**Lazarus + SQLite para principiantes.**

Autor: Ing. Héctor F. Fiandor Rosario

Email: hfiandor@infomed.sld.cu

Enero 2014

**Contenido**

[Introducción](#Introduccion)

[Algunas notas sobre Lazarus y SQLite](#AlgunasNotas)

[Obtención del fichero SQLite\_\_.dll y otros utilitarios](#Obtencion)

[Trabajos preliminares a la creación de la aplicación](#Trabajos)

[Creación de la aplicación](#Creacion)

[Realizar “consultas”](#Consultas)

[A modo de resumen](#Resumen)

[Bibliografía](#Bibliografia)

**Introducción**

El presente trabajo está dirigido a principiantes en tareas de programación con lenguaje Pascal y usando el Lazarus, que tienen una experiencia mínima en la creación de aplicaciones relativamente sencillas y algún conocimiento sobre “bases de datos”.

Se ha escogido el producto SQLite teniendo en cuenta que acopla con el Lazarus y que requiere un mínimo de recursos fácilmente adquiribles para realizar su trabajo.

En el trabajo se exponen algunas experiencias que pueden y deben ser punto de partida para empeños mas avanzados, y se ha tratado de usar ejemplos sencillos, perfectamente reproducibles.

El autor esta convencido de que solamente se ha asomado al complejo mundo de las Bases de Datos y aspira a incentivar el interés de los principiantes en ese mundo.

**Algunas notas sobre Lazarus y SQLite**

Lazarus es una herramienta de desarrollo libre y de código abierto para el compilador Free Pascal (Object Pascal), que a su vez lo es también. El IDE Lazarus ([Captura de pantalla](http://wiki.lazarus.freepascal.org/images/3/31/Lazarus_IDE_general.png)) es un entorno de programación estable y con abundantes características para crear aplicaciones tanto con interfaz gráficas como de consola. Actualmente funciona en sistemas operativos Linux, FreeBSD, Mac OS X, Windows 32/64 bits, y varios mas. Proporciona un editor de código fuente personalizable y un entorno visual para crear formularios, junto con un generador de paquetes, un depurador y una completa integración del GUI con el compilador FreePascal.(1)

SQLite es un motor de bases de datos basado en SQL, que requiere que tanto el desarrollador de una aplicación como el usuario final, solamente instalen un fichero .dll de aprox. 585 kB. Se ha usado la versión SQLite 3.0 y es de dominio público. En (3) se puede consultar literatura sobre SQLite.

El autor ha usado el ambiente Windows, pero es perfectamente adaptable a otros sistemas operativos.

**Obtención del fichero SQLite\_\_.dll y otros utilitarios**

Para trabajar con bases de datos basadas en SQLite, se requiere que esté presente un fichero .dll que puede ser obtenido en [www.sqlite.org](http://www.sqlite.org/).

Se recomienda instalar un producto similar al Navicat Premium versión 10.0.7 – Premium comercializado por <http://www.navicat.com/support>, u otro similar. Este producto resulta muy útil para visualizar el contenido de las tablas y modificar su estructura de forma simple.

**Trabajos preliminares a la creación de la aplicación**

Los especialistas recomiendan que para abordar un trabajo de base de datos resulta conveniente realizar SIEMPRE un trabajo previo “de mesa”, donde se conozca el objetivo de la base de datos y otros pormenores que debe brindar la aplicación. Se estudien las variables que tendrá el problema y se comience un diseño preliminar de las distintas tablas y consultas necesarias y se diseñen las ventanas de entrada/salida de los datos. En esta etapa se deben diseñar los cálculos (si se requieren) que deberán hacerse y decidir si se incluyen en las tablas o no.

**Creación de la aplicación**

Diseño de la aplicación: en esta etapa debemos decidir si nuestra aplicación tendrá una portada y una ventana de “final”. Decidir como será la pantalla de “navegación” dentro de la base de datos, de acuerdo con las tablas que hayamos diseñado para las distintas funciones.

La creación de la aplicación se hace de la manera acostumbrada, haciendo click en la pestaña Archivo/Nuevo/Aplicación. Con esto se crea una unit1.pas con su form asociada (que generalmente no usamos, pero no eliminar del Proyecto por el momento ).

Inmediatamente vamos Proyecto/Opciones del proyecto y seleccionamos un título para la aplicación, cargamos un icono para la misma, y en Opciones del Compilador/Enlazando, si queremos desentendernos de la Depuración, desmarcamos la casilla “Generar información de depuración…”. haciendo click en el botón “Aceptar” y se crea la aplicación deseada.

Si revisamos Proyecto/Opciones del proyecto/Formulario podemos observar que la Form1 de la unit1.pas es la única que aparece en el cuadro “Crear formularios automáticamente” y dependiendo de la configuración propia que se haya adoptado, aparecerá o no marcada la casilla “Al crear formularios nuevos…” que aparece en la parte inferior del cuadro de diálogo. La función de esta casilla es controlar si existirá en el archivo .lpr una instrucción similar a la siguiente: Application.CreateForm(TFormX.FormX) para cada nuevo formulario que se añada. Como se verá mas adelante, la experiencia del autor, para este tipo de aplicación, le resulta suficiente con que se creen el Formulario de la uDM y de la uFondo, como se verá mas adelante.

En este momento podemos escoger Salvar el trabajo realizado y mediante cuadros conocidos, escogemos el subdirectorio adecuado y ofrecerá salvar la unit1.pas (lo que aceptamos) y el proyecto, que se sugiere se le de el mismo nombre que a la aplicación.

Algunas units y rutinas convenientes: la experiencia de algunos especialistas recomiendan dedicar una unit independiente para las variables donde se declaran los tipos, las constantes y las variables globales (se sugiere nombrarla variables.pas), y otra para las rutinas (procedures y functions) comunes (se sugiere nombrarla varios.pas).

Si se cuenta con units de trabajos anteriores que puedan ser utilizadas en esta aplicación (la de variables, la de varios, la portada, el final, entre otras), que puedan ser adaptadas o “adoptadas” por la Aplicación, colocaremos sus ficheros en el subdirectorio donde este guardada o salvada la aplicación, y la incorporamos al Proyecto mediante las teclas CTRL+O, seleccionamos la unit que queremos y después, con ella activa, oprimimos CTRL+F10 y la adicionamos al Proyecto.

Los elementos gráficos que contienen texto, como los botones, en forma predeterminada aparecen en Ingles, para lograr que aparezcan en español, por ejemplo: Si por Yes, etc. Introducimos en la unit varios.pas el procedimiento TranslateLCL que tendrá un contenido como el que sigue:

procedure TranslateLCL;

var

PODirectory, Lang, FallbackLang: String;

begin

//PODirectory:='f:\lazarus\lcl\languages\';

PODirectory:='c:\lazarus\lcl\languages\';

Lang:='es';

FallbackLang:='';

LCLGetLanguageIDs(Lang,FallbackLang); // in unit LCLProc

Translations.TranslateUnitResourceStrings('LCLStrConsts',

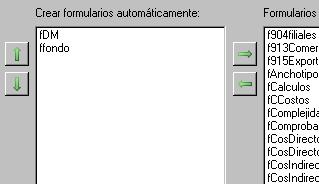
PODirectory+'lclstrconsts.%s.po',Lang,FallbackLang);

// ... add here a TranslateUnitResourceStrings call for every po file ...

end;

y en el fichero <programa>.lpr colocamos una llamada a dicho procedure.

En algunos Proyectos, donde las distintas ventanas no cubren toda la pantalla, resulta conveniente crear una unit que cree un fondo para toda la pantalla, y sobre dicho fondo, irán después apareciendo el resto de las pantallas. Esta unit (se sugiere llamarla ufondo.pas) será la pantalla principal de la Aplicación. Si asumimos que tenemos algo similar de otras aplicaciones, podemos copiarla en el subdirectorio, incorporarla al Proyecto y arreglar en Proyecto/Opciones del Proyecto/Formulario para que sea el formulario a aparecer cuando se active la Aplicación. La apariencia sería aproximadamente así:



En este momento podemos eliminar del Proyecto la unit1.pas (y su Form1 asociada), primero pasándola al cuadro “Formularios disponibles” en Proyecto/Opciones del Proyecto/Formulario, y después en Proyecto/Eliminar del Proyecto marcándola y oprimiendo el botón Aceptar.

Resulta conveniente dedicar una unit muy especial para colocar los componentes DataAccess (DataModule), que podemos llamar uDM.pas y a su correspondiente formulario fdm. Este formulario debe colocarse en Proyectos/Opciones del Proyecto/Formulario como primer formulario a crear automáticamente, como se muestra anteriormente.

Con estos ajustes, el fichero <programa>.lpr, debe quedar como sigue:

{$R \*.res}

begin

TranslateLCL;

RequireDerivedFormResource := True;

Application.Initialize;

Application.CreateForm(TfDM, fDM);

Application.CreateForm(Tffondo, ffondo);

Application.Run;

end.

En esta uDM podemos crear las tablas, haciendo doble click sobre el formulario, que prepara el evento OnCreate, y en el mismo podemos añadir el código necesario. Como ejemplo para una tabla pudiera ser similar al siguiente:

{ aqui es donde se crean las tablas }

//-------------------------------

with fdm.tCuentas do

begin

fdm.tCuentas.FileName := camino\_actual + '\Tablas.db';

PrimaryKey := 'IdCuentas';

if not TableExists('Cuentas') then

begin

FieldDefs.Clear;

FieldDefs.Add('IdCuentas', ftAutoInc);

FieldDefs.Add('NoCuentas', ftString);

FieldDefs.Add('NomCuentas', ftString);

CreateTable;

end

else

begin

end;

end;

Con el código anterior se ha creado una tabla que tiene tres campos. De forma similar se crean todas las tablas de nuestra base de datos.

Debe observarse que se ha desmarcado la casilla correspondiente a “Al crear formularios nuevo, …” en Proyecto/Opciones del Proyecto/Formulario. El resto de los formularios se crearán cuando se necesiten mediante las siguientes rutinas como ejemplo:

procedure UnFormulario.CerrarYMostrarSeleccion3;

...

Close;

fSeleccion3:=TfSeleccion3.Create(application);

fSeleccion3.Show;

...

end;

donde Close cierra el formulario “UnFormulario” y fSelección3 es el formulario que queremos crear y que aparezca. En Proyecto/Opciones del proyecto/Formularios deben aparecer sus nombres en la casida de la derecha.

Instalación de componentes: Ya se ha señalado que en la unit uDM.pas se colocarán los componentes DataAccess. El resto de los componentes necesarios se hará en cada formulario en particular.

Creación de un formulario de navegación y formularios para introducir datos: como se mencionó anteriormente, resulta conveniente disponer de un formulario que permita seleccionar en que parte de la Aplicación se va a trabajar. Este es un formulario relativamente sencillo, con una apariencia similar a la siguiente:



Como puede apreciarse, en el formulario sólo hay botones, que al presionarlos hacen aparecer el formulario que se desea y que está asociado con una función determinada

Como ejemplo, tenemos la función asociada con el Clasificador de Cuentas, que al presionar el botón “Cuentas“ aparece un formulario como el que sigue:

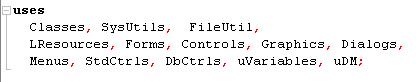


Este formulario tiene una doble función: permite ver lo que tiene la base de datos correspondiente (si existe) y permite introducir nuevas cuentas o modificar las que ya existen. Obsérvese que se ha colocado una “barra de navegación” en el pie del formulario para moverse a través de la tabla asociada. Además cuenta con una barra de herramientas en la parte superior para funciones específicas con dicha base de datos.

El diseño del mismo se ha hecho utilizando Controles DBEdit, pero puede hacerse utilizando DBGrid.

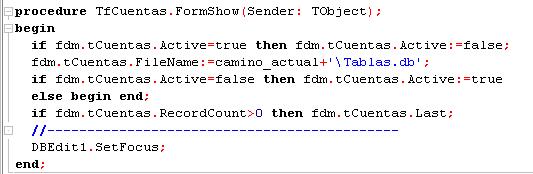
Rutinas comunes para abrir y cerrar una tabla, introducción (modificación) de datos, realización de cálculos: Para estos temas, usaremos como ejemplo el Clasificador de Cuentas visto anteriormente.

En la unit asociada al botón “Cuentas” –uCuentas.pas– hemos insertado el nombre de las units a usar por dicha unidad, como sigue:



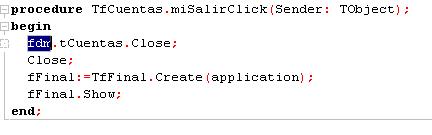
Obsérvese que se ha incluido la uDM para hacer referencia al componente tCuentas (tabla tCuentas) que será la que contenga la información de esta base de datos.

Para abrir la tabla, en el evento fCuentas.FormShow se ha colocado el siguiente código:

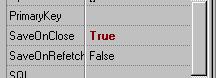


Obsérvese que se ha incluido en la segunda línea el “camino” donde estará el fichero Tablas.db, que contiene las diferentes tablas de la base de datos. Además se ha incluido una línea de código para que si la tabla no está vacía, vaya al último Registro.

Con respecto al cierre de la tabla, en las propiedades de la misma hay una de ellas que asegura salvar la información y en el evento bSalir.OnClick se ha incluido el siguiente código:



La propiedad de la tabla que asegura salvar tiene una apariencia similar a la siguiente:



La línea de código tCuentas.Close; puede utilizarse en cualquier otro evento, teniendo en cuenta que si se va a utilizar de nuevo la tabla, hay que abrirla otra vez.

El uso u obtención de un dato por la vía del cálculo, utilizando uno o más campos de una tabla (o de distintas tablas) tiene una especificidad: hay que decidir si dicho resultado se va a incluir como un campo en la tabla, o se conservará como una variable independiente para mostrarla o darle otro uso más adelante en la Aplicación.

A continuación se muestra una de las formas de acceder a la información que tiene una tabla. Supongamos que tenemos un fichero texto (.txt) con la información que debemos introducir en la tabla:

procedure TfCuentas.miIntroducirDatosClick(Sender: TObject);

begin

if FileExists(camino\_actual+'\Cuentas.txt') then

begin

nuevo\_Nombre:=camino\_actual+'\Cuentas.txt';

AssignFile(fdat,nuevo\_nombre);

{$I-} reset(fdat); {$I+};

fdm.tCuentas.First;

while not eof(fdat) do

begin

fdm.tCuentas.Insert;

lectura\_nom(fdat,st01,c); NoCuentas:=st01;

fdm.tCuentas.FieldByName('NoCuentas').AsString:=NoCuentas;

lectura\_nom(fdat,st01,c); NomCuentas:=st01;

fdm.tCuentas.FieldByName('NomCuentas').AsString:=NomCuentas;

fdm.tCuentas.Post;

fdm.tCuentas.ApplyUpdates;

readln(fdat); //saltar a la proxima linea

if not eof(fdat) then

begin

fdm.tCuentas.Next;

end;

end;

CloseFile(fdat);

miSalvarTxtClick(Sender);

fdm.tCuentas.Close;

fdm.tCuentas.FileName:=camino\_actual+'\Tablas.db';

if fdm.tCuentas.Active=false then fdm.tCuentas.Active:=true

else begin end;

if fdm.tCuentas.RecordCount>0 then fdm.tCuentas.Last;

end

else ShowMessagePos('El file Cuentas.txt no existe.',10,10);

end;

Otra forma es a través del formulario de entrada (visto anteriormente), donde se usará la barra de navegación para lograr que se introduzcan los datos dados en los componentes DBEdit.

Si se quisiera realizar un cálculo utilizando campos de la misma tabla, y el resultado se quisiera salvar en otro campo de dicha tabla, tendríamos el siguiente código:

//hallar sumas de Indirectos + traspasosxNoIngresos

for i := 1 to fdm.tCosIndirectosLA.RecordCount do

begin

fdm.tCosIndirectosLA.RecNo := i;

fdm.tCosIndirectosLA.Edit;

fdm.tCosIndirectosLA.FieldByName('MNSuma').AsFloat :=

fdm.tCosIndirectosLA.FieldByName('MN').AsFloat +

fdm.tCosIndirectosLA.FieldByName('MNTraslNoIngresos').AsFloat;

fdm.tCosIndirectosLA.FieldByName('CUCSuma').AsFloat :=

fdm.tCosIndirectosLA.FieldByName('CUC').AsFloat +

fdm.tCosIndirectosLA.FieldByName('CUCTraslNoIngresos').AsFloat;

fdm.tCosIndirectosLA.Post;

fdm.tCosIndirectosLA.ApplyUpdates;

ProgressBar1.Position := 10\*i;

end;

log('Terminé traspaso por no ingresos LA');

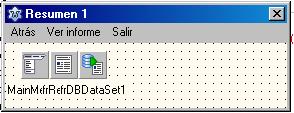
//-------------------------------------------------

En el caso anterior, en la Tabla tCosIndirectosLA, se ha sumado el campo MN+MNTraslNoIngresos y dicha suma se ha asignado al campo MNSuma (lo mismo para los campos CUC). Obsérvese que al inicio siempre hay que declarar el evento Insert o Edit o Append para añadir al final de la tabla y para fijar los nuevos valores hay que declarar los eventos Post y ApplyUpdates;

Diseño de Reportes y rutinas comunes asociadas a Reportes: Lazarus cuenta con el reporteador LazReport que brinda en una forma sencilla, la posibilidad de diseñar reportes adecuados a nuestra Aplicación.

El diseño del Reporte puede consultarse en el subdirectorio /Lazarus/examples/ donde hay un tutorial para el LazReport.

En nuestra Aplicación, una vez diseñado el Reporte, colocamos en el formulario correspondiente los componentes necesarios asociados al Reporte como sigue:



y en el editor de código, en el evento miVerInforme.OnClick escribimos:

procedure TfResumen1.miVerInformeClick(Sender: TObject);

begin

frReport1.LoadFromFile('Resumen1.lrf');

frReport1.ShowReport;

mousePos.x:=225; //miSalir

mousePos.y:=-10;

Mouse.CursorPos := ClientToScreen(mousePos) ;

end;

**Realizar “consultas”**

Esta es una de las funciones básicas de un Sistema de Base de Datos y que puede resumirse como sigue: de una base de datos, donde hay cientos o miles de Registros con información, seleccionar los que cumplan determinados requisitos. En este aspecto, el lenguaje SQL cumple una función excepcional, ya que permite, mediante una instrucción adecuada, realizar distintas funciones sobre la tabla que hayamos seleccionado.

Cada Aplicación tiene sus características propias, y las “consultas” serán diseñadas teniendo esto en cuenta, respondiendo siempre a la pregunta ¿qué se quiere?.

Pongamos por ejemplo una Base de Datos correspondiente a un *Curso de superación profesional*, y queremos mostrar una consulta con *todos* los alumnos. En una unit correspondiente a dicha consulta incorporamos el componente SQLQuery que tomamos de la pestaña SQLdb. El diseño del formulario será similar al que sigue:



Al hacer click en la barra de herramientas “Hacer consulta” se ejecuta el siguiente código:

procedure TfcTodos.miHacerConsultaClick(Sender: TObject);

var cTodos:TSQLQuery;

sqltext1,sqltext2,sqltext3,sqltext4,SQLText:string;

begin

fcTodos.tcTodos.FileName:=camino\_actual+'\estudiantes.db';

fcTodos.tcTodos.SQL:='cTodos.Delete'; //debe vaciarse la tabla

fcTodos.tcTodos.FieldDefs.Add('Alumno', ftString);

fcTodos.tcTodos.FieldDefs.Add('Grupo', ftString);

fcTodos.tcTodos.FieldDefs.Add('Asignatura', ftString);

fcTodos.tcTodos.FieldDefs.Add('Profesor', ftString);

fcTodos.tcTodos.FieldDefs.Add('Nota', ftInteger);

fcTodos.tcTodos.Active:=false;

sqltext1:='SELECT Alumnos.Alumno,Alumnos.Grupo,Profesores.Profesor,Profesores.Asignatura,Notas.Nota ';

sqltext2:='FROM (Alumnos INNER JOIN Profesores ON Alumnos.IdGrupo = Profesores.IdGrupo)';

sqltext3:='INNER JOIN Notas ON Alumnos.IdAlumno = Notas.IdAlumno AND Profesores.IdAsignatura = Notas.IdAsignatura ';

sqltext4:='ORDER BY Alumnos.IdAlumno';

SQLText:=sqltext1+sqltext2+sqltext3+sqltext4;

fcTodos.tcTodos.SQL:=SQLText;

// try

fcTodos.tcTodos.Active:=true;

// except

//...

// end;

mousePos.x:=190;

mousePos.y:=-10;

Mouse.CursorPos := ClientToScreen(mousePos) ;

end;

y el cursor pasa en la barra de herramientas al botón “Ver informe”, que al hacer click, se ejecuta el siguiente código:

procedure TfcTodos.miVerInformeClick(Sender: TObject);

begin

if FileExists(camino\_actual+'\cTodos.lrf') then

begin

frReport1.LoadFromFile('cTodos.lrf');

frReport1.ShowReport;

end

else ShowMessage('El fichero cTodos.lrf no existe');

mousePos.x:=25;

mousePos.y:=-10;

Mouse.CursorPos := ClientToScreen(mousePos) ;

end;

y aparece en pantalla el siguiente Reporte:



que no es más que un Reporte con la información de la Consulta.

**A modo de resumen**

Se ha indicado cómo obtener el fichero .dll necesario para operar bases de datos basadas en el producto SQLite, y como manejarlo en el Lazarus.

Se ha mencionado como incluir la instrucción Translate y crear un formulario como “fondo” para una mejor estética.

Se ha expuesto la utilidad de crear una unit (uDM.pas) para contener en su formulario los componentes Data Access correspondientes a las tablas que se quieren incluir en la Aplicación y no tener que incluirlas en cada formulario correspondiente a las distintas funciones que incluye la Aplicación.

Se ha mencionado las ventajas de crear una unit para las variables globales y otra para rutinas comunes (uVariables.pas y uVarios.pas).

Se muestran ejemplos para crear tablas y cómo acceder a la información que contienen. Se incluyen ejemplos para campos calculados.

Se muestran ejemplos para realizar consultas y su código asociado, así como para realizar Reportes.

No se ha mencionado los pasos necesarios para crear los Formularios para la introducción de los datos y como enlazar los componentes con las Tablas.

No se ha tratado como introducir algún cambio en el diseño de las Tablas.

Pensando en un principiante, que tenga poca experiencia en el uso de Lazarus y en aplicaciones que usen bases de datos, se espera que surjan muchas preguntas y de esa manera aprendemos todos.

**Bibliografía**

(1) Lazarus Tutorial/es. [About Free Pascal wiki](http://wiki.lazarus.freepascal.org/Free_Pascal_wiki:About)

(2) Lazarus Database Tutorial/es.

<http://wiki.lazarus.freepascal.org/Lazarus_Database_Tutorial/es>

(3) sqlite-doc-3071300\about.html

(4) Lazarus and SQLite. http://wiki.lazarus.freepascal.org.