha hecho el click
' Con el método BorrarIdEnfermo, borramos el IdEnfermo que ha seleccionado el usuario.
        RegistroEnfermo.BorrarIdEnfermo(GridViewEnfermos[GridViewEnfermos.row, 3].text)
'redibuja el gridview.
        Actualizartabla
'limpia el valor del label
        Label1.text = ""
    ENDIF

    IF Label1.text = "Pulse la fila a editar" THEN
'preguntamos los datos, poniendo por defecto los antiguos.., y se lo asignamos a la clase.
        RegistroEnfermo.nombre = InputBox("Introduce el nombre del enfermo: ",,
GridViewEnfermos[GridViewEnfermos.row, 0].text)

        RegistroEnfermo.apellido = InputBox("Introduce el apellido del enfermo: ",,
GridViewEnfermos[GridViewEnfermos.row, 1].text)

        RegistroEnfermo.telefono = Val(InputBox("Introduce el nº del telefono del enfermo: ",,
Val(GridViewEnfermos[GridViewEnfermos.row, 2].text)))

'el IdEnfermo del registro, es donde hemos hecho el click
        RegistroEnfermo.IdEnfermo = GridViewEnfermos[GridViewEnfermos.row, 3].text
```

Consulta Médica: Haciendo un programa desde cero y uso de la herramienta MapBD

```
'escribimos el registro en la base de datos, sabiendo cual es su Id
RegistroEnfermo.ModificarIdEnfermo(GridViewEnfermos[GridViewEnfermos.row, 3].text)
'redibuja el gridview.
Actualizartabla
'limpia el valor del label
Label1.text = ""
ENDIF

END

PUBLIC SUB ToolButtonRefrescarDatos_Click()
' Con el boton Refrescar Datos, formamos a redibujar todo el gridview
GridViewEnfermos.Rows.count = 0
resultado = RegistroEnfermo.contenido()
GridViewEnfermos.Rows.count = resultado.Count
END

PUBLIC SUB form_Close()
'cerramos la conexión a la base de datos
RegistroEnfermo.Cerrar

END

'Con este botón, solicitamos una sentencia SQL, y redibujara el gridviews, según los resultados
obtenidos.
PUBLIC SUB SQL_Click()
DIM SQLString AS String
'sentencia que ponemos inicialmente
SQLString = "Select * From TablaEnfermo WHERE nombre like '%u%' "
'preguntamos por la nueva sentencia, dejando como defecto la anterior.
SQLString = InputBox("Introduce la sentecia SQL: ", , SQLString)
TextLabelSentenciaSQL.text = SQLString

'Con esta sentencia:
' Redibujamos el gridViewEnfermosFiltradoSQL, usando la función .GridResultanteSQL, que lee el
resultado devuelto por el método RegistroEnfermo.SQL, que recibe la sentencia SQLString que ha
escrito el usuario.

GridViewEnfermosFiltradosSQL = RegistroEnfermo.GridResultanteSQL( RegistroEnfermo.sql( SQLString ),
GridViewEnfermosFiltradosSQL)

END
```

8.3. El código del Formulario Elige Enfermo.

Este formulario nos muestra todos los registros de enfermos existente en la B.D, cuando le hacemos click en alguno de ellos, pasa al formulario Expediente, para mostrarnos el expediente médico del enfermo elegido.

```
' Gambas class file
Private RegistroEnfermo AS NEW TablaEnfermo
Private resultado AS Result

PUBLIC SUB Form_Open()
'centramos el formulario
ME.center
'abrimos la conexión de RegistroEnfermo con la B.D
RegistroEnfermo.Abrir()
'formatea el gridviews, para que pueda presentar los datos.
GridViewListadoEnfermos = RegistroEnfermo.gridFormatearColumnas(GridViewListadoEnfermos)
'dibuja los datos en el gridviews
actualizartabla
END

PUBLIC SUB actualizartabla()
resultado = RegistroEnfermo.contenido()
GridViewListadoEnfermos.Rows.count = resultado.Count
END

PUBLIC SUB GridViewListadoEnfermos_Data(row AS Integer, column AS Integer)
resultado.MoveTo(row)
GridViewListadoEnfermos.Data.text = Str(resultado[GridViewListadoEnfermos.Columns[column].text])
IF row MOD 2 = 0 THEN GridViewListadoEnfermos.Data.Background = Color.LightBackground
END

PUBLIC SUB ButtonCancelar_Click()
'cerramos el formulario
ME.close
END

PUBLIC SUB GridViewListadoEnfermos_Click()
'Cuando el usuario hace click en alguna fila del gridviews, pasamos los siguientes valores al
formulario Expediente: el idenfermo, y el nombre (que esta en la columna 0) y apellidos (que esta
en la columna 1) y lo convertimos en mayusculas (upper$)
FormExpediente.idenfermo = GridViewListadoEnfermos[GridViewListadoEnfermos.row, 3].text
FormExpediente.nombreenfermo = Upper$(GridViewListadoEnfermos[GridViewListadoEnfermos.row, 0].text &
" " & GridViewListadoEnfermos[GridViewListadoEnfermos.row, 1].text)
'mostramos el formulario Expediente

FormExpediente.ShowModal
END

PUBLIC SUB form_Close()
'cuando se cierra el formulario también cerramos la conexión de RegistroEnfermo con la B.D
RegistroEnfermo.Cerrar

END
```

8.4. El código del Formulario Expediente

Este formulario se va a encargar de mostrarnos los expedientes relacionados con un enfermo.

Ademas podremos editarlos, insertar y borrarlos.

```
' Gambas class file
'Las siguientes propiedades son publicas ("Public"), haciendo que sean accesibles desde todos los
procedimientos de este mismo formulario, y también desde otros formulario.
'Aprovechamos esta propiedad de ser públicos, para que nos pase el formulario Elige Enfermo, el
enfermo elegido y solo mostraremos los expedientes asociado a ese enfermo.

PUBLIC idenfermo AS String 'id que identifica al enfermo
PUBLIC nombreEnfermo AS String 'id que identifica al enfermo

'ahora definimos la variables que solo serán accesible a los procedimientos de este formulario
Private RegistroExpediente AS NEW TablaExpediente
Private resultado AS Result
Private contador AS Integer
'cuenta el registro que voy mostrando en el gridExpedientefilas
Private SQLcondicion AS String 'cadena de texto que contendrá una sentencia SQL, que usaremos para
mostrar los expedientes que cumplan la condición de que el Idenfermo igual al elegido en el
anterior formulario
Private Numeroderegistros AS Integer 'es el numero de registros que cumplen la condicion

PUBLIC SUB Form_Open()
ME.Center
'pone el titulo de la ventana, el nombre del enfermo
ME.caption = "Expediente de " & nombreEnfermo

'abre la conexión a la base de datos.
RegistroExpediente.Abrir()
'formatea el grid que muestra los registros en columnas
GridViewExpedientesColumnas = RegistroExpediente.gridFormatearColumnas(GridViewExpedientesColumnas)
'dibuja el grid que muestra los registros en columnas
actualizartabla

'formatea el grid que muestra los registros en filas
GridViewExpedientesFilas = RegistroExpediente.gridFormatearFilas(GridViewExpedientesFilas)

' sqlcondicion: esta es la condicion que deben de cumplir los registros de expediente (que su
idenfermo sea igual al que nos han pasado del formulario Elige Enfermos)
sqlcondicion = "Select * From TablaExpediente where Idenfermo like '" & idenfermo & "'"
Numeroderegistros = RegistroExpediente.sql(sqlcondicion).Count
'nos posicionamos en el 1º registro.
ToolButtonPrimero_Click()
END

PUBLIC SUB actualizartabla()
'tenemos que buscar todos los expediente, cuyo Idenfermo sea igual al que nos han pasado
'porque estamos filtrando por un enfermo en este formulario
resultado = RegistroExpediente.BuscarIgualIdEnfermo(idenfermo)

'actualiza el numero de registros/filas a mostrar en el grid tipo Expediente Columnas
GridViewExpedientesColumnas.Rows.count = resultado.Count

'actualiza el numero de registros para el grid que muestra los expedientes en filas.
Numeroderegistros = resultado.count

END

PUBLIC SUB GridViewExpedientesColumnas_Data(row AS Integer, column AS Integer)
'redibujamos los datos
resultado.MoveTo(row)
GridViewExpedientesColumnas.Data.text = Str(resultado[GridViewExpedientesColumnas.Columns[column].text])
'coloreamos las filas según sea pares o impares.
IF row MOD 2 = 0 THEN GridViewExpedientesColumnas.Data.Background = Color.LightBackground
END

'al igual que hicimos en el formulario de Enfermos, vamos a basarnos en el mismo método para saber
```


Consulta Médica: Haciendo un programa desde cero y uso de la herramienta MapBD

```
que opciones ha pulsado el usuario

PUBLIC SUB ButtonEditar_Click()
    Label11.text = "Pulse la fila a editar"
END

PUBLIC SUB ButtonBorrar_Click()
    Label11.text = "Pulse la fila a borrar"
END

PUBLIC SUB ButtonInsertar_Click()
'insertarnos un registro (del mismo modo que lo hicimos en el formulario de Enfermos)
RegistroExpediente = NEW TablaExpediente
    RegistroExpediente.enfermedades = InputBox("Introduce el nombre de la enfermedad: ")
    RegistroExpediente.tratamiento = InputBox("Introduce el tratamiento: ")

'los datos de ".enfermedades" y ".tratamiento" son del enfermo que estamos editando, por lo tanto
sabemos cual es su id (el que nos paso el formulario Elige Enfermo)
    RegistroExpediente.IdEnfermo = idenfermo

'tenemos que dar un valor al Idexpediente, le pongo por ejemplo un valor que obtengo por la hora
del sistema
    RegistroExpediente.IdExpediente = "Id" & Str$(Now)

'nos conectamos a la base de datos
    RegistroExpediente.Abrir()
'insertamos el nuevo enfermo en la base de datos
    RegistroExpediente.Insertar()
'actualizamos el gridviews, para que se muestren los datos..
    actualizartabla

END

' El usuario hace click en alguna fila del gridViewExpedientesColumnas
PUBLIC SUB GridViewExpedientesColumnas_Click()
    IF Label11.text = "Pulse la fila a borrar" THEN
        'la 1º columna (que el el indice 0), es la que contiene el codigo Idexpediente
'aplicamos la sentencia que borra el registro
        RegistroExpediente.BorrarIdExpediente(GridViewExpedientesColumnas[GridViewExpedientesColumnas.row, 0].text)
'redibujamos la tabla
        actualizartabla
        Label11.text = ""
    ENDIF

    IF Label11.text = "Pulse la fila a editar" THEN
        'preguntamos los datos, poniendo por defecto los antiguos..
        RegistroExpediente.enfermedades = InputBox("Introduce la enfermedad: ",,
        GridViewExpedientesColumnas[GridViewExpedientesColumnas.row, 2].text)

        RegistroExpediente.tratamiento = InputBox("Introduce la enfermedad: ",,
        GridViewExpedientesColumnas[GridViewExpedientesColumnas.row, 3].text)

        RegistroExpediente.IdExpediente = GridViewExpedientesColumnas[GridViewExpedientesColumnas.row, 0].text
        RegistroExpediente.IdEnfermo = GridViewExpedientesColumnas[GridViewExpedientesColumnas.row, 1].text

        'escribimos el registro en la base de datos
        RegistroExpediente.ModificarIdExpediente(GridViewExpedientesColumnas[GridViewExpedientesColumnas.row, 0].text)
        actualizartabla
        Label11.text = ""
    ENDIF

END

PUBLIC SUB ToolButtonRefrescarDatos_Click()
'refrescamos todos los datos del gridviews:
'1º Modificamos el nº de filas a 0
    GridViewExpedientesColumnas.Rows.count = 0
'2º actualizamos el gridviews
    actualizartabla

END
```

Consulta Médica: Haciendo un programa desde cero y uso de la herramienta MapBD

```
'Esta es una forma de crear un informe con los datos. Vamos a crear un documento tipo HTML, (muy simple), pero nos sirve para mostrar como se puede hacer.

PUBLIC SUB ButtonCrearInforme_Click()
DIM texto AS String 'variable de texto, que va a contener todas las instrucciones en html
DIM a AS Integer
'Refresco la tabla
    ToolButtonRefrescarDatos_Click()

'Empezamos a "construir" el archivo HTML
'Escribo la parte inicial del archivo
texto = "<Html><head></head><body>"
'Empiezo a escribir el cuerpo del archivo html
'Escribo el nombre del enfermo, aprovecho que lo tengo en el titulo de la ventana
texto &= ME.Caption & "<br>"
texto &= "<hr>"

'empiezo a rellenar los datos, aprovechando que los tengo todos en el gridviews, los cojo de alli.
En la columna 2 tenemos la enfermedad, y en la columna 3 tenemos el tratameinto.
'Repetimos tantas veces este bucle (numero de filas es igual a .count-1, porque recuerda que empezamos a contar desde 0
FOR a = 0 TO GridViewExpedientesColumnas.Rows.Count - 1
texto &= ".....Enfermedad: " & GridViewExpedientesColumnas[a, 2].text & "<br>"
texto &= ".....Tratameinto: " & GridViewExpedientesColumnas[a, 3].text & "<br>"
'esta linea hace que si es el 1º registro o el ultimo no escriba "-----"
IF (GridViewExpedientesColumnas.Rows.Count - 1) > 0 AND a < (GridViewExpedientesColumnas.Rows.Count - 1) THEN texto &= "-----<br>"
NEXT
'dibuja una linea horizontal
texto &= "<hr>"
' cerramos el cuerpor del archivo html
texto &= "</body></html>"

'salvamos todo el codigo creado en el archivo temporal.html
File.Save(User.Home & "/temporal.html", texto)

'abrimos el archivo recién creado en el navegador Firefox,para poderlo verlo, ademas en firefox podremos imprimirlo (opcion imprimir pagina web)
SHELL "firefox " & User.Home & "/temporal.html"

END

PUBLIC SUB form_Close()
'cerramos la conexión a la base de datos.
    RegistroExpediente.Cerrar

END

'Estos son los botones que van a mostrar el registro en el gridviewsExpedienteFila

PUBLIC SUB ToolButtonPrimero_Click()
'muestra el 1º registro
contador = 0
RegistroExpediente.mostrarRegistro(contador, GridViewExpedientesFilas, sqlcondicion)
GridViewExpedientesFilas.Refresh
'el valor que muestra ValueBoxcontador, siempre es 1 mas, ya que los registros se enumeran desde el 0.
ValueBoxcontador.value = contador + 1
END

PUBLIC SUB ToolButtonAtras_Click()
'mostramos un registro menos
contador -= 1
'comprueba que el contador esta por encima de -1
IF contador > -1 THEN
RegistroExpediente.mostrarRegistro(contador, GridViewExpedientesFilas, sqlcondicion)
GridViewExpedientesFilas.Refresh
ValueBoxcontador.value = contador + 1
ELSE
'restituye el valor del contador, para deshacer la primera linea.
contador += 1
ENDIF
ENDIF
```

Consulta Médica: Haciendo un programa desde cero y uso de la herramienta MapBD

```
END

PUBLIC SUB ToolButtonAdelante_Click()
    contador += 1
    IF contador < (Numeroderegistros) THEN
        RegistroExpediente.mostrarRegistro(contador, GridViewExpedientesFilas, sqlcondicion)
        GridViewExpedientesFilas.Refresh
        ValueBoxcontador.value = contador + 1
    ELSE
        'restituye el valor del contador, para deshacer la primera linea.
        contador -= 1
    ENDIF
END

PUBLIC SUB ToolButtonUltimo_Click()
    contador = Numeroderegistros - 1
    RegistroExpediente.mostrarRegistro(contador, GridViewExpedientesFilas, sqlcondicion)
    GridViewExpedientesFilas.Refresh
    ValueBoxcontador.value = contador + 1
    TRY ValueBoxcontador.Value.Refresh
    END

'para cuando se escribe el numero del registro

PUBLIC SUB ValueBoxcontador_KeyPress()
    DIM numero AS Integer
    'comprueba que el usuario a pulsado la tecla "Enter" o "Return", y si es asi va a intentar mostrar
    el registro
    IF Key.Code = Key.enter OR Key.code = Key.return THEN
        numero = Val(ValueBoxcontador.text)
        'comprueba que el numero que ha escrito el usuario esta entre el intervalo válido de los registros
        existentes.
        IF numero > 0 AND numero <= Numeroderegistros THEN
            'el numero que ha introducido el usuario esta entre el intervalo válido de los registros
            existentes.

            'el contador siempre vale un numero menos que el que muestra el valueboxcontador (osea el numero
            que ha escrito el usuario, por el tema que los registros empiezan siempre por el 0)
            contador = numero - 1
            'Muestra en el gridviewsExpedintesFilas , el registro cuyo numero sea igual al "contador", dichos
            registros cumplen la condicion que indique "sqlcondicion"

            RegistroExpediente.mostrarRegistro(contador, GridViewExpedientesFilas, sqlcondicion)

            'refresca los datos del gridviewsExpedienteFilas
            GridViewExpedientesFilas.Refresh

        ELSE
            'muestra en pantalla un mensaje informando que se ha superado en intervalo de los registros
            existente.
            Message.Error("Fuera de rango de registros")

        ENDIF
    ENDIF
END
```

9. Mejoras que podrías introducir en el programa.

Seguramente se te ocurrieran incluir muchas mejoras al programa, yo te reto a que lo hagas...

Por ejemplo, las que se me ocurren a mi:

Inputbox: La primera podría ser que en vez de usar un inputbox, utilizaras un formulario donde se podría ver mucho mejor los datos que tienes que introducir o estas editando.

No existe “comprobaciones” de los datos que introduce el usuario: por ejemplo si introduce un texto en un sitio que tiene que introducir números, puede dar problemas, habría que hacer alguna comprobación e indicarle al usuario si los datos que ha introducido son correctos o no.

Mejorar el informe de salida de datos: el html que he realizado es demasiado básico, podrías formatearlo mucho mejor, haciendo lo mas bonito al usuario. Además gambas también incluye el componente gb.report, que puede llegar a generar informes con apariencia muy profesional.

En fin, seguro que encuentra muchos más detalles para mejorar....

Un cordial saludo

Julio Sánchez Berro

Dos hermanas, Enero de 2012

10 Anexos: Métodos que crea automáticamente MapBd 0.0.3

.Abrir()

Se encarga de abrir la conexión a 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 conexión a 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"
```

Consulta Médica: Haciendo un programa desde cero y uso de la herramienta MapBD

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.BuscarContenidonombre("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.BuscarIgualnombre("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.BuscarMenorQueidDNI(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.

Consulta Médica: Haciendo un programa desde cero y uso de la herramienta MapBD

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)

Escribe en el gridviews dato, un registro. Ese registro es que se indica en "numero". La lista de registros es la que se obtiene 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:



Consulta Médica: Haciendo un programa desde cero y uso de la herramienta MapBD

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(g

rid 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

Consulta Médica: Haciendo un programa desde cero y uso de la herramienta MapBD

```
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

11. Enlaces interesantes y Donde conseguir los Programas:

El curso que hice de analista funcional: <https://sites.google.com/site/cursofpeanalistafuncional/>

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

[Tema dedicado al MapBD en el foro de gambas-es.org](#)

<http://live.gnome.org/Dia>

<http://sqlitebrowser.sourceforge.net/>