Manual de usuario

Guía rápida

LAZARUS 

free pascal 

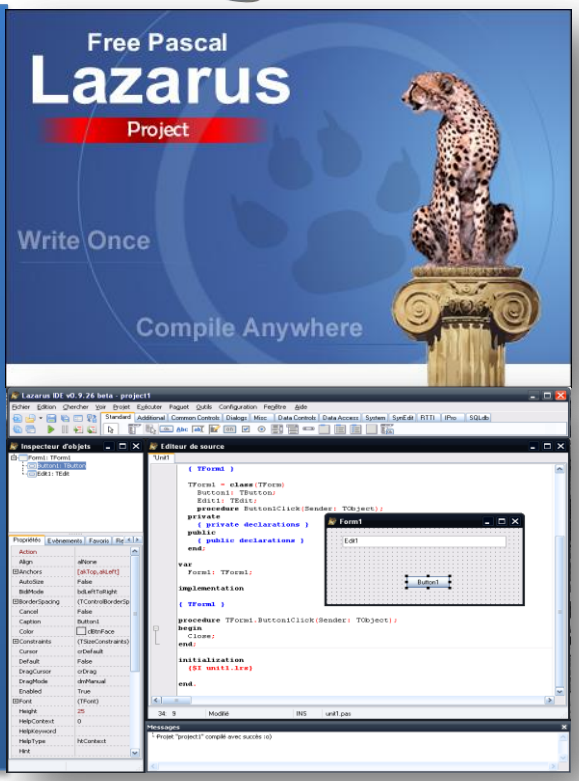
OPEN SOURCE COMPILER FOR PASCAL AND OBJECT PASCAL

Autor:
Luis Tobar Cabezas

Tabla de contenidos

¿Qué es Lazarus?

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 de Lazarus es un entorno de programación estable y con abundantes características para crear aplicaciones tanto con interfaz gráficas como de consola.



Actualmente funciona n sistemas operativos Linux, FreeBSD y Windows 32 bits, y 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.

¿Qué es Pascal?

Pascal es un lenguaje de alto nivel, y de propósito general, lo cual quiere decir que se puede utilizar para cualquier tipo de propósitos. El lenguaje de programación en Pascal se considera un lenguaje estructurado, sencillo y práctico para todos aquellos usuarios que se inician en el mundo de la programación, ya que fue creado con fines de aprendizaje.

Notas
Lenguaje de alto nivel: *lenguajes en los cuales un programador puede utilizar palabras de fácil compresión o expresiones sintácticas similares al inglés de una manera adecuada a la capacidad cognitiva humana*

Características

- Es un lenguaje de programación de alto nivel.
- Es un excelente lenguaje para quienes empiezan a programar debido su similitud con el pseudocódigo.
- Es un lenguaje de los llamados de propósito general, es decir, sirve para desarrollar aplicaciones de diversos tipos.
- Aplica la programación por módulos ya que utiliza procedimientos y funciones, esto ayuda a la estructuración del código.
- Maneja tanto datos simples y estructurados como aquellos definidos por el usuario.

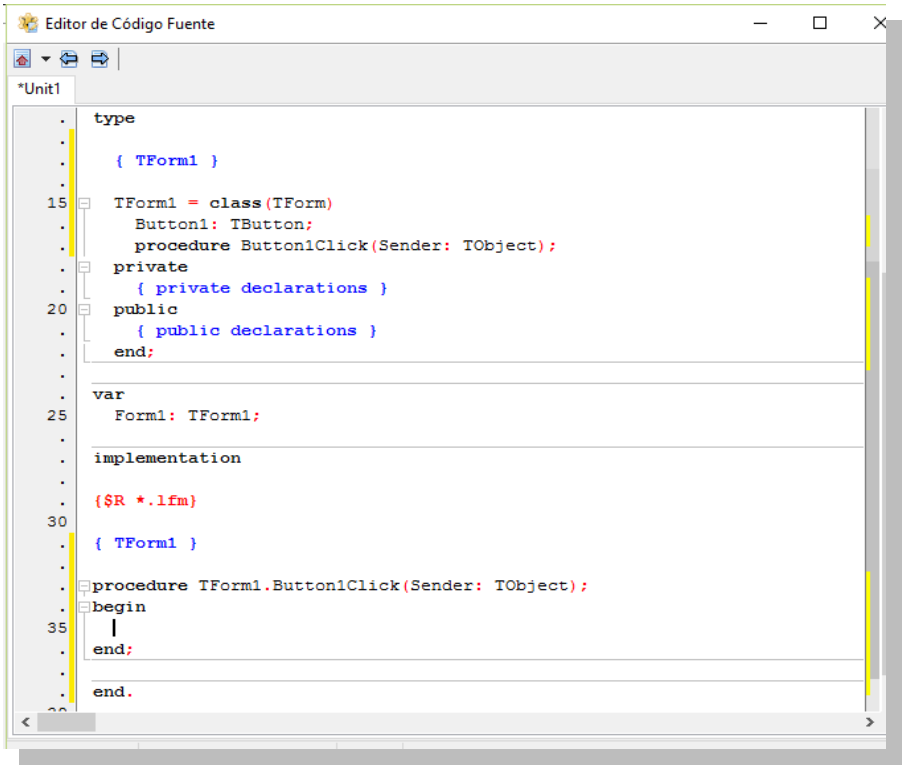
Entorno de desarrollo de Lazarus

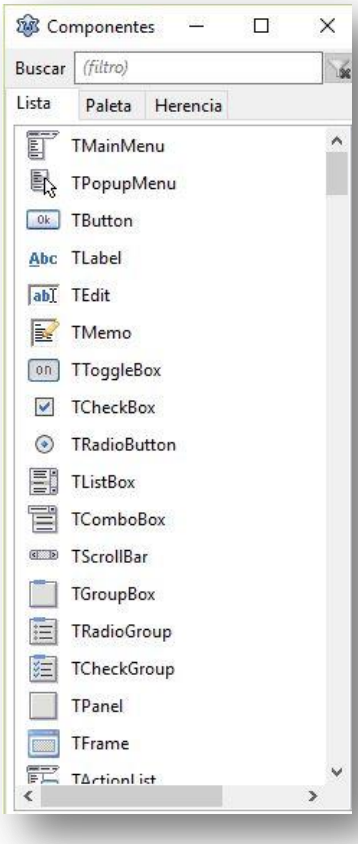
Ventana principal:

Ésta es la ventana principal que controla el proyecto, y contiene el Menú Principal, la Paleta de Componentes y el panel de Botones Rápidos.

Editor de código fuente:

La ventana principal donde se edita el código fuente. Su funcionamiento es muy parecido a la mayoría de los editores de texto gráficos, así que el ratón puede mover el cursor sobre el texto mostrado, y al hacer clic con el botón izquierdo mientras arrastramos se seleccionará y sombreará texto.

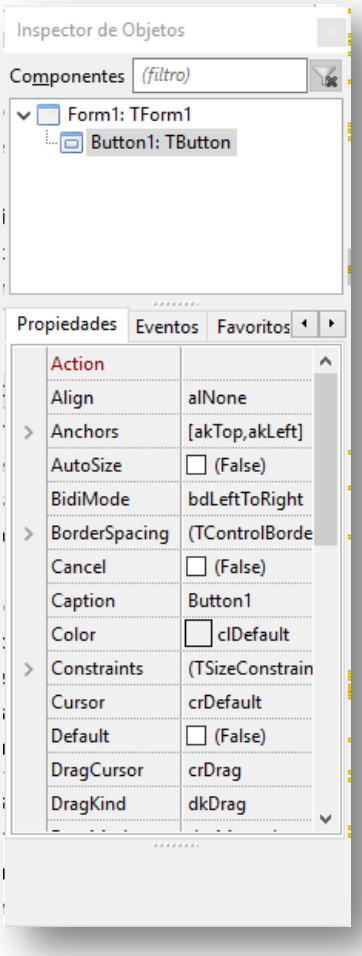




Componentes:
Muestra un gran número de iconos que representan los componentes más utilizados para construir formularios.

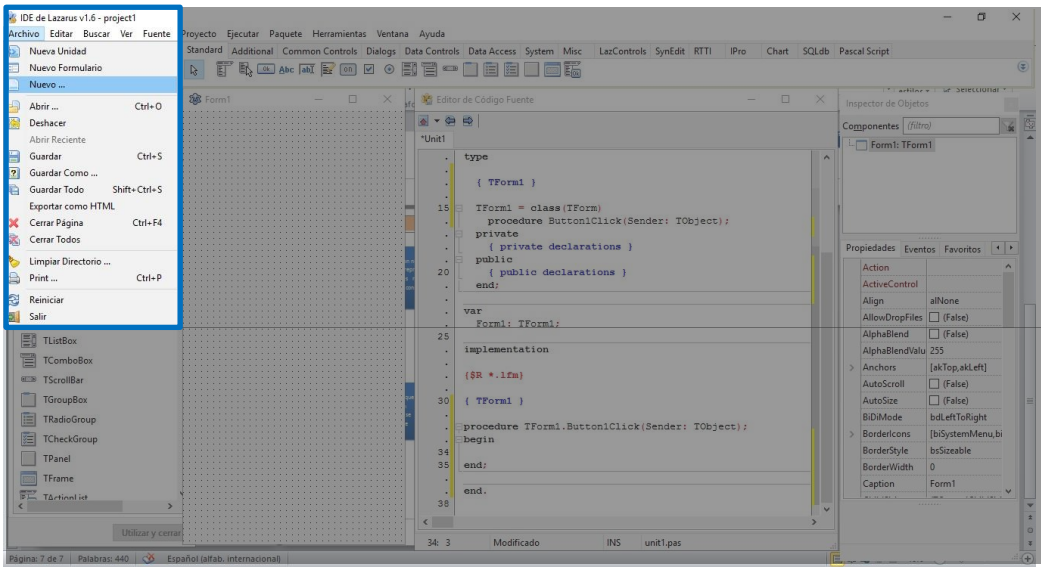
Notas:
Formulario: *es una sección del documento destinada a que el usuario introduzca o visualice datos que van a ser enviados a algún lado.*

Inspector de objetos:
Muestra las propiedades del formulario que aparece en él. Si hace clic con el ratón en cualquier componente de un formulario se mostrarán los detalles de ese componente en el Inspector de Objetos.

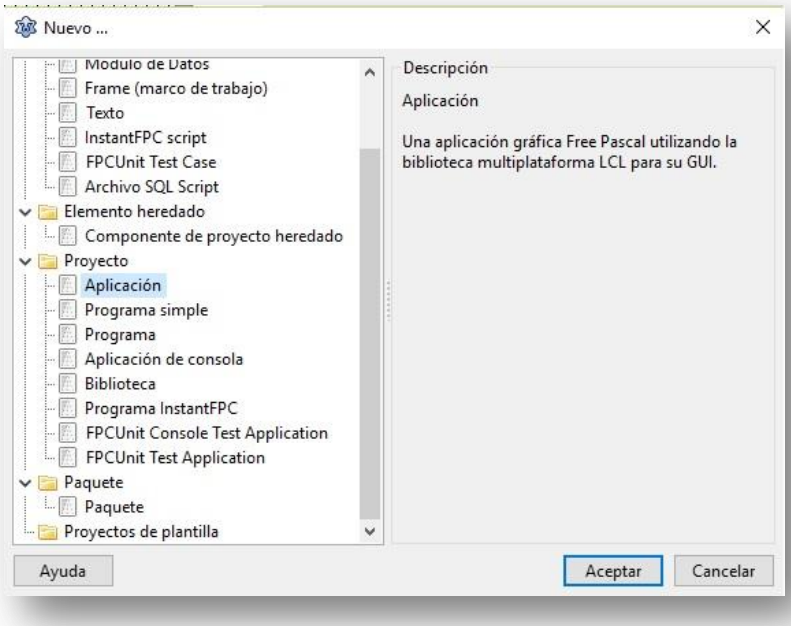


Creación de un nuevo proyecto

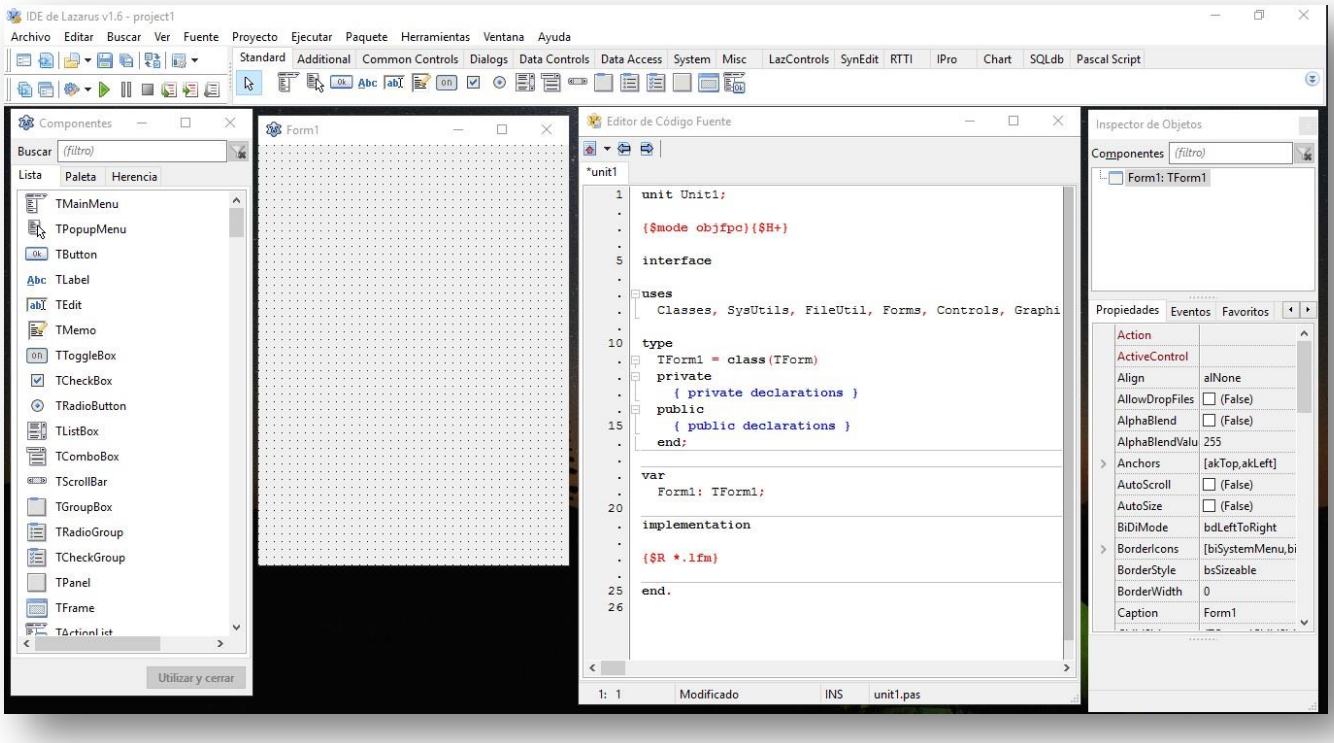
Primero nos dirigimos arriba a la izquierda como se resalta en la imagen y hacemos clic en “Nuevo”.



Luego se nos abrirá una ventana en la cual debemos buscar el menú proyecto, luego en el seleccionar la opción “Aplicación”.



Por último se nos abrirá la ventana principal con todos los componentes necesarios para trabajar en nuestro nuevo proyecto.

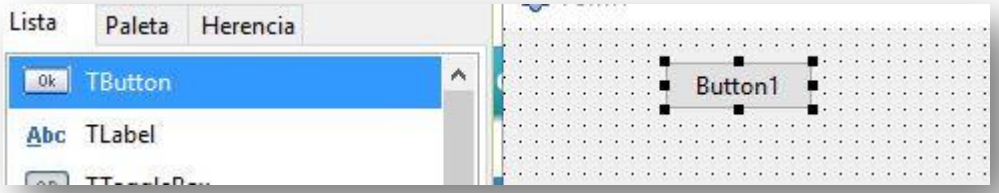


Objetos básicos

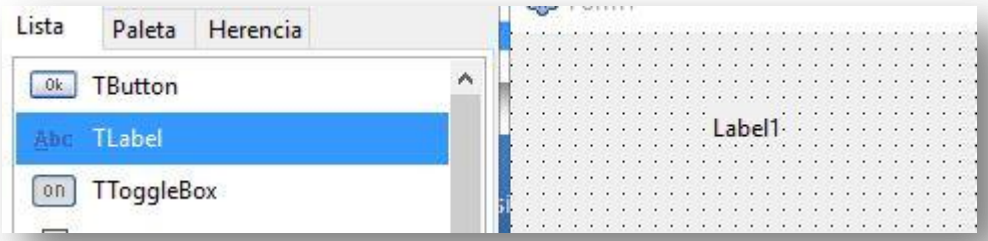
TEdit:
También conocido como textbox, permite la inserción de datos por parte del usuario.

Notas:
Objeto: *es una unidad dentro de un programa que consta de un estado y de un comportamiento, que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.*

TButton:
O conocido simplemente como button, permite programar acciones dentro de él, las cuales se ejecutaran una vez que el usuario pinche en él.

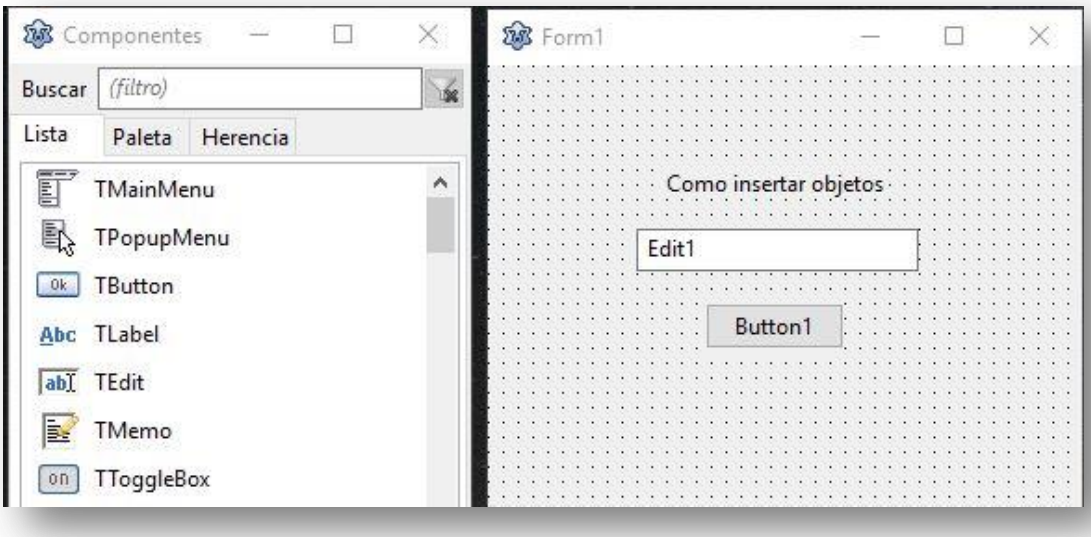


TLabel:
O conocido simplemente como label, generalmente se usa para nombrar objetos dentro de un form o visualizar información.



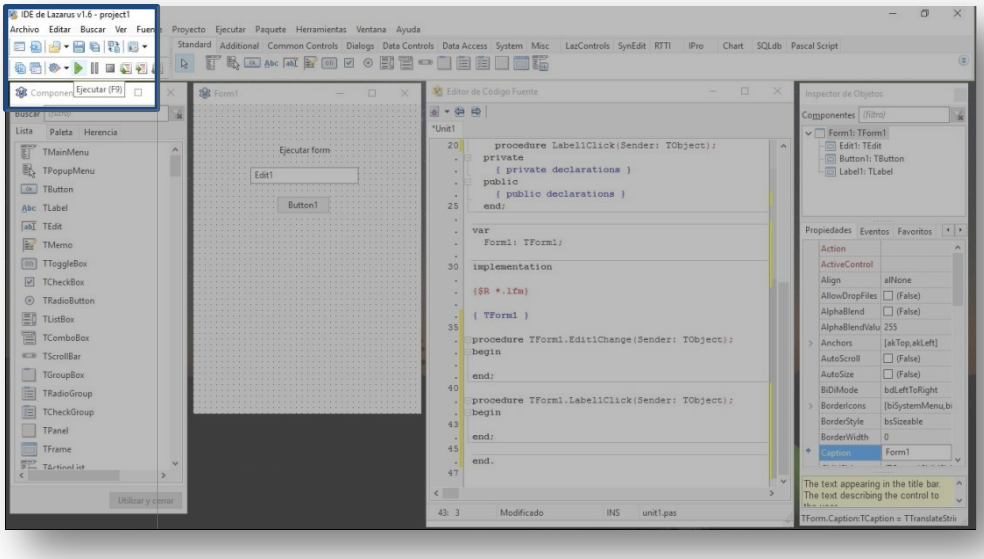
Inserción de objetos en el formulario

La inserción de un objeto dentro de nuestro form es una tarea muy sencilla, solo busque en el cuadro de componentes el objeto a insertar, haga clic sobre él y luego haga un clic sobre el formulario en la ubicación que desea colocar el objeto.

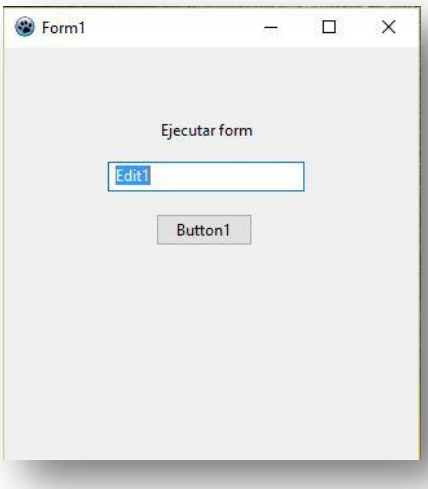


Ejecución del formulario

Primero nos dirigiremos a la parte superior izquierda de nuestro IDE y haremos clic en el icono verde de “play” como se ve en la imagen.



Una vez ejecutado el programa, esperaremos a que este compile y al finalizar veremos nuestro form ejecutándose.



Notas:
Compilar: *Proceso de traducción de un código fuente (escrito en un lenguaje de programación de alto nivel) a lenguaje máquina (código objeto) para que pueda ser ejecutado por la computadora.*

Estructura de un programa en pascal

```
program {nombre del programa};
const
    {declaración de constantes}
var
    {declaración de variables}
begin
    {código del programa}
end.
```

Declaración de variables

En la sección de declaración de variables, todas las variables que son utilizadas por el programa deben ser especificadas junto con su tipo.

```
{variable}: {tipo};

Ejemplo:

program Ejemplo;
var
    x: Real;
    y: Real;
    n: Integer;
begin
    Read(x);
    Read(y);
    Read(n);
    WriteLn(n * (x * x + y * y));
end.
```

Sentencia If

La sentencia if-then (en español: «si-entonces») ejecuta instrucciones sólo si se cumple una condición. Si la condición es falsa, no se hace nada.

```
if {condición} then
    {sentencia en el caso verdadero};

Ejemplo:

Read(nota);
if nota >= 55 then
    WriteLn('Felicitaciones');
```

Ciclo For

El ciclo for («para») ejecuta una secuencia de instrucciones un número predeterminado de veces. Un ciclo for utiliza una variable de control que toma diferentes valores en cada iteración.

```
for {variable} := {valor inicial} to {valor
final} do
    {sentencia};

Ejemplo:

suma := 0;
for i := 1 to 10 do
    suma := suma + i;
```

Notas:
Variable de control: *es una variable que hace de contador dentro del ciclo for y que va aumentando a medida que pasa cada vuelta de ciclo.*

Ciclo While

El ciclo while («mientras») ejecuta una secuencia de instrucciones mientras una condición sea verdadera. La condición es evaluada antes de cada iteración. Si la condición es inicialmente falsa, el ciclo no se ejecutará ninguna vez.

```
while {condición} do
    {sentencia};

Ejemplo (secuencia de Collatz 3n+1):

Read(n);
while n > 1 do
begin
    if n mod 2 = 0 then
        n := n div 2
    else
        n := 3 * n + 1;
    WriteLn(n);
end;
```

Funciones

Una función es una sección de un programa que calcula un valor de manera independiente al resto del programa.
En esencia, una función es un mini programa: tiene una entrada, un proceso y una salida.

En un programa Pascal, las funciones deben ser definidas antes del begin que indica el comienzo del programa.

```
function {nombre}({parámetro}: {tipo}; ...): {tipo del resultado};
var
    {variables locales}
begin
    {código de la función}
end;

Ejemplo:

function F(x: Real): Real;
begin
    F := x * x * x * x * x + 1;
end;
```

Procedimientos

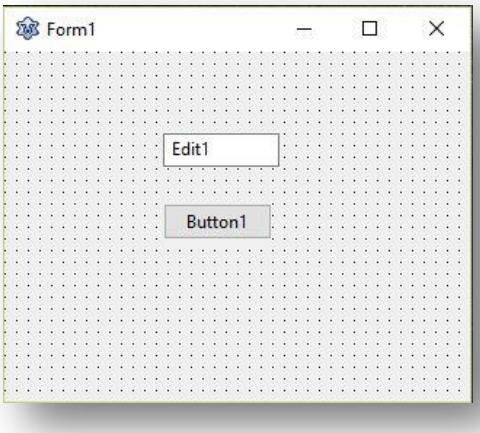
Un procedimiento es una sección de un programa (al igual que una función) que realiza varias sentencias de manera independiente al resto del programa. La diferencia con una función es que un procedimiento no entrega ningún valor como resultado.

```
procedure {nombre}({parámetro}: {tipo}; ...);
var
    {variables locales}
begin
    {código del procedimento}
end;

Ejemplo:

procedure LeerValores(m: Integer);
var
    i: Integer;
begin
    for i := 1 to m do
    begin
        Write('Ingrese valor ', i, ': ');
        Read(valores[i]);
    end;
end;
```

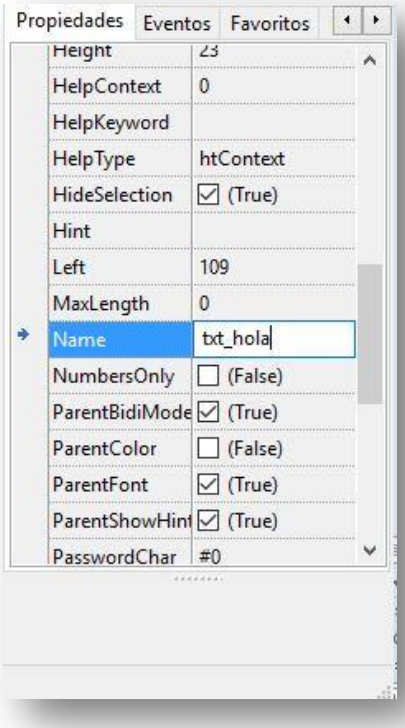
Creación de “Hola Mundo”

1° Creamos un nuevo proyecto (Véase página 6)

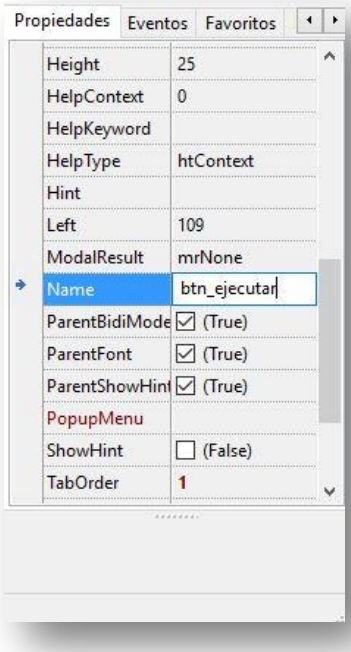
2° Agregamos un TEdit y un TButton a nuestro formulario quedando de la siguiente manera:



3° Seleccionamos nuestro TEdit y en el inspector de objetos, pestaña propiedades, lo renombramos como “txt_hola”



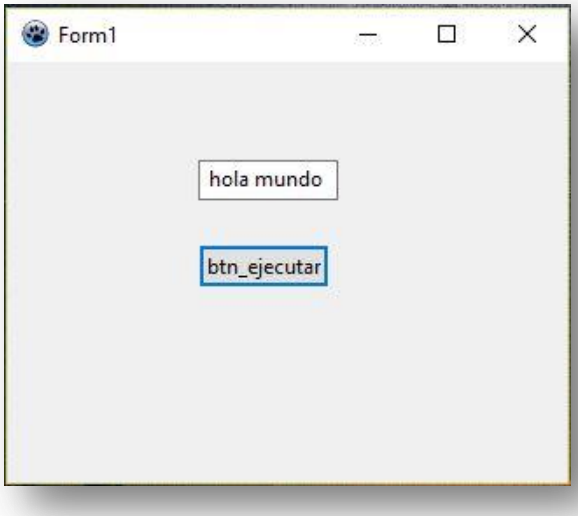
4° Seleccionamos nuestro TButton y en el inspector de objetos, pestaña propiedades, lo renombramos como “btn_ejecutar”



5° Hacemos doble clic sobre nuestro “btn_ejecutar” y entre nuestro “begin” y “end” escribimos lo siguiente:

```
procedure TForm1.btn_ejecutarClick(Sender: TObject);
begin
    txt_hola.Text:= 'hola mundo';
end;
```

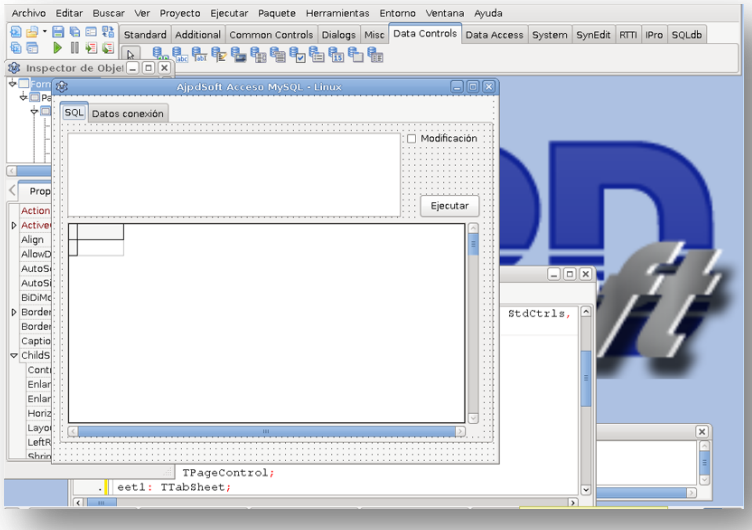
6° Finalmente ejecutamos nuestro programa (véase página ¿) y hacemos clic en nuestro “btn_ejecutar”



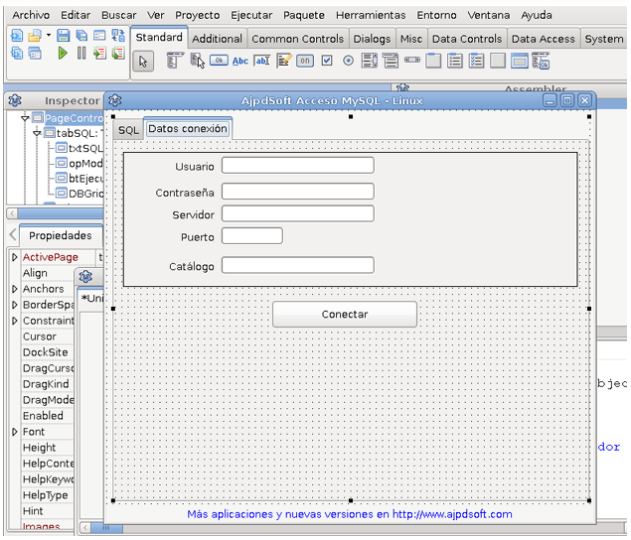
Conexión a Base de Datos

1° Creamos un nuevo proyecto (Véase pagina 6)

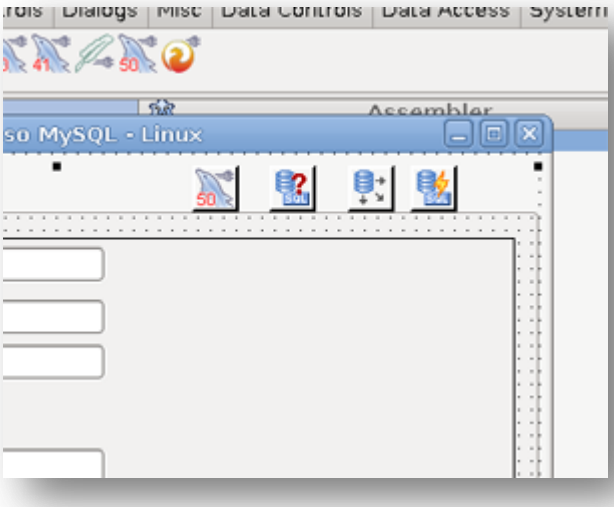
2° Desde la pestaña "Controles Comunes" agregaremos un TPageControl, pulsaremos con el botón derecho del ratón sobre el TPageControl y agregaremos dos pestañas (pulsando en "Insertar página"). En la primera pestaña del TPageControl añadiremos un TMemo, un TCheckBox y un TButton de la pestaña "Standard" de la paleta de componentes. Agregaremos también un TDBGrid de la pestaña "Data Controls" de la paleta de componentes:



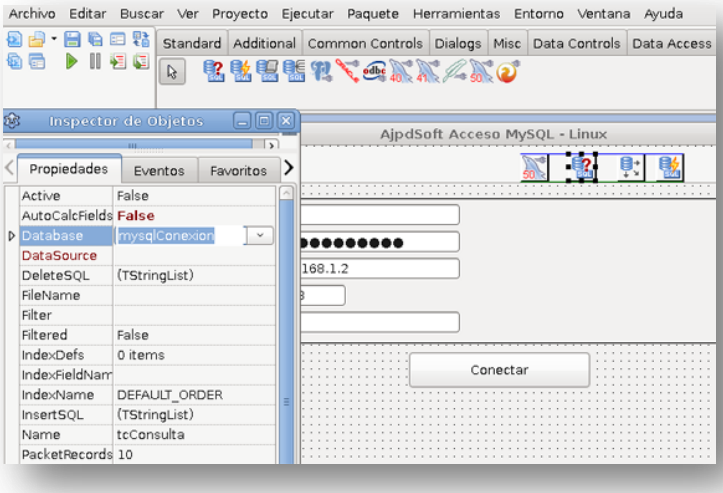
3° En la segunda pestaña del TPageControl añadiremos los componentes necesarios para los datos de acceso a MySQL Server, añadiremos varios TEdit: usuario, contraseña, servidor, puerto y catálogo. Añadiremos también un TButton:



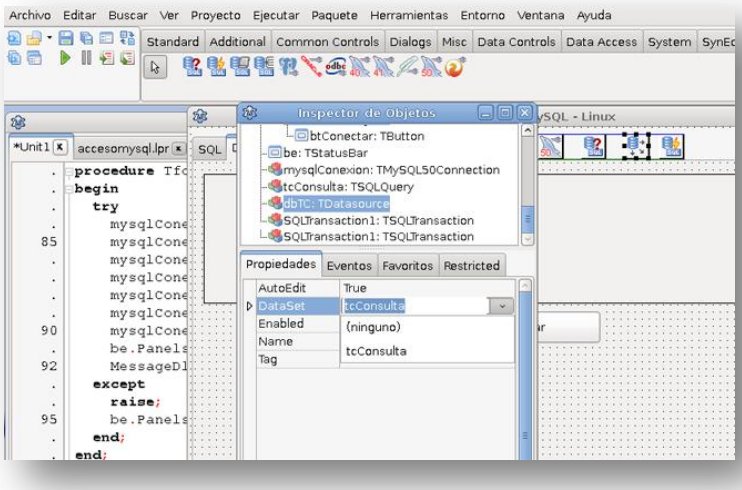
4° Desde la pestaña "SQLdb" agregaremos los siguientes componentes: TMySQL50Connection, TSQLQuery,TDataSource, TSQLTransaction.



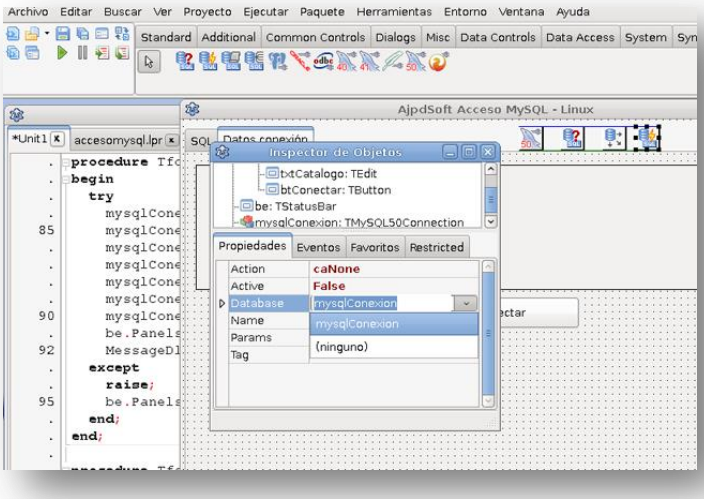
5° Configuraremos las siguientes propiedades del componente "TSQLQuery":
En la propiedad "Database", seleccionaremos el componente "TMySQL50Connection".
En la propiedad "Transaction", seleccionaremos el componente "TSQLTransaction".



6° Para el componente "TDatasource" configuraremos la propiedad "TDataset" seleccionando el TSQLQuery



7° Para el componente "TSQLTransaction", configuraremos la propiedad "Database" seleccionando el "TMySQL50Connection":



8° En el botón "Conectar" de la pestaña "Datos conexión" de nuestra aplicación, en el evento OnClick añadiremos el siguiente código:

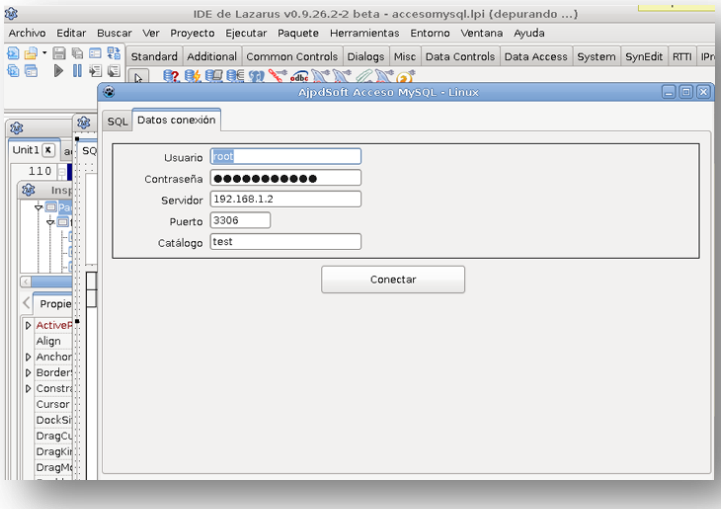
```
procedure TformMenuPrincipal.btConectarClick(Sender: TObject); begin
  try
    mysqlConexion.Close;
    mysqlConexion.UserName := txtUsuario.Text;
    mysqlConexion.Password := txtContrasena.Text;
    mysqlConexion.HostName := txtServidor.Text;
    mysqlConexion.port := StrToInt(txtPuerto.Text);
    mysqlConexion.DatabaseName:= txtCatalogo.Text;
    mysqlConexion.Open;
    be.Panels[1].Text := 'Conectado';
    MessageDlg('Conexión realizada con éxito.', mtInformation, [mbok], 0);
  except
    raise;
    be.Panels[1].Text := 'Error en la conexión';
  end;
end;
```

9° En el botón "Ejecutar" de la pestaña SQL de nuestra aplicación, en el evento OnClick, añadiremos el siguiente código:

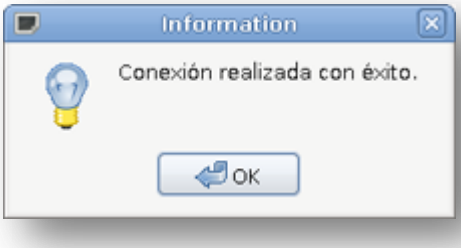
```
procedure TformMenuPrincipal.btEjecutarSQLClick(Sender: TObject);
begin
  if not mysqlConexion.Connected then
  begin
    MessageDlg('Primero debe realizar la conexión al servidor MySQL.',
        mtInformation, [mbok], 0);
    tabDatos.show;
  end
  else
  begin
    if txtSQL.Text <> '' then
    begin
      tcConsulta.Close;
      tcConsulta.SQL.Clear;
      tcConsulta.SQL.Add(txtSQL.Text);
      if opModificacion.Checked then
        tcConsulta.ExecSQL
      else
        tcConsulta.Open;
    end
    else
    begin
      MessageDlg('Introduzca una consulta SQL.',
          mtInformation, [mbok], 0);
      txtSQL.SetFocus;
    end;
  end;
```

10° Una vez añadidos todos los componentes y su código fuente compilaremos la aplicación desde el menú "Ejecutar" - "Ejecutar":

11° Si todo es correcto se ejecutará la aplicación, desde la pestaña "Datos conexión" introduciremos los datos del servidor de MySQL Server: usuario, contraseña, servidor, puerto, catálogo. Pulsaremos "Conectar":



12° Si el servidor de MySQL Server está activo, los datos de conexión son correctos y se establece la conexión correctamente nos mostrará el siguiente mensaje "Conexión realizada con éxito":



13° Desde la pestaña "SQL" podremos introducir cualquier consulta SQL y ejecutarla, mostrará el resultado en el DBGrid, por ejemplo “show tables”: