

Acceso a base de datos SQLite desde Gambas 3.

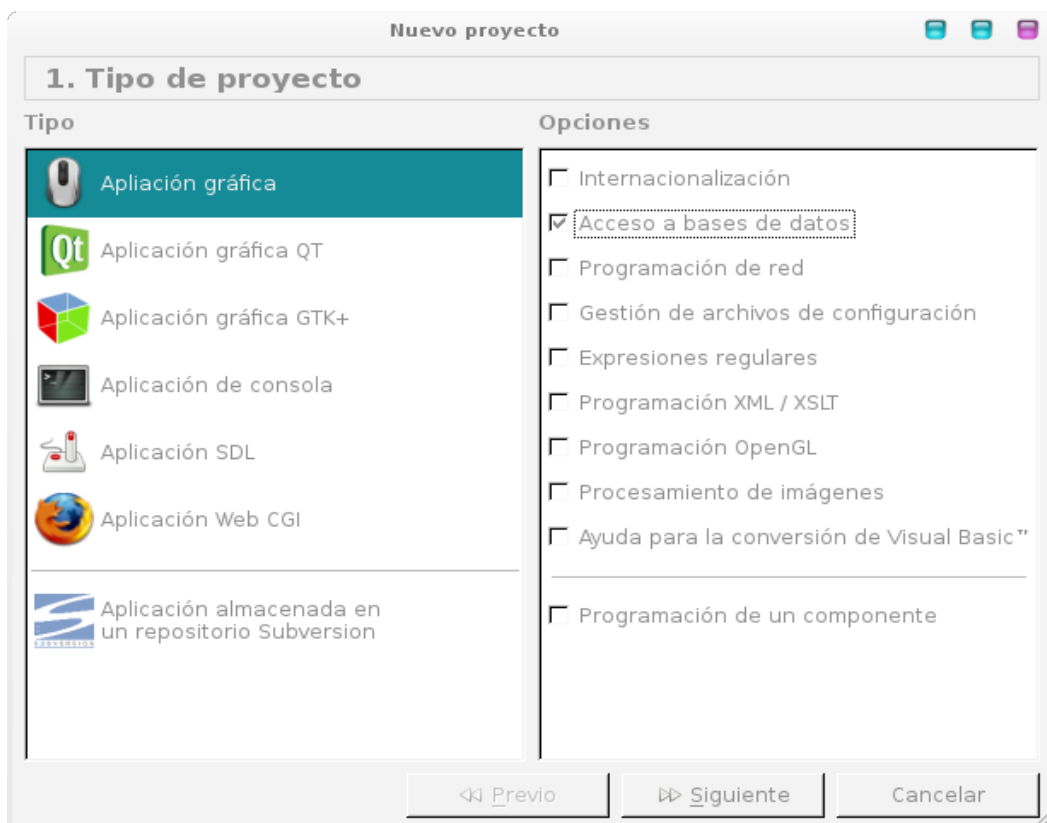
Si deseamos realizar una pequeña aplicación en Gambas que trabaje con una base de datos local (alojada en el mismo disco rígido que nuestra aplicación), la opción mas adecuada será el sistema de base de datos relacional SQLite. Las bases de datos SQLite poseen un diseño simple dado que el conjunto de la base de datos (definiciones, tablas, índices, y los propios datos) son guardados como un sólo fichero 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 él).

Gambas puede manejar diferentes tipos de base de datos, ellos son los populares MySQL, PostgreSQL, Firebird 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.

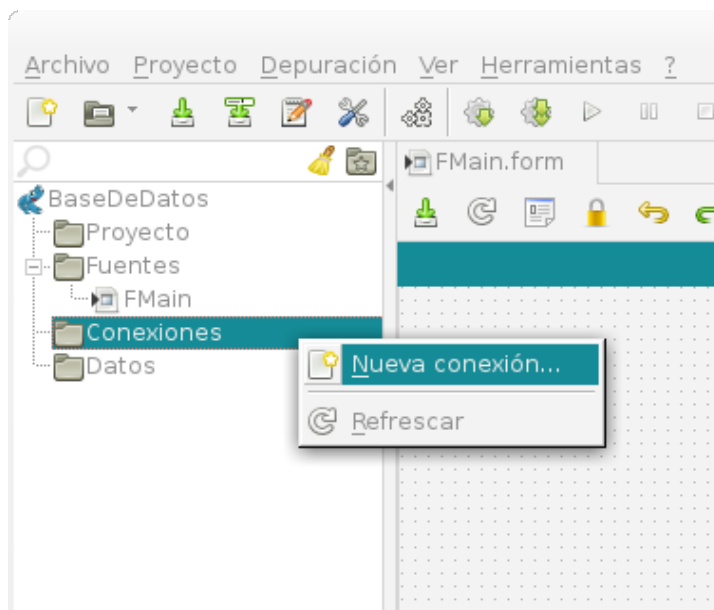
Diseño de la base de datos

Existen varias alternativas para crear una base de datos SQLite. Nosotros lo haremos desde el mismísimo entorno de Gambas 3.

Para crear nuestra primer base de datos (una simple agenda con datos de contactos) vamos a crear un nuevo proyecto del tipo **Aplicación gráfica y tildamos** que tendrá **acceso a base de datos** (esto activará los componente gb.db y gb.db.form):



En la vista de proyecto disponemos de una nueva carpeta llamada Conexiones. Si realizamos clic derecho podremos crear una nueva conexión hacia una base de datos nueva o existente.

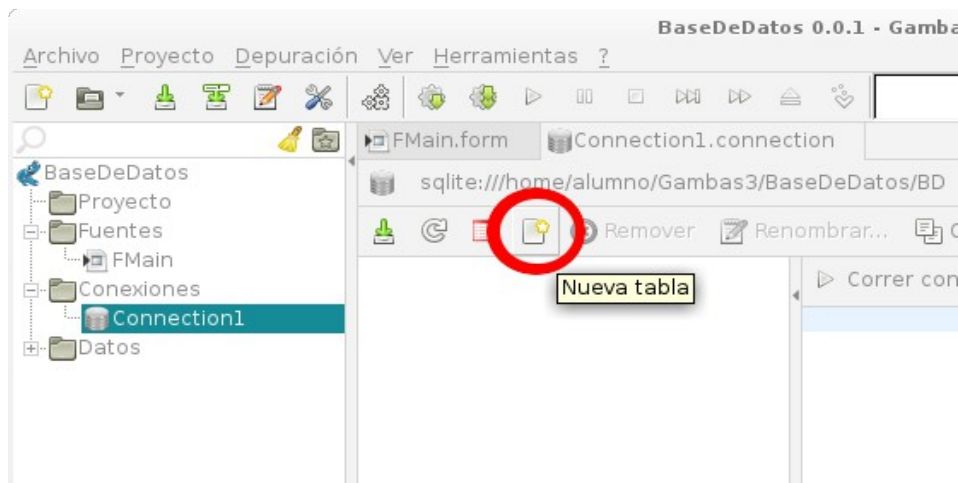


Y llegaremos a la siguiente ventana:

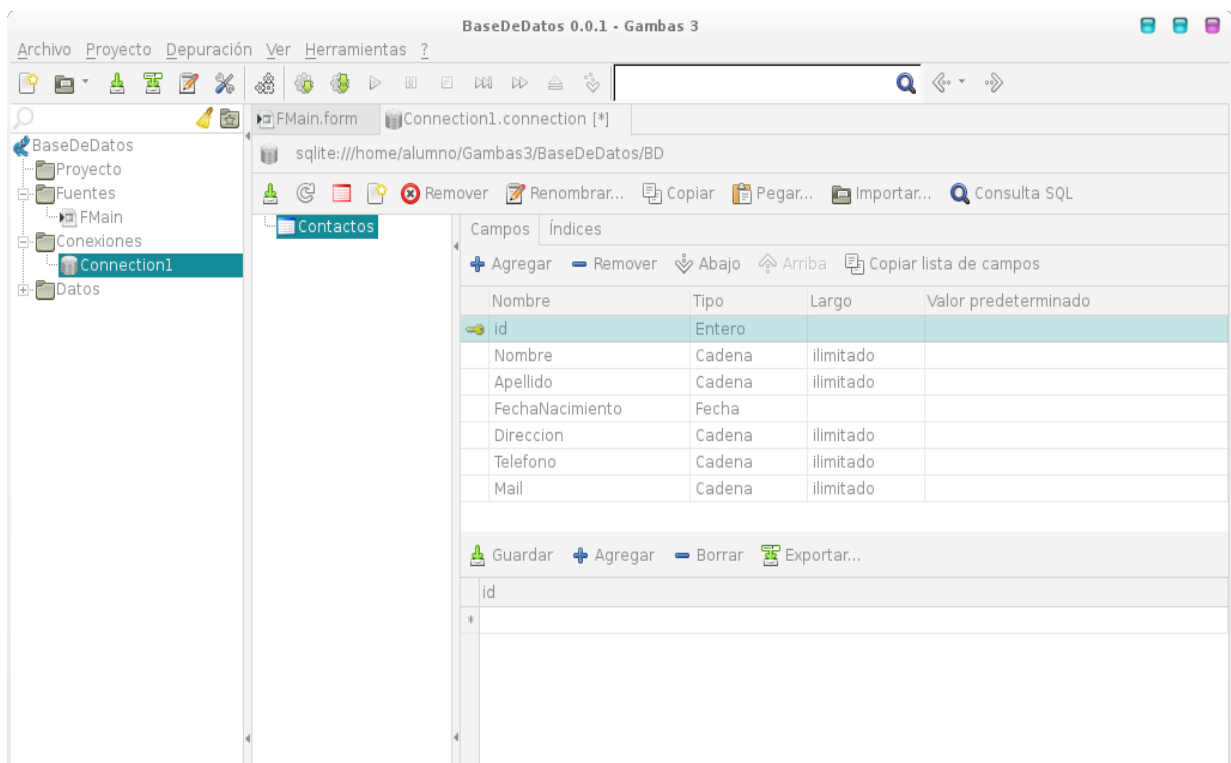


Allí debemos darle un nombre a la conexión. Indicaremos que el tipo de motor de base de datos será SQLite y en ruta seleccionamos en que carpeta del disco deseamos guardar la nueva base de datos. En Base de datos debemos ingresar el nombre que deseamos darle, en este caso la llamaremos DB y presionando el botón de la derecha nos ofrecerá un botón para crear la base de datos. Pulsamos allí y cerramos la ventana mediante el botón OK.

Ahora ha llegado el momento de diseñar nuestra tabla, es decir, crear los campos para almacenar los datos, lo hacemos a través del icono Nuevo, en este caso Nueva Tabla:



Como nombre de la tabla ingresamos Contactos. Y procedemos a crear los campos. La estructura propuesta es la siguiente:




Una vez diseñada la tabla presionamos el icono guardar y podremos incorporar nuevos registros a la base de datos desde la segunda mitad de la ventana. Cuando finalice la carga, cierre la solapa actual para así dar inicio a la programación de la aplicación que hará uso de la base de datos recién creada.

Diseño de la aplicación

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 sólo 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 mediante clic derecho en un área libre del formulario y seleccionando Editor de Menú. El Menú queda de la siguiente manera:



Programando el acceso a base de datos SQLite

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/alumno/Gambas3/BaseDeDatos"
    Conexion.Name = "BD"
    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
```

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 llamada 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()
    TxtID.Text = TablaContactos["id"]
    txtNombre.text = TablaContactos["Nombre"]
    txtapellido.text = TablaContactos["Apellido"]
    TxtFechaNacimiento.text = TablaContactos["FechaNacimiento"]
    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, en nuestro caso llamado TablaContactos. 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 BtnSiguiete_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 (altas, bajas, modificación). Existen muchas variantes para hacerlo, la que propongo aquí es muy 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, ocultar los 4 botones de movimiento y dejar todos los TextBox vacíos para que el usuario cargue los nuevos datos:

```
Public Sub MnuNuevo_Click()
    BtnGuardar.Visible = True
    BtnCancelar.Visible = True
    BtnPrimero.Visible = False
    BtnAnterior.Visible = False
    BtnSiguiete.Visible = False
    BtnUltimo.Visible = False
    TxtID.Text = ""
```

```

txtNombre.text = ""
txtapellido.text = ""
TxtFechaNacimiento.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["id"] = TxtID.Text
  VarResult["Nombre"] = txtNombre.Text
  VarResult["Apellido"] = txtapellido.Text
  VarResult["FechaNacimiento"] = TxtFechaNacimiento.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
  BtnPrimero.Visible = True
  BtnAnterior.Visible = True
  BtnSiguiente.Visible = True
  BtnUltimo.Visible = True

```

End

```

Public Sub BtnCancelar_Click()
  BtnGuardar.Visible = False
  BtnCancelar.Visible = False
  BtnPrimero.Visible = True
  BtnAnterior.Visible = True
  BtnSiguiente.Visible = True
  BtnUltimo.Visible = True
  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 y volvemos a mostrar los 4 botones de movimiento. Si el usuario se arrepiente y no desea guardar el nuevo contacto habrá que volver a

mostrar el registro anterior, mostrar los 4 botones de movimiento 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["id"] = TxtID.text
  VarResult["Nombre"] = txtNombre.text
  VarResult["Apellido"] = txtApellido.text
  VarResult["FechaNacimiento"] = TxtFechaNacimiento.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 que se encuentra editando:

```
Public Sub MnuEliminar_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")
    If TablaContactos.Available Then MostrarCampos
  End If
End
```

Sólo resta 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
```

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