Curso Básico de Delphi

Por Edwar Saliba Júnior

Índice:

- Diversos Cadastros
- Formulários MDI
- Formatação de Campos
- Relatórios

Escopo do Curso

1) Apresentação da Ferramenta Delphi
2) Conceitos Básicos de Programação em Delphi
3) Construção de DLL's
4) Construção de Objetos
5) Manipulação de Imagens
6) Banco de Dados

1) Apresentação e Introdução ao DELPHI

- Pessoal
  - Mini-curriculum Vitae
- Linguagem Visual (RAD)
  - Histórico das Linguagens de Programação
- Orientação a Objetos e Eventos
  - Introdução
- Ambiente de Trabalho e Principais Ferramentas (IDE)
  - Apresentação do Ambiente
  - Menus
  - Paleta de Componentes
  - Object Inspector
  - Project Options
  - SQL Explorer
  - SQL Monitor
  - DataBase Desktop
  - BDE Administrator
  - Image Editor
  - Form Wizard
- Sintaxe da Linguagem
- Lógica

Exemplo de Software

1) Exemplo 1: prjShowMessage.exe - Software simples com uso de "Show Message":

Fonte:

```
program prjShowMessage;

uses
  Forms,
  untShowMessage in 'untShowMessage.pas' {frmShowMessage};
```

```
{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TfrmShowMessage, frmShowMessage);
  Application.Run;
end.
```

-X-X-X-X-X-X-X-X-X-

```
unit untShowMessage;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls;

type
  TfrmShowMessage = class(TForm)
    Button1: TButton;
    edtFrase: TEdit;
    Label1: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmShowMessage: TfrmShowMessage;

implementation

{$R *.DFM}

procedure TfrmShowMessage.Button1Click(Sender: TObject);
begin
  ShowMessage (edtFrase.Text);
end;

end.
```

2) Exemplo opcional de uma aplicação mestre-detalhe:

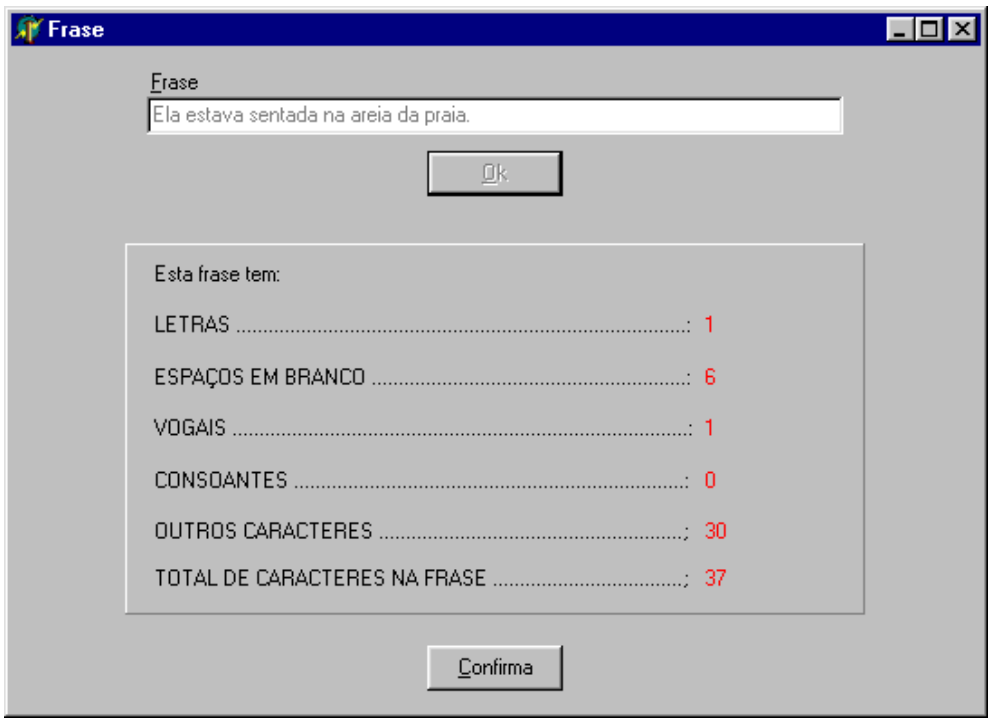
$(DELPHI)\Demos\Db\Filter

2) Conceitos Básicos de Programação em Delphi

- Características
  - Fácil entendimento
  - Fortemente tipada
  - Compilada
  - Possibilita reutilização de código
  - Possibilita criação e uso de DLL's (Dynamic Link Libraries)
  - Linguagem Híbrida
    - Orientada a Objetos
      - Herança
      - Encapsulamento
        - Propriedades
        - Métodos
      - Polimorfismo
    - Orientada a Eventos
  - Possibilita utilização de código em Assembler
- Símbolos Especiais
- Palavras Reservadas
- Números
- Constantes
- Expressões
- Identificadores
- Declarações
- Blocos de **procedimentos** ou **funções**.

Exercícios práticos:

1) Exemplo 2: prjFrase.exe - Software para contagem de letras:



Fonte:

```
program prjFrase;

uses
  Forms,
  untFrase in 'untFrase.pas' {frmFrase};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TfrmFrase, frmFrase);
  Application.Run;
end.
```

-X-X-X-X-X-X-X-X-X-

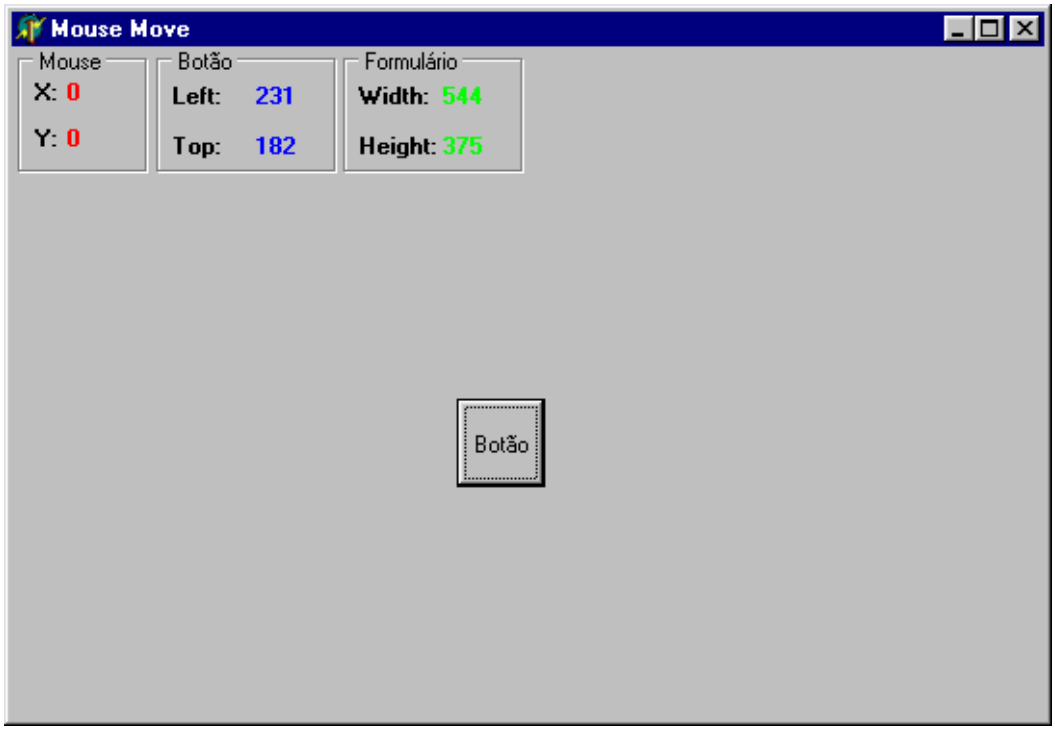
```
unit untFrase;

interface
```

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
Forms, Dialogs,
  StdCtrls, ExtCtrls;

type
  TfrmFrase = class(TForm)
    edtFrase: TEdit;
    btnOk: TButton;
    pnlResultados: TPanel;
    btnConfirma: TButton;
    lblFrase: TLabel;
    Label1: TLabel;
    Label2: TLabel;
    lblLetras: TLabel;
    Label4: TLabel;
    lblEspacosEmBranco: TLabel;
    Label6: TLabel;
    lblVogais: TLabel;
    Label8: TLabel;
    lblConsoantes: TLabel;
    Label10: TLabel;
    lblOutrosCaracteres: TLabel;
    Label3: TLabel;
    lblTotalCaracteres: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure btnOkClick(Sender: TObject);
    procedure btnConfirmaClick(Sender: TObject);
  private
    procedure ZeraLabels;
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmFrase: TfrmFrase;

implementation

{$R *.DFM}

procedure TfrmFrase.FormCreate(Sender: TObject);
begin
  ZeraLabels;
end;

procedure TfrmFrase.FormClose(Sender: TObject; var Action:
  TCloseAction);
```

```
begin
  Action := caFree;
  frmFrase := nil;
end;

procedure TfrmFrase.btnOkClick(Sender: TObject);
var
  Letras,
  EspacosEmBranco,
  Vogais,
  Consoantes,
  OutrosCaracteres,
  Cont,
  TamFrase : Integer;
  Frase : String;
begin
  Letras := 0;
  EspacosEmBranco := 0;
  Vogais := 0;
  Consoantes := 0;
  OutrosCaracteres := 0;

  edtFrase.Enabled := False;
  btnOk.Enabled := False;

  pnlResultados.Visible := True;
  btnConfirma.Visible := True;

  Frase := UpperCase (Trim (edtFrase.Text));
  TamFrase := Length (Frase);

  for Cont := 1 to TamFrase do
  begin
    if (Frase [Cont] in ['A'..'Z']) then
      Inc (Letras);

    if (Frase [Cont] = ' ') then
      Inc (EspacosEmBranco);

    if (Frase [Cont] in ['A', 'E', 'I', 'O', 'U']) then
      Inc (Vogais);

    if ((Frase [Cont] in ['A'..'Z']) and
      (not (Frase [Cont] in ['A', 'E', 'I', 'O', 'U'])))
      then
      Inc (Consoantes);

    if ((not (Frase [Cont] in ['A'..'Z'])) and
      (Frase [Cont] <> ' ')) then
      Inc (OutrosCaracteres);
  end;
```

```
  lblLetras.Caption := IntToStr (Letras);
  lblEspacosEmBranco.Caption := IntToStr (EspacosEmBranco);
  lblVogais.Caption := IntToStr (Vogais);
  lblConsoantes.Caption := IntToStr (Consoantes);
  lblOutrosCaracteres.Caption := IntToStr
    (OutrosCaracteres);
  lblTotalCaracteres.Caption := IntToStr (TamFrase);
end;

procedure TfrmFrase.btnConfirmaClick(Sender: TObject);
begin
  pnlResultados.Visible := False;
  btnConfirma.Visible := False;

  edtFrase.Enabled := True;
  btnOk.Enabled := True;

  ZeraLabels;

  edtFrase.Clear;
  if (edtFrase.CanFocus) then
    edtFrase.SetFocus;
end;

procedure TfrmFrase.ZeraLabels;
begin
  lblLetras.Caption := '0';
  lblEspacosEmBranco.Caption := '0';
  lblVogais.Caption := '0';
  lblConsoantes.Caption := '0';
  lblOutrosCaracteres.Caption := '0';
  lblTotalCaracteres.Caption := '0';
end;

end.
```

2) Exemplo 3: prjMouseMove.exe - Software para mover botão:



Fonte:

```
program prjMouseMove;

uses
  Forms,
  untMouseMove in 'untMouseMove.pas' {frmMouseMove};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TfrmMouseMove, frmMouseMove);
  Application.Run;
end.
```

```
unit untMouseMove;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
```

```
  Forms, Dialogs, StdCtrls, Buttons;

const
  DISTANCIA_MINIMA = 40;

type
  TfrmMouseMove = class(TForm)
    gbxMouse: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    lblX: TLabel;
    lblY: TLabel;
    gbxBotao: TGroupBox;
    Label3: TLabel;
    Label4: TLabel;
    lblLeft: TLabel;
    lblTop: TLabel;
    gbxFormulario: TGroupBox;
    Label5: TLabel;
    Label6: TLabel;
    lblWidth: TLabel;
    lblHeight: TLabel;
    bbnBotao: TBitBtn;
    procedure FormMouseMove(Sender: TObject; Shift:
      TShiftState; X, Y: Integer);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure FormResize(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmMouseMove: TfrmMouseMove;

implementation

{$R *.DFM}

procedure TfrmMouseMove.FormCreate(Sender: TObject);
begin
//
end;

procedure TfrmMouseMove.FormClose(Sender: TObject; var
  Action: TCloseAction);
begin
  Action := caFree;
```

```
  frmMouseMove := nil;
end;

procedure TfrmMouseMove.FormMouseMove(Sender: TObject;
  Shift: TShiftState; X, Y: Integer);
var
  Coluna,
  Linha : Integer;
begin
  lblX.Caption := IntToStr (X);
  lblY.Caption := IntToStr (Y);

  { Move botão para direita e para baixo. }
  if (Abs (X - bbnBotao.Left) <= DISTANCIA_MINIMA) and
    (Abs (Y - bbnBotao.Top) <= DISTANCIA_MINIMA) then
  begin
    Linha := (bbnBotao.Top + DISTANCIA_MINIMA) mod
      frmMouseMove.Height;
    if ((Linha + bbnBotao.Height) > (frmMouseMove.Height -
      30)) then
      bbnBotao.Top := 5
    else
      bbnBotao.Top := Linha;

    Coluna := (bbnBotao.Left + DISTANCIA_MINIMA) mod
      frmMouseMove.Width;
    if ((Coluna + bbnBotao.Width) > frmMouseMove.Width)
      then
      bbnBotao.Left := 5
    else
      bbnBotao.Left := Coluna;
  end
  else
    { Move botão para direita e para cima. }
    if (Abs (X - bbnBotao.Left) <= DISTANCIA_MINIMA) and
      (Abs (Y - (bbnBotao.Top + bbnBotao.Height)) <=
      DISTANCIA_MINIMA) then
    begin
      Linha := (bbnBotao.Top - DISTANCIA_MINIMA);
      if (Linha < 0) then
        bbnBotao.Top := (frmMouseMove.Height - 30) -
          bbnBotao.Height
      else
        bbnBotao.Top := Linha;
      Coluna := (bbnBotao.Left + DISTANCIA_MINIMA) mod
        frmMouseMove.Width;
      if ((Coluna + bbnBotao.Width) > frmMouseMove.Width)
        then
        bbnBotao.Left := 5
      else
        bbnBotao.Left := Coluna;
```

```
    end
    else
      { Move botão para esquerda e para baixo. }
      if (Abs (X - (bbnBotao.Left + bbnBotao.Width)) <=
        DISTANCIA_MINIMA) and (Abs (Y - bbnBotao.Top) <=
        DISTANCIA_MINIMA) then
      begin
        Linha := (bbnBotao.Top + DISTANCIA_MINIMA) mod
          frmMouseMove.Height;
        if ((Linha + bbnBotao.Height) >
          (frmMouseMove.Height - 30)) then
          bbnBotao.Top := 5
        else
          bbnBotao.Top := Linha;

        Coluna := (bbnBotao.Left - DISTANCIA_MINIMA);
        if (Coluna < 0) then
          bbnBotao.Left := frmMouseMove.Width -
          (bbnBotao.Width + 10)
        else
          bbnBotao.Left := Coluna;
      end
      else
        { Move botão para esquerda e para cima. }
        if (Abs (X - (bbnBotao.Left + bbnBotao.Width)) <=
          DISTANCIA_MINIMA) and (Abs (Y - (bbnBotao.Top +
          bbnBotao.Height)) <= DISTANCIA_MINIMA) then
        begin
          Linha := (bbnBotao.Top - DISTANCIA_MINIMA);
          if (Linha < 0) then
            bbnBotao.Top := (frmMouseMove.Height - 30) -
            bbnBotao.Height
          else
            bbnBotao.Top := Linha;

          Coluna := (bbnBotao.Left - DISTANCIA_MINIMA);
          if (Coluna < 0) then
            bbnBotao.Left := frmMouseMove.Width -
            (bbnBotao.Width + 10)
          else
            bbnBotao.Left := Coluna;
        end;

  lblLeft.Caption := IntToStr (bbnBotao.Left);
  lblTop.Caption := IntToStr (bbnBotao.Top);
end;

procedure TfrmMouseMove.FormResize(Sender: TObject);
begin
  lblHeight.Caption := IntToStr (frmMouseMove.Height);
  lblWidth.Caption := IntToStr (frmMouseMove.Width);
```

```
end;

end.
```

Dica → Como construir as comparações do software `prjMouseMove` **no evento** `TfrmMouseMove.FormMouseMove`**:**

PASSO 1 (Direita, Esquerda, Cima, Baixo)
=======

```
  { Move botão para direita. }
  if (Abs (X - bbnBotao.Left) <= DISTANCIA_MINIMA) then
  begin
    Coluna := (bbnBotao.Left + DISTANCIA_MINIMA) mod
      frmMouseMove.Width;
    if ((Coluna + bbnBotao.Width) > frmMouseMove.Width)
      then
      bbnBotao.Left := 5
    else
      bbnBotao.Left := Coluna;
  end;

  { Move botão para esquerda. }
  if (Abs (X - (bbnBotao.Left + bbnBotao.Width)) <=
    DISTANCIA_MINIMA) then
  begin
    Coluna := (bbnBotao.Left - DISTANCIA_MINIMA);
    if (Coluna < 0) then
      bbnBotao.Left := frmMouseMove.Width - (bbnBotao.Width
      + 10)
    else
      bbnBotao.Left := Coluna;
  end;

  { Move botão para baixo. }
  if (Abs (Y - bbnBotao.Top) <= DISTANCIA_MINIMA) then
  begin
    Linha := (bbnBotao.Top + DISTANCIA_MINIMA) mod
      frmMouseMove.Height;
    if ((Linha + bbnBotao.Height) > (frmMouseMove.Height -
      30)) then
      bbnBotao.Top := 5
    else
      bbnBotao.Top := Linha;
  end;

  { Move botão para cima. }
  if (Abs (Y - (bbnBotao.Top + bbnBotao.Height)) <=
    DISTANCIA_MINIMA) then
  begin
    Linha := (bbnBotao.Top - DISTANCIA_MINIMA);
    if (Linha < 0) then
      bbnBotao.Top := (frmMouseMove.Height - 30) -
      bbnBotao.Height
```

```
    else
      bbnBotao.Top := Linha;
  end;
```

PASSO 2 (Interseção Direita-cima, Direita-baixo, Esquerda-cima, Esquerda-baixo)

```
  { Move botão para direita e para baixo. }
  if (Abs (X - bbnBotao.Left) <= DISTANCIA_MINIMA) and
    (Abs (Y - bbnBotao.Top) <= DISTANCIA_MINIMA) then
  begin
    Linha := (bbnBotao.Top + DISTANCIA_MINIMA) mod
      frmMouseMove.Height;
    if ((Linha + bbnBotao.Height) > (frmMouseMove.Height -
      30)) then
      bbnBotao.Top := 5
    else
      bbnBotao.Top := Linha;

    Coluna := (bbnBotao.Left + DISTANCIA_MINIMA) mod
      frmMouseMove.Width;
    if ((Coluna + bbnBotao.Width) > frmMouseMove.Width)
      then
      bbnBotao.Left := 5
    else
      bbnBotao.Left := Coluna;
  end;

  { Move botão para direita e para cima. }
  if (Abs (X - bbnBotao.Left) <= DISTANCIA_MINIMA) and
    (Abs (Y - (bbnBotao.Top + bbnBotao.Height)) <=
    DISTANCIA_MINIMA) then
  begin
    Linha := (bbnBotao.Top - DISTANCIA_MINIMA);
    if (Linha < 0) then
      bbnBotao.Top := (frmMouseMove.Height - 30) -
      bbnBotao.Height
    else
      bbnBotao.Top := Linha;
    Coluna := (bbnBotao.Left + DISTANCIA_MINIMA) mod
      frmMouseMove.Width;
    if ((Coluna + bbnBotao.Width) > frmMouseMove.Width)
      then
      bbnBotao.Left := 5
    else
      bbnBotao.Left := Coluna;
  end;

  { Move botão para esquerda e para baixo. }
```

```
  if (Abs (X - (bbnBotao.Left + bbnBotao.Width)) <=
    DISTANCIA_MINIMA) and (Abs (Y - bbnBotao.Top) <=
    DISTANCIA_MINIMA) then
  begin
    Linha := (bbnBotao.Top + DISTANCIA_MINIMA) mod
      frmMouseMove.Height;
    if ((Linha + bbnBotao.Height) > (frmMouseMove.Height -
      30)) then
      bbnBotao.Top := 5
    else
      bbnBotao.Top := Linha;

    Coluna := (bbnBotao.Left - DISTANCIA_MINIMA);
    if (Coluna < 0) then
      bbnBotao.Left := frmMouseMove.Width - (bbnBotao.Width
      + 10)
    else
      bbnBotao.Left := Coluna;
  end;

  { Move botão para esquerda e para cima. }
  if (Abs (X - (bbnBotao.Left + bbnBotao.Width)) <=
    DISTANCIA_MINIMA) and (Abs (Y - (bbnBotao.Top +
    bbnBotao.Height)) <= DISTANCIA_MINIMA) then
  begin
    Linha := (bbnBotao.Top - DISTANCIA_MINIMA);
    if (Linha < 0) then
      bbnBotao.Top := (frmMouseMove.Height - 30) -
      bbnBotao.Height
    else
      bbnBotao.Top := Linha;

    Coluna := (bbnBotao.Left - DISTANCIA_MINIMA);
    if (Coluna < 0) then
      bbnBotao.Left := frmMouseMove.Width - (bbnBotao.Width
      + 10)
    else
      bbnBotao.Left := Coluna;
  end;
```

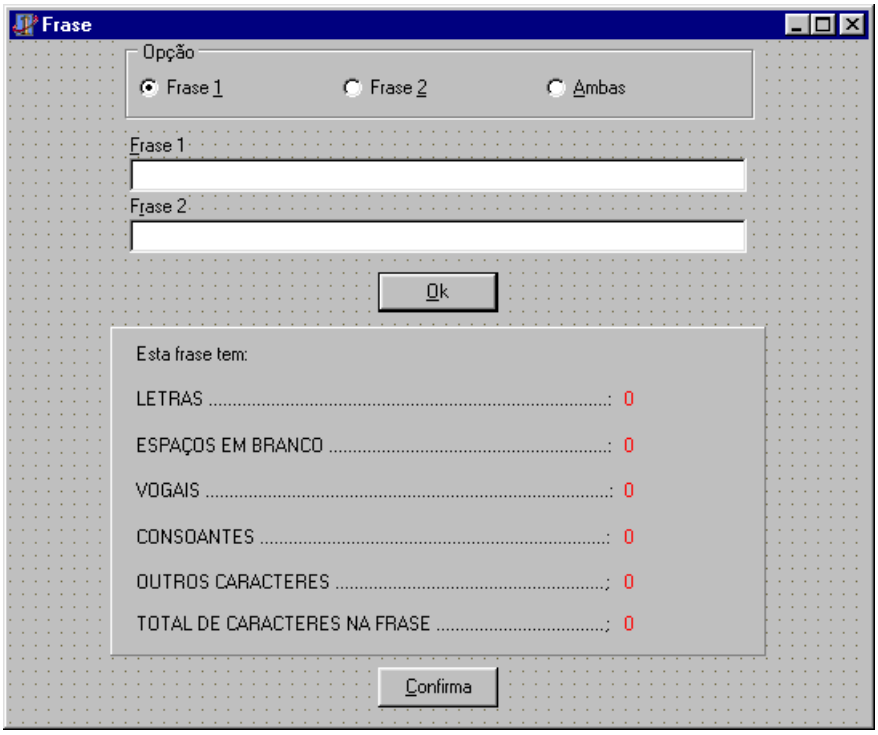
PASSO 3 (Aninhamento IF-ELSE)
=======

```
  { Move botão para direita e para baixo. }
  if (Abs (X - bbnBotao.Left) <= DISTANCIA_MINIMA) and
    (Abs (Y - bbnBotao.Top) <= DISTANCIA_MINIMA) then
  begin
    Linha := (bbnBotao.Top + DISTANCIA_MINIMA) mod
      frmMouseMove.Height;
    if ((Linha + bbnBotao.Height) > (frmMouseMove.Height -
```

```
      30)) then
      bbnBotao.Top := 5
    else
      bbnBotao.Top := Linha;

    Coluna := (bbnBotao.Left + DISTANCIA_MINIMA) mod
      frmMouseMove.Width;
    if ((Coluna + bbnBotao.Width) > frmMouseMove.Width)
      then
      bbnBotao.Left := 5
    else
      bbnBotao.Left := Coluna;
  end
  else
    { Move botão para direita e para cima. }
    if (Abs (X - bbnBotao.Left) <= DISTANCIA_MINIMA) and
      (Abs (Y - (bbnBotao.Top + bbnBotao.Height)) <=
      DISTANCIA_MINIMA) then
    begin
      Linha := (bbnBotao.Top - DISTANCIA_MINIMA);
      if (Linha < 0) then
        bbnBotao.Top := (frmMouseMove.Height - 30) -
        bbnBotao.Height
      else
        bbnBotao.Top := Linha;
      Coluna := (bbnBotao.Left + DISTANCIA_MINIMA) mod
        frmMouseMove.Width;
      if ((Coluna + bbnBotao.Width) > frmMouseMove.Width)
        then
        bbnBotao.Left := 5
      else
        bbnBotao.Left := Coluna;
    end
    else
      { Move botão para esquerda e para baixo. }
      if (Abs (X - (bbnBotao.Left + bbnBotao.Width)) <=
        DISTANCIA_MINIMA) and (Abs (Y - bbnBotao.Top) <=
        DISTANCIA_MINIMA) then
      begin
        Linha := (bbnBotao.Top + DISTANCIA_MINIMA) mod
          frmMouseMove.Height;
        if ((Linha + bbnBotao.Height) >
          (frmMouseMove.Height - 30)) then
          bbnBotao.Top := 5
        else
          bbnBotao.Top := Linha;

        Coluna := (bbnBotao.Left - DISTANCIA_MINIMA);
        if (Coluna < 0) then
          bbnBotao.Left := frmMouseMove.Width -
          (bbnBotao.Width + 10)
```

```
        else
          bbnBotao.Left := Coluna;
      end
      else
        { Move botão para esquerda e para cima. }
        if (Abs (X - (bbnBotao.Left + bbnBotao.Width)) <=
          DISTANCIA_MINIMA) and (Abs (Y - (bbnBotao.Top +
          bbnBotao.Height)) <= DISTANCIA_MINIMA) then
        begin
          Linha := (bbnBotao.Top - DISTANCIA_MINIMA);
          if (Linha < 0) then
            bbnBotao.Top := (frmMouseMove.Height - 30) -
            bbnBotao.Height
          else
            bbnBotao.Top := Linha;

          Coluna := (bbnBotao.Left - DISTANCIA_MINIMA);
          if (Coluna < 0) then
            bbnBotao.Left := frmMouseMove.Width -
            (bbnBotao.Width + 10)
          else
            bbnBotao.Left := Coluna;
        end;
```

3) Exemplo 4\RadioGroup: prjFrase.exe – Software para contagem de letras utilizando dois TEdit's e um RadioGroup:



Fonte:

```
program prjFrase;

uses
  Forms,
  untFrase in 'untFrase.pas' {frmFrase};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TfrmFrase, frmFrase);
  Application.Run;
end.
```

-X-X-X-X-X-X-X-X-X-

```
unit untFrase;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
```

```
  Forms, Dialogs,
  StdCtrls, ExtCtrls;

type
  TfrmFrase = class(TForm)
    edtFrase1: TEdit;
    btnOk: TButton;
    pnlResultados: TPanel;
    btnConfirma: TButton;
    lblFrase1: TLabel;
    Label1: TLabel;
    Label2: TLabel;
    lblLetras: TLabel;
    Label4: TLabel;
    lblEspacosEmBranco: TLabel;
    Label6: TLabel;
    lblVogais: TLabel;
    Label8: TLabel;
    lblConsoantes: TLabel;
    Label10: TLabel;
    lblOutrosCaracteres: TLabel;
    Label3: TLabel;
    lblTotalCaracteres: TLabel;
    lblFrase2: TLabel;
    edtFrase2: TEdit;
    rgpOpcao: TRadioGroup;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure btnOkClick(Sender: TObject);
    procedure btnConfirmaClick(Sender: TObject);
    procedure rgpOpcaoClick(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private
    procedure ZeraLabels;
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmFrase: TfrmFrase;

implementation

{$R *.DFM}

procedure TfrmFrase.FormCreate(Sender: TObject);
begin
  ZeraLabels;
end;
```

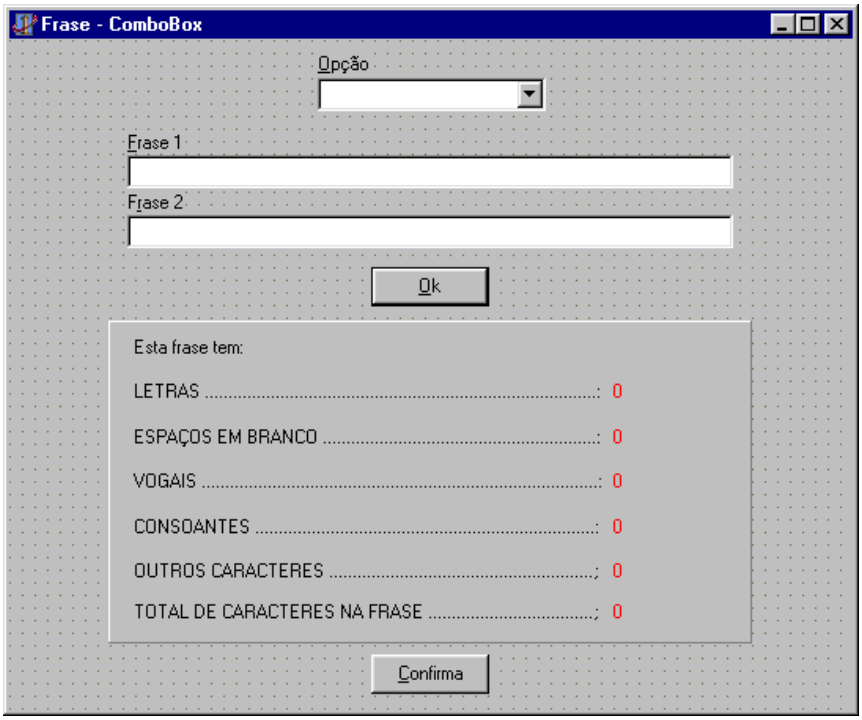
```
procedure TfrmFrase.FormClose(Sender: TObject; var Action:
TCloseAction);
begin
  Action := caFree;
  frmFrase := nil;
end;

procedure TfrmFrase.btnOkClick(Sender: TObject);
var
  Letras,
  EspacosEmBranco,
  Vogais,
  Consoantes,
  OutrosCaracteres,
  Cont,
  TamFrase : Integer;
  Frase : String;
begin
  Letras := 0;
  EspacosEmBranco := 0;
  Vogais := 0;
  Consoantes := 0;
  OutrosCaracteres := 0;

  rgpOpcao.Enabled := False;
  edtFrase1.Enabled := False;
  edtFrase2.Enabled := False;
  btnOk.Enabled := False;

  pnlResultados.Visible := True;
  btnConfirma.Visible := True;

  case (rgpOpcao.ItemIndex) of
    0 :
      Frase := Trim (edtFrase1.Text);

    1 :
      Frase := Trim (edtFrase2.Text);

    2 :
      Frase := Trim (edtFrase1.Text) + Trim
       (edtFrase2.Text);
  end;

  Frase := UpperCase (Frase);
  TamFrase := Length (Frase);

  for Cont := 1 to TamFrase do
  begin
    if (Frase [Cont] in ['A'..'Z']) then
```

```
      Inc (Letras);

    if (Frase [Cont] = ' ') then
      Inc (EspacosEmBranco);

    if (Frase [Cont] in ['A', 'E', 'I', 'O', 'U']) then
      Inc (Vogais);

    if ((Frase [Cont] in ['A'..'Z']) and
      (not (Frase [Cont] in ['A', 'E', 'I', 'O', 'U'])))
      then
      Inc (Consoantes);

    if ((not (Frase [Cont] in ['A'..'Z'])) and
      (Frase [Cont] <> ' ')) then
      Inc (OutrosCaracteres);
  end;

  lblLetras.Caption := IntToStr (Letras);
  lblEspacosEmBranco.Caption := IntToStr (EspacosEmBranco);
  lblVogais.Caption := IntToStr (Vogais);
  lblConsoantes.Caption := IntToStr (Consoantes);
  lblOutrosCaracteres.Caption := IntToStr
    (OutrosCaracteres);
  lblTotalCaracteres.Caption := IntToStr (TamFrase);
end;

procedure TfrmFrase.btnConfirmaClick(Sender: TObject);
begin
  pnlResultados.Visible := False;
  btnConfirma.Visible := False;

  rgpOpcao.Enabled := True;
  rgpOpcaoClick (Sender);
  btnOk.Enabled := True;

  ZeraLabels;

  edtFrase1.Clear;
  edtFrase2.Clear;
  rgpOpcaoClick (Sender);
end;

procedure TfrmFrase.ZeraLabels;
begin
  lblLetras.Caption := '0';
  lblEspacosEmBranco.Caption := '0';
  lblVogais.Caption := '0';
  lblConsoantes.Caption := '0';
  lblOutrosCaracteres.Caption := '0';
  lblTotalCaracteres.Caption := '0';
```

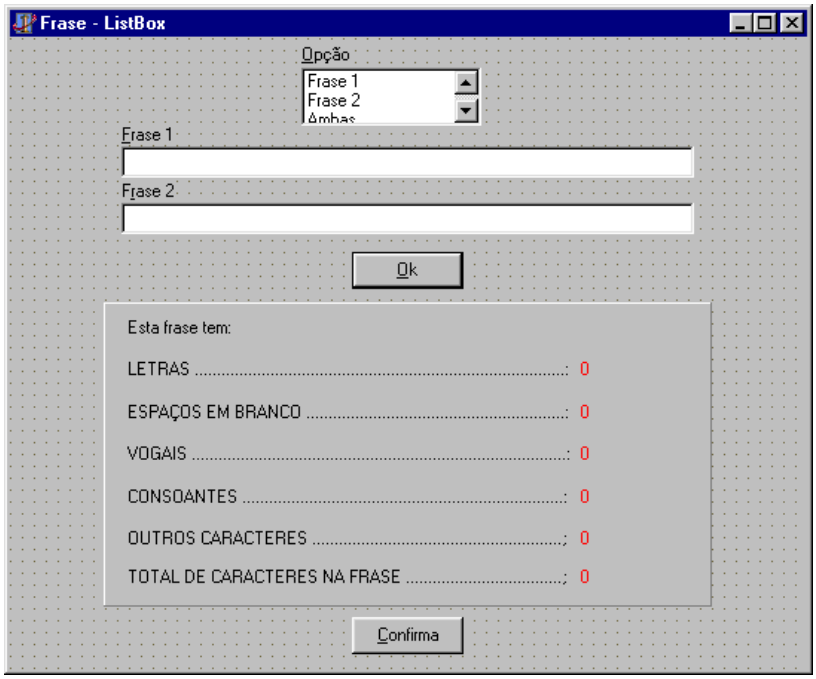
```
end;

procedure TfrmFrase.rgpOpcaoClick(Sender: TObject);
begin
  case (rgpOpcao.ItemIndex) of
    0 :
    begin
      edtFrase1.Enabled := True;
      edtFrase2.Enabled := False;
      if (edtFrase1.CanFocus) then
        edtFrase1.SetFocus;
    end;

    1 :
    begin
      edtFrase2.Enabled := True;
      edtFrase1.Enabled := False;
      if (edtFrase2.CanFocus) then
        edtFrase2.SetFocus;
    end;

    2 :
    begin
      edtFrase1.Enabled := True;
      edtFrase2.Enabled := True;
      if (edtFrase1.CanFocus) then
        edtFrase1.SetFocus;
    end;
  end;
end;

procedure TfrmFrase.FormActivate(Sender: TObject);
begin
  rgpOpcaoClick (Sender);
end;

end.
```

Variações:

4) Exemplo 4\ComboBox: prjFrase.exe – Software para contagem de letras utilizando dois TEdit's e um ComboBox:



5) Exemplo 4\ListBox: prjFrase.exe – Software para contagem de letras utilizando dois TEdit's e um ListBox:



6) Exemplo 5\Simples: prjCalculadora.exe – Software de calculadora:

Descrição: Calculadora simples para operações com apenas dois operandos.

Funções: Adição, subtração, multiplicação, divisão, raiz quadrada e Exponenciação = a^x = exp (x * ln (a))..

Teclas: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, Limpar, +, -, x, /, ^, √, Beep, Ponto e =.

Funcionamento (Especificação do Cliente):

- Todas as teclas da calculadora deverão estar desabilitadas ao se iniciar a utilização do software, com exceção da tecla "Limpar".
- A tecla "Limpar" tem a função de limpar o visor, habilitar todas as teclas e preparar a calculadora para uma nova entrada de dados.
- O usuário deverá entrar com um número, um operador, e se for o caso outro número (sempre nesta ordem, caso contrário a operação deverá ser cancelada, uma mensagem de erro deverá ser emitida e a calculadora deverá estar preparada para uma nova entrada de dados após o usuário clicar "Ok" na mensagem de erro.).
- A calculadora deverá emitir um beep para toda tecla que for apertada se o usuário deixar a tecla Beep ativada.
- Os números que aparecerão no visor da calculadora deverão estar em negrito.
- Toda vez que a calculadora gerar um total, todas as teclas, com exceção da tecla "Limpar" deverão ser desabilitadas. Sendo novamente habilitadas quando o usuário apertar a tecla "Limpar".
- Deverá ser feito controle de teclas a partir da função selecionada pelo usuário. Ex: Se a função exigir apenas um operando, o teclado numérico deverá ser desabilitado até que o usuário clique no total (=).
- Os totais de qualquer operação deverão ser emitidos somente com o pressionamento da tecla =.

Observação: Deverá ser criada uma unit com nome de "Util" onde deverão ser declaradas todas as funções matemáticas da calculadora. Se o aluno quiser adicionar mais alguma funcionalidade a calculadora, esta deverá ser apresentada ao professor.

Fonte:

```
program prjCalculadora;

uses
  Forms,
  untCalculadora in 'untCalculadora.pas' {frmCalculadora},
  untUtil in 'untUtil.pas';

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TfrmCalculadora, frmCalculadora);
```

```
  Application.Run;
end.
```

```
unit untCalculadora;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Buttons;

type
  TOperacao = (toNenhuma, toAdicao, toSubtracao,
               toMultiplicacao, toDivisao,
               toExponenciacao, toRaizQuadrada);

  TfrmCalculadora = class(TForm)
    gbxTecladoNumerico: TGroupBox;
    sbnOito: TSpeedButton;
    sbnSete: TSpeedButton;
    sbnNove: TSpeedButton;
    sbnSeis: TSpeedButton;
    sbnCinco: TSpeedButton;
    sbnQuatro: TSpeedButton;
    sbnUm: TSpeedButton;
    sbnDois: TSpeedButton;
    sbnTres: TSpeedButton;
    sbnPonto: TSpeedButton;
    sbnZero: TSpeedButton;
    gbxFuncoes: TGroupBox;
    sbnMultiplicacao: TSpeedButton;
    sbnSubtracao: TSpeedButton;
    sbnDivisao: TSpeedButton;
    sbnLimpar: TSpeedButton;
    sbnExponenciacao: TSpeedButton;
    sbnRaizQuadrada: TSpeedButton;
    sbnBeep: TSpeedButton;
    sbnIgual: TSpeedButton;
    sbnAdicao: TSpeedButton;
    lblVisor: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure sbnDivisaoClick(Sender: TObject);
    procedure sbnMultiplicacaoClick(Sender: TObject);
    procedure sbnSubtracaoClick(Sender: TObject);
    procedure sbnAdicaoClick(Sender: TObject);
    procedure sbnExponenciacaoClick(Sender: TObject);
    procedure sbnRaizQuadradaClick(Sender: TObject);
```

```
    procedure sbnIgualClick(Sender: TObject);
    procedure sbnBeepClick(Sender: TObject);
    procedure sbnLimparClick(Sender: TObject);
    procedure sbnPontoClick(Sender: TObject);
    procedure sbnZeroClick(Sender: TObject);
    procedure sbnUmClick(Sender: TObject);
    procedure sbnDoisClick(Sender: TObject);
    procedure sbnTresClick(Sender: TObject);
    procedure sbnQuatroClick(Sender: TObject);
    procedure sbnCincoClick(Sender: TObject);
    procedure sbnSeisClick(Sender: TObject);
    procedure sbnSeteClick(Sender: TObject);
    procedure sbnOitoClick(Sender: TObject);
    procedure sbnNoveClick(Sender: TObject);
  private
    { Private declarations }
    OperandoA,
    OperandoB : String;
    Operacao : TOperacao;
    TemPonto : Boolean;

    procedure HabilitaTecladoNumerico;
    procedure DesabilitaTecladoNumerico;
    procedure HabilitaFuncoesMatematicas;
    procedure DesabilitaFuncoesMatematicas;
    procedure HabilitaSom;
    procedure DesabilitaSom;
    procedure EmiteBeep;
    procedure LimpaVisor;
    procedure AposTeclarIgualOuOcorrerInconsistencia;
    procedure AposTeclarLimpar;
    procedure HabilitaCalculadoraParaNovaOperacao;
    function UsuarioJaEntrouComOperandoA: Boolean;
    function UsuarioJaEntrouComTodosOperandos: Boolean;
    function FaltaOperandosOuOperadorNoTotal : Boolean;
    function TemErroDeFaltaDeOperando : Boolean;
    procedure ObtemOperando;
    procedure FormaOperando(Numero: String);
    function OperandoValido(Operando: String): Boolean;
    procedure VerificaOperacao;
  public
    { Public declarations }
  end;

var
  frmCalculadora: TfrmCalculadora;

implementation

uses
  untUtil;
```

```
{$R *.DFM}

procedure TfrmCalculadora.FormCreate(Sender: TObject);
begin
  OperandoA := '';
  OperandoB := '';
  Operacao := toNenhuma;
  lblVisor.Caption := '';
  sbnBeep.Caption := 'Off';
  TemPonto := False;

  DecimalSeparator := '.';

  { Desabilita todo o teclado conforme especificação do
    usuário. }
  AposTeclarIgualOuOcorrerInconsistencia;
end;

procedure TfrmCalculadora.FormClose(Sender: TObject; var
  Action: TCloseAction);
begin
  Action := caFree;
  frmCalculadora := nil;
end;

{ Escolha da operações matemática. ====================== }

procedure TfrmCalculadora.sbnLimparClick(Sender: TObject);
begin
  EmiteBeep;
  LimpaVisor;

  { Habita todo o teclado conforme especificação do
    usuário. }
  AposTeclarLimpar;

  { Prepara a calculadora para uma nova operação conforme
    especificação do usuário. }
  HabilitaCalculadoraParaNovaOperacao;
end;

procedure TfrmCalculadora.sbnDivisaoClick(Sender: TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toDivisao;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;
```

```
procedure TfrmCalculadora.sbnMultiplicacaoClick(Sender:
  TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toMultiplicacao;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnSubtracaoClick(Sender:
  TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toSubtracao;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnAdicaoClick(Sender: TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toAdicao;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnExponenciacaoClick(Sender:
  TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toExponenciacao;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnRaizQuadradaClick(Sender:
  TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toRaizQuadrada;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnIgualClick(Sender: TObject);
var
```

```
  Operando_A,
  Operando_B,
  Resultado : Extended;
begin
  EmiteBeep;

  if (Operacao <> toNenhuma) then
  begin
    ObtemOperando;

    if (not (FaltaOperandosOuOperadorNoTotal)) then
    begin
      Operando_B := 0;
      Resultado := 0;

      Operando_A := StrToFloat (OperandoA);
      if (Operacao <> toRaizQuadrada) then
        Operando_B := StrToFloat (OperandoB);

      case (Operacao) of
        toAdicao :
          Resultado := Adicao (Operando_A, Operando_B);

        toSubtracao :
          Resultado := Subtracao (Operando_A, Operando_B);

        toMultiplicacao :
          Resultado := Multiplicacao (Operando_A,
            Operando_B);

        toDivisao :
          Resultado := Divisao (Operando_A, Operando_B);

        toExponenciacao :
          Resultado := Exponenciacao (Operando_A,
            Operando_B);

        toRaizQuadrada :
          Resultado := RaizQuadrada (Operando_A);
      end;

      lblVisor.Caption := FloatToStr (Resultado);
    end;
  end
  else
    MessageDlg('Você precisa selecionar uma operação antes
      de tentar totalizá-la. A '+#13+#10+'operação será
      abortada.', mtError, [mbOK], 0);

  AposTeclarIgualOuOcorrerInconsistencia;
end;
```

```
{ Habilita e desabilita botões. ======================== }

procedure TfrmCalculadora.HabilitaTecladoNumerico;
begin
  sbnZero.Enabled := True;
  sbnUm.Enabled := True;
  sbnDois.Enabled := True;
  sbnTres.Enabled := True;
  sbnQuatro.Enabled := True;
  sbnCinco.Enabled := True;
  sbnSeis.Enabled := True;
  sbnSete.Enabled := True;
  sbnOito.Enabled := True;
  sbnNove.Enabled := True;
  sbnPonto.Enabled := True;
end;

procedure TfrmCalculadora.DesabilitaTecladoNumerico;
begin
  sbnZero.Enabled := False;
  sbnUm.Enabled := False;
  sbnDois.Enabled := False;
  sbnTres.Enabled := False;
  sbnQuatro.Enabled := False;
  sbnCinco.Enabled := False;
  sbnSeis.Enabled := False;
  sbnSete.Enabled := False;
  sbnOito.Enabled := False;
  sbnNove.Enabled := False;
  sbnPonto.Enabled := False;
end;

procedure TfrmCalculadora.HabilitaFuncoesMatematicas;
begin
  sbnDivisao.Enabled := True;
  sbnMultiplicacao.Enabled := True;
  sbnSubtracao.Enabled := True;
  sbnAdicao.Enabled := True;
  sbnExponenciacao.Enabled := True;
  sbnRaizQuadrada.Enabled := True;
  sbnIgual.Enabled := True;
end;

procedure TfrmCalculadora.DesabilitaFuncoesMatematicas;
begin
  sbnDivisao.Enabled := False;
  sbnMultiplicacao.Enabled := False;
  sbnSubtracao.Enabled := False;
  sbnAdicao.Enabled := False;
  sbnExponenciacao.Enabled := False;
```

```
  sbnRaizQuadrada.Enabled := False;
  sbnIgual.Enabled := False;
end;

procedure TfrmCalculadora.HabilitaSom;
begin
  sbnBeep.Enabled := True;
end;

procedure TfrmCalculadora.DesabilitaSom;
begin
  sbnBeep.Enabled := False;
end;

procedure TfrmCalculadora.AposTeclarLimpar;
begin
  HabilitaTecladoNumerico;
  HabilitaFuncoesMatematicas;
  HabilitaSom;
end;

procedure
  TfrmCalculadora.AposTeclarIgualOuOcorrerInconsistencia;
begin
  DesabilitaTecladoNumerico;
  DesabilitaFuncoesMatematicas;
  DesabilitaSom;

  HabilitaCalculadoraParaNovaOperacao;
end;

{ Manipulação do som do teclado da calculadora. ========= }

procedure TfrmCalculadora.EmiteBeep;
begin
  if (sbnBeep.Caption = 'On') then
    Beep;
end;

procedure TfrmCalculadora.sbnBeepClick(Sender: TObject);
begin
  EmiteBeep;

  if (TSpeedButton (Sender).Caption = 'Off') then
    TSpeedButton (Sender).Caption := 'On'
  else
    TSpeedButton (Sender).Caption := 'Off';
end;

{ Funções de reset da calculadora. ====================== }
```

```
procedure TfrmCalculadora.LimpaVisor;
begin
  lblVisor.Caption := '';
end;

procedure
  TfrmCalculadora.HabilitaCalculadoraParaNovaOperacao;
begin
  Operacao := toNenhuma;
  OperandoA := '';
  OperandoB := '';
  TemPonto := False;
end;

{ Verificações. ======================================== }

function TfrmCalculadora.OperandoValido (Operando : String)
  : Boolean;
begin
  Result := True;

  try
    StrToFloat (Operando);
  except
    Result := False;
  end;
end;

function TfrmCalculadora.UsuarioJaEntrouComOperandoA :
Boolean;
begin
  Result := OperandoValido (OperandoA);
end;

function TfrmCalculadora.UsuarioJaEntrouComTodosOperandos :
  Boolean;
begin
  if (Operacao = toRaizQuadrada) then
    Result := OperandoValido (OperandoA)
  else
    Result := (OperandoValido (OperandoA) and
      OperandoValido (OperandoB));
end;

{ Desabilita teclado numérico caso operação solicite apenas
  um operando. }
procedure TfrmCalculadora.VerificaOperacao;
begin
  if (Operacao = toRaizQuadrada) then
    DesabilitaTecladoNumerico;
end;
```

```
{ Mensagens de erro. ================================== }

function TfrmCalculadora.TemErroDeFaltaDeOperando :
  Boolean;
begin
  Result := False;

  if (not (UsuarioJaEntrouComOperandoA)) then
  begin
    Result := True;

    MessageDlg('Você precisa entrar com um número válido
      antes de escolher uma '+#13+#10+'operação matemática.
      A operação será abortada.', mtError, [mbOK], 0);

    AposTeclarIgualOuOcorrerInconsistencia;
  end
  else
    VerificaOperacao;
end;

function TfrmCalculadora.FaltaOperandosOuOperadorNoTotal :
  Boolean;
begin
  Result := False;

  if (not (UsuarioJaEntrouComTodosOperandos)) then
  begin
    if (Operacao = toRaizQuadrada) then
      MessageDlg('Para operações envolvendo Raiz Quadrada,
        você deverá escolher '+#13+#10+'apenas um operando.
        Esta operação deverá ser realizada na seguinte '
        +#13+#10+'ordem: OPERANDO e OPERADOR. A operação
        será abortada.', mtError, [mbOK], 0)
    else
      MessageDlg('O número de operandos para o tipo de
        operação matemática escolhida '+#13+#10+'deve ser
        dois. Estes devem seguir a seguinte ordem:
        OPERANDO, '+#13+#10+'OPERADOR e OPERANDO. A
        operação será abortada.', mtError, [mbOK], 0);

    AposTeclarIgualOuOcorrerInconsistencia;

    Result := True;
  end;
end;

{ Captura da entrada de operandos. ==================== }

procedure TfrmCalculadora.ObtemOperando;
```

```
begin
  if (OperandoA = '') then
    OperandoA := Trim (lblVisor.Caption)
  else
    if (Operacao <> toRaizQuadrada) then
      OperandoB := Trim (lblVisor.Caption);
end;

procedure TfrmCalculadora.FormaOperando (Numero : String);
begin
  if (Numero = '.') then
  begin
    if (not (TemPonto)) then
    begin
      TemPonto := True;

      if (lblVisor.Caption = '') then
        Numero := '0.';

      lblVisor.Caption := lblVisor.Caption + Numero;
    end;
  end
  else
    lblVisor.Caption := lblVisor.Caption + Numero;
end;

{ Entrada de valores. ================================== }

procedure TfrmCalculadora.sbnPontoClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('.');
end;

procedure TfrmCalculadora.sbnZeroClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('0');
end;

procedure TfrmCalculadora.sbnUmClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('1');
end;

procedure TfrmCalculadora.sbnDoisClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('2');
end;
```

```
procedure TfrmCalculadora.sbnTresClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('3');
end;

procedure TfrmCalculadora.sbnQuatroClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('4');
end;

procedure TfrmCalculadora.sbnCincoClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('5');
end;

procedure TfrmCalculadora.sbnSeisClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('6');
end;

procedure TfrmCalculadora.sbnSeteClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('7');
end;

procedure TfrmCalculadora.sbnOitoClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('8');
end;

procedure TfrmCalculadora.sbnNoveClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('9');
end;

end.
```

```
unit untUtil;

interface
```

```
function Adicao (A, B : Extended) : Extended;
function Subtracao (A, B : Extended) : Extended;
function Multiplicacao (A, B : Extended) : Extended;
function Divisao (A, B : Extended) : Extended;
function Exponenciacao (A, B : Extended) : Extended;
function RaizQuadrada (A : Extended) : Extended;

implementation

function Adicao (A, B : Extended) : Extended;
begin
  Result := A + B;
end;

function Subtracao (A, B : Extended) : Extended;
begin
  Result := A - B;
end;

function Multiplicacao (A, B : Extended) : Extended;
begin
  Result := A * B;
end;

function Divisao (A, B : Extended) : Extended;
begin
  Result := A / B;
end;

function Exponenciacao (A, B : Extended) : Extended;
begin
  Result := Exp (B * Ln (A));
end;

function RaizQuadrada (A : Extended) : Extended;
begin
  Result := Sqrt (A);
end;

end.
```

3) Construção de DLL's

- Conceitos
  - Pra que serve ?
    - Garantir segurança
    - Diminuir tamanho do Executável
    - Economia de memória
  - Onde e quando usar ?
  - Chamada Estática e Dinâmica
  - Onde deve ser colocada ?

Exemplo de Software

1) Exemplo 5\DLL\Estatica: prjCalculadora.exe – Software de calculadora:

```
library Util;

{ Important note about DLL memory management: ShareMem must
  be the first unit in your library's USES clause AND your
  project's (select Project-View Source) USES clause if
  your DLL exports any procedures or functions that pass
  strings as parameters or function results. This applies
  to all strings passed to and from your DLL--even those
  that are nested in records and classes. ShareMem is the
  interface unit to the BORLNDMM.DLL shared memory manager,
  which must be deployed along with your DLL. To avoid
  using BORLNDMM.DLL, pass string information using PChar
  or ShortString parameters. }

uses
  SysUtils,
  Classes;

{$R *.RES}

function Adicao (A, B : Extended) : Extended;
begin
  Result := A + B;
end;

function Subtracao (A, B : Extended) : Extended;
begin
  Result := A - B;
end;

function Multiplicacao (A, B : Extended) : Extended;
begin
  Result := A * B;
```

```
end;

function Divisao (A, B : Extended) : Extended;
begin
  Result := A / B;
end;

function Exponenciacao (A, B : Extended) : Extended;
begin
  Result := Exp (B * Ln (A));
end;

function RaizQuadrada (A : Extended) : Extended;
begin
  Result := Sqrt (A);
end;

exports
  Adicao index 1,
  Subtracao index 2,
  Multiplicacao index 3,
  Divisao index 4,
  Exponenciacao index 5,
  RaizQuadrada index 6;

begin
end.
```

```
program prjCalculadora;

uses
  Forms,
  untCalculadora in 'untCalculadora.pas' {frmCalculadora};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TfrmCalculadora, frmCalculadora);
  Application.Run;
end.
```

```
unit untCalculadora;

interface
```

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Buttons;

type
  TOperacao = (toNenhuma, toAdicao, toSubtracao,
    toMultiplicacao, toDivisao, toExponenciacao,
    toRaizQuadrada);

  TfrmCalculadora = class(TForm)
    gbxTecladoNumerico: TGroupBox;
    sbnOito: TSpeedButton;
    sbnSete: TSpeedButton;
    sbnNove: TSpeedButton;
    sbnSeis: TSpeedButton;
    sbnCinco: TSpeedButton;
    sbnQuatro: TSpeedButton;
    sbnUm: TSpeedButton;
    sbnDois: TSpeedButton;
    sbnTres: TSpeedButton;
    sbnPonto: TSpeedButton;
    sbnZero: TSpeedButton;
    gbxFuncoes: TGroupBox;
    sbnMultiplicacao: TSpeedButton;
    sbnSubtracao: TSpeedButton;
    sbnDivisao: TSpeedButton;
    sbnLimpar: TSpeedButton;
    sbnExponenciacao: TSpeedButton;
    sbnRaizQuadrada: TSpeedButton;
    sbnBeep: TSpeedButton;
    sbnIgual: TSpeedButton;
    sbnAdicao: TSpeedButton;
    lblVisor: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure sbnDivisaoClick(Sender: TObject);
    procedure sbnMultiplicacaoClick(Sender: TObject);
    procedure sbnSubtracaoClick(Sender: TObject);
    procedure sbnAdicaoClick(Sender: TObject);
    procedure sbnExponenciacaoClick(Sender: TObject);
    procedure sbnRaizQuadradaClick(Sender: TObject);
    procedure sbnIgualClick(Sender: TObject);
    procedure sbnBeepClick(Sender: TObject);
    procedure sbnLimparClick(Sender: TObject);
    procedure sbnPontoClick(Sender: TObject);
    procedure sbnZeroClick(Sender: TObject);
    procedure sbnUmClick(Sender: TObject);
    procedure sbnDoisClick(Sender: TObject);
    procedure sbnTresClick(Sender: TObject);
    procedure sbnQuatroClick(Sender: TObject);
```

```
    procedure sbnCincoClick(Sender: TObject);
    procedure sbnSeisClick(Sender: TObject);
    procedure sbnSeteClick(Sender: TObject);
    procedure sbnOitoClick(Sender: TObject);
    procedure sbnNoveClick(Sender: TObject);
  private
    { Private declarations }
    OperandoA,
    OperandoB : String;
    Operacao : TOperacao;
    TemPonto : Boolean;

    procedure HabilitaTecladoNumerico;
    procedure DesabilitaTecladoNumerico;
    procedure HabilitaFuncoesMatematicas;
    procedure DesabilitaFuncoesMatematicas;
    procedure HabilitaSom;
    procedure DesabilitaSom;
    procedure EmiteBeep;
    procedure LimpaVisor;
    procedure AposTeclarIgualOuOcorrerInconsistencia;
    procedure AposTeclarLimpar;
    procedure HabilitaCalculadoraParaNovaOperacao;
    function UsuarioJaEntrouComOperandoA: Boolean;
    function UsuarioJaEntrouComTodosOperandos: Boolean;
    function FaltaOperandosOuOperadorNoTotal : Boolean;
    function TemErroDeFaltaDeOperando : Boolean;
    procedure ObtemOperando;
    procedure FormaOperando(Numero: String);
    function OperandoValido(Operando: String): Boolean;
    procedure VerificaOperacao;
  public
    { Public declarations }
  end;

var
  frmCalculadora: TfrmCalculadora;

implementation

function Adicao (A, B : Extended) : Extended; external
  'Util.dll';
function Subtracao (A, B : Extended) : Extended; external
  'Util.dll';
function Multiplicacao (A, B : Extended) : Extended;
  external 'Util.dll';
function Divisao (A, B : Extended) : Extended; external
  'Util.dll';
function Exponenciacao (A, B : Extended) : Extended;
  external 'Util.dll';
function RaizQuadrada (A : Extended) : Extended; external
```

```
  'Util.dll';

{$R *.DFM}

procedure TfrmCalculadora.FormCreate(Sender: TObject);
begin
  OperandoA := '';
  OperandoB := '';
  Operacao := toNenhuma;
  lblVisor.Caption := '';
  sbnBeep.Caption := 'Off';
  TemPonto := False;

  DecimalSeparator := '.';

  { Desabilita todo o teclado conforme especificação do
    usuário. }
  AposTeclarIgualOuOcorrerInconsistencia;
end;

procedure TfrmCalculadora.FormClose(Sender: TObject; var
  Action: TCloseAction);
begin
  Action := caFree;
  frmCalculadora := nil;
end;

{ Escolha da operações matemática. ====================== }

procedure TfrmCalculadora.sbnLimparClick(Sender: TObject);
begin
  EmiteBeep;
  LimpaVisor;

  { Habita todo o teclado conforme especificação do
usuário. }
  AposTeclarLimpar;

  { Prepara a calculadora para uma nova operação conforme
    especificação do usuário. }
  HabilitaCalculadoraParaNovaOperacao;
end;

procedure TfrmCalculadora.sbnDivisaoClick(Sender: TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toDivisao;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;
```

```
procedure TfrmCalculadora.sbnMultiplicacaoClick(Sender:
  TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toMultiplicacao;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnSubtracaoClick(Sender:
  TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toSubtracao;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnAdicaoClick(Sender: TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toAdicao;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnExponenciacaoClick(Sender:
  TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toExponenciacao;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnRaizQuadradaClick(Sender:
  TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toRaizQuadrada;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnIgualClick(Sender: TObject);
```

```
var
  Operando_A,
  Operando_B,
  Resultado : Extended;
begin
  EmiteBeep;

  if (Operacao <> toNenhuma) then
  begin
    ObtemOperando;

    if (not (FaltaOperandosOuOperadorNoTotal)) then
    begin
      Operando_B := 0;
      Resultado := 0;

      Operando_A := StrToFloat (OperandoA);
      if (Operacao <> toRaizQuadrada) then
        Operando_B := StrToFloat (OperandoB);

      case (Operacao) of
        toAdicao :
          Resultado := Adicao (Operando_A, Operando_B);

        toSubtracao :
          Resultado := Subtracao (Operando_A, Operando_B);

        toMultiplicacao :
          Resultado := Multiplicacao (Operando_A,
            Operando_B);

        toDivisao :
          Resultado := Divisao (Operando_A, Operando_B);

        toExponenciacao :
          Resultado := Exponenciacao (Operando_A,
            Operando_B);

        toRaizQuadrada :
          Resultado := RaizQuadrada (Operando_A);
      end;

      lblVisor.Caption := FloatToStr (Resultado);
    end;
  end
  else
    MessageDlg('Você precisa selecionar uma operação antes
      de tentar totalizá-la. A ' +#13+#10+'operação será
      abortada.', mtError, [mbOK], 0);

  AposTeclarIgualOuOcorrerInconsistencia;
```

```
end;

{ Habilita e desabilita botões. ========================= }

procedure TfrmCalculadora.HabilitaTecladoNumerico;
begin
  sbnZero.Enabled := True;
  sbnUm.Enabled := True;
  sbnDois.Enabled := True;
  sbnTres.Enabled := True;
  sbnQuatro.Enabled := True;
  sbnCinco.Enabled := True;
  sbnSeis.Enabled := True;
  sbnSete.Enabled := True;
  sbnOito.Enabled := True;
  sbnNove.Enabled := True;
  sbnPonto.Enabled := True;
end;

procedure TfrmCalculadora.DesabilitaTecladoNumerico;
begin
  sbnZero.Enabled := False;
  sbnUm.Enabled := False;
  sbnDois.Enabled := False;
  sbnTres.Enabled := False;
  sbnQuatro.Enabled := False;
  sbnCinco.Enabled := False;
  sbnSeis.Enabled := False;
  sbnSete.Enabled := False;
  sbnOito.Enabled := False;
  sbnNove.Enabled := False;
  sbnPonto.Enabled := False;
end;

procedure TfrmCalculadora.HabilitaFuncoesMatematicas;
begin
  sbnDivisao.Enabled := True;
  sbnMultiplicacao.Enabled := True;
  sbnSubtracao.Enabled := True;
  sbnAdicao.Enabled := True;
  sbnExponenciacao.Enabled := True;
  sbnRaizQuadrada.Enabled := True;
  sbnIgual.Enabled := True;
end;

procedure TfrmCalculadora.DesabilitaFuncoesMatematicas;
begin
  sbnDivisao.Enabled := False;
  sbnMultiplicacao.Enabled := False;
  sbnSubtracao.Enabled := False;
  sbnAdicao.Enabled := False;
```

```
  sbnExponenciacao.Enabled := False;
  sbnRaizQuadrada.Enabled := False;
  sbnIgual.Enabled := False;
end;

procedure TfrmCalculadora.HabilitaSom;
begin
  sbnBeep.Enabled := True;
end;

procedure TfrmCalculadora.DesabilitaSom;
begin
  sbnBeep.Enabled := False;
end;

procedure TfrmCalculadora.AposTeclarLimpar;
begin
  HabilitaTecladoNumerico;
  HabilitaFuncoesMatematicas;
  HabilitaSom;
end;

procedure
  TfrmCalculadora.AposTeclarIgualOuOcorrerInconsistencia;
begin
  DesabilitaTecladoNumerico;
  DesabilitaFuncoesMatematicas;
  DesabilitaSom;

  HabilitaCalculadoraParaNovaOperacao;
end;

{ Manipulação do som do teclado da calculadora. ========= }

procedure TfrmCalculadora.EmiteBeep;
begin
  if (sbnBeep.Caption = 'On') then
    Beep;
end;

procedure TfrmCalculadora.sbnBeepClick(Sender: TObject);
begin
  EmiteBeep;

  if (TSpeedButton (Sender).Caption = 'Off') then
    TSpeedButton (Sender).Caption := 'On'
  else
    TSpeedButton (Sender).Caption := 'Off';
end;

{ Funções de reset da calculadora. ====================== }
```

```
procedure TfrmCalculadora.LimpaVisor;
begin
  lblVisor.Caption := '';
end;

procedure
  TfrmCalculadora.HabilitaCalculadoraParaNovaOperacao;
begin
  Operacao := toNenhuma;
  OperandoA := '';
  OperandoB := '';
  TemPonto := False;
end;

{ Verificações. ======================================== }

function TfrmCalculadora.OperandoValido (Operando : String)
  : Boolean;
begin
  Result := True;

  try
    StrToFloat (Operando);
  except
    Result := False;
  end;
end;

function TfrmCalculadora.UsuarioJaEntrouComOperandoA :
  Boolean;
begin
  Result := OperandoValido (OperandoA);
end;

function TfrmCalculadora.UsuarioJaEntrouComTodosOperandos :
  Boolean;
begin
  if (Operacao = toRaizQuadrada) then
    Result := OperandoValido (OperandoA)
  else
    Result := (OperandoValido (OperandoA) and
OperandoValido (OperandoB));
end;

{ Desabilita teclado numérico caso operação solicite apenas
  um operando. }
procedure TfrmCalculadora.VerificaOperacao;
begin
  if (Operacao = toRaizQuadrada) then
    DesabilitaTecladoNumerico;
```

```
end;

{ Mensagens de erro. ================================== }

function TfrmCalculadora.TemErroDeFaltaDeOperando :
  Boolean;
begin
  Result := False;

  if (not (UsuarioJaEntrouComOperandoA)) then
  begin
    Result := True;

    MessageDlg('Você precisa entrar com um número válido
      antes de escolher uma ' +#13+#10+'operação
      matemática. A operação será abortada.', mtError,
      [mbOK], 0);

    AposTeclarIgualOuOcorrerInconsistencia;
  end
  else
    VerificaOperacao;
end;

function TfrmCalculadora.FaltaOperandosOuOperadorNoTotal :
  Boolean;
begin
  Result := False;

  if (not (UsuarioJaEntrouComTodosOperandos)) then
  begin
    if (Operacao = toRaizQuadrada) then
      MessageDlg('Para operações envolvendo Raiz Quadrada,
        você deverá escolher ' +#13+#10+'apenas um
        operando. Esta operação deverá ser realizada na
        seguinte ' +#13+#10+'ordem: OPERANDO e OPERADOR. A
        operação será abortada.', mtError, [mbOK], 0)
    else
      MessageDlg('O número de operandos para o tipo de
        operação matemática escolhida ' +#13+#10+'deve ser
        dois. Estes devem seguir a seguinte ordem:
        OPERANDO, ' +#13+#10+'OPERADOR e OPERANDO. A
        operação será abortada.', mtError, [mbOK], 0);

    AposTeclarIgualOuOcorrerInconsistencia;

    Result := True;
  end;
end;

{ Captura da entrada de operandos. ==================== }
```

```
procedure TfrmCalculadora.ObtemOperando;
begin
  if (OperandoA = '') then
    OperandoA := Trim (lblVisor.Caption)
  else
    if (Operacao <> toRaizQuadrada) then
      OperandoB := Trim (lblVisor.Caption);
end;

procedure TfrmCalculadora.FormaOperando (Numero : String);
begin
  if (Numero = '.') then
  begin
    if (not (TemPonto)) then
    begin
      TemPonto := True;

      if (lblVisor.Caption = '') then
        Numero := '0.';

      lblVisor.Caption := lblVisor.Caption + Numero;
    end;
  end
  else
    lblVisor.Caption := lblVisor.Caption + Numero;
end;

{ Entrada de valores. ================================== }

procedure TfrmCalculadora.sbnPontoClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('.');
end;

procedure TfrmCalculadora.sbnZeroClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('0');
end;

procedure TfrmCalculadora.sbnUmClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('1');
end;

procedure TfrmCalculadora.sbnDoisClick(Sender: TObject);
begin
  EmiteBeep;
```

```
  FormaOperando ('2');
end;

procedure TfrmCalculadora.sbnTresClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('3');
end;

procedure TfrmCalculadora.sbnQuatroClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('4');
end;

procedure TfrmCalculadora.sbnCincoClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('5');
end;

procedure TfrmCalculadora.sbnSeisClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('6');
end;

procedure TfrmCalculadora.sbnSeteClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('7');
end;

procedure TfrmCalculadora.sbnOitoClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('8');
end;

procedure TfrmCalculadora.sbnNoveClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('9');
end;

end.
```

2) Exemplo 5\DLL\Dinamica: prjCalculadora.exe – Software de calculadora:

```
library Util;

{ Important note about DLL memory management: ShareMem must
  be the first unit in your library's USES clause AND your
  project's (select Project-View Source) USES clause if
  your DLL exports any procedures or functions that pass
  strings as parameters or function results. This applies
  to all strings passed to and from your DLL--even those
  that are nested in records and classes. ShareMem is the
  interface unit to the BORLNDMM.DLL shared memory manager,
  which must be deployed along with your DLL. To avoid
  using BORLNDMM.DLL, pass string information using PChar
  or ShortString parameters. }

uses
  SysUtils,
  Classes;

{$R *.RES}

function Adicao (A, B : Extended) : Extended;
begin
  Result := A + B;
end;

function Subtracao (A, B : Extended) : Extended;
begin
  Result := A - B;
end;

function Multiplicacao (A, B : Extended) : Extended;
begin
  Result := A * B;
end;

function Divisao (A, B : Extended) : Extended;
begin
  Result := A / B;
end;

function Exponenciacao (A, B : Extended) : Extended;
begin
  Result := Exp (B * Ln (A));
end;

function RaizQuadrada (A : Extended) : Extended;
begin
```

```
  Result := Sqrt (A);
end;

exports
  Adicao index 1,
  Subtracao index 2,
  Multiplicacao index 3,
  Divisao index 4,
  Exponenciacao index 5,
  RaizQuadrada index 6;

begin
end.
```

```
program prjCalculadora;

uses
  Forms,
  untCalculadora in 'untCalculadora.pas' {frmCalculadora};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TfrmCalculadora, frmCalculadora);
  Application.Run;
end.
```

```
unit untCalculadora;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
Forms, Dialogs,
  StdCtrls, Buttons;

type
  TOperacao = (toNenhuma, toAdicao, toSubtracao,
    toMultiplicacao, toDivisao,   toExponenciacao,
    toRaizQuadrada);

  TfrmCalculadora = class(TForm)
    gbxTecladoNumerico: TGroupBox;
    sbnOito: TSpeedButton;
    sbnSete: TSpeedButton;
```

```
    sbnNove: TSpeedButton;
    sbnSeis: TSpeedButton;
    sbnCinco: TSpeedButton;
    sbnQuatro: TSpeedButton;
    sbnUm: TSpeedButton;
    sbnDois: TSpeedButton;
    sbnTres: TSpeedButton;
    sbnPonto: TSpeedButton;
    sbnZero: TSpeedButton;
    gbxFuncoes: TGroupBox;
    sbnMultiplicacao: TSpeedButton;
    sbnSubtracao: TSpeedButton;
    sbnDivisao: TSpeedButton;
    sbnLimpar: TSpeedButton;
    sbnExponenciacao: TSpeedButton;
    sbnRaizQuadrada: TSpeedButton;
    sbnBeep: TSpeedButton;
    sbnIgual: TSpeedButton;
    sbnAdicao: TSpeedButton;
    lblVisor: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure sbnDivisaoClick(Sender: TObject);
    procedure sbnMultiplicacaoClick(Sender: TObject);
    procedure sbnSubtracaoClick(Sender: TObject);
    procedure sbnAdicaoClick(Sender: TObject);
    procedure sbnExponenciacaoClick(Sender: TObject);
    procedure sbnRaizQuadradaClick(Sender: TObject);
    procedure sbnIgualClick(Sender: TObject);
    procedure sbnBeepClick(Sender: TObject);
    procedure sbnLimparClick(Sender: TObject);
    procedure sbnPontoClick(Sender: TObject);
    procedure sbnZeroClick(Sender: TObject);
    procedure sbnUmClick(Sender: TObject);
    procedure sbnDoisClick(Sender: TObject);
    procedure sbnTresClick(Sender: TObject);
    procedure sbnQuatroClick(Sender: TObject);
    procedure sbnCincoClick(Sender: TObject);
    procedure sbnSeisClick(Sender: TObject);
    procedure sbnSeteClick(Sender: TObject);
    procedure sbnOitoClick(Sender: TObject);
    procedure sbnNoveClick(Sender: TObject);
  private
    { Private declarations }
    OperandoA,
    OperandoB : String;
    Operacao : TOperacao;
    TemPonto : Boolean;

    procedure HabilitaTecladoNumerico;
```

```
    procedure DesabilitaTecladoNumerico;
    procedure HabilitaFuncoesMatematicas;
    procedure DesabilitaFuncoesMatematicas;
    procedure HabilitaSom;
    procedure DesabilitaSom;
    procedure EmiteBeep;
    procedure LimpaVisor;
    procedure AposTeclarIgualOuOcorrerInconsistencia;
    procedure AposTeclarLimpar;
    procedure HabilitaCalculadoraParaNovaOperacao;
    function UsuarioJaEntrouComOperandoA: Boolean;
    function UsuarioJaEntrouComTodosOperandos: Boolean;
    function FaltaOperandosOuOperadorNoTotal : Boolean;
    function TemErroDeFaltaDeOperando : Boolean;
    procedure ObtemOperando;
    procedure FormaOperando(Numero: String);
    function OperandoValido(Operando: String): Boolean;
    procedure VerificaOperacao;
  public
    { Public declarations }
  end;

var
  frmCalculadora: TfrmCalculadora;

implementation

{$R *.DFM}

procedure TfrmCalculadora.FormCreate(Sender: TObject);
begin
  OperandoA := '';
  OperandoB := '';
  Operacao := toNenhuma;
  lblVisor.Caption := '';
  sbnBeep.Caption := 'Off';
  TemPonto := False;

  DecimalSeparator := '.';

  { Desabilita todo o teclado conforme especificação do
    usuário. }
  AposTeclarIgualOuOcorrerInconsistencia;
end;

procedure TfrmCalculadora.FormClose(Sender: TObject; var
  Action: TCloseAction);
begin
  Action := caFree;
  frmCalculadora := nil;
end;
```

```
{ Escolha da operações matemática. ====================== }

procedure TfrmCalculadora.sbnLimparClick(Sender: TObject);
begin
  EmiteBeep;
  LimpaVisor;

  { Habita todo o teclado conforme especificação do
usuário. }
  AposTeclarLimpar;

  { Prepara a calculadora para uma nova operação conforme
especificação do usuário. }
  HabilitaCalculadoraParaNovaOperacao;
end;

procedure TfrmCalculadora.sbnDivisaoClick(Sender: TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toDivisao;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnMultiplicacaoClick(Sender:
  TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toMultiplicacao;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnSubtracaoClick(Sender:
  TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toSubtracao;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnAdicaoClick(Sender: TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toAdicao;
```

```
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnExponenciacaoClick(Sender:
  TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toExponenciacao;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnRaizQuadradaClick(Sender:
  TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toRaizQuadrada;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnIgualClick(Sender: TObject);
type
  TAdicao = function (A, B : Extended) : Extended;
  TSubtracao = function (A, B : Extended) : Extended;
  TMultiplicacao = function (A, B : Extended) : Extended;
  TDivisao = function (A, B : Extended) : Extended;
  TExponenciacao = function (A, B : Extended) : Extended;
  TRaizQuadrada = function (A : Extended) : Extended;
var
  DLLInstance : THandle;
  wAdicao : TAdicao;
  wSubtracao : TSubtracao;
  wMultiplicacao : TMultiplicacao;
  wDivisao : TDivisao;
  wExponenciacao : TExponenciacao;
  wRaizQuadrada : TRaizQuadrada;
  Operando_A,
  Operando_B,
  Resultado : Extended;
begin
  EmiteBeep;

  if (Operacao <> toNenhuma) then
  begin
    ObtemOperando;

    if (not (FaltaOperandosOuOperadorNoTotal)) then
```

```
    begin
      { Procura pela DLL que está sendo referenciada. }
      DLLInstance := LoadLibrary ('Util');
      try
        if (DLLInstance = 0) then
           MessageDlg('A DLL "Util" não foi encontrada
             !!!', mtError, [mbOk], 0)
        else
        begin
          Operando_B := 0;
          Resultado := 0;

          Operando_A := StrToFloat (OperandoA);
          if (Operacao <> toRaizQuadrada) then
            Operando_B := StrToFloat (OperandoB);

          case (Operacao) of
            toAdicao :
            begin
              { Procurando a função ou procedimento
                desejado. }
              @wAdicao := GetProcAddress (DLLInstance,
                 'Adicao');

              if (@wAdicao <> nil) then
                Resultado := wAdicao (Operando_A,
                  Operando_B)
              else
                MessageDlg('Não foi possível encontrar a
                  função.', mtError, [mbOk], 0);
            end;

            toSubtracao :
            begin
              { Procurando a função ou procedimento
                desejado. }
              @wSubtracao := GetProcAddress (DLLInstance,
                 'Subtracao');

              if (@wSubtracao <> nil) then
                Resultado := wSubtracao (Operando_A,
                  Operando_B)
              else
                MessageDlg('Não foi possível encontrar a
                  função.', mtError, [mbOk], 0);
            end;

            toMultiplicacao :
            begin
              { Procurando a função ou procedimento
                desejado. }
```

```
    @wMultiplicacao := GetProcAddress
       (DLLInstance, 'Multiplicacao');

    if (@wMultiplicacao <> nil) then
      Resultado := wMultiplicacao (Operando_A,
        Operando_B)
    else
      MessageDlg('Não foi possível encontrar a
        função.', mtError, [mbOk], 0);
end;

toDivisao :
begin
    { Procurando a função ou procedimento
      desejado. }
    @wDivisao := GetProcAddress (DLLInstance,
       'Divisao');

    if (@wDivisao <> nil) then
      Resultado := wDivisao (Operando_A,
        Operando_B)
    else
      MessageDlg('Não foi possível encontrar a
        função.', mtError, [mbOk], 0);
end;

toExponenciacao :
begin
    { Procurando a função ou procedimento
      desejado. }
    @wExponenciacao := GetProcAddress
       (DLLInstance, 'Exponenciacao');

    if (@wExponenciacao <> nil) then
      Resultado := wExponenciacao (Operando_A,
        Operando_B)
    else
      MessageDlg('Não foi possível encontrar a
        função.', mtError, [mbOk], 0);
end;

toRaizQuadrada :
begin
    { Procurando a função ou procedimento
      desejado. }
    @wRaizQuadrada := GetProcAddress
      (DLLInstance, 'RaizQuadrada');

    if (@wRaizQuadrada <> nil) then
      Resultado := wRaizQuadrada (Operando_A)
    else
```

```
                MessageDlg('Não foi possível encontrar a
                  função.', mtError, [mbOk], 0);
            end;
          end;

          lblVisor.Caption := FloatToStr (Resultado);
        end;
      finally
        { Liberando a DLL da memória. }
        FreeLibrary (DLLInstance);
      end;
    end;
  end
  else
    MessageDlg('Você precisa selecionar uma operação antes
      de tentar totalizá-la. A ' +#13+#10+'operação será
      abortada.', mtError, [mbOK], 0);

  AposTeclarIgualOuOcorrerInconsistencia;
end;

{ Habilita e desabilita botões. ========================= }

procedure TfrmCalculadora.HabilitaTecladoNumerico;
begin
  sbnZero.Enabled := True;
  sbnUm.Enabled := True;
  sbnDois.Enabled := True;
  sbnTres.Enabled := True;
  sbnQuatro.Enabled := True;
  sbnCinco.Enabled := True;
  sbnSeis.Enabled := True;
  sbnSete.Enabled := True;
  sbnOito.Enabled := True;
  sbnNove.Enabled := True;
  sbnPonto.Enabled := True;
end;

procedure TfrmCalculadora.DesabilitaTecladoNumerico;
begin
  sbnZero.Enabled := False;
  sbnUm.Enabled := False;
  sbnDois.Enabled := False;
  sbnTres.Enabled := False;
  sbnQuatro.Enabled := False;
  sbnCinco.Enabled := False;
  sbnSeis.Enabled := False;
  sbnSete.Enabled := False;
  sbnOito.Enabled := False;
  sbnNove.Enabled := False;
  sbnPonto.Enabled := False;
```

```
end;

procedure TfrmCalculadora.HabilitaFuncoesMatematicas;
begin
  sbnDivisao.Enabled := True;
  sbnMultiplicacao.Enabled := True;
  sbnSubtracao.Enabled := True;
  sbnAdicao.Enabled := True;
  sbnExponenciacao.Enabled := True;
  sbnRaizQuadrada.Enabled := True;
  sbnIgual.Enabled := True;
end;

procedure TfrmCalculadora.DesabilitaFuncoesMatematicas;
begin
  sbnDivisao.Enabled := False;
  sbnMultiplicacao.Enabled := False;
  sbnSubtracao.Enabled := False;
  sbnAdicao.Enabled := False;
  sbnExponenciacao.Enabled := False;
  sbnRaizQuadrada.Enabled := False;
  sbnIgual.Enabled := False;
end;

procedure TfrmCalculadora.HabilitaSom;
begin
  sbnBeep.Enabled := True;
end;

procedure TfrmCalculadora.DesabilitaSom;
begin
  sbnBeep.Enabled := False;
end;

procedure TfrmCalculadora.AposTeclarLimpar;
begin
  HabilitaTecladoNumerico;
  HabilitaFuncoesMatematicas;
  HabilitaSom;
end;

procedure
  TfrmCalculadora.AposTeclarIgualOuOcorrerInconsistencia;
begin
  DesabilitaTecladoNumerico;
  DesabilitaFuncoesMatematicas;
  DesabilitaSom;

  HabilitaCalculadoraParaNovaOperacao;
end;
```

```
{ Manipulação do som do teclado da calculadora. ========= }

procedure TfrmCalculadora.EmiteBeep;
begin
  if (sbnBeep.Caption = 'On') then
    Beep;
end;

procedure TfrmCalculadora.sbnBeepClick(Sender: TObject);
begin
  EmiteBeep;

  if (TSpeedButton (Sender).Caption = 'Off') then
    TSpeedButton (Sender).Caption := 'On'
  else
    TSpeedButton (Sender).Caption := 'Off';
end;

{ Funções de reset da calculadora. ====================== }

procedure TfrmCalculadora.LimpaVisor;
begin
  lblVisor.Caption := '';
end;

procedure
TfrmCalculadora.HabilitaCalculadoraParaNovaOperacao;
begin
  Operacao := toNenhuma;
  OperandoA := '';
  OperandoB := '';
  TemPonto := False;
end;

{ Verificações. ========================================= }

function TfrmCalculadora.OperandoValido (Operando : String)
  : Boolean;
begin
  Result := True;

  try
    StrToFloat (Operando);
  except
    Result := False;
  end;
end;

function TfrmCalculadora.UsuarioJaEntrouComOperandoA :
  Boolean;
begin
```

```
  Result := OperandoValido (OperandoA);
end;

function TfrmCalculadora.UsuarioJaEntrouComTodosOperandos :
  Boolean;
begin
  if (Operacao = toRaizQuadrada) then
    Result := OperandoValido (OperandoA)
  else
    Result := (OperandoValido (OperandoA) and
      OperandoValido (OperandoB));
end;

{ Desabilita teclado numérico caso operação solicite apenas
  um operando. }
procedure TfrmCalculadora.VerificaOperacao;
begin
  if (Operacao = toRaizQuadrada) then
    DesabilitaTecladoNumerico;
end;

{ Mensagens de erro. ================================== }

function TfrmCalculadora.TemErroDeFaltaDeOperando :
  Boolean;
begin
  Result := False;

  if (not (UsuarioJaEntrouComOperandoA)) then
  begin
    Result := True;

    MessageDlg('Você precisa entrar com um número válido
      antes de escolher uma ' +#13+#10+'operação
      matemática. A operação será abortada.', mtError,
      [mbOK], 0);

    AposTeclarIgualOuOcorrerInconsistencia;
  end
  else
    VerificaOperacao;
end;

function TfrmCalculadora.FaltaOperandosOuOperadorNoTotal :
Boolean;
begin
  Result := False;

  if (not (UsuarioJaEntrouComTodosOperandos)) then
  begin
    if (Operacao = toRaizQuadrada) then
```

```
      MessageDlg('Para operações envolvendo Raiz Quadrada,
        você deverá escolher ' +#13+#10+'apenas um
        operando. Esta operação deverá ser realizada na
        seguinte ' +#13+#10+'ordem: OPERANDO e OPERADOR. A
        operação será abortada.', mtError, [mbOK], 0)
    else
      MessageDlg('O número de operandos para o tipo de
        operação matemática escolhida ' +#13+#10+'deve ser
        dois. Estes devem seguir a seguinte ordem:
        OPERANDO, ' +#13+#10+'OPERADOR e OPERANDO. A
        operação será abortada.', mtError, [mbOK], 0);

    AposTeclarIgualOuOcorrerInconsistencia;

    Result := True;
  end;
end;

{ Captura da entrada de operandos. ===================== }

procedure TfrmCalculadora.ObtemOperando;
begin
  if (OperandoA = '') then
    OperandoA := Trim (lblVisor.Caption)
  else
    if (Operacao <> toRaizQuadrada) then
      OperandoB := Trim (lblVisor.Caption);
end;

procedure TfrmCalculadora.FormaOperando (Numero : String);
begin
  if (Numero = '.') then
  begin
    if (not (TemPonto)) then
    begin
      TemPonto := True;

      if (lblVisor.Caption = '') then
        Numero := '0.';

      lblVisor.Caption := lblVisor.Caption + Numero;
    end;
  end
  else
    lblVisor.Caption := lblVisor.Caption + Numero;
end;

{ Entrada de valores. ================================== }

procedure TfrmCalculadora.sbnPontoClick(Sender: TObject);
begin
```

```
  EmiteBeep;
  FormaOperando ('.');
end;

procedure TfrmCalculadora.sbnZeroClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('0');
end;

procedure TfrmCalculadora.sbnUmClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('1');
end;

procedure TfrmCalculadora.sbnDoisClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('2');
end;

procedure TfrmCalculadora.sbnTresClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('3');
end;

procedure TfrmCalculadora.sbnQuatroClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('4');
end;

procedure TfrmCalculadora.sbnCincoClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('5');
end;

procedure TfrmCalculadora.sbnSeisClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('6');
end;

procedure TfrmCalculadora.sbnSeteClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('7');
end;
```

```
procedure TfrmCalculadora.sbnOitoClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('8');
end;

procedure TfrmCalculadora.sbnNoveClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('9');
end;

end.
```

4) Construção de Objetos

- Introdução
  - Pra que serve ?
  - Onde e quando usar ?
  - Encapsulamento
  - Herança
  - Polimorfismo

Exemplo de Software

1) Exemplo_5\OO: prjCalculadora.exe – Software de calculadora:

```
unit untUtil;

interface

type
  TCalculo = class
  private
    FOperando_A,
    FOperando_B,
    FResultado : Extended;
  public
    constructor Create;
    destructor Destroy; override;
    procedure Adicao;
    procedure Subtracao;
    procedure Multiplicacao;
    procedure Divisao;
    procedure Exponenciacao;
    procedure RaizQuadrada;

    property Operando_A : Extended read FOperando_A write
      FOperando_A;
    property Operando_B : Extended read FOperando_B write
      FOperando_B;
    property Resultado : Extended read FResultado write
      FResultado;
  end;

implementation

constructor TCalculo.Create;
begin
  inherited;
  FOperando_A := 0;
```

```
  FOperando_B := 0;
  FResultado := 0;
end;

destructor TCalculo.Destroy;
begin
  inherited;
end;

procedure TCalculo.Adicao;
begin
  FResultado := FOperando_A + FOperando_B;
end;

procedure TCalculo.Subtracao;
begin
  FResultado := FOperando_A - FOperando_B;
end;

procedure TCalculo.Multiplicacao;
begin
  FResultado := FOperando_A * FOperando_B;
end;

procedure TCalculo.Divisao;
begin
  FResultado := FOperando_A / FOperando_B;
end;

procedure TCalculo.Exponenciacao;
begin
  FResultado := Exp (FOperando_B * Ln (FOperando_A));
end;

procedure TCalculo.RaizQuadrada;
begin
  FResultado := Sqrt (FOperando_A);
end;

end.
```

```
program prjCalculadora;

uses
  Forms,
  untCalculadora in 'untCalculadora.pas' {frmCalculadora},
  untUtil in 'untUtil.pas';

{$R *.RES}
```

```
begin
  Application.Initialize;
  Application.CreateForm(TfrmCalculadora, frmCalculadora);
  Application.Run;
end.
```

```
unit untCalculadora;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Buttons, untUtil;

type
  TOperacao = (toNenhuma, toAdicao, toSubtracao,
    toMultiplicacao, toDivisao, toExponenciacao,
    toRaizQuadrada);

  TfrmCalculadora = class(TForm)
    gbxTecladoNumerico: TGroupBox;
    sbnOito: TSpeedButton;
    sbnSete: TSpeedButton;
    sbnNove: TSpeedButton;
    sbnSeis: TSpeedButton;
    sbnCinco: TSpeedButton;
    sbnQuatro: TSpeedButton;
    sbnUm: TSpeedButton;
    sbnDois: TSpeedButton;
    sbnTres: TSpeedButton;
    sbnPonto: TSpeedButton;
    sbnZero: TSpeedButton;
    gbxFuncoes: TGroupBox;
    sbnMultiplicacao: TSpeedButton;
    sbnSubtracao: TSpeedButton;
    sbnDivisao: TSpeedButton;
    sbnLimpar: TSpeedButton;
    sbnExponenciacao: TSpeedButton;
    sbnRaizQuadrada: TSpeedButton;
    sbnBeep: TSpeedButton;
    sbnIgual: TSpeedButton;
    sbnAdicao: TSpeedButton;
    lblVisor: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure sbnDivisaoClick(Sender: TObject);
    procedure sbnMultiplicacaoClick(Sender: TObject);
    procedure sbnSubtracaoClick(Sender: TObject);
```

```
    procedure sbnAdicaoClick(Sender: TObject);
    procedure sbnExponenciacaoClick(Sender: TObject);
    procedure sbnRaizQuadradaClick(Sender: TObject);
    procedure sbnIgualClick(Sender: TObject);
    procedure sbnBeepClick(Sender: TObject);
    procedure sbnLimparClick(Sender: TObject);
    procedure sbnPontoClick(Sender: TObject);
    procedure sbnZeroClick(Sender: TObject);
    procedure sbnUmClick(Sender: TObject);
    procedure sbnDoisClick(Sender: TObject);
    procedure sbnTresClick(Sender: TObject);
    procedure sbnQuatroClick(Sender: TObject);
    procedure sbnCincoClick(Sender: TObject);
    procedure sbnSeisClick(Sender: TObject);
    procedure sbnSeteClick(Sender: TObject);
    procedure sbnOitoClick(Sender: TObject);
    procedure sbnNoveClick(Sender: TObject);
  private
    { Private declarations }
    OperandoA,
    OperandoB : String;
    Operacao : TOperacao;
    TemPonto : Boolean;

    { Criação do objeto. }
    Calculo : TCalculo;

    procedure HabilitaTecladoNumerico;
    procedure DesabilitaTecladoNumerico;
    procedure HabilitaFuncoesMatematicas;
    procedure DesabilitaFuncoesMatematicas;
    procedure HabilitaSom;
    procedure DesabilitaSom;
    procedure EmiteBeep;
    procedure LimpaVisor;
    procedure AposTeclarIgualOuOcorrerInconsistencia;
    procedure AposTeclarLimpar;
    procedure HabilitaCalculadoraParaNovaOperacao;
    function UsuarioJaEntrouComOperandoA: Boolean;
    function UsuarioJaEntrouComTodosOperandos: Boolean;
    function FaltaOperandosOuOperadorNoTotal : Boolean;
    function TemErroDeFaltaDeOperando : Boolean;
    procedure ObtemOperando;
    procedure FormaOperando(Numero: String);
    function OperandoValido(Operando: String): Boolean;
    procedure VerificaOperacao;
  public
    { Public declarations }
  end;

var
```

```
  frmCalculadora: TfrmCalculadora;

implementation

{$R *.DFM}

procedure TfrmCalculadora.FormCreate(Sender: TObject);
begin
  { Instânciação do objeto. }
  Calculo := TCalculo.Create;

  Operacao := toNenhuma;
  lblVisor.Caption := '';
  sbnBeep.Caption := 'Off';
  TemPonto := False;

  DecimalSeparator := '.';

  { Desabilita todo o teclado conforme especificação do
    usuário. }
  AposTeclarIgualOuOcorrerInconsistencia;
end;

procedure TfrmCalculadora.FormClose(Sender: TObject; var
  Action: TCloseAction);
begin
  Calculo.Free;

  Action := caFree;
  frmCalculadora := nil;
end;

{ Escolha da operações matemática. ====================== }

procedure TfrmCalculadora.sbnLimparClick(Sender: TObject);
begin
  EmiteBeep;
  LimpaVisor;

  { Habita todo o teclado conforme especificação do
    usuário. }
  AposTeclarLimpar;

  { Prepara a calculadora para uma nova operação conforme
    especificação do usuário. }
  HabilitaCalculadoraParaNovaOperacao;
end;

procedure TfrmCalculadora.sbnDivisaoClick(Sender: TObject);
begin
  EmiteBeep;
```

```
  ObtemOperando;
  Operacao := toDivisao;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnMultiplicacaoClick(Sender:
  TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toMultiplicacao;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnSubtracaoClick(Sender:
  TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toSubtracao;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnAdicaoClick(Sender: TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toAdicao;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnExponenciacaoClick(Sender:
  TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toExponenciacao;
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnRaizQuadradaClick(Sender:
  TObject);
begin
  EmiteBeep;
  ObtemOperando;
  Operacao := toRaizQuadrada;
```

```
  if (not (TemErroDeFaltaDeOperando)) then
    LimpaVisor;
end;

procedure TfrmCalculadora.sbnIgualClick(Sender: TObject);
begin
  EmiteBeep;

  if (Operacao <> toNenhuma) then
  begin
    ObtemOperando;

    if (not (FaltaOperandosOuOperadorNoTotal)) then
    begin
      Calculo.Operando_B := 0;
      Calculo.Resultado := 0;

      Calculo.Operando_A := StrToFloat (OperandoA);
      if (Operacao <> toRaizQuadrada) then
        Calculo.Operando_B := StrToFloat (OperandoB);

      case (Operacao) of
        toAdicao :
          Calculo.Adicao;

        toSubtracao :
          Calculo.Subtracao;

        toMultiplicacao :
          Calculo.Multiplicacao;

        toDivisao :
          Calculo.Divisao;

        toExponenciacao :
          Calculo.Exponenciacao;

        toRaizQuadrada :
          Calculo.RaizQuadrada;
      end;

      lblVisor.Caption := FloatToStr (Calculo.Resultado);
    end;
  end
  else
    MessageDlg('Você precisa selecionar uma operação antes
      de tentar totalizá-la. A ' +#13+#10+'operação será
     abortada.', mtError, [mbOK], 0);

  AposTeclarIgualOuOcorrerInconsistencia;
end;
```

```
{ Habilita e desabilita botões. ======================== }

procedure TfrmCalculadora.HabilitaTecladoNumerico;
begin
  sbnZero.Enabled := True;
  sbnUm.Enabled := True;
  sbnDois.Enabled := True;
  sbnTres.Enabled := True;
  sbnQuatro.Enabled := True;
  sbnCinco.Enabled := True;
  sbnSeis.Enabled := True;
  sbnSete.Enabled := True;
  sbnOito.Enabled := True;
  sbnNove.Enabled := True;
  sbnPonto.Enabled := True;
end;

procedure TfrmCalculadora.DesabilitaTecladoNumerico;
begin
  sbnZero.Enabled := False;
  sbnUm.Enabled := False;
  sbnDois.Enabled := False;
  sbnTres.Enabled := False;
  sbnQuatro.Enabled := False;
  sbnCinco.Enabled := False;
  sbnSeis.Enabled := False;
  sbnSete.Enabled := False;
  sbnOito.Enabled := False;
  sbnNove.Enabled := False;
  sbnPonto.Enabled := False;
end;

procedure TfrmCalculadora.HabilitaFuncoesMatematicas;
begin
  sbnDivisao.Enabled := True;
  sbnMultiplicacao.Enabled := True;
  sbnSubtracao.Enabled := True;
  sbnAdicao.Enabled := True;
  sbnExponenciacao.Enabled := True;
  sbnRaizQuadrada.Enabled := True;
  sbnIgual.Enabled := True;
end;

procedure TfrmCalculadora.DesabilitaFuncoesMatematicas;
begin
  sbnDivisao.Enabled := False;
  sbnMultiplicacao.Enabled := False;
  sbnSubtracao.Enabled := False;
  sbnAdicao.Enabled := False;
  sbnExponenciacao.Enabled := False;
```

```
  sbnRaizQuadrada.Enabled := False;
  sbnIgual.Enabled := False;
end;

procedure TfrmCalculadora.HabilitaSom;
begin
  sbnBeep.Enabled := True;
end;

procedure TfrmCalculadora.DesabilitaSom;
begin
  sbnBeep.Enabled := False;
end;

procedure TfrmCalculadora.AposTeclarLimpar;
begin
  HabilitaTecladoNumerico;
  HabilitaFuncoesMatematicas;
  HabilitaSom;
end;

procedure
  TfrmCalculadora.AposTeclarIgualOuOcorrerInconsistencia;
begin
  DesabilitaTecladoNumerico;
  DesabilitaFuncoesMatematicas;
  DesabilitaSom;

  HabilitaCalculadoraParaNovaOperacao;
end;

{ Manipulação do som do teclado da calculadora. ========= }

procedure TfrmCalculadora.EmiteBeep;
begin
  if (sbnBeep.Caption = 'On') then
    Beep;
end;

procedure TfrmCalculadora.sbnBeepClick(Sender: TObject);
begin
  EmiteBeep;

  if (TSpeedButton (Sender).Caption = 'Off') then
    TSpeedButton (Sender).Caption := 'On'
  else
    TSpeedButton (Sender).Caption := 'Off';
end;

{ Funções de reset da calculadora. ====================== }
```

```
procedure TfrmCalculadora.LimpaVisor;
begin
  lblVisor.Caption := '';
end;

procedure
  TfrmCalculadora.HabilitaCalculadoraParaNovaOperacao;
begin
  Operacao := toNenhuma;
  OperandoA := '';
  OperandoB := '';
  TemPonto := False;
end;

{ Verificações. ======================================== }

function TfrmCalculadora.OperandoValido (Operando : String)
  : Boolean;
begin
  Result := True;

  try
    StrToFloat (Operando);
  except
    Result := False;
  end;
end;

function TfrmCalculadora.UsuarioJaEntrouComOperandoA :
  Boolean;
begin
  Result := OperandoValido (OperandoA);
end;

function TfrmCalculadora.UsuarioJaEntrouComTodosOperandos :
  Boolean;
begin
  if (Operacao = toRaizQuadrada) then
    Result := OperandoValido (OperandoA)
  else
    Result := (OperandoValido (OperandoA) and
      OperandoValido (OperandoB));
end;

{ Desabilita teclado numérico caso operação solicite apenas
  um operando. }
procedure TfrmCalculadora.VerificaOperacao;
begin
  if (Operacao = toRaizQuadrada) then
    DesabilitaTecladoNumerico;
end;
```

```
{ Mensagens de erro. ================================== }

function TfrmCalculadora.TemErroDeFaltaDeOperando :
  Boolean;
begin
  Result := False;

  if (not (UsuarioJaEntrouComOperandoA)) then
  begin
    Result := True;

    MessageDlg('Você precisa entrar com um número válido
      antes de escolher uma ' +#13+#10+'operação
      matemática. A operação será abortada.', mtError,
      [mbOK], 0);

    AposTeclarIgualOuOcorrerInconsistencia;
  end
  else
    VerificaOperacao;
end;

function TfrmCalculadora.FaltaOperandosOuOperadorNoTotal :
  Boolean;
begin
  Result := False;

  if (not (UsuarioJaEntrouComTodosOperandos)) then
  begin
    if (Operacao = toRaizQuadrada) then
      MessageDlg('Para operações envolvendo Raiz Quadrada,
        você deverá escolher ' +#13+#10+'apenas um
        operando. Esta operação deverá ser realizada na
        seguinte ' +#13+#10+'ordem: OPERANDO e OPERADOR. A
        operação será abortada.', mtError, [mbOK], 0)
    else
      MessageDlg('O número de operandos para o tipo de
        operação matemática escolhida ' +#13+#10+'deve ser
        dois. Estes devem seguir a seguinte ordem:
        OPERANDO, ' +#13+#10+'OPERADOR e OPERANDO. A
        operação será abortada.', mtError, [mbOK], 0);

    AposTeclarIgualOuOcorrerInconsistencia;

    Result := True;
  end;
end;

{ Captura da entrada de operandos. ==================== }
```

```
procedure TfrmCalculadora.ObtemOperando;
begin
  if (OperandoA = '') then
    OperandoA := Trim (lblVisor.Caption)
  else
    if (Operacao <> toRaizQuadrada) then
      OperandoB := Trim (lblVisor.Caption);
end;

procedure TfrmCalculadora.FormaOperando (Numero : String);
begin
  if (Numero = '.') then
  begin
    if (not (TemPonto)) then
    begin
      TemPonto := True;

      if (lblVisor.Caption = '') then
        Numero := '0.';

      lblVisor.Caption := lblVisor.Caption + Numero;
    end;
  end
  else
    lblVisor.Caption := lblVisor.Caption + Numero;
end;

{ Entrada de valores. ================================== }

procedure TfrmCalculadora.sbnPontoClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('.');
end;

procedure TfrmCalculadora.sbnZeroClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('0');
end;

procedure TfrmCalculadora.sbnUmClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('1');
end;

procedure TfrmCalculadora.sbnDoisClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('2');
```

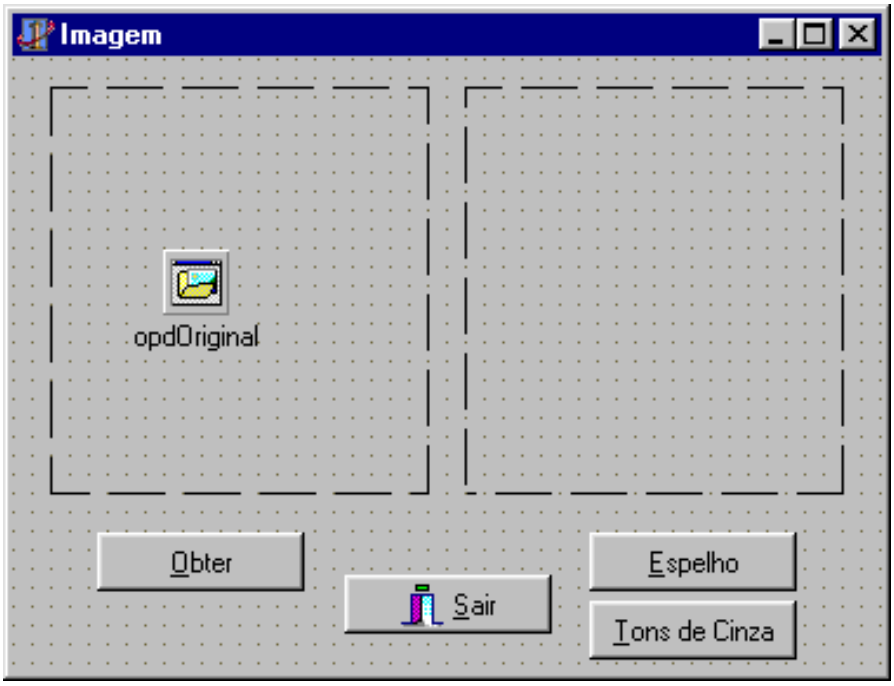
```
end;

procedure TfrmCalculadora.sbnTresClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('3');
end;

procedure TfrmCalculadora.sbnQuatroClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('4');
end;

procedure TfrmCalculadora.sbnCincoClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('5');
end;

procedure TfrmCalculadora.sbnSeisClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('6');
end;

procedure TfrmCalculadora.sbnSeteClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('7');
end;

procedure TfrmCalculadora.sbnOitoClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('8');
end;

procedure TfrmCalculadora.sbnNoveClick(Sender: TObject);
begin
  EmiteBeep;
  FormaOperando ('9');
end;

end.
```

5) Manipulação de Imagens

- Intrdoução Conceito de Cores (R, G, B)
  - Canhões de cores
  - Varredura do monitor
  - O que se é capaz de fazer

Exemplo de Software

1) Exemplo 6: prjImagem.exe – Software de Manipulação de Imagens:




```
program prjImagem;

uses
  Forms,
  untImagem in 'untImagem.pas' {frmImagem};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TfrmImagem, frmImagem);
  Application.Run;
end.
```

```
unit untImagem;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, ExtCtrls, ExtDlgs, StdCtrls, Buttons;

type
  TfrmImagem = class(TForm)
    imgOriginal: TImage;
    imgResultado: TImage;
    opdOriginal: TOpenPictureDialog;
    bbnObter: TBitBtn;
    bbnEspelho: TBitBtn;
    bbnSair: TBitBtn;
    bbnTonsCinza: TBitBtn;
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure bbnObterClick(Sender: TObject);
    procedure bbnEspelhoClick(Sender: TObject);
    procedure bbnSairClick(Sender: TObject);
    procedure bbnTonsCinzaClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmImagem: TfrmImagem;

implementation

{$R *.DFM}

procedure TfrmImagem.FormClose(Sender: TObject; var Action:
  TCloseAction);
begin
  Action := caFree;
  frmImagem := nil;
end;

procedure TfrmImagem.bbnObterClick(Sender: TObject);
begin
  opdOriginal.Execute;
  if (opdOriginal.FileName <> '')then
    imgOriginal.Picture.LoadFromFile (Trim
      (opdOriginal.FileName))
  else
```

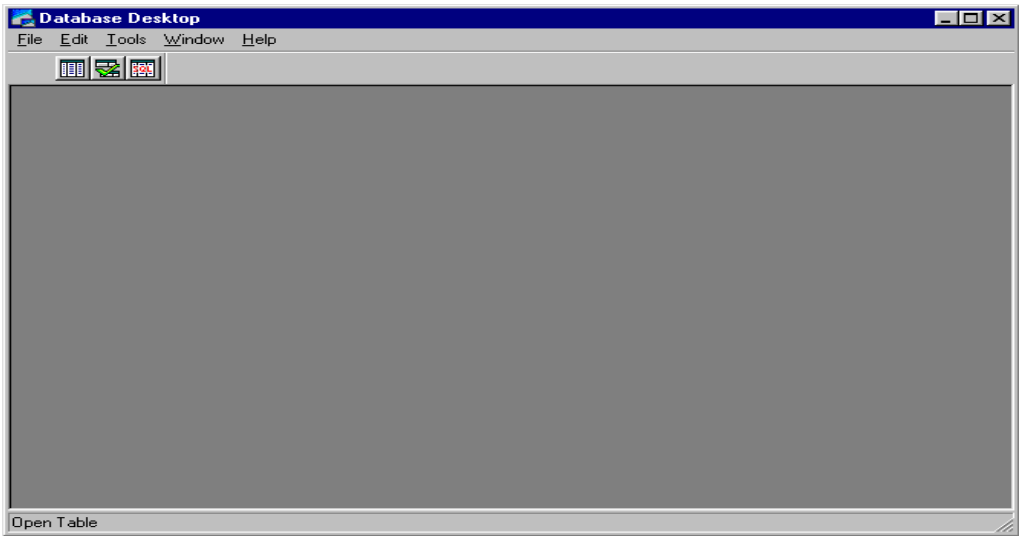
```
    MessageDlg ('Entre com um caminho válido para uma
      figura.', mtInformation, [mbOk], 0);
end;

procedure TfrmImagem.bbnEspelhoClick(Sender: TObject);
var
  Altura,
  Largura,
  X,
  Y : Integer;
begin
  Largura := imgOriginal.Width;
  Altura := imgOriginal.Height;

  for Y := 0 to Altura - 1 do
  begin
    for X := 0 to Largura - 1 do
      imgResultado.Canvas.Pixels [Largura - (X + 1), Y] :=
        imgOriginal.Canvas.Pixels [X, Y];
    Application.ProcessMessages;
  end;
end;

procedure TfrmImagem.bbnSairClick(Sender: TObject);
begin
  Close;
end;

procedure TfrmImagem.bbnTonsCinzaClick(Sender: TObject);
var
  I,
  J,
  R,
  G,
  B,
  NC,
  Cor : Integer;
begin
  for I := 0 to imgOriginal.Picture.Width - 1 do
  begin
    for j := 0 to imgOriginal.Picture.Height - 1 do
      if (imgOriginal.Canvas.Pixels [I, J] <> -1) then
      begin
        Cor := imgOriginal.Canvas.Pixels [I, J];
        B := (Cor and $0000FF);
        G := (Cor and $00FF00) shr 8;
        R := (Cor and $FF0000) shr 16;
        NC := Trunc ((R + G + B) / 3);
        imgResultado.Canvas.Pixels [I, J] := NC + (NC shl
          8) + (NC shl 16);
      end;
```

```
    Application.ProcessMessages;
  end;
end;

end.
```

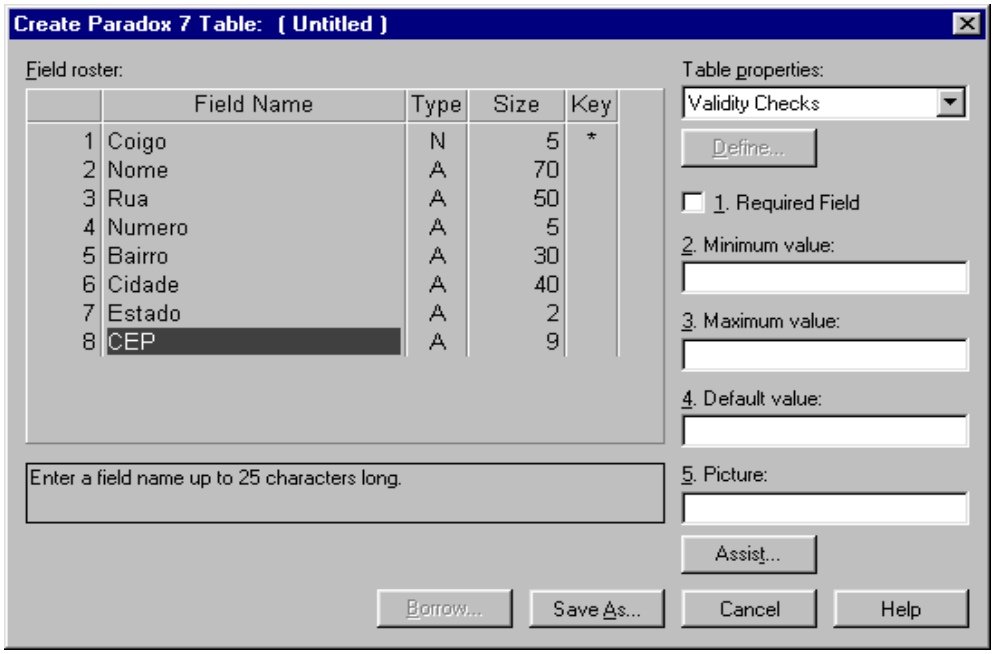
6) Banco de Dados

- Usando o “Database Desktop”
  - Criando e Modificando tabelas
  - Working Directory
- Usando o “SQL Explorer”

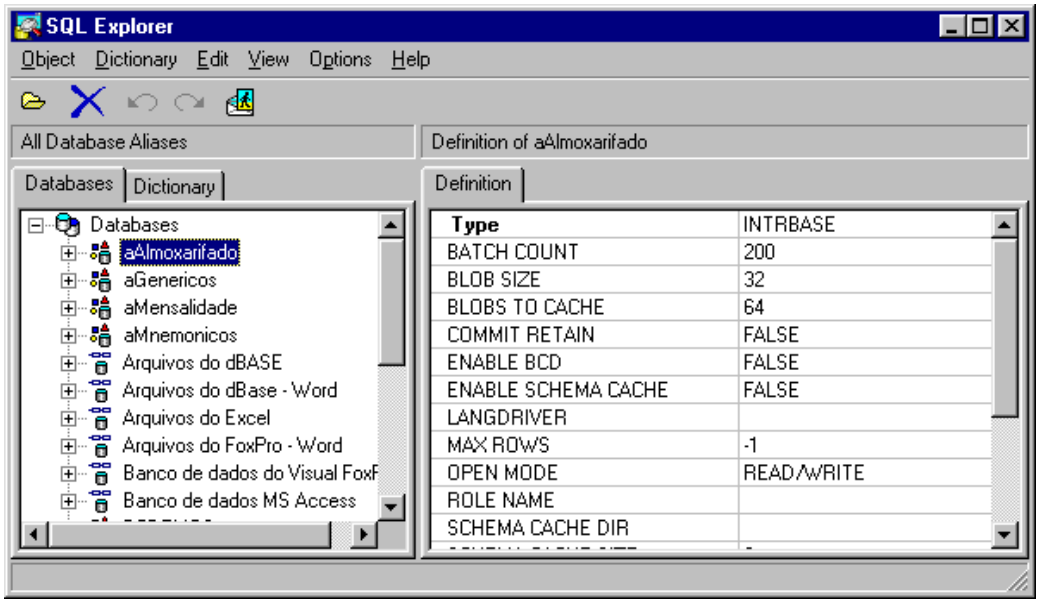
- **Usiando o Database Desktop:**



- **Criando e Modificando a TABELAS:**
  - Paradox 7
  - Nome:
    - Endereco
  - Campos:
    - Código **(Primary Key)** (Integer)
    - Nome (Alpha 70)
    - Rua (Alpha 50)
    - Numero (Alpha 5)
    - Bairro (Alpha 30)
    - Cidade (Alpha 40)
    - Estado (Alpha 2)
    - CPE (Alpha 9)
  - Extensões dos arquivos criados para cada tabela Paradox
    - .db, .PX, .VAL



- **Usando o "SQL Explorer"**
  - Criando o ALIAS
    - Nome:
      - aEndereco
  - Path



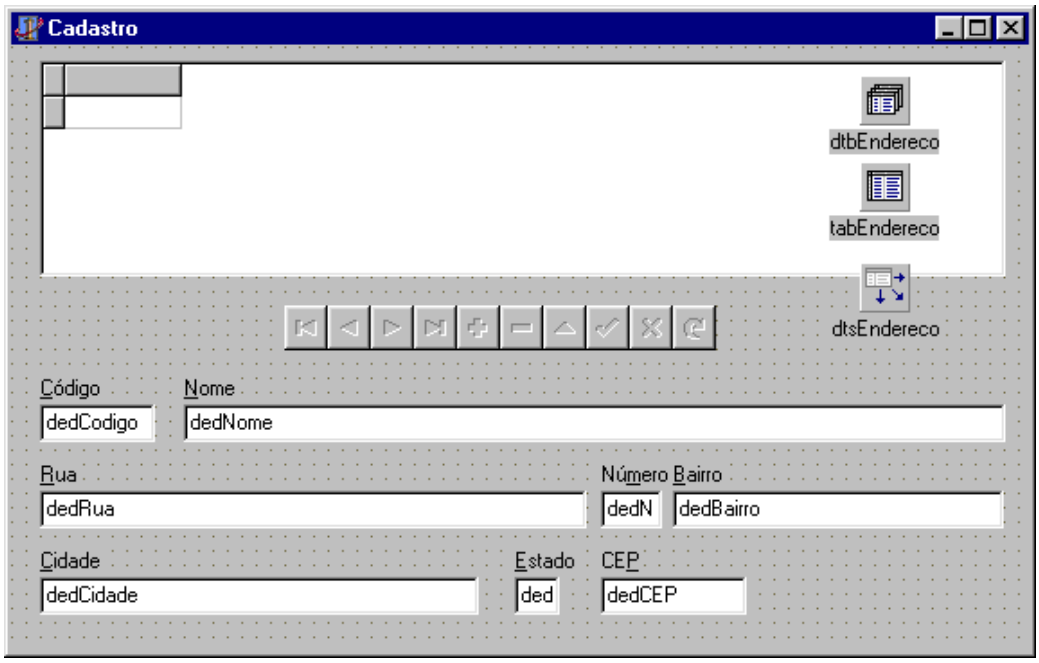
Cadastros Simples

- **Controle através do componente "DBNavigator".**
- **Controle através de botões independentes.**
- **Utilização de TDataBase**

Exemplo de Software

1) Exemplo_7\TabelaSimples\Cad1: prjCadastro.exe – Software de Cadastro:

- **Controle através do componente "DBNavigator".**




```
program prjCadastro;

uses
  Forms,
  untCadastro in 'untCadastro.pas' {frmCadastro};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TfrmCadastro, frmCadastro);
  Application.Run;
```

```
end.
```

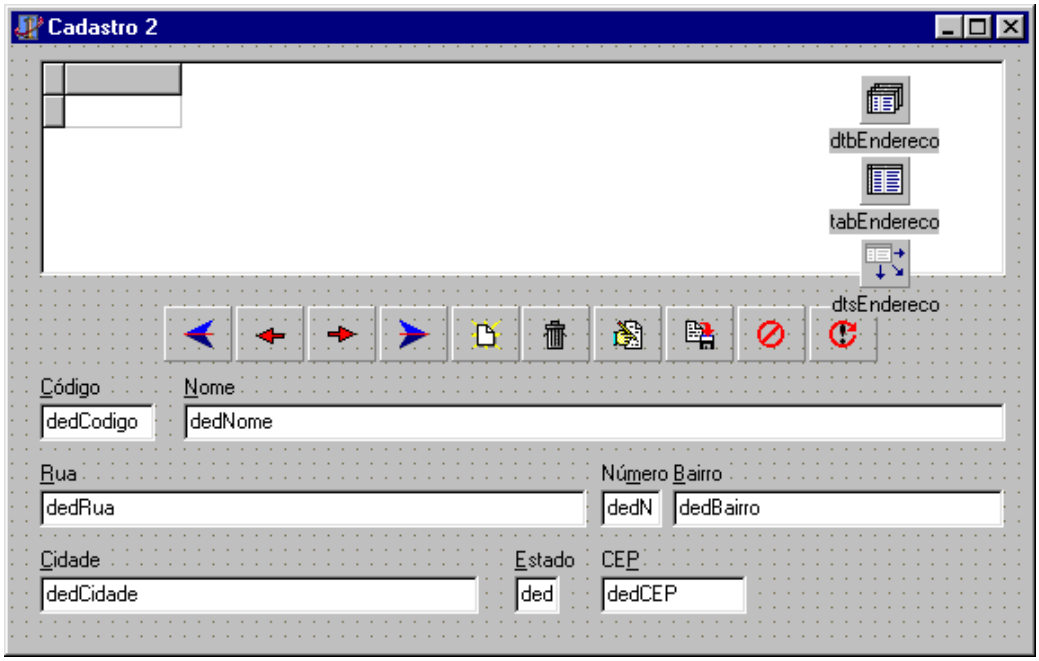
```
unit untCadastro;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, Db, DBTables, StdCtrls, Mask, DBCtrls,
  ExtCtrls, Grids, DBGrids;

type
  TfrmCadastro = class(TForm)
    dtbEndereco: TDatabase;
    tabEndereco: TTable;
    dtsEndereco: TDataSource;
    tabEnderecoCoigo: TFloatField;
    tabEnderecoNome: TStringField;
    tabEnderecoRua: TStringField;
    tabEnderecoNumero: TStringField;
    tabEnderecoBairro: TStringField;
    tabEnderecoCidade: TStringField;
    tabEnderecoEstado: TStringField;
    tabEnderecoCEP: TStringField;
    dgrEndereco: TDBGrid;
    dnvEndereco: TDBNavigator;
    Label1: TLabel;
    dedCodigo: TDBEdit;
    Label2: TLabel;
    dedNome: TDBEdit;
    Label3: TLabel;
    dedRua: TDBEdit;
    Label4: TLabel;
    dedNumero: TDBEdit;
    Label5: TLabel;
    dedBairro: TDBEdit;
    Label6: TLabel;
    dedCidade: TDBEdit;
    Label7: TLabel;
    dedEstado: TDBEdit;
    Label8: TLabel;
    dedCEP: TDBEdit;
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure FormCreate(Sender: TObject);
    procedure dnvEnderecoClick(Sender: TObject; Button:
      TNavigateBtn);
  private
    { Private declarations }
  public
```

```
    { Public declarations }
  end;

var
  frmCadastro: TfrmCadastro;

implementation

{$R *.DFM}

procedure TfrmCadastro.FormCreate(Sender: TObject);
begin
  dtbEndereco.Open;
  tabEndereco.Open;
end;

procedure TfrmCadastro.FormClose(Sender: TObject; var
  Action: TCloseAction);
begin
  tabEndereco.Close;
  dtbEndereco.Close;

  Action := caFree;
  frmCadastro := nil;
end;

procedure TfrmCadastro.dnvEnderecoClick(Sender: TObject;
  Button: TNavigateBtn);
begin
  case (Button) of
    nbInsert :
      if (dedCodigo.CanFocus) then
        dedCodigo.SetFocus;
  end;
end;

end.
```

2) Exemplo_7\TabelaSimples\Cad2: prjCadastro2.exe – Software de Cadastro:

- **Controle através de botões independentes.**

- Substituição do componente **DBNavigator** por botões simples.
- Programação de cada botão.
- Programação do evento “OnStateChange” do TdataSource.




```
program prjCadastro2;

uses
  Forms,
  untCadastro2 in 'untCadastro2.pas' {frmCadastro};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TfrmCadastro2, frmCadastro2);
  Application.Run;
end.
```

```
unit untCadastro2;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, Db, DBTables, StdCtrls, Mask, DBCtrls,
  ExtCtrls, Grids, DBGrids, Buttons;

type
  TfrmCadastro2 = class(TForm)
```

```
    dtbEndereco: TDatabase;
    tabEndereco: TTable;
    dtsEndereco: TDataSource;
    tabEnderecoCoigo: TFloatField;
    tabEnderecoNome: TStringField;
    tabEnderecoRua: TStringField;
    tabEnderecoNumero: TStringField;
    tabEnderecoBairro: TStringField;
    tabEnderecoCidade: TStringField;
    tabEnderecoEstado: TStringField;
    tabEnderecoCEP: TStringField;
    dgrEndereco: TDBGrid;
    Label1: TLabel;
    dedCodigo: TDBEdit;
    Label2: TLabel;
    dedNome: TDBEdit;
    Label3: TLabel;
    dedRua: TDBEdit;
    Label4: TLabel;
    dedNumero: TDBEdit;
    Label5: TLabel;
    dedBairro: TDBEdit;
    Label6: TLabel;
    dedCidade: TDBEdit;
    Label7: TLabel;
    dedEstado: TDBEdit;
    Label8: TLabel;
    dedCEP: TDBEdit;
    sbnPrimeiro: TSpeedButton;
    sbnAnterior: TSpeedButton;
    sbnProximo: TSpeedButton;
    sbnUltimo: TSpeedButton;
    sbnNovo: TSpeedButton;
    sbnGravar: TSpeedButton;
    sbnExcluir: TSpeedButton;
    sbnEditar: TSpeedButton;
    sbnCancelar: TSpeedButton;
    sbnRefresh: TSpeedButton;
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure FormCreate(Sender: TObject);
    procedure dnvEnderecoClick(Sender: TObject; Button:
      TNavigateBtn);
    procedure sbnPrimeiroClick(Sender: TObject);
    procedure sbnAnteriorClick(Sender: TObject);
    procedure sbnProximoClick(Sender: TObject);
    procedure sbnUltimoClick(Sender: TObject);
    procedure sbnNovoClick(Sender: TObject);
    procedure sbnExcluirClick(Sender: TObject);
    procedure sbnEditarClick(Sender: TObject);
    procedure sbnGravarClick(Sender: TObject);
```

```
    procedure sbnCancelarClick(Sender: TObject);
    procedure sbnRefreshClick(Sender: TObject);
    procedure dtsEnderecoStateChange(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmCadastro2: TfrmCadastro2;

implementation

{$R *.DFM}

procedure TfrmCadastro2.FormCreate(Sender: TObject);
begin
  dtbEndereco.Open;
  tabEndereco.Open;
end;

procedure TfrmCadastro2.FormClose(Sender: TObject; var
  Action: TCloseAction);
begin
  tabEndereco.Close;
  dtbEndereco.Close;

  Action := caFree;
  frmCadastro2 := nil;
end;

procedure TfrmCadastro2.sbnPrimeiroClick(Sender: TObject);
begin
  tabEndereco.First;
end;

procedure TfrmCadastro2.sbnAnteriorClick(Sender: TObject);
begin
  tabEndereco.Prior;
end;

procedure TfrmCadastro2.sbnProximoClick(Sender: TObject);
begin
  tabEndereco.Next;
end;

procedure TfrmCadastro2.sbnUltimoClick(Sender: TObject);
begin
  tabEndereco.Last;
end;
```

```
procedure TfrmCadastro2.sbnNovoClick(Sender: TObject);
begin
  tabEndereco.Append;
  if (dedCodigo.CanFocus) then
    dedCodigo.SetFocus;
end;

procedure TfrmCadastro2.sbnExcluirClick(Sender: TObject);
begin
  tabEndereco.Delete;
end;

procedure TfrmCadastro2.sbnEditarClick(Sender: TObject);
begin
  tabEndereco.Edit;
end;

procedure TfrmCadastro2.sbnGravarClick(Sender: TObject);
begin
  tabEndereco.Post;
end;

procedure TfrmCadastro2.sbnCancelarClick(Sender: TObject);
begin
  tabEndereco.Cancel;
end;

procedure TfrmCadastro2.sbnRefreshClick(Sender: TObject);
begin
  tabEndereco.Refresh;
end;

procedure TfrmCadastro2.dtsEnderecoStateChange(Sender:
  TObject);
begin
  case (dtsEndereco.DataSet.State) of
    dsBrowse :
    begin
      if (dtsEndereco.DataSet.IsEmpty) then
      begin
        sbnNovo.Enabled := True;
        sbnCancelar.Enabled := False;
        sbnExcluir.Enabled := False;
        sbnGravar.Enabled := False;
        sbnAnterior.Enabled := False;
        sbnProximo.Enabled := False;
        sbnPrimeiro.Enabled := False;
        sbnUltimo.Enabled := False;
      end
      else
```

```
      begin
        sbnNovo.Enabled := True;
        sbnCancelar.Enabled := False;
        sbnExcluir.Enabled := True;
        sbnGravar.Enabled := False;
        sbnAnterior.Enabled := True;
        sbnProximo.Enabled := True;
        sbnPrimeiro.Enabled := True;
        sbnUltimo.Enabled := True;
      end;
    end;

    dsEdit :
    begin
      sbnNovo.Enabled := False;
      sbnCancelar.Enabled := True;
      sbnExcluir.Enabled := False;
      sbnGravar.Enabled := True;
      sbnAnterior.Enabled := False;
      sbnProximo.Enabled := False;
      sbnPrimeiro.Enabled := False;
      sbnUltimo.Enabled := False;
    end;

    dsInsert :
    begin
      sbnNovo.Enabled := False;
      sbnCancelar.Enabled := True;
      sbnExcluir.Enabled := False;
      sbnGravar.Enabled := True;
      sbnAnterior.Enabled := False;
      sbnProximo.Enabled := False;
      sbnPrimeiro.Enabled := False;
      sbnUltimo.Enabled := False;
    end;

    dsInactive :
    begin
      sbnNovo.Enabled := False;
      sbnCancelar.Enabled := False;
      sbnExcluir.Enabled := False;
      sbnGravar.Enabled := False;
      sbnAnterior.Enabled := False;
      sbnProximo.Enabled := False;
      sbnPrimeiro.Enabled := False;
      sbnUltimo.Enabled := False;
    end;
  end;
end;

end.
```

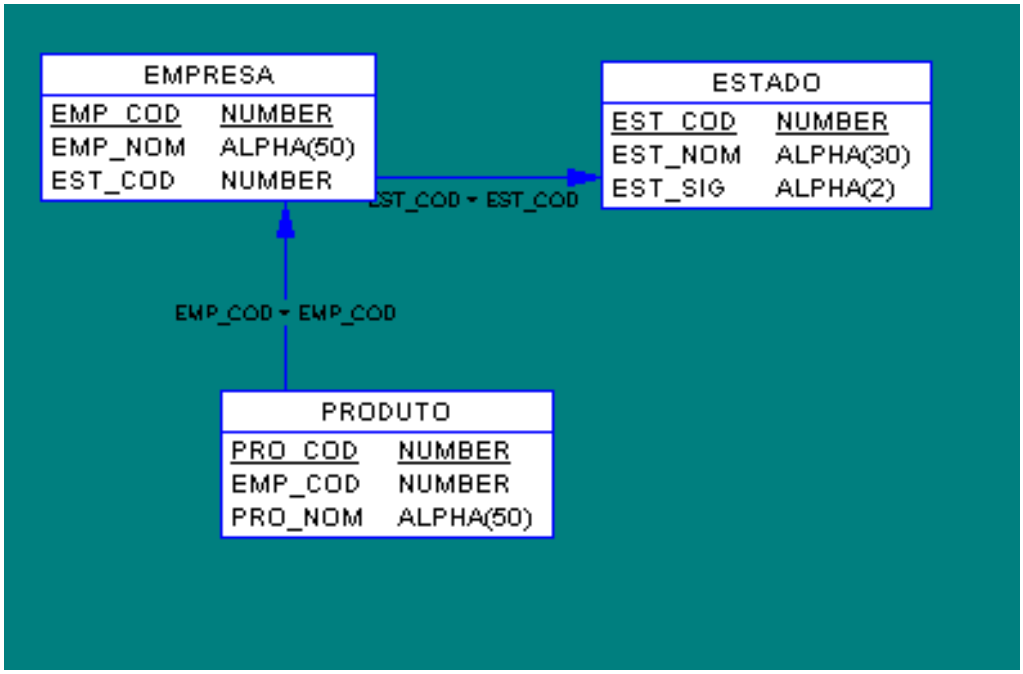
Cadastros Complexos

- **Utilização de TDataBase**
- **Utilização de Menu**
- **Visão Mestre-detalhe**
- **Diversos Cadastros**
- **Formulários MDI**

Exemplo de Software

2) Exemplo_7\MestreDetalhe: prjMestreDetalhe.exe – Software de Cadastro usando formulários MDI:

TABELAS PARADOX:



- **Estado**
  - Est_Cod Number **(Primary Key)**
  - Est_Nom Alpha 30
  - Est_Sig Alpha 02
- **Empresa**
  - Emp_Cód Number **(Primary Key)**
  - Emp_Nom Alpha 50

    - Est_Cod Number **(Foreign Key)**
- **Produto**
    - Pro_Cod Number **(Primary Key)**
    - Pro_Nom Alpha 50
    - Emp_Cod Number **(Foreign Key)**

Criar ALIAS: aEmpresasProdutos

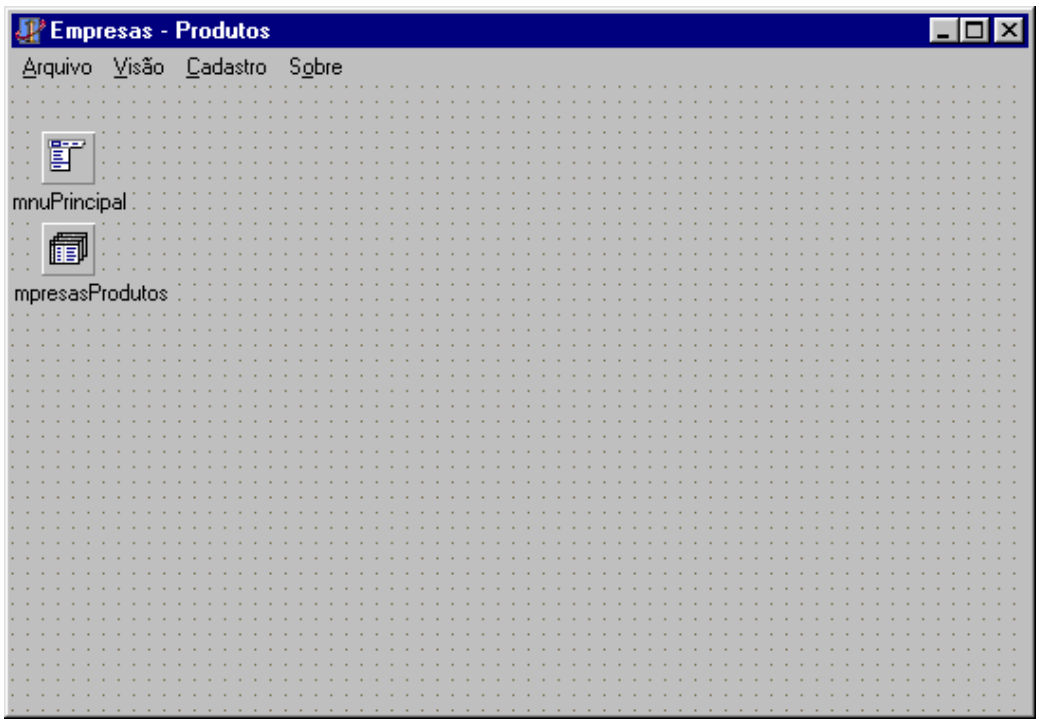
Propriedades de destaque no formulário Principal:

- Name = frmPrincipal
- WindowState = wsMaximized
- FormStyle = fsMDIForm
- Position = poScreenCenter

Propridades de destaque nos formulários Filhos:

- FormStyle = fsMDIChild
- Position = poOwnerFormCenter

```
program prjMestreDetalhe;

uses
  Forms,
  untPrincipal in 'untPrincipal.pas' {frmPrincipal},
  untCadastroEmpresas in 'untCadastroEmpresas.pas'
    {frmEmpresas},
  untCadastroProdutos in 'untCadastroProdutos.pas'
    {frmProdutos},
  untVisaoEmpresasProdutos in
    'untVisaoEmpresasProdutos.pas'
    {frmVisaoEmpresasProdutos},
  untSobre in 'untSobre.pas' {frmSobre};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TfrmPrincipal, frmPrincipal);
  Application.Run;
end.
```



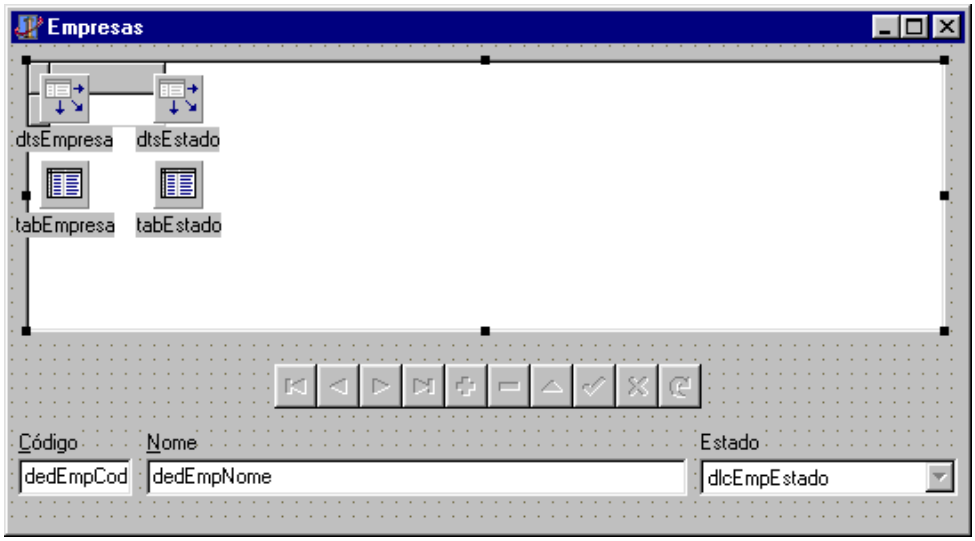

```
unit untPrincipal;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, Menus, Db, DBTables;

type
  TfrmPrincipal = class(TForm)
    mnuPrincipal: TMainMenu;
    Arquivo1: TMenuItem;
    Sair1: TMenuItem;
    Cadastro1: TMenuItem;
    Empresas1: TMenuItem;
    Produtos1: TMenuItem;
    Sobre1: TMenuItem;
    dtbEmpresasProdutos: TDatabase;
    Viso1: TMenuItem;
    Relao1: TMenuItem;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure Relao1Click(Sender: TObject);
    procedure Empresas1Click(Sender: TObject);
    procedure Produtos1Click(Sender: TObject);
```

```
    procedure Sobre1Click(Sender: TObject);
    procedure Sair1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmPrincipal: TfrmPrincipal;

implementation

uses
  untVisaoEmpresasProdutos, untCadastroEmpresas,
  untCadastroProdutos, untSobre;

{$R *.DFM}

procedure TfrmPrincipal.FormCreate(Sender: TObject);
begin
  dtbEmpresasProdutos.Open;
end;

procedure TfrmPrincipal.FormClose(Sender: TObject; var
Action: TCloseAction);
begin
  dtbEmpresasProdutos.Close;

  Action := caFree;
  frmPrincipal := nil;
end;

procedure TfrmPrincipal.Sair1Click(Sender: TObject);
begin
  Close;
end;

procedure TfrmPrincipal.Relao1Click(Sender: TObject);
begin
  if (frmVisaoEmpresasProdutos = nil) then
    frmVisaoEmpresasProdutos :=
      TfrmVisaoEmpresasProdutos.Create (Self);
  frmVisaoEmpresasProdutos.Show;
end;

procedure TfrmPrincipal.Empresas1Click(Sender: TObject);
begin
  if (frmEmpresas = nil) then
    frmEmpresas := TfrmEmpresas.Create (Self);
  frmEmpresas.Show;
```

```
end;

procedure TfrmPrincipal.Produtos1Click(Sender: TObject);
begin
  if (frmProdutos = nil) then
    frmProdutos := TfrmProdutos.Create (Self);
  frmProdutos.Show;
end;

procedure TfrmPrincipal.Sobre1Click(Sender: TObject);
begin
  if (frmSobre = nil) then
    frmSobre := TfrmSobre.Create (Self);
  frmSobre.Show;
end;

end.
```

Destaque:

- tabEmpresa
  - Campo "Estado" lookup com tabEstado



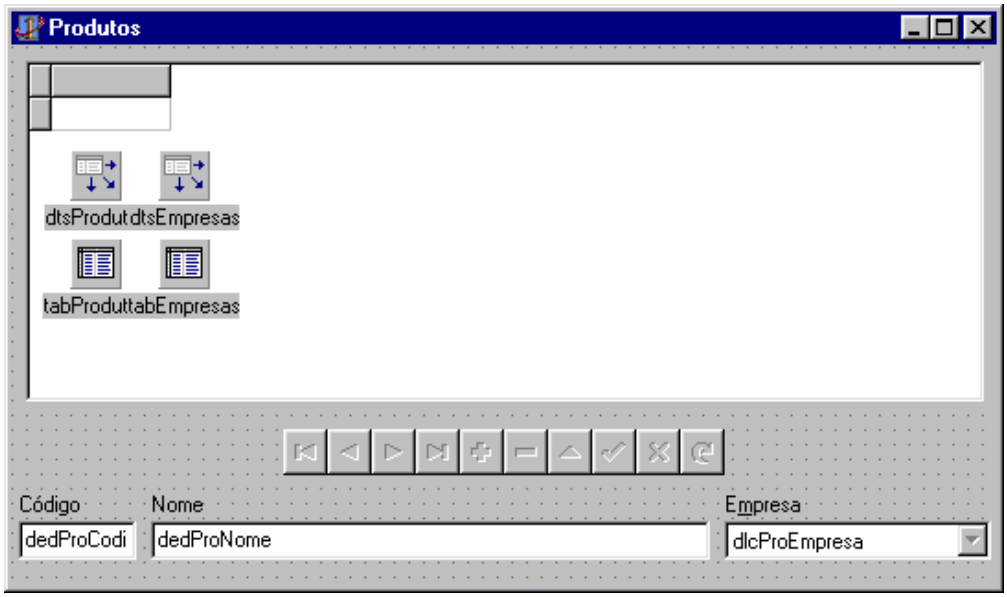

```
unit untCadastroEmpresas;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, Db, DBTables, StdCtrls, Mask, DBCtrls,
  ExtCtrls, Grids, DBGrids;
```

```
type
  TfrmEmpresas = class(TForm)
    dtsEmpresa: TDataSource;
    tabEmpresa: TTable;
    dgrEmpresas: TDBGrid;
    dnvEmpresas: TDBNavigator;
    tabEmpresaEmp_Cod: TFloatField;
    tabEmpresaEmp_Nom: TStringField;
    tabEmpresaEst_Cod: TFloatField;
    Label1: TLabel;
    dedEmpCodigo: TDBEdit;
    Label2: TLabel;
    dedEmpNome: TDBEdit;
    Label3: TLabel;
    tabEstado: TTable;
    dtsEstado: TDataSource;
    dlcEmpEstado: TDBLookupComboBox;
    tabEstadoEst_Cod: TFloatField;
    tabEstadoEst_Nom: TStringField;
    tabEstadoEst_Sig: TStringField;
    tabEmpresaEstado: TStringField;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure dnvEmpresasClick(Sender: TObject; Button:
      TNavigateBtn);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmEmpresas: TfrmEmpresas;

implementation

{$R *.DFM}

procedure TfrmEmpresas.FormCreate(Sender: TObject);
begin
  tabEstado.Open;
  tabEmpresa.Open;
end;

procedure TfrmEmpresas.FormClose(Sender: TObject; var
Action: TCloseAction);
begin
  tabEmpresa.Close;
  tabEstado.Close;
```

```
  Action := caFree;
  frmEmpresas := nil;
end;

procedure TfrmEmpresas.dnvEmpresasClick(Sender: TObject;
  Button: TNavigateBtn);
begin
  case (Button) of
    nbInsert :
      if (dedEmpCodigo.CanFocus) then
        dedEmpCodigo.SetFocus;
  end;
end;

end.
```

Destaque:

- tabProdutos
  - Campo "Empresa" lookup com tabEmpresa



unit untCadastroProdutos;

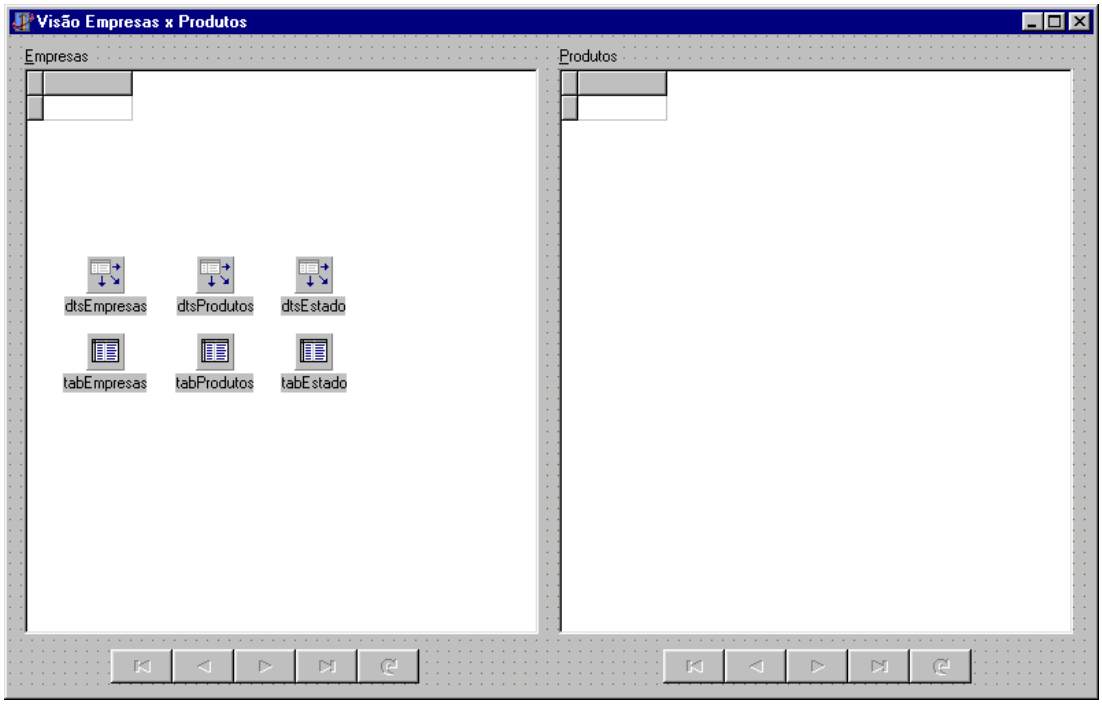
```
interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, Grids, DBGrids, Db, DBTables, ExtCtrls,
  DBCtrls, StdCtrls, Mask;
```

```
type
  TfrmProdutos = class(TForm)
    dgrProdutos: TDBGrid;
    dnvProdutos: TDBNavigator;
    dtsProdutos: TDataSource;
    tabProdutos: TTable;
    dtsEmpresas: TDataSource;
    tabEmpresas: TTable;
    tabProdutosPro_Cod: TFloatField;
    tabProdutosPro_Nom: TStringField;
    tabProdutosEmp_Cod: TFloatField;
    Label1: TLabel;
    dedProCodigo: TDBEdit;
    Label2: TLabel;
    dedProNome: TDBEdit;
    Label3: TLabel;
    dlcProEmpresa: TDBLookupComboBox;
    tabEmpresasEmp_Cod: TFloatField;
    tabEmpresasEmp_Nom: TStringField;
    tabEmpresasEst_Cod: TFloatField;
    tabProdutosEmpresa: TStringField;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure dnvProdutosClick(Sender: TObject; Button:
      TNavigateBtn);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmProdutos: TfrmProdutos;

implementation

{$R *.DFM}

procedure TfrmProdutos.FormCreate(Sender: TObject);
begin
  tabEmpresas.Open;
  tabProdutos.Open;
end;

procedure TfrmProdutos.FormClose(Sender: TObject; var
  Action: TCloseAction);
begin
  tabProdutos.Close;
  tabEmpresas.Close;
```

```
  Action := caFree;
  frmProdutos := nil;
end;

procedure TfrmProdutos.dnvProdutosClick(Sender: TObject;
  Button: TNavigateBtn);
begin
  case (Button) of
    nbInsert :
      if (dedProCodigo.CanFocus) then
        dedProCodigo.SetFocus;
  end;
end;

end.
```

Destaque:

- tabEmpresa
  - Campo "Estado" lookup com tabEstado




```
unit untVisaoEmpresasProdutos;

interface
```

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, ExtCtrls, DBCtrls, StdCtrls, Grids,
  DBGrids, Db, DBTables;

type
  TfrmVisaoEmpresasProdutos = class(TForm)
    dgrVisaoEmpresas: TDBGrid;
    dgrVisaoProdutos: TDBGrid;
    Label1: TLabel;
    Label2: TLabel;
    dnvVisaoEmpresas: TDBNavigator;
    dnvVisaoProdutos: TDBNavigator;
    dtsEmpresas: TDataSource;
    dtsProdutos: TDataSource;
    tabEmpresas: TTable;
    tabProdutos: TTable;
    tabEmpresasEmp_Cod: TFloatField;
    tabEmpresasEmp_Nom: TStringField;
    tabEmpresasEst_Cod: TFloatField;
    tabProdutosPro_Cod: TFloatField;
    tabProdutosPro_Nom: TStringField;
    tabProdutosEmp_Cod: TFloatField;
    dtsEstado: TDataSource;
    tabEstado: TTable;
    tabEstadoEst_Cod: TFloatField;
    tabEstadoEst_Nom: TStringField;
    tabEstadoEst_Sig: TStringField;
    tabEmpresasEstado: TStringField;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmVisaoEmpresasProdutos: TfrmVisaoEmpresasProdutos;

implementation

{$R *.DFM}

procedure TfrmVisaoEmpresasProdutos.FormCreate(Sender:
  TObject);
begin
  tabEstado.Open;
  tabEmpresas.Open;
  tabProdutos.Open;
```

```
end;

procedure TfrmVisaoEmpresasProdutos.FormClose(Sender:
  TObject; var Action: TCloseAction);
begin
  tabProdutos.Close;
  tabEmpresas.Close;
  tabEstado.Close;

  Action := caFree;
  frmVisaoEmpresasProdutos := nil;
end;

end.
```


```
unit untSobre;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Buttons;

type
  TfrmSobre = class(TForm)
    mnoSobre: TMemo;
    bbnFechar: TBitBtn;
    procedure FormClose(Sender: TObject; var Action:
```

```
      TCloseAction);
    procedure bbnFecharClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmSobre: TfrmSobre;

implementation

{$R *.DFM}

procedure TfrmSobre.FormClose(Sender: TObject; var Action:
  TCloseAction);
begin
  Action := caFree;
  frmSobre := nil;
end;

procedure TfrmSobre.bbnFecharClick(Sender: TObject);
begin
  Close;
end;

end.
```

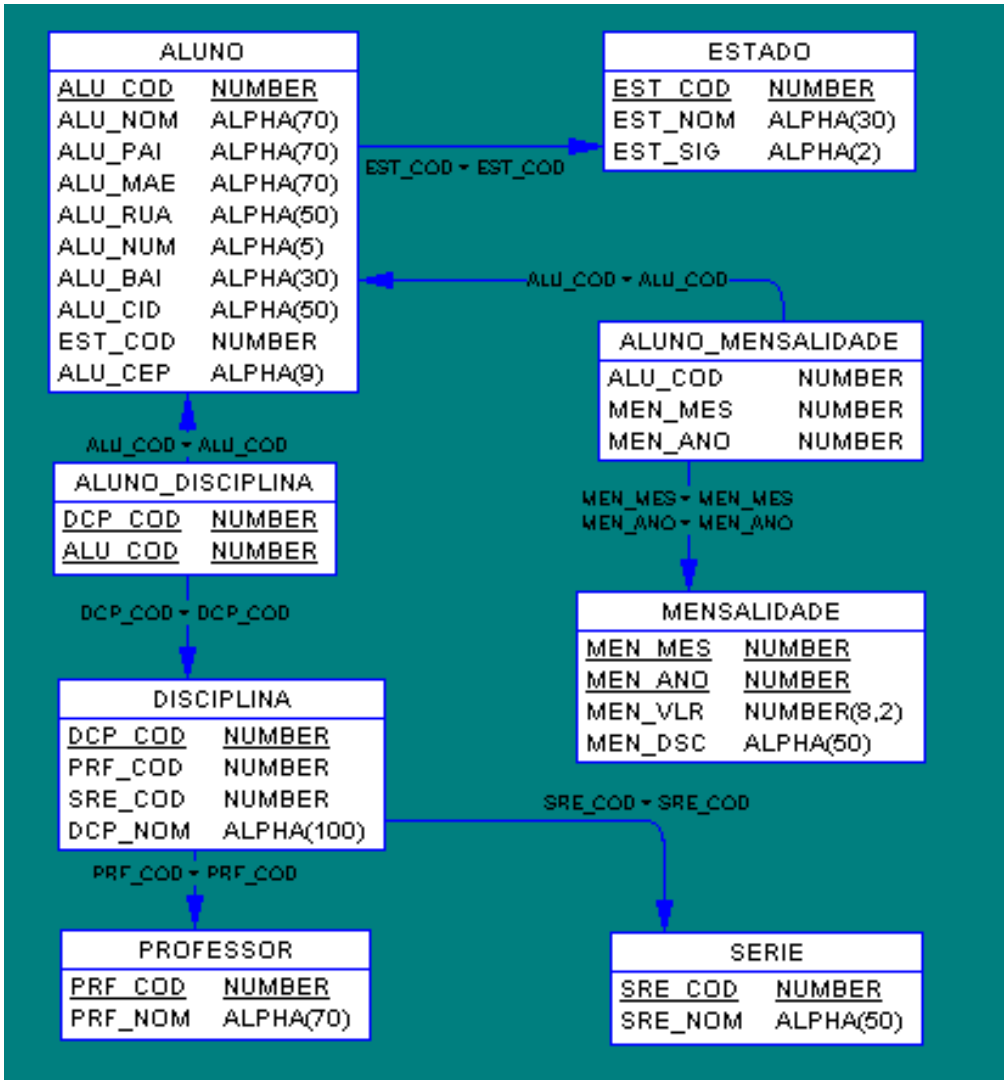
Sistema de Gerenciamento de Escola (SGE):

- **Utilização de TDataBase**
- **Utilização de Menu**
- **Visão Mestre-detalhe**
- **Diversos Cadastros**
- **Formulários MDI**
- **Relacionamento N x N entre tabelas**
- **Formatação de campos**
- **Relatórios**

Exemplo de Software

3) Exemplo_7\SGE: prjSGE.exe – Software de Gerenciamento de Escolal:

TABELAS PARADOX:



- **Estado**
  - Est_Cod Number **(Primary Key)**
  - Est_Nom Alpha 30
  - Est_Sig Alpha 02
- **Aluno**
  - Alu_Cod Number **(Primary Key)**
  - Alu_Nom Alpha 70
  - Alu_Pai Alpha 70
  - Alu_Mãe Alpha 70
  - Alu_Rua Alpha 50
  - Alu_Num Alpha 05
  - Alu_Bai Alpha 30
  - Alu_Cid Alpha 50
  - Alu_CEP Alpha 09
  - Est_Cod Number **(Foreign Key)**
- **Serie**
  - Sre_Cod Number **(Primary Key)**
  - Sre_Nom Alpha 50
- **Professor**
  - Prf_Cod Number **(Primary Key)**
  - Prf_Nom Aplha 70
- **Disciplina**
  - Dcp_Cod Number **(Primary Key)**
  - Dcp_Nom Alpha 100
  - Sre_Cód Number **(Foreign Key)**
  - Prf_Cod Number **(Foreign Key)**
- **Aluno_Disciplina**
  - Dcp_Cod Number **(Primary Key) (Foreign Key)**
  - Alu_Cod Number **(Primary Key) (Foreign Key)**
- **Mes**
  - Mes_Cod Number **(Primary Key)**
  - Mes_Nom Alpha 09
- **Mensalidade**
  - Men_Ano Number **(Primary key)**
  - Mes_Cod Number **(Primary Key) (Foreign Key)**
  - Men_Vlr Number
  - Men_Dsc Alpha 50
- **Aluno_Mensalidade**
  - Men_Ano Number **(Primary Key) (Foreign Key)**
  - Mes_Cod Number **(Primary Key) (Foreign Key)**
  - Alu_Cod Number **(Primary Key) (Foreign Key)**

Criar ALIAS: aSGE

Formulários: MDI

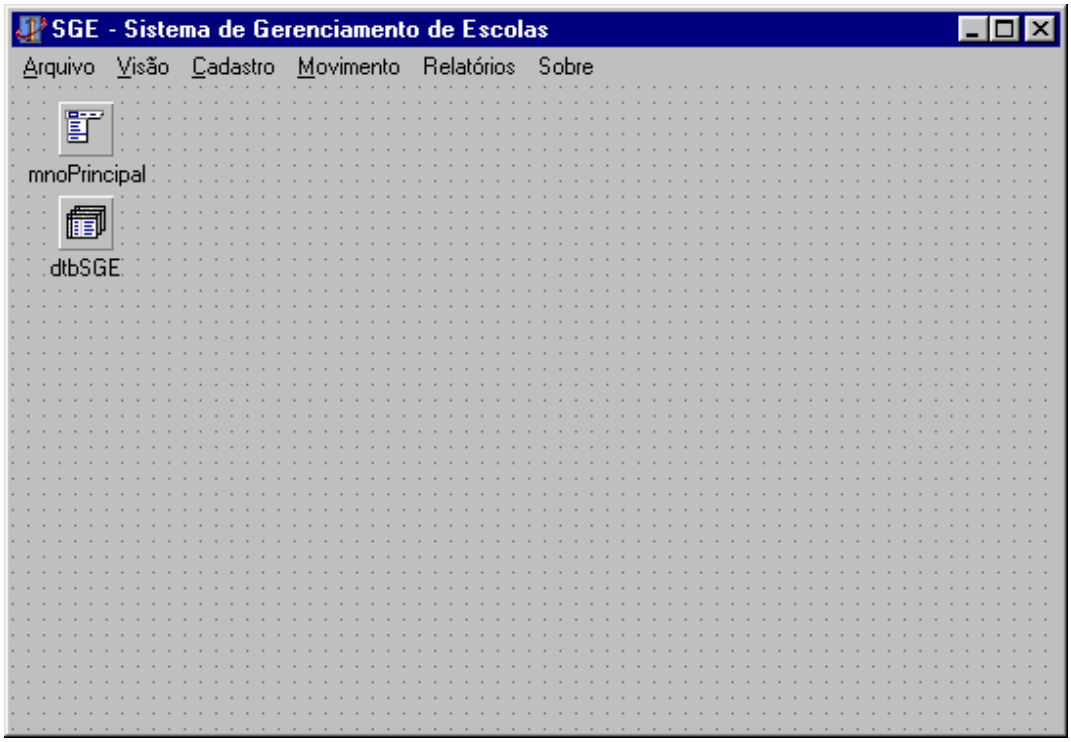
IMPORTANTE:

O sistema será subdivido nas sete partes a seguir:

- Tela Principal (untPrincipal)
- Data Modules
- Visões
- Cadastro
- Movimento
- Relatórios
- Informações Gerais (Sobre)

```
program prjSGE;

uses
  Forms,
  untPrincipal in 'untPrincipal.pas' {frmPrincipal},
  untSobre in 'untSobre.pas' {frmSobre},
  untdmdPrincipal in 'untdmdPrincipal.pas' {dmdPrincipal:
    TDataModule},
  untCadAluno in 'untCadAluno.pas' {frmAlunos},
  untCadDisciplina in 'untCadDisciplina.pas'
    {frmDisciplinas},
  untCadMensalidade in 'untCadMensalidade.pas'
    {frmMensalidades},
  untCadProfessor in 'untCadProfessor.pas'
    {frmProfessores},
  untCadSerie in 'untCadSerie.pas' {frmSeries},
  untVisAlunosDisciplinas in 'untVisAlunosDisciplinas.pas'
    {frmVisAlunosDisciplinas},
  untVisAlunosMensalidades in
    'untVisAlunosMensalidades.pas'
    {frmVisaoAlunosMensalidades},
  untVisProfessoresDisciplinas in
    'untVisProfessoresDisciplinas.pas'
    {frmVisProfessoresDisciplinas},
  untRecebimentoMensalidades in
    'untRecebimentoMensalidades.pas'
    {frmRecebimentoMensalidades},
  untdmdMovimento in 'untdmdMovimento.pas' {dmdMovimento:
    TDataModule},
  untAlunoDisciplinas in 'untAlunoDisciplinas.pas'
    {frmAlunoDisciplinas},
  untRelAlunos in 'untRelAlunos.pas' {frmRelAlunos},
  untdmdRelatorios in 'untdmdRelatorios.pas'
    {dmdRelatorios: TDataModule},
  untRelDisciplinas in 'untRelDisciplinas.pas'
    {frmRelDisciplinas},
  untRelProfessores in 'untRelProfessores.pas'
    {frmRelProfessores},
  untRelAlunosDisciplinas in 'untRelAlunosDisciplinas.pas'
    {frmRelAlunosDisciplinas};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TfrmPrincipal, frmPrincipal);
  Application.Run;
end.
```

Tela Principal




```
unit untPrincipal;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, Db, DBTables, Menus;

type
  TfrmPrincipal = class(TForm)
    mnoPrincipal: TMainMenu;
    Arquivo1: TMenuItem;
    Sair1: TMenuItem;
    Viso1: TMenuItem;
    AlunoMensalidades1: TMenuItem;
    AlunoDisciplinas1: TMenuItem;
    Cadastro1: TMenuItem;
    Aluno1: TMenuItem;
    Disciplina1: TMenuItem;
    Srie1: TMenuItem;
    Professor1: TMenuItem;
    ProfessorDisciplina1: TMenuItem;
    Mensalidade1: TMenuItem;
    Sobre1: TMenuItem;
    dtbSGE: TDatabase;
    Relatrios1: TMenuItem;
```

```
    Alunos1: TMenuItem;
    Disciplinas1: TMenuItem;
    Professores1: TMenuItem;
    AlunosxDisciplinas1: TMenuItem;
    Movimento1: TMenuItem;
    RecebimentoMensalidades1: TMenuItem;
    CadastroAlunoemDisciplina1: TMenuItem;
    procedure Sair1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure AlunoMensalidades1Click(Sender: TObject);
    procedure AlunoDisciplinas1Click(Sender: TObject);
    procedure ProfessorDisciplina1Click(Sender: TObject);
    procedure Aluno1Click(Sender: TObject);
    procedure Disciplina1Click(Sender: TObject);
    procedure Srie1Click(Sender: TObject);
    procedure Professor1Click(Sender: TObject);
    procedure Mensalidade1Click(Sender: TObject);
    procedure Sobre1Click(Sender: TObject);
    procedure RecebimentoMensalidades1Click(Sender:
      TObject);
    procedure CadastroAlunoemDisciplina1Click(Sender:
      TObject);
    procedure Alunos1Click(Sender: TObject);
    procedure Disciplinas1Click(Sender: TObject);
    procedure Professores1Click(Sender: TObject);
    procedure AlunosxDisciplinas1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmPrincipal: TfrmPrincipal;

implementation

uses
  untdmdPrincipal, untSobre, untCadAluno, untCadDisciplina,
  untCadSerie, untCadProfessor, untCadMensalidade,
  untVisAlunosMensalidades, untVisAlunosDisciplinas,
  untVisProfessoresDisciplinas, untRecebimentoMensalidades,
  untdmdMovimento, untAlunoDisciplinas,
  untdmdRelatorios, untQkrRelAluno, untRelDisciplinas,
  untRelProfessores, untRelAlunosDisciplinas,
  untQkrRelAlunoDisciplinas, untQkrRelDisciplina,
  untQkrRelProfessor;

{$R *.DFM}
```

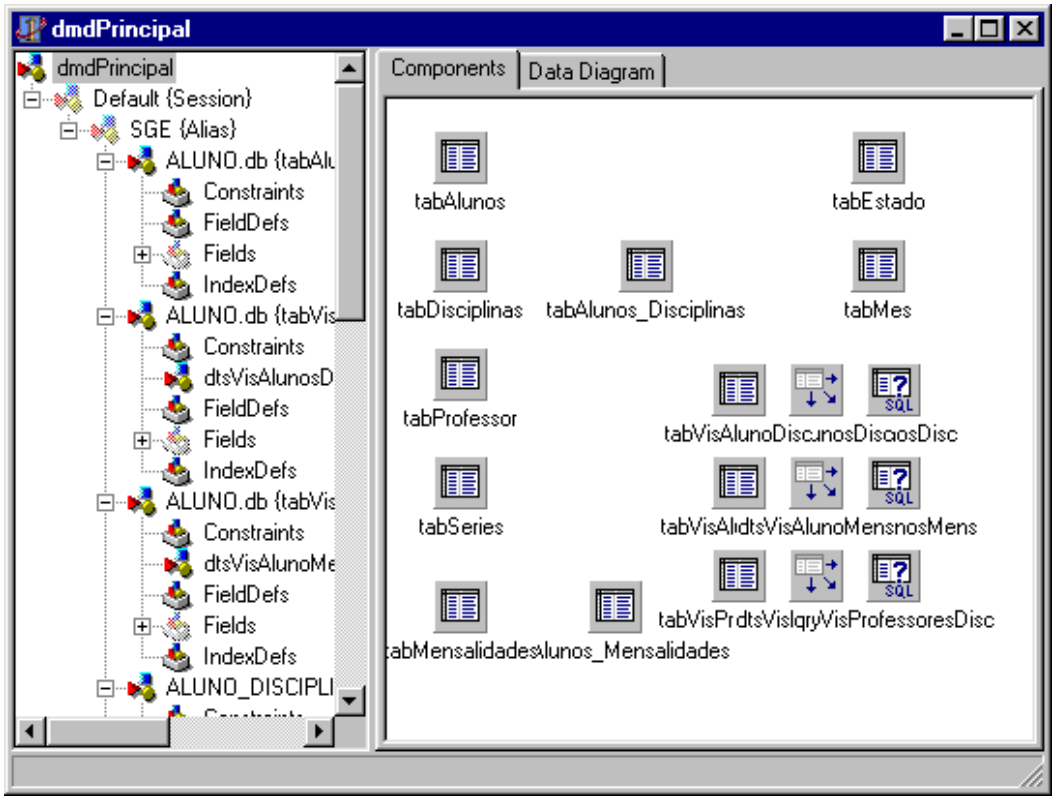
```
procedure TfrmPrincipal.FormCreate(Sender: TObject);
begin
  dtbSGE.Open;

  if (dtbSGE.Connected) then
  begin
    if (dmdPrincipal = nil) then
      dmdPrincipal := TdmdPrincipal.Create (Self);
    if (dmdMovimento = nil) then
      dmdMovimento := TdmdMovimento.Create (Self);
    if (dmdRelatorios = nil) then
      dmdRelatorios := TdmdRelatorios.Create (Self);
  end;
end;

procedure TfrmPrincipal.FormClose(Sender: TObject; var
  Action: TCloseAction);
begin
  dtbSGE.Close;
  FreeAndNil (dmdPrincipal);
  FreeAndNil (dmdMovimento);

  Action := caFree;
  frmPrincipal := nil;
end;

procedure TfrmPrincipal.Sair1Click(Sender: TObject);
begin
  Close;
end;

procedure TfrmPrincipal.AlunoMensalidades1Click(Sender:
  TObject);
begin
  if (frmVisaoAlunosMensalidades = nil) then
    frmVisaoAlunosMensalidades :=
      TfrmVisaoAlunosMensalidades.Create (Self);
  frmVisaoAlunosMensalidades.Show;
end;

procedure TfrmPrincipal.AlunoDisciplinas1Click(Sender:
  TObject);
begin
  if (frmVisAlunosDisciplinas = nil) then
    frmVisAlunosDisciplinas :=
      TfrmVisAlunosDisciplinas.Create (Self);
  frmVisAlunosDisciplinas.Show;
end;

procedure TfrmPrincipal.ProfessorDisciplina1Click(Sender:
```

```
  TObject);
begin
  if (frmVisProfessoresDisciplinas = nil) then
    frmVisProfessoresDisciplinas :=
      TfrmVisProfessoresDisciplinas.Create (Self);
  frmVisProfessoresDisciplinas.Show;
end;

procedure TfrmPrincipal.Aluno1Click(Sender: TObject);
begin
  if (frmAlunos = nil) then
    frmAlunos := TfrmAlunos.Create (Self);
  frmAlunos.Show;
end;

procedure TfrmPrincipal.Disciplina1Click(Sender: TObject);
begin
  if (frmDisciplinas = nil) then
    frmDisciplinas := TfrmDisciplinas.Create (Self);
  frmDisciplinas.Show;
end;

procedure TfrmPrincipal.Srie1Click(Sender: TObject);
begin
  if (frmSeries = nil) then
    frmSeries := TfrmSeries.Create (Self);
  frmSeries.Show;
end;

procedure TfrmPrincipal.Professor1Click(Sender: TObject);
begin
  if (frmProfessores = nil) then
    frmProfessores := TfrmProfessores.Create (Self);
  frmProfessores.Show;
end;

procedure TfrmPrincipal.Mensalidade1Click(Sender: TObject);
begin
  if (frmMensalidades = nil) then
    frmMensalidades := TfrmMensalidades.Create (Self);
  frmMensalidades.Show;
end;

procedure TfrmPrincipal.Sobre1Click(Sender: TObject);
begin
  if (frmSobre = nil) then
    frmSobre := TfrmSobre.Create (Self);
  frmSobre.Show;
end;

procedure
```

```
  TfrmPrincipal.RecebimentoMensalidades1Click(Sender:
  TObject);
begin
  if (frmRecebimentoMensalidades = nil) then
    frmRecebimentoMensalidades :=
      TfrmRecebimentoMensalidades.Create (Self);
  frmRecebimentoMensalidades.Show;
end;

procedure
  TfrmPrincipal.CadastroAlunoemDisciplina1Click(Sender:
  TObject);
begin
  if (frmAlunoDisciplinas = nil) then
    frmAlunoDisciplinas := TfrmAlunoDisciplinas.Create
      (Self);
  frmAlunoDisciplinas.Show;
end;

procedure TfrmPrincipal.Alunos1Click(Sender: TObject);
begin
  try
    if (frmQkrRelAluno = nil) then
      frmQkrRelAluno := TfrmQkrRelAluno.Create (Self);

    dmdRelatorios.tabRelAlunos.Open;
    frmQkrRelAluno.Preview;
  finally
    dmdRelatorios.tabRelAlunos.Close;
    FreeAndNil (frmQkrRelAluno);
  end;
end;

procedure TfrmPrincipal.Disciplinas1Click(Sender: TObject);
begin
  try
    if (frmQkrRelDisciplina = nil) then
      frmQkrRelDisciplina := TfrmQkrRelDisciplina.Create
        (Self);

    dmdRelatorios.tabRelDisciplinas.Open;
    frmQkrRelDisciplina.Preview;
  finally
    dmdRelatorios.tabRelDisciplinas.Close;
    FreeAndNil (frmQkrRelDisciplina);
  end;
end;

procedure TfrmPrincipal.Professores1Click(Sender: TObject);
begin
  try
```

```
    if (frmQkrRelProfessor = nil) then
      frmQkrRelProfessor := TfrmQkrRelProfessor.Create
        (Self);

    dmdRelatorios.tabRelProfessores.Open;
    frmQkrRelProfessor.Preview;
  finally
    dmdRelatorios.tabRelProfessores.Close;
    FreeAndNil (frmQkrRelProfessor);
  end;
end;

procedure TfrmPrincipal.AlunosxDisciplinas1Click(Sender:
  TObject);
begin
  try
    if (frmQkrRelAlunoDisciplinas = nil) then
      frmQkrRelAlunoDisciplinas :=
        TfrmQkrRelAlunoDisciplinas.Create (Self);

    dmdRelatorios.tabRelAlunoDisc.Open;
    dmdRelatorios.qryRelAlunoDisc.Open;
    frmQkrRelAlunoDisciplinas.Preview;
  finally
    dmdRelatorios.qryRelAlunoDisc.Close;
    dmdRelatorios.tabRelAlunoDisc.Close;
    FreeAndNil (frmQkrRelAlunoDisciplinas);
  end;
end;

end.
```

Data Modules



Componentes:

TTable

- tabAlunos
  - Active = False
  - DataBaseName = SGE
  - TableName = ALUNO.db

- tabDisciplinas
  - Active = False
  - DataBaseName = SGE
  - TableName = DISCIPLINA.DB

- tabProfessor
  - Active = False
  - DataBaseName = SGE
  - TableName = PROFESSOR.db

- tabSeries
  - Active = False
  - DataBaseName = SGE
  - TableName = SERIE.db

- tabMensalidades
  - Active = False
  - DataBaseName = SGE
  - TableName = MENSALIDADE.db

- tabAlunos_Disciplinas
  - Active = False
  - DataBaseName = SGE
  - TableName = ALUNO_DISCIPLINA.db

- tabAlunos_Mensalidades
  - Active = False
  - DataBaseName = SGE
  - TableName = ALUNO_MENSALIDADE.db

- tabEstado
  - Active = False
  - DataBaseName = SGE
  - TableName = Estado.DB

- tabMes
  - Active = False
  - DataBaseName = SGE
  - TableName = MES.db

- tabVisAlunoDisc
  - Active = False
  - DataBaseName = SGE
  - TableName = ALUNO.db

- tabVisAlunoMens
  - Active = False
  - DataBaseName = SGE
  - TableName = ALUNO.db

- tabVisProfessores
  - Active = False
  - DataBaseName = SGE
  - TableName = PROFESSOR.db

TDataSource

- dtsVisAlunosDisc
  - DataSet = tabVisAlunoDisc

- dtsVisAlunoMens
  - DataSet = tabVisAlunoMens

- dtsVisProfessores
  - DataSet = tabVisProfessores

TQuery

- qryAlunosDisc
  - Active = False
  - DataBaseName = SGE
  - DataSource = dtsVisAlunosDisc
  - Params
    - ALU_COD
      - DataType = ftInteger
      - ParamType = ptInput
  - SQL =

```
SELECT DCP.DCP_COD, DCP.DCP_NOM, SRE.SRE_NOM, PRF.PRF_NOM
  FROM ALUNO_DISCIPLINA ADC, DISCIPLINA DCP, SERIE SRE,
       PROFESSOR PRF
 WHERE ADC.ALU_COD = :ALU_COD
   AND ADC.DCP_COD = DCP.DCP_COD
   AND DCP.SRE_COD = SRE.SRE_COD
   AND DCP.PRF_COD = PRF.PRF_COD
ORDER BY DCP.DCP_NOM
```

- qryVisAlunosMens
  - Active = False
  - DataBaseName = SGE
  - DataSource = dtsVisAlunoMens
  - Params
    - ALU_COD
      - DataType = ftInteger
      - ParamType = ptInput
  - SQL =

```
SELECT MEN.MEN_ANO, MEN.MES_COD, MEN.MEN_VLR, MEN.MEN_DSC
  FROM ALUNO_MENSALIDADE AME, MENSALIDADE MEN
 WHERE AME.ALU_COD = :ALU_COD
   AND AME.MEN_ANO = MEN.MEN_ANO
   AND AME.MES_COD = MEN.MES_COD
ORDER BY MEN.MEN_ANO, MEN.MES_COD DESC
```

- qryVisProfessoresDisc
  - Active = False
  - DataBaseName = SGE
  - DataSource = dtsVisProfessores
  - Params
    - PRF_COD
      - DataType = ftInteger
      - ParamType = ptInput
  - SQL =

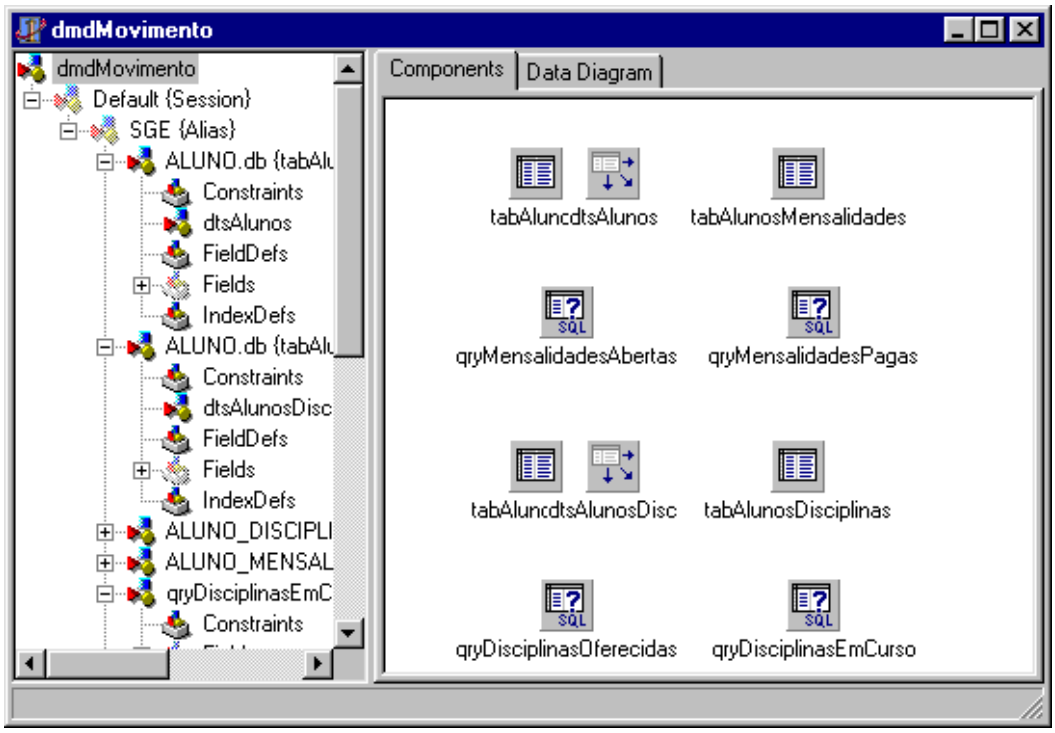
```
SELECT DCP.DCP_COD, DCP.DCP_NOM, SRE.SRE_NOM
  FROM DISCIPLINA DCP, SERIE SRE
```

```
   WHERE DCP.PRF_COD = :PRF_COD
     AND DCP.SRE_COD = SRE.SRE_COD
ORDER BY DCP.DCP_NOM
```

```
unit untdmdPrincipal;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, Db, DBTables;

type
  TdmdPrincipal = class(TDataModule)
    tabAlunos: TTable;
    tabAlunosALU_COD: TFloatField;
    tabAlunosALU_NOM: TStringField;
    tabAlunosALU_PAI: TStringField;
    tabAlunosALU_MAE: TStringField;
    tabAlunosALU_RUA: TStringField;
    tabAlunosALU_NUM: TStringField;
    tabAlunosALU_BAI: TStringField;
    tabAlunosALU_CID: TStringField;
    tabAlunosEST_COD: TFloatField;
    tabAlunosALU_CEP: TStringField;
    tabEstado: TTable;
    tabEstadoEst_Cod: TFloatField;
    tabEstadoEst_Nom: TStringField;
    tabEstadoEst_Sig: TStringField;
    tabDisciplinas: TTable;
    tabDisciplinasDCP_COD: TFloatField;
    tabDisciplinasDCP_NOM: TStringField;
    tabDisciplinasSRE_COD: TFloatField;
    tabDisciplinasPRF_COD: TFloatField;
    tabProfessor: TTable;
    tabSeries: TTable;
    tabDisciplinasSrie: TStringField;
    tabDisciplinasProfessor: TStringField;
    tabProfessorPRF_COD: TFloatField;
    tabProfessorPFR_NOM: TStringField;
    tabSeriesSRE_COD: TFloatField;
    tabSeriesSRE_NOM: TStringField;
    tabAlunos_Disciplinas: TTable;
    tabAlunos_Mensalidades: TTable;
    tabMensalidades: TTable;
    tabMes: TTable;
    tabMensalidadesMES_COD: TFloatField;
    tabMensalidadesMEN_ANO: TFloatField;
    tabMensalidadesMEN_VLR: TFloatField;
    tabMensalidadesMEN_DSC: TStringField;
```

```
    tabAlunos_MensalidadesALU_COD: TFloatField;
    tabAlunos_MensalidadesMES_COD: TFloatField;
    tabAlunos_MensalidadesMEN_ANO: TFloatField;
    tabMesMES_COD: TFloatField;
    tabMesMES_NOM: TStringField;
    tabAlunos_DisciplinasDCP_COD: TFloatField;
    tabAlunos_DisciplinasALU_COD: TFloatField;
    tabMensalidadesMs: TStringField;
    qryAlunosDisc: TQuery;
    qryVisAlunosMens: TQuery;
    dtsVisAlunosDisc: TDataSource;
    tabVisAlunoDisc: TTable;
    tabVisAlunoDiscALU_COD: TFloatField;
    tabVisAlunoDiscALU_NOM: TStringField;
    tabVisAlunoDiscALU_PAI: TStringField;
    tabVisAlunoDiscALU_MAE: TStringField;
    tabVisAlunoDiscALU_RUA: TStringField;
    tabVisAlunoDiscALU_NUM: TStringField;
    tabVisAlunoDiscALU_BAI: TStringField;
    tabVisAlunoDiscALU_CID: TStringField;
    tabVisAlunoDiscEST_COD: TFloatField;
    tabVisAlunoDiscALU_CEP: TStringField;
    tabVisAlunoMens: TTable;
    FloatField1: TFloatField;
    StringField1: TStringField;
    StringField2: TStringField;
    StringField3: TStringField;
    StringField4: TStringField;
    StringField5: TStringField;
    StringField6: TStringField;
    StringField7: TStringField;
    FloatField2: TFloatField;
    StringField8: TStringField;
    dtsVisAlunoMens: TDataSource;
    tabVisProfessores: TTable;
    FloatField3: TFloatField;
    StringField9: TStringField;
    dtsVisProfessores: TDataSource;
    qryVisProfessoresDisc: TQuery;
    qryAlunosDiscDCP_COD: TFloatField;
    qryAlunosDiscDCP_NOM: TStringField;
    qryAlunosDiscSRE_NOM: TStringField;
    qryAlunosDiscPRF_NOM: TStringField;
    qryVisAlunosMensMEN_ANO: TFloatField;
    qryVisAlunosMensMES_COD: TFloatField;
    qryVisAlunosMensMEN_VLR: TFloatField;
    qryVisAlunosMensMEN_DSC: TStringField;
    qryVisProfessoresDiscDCP_COD: TFloatField;
    qryVisProfessoresDiscDCP_NOM: TStringField;
    qryVisProfessoresDiscSRE_NOM: TStringField;
  private
```

```
    { Private declarations }
  public
    { Public declarations }
  end;

var
  dmdPrincipal: TdmdPrincipal;

implementation

{$R *.DFM}

end.
```



Componentes:

TTable

- tabAlunos
  - Active = False
  - DataBaseName = SGE
  - TableName = ALUNO.db

- tabAlunosMensalidades
  - Active = False
  - DataBaseName = SGE
  - TableName = ALUNO_MENSALIDADE.DB

- tabAlunosDisc
  - Active = False
  - DataBaseName = SGE
  - TableName = ALUNO.db

- tabAlunosDisciplinas
  - Active = False
  - DataBaseName = SGE
  - TableName = ALUNO_DISCIPLINA.db

TDataSource

- dtsAlunos
  - DataSet = tabAlunos

- dtsAlunosDisc
  - DataSet = tabAlunosDisc

TQuery

- qryMensalidadesAbertas
  - Active = False
  - DataBaseName = SGE
  - DataSource = dtsAlunos
  - Params
    - ALU_COD
      - DataType = ftInteger
      - ParamType = ptInput
  - SQL =

```
SELECT MEN.MEN_ANO, MEN.MES_COD, MEN.MEN_VLR
  FROM MENSALIDADE MEN
 WHERE NOT EXISTS (SELECT 1
                     FROM ALUNO_MENSALIDADE AM2
                    WHERE AM2.ALU_COD = :ALU_COD
                      AND AM2.MES_COD = MEN.MES_COD
                      AND AM2.MEN_ANO = MEN.MEN_ANO)
ORDER BY MEN.MEN_ANO, MEN.MES_COD DESC
```

- qryMensalidadesPagas
  - Active = False
  - DataBaseName = SGE
  - DataSource = dtsAlunos
  - Params
    - ALU_COD
      - DataType = ftInteger
      - ParamType = ptInput
  - SQL =

```
SELECT MEN.MEN_ANO, MEN.MES_COD, MEN.MEN_VLR
  FROM ALUNO_MENSALIDADE AME, MENSALIDADE MEN
 WHERE AME.ALU_COD = :ALU_COD
   AND AME.MEN_ANO = MEN.MEN_ANO
   AND AME.MES_COD = MEN.MES_COD
ORDER BY MEN.MEN_ANO, MEN.MES_COD DESC
```

- qryDisciplinasOferecidas
  - Active = False
  - DataBaseName = SGE
  - DataSource = dtsAlunosDisc
  - Params
    - ALU_COD

- DataType = ftInteger
- ParamType = ptInput

- SQL =

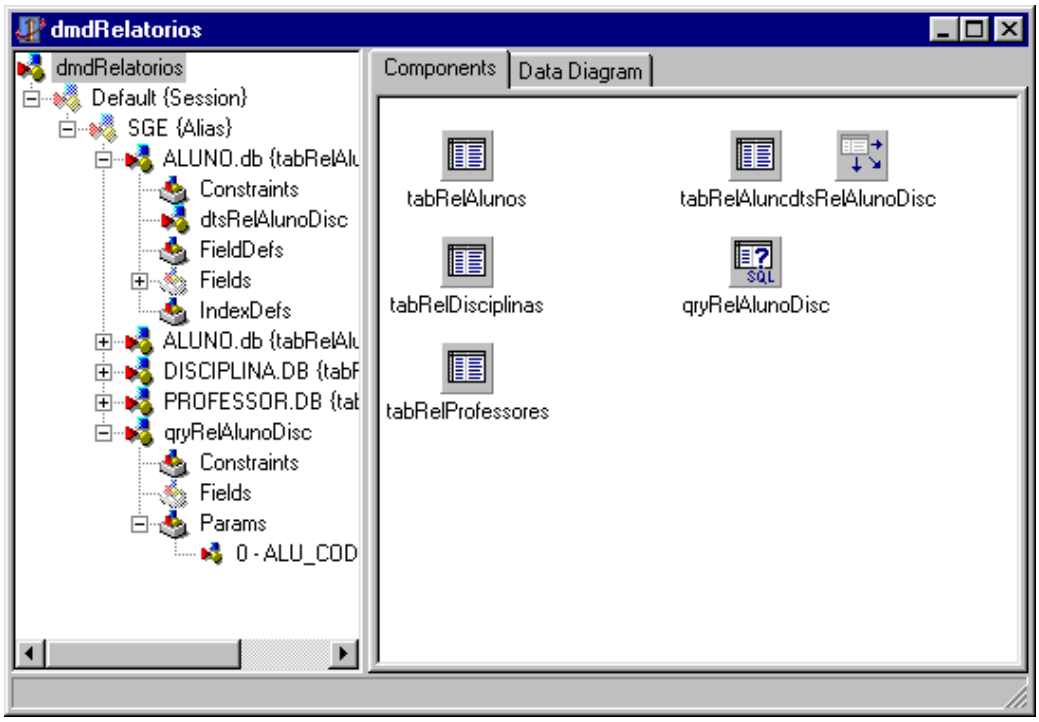
```
  SELECT DCP.DCP_COD, DCP.DCP_NOM
    FROM DISCIPLINA DCP
   WHERE DCP.DCP_COD NOT IN (SELECT AD2.DCP_COD
                               FROM ALUNO_DISCIPLINA AD2
                              WHERE AD2.ALU_COD = :ALU_COD)
ORDER BY DCP.DCP_NOM
```

- qryDisciplinasEmCurso
  - Active = False
  - DataBaseName = SGE
  - DataSource = dtsAlunosDisc
  - Params
    - ALU_COD
      - DataType = ftInteger
      - ParamType = ptInput
  - SQL =

```
  SELECT DCP.DCP_COD, DCP.DCP_NOM
    FROM ALUNO_DISCIPLINA ADI, DISCIPLINA DCP
   WHERE ADI.ALU_COD = :ALU_COD
     AND ADI.DCP_COD = DCP.DCP_COD
ORDER BY DCP.DCP_NOM
```

```
unit untdmdMovimento;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, Db, DBTables;

type
  TdmdMovimento = class(TDataModule)
    tabAlunos: TTable;
    tabAlunosMensalidades: TTable;
    dtsAlunos: TDataSource;
    qryMensalidadesPagas: TQuery;
    qryMensalidadesAbertas: TQuery;
    tabAlunosALU_COD: TFloatField;
    tabAlunosALU_NOM: TStringField;
    tabAlunosDisc: TTable;
    FloatField1: TFloatField;
    StringField1: TStringField;
```

```
    tabAlunosDisciplinas: TTable;
    dtsAlunosDisc: TDataSource;
    qryDisciplinasEmCurso: TQuery;
    qryDisciplinasOferecidas: TQuery;
    qryDisciplinasOferecidasDCP_COD: TFloatField;
    qryDisciplinasOferecidasDCP_NOM: TStringField;
    qryDisciplinasEmCursoDCP_COD: TFloatField;
    qryDisciplinasEmCursoDCP_NOM: TStringField;
    qryMensalidadesPagasMEN_ANO: TFloatField;
    qryMensalidadesPagasMES_COD: TFloatField;
    qryMensalidadesPagasMEN_VLR: TFloatField;
    qryMensalidadesAbertasMEN_ANO: TFloatField;
    qryMensalidadesAbertasMES_COD: TFloatField;
    qryMensalidadesAbertasMEN_VLR: TFloatField;
    tabAlunosDisciplinasDCP_COD: TFloatField;
    tabAlunosDisciplinasALU_COD: TFloatField;
    tabAlunosMensalidadesALU_COD: TFloatField;
    tabAlunosMensalidadesMES_COD: TFloatField;
    tabAlunosMensalidadesMEN_ANO: TFloatField;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  dmdMovimento: TdmdMovimento;

implementation

{$R *.DFM}

end.
```



Componentes:

TTable

- tabRelAlunos
  - Active = False
  - DataBaseName = SGE
  - TableName = ALUNO.db

- tabRelDisciplinas
  - Active = False
  - DataBaseName = SGE
  - TableName = DISCIPLINA.DB

- tabRelProfessores
  - Active = False
  - DataBaseName = SGE
  - TableName = PROFESSOR.DB

- tabRelAlunoDisc
  - Active = False
  - DataBaseName = SGE
  - TableName = ALUNO.db

TDataSource

- dtsRelAlunoDisc
  - DataSet = tabRelAlunoDisc

TQuery

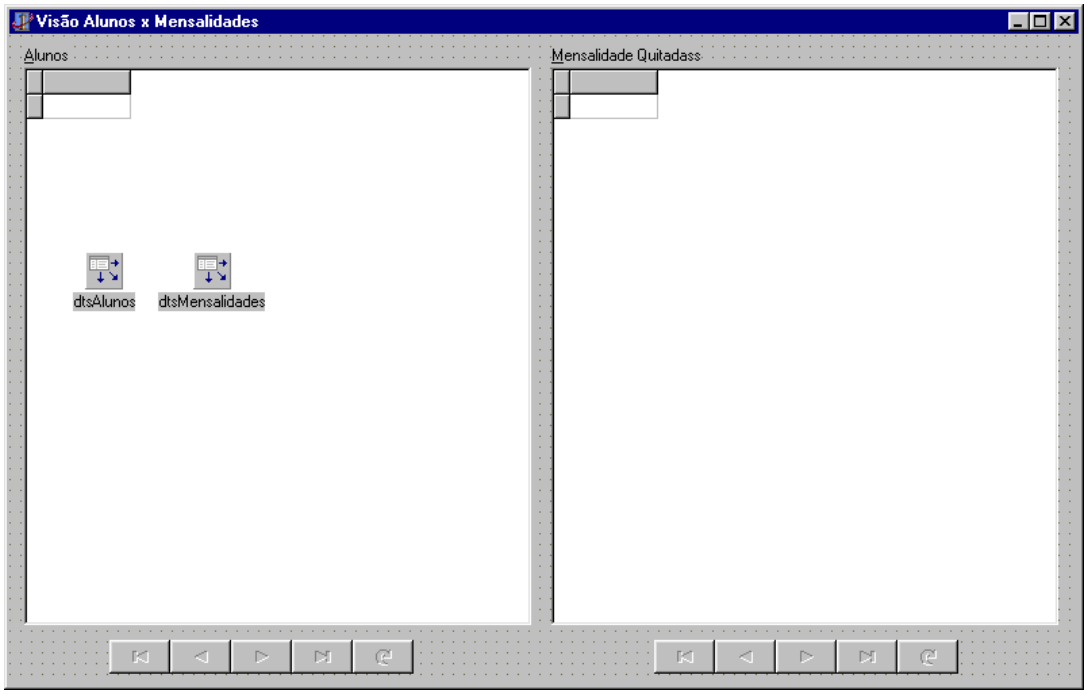
- qryRelAlunoDisc
  - Active = False
  - DataBaseName = SGE
  - DataSource = dtsRelAlunoDisc
  - Params
    - ALU_COD
      - DataType = ftInteger
      - ParamType = ptInput
  - SQL =

```
SELECT DCP.DCP_COD, DCP.DCP_NOM, SRE.SRE_NOM, PRF.PRF_NOM
  FROM ALUNO_DISCIPLINA ADC, DISCIPLINA DCP, SERIE SRE,
       PROFESSOR PRF
 WHERE ADC.ALU_COD = :ALU_COD
   AND ADC.DCP_COD = DCP.DCP_COD
   AND DCP.SRE_COD = SRE.SRE_COD
   AND DCP.PRF_COD = PRF.PRF_COD
ORDER BY DCP.DCP_NOM
```

```
unit untdmdRelatorios;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, Db, DBTables;

type
  TdmdRelatorios = class(TDataModule)
    tabRelAlunos: TTable;
    tabRelDisciplinas: TTable;
    tabRelProfessores: TTable;
    tabRelAlunoDisc: TTable;
    dtsRelAlunoDisc: TDataSource;
    qryRelAlunoDisc: TQuery;
    tabRelAlunoDiscALU_COD: TFloatField;
    tabRelAlunoDiscALU_NOM: TStringField;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  dmdRelatorios: TdmdRelatorios;

implementation
```

```
{$R *.DFM}

end.
```

Visões



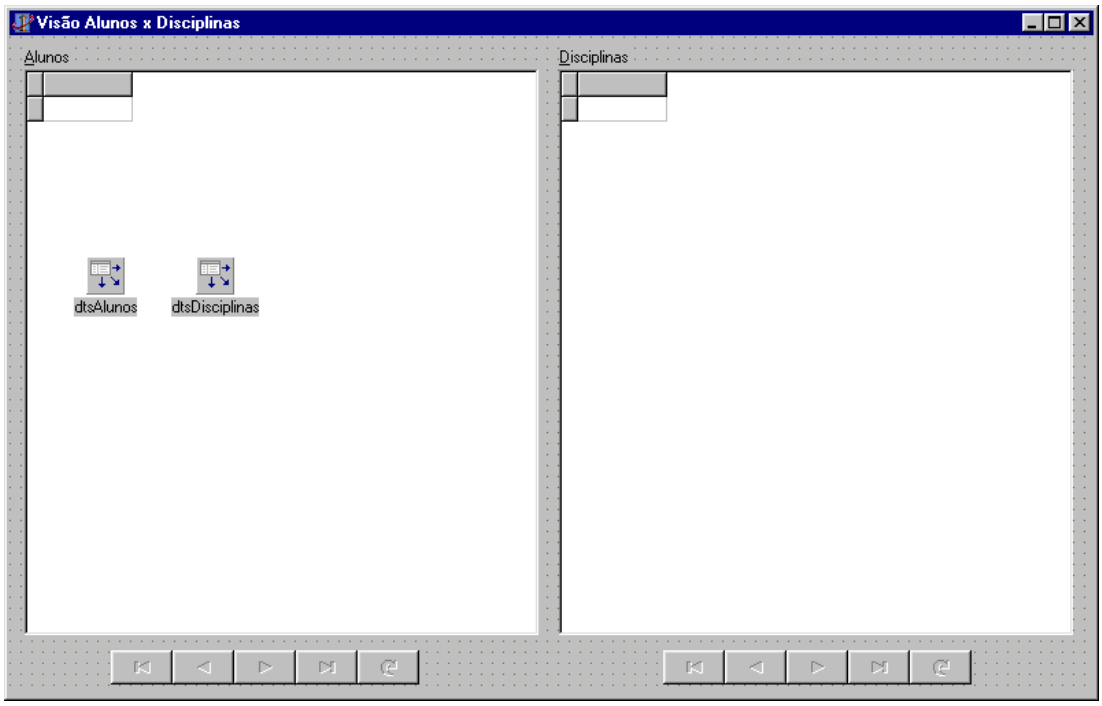
TDataSource

- dtsAlunos
  - DataSet = dmdPrincipal.tabVisAlunoMens

- dtsMensalidades
  - DataSet = dmdPrincipal.qryVisAlunosMens

```
unit untVisAlunosMensalidades;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, ExtCtrls, DBCtrls, StdCtrls, Grids,
  DBGrids, Db, DBTables;

type
  TfrmVisaoAlunosMensalidades = class(TForm)
    dgrVisAlunos: TDBGrid;
    dgrVisMensalidades: TDBGrid;
    Label1: TLabel;
    Label2: TLabel;
    dnvVisaoEmpresas: TDBNavigator;
    dnvVisaoProdutos: TDBNavigator;
    dtsAlunos: TDataSource;
    dtsMensalidades: TDataSource;
```

```
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmVisaoAlunosMensalidades: TfrmVisaoAlunosMensalidades;

implementation

uses
  untdmdPrincipal;

{$R *.DFM}

procedure TfrmVisaoAlunosMensalidades.FormCreate(Sender:
  TObject);
begin
  dmdPrincipal.tabVisAlunoMens.Open;
  dmdPrincipal.qryVisAlunosMens.Open;
end;

procedure TfrmVisaoAlunosMensalidades.FormClose(Sender:
  TObject; var Action: TCloseAction);
begin
  dmdPrincipal.qryVisAlunosMens.Close;
  dmdPrincipal.tabVisAlunoMens.Close;

  Action := caFree;
  frmVisaoAlunosMensalidades := nil;
end;

end.
```



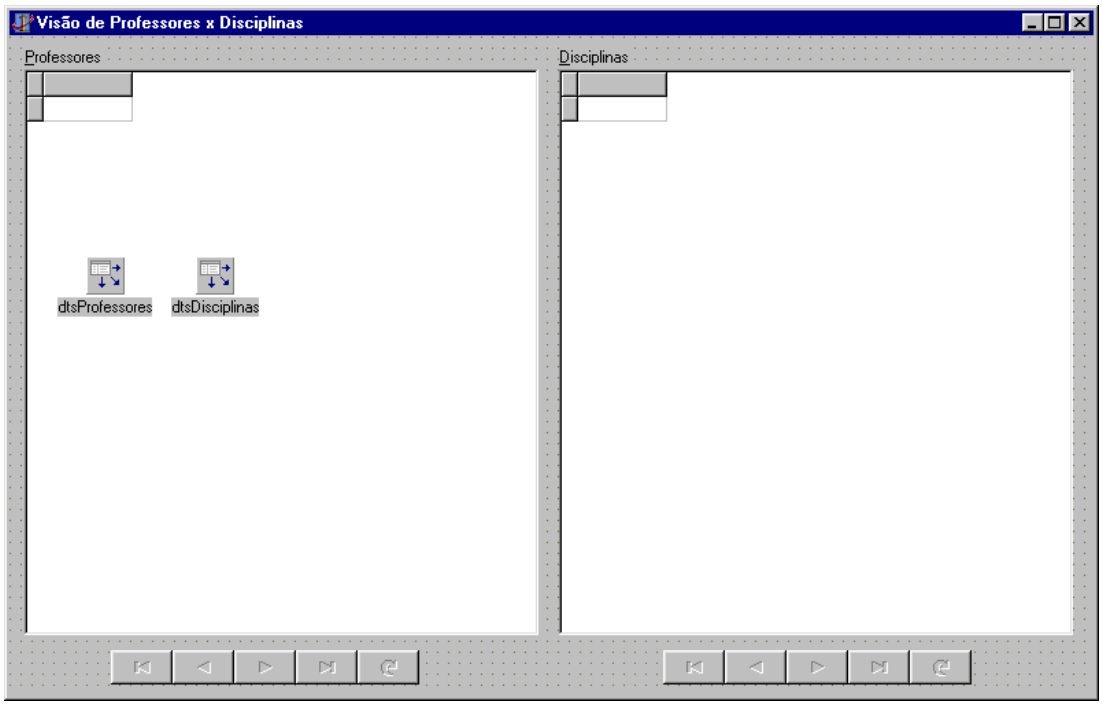
TDataSource

- dtsAlunos
  - DataSet = dmdPrincipal.tabVisAlunoDisc

- dtsDisciplinas
  - DataSet = dmdPrincipal.qryAlunosDisc

```
unit untVisAlunosDisciplinas;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, ExtCtrls, DBCtrls, StdCtrls, Grids,
  DBGrids, Db, DBTables;

type
  TfrmVisAlunosDisciplinas = class(TForm)
    dgrVisaoEmpresas: TDBGrid;
    dgrVisaoProdutos: TDBGrid;
    Label1: TLabel;
    Label2: TLabel;
    dnvVisaoEmpresas: TDBNavigator;
    dnvVisaoProdutos: TDBNavigator;
    dtsAlunos: TDataSource;
    dtsDisciplinas: TDataSource;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
```

```
      TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmVisAlunosDisciplinas: TfrmVisAlunosDisciplinas;

implementation

uses
  untdmdPrincipal;

{$R *.DFM}

procedure TfrmVisAlunosDisciplinas.FormCreate(Sender:
  TObject);
begin
  dmdPrincipal.tabVisAlunoDisc.Open;
  dmdPrincipal.qryAlunosDisc.Open;
end;

procedure TfrmVisAlunosDisciplinas.FormClose(Sender:
  TObject; var Action: TCloseAction);
begin
  dmdPrincipal.qryAlunosDisc.Close;
  dmdPrincipal.tabVisAlunoDisc.Close;

  Action := caFree;
  frmVisAlunosDisciplinas := nil;
end;

end.
```



TDataSource

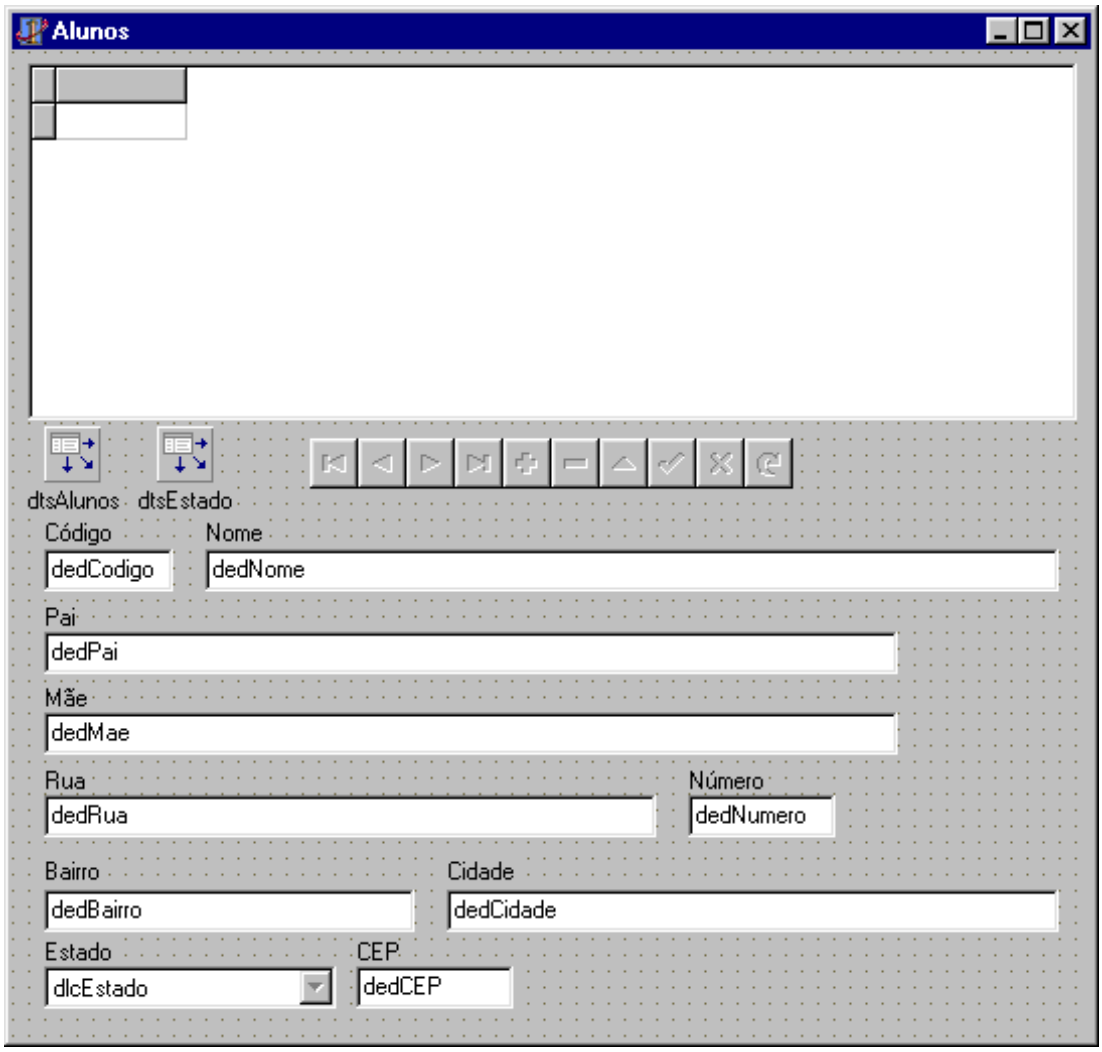
- dtsProfessores
  - DataSet = dmdPrincipal.tabVisProfessores

- dtsDisciplinas
  - DataSet = dmdPrincipal.qryVisProfessoresDisc

```
unit untVisProfessoresDisciplinas;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, ExtCtrls, DBCtrls, StdCtrls, Grids,
  DBGrids, Db, DBTables;

type
  TfrmVisProfessoresDisciplinas = class(TForm)
    dgrVisProfessores: TDBGrid;
    dgrVisDsicpiplinas: TDBGrid;
    Label1: TLabel;
    Label2: TLabel;
    dnvVisaoEmpresas: TDBNavigator;
    dnvVisaoProdutos: TDBNavigator;
    dtsProfessores: TDataSource;
    dtsDisciplinas: TDataSource;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
```

```
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmVisProfessoresDisciplinas:
    TfrmVisProfessoresDisciplinas;

implementation

uses
  untdmdPrincipal;

{$R *.DFM}

procedure TfrmVisProfessoresDisciplinas.FormCreate(Sender:
  TObject);
begin
  dmdPrincipal.tabVisProfessores.Open;
  dmdPrincipal.qryVisProfessoresDisc.Open;
end;

procedure TfrmVisProfessoresDisciplinas.FormClose(Sender:
  TObject; var Action: TCloseAction);
begin
  dmdPrincipal.qryVisProfessoresDisc.Close;
  dmdPrincipal.tabVisProfessores.Close;

  Action := caFree;
  frmVisProfessoresDisciplinas := nil;
end;

end.
```

Cadastro



TDataSource

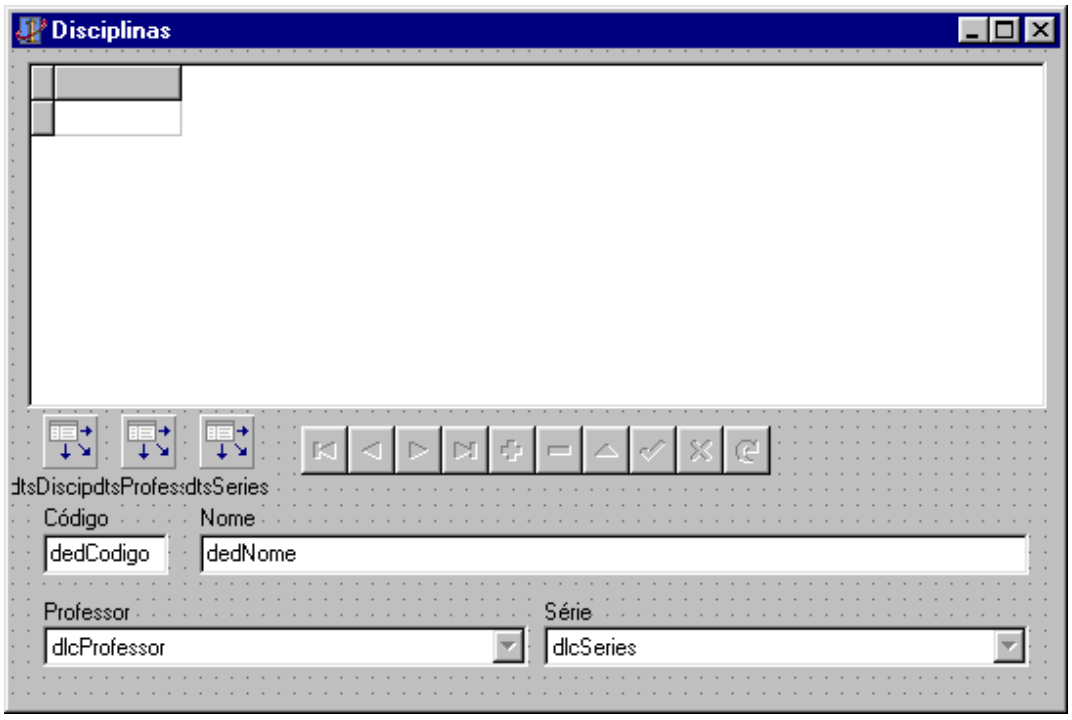
- dtsAlunos
  - DataSet = dmdPrincipal.tabAlunos

- dtsEstado
  - DataSet = dmdPrincipal.tabEstado

```
unit untCadAluno;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, ExtCtrls, DBCtrls, Grids, DBGrids, Db,
  StdCtrls, Mask;

type
```

```
  TfrmAlunos = class(TForm)
    dgrAlunos: TDBGrid;
    dnvAlunos: TDBNavigator;
    dtsAlunos: TDataSource;
    Label1: TLabel;
    dedCodigo: TDBEdit;
    Label2: TLabel;
    dedNome: TDBEdit;
    Label3: TLabel;
    dedPai: TDBEdit;
    Label4: TLabel;
    dedMae: TDBEdit;
    Label5: TLabel;
    dedRua: TDBEdit;
    Label6: TLabel;
    dedNumero: TDBEdit;
    Label7: TLabel;
    dedBairro: TDBEdit;
    Label8: TLabel;
    dedCidade: TDBEdit;
    Label9: TLabel;
    Label10: TLabel;
    dedCEP: TDBEdit;
    dlcEstado: TDBLookupComboBox;
    dtsEstado: TDataSource;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure dnvAlunosClick(Sender: TObject; Button:
      TNavigateBtn);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmAlunos: TfrmAlunos;

implementation

uses
  untdmdPrincipal;

{$R *.DFM}

procedure TfrmAlunos.FormCreate(Sender: TObject);
begin
  dmdPrincipal.tabEstado.Open;
  dmdPrincipal.tabAlunos.Open;
end;
```

```
procedure TfrmAlunos.FormClose(Sender: TObject; var Action:
  TCloseAction);
begin
  dmdPrincipal.tabAlunos.Close;
  dmdPrincipal.tabEstado.Close;

  Action := caFree;
  frmAlunos := nil;
end;

procedure TfrmAlunos.dnvAlunosClick(Sender: TObject;
  Button: TNavigateBtn);
begin
  case (Button) of
    nbInsert :
      if (dedCodigo.CanFocus) then
        dedCodigo.SetFocus;
  end;
end;

end.
```



TDataSource

- dtsDisciplinas
  - DataSet = dmdPrincipal.tabDisciplinas

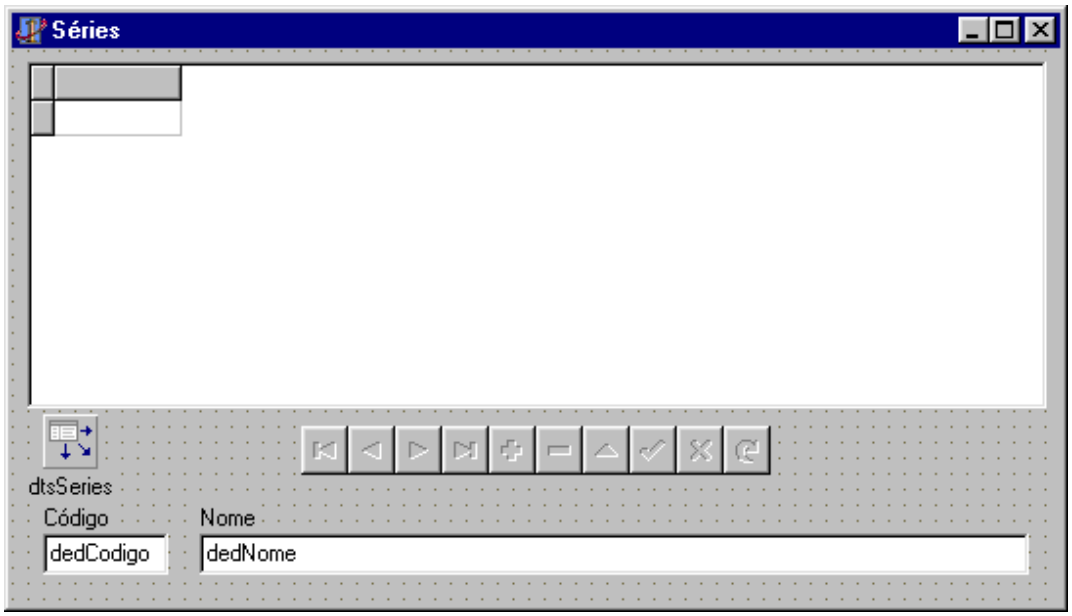
- dtsProfessor
  - DataSet = dmdPrincipal.tabProfessor

- dtsSeries
  - DataSet = dmdPrincipal.tabSeries

```
unit untCadDisciplina;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, ExtCtrls, DBCtrls, Grids, DBGrids, Db,
  StdCtrls, Mask;

type
  TfrmDisciplinas = class(TForm)
    dgrDisciplinas: TDBGrid;
    dnvDisciplinas: TDBNavigator;
    dtsDisciplinas: TDataSource;
    Label1: TLabel;
    dedCodigo: TDBEdit;
    Label2: TLabel;
    dedNome: TDBEdit;
```

```
    Label9: TLabel;
    dlcProfessor: TDBLookupComboBox;
    dtsProfessor: TDataSource;
    Label3: TLabel;
    dlcSeries: TDBLookupComboBox;
    dtsSeries: TDataSource;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure dnvDisciplinasClick(Sender: TObject; Button:
      TNavigateBtn);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmDisciplinas: TfrmDisciplinas;

implementation

uses
  untdmdPrincipal;

{$R *.DFM}

procedure TfrmDisciplinas.FormCreate(Sender: TObject);
begin
  dmdPrincipal.tabSeries.Open;
  dmdPrincipal.tabProfessor.Open;
  dmdPrincipal.tabDisciplinas.Open;
end;

procedure TfrmDisciplinas.FormClose(Sender: TObject; var
  Action: TCloseAction);
begin
  dmdPrincipal.tabDisciplinas.Close;
  dmdPrincipal.tabSeries.Close;
  dmdPrincipal.tabProfessor.Close;

  Action := caFree;
  frmDisciplinas := nil;
end;

procedure TfrmDisciplinas.dnvDisciplinasClick(Sender:
  TObject; Button: TNavigateBtn);
begin
  case (Button) of
    nbInsert :
      if (dedCodigo.CanFocus) then
```

```
        dedCodigo.SetFocus;
  end;
end;

end.
```

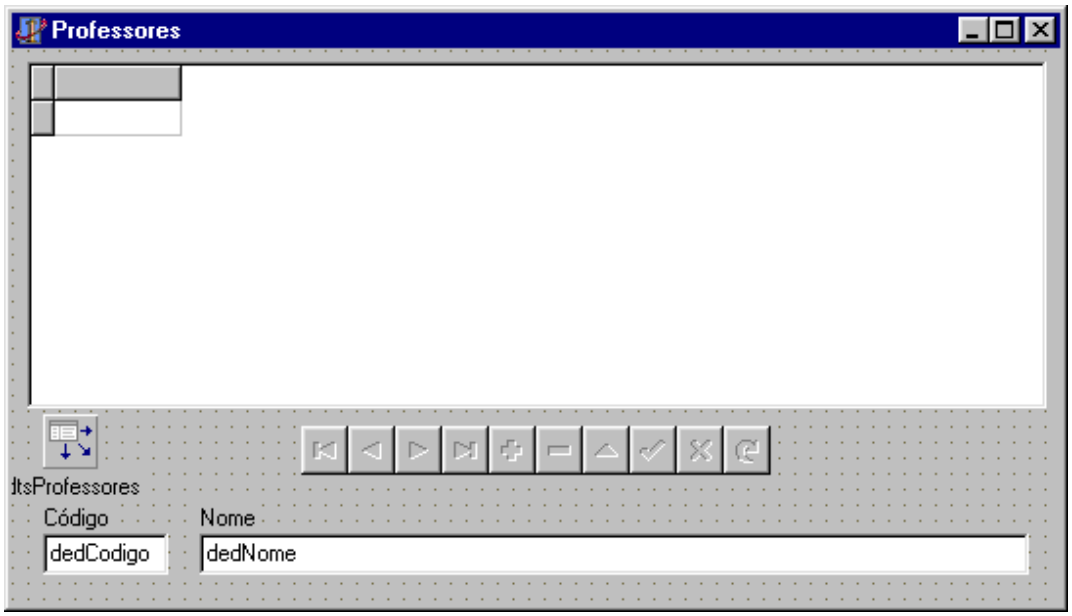


TDataSource

- dtsSeries
  - DataSet = dmdPrincipal.tabSeries

```
unit untCadSerie;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, ExtCtrls, DBCtrls, Grids, DBGrids, Db,
  StdCtrls, Mask;

type
  TfrmSeries = class(TForm)
    dgrAlunos: TDBGrid;
    dnvSeries: TDBNavigator;
    dtsSeries: TDataSource;
    Label1: TLabel;
    dedCodigo: TDBEdit;
    Label2: TLabel;
    dedNome: TDBEdit;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure dnvSeriesClick(Sender: TObject; Button:
      TNavigateBtn);
  private
    { Private declarations }
  public
    { Public declarations }
```

```
  end;

var
  frmSeries: TfrmSeries;

implementation

uses
  untdmdPrincipal;

{$R *.DFM}

procedure TfrmSeries.FormCreate(Sender: TObject);
begin
  dmdPrincipal.tabSeries.Open;
end;

procedure TfrmSeries.FormClose(Sender: TObject; var Action:
  TCloseAction);
begin
  dmdPrincipal.tabSeries.Close;

  Action := caFree;
  frmSeries := nil;
end;

procedure TfrmSeries.dnvSeriesClick(Sender: TObject;
  Button: TNavigateBtn);
begin
  case (Button) of
    nbInsert :
      if (dedCodigo.CanFocus) then
        dedCodigo.SetFocus;
  end;
end;

end.
```

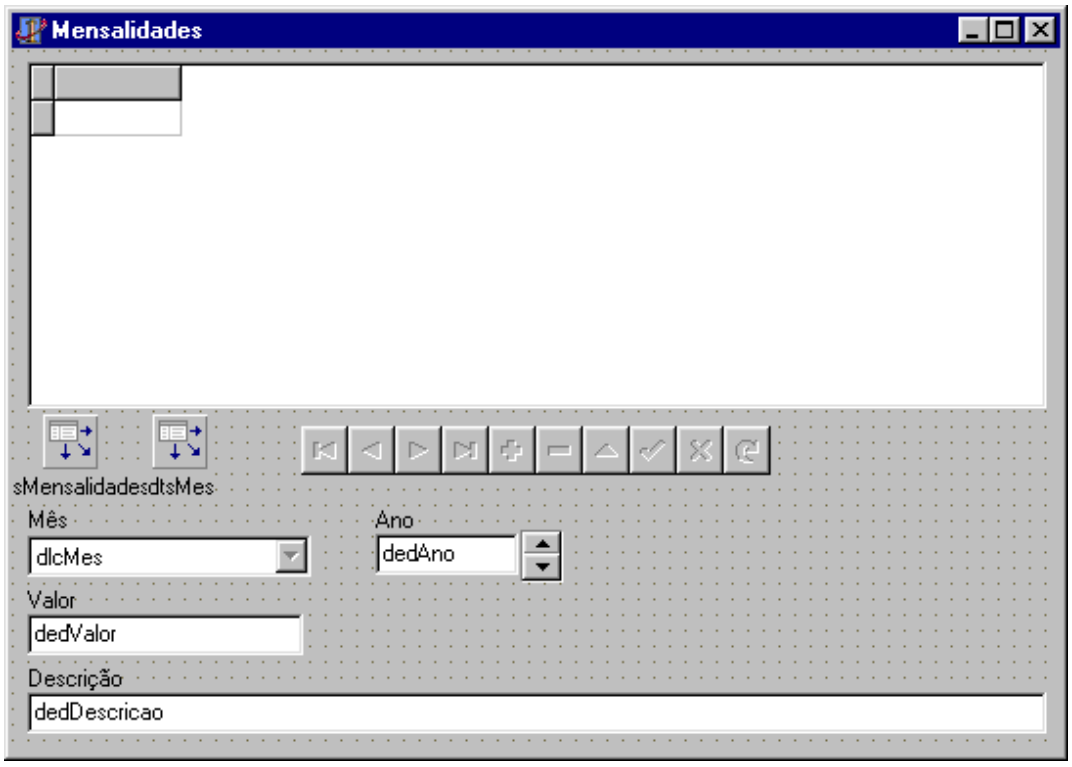


TDataSource

- dtsProfessores
  - DataSet = dmdPrincipal.tabProfessor

```
unit untCadProfessor;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, ExtCtrls, DBCtrls, Grids, DBGrids, Db,
  StdCtrls, Mask;

type
  TfrmProfessores = class(TForm)
    dgrProfessores: TDBGrid;
    dnvProfessores: TDBNavigator;
    dtsProfessores: TDataSource;
    Label1: TLabel;
    dedCodigo: TDBEdit;
    Label2: TLabel;
    dedNome: TDBEdit;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure dnvProfessoresClick(Sender: TObject; Button:
      TNavigateBtn);
  private
    { Private declarations }
  public
    { Public declarations }
```

```
  end;

var
  frmProfessores: TfrmProfessores;

implementation

uses
  untdmdPrincipal;

{$R *.DFM}

procedure TfrmProfessores.FormCreate(Sender: TObject);
begin
  dmdPrincipal.tabProfessor.Open;
end;

procedure TfrmProfessores.FormClose(Sender: TObject; var
  Action: TCloseAction);
begin
  dmdPrincipal.tabProfessor.Close;

  Action := caFree;
  frmProfessores := nil;
end;

procedure TfrmProfessores.dnvProfessoresClick(Sender:
  TObject; Button: TNavigateBtn);
begin
  case (Button) of
    nbInsert :
      if (dedCodigo.CanFocus) then
        dedCodigo.SetFocus;
  end;
end;

end.
```



TDataSource

- dtsMensalidades
  - DataSet = dmdPrincipal.tabMensalidades

- dtsMes
  - DataSet = dmdPrincipal.tabMes

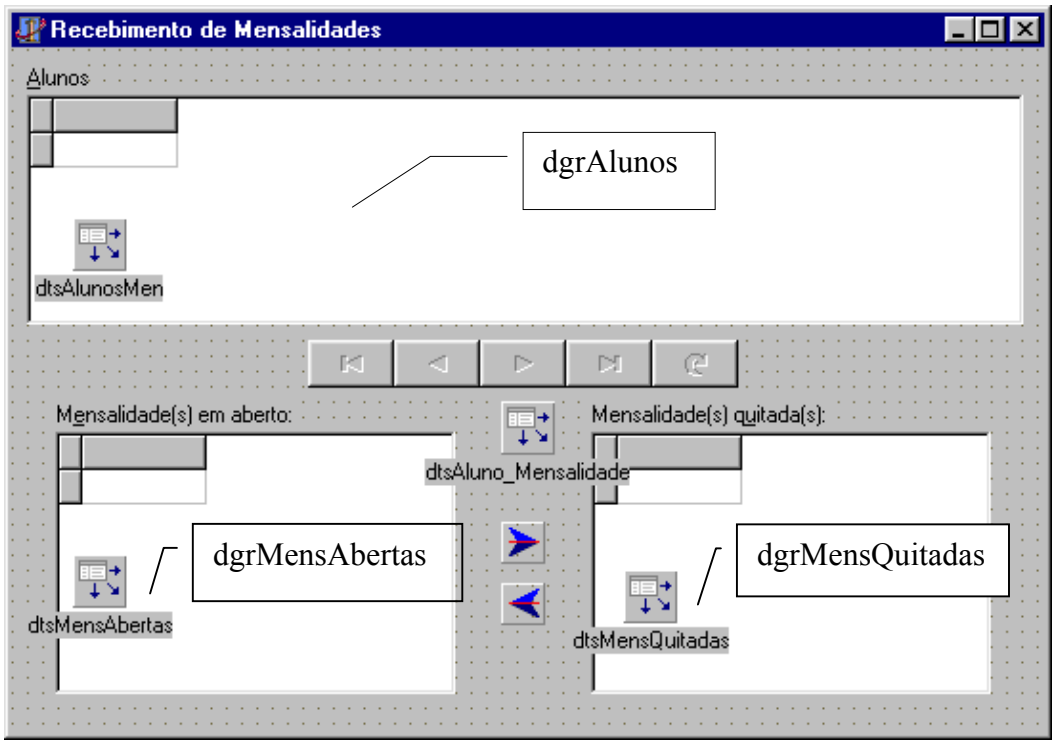
```
unit untCadMensalidade;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, ExtCtrls, DBCtrls, Grids, DBGrids, Db,
  StdCtrls, Mask, Spin;

type
  TfrmMensalidades = class(TForm)
    dgrMensalidades: TDBGrid;
    dnvMensalidades: TDBNavigator;
    dtsMensalidades: TDataSource;
    Label1: TLabel;
    Label2: TLabel;
    dedAno: TDBEdit;
    Label3: TLabel;
    dedValor: TDBEdit;
    Label4: TLabel;
```

```
    dedDescricao: TDBEdit;
    dtsMes: TDataSource;
    dlcMes: TDBLookupComboBox;
    spnAno: TSpinButton;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure spnAnoDownClick(Sender: TObject);
    procedure spnAnoUpClick(Sender: TObject);
    procedure dedAnoExit(Sender: TObject);
    procedure dnvMensalidadesClick(Sender: TObject; Button:
      TNavigateBtn);
    procedure dtsMensalidadesStateChange(Sender: TObject);
  private
    { Private declarations }
    Dia,
    Mes,
    Ano : Word;
  public
    { Public declarations }
  end;

var
  frmMensalidades: TfrmMensalidades;

implementation

uses
  untdmdPrincipal;

{$R *.DFM}

procedure TfrmMensalidades.FormCreate(Sender: TObject);
begin
  dmdPrincipal.tabMes.Open;
  dmdPrincipal.tabMensalidades.Open;

  DecodeDate (Now, Ano, Mes, Dia);

end;

procedure TfrmMensalidades.FormClose(Sender: TObject; var
  Action: TCloseAction);
begin
  dmdPrincipal.tabMensalidades.Close;
  dmdPrincipal.tabMes.Close;

  Action := caFree;
  frmMensalidades := nil;
end;
```

```
procedure TfrmMensalidades.spnAnoDownClick(Sender:
  TObject);
begin
  with (dtsMensalidades.DataSet) do
    if ((FieldByName ('MEN_ANO').IsNull) or (FieldByName
      ('MEN_ANO').AsInteger <= 0)) then
      FieldByName ('MEN_ANO').AsInteger := 0
    else
      FieldByName ('MEN_ANO').AsInteger := FieldByName
        ('MEN_ANO').AsInteger - 1;
end;

procedure TfrmMensalidades.spnAnoUpClick(Sender: TObject);
begin
  with (dtsMensalidades.DataSet) do
    if ((FieldByName ('MEN_ANO').IsNull) or (FieldByName
      ('MEN_ANO').AsInteger = 0)) then
      FieldByName ('MEN_ANO').AsInteger := 1
    else
      FieldByName ('MEN_ANO').AsInteger := FieldByName
        ('MEN_ANO').AsInteger + 1;
end;

procedure TfrmMensalidades.dedAnoExit(Sender: TObject);
begin
  if ((dtsMensalidades.DataSet.FieldByName
    ('MEN_ANO').AsInteger > 2010) or
    (dtsMensalidades.DataSet.FieldByName
    ('MEN_ANO').AsInteger < 1980)) then
    if (MessageDlg('A ano que você colocou possui uma
      diferença maior ou menor que 10 '+#13+#10+
      'anos da data atual. Esta data está correta ?',
      mtConfirmation, [mbYes, mbNo], 0) = mrNo) then
      dedAno.SetFocus;
end;

procedure TfrmMensalidades.dnvMensalidadesClick(Sender:
  TObject; Button: TNavigateBtn);
begin
  case (Button) of
    nbInsert :
      if (dlcMes.CanFocus) then
        dlcMes.SetFocus;
  end;
end;

procedure
  TfrmMensalidades.dtsMensalidadesStateChange(Sender:
  TObject);
begin
  with (dtsMensalidades.DataSet) do
```

```
    if (State = dsInsert) then
    begin
      FieldByName ('MEN_ANO').AsInteger := Ano;
      FieldByName ('MES_COD').AsInteger := Mes;
    end;
end;

end.
```

Movimento



TDataSource

- dtsAlunosMen
  - DataSet = dmdMovimento.tabAlunos
- dtsMensAbertas
  - DataSet = dmdMovimento.qryMensalidadesAbertas
- dtsMensQuitadas
  - DataSet = dmdMovimento.qryMensalidadesPagas
- dtsAluno_Mensalidade
  - DataSet = dmdMovimento.tabAlunosMensalidades

TDBGrid

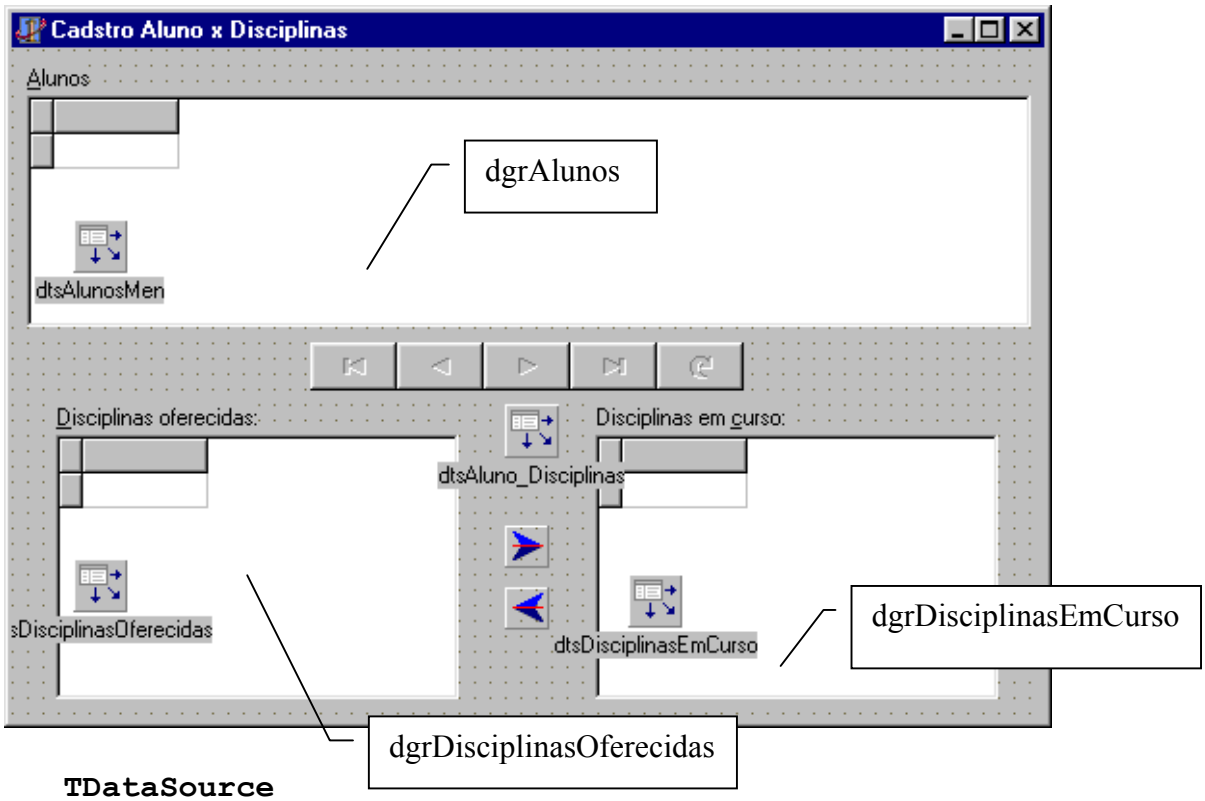
- dgrAlunos
  - DataSource = dtsAlunosMen
- dgrMensAbertas
  - DataSource = dtsMensAbertas
- dgrMensQuitadas
  - DataSource = dtsMensQuitadas

```
unit untRecebimentoMensalidades;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Grids, DBGrids, Db, ExtCtrls,
  DBCtrls, Buttons;

type
  TfrmRecebimentoMensalidades = class(TForm)
    dgrAlunos: TDBGrid;
    Label1: TLabel;
    dnvAlunosMen: TDBNavigator;
    dtsAlunosMen: TDataSource;
    dgrMensAbertas: TDBGrid;
    dgrMensQuitadas: TDBGrid;
    sbnDesfazPagamento: TSpeedButton;
    sbnFazPagamento: TSpeedButton;
    Label2: TLabel;
    Label3: TLabel;
    dtsMensAbertas: TDataSource;
    dtsMensQuitadas: TDataSource;
    dtsAluno_Mensalidade: TDataSource;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure sbnFazPagamentoClick(Sender: TObject);
    procedure sbnDesfazPagamentoClick(Sender: TObject);
    procedure dtsMensQuitadasStateChange(Sender: TObject);
    procedure dtsMensAbertasStateChange(Sender: TObject);
    procedure dtsAlunosMenDataChange(Sender: TObject;
      Field: TField);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmRecebimentoMensalidades: TfrmRecebimentoMensalidades;

implementation

uses
  untdmdMovimento;

{$R *.DFM}

procedure TfrmRecebimentoMensalidades.FormCreate(Sender:
```

```
  TObject);
begin
  dmdMovimento.tabAlunos.Open;
  dmdMovimento.qryMensalidadesAbertas.Open;
  dmdMovimento.qryMensalidadesPagas.Open;
end;

procedure TfrmRecebimentoMensalidades.FormClose(Sender:
  TObject; var Action: TCloseAction);
begin
  dmdMovimento.qryMensalidadesPagas.Close;
  dmdMovimento.qryMensalidadesAbertas.Close;
  dmdMovimento.tabAlunos.Close;

  Action := caFree;
  frmRecebimentoMensalidades := nil;
end;

procedure
  TfrmRecebimentoMensalidades.sbnFazPagamentoClick(Sender:
  TObject);
begin
  with (dtsAluno_Mensalidade.DataSet) do
  begin
    Open;
    Append;
    FieldByName ('ALU_COD').AsInteger :=
      dtsAlunosMen.DataSet.FieldByName
      ('ALU_COD').AsInteger;
    FieldByName ('MEN_ANO').AsInteger :=
      dtsMensAbertas.DataSet.FieldByName
      ('MEN_ANO').AsInteger;
    FieldByName ('MES_COD').AsInteger :=
      dtsMensAbertas.DataSet.FieldByName
      ('MES_COD').AsInteger;
    Post;
    Close;
  end;

  { Refresh nos dados que estão sendo visualizados. }
  dtsMensAbertas.DataSet.Close;
  dtsMensAbertas.DataSet.Open;
  dtsMensQuitadas.DataSet.Close;
  dtsMensQuitadas.DataSet.Open;
end;

procedure
TfrmRecebimentoMensalidades.sbnDesfazPagamentoClick(Sender:
  TObject);
begin
  with (dtsAluno_Mensalidade.DataSet) do
```

```
  begin
    Open;
    if (Locate ('ALU_COD; MEN_ANO; MES_COD', VarArrayOf ([
      dtsAlunosMen.DataSet.FieldByName
      ('ALU_COD').AsInteger,
      dtsMensQuitadas.DataSet.FieldByName
      ('MEN_ANO').AsInteger,
      dtsMensQuitadas.DataSet.FieldByName
      ('MES_COD').AsInteger]), [])) then
      Delete
    else
      MessageDlg('O sistema está apresentando
        inconsistências nos registros '+#13+#10+
        'apresentados. Favor reinicializar seu sistema, se
        isto não resolver o '+#13+#10+
        'problema favor entrar em contato com o fabricante
        do software.', mtWarning, [mbOK], 0);
    Close;
  end;

  { Refresh nos dados que estão sendo visualizados. }
  dtsMensAbertas.DataSet.Close;
  dtsMensAbertas.DataSet.Open;
  dtsMensQuitadas.DataSet.Close;
  dtsMensQuitadas.DataSet.Open;
end;

procedure
  TfrmRecebimentoMensalidades.dtsMensQuitadasStateChange(
  Sender: TObject);
begin
  sbnDesfazPagamento.Enabled := (not
    (dtsMensQuitadas.DataSet.IsEmpty));
end;

procedure
  TfrmRecebimentoMensalidades.dtsMensAbertasStateChange(
  Sender: TObject);
begin
  sbnFazPagamento.Enabled := (not
    (dtsMensAbertas.DataSet.IsEmpty));
end;

procedure
 TfrmRecebimentoMensalidades.dtsAlunosMenDataChange(Sender:
  TObject; Field: TField);
begin
  sbnDesfazPagamento.Enabled := (not
    (dtsMensQuitadas.DataSet.IsEmpty));
  sbnFazPagamento.Enabled := (not
    (dtsMensAbertas.DataSet.IsEmpty));
```

```
end;

end.
```



TDataSource

- dtsAlunosMen
  - DataSet = dmdMovimento.tabAlunosDisc
- dtsDisciplinasOferecidas
  - DataSet = dmdMovimento.qryDisciplinasOferecidas
- dtsDisciplinasEmCurso
  - DataSet = dmdMovimento.qryDisciplinasEmCurso
- dtsAluno_Disciplinas
  - DataSet = dmdMovimento.tabAlunosDisciplinas

TDBGrid

- dgrAlunos
  - DataSource = dtsAlunosMen
- dgrDisciplinasOferecidas
  - DataSource = dtsDisciplinasOferecidas
- dgrDisciplinasEmCurso
  - DataSource = dtsDisciplinasEmCurso

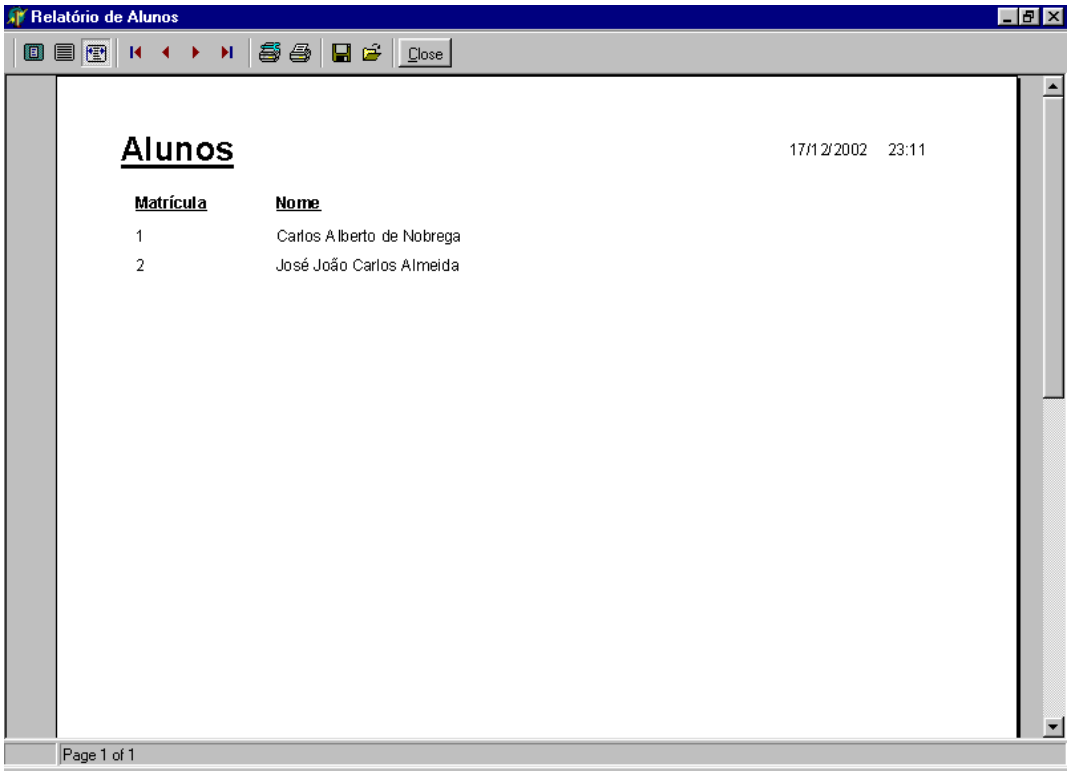
```
unit untAlunoDisciplinas;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, Buttons, ExtCtrls, DBCtrls, Db, Grids,
  DBGrids, StdCtrls;

type
  TfrmAlunoDisciplinas = class(TForm)
    Label1: TLabel;
    dgrAlunos: TDBGrid;
    dtsAlunosMen: TDataSource;
    dnvAlunosMen: TDBNavigator;
    dgrDisciplinasOferecidas: TDBGrid;
    dtsAluno_Disciplinas: TDataSource;
    dgrDisciplinasEmCurso: TDBGrid;
    dtsDisciplinasEmCurso: TDataSource;
    Label3: TLabel;
    Label2: TLabel;
    sbnCursaDisciplina: TSpeedButton;
    sbnNaoCursaDisciplina: TSpeedButton;
    dtsDisciplinasOferecidas: TDataSource;
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure FormCreate(Sender: TObject);
    procedure sbnCursaDisciplinaClick(Sender: TObject);
    procedure sbnNaoCursaDisciplinaClick(Sender: TObject);
    procedure dtsDisciplinasOferecidasStateChange(Sender:
      TObject);
    procedure dtsDisciplinasEmCursoStateChange(Sender:
      TObject);
    procedure dtsAlunosMenDataChange(Sender: TObject;
      Field: TField);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmAlunoDisciplinas: TfrmAlunoDisciplinas;


implementation


uses
  untdmdMovimento;
```

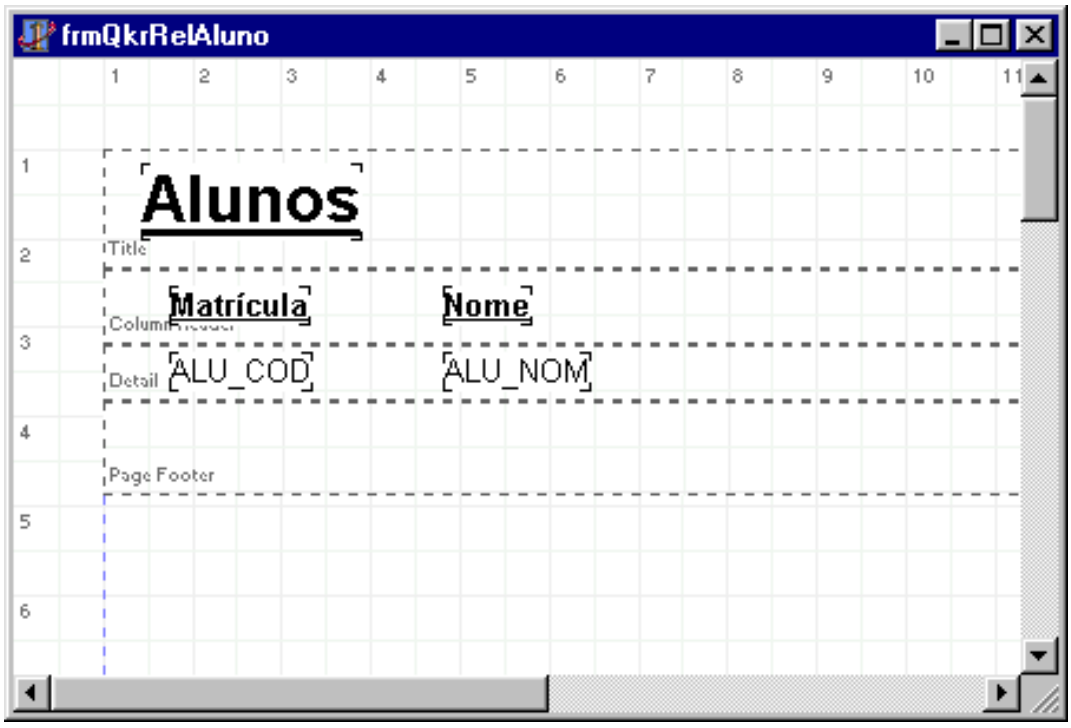
```
{$R *.DFM}

procedure TfrmAlunoDisciplinas.FormCreate(Sender: TObject);
begin
  dmdMovimento.tabAlunosDisc.Open;
  dmdMovimento.qryDisciplinasOferecidas.Open;
  dmdMovimento.qryDisciplinasEmCurso.Open;
end;

procedure TfrmAlunoDisciplinas.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  dmdMovimento.qryDisciplinasEmCurso.Close;
  dmdMovimento.qryDisciplinasOferecidas.Close;
  dmdMovimento.tabAlunosDisc.Close;

  Action := caFree;
  frmAlunoDisciplinas := nil;
end;

procedure
  TfrmAlunoDisciplinas.sbnCursaDisciplinaClick(Sender:
  TObject);
begin
  with (dtsAluno_Disciplinas.DataSet) do
  begin
    Open;
    Append;
    FieldByName ('ALU_COD').AsInteger :=
      dtsAlunosMen.DataSet.FieldByName
      ('ALU_COD').AsInteger;
    FieldByName ('DCP_COD').AsInteger :=
      dtsDisciplinasOferecidas.DataSet.FieldByName
      ('DCP_COD').AsInteger;
    Post;
    Close;
  end;

  { Refresh nos dados que estão sendo visualizados. }
  dtsDisciplinasOferecidas.DataSet.Close;
  dtsDisciplinasOferecidas.DataSet.Open;
  dtsDisciplinasEmCurso.DataSet.Close;
  dtsDisciplinasEmCurso.DataSet.Open;
end;

procedure
  TfrmAlunoDisciplinas.sbnNaoCursaDisciplinaClick(Sender:
  TObject);
begin
  with (dtsAluno_Disciplinas.DataSet) do
```

```
  begin
    Open;
    if (Locate ('ALU_COD; DCP_COD', VarArrayOf ([
      dtsAlunosMen.DataSet.FieldByName
      ('ALU_COD').AsInteger,
      dtsDisciplinasEmCurso.DataSet.FieldByName
      ('DCP_COD').AsInteger]), [])) then
      Delete
    else
      MessageDlg('O sistema está apresentando
        inconsistências nos registros '+#13+#10+
        'apresentados. Favor reinicializar seu sistema, se
        isto não resolver o '+#13+#10+
        'problema favor entrar em contato com o fabricante
        do software.', mtWarning, [mbOK], 0);
    Close;
  end;

  { Refresh nos dados que estão sendo visualizados. }
  dtsDisciplinasOferecidas.DataSet.Close;
  dtsDisciplinasOferecidas.DataSet.Open;
  dtsDisciplinasEmCurso.DataSet.Close;
  dtsDisciplinasEmCurso.DataSet.Open;
end;

procedure
  TfrmAlunoDisciplinas.dtsDisciplinasOferecidasStateChange(
  Sender: TObject);
begin
  sbnCursaDisciplina.Enabled := (not
    (dtsDisciplinasOferecidas.DataSet.IsEmpty));
end;

procedure
  TfrmAlunoDisciplinas.dtsDisciplinasEmCursoStateChange(
  Sender: TObject);
begin
  sbnNaoCursaDisciplina.Enabled := (not
    (dtsDisciplinasEmCurso.DataSet.IsEmpty));
end;

procedure
  TfrmAlunoDisciplinas.dtsAlunosMenDataChange(Sender:
  TObject; Field: TField);
begin
  sbnCursaDisciplina.Enabled := (not
    (dtsDisciplinasOferecidas.DataSet.IsEmpty));
  sbnNaoCursaDisciplina.Enabled := (not
    (dtsDisciplinasEmCurso.DataSet.IsEmpty));
end;
```

```
end.
```

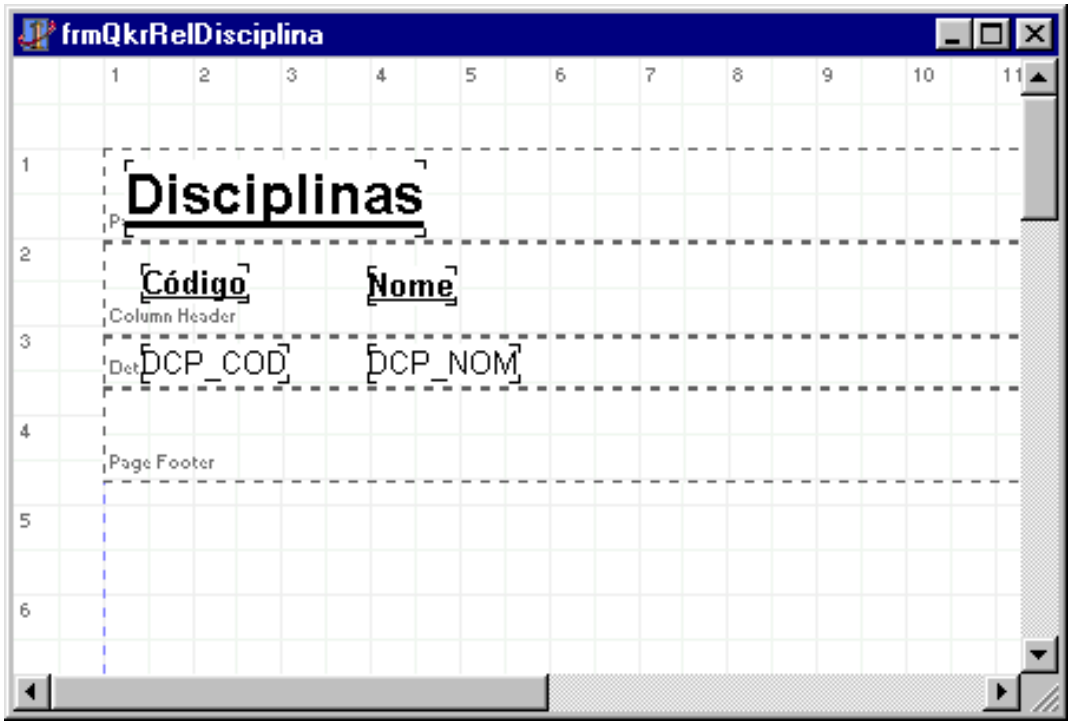
Relatórios:





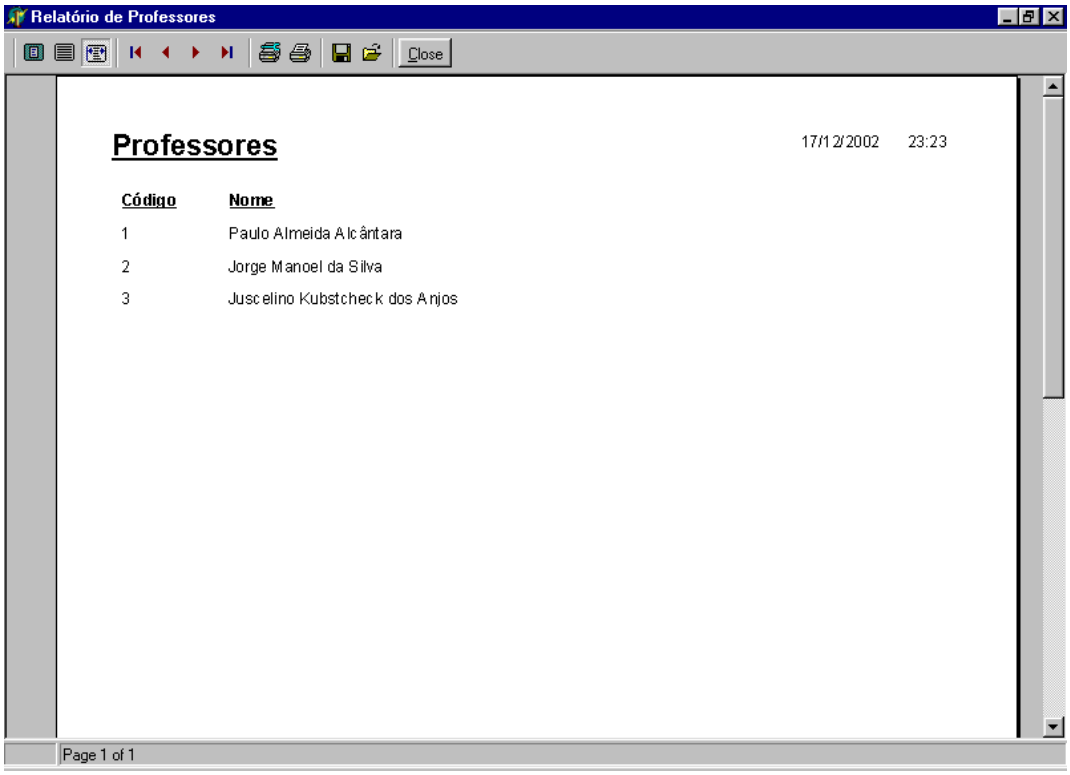

```
unit untQkrRelAluno;

interface

uses
```

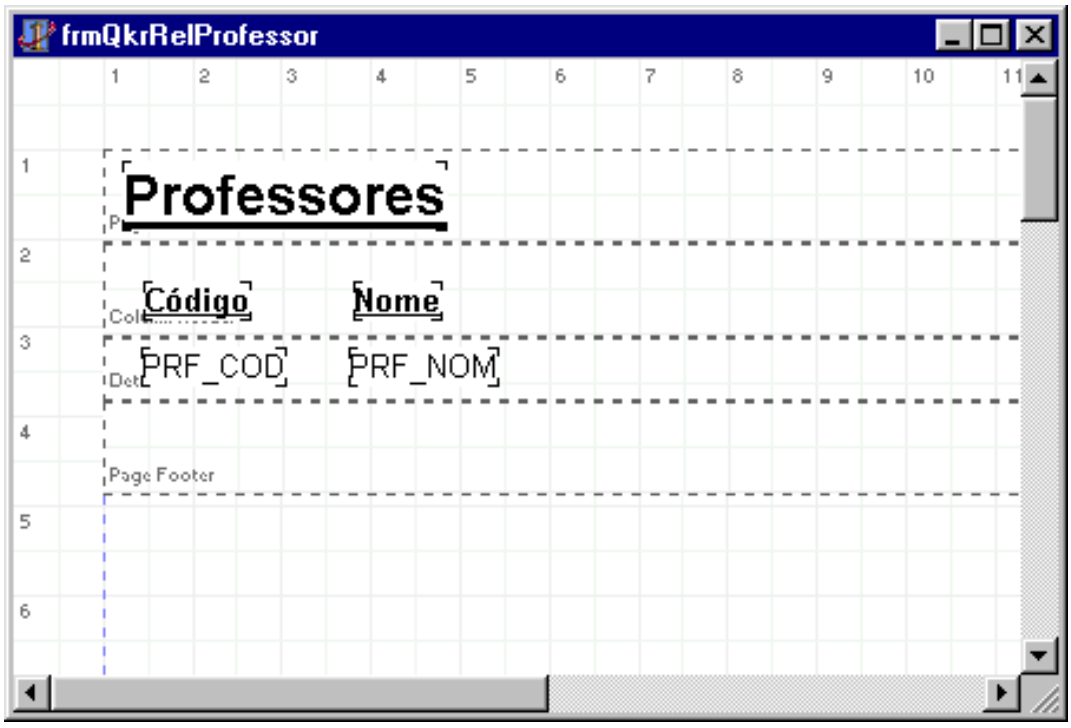
```
  Windows, SysUtils, Messages, Classes, Graphics, Controls,
  StdCtrls, ExtCtrls, Forms, Quickrpt, QRCtrls;

type
  TfrmQkrRelAluno = class(TQuickRep)
    QRBand1: TQRBand;
    QRLabel1: TQRLabel;
    QRSysData2: TQRSysData;
    QRBand2: TQRBand;
    QRDBText2: TQRDBText;
    QRDBText1: TQRDBText;
    QRBand3: TQRBand;
    QRSysData3: TQRSysData;
    ColumnHeaderBand1: TQRBand;
    QRLabel2: TQRLabel;
    QRLabel3: TQRLabel;
    QRSysData1: TQRSysData;
    QRSysData4: TQRSysData;
  private

  public

  end;

var
  frmQkrRelAluno: TfrmQkrRelAluno;

implementation

uses
  untdmdRelatorios;

{$R *.DFM}

end.
```




```
unit untQkrRelDisciplina;

interface

uses
  Windows, SysUtils, Messages, Classes, Graphics, Controls,
```

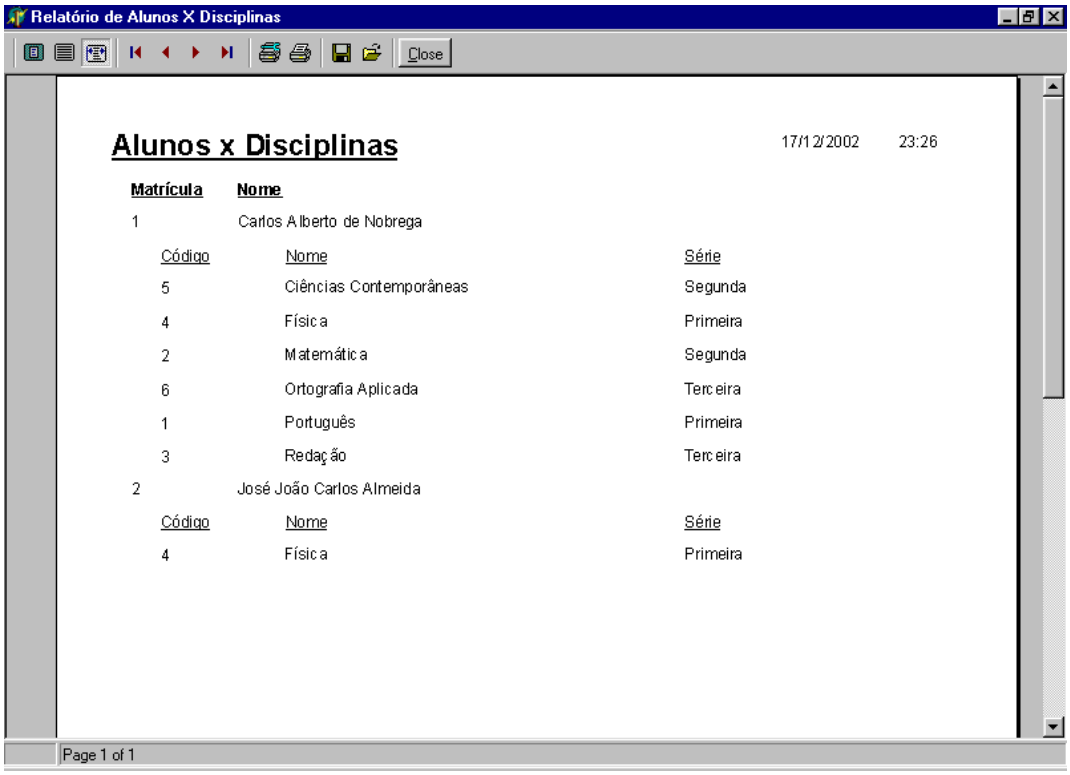
```
  StdCtrls, ExtCtrls, Forms, Quickrpt, QRCtrls;

type
  TfrmQkrRelDisciplina = class(TQuickRep)
    ColumnHeaderBand1: TQRBand;
    QRLabel2: TQRLabel;
    QRLabel3: TQRLabel;
    DetailBand1: TQRBand;
    QRDBText1: TQRDBText;
    QRDBText2: TQRDBText;
    PageFooterBand1: TQRBand;
    QRSysData3: TQRSysData;
    PageHeaderBand1: TQRBand;
    QRLabel1: TQRLabel;
    QRSysData1: TQRSysData;
    QRSysData2: TQRSysData;
  private

  public

  end;

var
  frmQkrRelDisciplina: TfrmQkrRelDisciplina;

implementation

uses
  untdmdRelatorios;

{$R *.DFM}

end.
```

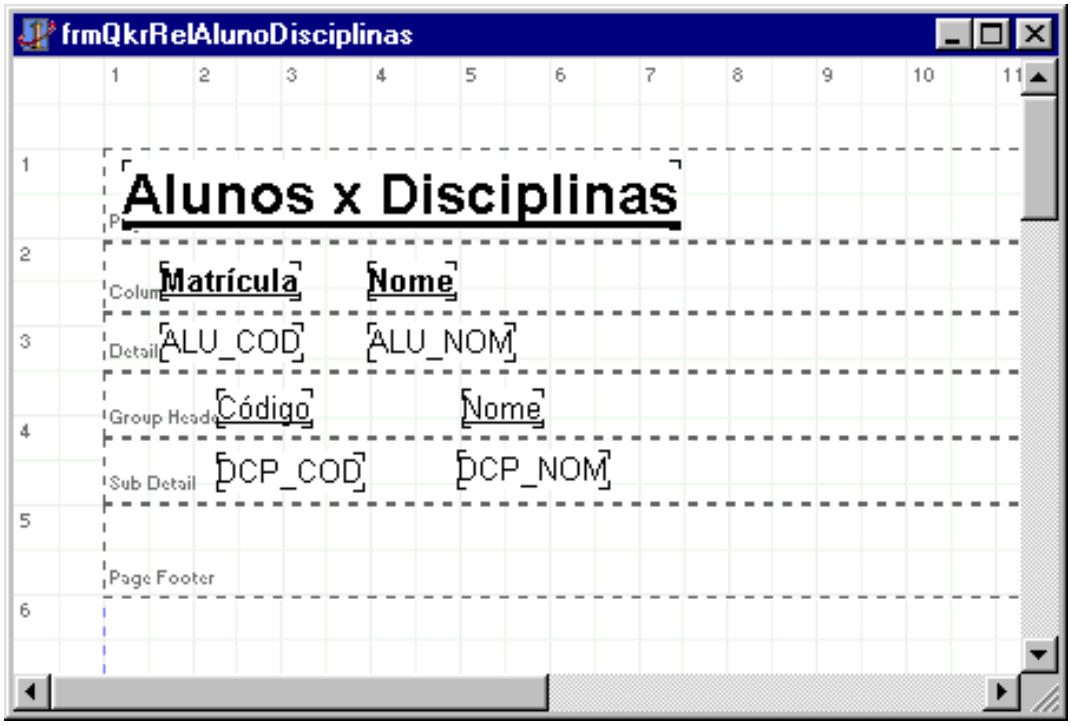





```
unit untQkrRelProfessor;

interface

uses
  Windows, SysUtils, Messages, Classes, Graphics, Controls,
  StdCtrls, ExtCtrls, Forms, Quickrpt, QRCtrls;
```

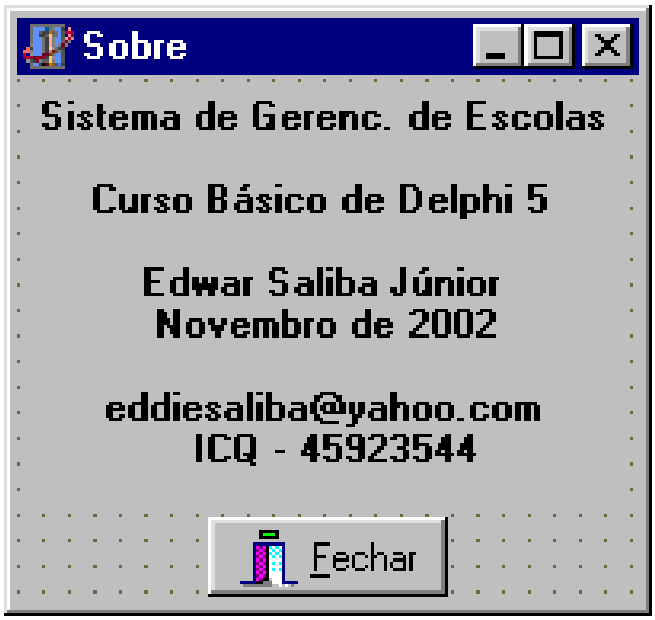
```
type
  TfrmQkrRelProfessor = class(TQuickRep)
    ColumnHeaderBand1: TQRBand;
    QRLabel2: TQRLabel;
    QRLabel3: TQRLabel;
    DetailBand1: TQRBand;
    QRDBText1: TQRDBText;
    QRDBText2: TQRDBText;
    PageFooterBand1: TQRBand;
    QRSysData3: TQRSysData;
    PageHeaderBand1: TQRBand;
    QRLabel1: TQRLabel;
    QRSysData1: TQRSysData;
    QRSysData2: TQRSysData;
  private

  public

  end;

var
  frmQkrRelProfessor: TfrmQkrRelProfessor;

implementation

uses
  untdmdRelatorios;

{$R *.DFM}

end.
```






```
unit untQkrRelAlunoDisciplinas;

interface

uses
```

```
  Windows, SysUtils, Messages, Classes, Graphics, Controls,
  StdCtrls, ExtCtrls, Forms, Quickrpt, QRCtrls;

type
  TfrmQkrRelAlunoDisciplinas = class(TQuickRep)
    ColumnHeaderBand1: TQRBand;
    QRLabel2: TQRLabel;
    QRLabel3: TQRLabel;
    DetailBand1: TQRBand;
    QRDBText2: TQRDBText;
    QRDBText5: TQRDBText;
    PageFooterBand1: TQRBand;
    QRSysData3: TQRSysData;
    PageHeaderBand1: TQRBand;
    QRLabel1: TQRLabel;
    QRSysData1: TQRSysData;
    QRSysData2: TQRSysData;
    QRSubDetail1: TQRSubDetail;
    QRDBText1: TQRDBText;
    QRDBText3: TQRDBText;
    QRDBText4: TQRDBText;
    TitleBand1: TQRBand;
    QRLabel4: TQRLabel;
    QRLabel5: TQRLabel;
    QRLabel6: TQRLabel;
  private

  public

  end;

var
  frmQkrRelAlunoDisciplinas: TfrmQkrRelAlunoDisciplinas;

implementation

uses
  untdmdRelatorios;

{$R *.DFM}

end.
```




```
unit untSobre;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Buttons;

type
  TfrmSobre = class(TForm)
    mnoSobre: TMemo;
    bbnFechar: TBitBtn;
    procedure FormClose(Sender: TObject; var Action:
      TCloseAction);
    procedure bbnFecharClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmSobre: TfrmSobre;

implementation

{$R *.DFM}

procedure TfrmSobre.FormClose(Sender: TObject; var Action:
```

```
  TCloseAction);
begin
  Action := caFree;
  frmSobre := nil;
end;

procedure TfrmSobre.bbnFecharClick(Sender: TObject);
begin
  Close;
end;

end.
```