

Tutorial Programación Delphi N-capas Partes 1 y 2

Presentación :

Hola a tod@s,

hace mucho, mucho tiempo en una galaxia muy lejana. Bueno casi. Hace años hice un Cicloformativo Grado III o FP de Grado Superior en Desarrollo de productos electrónicos, ahí vimos como desarrollar un circuito en su totalidad, esto incluía su control, así que aprendí ANSI c y Ensamblador 8051. Mi personalidad curiosa y un curso de Pascal me hicieron indagar en Delphi 3. Pero hice cosas de “andar por casa”, programas para amigo y poco más. Mis primeros trabajos profesionales no fueron en Delphi, si no en PHP y HTML. Con el tiempo lo deje algo olvidado, pero con GPT y circunstancias personales lo he retomado hace poco.

En un periodo corto de tiempo he aprendido varias cosas, sobretodo a utilizar mi filosofia de todo a la programación y al desarrollo de aplicaciones. Poco a poco , según iba preguntando dudas a GPT note que básicamente pasaban a ser puestas en común. De preguntar como se hace esto, pase a quiero hacer esto y tengo esas maneras, ¿ Cual es mejor?. Así me di cuenta de que gracias a la IA aprendí en tan solo unas semanas:


- Programación Orientada a Objetos
- Manejo de Bases de datos MySQL, MsAccess, SQLQuery3
- Diseño de clases con sus constructores y destructores
- Manejo de archivos XML
- Manejo de archivos Json
- Conocimientos de API Rest con Horse y DataSnap
- Creación dinámica de componentes
- Búsqueda dinámica de componentes
- Enciptación Md5 y SHA de contraseñas
- MVC, Diseño Multicapa
- Diseño, Carga y Descarga de BPL
- Diseño y uso de DLL
- Uso de VCL y Fire Monkey
- Uso de TADO y Actualmente FireDAC
- Manejo de MySQL: Triggers y Funciones, consultas con JOIN
- Manejo de archivos INI
- Manejo de Marshal y UnMarShal (Aun estoy con ello)
- Manejo de archivos
- Uso avanzado de TStringGrid, TComboBox, TListBox...
- Instalación de controles
- y mucho mas...

Basicamente uso la IA GPT junto con videos y turiales en PDF para aprender y mejorar en mis conocimientos. Tengo varios proyectos colaborativos en mente uno de ellos es la creación de una librería DLL para el sistema Veri Factu, que por problemas de homologación no puede ser Open Source, pero si de libre uso. Además sera compilada en C#, Delphi y C++. Y un Juego Educativo en colaboración con un colegio (este sera Open Source). Quiero demostrar que el conocimiento humano colaborando con la IA puede acelerar mucho en el campo del desarrollo de aplicaciones y sobretodo que no hay que tener miedo a pedir ayuda, ya sea en foros, a una IA o alguna persona o utilizando otros medios.

Explicación :


Usuarios	
Nombre del campo	Tipo de datos
 Id	Autonumérico
IdUsuario	Texto
Contraseña	Texto
Activo	Sí/No
Rol	Texto

Esta es la estructura de nuestra base de datos MSAccess. Es bastante sencilla.

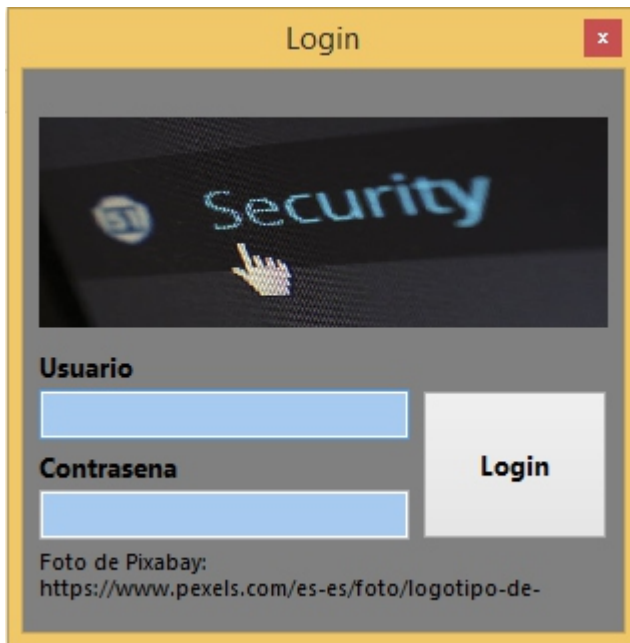
Usuarios					
Id	IdUsuario	Contraseña	Activo	Rol	Agregar nuevo campo
 admin	admin	1234	<input checked="" type="checkbox"/>	roAdmin	
* (Nuevo)			<input type="checkbox"/>		

Debemos guardar el archivo de MSAccess en la carpeta donde se encuentra el ejecutable de nuestro proyecto.

Disco local (C:) > githubDelphi > tutorial NCAPAS > Win32 > Debug

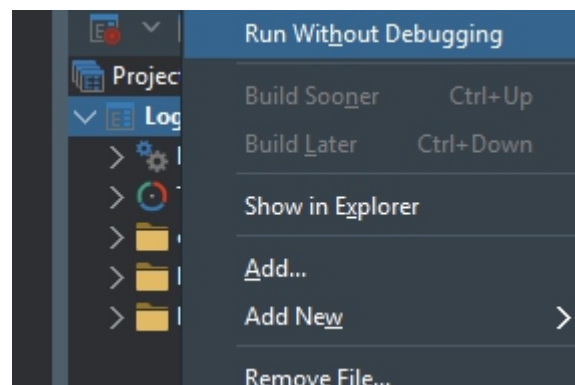
- classRoles.dcu
- classUsuario.dcu
- dbejemplo.mdb
- FormApp.dcu
- FormLogin.dcu
- logica_login.dcu
-  LoginNcapas.exe

Explicación :

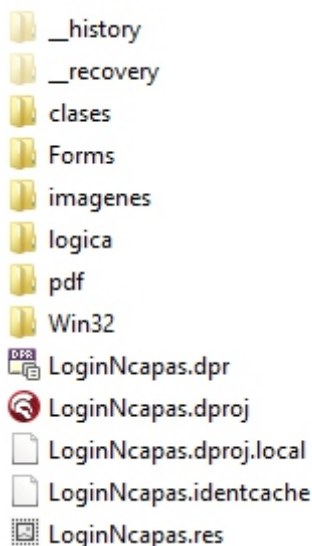


Lo primero es diseñar nuestro formulario, para ello utilicé una imagen con derechos de atribución. El formulario lleva componentes comunes en Delphi. A pesar de que utilizo la versión 11, podéis encontrarlos en versiones anteriores y si no están se pueden sustituir por TEdit+TLabel, si, se trata de TLabel+TEdit. En los archivos del proyecto podéis ver como esta estructurada y los valores de cada control utilizado en el Form.

Lo siguiente es Crear la clase Tusuario, para ello en nuestro proyecto pulsamos botón derecho y Add New->Unit. Damos a guardar, elegimos la carpeta del proyecto y creamos una carpeta Clases.



Nuestro proyecto debería de quedarnos así:



Clases : Contiene las unidades de las clases.

Forms : Contiene las unidades de los formularios.

Logica: Contiene la logica de cada Form o Cada Clase.

Pdf: Contien este documento

Imagenes: Contiene las imagenes utilizadas

El resto son los creados por RAD Studio.

Explicación :

La clase Tusuario:

Nuestro Usuario tiene

- IdUsuario : es la que cada usuario utiliza para realizar el login.
- Contraseña: es la contraseña que tiene cada usuario .
- Rol : Indica que nivel de acceso tiene cada usuario (esto lo veremos en la versión 2 y 3)
- Activo: Nos indica si el usuario está o no activado.

```
1  unit classUsuario;
2
3  interface
4  uses system.SysUtils,
5        system.Classes;
6
7  Type Tusuario = Class
8      Private
9          FIDusuario:string;
10         Fcontrasena:string;
11         Factivo:boolean;
12         Frol:string;
13     Public
14         property IdUsuario:string read FIDusuario write FIDusuario;
15         property Contraseña:String read FContrasena write Fcontrasena;
16         property Activo:boolean read Factivo write Factivo;
17         property Rol:string read Frol write Frol;
18         Constructor Create;
19         Destructor Destroy;Override;
20         Procedure Free;
21     End;
22
23 implementation
24
25 Procedure Tusuario.Free;
26 Begin
27     if self<>nil then Destroy;
28 End;
29
30 Destructor Tusuario.Destroy;
31 Begin
32     inherited Destroy;
33 End;
34 Constructor Tusuario.Create;
35 Begin
36     Idusuario:='';
37     Contraseña:='';
38     Activo:=true;
39     Rol:='roUsuario';
40 End;
41 end.
```

Aqui podemos ver la clase en su totalidad, es bastante sencilla de comprender, Tusuario tiene 4 propiedades relacionadas con los datos que hemos visto mas arriba. Para acceder a cada propiedad se le asigna una Propiedad , esto hace que podamos acceder a cada dato del usuario en tiempo de ejecución creandolo, modificandolo y eliminandolo. Para poder Utilizar nuestra clase necesitamos 3 cosas:

- Constructor de la clase
- Destructor de la clase
- Método Free para poder liberar nuestra Clase de la memoria.

Como se puede observar el rol que asignamos por directo es roUsuario, por seguridad es mejor jno asignar el usuario roAdmin o adminitrador.

Explicación :

Ahora debemos crear la logica para que podamos realizar nuestro sistema de Login.

```
1  unit logica_login;
2
3  interface
4  Uses system.SysUtils,
5        vcl.Dialogs,
6        data.Win.ADODB;
7      Function Login(IdUsuario:string;Contrasena:string):boolean;
8  Const CadenaConexion=
9      'Provider=Microsoft.Jet.OLEDB.4.0; '
10     +'Data Source=dbejemplo.mdb;';
11  implementation
12  uses classUsuario;
13
14  // esta version 1.0 no tiene encriptacion de contraseña y unicamente trabaja con MSAccess
15  Function Login(IdUsuario:string;Contrasena:string):boolean;
16  Var cSQL:TADOQuery;
17      con:TADOConnection;
18      x:integer;
19  var Usuario:Tusuario;
20  Begin
21      result:=false;
22      x:=0;
23      try
24          Usuario:=Tusuario.Create;
25          usuario.IdUsuario:=Idusuario;
26          Usuario.Contrasena:=Contrasena;
27          usuario.Activo:=false;
28          usuario.Rol:='roUsuario';
29      except
30          on e:exception do showmessage('Error :: '+e.Message);
31      end;
32      // el usuario es el que efectua el Login
33      try
34          con:=TADOConnection.Create(nil);
35          con.Connected:=false;
36          con.ConnectionString:=CadenaConexion;
37          cSQL:=TADOQuery.Create(nil);
38          cSQL.Close;
39          cSQL.Connection:=con;
40          cSQL.SQL.Clear;
41          cSQL.SQL.Text:='SELECT IdUsuario,Contrasena,Activo,Rol
42          'WHERE IdUsuario =' +QuotedStr(usuario.IdUsuario)
43          +'AND Contrasena=' +QuotedStr(usuario.Contrasena)+
44          'AND Activo=true';
45          con.Connected:=true;
46          cSQL.Open;
47          if cSQL.Eof then ShowMessage('Usuario Y/O contraseña incorrectos')
48          else
49              Begin
50                  ShowMessage('Bienvenido '+Usuario.IdUsuario);
51                  usuario.Rol:=cSQL.FieldByName('Rol').AsString;
52                  usuario.Activo:=cSQL.FieldByName('Activo').AsBoolean;
53                  result:=true;
54              End;
55          finally
56              cSQL.Free;
57              con.Free;
58              usuario.Free;
59          end;
60      end;
61  End;
62  end.
```

Diseñamos nuestra función de login con los datos que va a necesitas y lo que va a devolver. Tambien creamos una constante para la conexion a nuestro base de datos.

Lo primero es inicializar nuestras variables. Si queremos evitar alertas a la hora de compilar es necesario hacer esto. Tambien creamos nuestro usuario utilizando la clase. Además, podemos ver que añadimos la conexión y las consultas SQL como tipo var. Utilizamos try except para evitar errores al crear nuestro usuario utilizando su clase.

Aqui creamos nuestra conexión TADOConnection y nuestra consulta SQL TADOQuery. Ademas añadimos la consulta SQL que va a llevar nuestra Query.

Lo siguiente es ejecutar nuestra consulta. Para ello utilizamos TADOQuery.Open, ya que devuelve datos. Si modificamos, añadimos o eliminamos datos, se utiliza TADOQuery.ExecSQL, ya que no devuelve datos. Comprobamos que los datos sean los correctos y si son así, le asignamos el rol que tienen en la base de datos y su estado de activación(Esto nos sera util en la version 2 y 3).

Explicación :

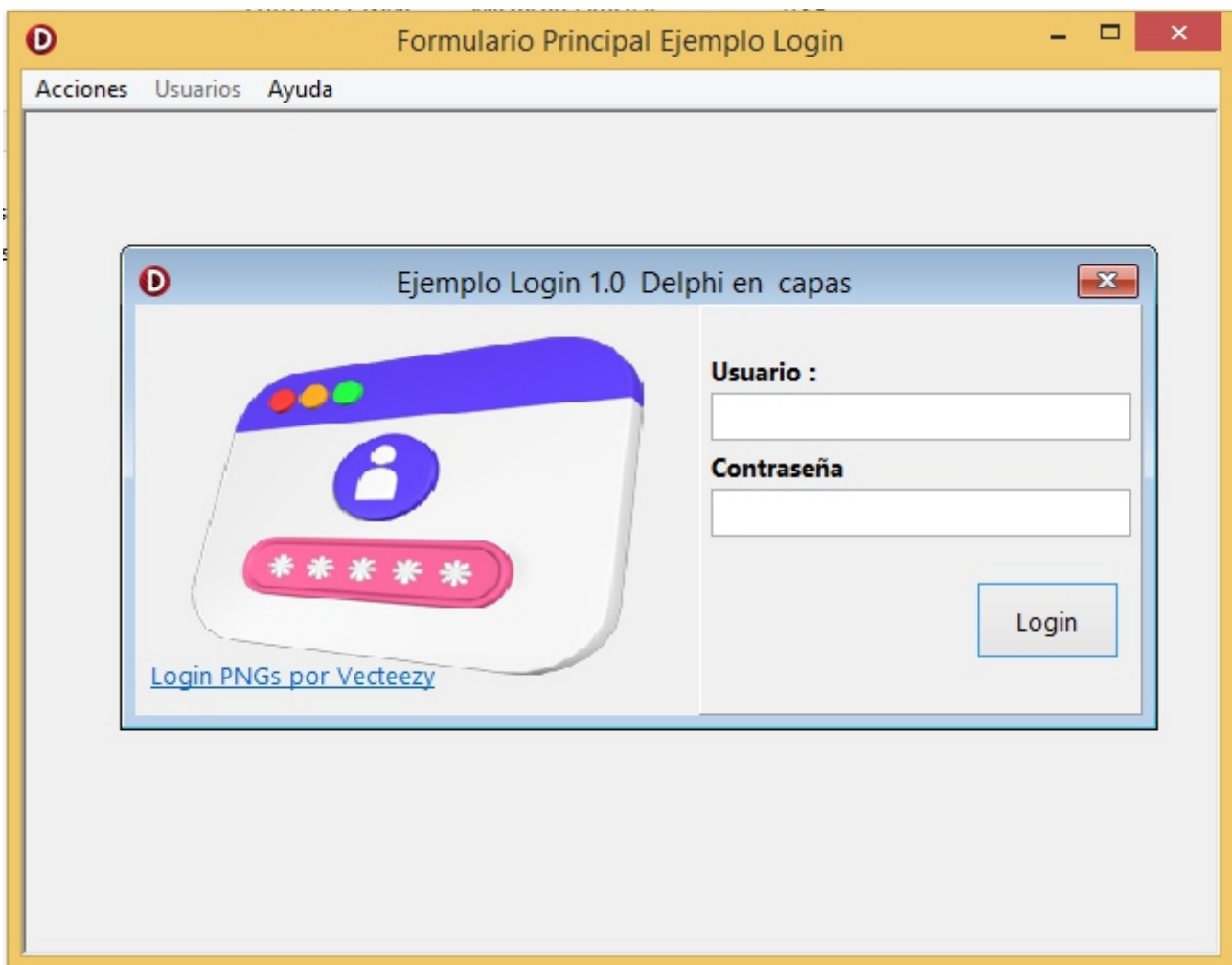
```
1  unit FormLogin;
2
3  interface
4
5  uses
6      Winapi.Windows,
7      Winapi.Messages,
8      System.SysUtils,
9      System.Variants,
10     System.Classes,
11     Vcl.Graphics,
12     Vcl.Controls,
13     Vcl.Forms,
14     Vcl.Dialogs,
15     Vcl.StdCtrls,
16     Vcl.ExtCtrls,
17     Vcl.Mask,
18     logica_login,
19     Formapp,
20     Vcl.Imaging.jpeg;
21
22 type
23     TForm_Login = class(TForm)
24         Login_Contrasena: TLabeledEdit;
25         Login_Usuario: TLabeledEdit;
26         login_imagen: TImage;
27         btn_Login: TButton;
28         atribucion_imagen: TLabel;
29         procedure btn_LoginClick(Sender: TObject);
30     private
31         { Private declarations }
32     public
33         { Public declarations }
34     end;
35
36 var
37     Form_Login: TForm_Login;
38
39 implementation
40     {$R *.dfm}
41
42     procedure TForm_Login.btn_LoginClick(Sender: TObject);
43     var App:TformAplicacion;
44     begin
45         if Login(login_usuario.Text,login_contrasena.Text) = true
46             then
47                 begin
48                     App := TformAplicacion.create(application);
49                     App.Caption:= App.Caption + login_usuario.Text;
50                     App.showmodal();
51
52                 end;
53     end;
54
55 end.
```

Podemos observar que en el form apenas tenemos que teclear código, ya que todo esta en la clase y la lógica del login. Simplemente usamos nuestra función y si nos da login true, abrimos el siguiente Form.

Explicación :

```
1  unit FormApp;
2
3  interface
4
5  uses
6      Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.
        Graphics,
7      Vcl.Controls, Vcl.Forms, Vcl.Dialogs;
8
9  type
10     TFormAplicacion = class(TForm)
11     private
12         { Private declarations }
13     public
14         { Public declarations }
15     end;
16
17  var
18     FormAplicacion: TFormAplicacion;
19
20  implementation
21
22     {$R *.dfm}
23
24  end.
25
```









En este Form realizaremos un CRUD completo, pero configurable. Podremos elegir base de datos Local con MSAccess o Servidor con MySQL 9 (En mi caso). Tendremos en cuenta los roles : solo el usuario admin puede modificar roles y activar o desactivar usuarios, cada usuario puede modificar su contraseña y el usuario admin puede moficar todas las contraseñas. Activaremos o desactivaremos menus , botones etc... atendiendo a los roles.




Parte 2: CRUD con MSAccess y MySQL

Hay que añadir al Search Path de nuestro proyecto las carpetas Forms, Clases y Logica. Para ello pulsamos en el menu proyecto->opciones->Compilador Delphi->Search Path: a continuación buscamos las carpetas y las seleccionamos. Posteriormente pulsamos ADD para añadirlas al path. Una vez finalizado pulsamos SAVE y listo.


Contenido de FORMS:

-  uFormCambioContrasena.dfm
-  uFormCambioContrasena.pas
-  uFormGestionUsuarios.dfm
-  uFormGestionUsuarios.pas
-  uFormLogin.dfm
-  uFormLogin.pas
-  uformprincipal.dfm
-  uformprincipal.pas

Contenido de Logica:

-  logica.pas

Contenido de clases

-  uClassUsuario.pas

Clase usuario

```
interface
uses
  System.SysUtils,
  System.Classes;
Type IUserio = interface
  ['{BBEFDB7F-CE14-4589-9FDB-7B52E3F2A922}']
end;
Type TUsuario = Class(TInterfacedObject,IUsuario)
  Private
    FIdUsuario:string;
    FContrasena:String;
    FActivo:boolean;
    FRol:integer;
  Public
    Property IdUsuario:String read FIdUsuario write FIdUsuario;
    Property Contrasena:String read FContrasena write FContrasena;
    Property Activo:boolean read FActivo write FActivo;
    Property Rol:integer read FRol write FRol;
    Constructor Create;
    Destructor Destroy;
    Procedure Free;
End;
```

Aquí definimos el interface al que pertenece la clase, con control+mayúsculas+G generamos el GUID de la interface. Después definimos las propiedades de la clase, así como sus métodos, constructores y destructores. Debemos tener en cuenta que Destroy no se usa directamente, para usarlo creamos un procedimiento Free. También un error muy común es no declarar el destructor como Override :

```
Destructor Destroy;Override;
```

```
• |
• | Procedure TUsuario.Free;
• | Begin
• |   if self<>nil then destroy;
• | End;
• | Destructor TUsuario.Destroy;
• | Begin
• |   inherited Destroy;
40 | End;
• | Constructor TUsuario.create;
• | Begin
• |   idUsuario:='';
• |   contrasena:='';
• |   activo:=true;
• |   rol:=1; // no crear el usuario 0 o admin por defecto
• | End;
```

unidad logica

```
function Loginv2(IdUsuario: string; contrasena: string): boolean;
Function NuevoUsuario(IdUsuario:String;Contrasena:string;Rol:integer;activo:boolean):boolean;
Function ModificarUsuario(IdUsuario:String;Contrasena:string;Rol:integer;activo:boolean):boolean;
Function CambioContrasena(IDUsuario:String;ContrasenaActual:string;ContrasenaNueva:string):boolean;
Function EliminarUsuario(IdUsuario:string):boolean;
Function Encriptar(const APassword: string): string; //SHA256
Function BuscarUsuario(IdUsuario:string):TdataSource;
Procedure ReiniciarCamposGrupo(grupo:TGroupBox); //groupbox
Function VerificarCamposGrupo(grupo:TGroupBox):integer ;//groupbox
procedure ConfigurarProveedorConexion(BD: string);
```

La unidad logica contiene todo el CRUD de nuestro Login y la gestión de usuarios.
Para ver los detalles del crud se puede ver el contenido de esta unidad en el Editor de RadStudio.

Formularios

