Acceso a base de datos SQLite desde Gambas

Si deseamos realizar una pequeña aplicación en Gambas que trabaje con una base de datos local (alojada en el mismo disco rígido que nuestra aplicación), la opción mas adecuada será el sistema de base de datos relacional SQLite. Las bases de datos SQLite poseen un diseño simple dado que el conjunto de la base de datos (definiciones, tablas, índices, y los propios datos) son guardados como un sólo fichero estándar en la máquina host. Se podría decir que una base de datos SQLite es como una base de datos de Access (en el sentido que se almacena como un archivo, y no hay un servidor de base de datos detrás de el).

Gambas puede manejar diferentes tipos de base de datos, ellos son los populares MySQL, Postgres y el ya mencionado SQLite. Para acceder a ellos Gambas dispone de un componente llamado gb.db el cual contiene los drivers específicos para manejar cada una de estas bases de datos. Lo bueno del componente gb.db es que accede a cada una de estas bases de datos de la misma manera, con el mismo código. Este componente puede manejar las bases de datos SQLite en sus versiones 1, 2 y 3.

Diseño de la base de datos

Existen varias alternativas para crear una base de datos SQLite. La que veremos en este articulo es desde el mismísimo entorno de Gambas. También es posible hacerlo desde la consola, o, mas fácil aún, desde el navegador web Firefox instalando el agregado SQLite Manager.

Para crear nuestra primer base de datos (una simple agenda con datos de contactos) vamos a utilizar el Gestor de base de datos que ofrece Gambas, para hacerlo debemos iniciar un nuevo proyecto en Gambas del tipo aplicación gráfica. En el menú herramientas encontramos el Gestor de base de datos. Verán que la iniciarlo nos pide una contraseña, esta contraseña es necesaria para almacenar encriptados los datos de usuarios y contraseñas. Debemos ingresar allí una contraseña de 8 caracteres como mínimo y accederemos al gestor que inicialmente se encuentra vacío por no tener creadas conexiones a base de datos. Para crear una lo hacemos desde su único menú llamado Servidor y allí elegimos Nuevo Servidor que nos lleva a la siguiente ventana:

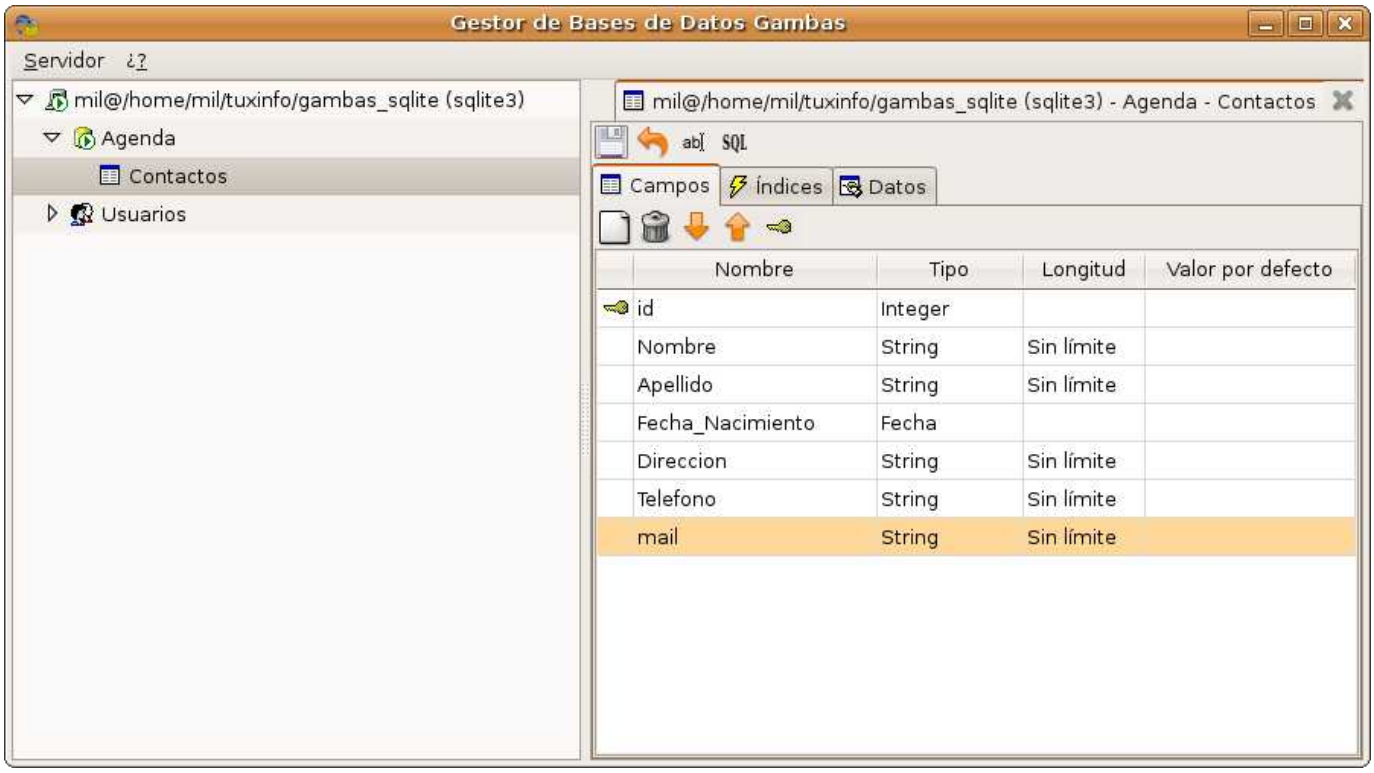
El primer dato, Tipo, hace referencia al driver que utilizaremos para acceder a la base de datos, es decir, que tipo de base de datos que deseamos utilizar. Las opciones posibles son las mencionadas anteriormente: postgresql, mysql, sqlite, sqlite2, sqlite3 y odbc. Seleccionamos aquí sqlite3.

Al optar por sqlite3 solo resta indicar la ruta a la base de datos, en mi caso voy a guardar la base de datos en el mismo directorio del proyecto de Gambas que acabamos de iniciar. Por lo tanto la ruta absoluta para mí será /home/mil/tuxinfo/gambas_sqlite. Pongan aquí la que les correspondan a ustedes.

Al aceptar veremos en el gestor de base de datos la existencia de una conexión, la abrimos con el botón derecho del mouse y luego otra vez con el botón derecho optamos por Crear base. Nos solicita un nombre para la base de datos, la llamaremos Agenda. Ahora sobre la base de datos Agenda desplegamos el menú contextual, siempre con el botón derecho del mouse, y seleccionamos Crear para generar una tabla que llamaremos Contactos.

El gestor de base de datos nos muestra con una estructura de árbol cada uno de los componentes de nuestra base de datos. Ahora ha llegado el momento de diseñar nuestra tabla, es decir, crear los campos para almacenar los datos, lo hacemos a través del menú contextual de la tabla Contactos seleccionando Crear. Atención en este punto. Puede ser que al hacer clic en Crear no note ningún cambio en el gestor de base de datos, si así ocurre no desespere, lo que ha ocurrido es lo siguiente: el gestor de base de datos dispone de dos paneles, uno con la estructura de árbol de la base de datos, que se encuentra ocupando todo el espacio del gestor, y otro panel para el acceso a los datos, que no se visualiza por culpa del anterior, el cual ocupa toda la ventana. La solución es llevar el cursor del mouse hacia el extremo derecho de la ventana y cuando el puntero adopte la forma de flecha bidireccional arrastrarla hacia la izquierda con el fin de liberar espacio de trabajo. Vera como emerge el segundo panel, tal cual se observa a continuación.

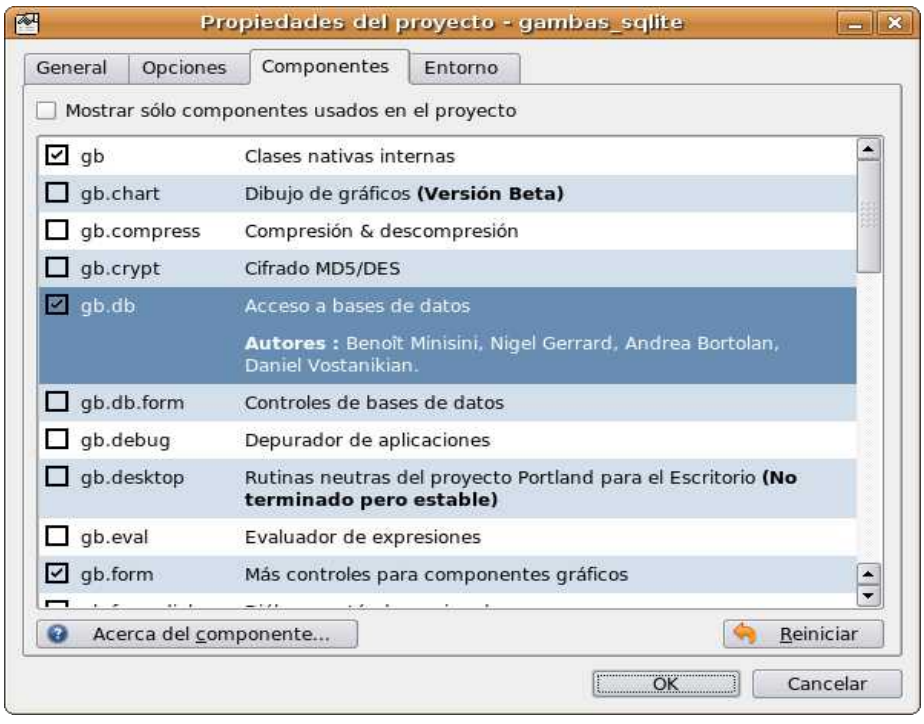
Ahora es momento de diseñar la estructura de nuestra agenda. Para crear los campos lo hacemos con el icono de hoja en blanco. La estructura propuesta es la siguiente:



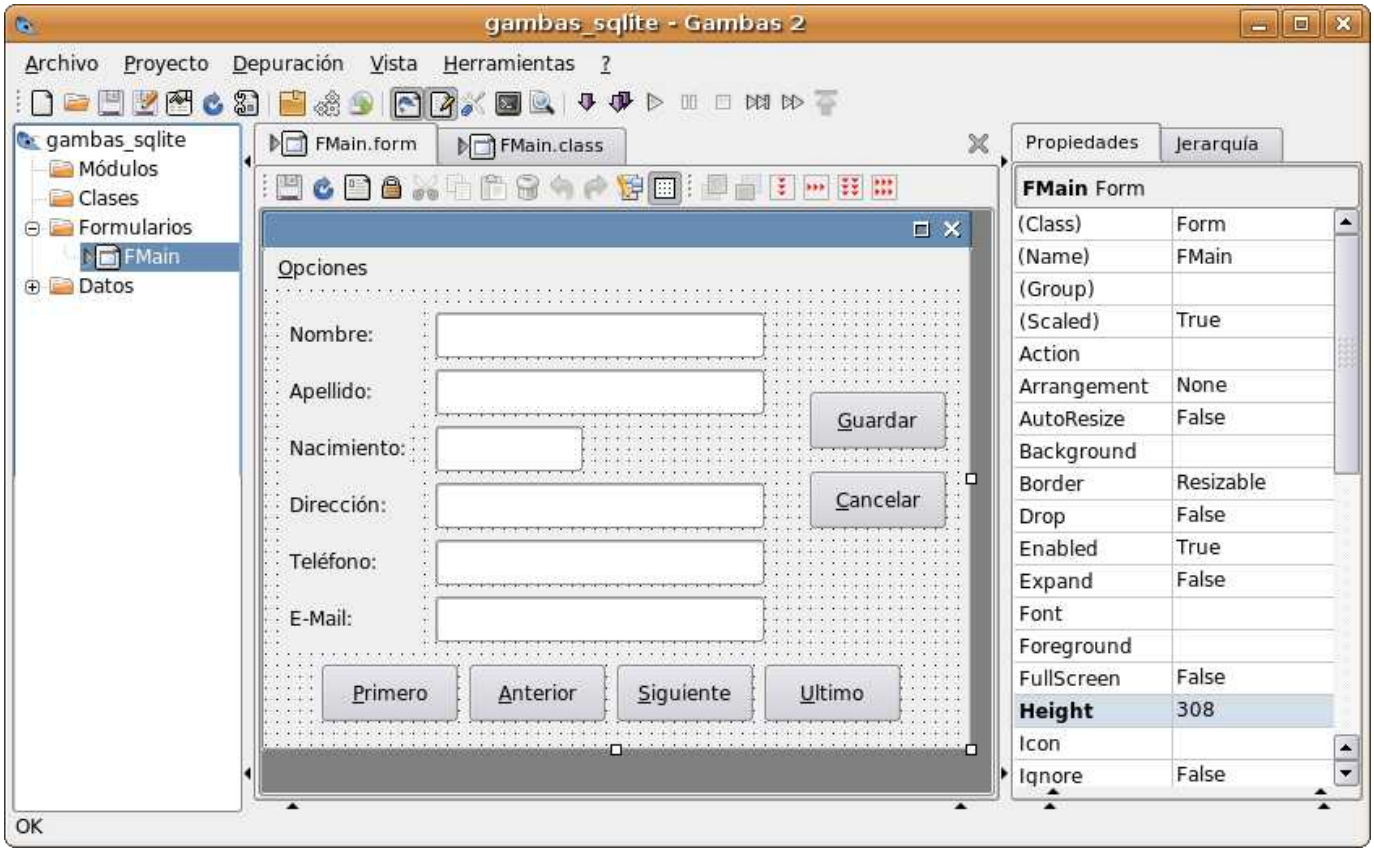
Tenga presente cuando finalice el diseño guardar la estructura de la base de datos mediante el icono de diskette. Ahora puede desde la solapa Datos ingresar los datos de alguno de sus contactos. Cuando finalice vuelva a guardar y cierre el gestor de base de datos para así dar inicio a la programación de la aplicación que hará uso de la base de datos recién creada.

Programando el acceso a base de datos SQLite

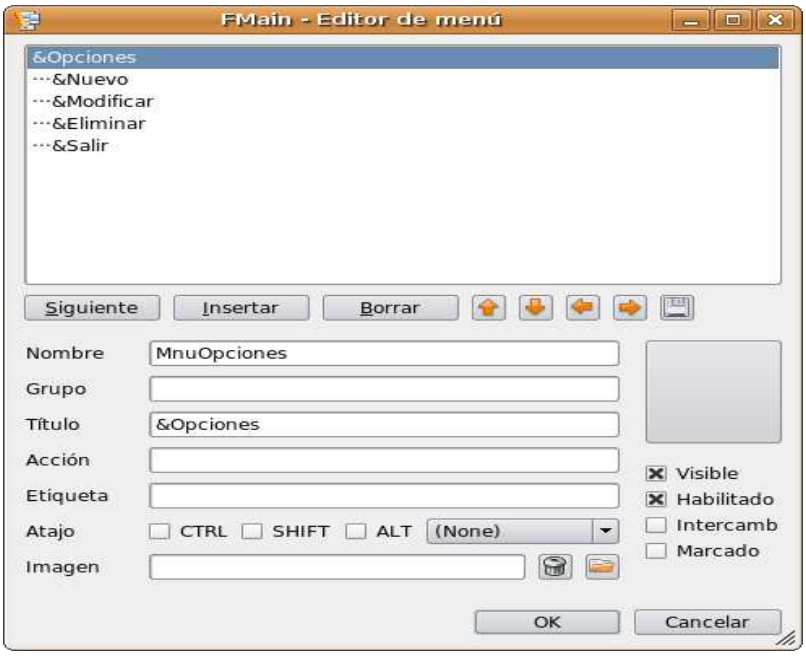
Lo primero que debemos hacer es "avisar" a Gambas que nuestra aplicación tendrá acceso a una base de datos. Esto lo hacemos desde el menú Proyecto → Propiedades y luego desde la solapa Componentes activando el componente gb.db como vemos a continuación:



Ahora es el momento de diseñar la aplicación que permita llevar a cabo el clásico ABM (altas, bajas y modificación) sobre nuestra tabla Contactos. El diseño propuesto es el siguiente:



Como se observa, al pie del formulario contamos con 4 botones que permitirán desplazarnos de un registro a otro. Los botones Guardar y Cancelar deben tener la propiedad visible a Falso, ya que solo se harán visibles en el momento en que se solicite incorporar un nuevo registro a la base de datos. Para agregar, modificar y eliminar vamos a crear el menú Opciones que se observa en la parte superior del formulario. Sobre este menú se desprenden el menú Nuevo, Modificar, Eliminar y Salir. Para crear el menú lo hacemos mediante el atajo de teclado Ctrl+E o buscando la opción correspondiente al Editor de Menú en la barra de herramientas. El Menú queda de la siguiente manera:



Ha llegado el momento entonces de programar. Primero declaramos la variables para acceder a la base de datos y luego las inicializamos en el evento Open del formulario por ser el primero que se ejecuta al iniciar la aplicación.

```
PRIVATE Conexion AS Connection
PRIVATE TablaContactos AS Result

PUBLIC SUB Form_Open()
  Conexion = NEW Connection
  Conexion.Type = "sqlite3"
  Conexion.Host = "/home/mil/tuxinfo/gambas_sqlite"
  Conexion.Name = "Agenda"
  TRY Conexion.Open()
  IF ERROR THEN
    Message.Error("Error al conectar a la base de datos.")
    Conexion = NULL
  ELSE
    TablaContactos = Conexion.Exec("Select * from Contactos")
    IF TablaContactos.Available THEN MostrarCampos
  END IF
END
```

En el código anterior se declara una variable llamada Conexion del tipo Connection que tendrá acceso a toda la base de datos, luego se declaró otra variable llamada TablaContactos que contendrá en memoria todos los registros cargados en la tabla Contactos.

Al iniciar el programa se dispara el evento Open y es allí donde se inicializan las variables previamente declaradas. Son tres las propiedades necesarias para configurar la conexión: Type, Host y Name. La primera indica el tipo de base de datos al que accederemos, la segunda establece la ruta absoluta en donde se ubica y la tercera el nombre de la base de datos. Luego Intentamos abrir la conexión con el método Open, si arroja un error lo informamos, en caso contrario la conexión a la base de datos fue exitosa y procedemos a inicializar la variable TablaContactos con una consulta SQL que nos devuelva la totalidad de registros de la tabla Contactos. La siguiente línea pregunta si hay algún registro disponible, de ser así llamamos al procedimiento MostrarCampos que se encarga de mostrar cada campo en su correspondiente TextBox. El código de este procedimiento a continuación:

```
PUBLIC SUB MostrarCampos()
  txtNombre.text = TablaContactos["Nombre"]
  txtapellido.text = TablaContactos["Apellido"]
  TxtNacimiento.text = TablaContactos["Fecha_Nacimiento"]
  TxtDireccion.text = TablaContactos["Direccion"]
  TxtTelefono.text = TablaContactos["Telefono"]
  TxtMail.text = TablaContactos["mail"]
END
```

Desplazarnos a través de los registros

Ahora es momento de programar los 4 botones de movimiento con el fin de permitir al usuario la navegación a través de cada uno de los registros. Disponemos para ello de 4 métodos que se encargan de cambiar el registro activo. Estos métodos son MoveFirst, MovePrevious, MoveNext y MoveLast y se aplican a las objetos del tipo Result. Los vemos en acción a continuación:

```
PUBLIC SUB BtnPrimero_Click()
  TablaContactos.Movefirst()
  MostrarCampos
END

PUBLIC SUB BtnAnterior_Click()
  TablaContactos.MovePrevious()
  IF NOT TablaContactos.Available THEN TablaContactos.Movefirst()
  MostrarCampos
END

PUBLIC SUB BtnSiguiente_Click()
  TablaContactos.MoveNext()
  IF NOT (TablaContactos.Available) THEN TablaContactos.MoveLast()
  MostrarCampos
END

PUBLIC SUB BtnUltimo_Click()
  TablaContactos.MoveLast()
  MostrarCampos
END
```

Muy bien, hasta aquí el programa es capaz de mostrarnos todos los contactos de nuestra base de datos. Es momento ahora de programar el ABM. Existen muchas variantes para hacerlo, la que propongo aquí es la mas sencilla pero no por ello la mas elegante.

Agregar un nuevo registro

Para incorporar un nuevo registro a la base de datos lo haremos desde el menú Nuevo. Debemos Mostrar los botones Guardar y Cancelar y dejar todos los TextBox vacíos para que el usuario cargue los nuevos datos:

```
PUBLIC SUB MnuNuevo_Click()
  BtnGuardar.Visible = TRUE
  BtnCancelar.Visible = TRUE
  txtNombre.text = ""
  txtapellido.text = ""
  TxtNacimiento.text = ""
  TxtDireccion.text = ""
  TxtTelefono.text = ""
  TxtMail.text = ""
END
```

Cuando el usuario termino de cargar los datos de su nuevo contacto podrá incorporarlo a la base de datos por medio del botón Guardar. Si desea cancelar la operación dispone del botón Cancelar. Estos botones se programan de la siguiente manera:

```
 PUBLIC SUB BtnGuardar_Click()
  DIM VarResult AS Result
  VarResult = Conexion.Create("Contactos")
  VarResult["Nombre"] = txtNombre.Text
  VarResult["Apellido"] = txtapellido.Text
  VarResult["Fecha_Nacimiento"] = TxtNacimiento.Text
  VarResult["Direccion"] = TxtDireccion.Text
  VarResult["Telefono"] = TxtTelefono.Text
  VarResult["mail"] = TxtMail.Text
  VarResult.Update
  TablaContactos = Conexion.Exec("Select * from Contactos")
  BtnGuardar.Visible = FALSE
  BtnCancelar.Visible = FALSE
END
```

```
PUBLIC SUB BtnCancelar_Click()
  BtnGuardar.Visible = FALSE
  BtnCancelar.Visible = FALSE
  MostrarCampos
END
```

Para guardar un nuevo registro se utilizó una variable local del tipo Result que efectúa una petición de incorporar un nuevo registro a la conexión a la base de datos. Luego se pasa el valor de cada TextBox a cada campo y lo confirmamos con el método Update. Es también necesario volver a inicializar la variable TablaContactos para que incorpore el registro recién adicionado. Por último ocultamos los botones Guardar y Cancelar. Si el usuario se arrepiente y no desea guardar el nuevo contacto habrá que volver a mostrar el registro anterior y ocultar los botones Guardar y Cancelar.

Modificar un registro

Para modificar un registro el proceso es similar al realizado anteriormente, la diferencia está en la apertura de la variable del tipo Result, que se hará mediante el método Edit que recibe un parámetro que indica el registro a modificar. Allí podemos utilizar el campo ID que es un número único de identificación para cada contacto:

```
PUBLIC SUB MnuModificar_Click()
  DIM VarResult AS Result
  VarResult = Conexion.Edit("Contactos", "id=" & TablaContactos["id"])
  VarResult["Nombre"] = txtNombre.text
  VarResult["Apellido"] = txtapellido.text
  VarResult["Fecha_Nacimiento"] = TxtNacimiento.text
  VarResult["Direccion"] = TxtDireccion.text
  VarResult["Telefono"] = TxtTelefono.text
  VarResult["mail"] = TxtMail.text
  VarResult.Update
  TablaContactos = Conexion.Exec("Select * from Contactos")
END
```

Eliminar un registro

El código para eliminar un registro es muy sencillo, se procede de la misma manera anterior con la variante de ejecutar el método Delete sobre el registro a modificar:

```
PUBLIC SUB Mnu_Eliminar_Click()
DIM VarResult AS Result
IF Message.Question("¿Desea eliminar el registro?", "Si", "No") = 1 THEN
   VarResult = Conexion.edit("Contactos", "id=" & TablaContactos["id"])
   VarResult.Delete
   TablaContactos = Conexion.Exec("Select * from Contactos")
END IF
END
```

De esta manera damos por finalizada la programación de nuestro sistema ABM. Tengan presente que se podría haber logrado el mismo resultado, pero con una programación más estándar ejecutando instrucciones SQL del tipo Insert, Update y Delete. Los invito a que investiguen como hacerlo, ya que será de ayuda para situaciones mas complejas.

Pablo Mileti
pablomileti@gmail.com

Nos quedó pendiente programar el menú Salir. Lo hacemos con salida profesional de la siguiente manera:

```
PUBLIC SUB MnuSalir_Click()
  ME.Close
END

PUBLIC SUB Form_Close()
  IF Message.Question("¿Desea salir del programa?", "Si", "No") = 2 THEN
     STOP EVENT
  ELSE
     Conexion.Close
  END IF
END
```

Pablo Mileti
pablomileti@gmail.com