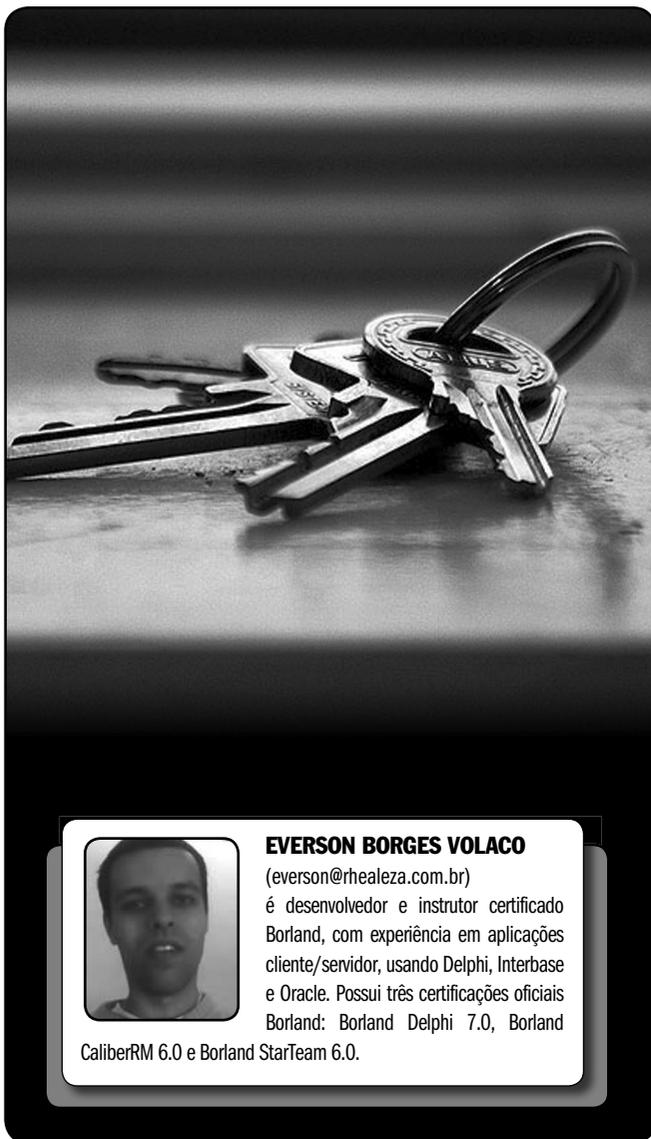


Controle de Acesso

Realizando o controle de acesso de usuários e grupos da aplicação através de menu dinâmico



EVERSON BORGES VOLACO

(everson@rhealeza.com.br)

é desenvolvedor e instrutor certificado Borland, com experiência em aplicações cliente/servidor, usando Delphi, Interbase e Oracle. Possui três certificações oficiais Borland: Borland Delphi 7.0, Borland CaliberRM 6.0 e Borland StarTeam 6.0.

Neste artigo veremos como realizar o controle de acesso em uma aplicação Delphi, utilizando menu dinâmico. A idéia é definir os privilégios de acesso dos usuários e grupos do sistema no banco de dados e habilitar as opções do menu principal (*MainMenu*) e dos botões das telas de cadastro, de acordo com esses privilégios.

Nota: Para o exemplo deste artigo utilizei o Delphi 7 com o InterBase 7.5.1, porém você pode utilizar qualquer outra versão do Delphi e do InterBase ou Firebird.

Exemplo

Para o exemplo deste artigo criaremos uma aplicação VCL (Win32) contendo inúmeras opções de menu e telas de cadastro e movimentação. Para não sair do foco do assunto deste artigo, controle de acesso, não implementaremos todas as telas desse sistema, porém as funcionalidades que serão necessárias para fazer o controle de acesso dos usuários e grupos, serão mostradas.

Serão utilizados os privilégios de *Consulta*, *Inserção*, *Alteração* e *Exclusão*, os quais poderão ser definidos tanto para um grupo quanto para um usuário do sistema. Um usuário poderá pertencer a mais de um grupo, onde, caso o mesmo possua privilégios de acesso diferentes em cada grupo, os privilégios serão concatenados.

Por exemplo, se definirmos que o usuário *Everson* pertence aos grupos *Desenvolvedores* e *Analistas* e no grupo *Desenvolvedores* o mesmo tenha apenas permissão de *Inserção* na tela de clientes, porém, no grupo *Analistas* tenha permissões de *Alteração* e *Exclusão*, então o *Everson* será capaz de *Inserir*, *Alterar* e *Excluir* os clientes.

O controle de acesso poderá ser definido ainda ao nível

de usuário sem a necessidade de vinculá-lo a um ou mais grupos. No entanto, caso existam permissões de acesso definidas diretamente no usuário e esse pertença a um grupo que também possui permissões definidas, as mesmas serão concatenadas da mesma forma que ocorre entre os grupos.

Nota: Por questões de espaço, mostraremos neste artigo apenas os códigos relacionados ao controle de acesso da aplicação. Você pode ter acesso ao código-fonte completo do exemplo através do link para download.

Definindo o banco de dados

Começaremos o exemplo, definindo as tabelas que serão responsáveis em armazenar as informações referentes ao controle de acesso do sistema. Abra a sua ferramenta de gerenciamento de banco de dados InterBase ou Firebird, como por exemplo o IBConsole ou IBExpert e crie um banco de dados com o nome de "ACESSO.IB". Criaremos seis tabelas no banco de dados, cuja descrição das mesmas consta na **Tabela 1**.

Nome da Tabela	Descrição
MENUS	Tabela contendo todas as opções disponíveis no menu principal da aplicação.
GRUPOS	Tabela contendo todos os grupos cadastrados no sistema.
USUARIOS	Tabela com todos os usuários do sistema.
GRUPOS_USUARIOS	Tabela de vínculo entre os usuários e grupos do sistema.
PERMISSOES_GRUPOS	Tabela contendo as permissões de acesso de cada item de menu e tela para os grupos do sistema.
PERMISSOES_USUARIOS	Tabela contendo as permissões de acesso de cada item de menu e tela para os usuários do sistema.

Tabela 1. Descrição das tabelas do banco

Na **Listagem 1** temos as instruções DDL para a criação das tabelas.

Listagem 1. Código para criação das tabelas

```
CREATE TABLE MENUS (
  MEN_CODIGO INTEGER NOT NULL,
  MEN_NOME VARCHAR(25) NOT NULL,
  MEN_CODIGOPAI INTEGER,
  MEN_EXISTE BOOLEAN NOT NULL,
  CONSTRAINT PK_MENUS PRIMARY KEY (MEN_CODIGO),
  CONSTRAINT FK_MENUS FOREIGN KEY (MEN_CODIGOPAI)
  REFERENCES MENUS (MEN_CODIGO)
  ON UPDATE CASCADE ON DELETE CASCADE);

CREATE TABLE GRUPOS (
  GRU_CODIGO INTEGER NOT NULL,
  GRU_DESCRICAO VARCHAR(30) NOT NULL,
  CONSTRAINT PK_GRUPOS PRIMARY KEY (GRU_CODIGO));

CREATE TABLE USUARIOS (
```

```
  USU_CODIGO INTEGER NOT NULL,
  USU_NOMECompleto VARCHAR(70) NOT NULL,
  USU_LOGIN VARCHAR(15) NOT NULL,
  USU_SENHA VARCHAR(15) NOT NULL,
  USU_ADMIN BOOLEAN DEFAULT FALSE,
  CONSTRAINT PK_USUARIOS PRIMARY KEY (USU_CODIGO));
```

```
CREATE TABLE GRUPOS_USUARIOS (
  GRU_CODIGO INTEGER NOT NULL,
  USU_CODIGO INTEGER NOT NULL,
  CONSTRAINT PK_GRUPOS_USUARIOS
  PRIMARY KEY (GRU_CODIGO, USU_CODIGO),
  CONSTRAINT FK_GU_GRUPOS FOREIGN KEY (GRU_CODIGO)
  REFERENCES GRUPOS (GRU_CODIGO)
  ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT FK_GU_USUARIOS FOREIGN KEY (USU_CODIGO)
  REFERENCES USUARIOS (USU_CODIGO)
  ON UPDATE CASCADE ON DELETE CASCADE);
```

```
CREATE TABLE PERMISSOES_GRUPOS (
  GRU_CODIGO INTEGER NOT NULL,
  MEN_CODIGO INTEGER NOT NULL,
  PG_NOVO BOOLEAN DEFAULT FALSE,
  PG_ALTERAR BOOLEAN DEFAULT FALSE,
  PG_APAGAR BOOLEAN DEFAULT FALSE,
  PG_CONSULTAR BOOLEAN DEFAULT FALSE,
  CONSTRAINT PK_PERMISSOES_GRUPOS
  PRIMARY KEY (GRU_CODIGO, MEN_CODIGO),
  CONSTRAINT FK_PG_USUARIOS FOREIGN KEY (GRU_CODIGO)
  REFERENCES GRUPOS (GRU_CODIGO)
  ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT FK_PG_MENUS FOREIGN KEY (MEN_CODIGO)
  REFERENCES MENUS (MEN_CODIGO)
  ON UPDATE CASCADE ON DELETE CASCADE);
```

```
CREATE TABLE PERMISSOES_USUARIOS (
  USU_CODIGO INTEGER NOT NULL,
  MEN_CODIGO INTEGER NOT NULL,
  PU_NOVO BOOLEAN DEFAULT FALSE,
  PU_ALTERAR BOOLEAN DEFAULT FALSE,
  PU_APAGAR BOOLEAN DEFAULT FALSE,
  PU_CONSULTAR BOOLEAN DEFAULT FALSE,
  CONSTRAINT PK_PERMISSOES_USUARIOS
  PRIMARY KEY (USU_CODIGO, MEN_CODIGO),
  CONSTRAINT FK_PU_USUARIOS FOREIGN KEY (USU_CODIGO)
  REFERENCES USUARIOS (USU_CODIGO)
  ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT FK_PU_MENUS FOREIGN KEY (MEN_CODIGO)
  REFERENCES MENUS (MEN_CODIGO)
  ON UPDATE CASCADE ON DELETE CASCADE);
```

Nota: O Firebird não suporta campos do tipo *Boolean*. Você pode contornar, criando um campo *Char* de tamanho 1. Não esqueça de fazer as respectivas alterações no código do artigo

Para as tabelas *Grupos* e *Usuarios* vamos definir *Generators* e *Triggers* para realizar o auto incremento do valor da chave primária dessas tabelas. Na **Listagem 2** temos as instruções SQL para criação desses objetos no banco de dados.

Listagem 2. Generators e Triggers das tabelas

```
CREATE GENERATOR GEN_GRUPOS;

CREATE GENERATOR GEN_USUARIOS;

CREATE TRIGGER TRG_INC_GRUPOS
FOR GRUPOS BEFORE INSERT POSITION 0
AS
BEGIN
  IF (NEW.GRU_CODIGO IS NULL) THEN
    NEW.GRU_CODIGO = GEN_ID(GEN_GRUPOS, 1);
END

CREATE TRIGGER TRG_INC_USUARIOS
FOR USUARIOS BEFORE INSERT POSITION 0
AS
BEGIN
  IF (NEW.USU_CODIGO IS NULL) THEN
    NEW.USU_CODIGO = GEN_ID(GEN_USUARIOS, 1);
END
```

Para finalizar a criação dos objetos do banco de dados, vamos criar duas *Stored Procedures* que serão responsáveis em verificar os direitos de acesso do usuário logado no sistema. Na **Listagem 3** temos o código para criação da *Stored Procedure* PROC_VERIFICA_ACESSO_MENU, que receberá o código do usuário logado e o código do menu corrente como parâmetro de entrada e retornará o direito de acesso do usuário sobre o menu em questão.

Listagem 3. Criação da SP para retornar o direito de acesso do usuário

```
CREATE PROCEDURE PROC_VERIFICA_ACESSO_MENU (
  CODUSUARIO INTEGER, CODMENU INTEGER)
RETURNS (CONSULTAR CHAR(1))
AS
DECLARE VARIABLE GU_CONSULTAR BOOLEAN;
BEGIN
  CONSULTAR = 'N';
  FOR SELECT PU_CONSULTAR FROM PERMISSOES_USUARIOS
  WHERE USU_CODIGO = :CODUSUARIO AND
  MEN_CODIGO = :CODMENU
  UNION
  SELECT PG_CONSULTAR FROM PERMISSOES_GRUPOS
  WHERE GRU_CODIGO IN (
  SELECT GRU_CODIGO FROM GRUPOS_USUARIOS
  WHERE USU_CODIGO = :CODUSUARIO) AND
  MEN_CODIGO = :CODMENU
  INTO :GU_CONSULTAR
  DO
  BEGIN
    IF (:GU_CONSULTAR = TRUE) THEN
      BEGIN
        CONSULTAR = 'S';
        EXIT;
      END
    END
  END
END
```

Para verificar os direitos de acesso do usuário a uma tela do sistema utilizaremos também uma *Stored Procedure*. A PROC_VERIFICA_DIREITOS_ACESSO descrita na **Listagem 4**, que retornará os direitos de inclusão, alteração e exclusão do usuário sobre a tela verificada.

Listagem 4. Procedure que retorna os direitos do usuário

```
CREATE PROCEDURE PROC_VERIFICA_DIREITOS_ACESSO (
  CODUSUARIO INTEGER, CODMENU INTEGER)
RETURNS (NOVO CHAR(1), ALTERAR CHAR(1),
  APAGAR CHAR(1))
AS
DECLARE VARIABLE ADM BOOLEAN;
DECLARE VARIABLE GU_NOVO BOOLEAN;
DECLARE VARIABLE GU_ALTERAR BOOLEAN;
DECLARE VARIABLE GU_APAGAR BOOLEAN;
BEGIN
  NOVO = 'N'; ALTERAR = 'N'; APAGAR = 'N';
  SELECT USU_ADMIN FROM USUARIOS
  WHERE USU_CODIGO = :CODUSUARIO
  INTO :ADM;
  IF (:ADM = TRUE) THEN
    BEGIN
      NOVO = 'S';
      ALTERAR = 'S';
      APAGAR = 'S';
      EXIT;
    END
  END

  FOR SELECT PU_NOVO, PU_ALTERAR, PU_APAGAR
  FROM PERMISSOES_USUARIOS
  WHERE USU_CODIGO = :CODUSUARIO AND
  MEN_CODIGO = :CODMENU
  UNION
  SELECT PG_NOVO, PG_ALTERAR, PG_APAGAR
  FROM PERMISSOES_GRUPOS
  WHERE GRU_CODIGO IN (SELECT GRU_CODIGO
  FROM GRUPOS_USUARIOS
  WHERE USU_CODIGO = :CODUSUARIO) AND
  MEN_CODIGO = :CODMENU
  INTO :GU_NOVO, :GU_ALTERAR, :GU_APAGAR
  DO
  BEGIN
```

```
IF (:GU_NOVO = TRUE) THEN
  NOVO = 'S';
IF (:GU_ALTERAR = TRUE) THEN
  ALTERAR = 'S';
IF (:GU_APAGAR = TRUE) THEN
  APAGAR = 'S';
END
END
```

Construindo a aplicação Delphi

Inicie uma nova aplicação Delphi (*File>New>Application*) e altere o nome do formulário para "FrmPrincipal". Salve sua *unit* como "untFrmPrincipal.pas". Para o arquivo de projeto salve-o como "ControleAcesso.dpr".

Crie um Data Module (*File>New>Data Module*), altere seu nome para "DMPrincipal" e salve sua *unit* como "untDMPrincipal.pas". Adicione alguns componentes não-visuais de acesso a dados da paleta *dbExpress* (**Figura 1**).

Altere o nome dos componentes como mostrado na figura e configure o *SQLConnection* para acessar o banco *ACESSO.IB*. Selecione os quatro componentes do Data Module (com exceção do *conAcesso*) e aponte a propriedade de *SQLConnection* para *conAcesso*.

No *spAcessoMenu*, altere a propriedade *StoredProcName* para PROC_VERIFICA_ACESSO_MENU. Para o *SPDireitoAcesso* aponte a propriedade *StoredProcName* para PROC_VERIFICA_DIREITOS_ACESSO.

Nota: Manteremos a propriedade *CommandText* dos componentes *datasetAux* e *datasetAux2* vazia, pois utilizaremos os mesmos por diversas funções do sistema, que enviarão instruções SQL dinâmicas.

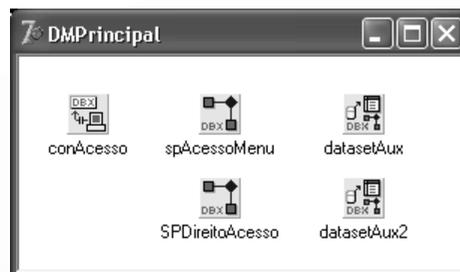


Figura 1. Componentes para acesso aos objetos do banco de dados



Voltando ao *FrmPrincipal* adicione um *MainMenu*, altere seu nome para “*mmPrincipal*” e através de um duplo clique inclua os seguintes itens de menu:

- Cadastros
 - o Clientes
 - o Fornecedores
 - o Auxiliares
 - ⊙ Cidades
 - ⊙ Estados
 - ⊙ Países
- Movimentações
 - o Notas Fiscais
 - ⊙ Entrada
 - ⊙ Saída
 - o Ordem de Serviço
- Segurança
 - o Grupos
 - o Usuários
 - o Permissões de Grupo
 - o Permissões de Usuário
- Outros
 - o Calculadora
 - o MS Excel
 - o MS Word
 - o Caixa Sobre

Utilizaremos a tabela *MENUS* do banco de dados para armazenar todos os itens de menu cadastrados no *MainMenu*. Esse cadastramento ocorrerá no momento em que o sistema for executado. A idéia é tornar esse cadastro o mais dinâmico possível, isso é, caso seja incluído um novo item o mesmo será cadastrado automaticamente na próxima vez que o sistema for executado.

O mesmo ocorrerá se um item for excluído do menu, nesse caso o item será removido da tabela na próxima execução do sistema. Antes de implementarmos essa funcionalidade altere a ordem de criação do *DMPPrincipal* para que o mesmo seja criado antes do *FrmPrincipal*.

Para isso configure a seção *Auto-create forms* disponível na janela *Project Options>Forms*. No *FrmPrincipal* digite no evento *OnCreate* o código da **Listagem 5**.

No código da listagem anterior, utilizamos o *datasetAux* para atualizar a coluna *MEN_EXISTE* da tabela *MENUS* com o valor *False*. Na seqüência fazemos a chamada ao método *verificaOpcoesMenu*, que será detalhado mais adiante e é responsável em atualizar a tabela *MENUS* de acordo com os itens de menu disponíveis no *mmPrincipal*.

No final do evento realizamos a exclusão dos registros da tabela *MENUS*, onde a coluna *MEN_EXISTE* continua com o valor *False*. Veja na **Listagem 6** a implementação do método *verificaOpcoesMenu*.

Listagem 5. Código do evento OnCreate do FrmPrincipal

```
procedure TFrmPrincipal.FormCreate(Sender: TObject);
var
  i: Integer;
begin
  { Declare untDMPPrincipal no uses }
  with DMPPrincipal.datasetAux do
  begin
    Close;
    CommandText :=
      'UPDATE MENUS SET MEN_EXISTE = False';
    ExecSQL(True);
  end;
  for i := 0 to Pred(mmPrincipal.Items.Count) do
    verificaOpcoesMenu(mmPrincipal.Items[i]);
  with DMPPrincipal.datasetAux do
  begin
    Close;
    CommandText :=
      'DELETE FROM MENUS WHERE MEN_EXISTE = False';
    ExecSQL(True);
  end;
end;
```

Listagem 6. Método VerificaOpcoesMenu

```
procedure TFrmPrincipal.verificaOpcoesMenu(
  const menu: TMenuItem);
var
  i, seq, codMenu, codMenuPai: Integer;
begin
  with DMPPrincipal.datasetAux do
  begin
    Close;
    CommandText :=
      'SELECT MEN_CODIGO FROM MENUS WHERE MEN_NOME= ' +
      QuotedStr(Trim(RetirarCaracterAtalho(
        menu.Caption)));
    Open;
    if IsEmpty then
    begin
      seq := retornaProximoCodigoMenu;
      if Assigned(menu.Parent) and
        (menu.Parent.Caption <> '') then
      begin
        codMenuPai := retornarCodMenu(Trim(
          RetirarCaracterAtalho(menu.Parent.Caption)));
        Close;
        CommandText :=
          'INSERT INTO MENUS (MEN_CODIGO, MEN_NOME, ' +
          'MEN_CODIGOPAI, MEN_EXISTE) VALUES (' +
          IntToStr(seq) + ', ' +
          QuotedStr(Trim(RetirarCaracterAtalho(
            menu.Caption))) + ', ' + IntToStr(
            codMenuPai) + ', True)';
        end
      else
      begin
        Close;
        CommandText :=
          'INSERT INTO MENUS (MEN_CODIGO, MEN_NOME, ' +
          'MEN_EXISTE) VALUES (' + IntToStr(seq) + ', ' +
          QuotedStr(Trim(RetirarCaracterAtalho(
            menu.Caption))) + ', True)';
        end;
        ExecSQL(True);
      end
    else
    begin
      codMenu := Fields[0].AsInteger;
      Close;
      CommandText :=
        'UPDATE MENUS SET MEN_EXISTE = True ' +
        'WHERE MEN_CODIGO = ' + IntToStr(codMenu);
      ExecSQL(True);
    end;
  end;
  for i := 0 to Pred(menu.Count) do
    verificaOpcoesMenu(menu.Items[i]);
  end;
```

O método *verificaOpcoesMenu* recebe como parâmetro de entrada o item de menu corrente e realiza o cadastro do mesmo dentro da tabela *MENUS*. Como o item de menu em questão pode conter submenus, fazemos a chamada recursiva ao método.

Esse método utiliza ainda três métodos auxiliares para realizar as operações na tabela MENUS. Na **Listagem 7** temos essa implementação.

Listagem 7. Implementação dos métodos auxiliares

```

{ Utilizado para remover o "g" colocado automaticamente pelo Delphi em cada item de menu do MainMenu. }

function TFrmPrincipal.RetirarCaracterAtalho(
  const texto: string): string;
var
  i, tamanho : Integer;
begin
  tamanho := Length(texto);
  for i := 1 to tamanho do
    if texto[i] <> 'g' then
      Result := Result + texto[i];
  end;
end;

{ Utilizado para trazer próximo código válido para inserção de um novo registro na tabela MENUS. }

function TFrmPrincipal.
  retornaProximoCodigoMenu: Integer;
begin
  with DMPrincipal.datasetAux2 do
  begin
    Close;
    CommandText :=
      'SELECT MAX(MEN_CODIGO) AS MAIOR FROM MENUS';
    Open;
    Result := Fields[0].AsInteger + 1;
  end;
end;

{ Utilizado para retornar o código do menu passado como parâmetro de entrada. }

function TFrmPrincipal.retornarCodMenu(
  const textoMenu: string): Integer;
begin
  with DMPrincipal.datasetAux do
  begin
    Close;
    CommandText :=
      'SELECT MEN_CODIGO FROM MENUS WHERE MEN_NOME= ' +
      QuotedStr(Trim(textoMenu));
    Open;
    if not IsEmpty then
      Result := Fields[0].AsInteger
    else
      Result := -1;
  end;
end;
end;

```

Se rodarmos nossa aplicação, veremos que todos os itens de menu serão cadastrados automaticamente dentro da tabela MENUS. Crie formulários de cadastro simples para inclusão dos Grupos e Usuários do sistema. Esses cadastros serão utilizados apenas para a inserção, alteração e exclusão dos grupos e usuários que terão os direitos de acesso definidos.

Nota: Você pode inserir alguns grupos e usuários diretamente no banco de dados. No exemplo disponível para download implementei tais telas de cadastro, onde, inclusive na tela de cadastro de usuários mostra como fazer a vinculação do mesmo a um ou mais grupos cadastrados armazenando a informação na tabela GRUPOS_USUARIOS.

Veja um layout de exemplo dos cadastros na **Figura 2**.

O próximo passo é definir os direitos de acesso de cada grupo e/ou usuário através das tabelas PERMISSOES_



Figura 2. Cadastros de Grupos e Usuários do sistema

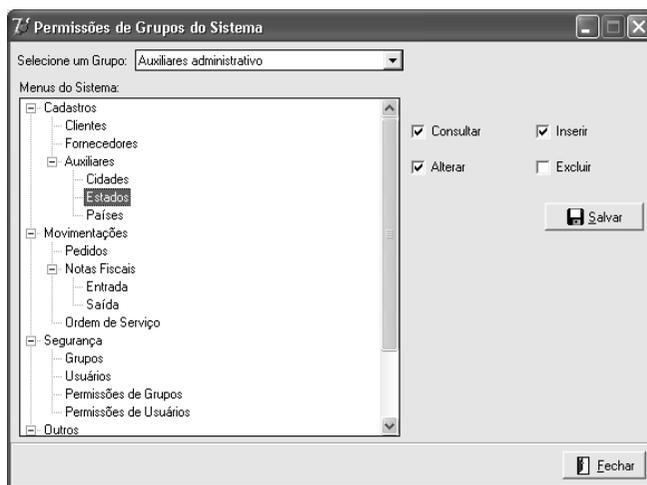


Figura 3. Exemplo de tela para definição das permissões de acesso aos grupos do sistema

GRUPOS e PERMISSOES_USUARIOS. Da mesma maneira você pode criar uma tela para cadastro dessas permissões de acesso, ou, se preferir inseri-las diretamente no banco de dados. Veja na **Figura 3** um exemplo de tela para cadastro dos direitos de acesso para os grupos do sistema.

Repare que no banco de dados definimos as chaves estrangeiras das tabelas de permissões com a cláusula DELETE CASCADE. Isso é necessário, pois caso algum item de menu for apagado do *mmPrincipal*, na próxima vez que o sistema for executado os registros relacionados a esse item nas tabelas de permissões serão apagados automaticamente.

Para podermos carregar os direitos de acesso do usuário logado no sistema referente aos itens de menu e telas, devemos declarar as seguintes variáveis na seção *public* do *FrmPrincipal*:

```

public
  codUsuario: Integer;
  codMenu: Integer;
  admin: Boolean;

```

Essas variáveis serão definidas e utilizadas em vários pontos do sistema, para que possamos controlar as permissões de acesso do usuário ativo. Vamos implementar agora a rotina que será responsável em chamar a *Stored Procedure* que fará a verificação dos direitos de acesso do

usuário logado nos itens de menu disponíveis no sistema. No *FrmPrincipal* digite o código da **Listagem 8** para o evento *OnShow* do formulário.

Listagem 8. Código do OnShow do formulário

```
procedure TFrmPrincipal.FormShow(Sender: TObject);
var
  i: Integer;
begin
  if admin then
    Exit;
  for i := 0 to Pred(mmPrincipal.Items.Count) do
    verificaAcessoSubMenu(mmPrincipal.Items[i]);
end;
```

No código da listagem anterior verificamos se a variável *admin* está definida como *True*, caso positivo, saímos do evento sem fazer a verificação de acesso aos menus do sistema, disponibilizando todos ao usuário, visto que o mesmo está definido no cadastro de usuários como Administrador.

Caso o usuário não seja Administrador realizamos um *loop* (*for..do*) nos itens de *mmPrincipal* passando cada item como parâmetro de entrada ao método *verificaAcessoSubMenu*. Na **Listagem 9** temos a implementação do método.

Listagem 9. Implementação do método verificaAcessoSubMenu

```
procedure TFrmPrincipal.verificaAcessoSubMenu(
  const menu: TMenuItem);
var
  i: Integer;
begin
  if not VerificarAcesso(retornarCodMenu(
    RetirarCaracterAtalho(menu.Caption))) then
    menu.Visible := False;
  for i := 0 to Pred(menu.Items.Count) do
    verificaAcessoSubMenu(menu.Items[i]);
end;
```

O *verificaAcessoSubMenu* por sua vez faz a chamada a um método auxiliar denominado *VerificarAcesso* e na seqüência realiza um *loop* em todos os sub-itens do menu corrente fazendo uma chamada recursiva a ele mesmo.

O método *verificarAcesso* recebe como parâmetro de entrada o código do menu corrente. Utilizamos os métodos auxiliares *RetirarCaracterAtalho* e *retornarCodMenu* para capturar o código do menu em questão. Veja na **Listagem 10** a implementação do *verificarAcesso*.

Listagem 10. Implementação do método verificarAcesso

```
function TFrmPrincipal.VerificarAcesso(
  const codMenu: Integer) : Boolean;
begin
  with DMPrincipal.spAcessoMenu do
  begin
    Close;
    ParamByName('CODUSUARIO').AsInteger := codUsuario;
    ParamByName('CODMENU').AsInteger := codMenu;
    ExecProc;
    if (ParamByName('CONSULTAR').AsString = 'S') or
      (codMenu = -1) then
      Result := True
    else
      Result := False;
  end;
end;
```

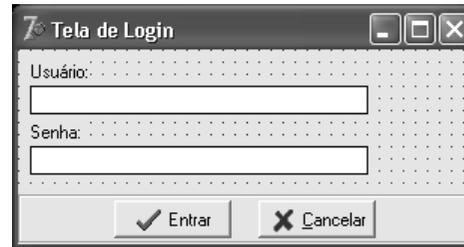


Figura 4. Tela de Login da aplicação de exemplo

O método é bastante simples, ele apenas faz a chamada a *PROC_VERIFICA_ACESSO_MENU* através do *spAcessoMenu* do *DMPrincipal* passando o código do usuário logado e o código do menu a ser verificado. Caso esse método retorne *False* alteramos a propriedade *Visible* do item de menu para *False* desabilitando assim o acesso ao mesmo pelo usuário.

Para que possamos testar os recursos de controle de acesso de nossa aplicação vamos criar uma tela de login. Crie um novo formulário e altere nome do mesmo para "FrmLogin". Salve sua *unit* como "untFrmLogin.pas". Adicione alguns componentes visuais e configure-os como mostra a **Figura 4**.

Declare as seguintes variáveis na seção *public* do formulário:

```
public
  codigoUsuario: Integer;
  nomeUsuario: string;
  administrador: Boolean;
```

Para o evento *OnClick* do botão *Entrar* digite o código da **Listagem 11**.





Listagem 11. Código para validar o usuário

```

procedure TFrmLogin.btnOKClick(Sender: TObject);
begin
  { Atenção no nome dos componentes Edits }
  if edtUsuario.Text = '' then
    begin
      MessageDlg('Informe o Usuário', mtInformation,
        [mbOk], 0);
      edtUsuario.SetFocus;
      Exit;
    end;
  if edtSenha.Text = '' then
    begin
      MessageDlg('Informe a Senha', mtInformation,
        [mbOk], 0);
      edtSenha.SetFocus;
      Exit;
    end;
  ValidarUsuario(AnsiUpperCase(edtUsuario.Text),
    edtSenha.Text);
end;

```

O código da listagem anterior faz a verificação do preenchimento dos campos da tela de login e realiza a chamada ao método *ValidarUsuario*, passando como parâmetro o usuário e senha informada. Veja na **Listagem 12** a implementação do método *ValidarUsuario*.

Listagem 12. Método ValidarUsuario

```

procedure TFrmLogin.ValidarUsuario(
  const usuario, senha: string);
begin
  { Declare em uses untDMPrincipal }
  with DMPrincipal.datasetAux do
    begin
      Close;
      CommandText :=
        'SELECT USU_CODIGO, USU_NOMECompleto, USU_ADMIN'+
        ' FROM USUARIOS ' +
        ' WHERE UPPER(USU_LOGIN) = '+ QuotedStr(usuario)+
        ' AND USU_SENHA = '+ QuotedStr(senha);
      Open;
      if not IsEmpty then
        begin
          codigoUsuario := Fields[0].AsInteger;
          nomeUsuario := Fields[1].AsString;
          administrador := Fields[2].AsBoolean;
          ModalResult := mrOk;
        end
      else
        begin
          MessageDlg('Usuário ou senha inválida!',
            mtInformation, [mbOk], 0);
          edtUsuario.SetFocus;
        end;
      end;
    end;
end;

```

Mais uma vez utilizamos o *datasetAux* do *DMPrincipal* para enviar uma instrução SQL pra fazer a validação do usuário e senha. Para chamar a tela de Login, altere o arquivo de projeto *ControleAcesso.dpr* (menu *Project>View Source*) como mostrado na **Listagem 13**.

Listagem 13. Código do projeto para abertura da tela de Login

```

program ControleAcesso;

uses
  Forms,
  Controls,
  SysUtils,
  untFrmPrincipal in 'untFrmPrincipal.pas'
  (FrmPrincipal),
  untDMPrincipal in 'untDMPrincipal.pas'
  (DMPrincipal: TDataModule),
  untFrmLogin in 'untFrmLogin.pas' (FrmLogin);

($R *.res)

begin
  Application.Initialize;
  Application.CreateForm(TDMPrincipal, DMPrincipal);
  FrmLogin := TFrmLogin.Create(nil);
  if FrmLogin.ShowModal = mrOk then
    begin
      Application.CreateForm(
        TFrmPrincipal, FrmPrincipal);
      FrmPrincipal.codUsuario := FrmLogin.codigoUsuario;
      FrmPrincipal.admin := FrmLogin.administrador;
      FreeAndNil(FrmLogin);
      Application.Run;
    end
  else
    Application.Terminate;
  end.

```

Nota: Para mais informações sobre a criação e utilização de telas de login acesse o artigo *Utilizando tela de login* disponível no Portal do Assinante (www.clubedelphi.net/portal).

Para testar a aplicação adicione um usuário no banco de dados, defina alguns direitos de acesso ao mesmo e rode a aplicação. Veja a aplicação de exemplo rodando na **Figura 5**. Para finalizar o exemplo vamos habilitar/desabilitar os

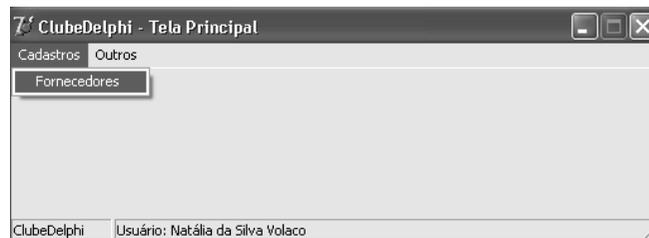


Figura 5. Aplicação de exemplo em execução mostrando apenas os menus disponíveis para o usuário logado

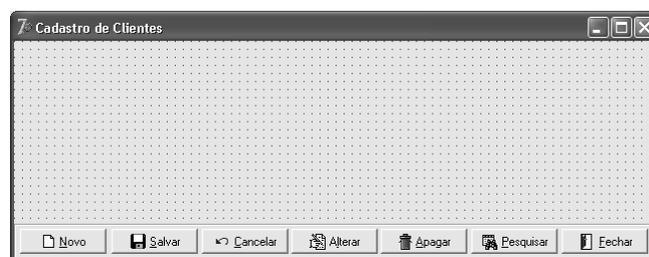


Figura 6. Formulário para cadastro de clientes

botões de uma tela de cadastro, obedecendo os direitos de acesso definidos no banco de dados. Crie um novo formulário e altere o nome do mesmo para "FrmCadClientes". Salve sua *unit* como "untFrmCadClientes.pas", adicione alguns botões ao formulário e configure-os como mostra a **Figura 6**.

Implementaremos nessa tela apenas as funcionalidades referentes ao controle de acesso aos botões. Declare as seguintes variáveis na seção *private* do formulário:

```
private
  novo, alterar, apagar: string;
```

Digite o código da **Listagem 14** no evento *OnCreate* do *FrmCadClientes*.

Listagem 14. Verificando os direitos de acesso do usuário

```
procedure TFrmCadClientes.FormCreate(Sender: TObject);
begin
  { Declare em uses untDMPrincipal e untFrmPrincipal }

  with DMPrincipal.SPDireitoAcesso do
  begin
    Close;
    ParamByName('CODUSUARIO').AsInteger :=
      FrmPrincipal.codUsuario;
    ParamByName('CODMENU').AsInteger :=
      FrmPrincipal.codMenu;
    ExecProc;
    novo := ParamByName('NOVO').AsString;
    alterar := ParamByName('ALTERAR').AsString;
    apagar := ParamByName('APAGAR').AsString;
  end;

  { Atenção ao nome dos componentes }
  btnNovo.Enabled := (novo = 'S');
  btnAlterar.Enabled := (alterar = 'S');
  btnApagar.Enabled := (apagar = 'S');
```

end;

No código da listagem anterior fazemos a chamada à SP PROC_VERIFICA_DIREITOS_ACESSO através do *spDireitoAcesso* do *DMPrincipal*. A partir do resultado retornado na execução da SP habilitamos ou não os botões *Novo*, *Alterar* e *Apagar*.

Conceda o direito de acesso *Consultar* a tela de Cadastro de Clientes ao seu usuário de teste e rode novamente a aplicação. Entre na tela de Clientes e verifique o estado dos botões. Para fazer a chamada aos formulários da aplicação a partir dos itens de menu, utilizei a função *CriarForm* que centraliza a chamada aos formulários.

Antes da chamada dos formulários é necessário preencher a variável *codMenu*, com o seguinte código:

```
codMenu := retornarCodMenu(RetirarCaracterAtalho(
  (Sender as TMenuItem).Caption));
```

Nota: Para obter mais informações sobre a centralização das chamadas aos formulários da aplicação, acesse o artigo *Chamando Formulários* disponível no Portal do Assinante.

Conclusão

Vimos neste artigo uma maneira interessante e flexível de controlar os direitos de acesso às opções de nossa aplicação. Você pode facilmente implantar e até mesmo aperfeiçoar essa idéia de controle de acesso em suas aplicações Delphi. Um grande abraço e até a próxima. ■

LIVROS DE DELPHI



Estudo Dirigido de Delphi 2005
Editora Érica
R\$ 76,00

Delphi 2005: Aplicações com Banco de Dados com Interbase 7.5 e MySQL 4.0.23

Editora Érica
R\$ 145,00



Guia do Desenvolvedor de Delphi for .NET
Makron Books
R\$ 165,00

Rave Report com Delphi
Ciência Moderna
R\$ 25,00



Universidade Delphi
Digerati Books
R\$ 49,90

Delphi: Progra. Banco de Dados e Web - R\$69,00
Delphi 7 & E-commerce: Trein. Inter. CD - R\$381,00
Delphi - R\$39,00
Aplic. das Estruturas de Dados em Delphi - R\$59,00
Impl. .NET no Delphi 8: Uma Visão Geral - R\$31,00
CD com 50 Program./Ex. Fon. p/ Delphi - R\$29,90
Delphi 8 Para Plat. .NET: Curso Completo- R\$249,00
Program. em Delphi 6: Orient. por Projeto - R\$56,00
Int. ao Desen. de Aplicações em Delphi - R\$50,00
Conhecendo e Trabalhando com Delphi 8 - R\$80,00
Estudo Dirigido de Delphi 8 - R\$58,00
Pr Pascal: Ling. do Turbo Pascal do Delphi -R\$52,00
Kylux: Delphi Linux com Interbase/Firebird - R\$52,00
Delphi 6: Conceitos Básicos (E-Book) - R\$27,90
Acessando MySQL com Delphi 7 (em CD) - R\$20,00
Dominando o Delphi 7: A Bíblia - R\$189,00
Redes Neurais em Delphi - R\$24,00
Estudo Dirigido de Delphi 7: Avançado - R\$65,00
CLX Portabilidade com Delphi 7 e Kylux 3 - R\$39,00
Delphi 7: Apl. Avan. de Banco de Dados - R\$87,00
Delphi 7: Conceitos Básicos - R\$42,00
Estudo Dirigido de Delphi 7 - R\$74,00
Programação Gráfica em Delphi 6 - R\$50,00
Delphi/Kylux: Desenv. de Banco de Dados - R\$72,00
Boleto Bancário em Delphi - R\$35,00
Conectividade Utilizando Delphi 6 - R\$40,00



FRETE GRÁTIS
Para todo o Brasil!

LivrosdeProgramação.com.br