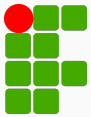
Lazarus

04 - Componentes Avançados e Criação de Interfaces no Lazarus

Marcos Roberto Ribeiro



Instituto Federal Minas Gerais - Campus Bambuí

2018

Introdução

- Nesta aula trataremos de alguns componentes avançados do Lazarus e também introduziremos algumas técnicas para construção de formulário com interfaces de maior usabilidade.
- Os componentes abordados serão

  Image: Componente para exibir imagens;

  StringGrid: Grade semelhante a uma planilha;

  Dialogs: Caixas de diálogo;

  SpinEdit: Seletor de números inteiros;

  Splitter: Permite que o usuário altere o tamanho de componentes com alinhamento relativo;

O Componente *Image*

- O componente *Image* é uma caixa para exibição de figuras, suas propriedades mais importantes são:
  - **AutoSize:** Redimensiona o componente para o tamanho da imagem;
  - **Center:** Centraliza a figura no componente;
  - **Picture:** Figura a ser exibida;
  - **Stretch:** Redimensiona a figura para o tamanho do componente;

Acessando figuras em Disco

A propriedade *Picture* na verdade é um objeto da classe *TPicture* e possui os métodos *LoadFromFile* e *SaveToFile* que permitem ler e escrever a figura em disco, respectivamente.

Exemplo com *Image*


```
procedure TFormPrincipal.FormCreate(Sender: TObject);
begin
  ImagePrincipal.Picture := ImageTux.Picture;
  atual := 0;
end;

procedure TFormPrincipal.ImagePrincipalClick(Sender:
 ↪   TObject);
begin
  if atual=1 then
  begin
    atual := 0;
    ImagePrincipal.Picture := ImageTux.Picture;
  end
  else
  begin
    atual := 1;
    ImagePrincipal.Picture := ImageKonqi.Picture;
  end;
end;
```

O Componente *StringGrid* I

- O componente *StringGrid* é uma grade muito utilizada para visualização de matrizes e planilhas. Vamos destacar as seguintes propriedades:

  ColCount: Número de colunas;

  Columns: Esta propriedade é um objeto *TGridColumns*. Com ela podemos personalizar cada coluna como veremos mais adiante;

  FixedCols: Número de colunas fixas (que o usuário não pode alterar);

  FixedRows: Número de linhas fixas;

  Name: Nome (abreviação *Grid*);

  RowCount: Número de linhas;

  Cells: Matriz que representa as células do *StringGrid*. Uma célula na linha L e coluna C é acessada por meio de *Cells[C,L]*.

O Componente *StringGrid* II

Options: Esta propriedades é composta por várias sub-propriedades booleanas que realizam ajuste finos sobre o *StringGrid*. As principais são:

goAlwaysShowEditor: Sempre mostrar editor;

goColMoving: Permite mover colunas;

goColSizing: Permite redimensionar colunas;

goDblClickAutoSize: Redimensiona com duplo clique;

doEditing: Permite editar;

goRowMoving: Permite mover linhas;

goRowSizing: Permite redimensionar linas;

- Principais eventos:

EditingDone: Quando uma edição em uma célula é finalizada;

GetEditText: Ao iniciar a edição do texto de uma célula;

SelectCell: Ao selecionar uma célula;

O Componente *StringGrid* III

SetEditText: Quando ocorre uma edição (mesmo que não seja finalizada);

- Muitos destes eventos possuem os parâmetros *ACol* e *ARow* que representam a linha e a coluna da célula, respectivamente;

O Componente *StringGrid* IV

Personalizando as colunas

Como mencionamos, a propriedade *Columns* é uma objeto da classe *TGridColumns* que possui um vetor de colunas que podem ser personalizadas. Quando alteramos esta propriedade pelo *Object Inspector* o Lazarus exibe um editor visual que nos permite manipular as colunas. Cada coluna é um objeto da classe *TGridColumn*. Suas propriedades mais relevantes são:

ButtonStyle: Define o estilo da coluna. As possibilidades interessantes são *cbsCheckBoxColumn* (formato de *CheckBox*) e *cbsPickList* (formato de *ComboBox*);

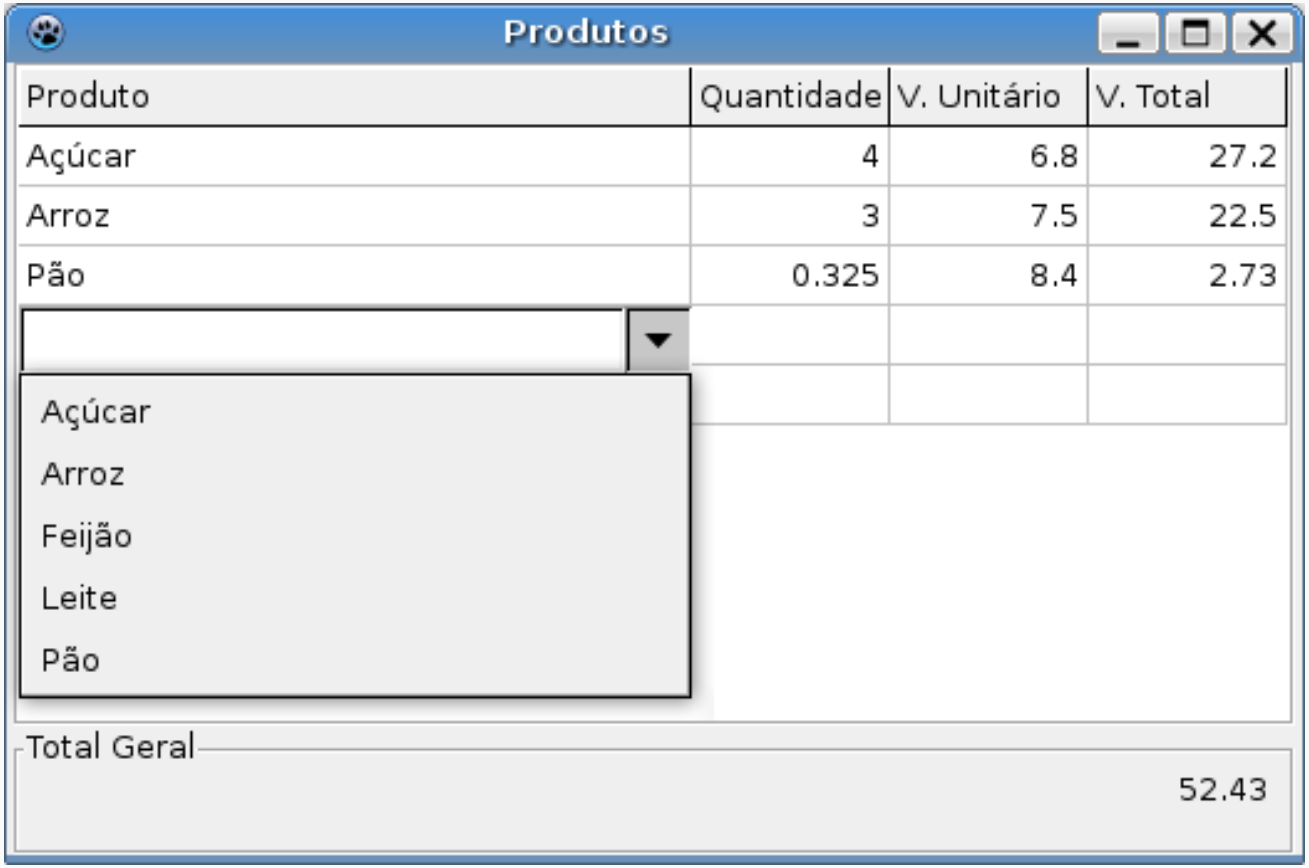
Title: Título;

PickList: Lista de valores para o estilo *ComboBox*.

Exemplo com *StringGrid*

Edição Visual do *StringGrid*

Neste exemplo não utilizaremos, mas o Lazarus possui um editor de *StringGrid* que pode ser acessado clicando com o botão direito do mouse sobre o *StringGrid*.



Exemplo com *StringGrid* - Propriedades I

```
object FormPrincipal: TFormPrincipal
  Caption = 'Produtos'
  object GridProdutos: TStringGrid
    Columns = <
      item
        ButtonStyle = cbsPickList
        PickList.Strings = ('Açúcar' 'Arroz' 'Feijão' 'Leite' 'Pão')
        Title.Caption = 'Produto'
        Width = 250
      end
      item
        Title.Caption = 'Quantidade'
        Width = 90
      end
      item
        Title.Caption = 'V. Unitário'
        Width = 90
      end
      item
        ReadOnly = True
```

Exemplo com *StringGrid* - Propriedades II

```
        Title.Caption = 'V. Total'
        Width = 90
      end>
    FixedCols = 0
  end
  object GroupTotal: TGroupBox
    Caption = 'Total Geral'
    object LabelTotal: TLabel
      Alignment = taRightJustify
      Caption = '0.00'
    end
  end
end
```

Exemplo com *StringGrid* - Código I

```
procedure TFormPrincipal.GridProdutosSetEditText(Sender: TObject; ACol, ARow:
 ↪   Integer;
  const Value: string);
begin
  if ACol in [1,2] then
    Calcula(ARow);
end;

procedure TFormPrincipal.Calcula(linha: Integer);
var
  quant, valor, total: Real;
  cont, erro: Integer;
begin
  val(GridProdutos.Cells[1, linha], quant, erro);
  if (erro = 0 ) then
    val(GridProdutos.Cells[2, linha], valor, erro);

  if (erro <> 0 ) then
    GridProdutos.Cells[3, linha] := ''
  else
```

Exemplo com *StringGrid* - Código II

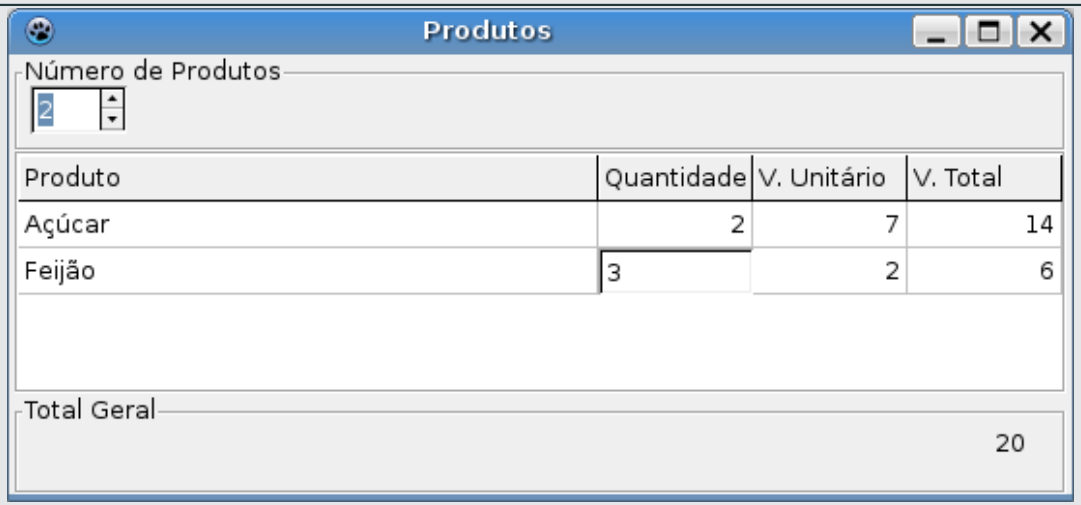
```
  begin
    valor := quant * valor;
    GridProdutos.Cells[3, linha] := FloatToStr(valor);
  end;
  total := 0;
  for cont := 1 to GridProdutos.RowCount -1 do
  begin
    val(GridProdutos.Cells[3, cont], valor, erro);
    if erro = 0 then
      total := total + valor;
  end;
  LabelTotal.Caption := FloatToStr(total);
end;
```

O Componente *SpinEdit*

- O *SpinEdit* é um componente para capturar números inteiros de forma automática (sem a necessidade de conversão). Suas principais propriedades são:
  - **MaxValue:** Valor máximo;
  - **MinValue:** Valor mínimo;
  - **Name:** Nome (abreviação *Spin* ou *Spin*);
  - **Value:** Valor;

Exemplo

```
GridProdutos.RowCount := SpinNumProdutos.Value+1;
SpinNumProdutos.SetFocus();
```



Os Componentes *Dialogs*

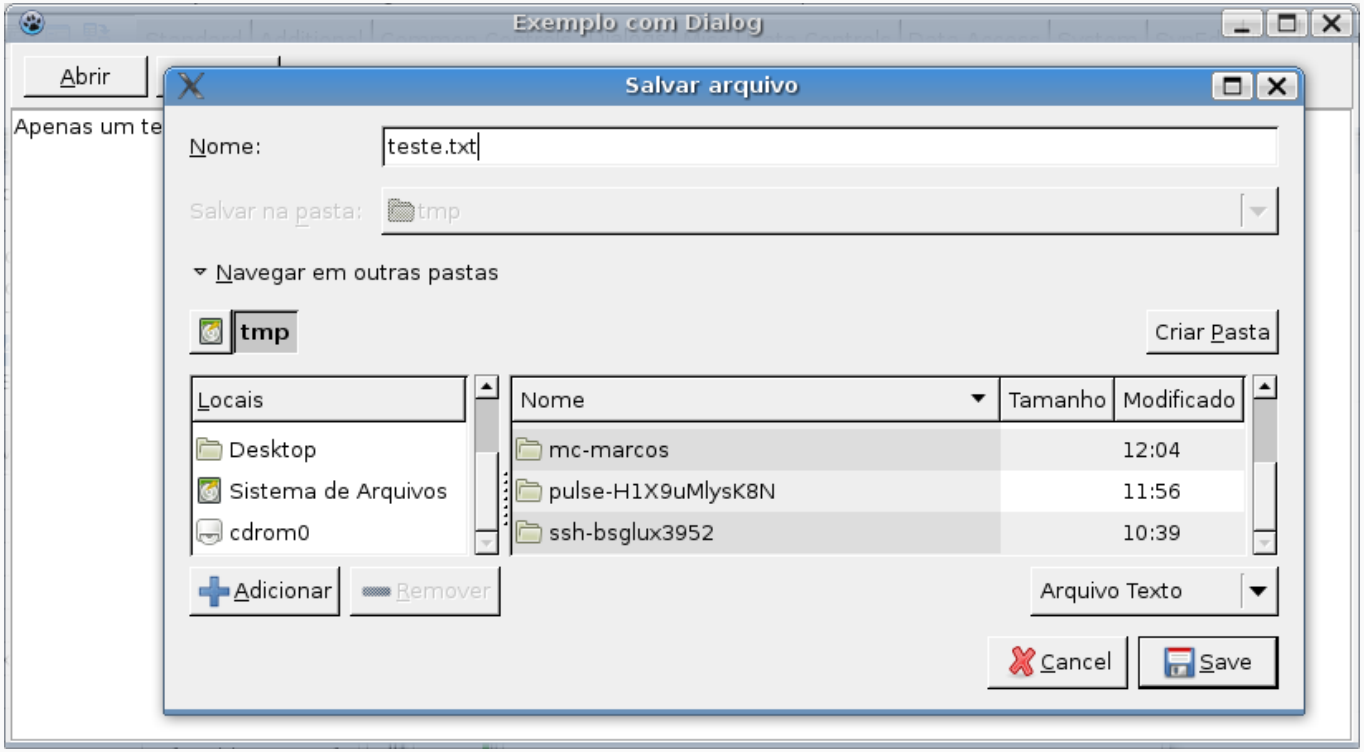
- Os componentes *Dialogs* são caixas de diálogo que podem ser utilizadas em funções específicas, como: abrir arquivo, salvar arquio, selecionar diretório (pasta), entre outras.
- Estas caixas de diálogos são componentes não visuais, ou seja, o componente inserido no formulário não é exibido no formulário;

Os componentes *OpenDialog* e *SaveDialog*

As principais propriedades dos componentes *OpenDialog* e *SaveDialog* são:

FileName: Nome do arquivo;
Filter: Filtro de extensões de arquivos a serem exibidos;
InitialDir: Diretório inicial;
Name: Nome (abreviação *Dialog* ou *Dialog*);
Options: Diversos ajustes finos sobre a caixa de diálogo;
Title: Título;

Exemplo com *Dialogs*

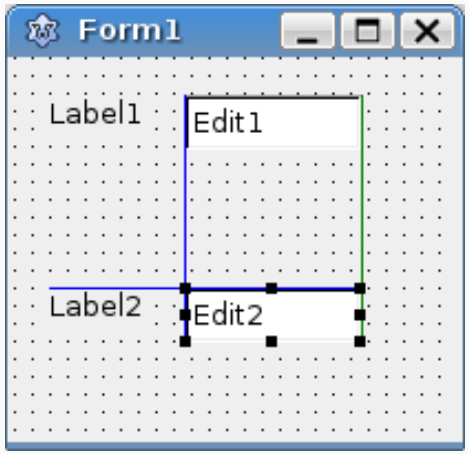


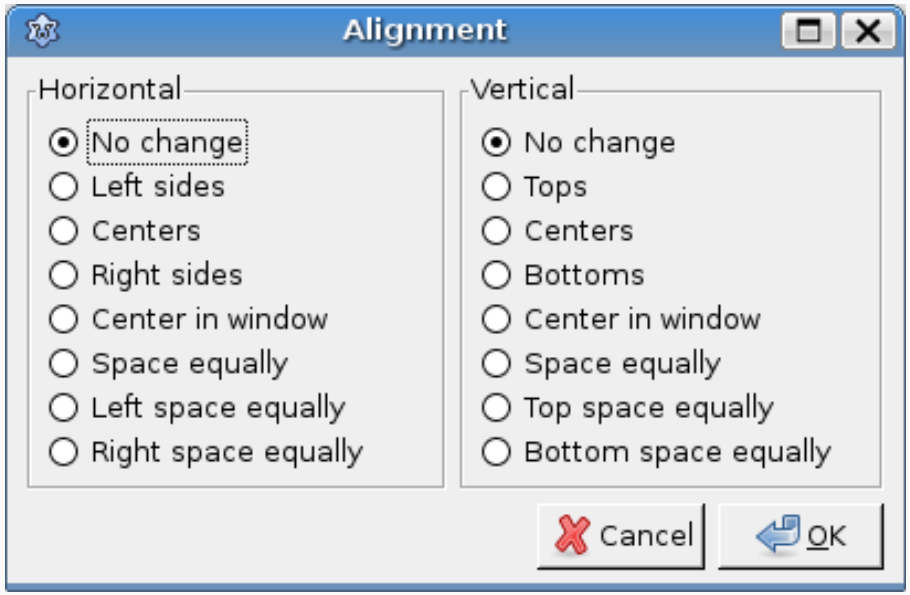

```
procedure TformPrincipal.buttonAbrirClick(Sender: TObject);
begin
  if DialogAbre.Execute then
    memoTexto.Lines.LoadFromFile(DialogAbre.FileName);
end;

procedure TformPrincipal.buttonSalvarClick(Sender: TObject);
begin
  if DialogSalva.Execute then
    memoTexto.Lines.SaveToFile(DialogSalva.FileName);
end;
```

Alinhamento de Componentes

- Em exemplos anteriores já utilizamos o propriedade *FocusControl* e o & na propriedade *Caption* em alguns componentes visando melhorar a usabilidade de nossos programas;
- Outro recurso interessante do Lazarus é o desenho de linhas quando estamos arrastando ou redimensionando componentes em formulários. Este recurso é muito útil para verificarmos se os componentes estão alinhados;
- Além disto, pode ser usada a ferramenta para alinhamento. Basta selecionarmos os componentes a serem alinhados e clicarmos no menu *popup "Align"*.





Alinhamento Relativo

- Além do alinhamento normal de componentes, podemos utilizar o alinhamento relativo;
- Neste tipo de alinhamento alguns componentes assumem posições fixas e outros são redimensionados automaticamente de acordo com o tamanho do formulário;
- Para trabalharmos desta forma devemos modificar a propriedade *Align* de cada componente. Seus possíveis valores são:

alBottom: Posição inferior (altura fixa, largura variável);
alClient: Posição cliente (altura e largura variáveis)
alLeft: Posição esquerda (altura variável, largura fixa);
alNone: Nenhum alinhamento relativo usado;
alRight: Posição direita (altura variável, largura fixa);
alTop: Posição superior (altura fixa, largura variável);

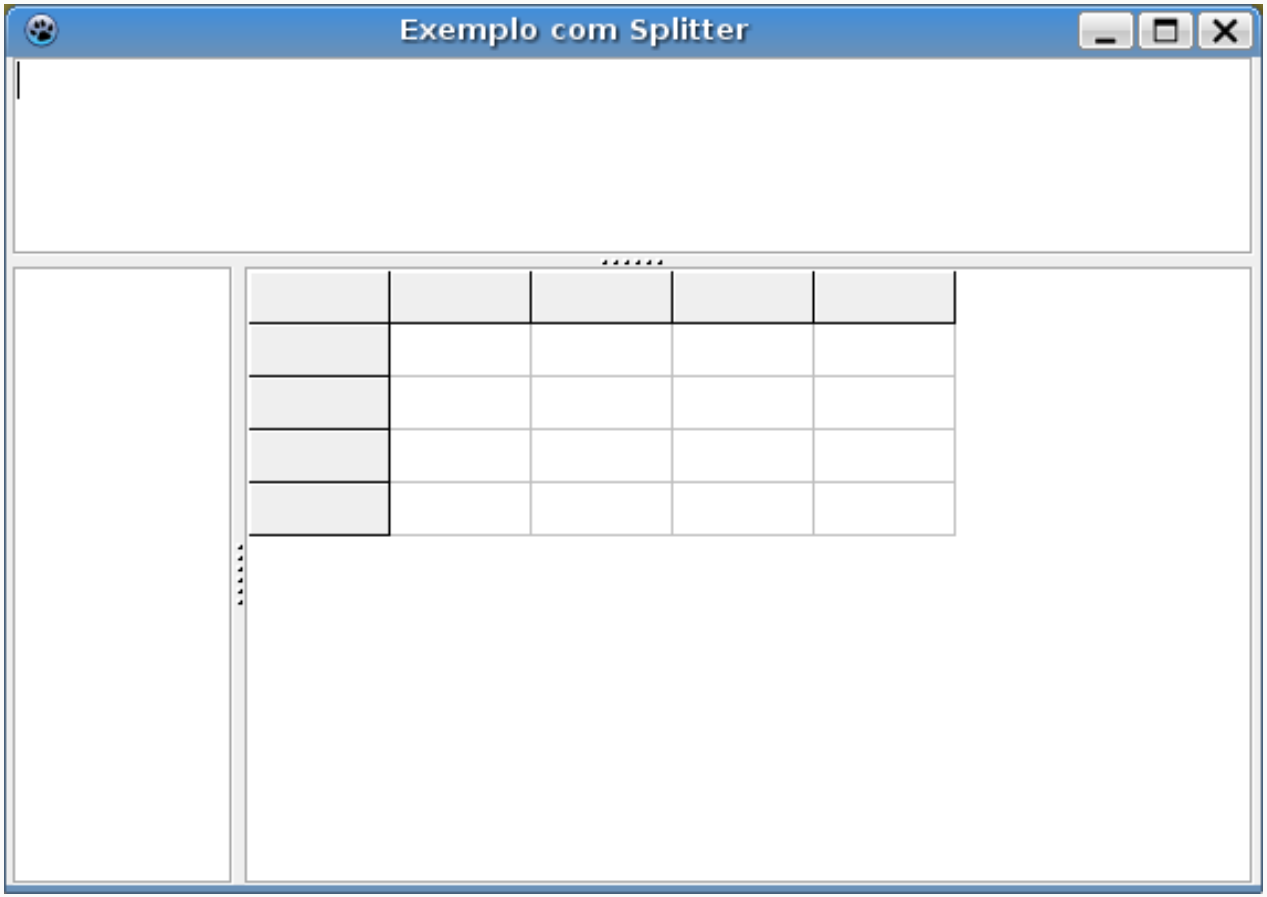
Exemplo com Alinhamento Relativo

Dica

Neste exemplo modificamos o alinhamento relativo apenas dos componentes inseridos no formulário. Todavia, podemos inserir componentes dentro de outros componentes como painéis e caixas de grupo e modificar o alinhamento relativo dentro deste componente.

O Componentes *Splitter*

- O componente *Splitter* é bastante adequando com usamos alinhamento relativo e desejamos que o usuário tenha a possibilidade de modificar o tamanho dos componentes alinhados. Sua principal propriedade é *Align*.



O Componente Timer

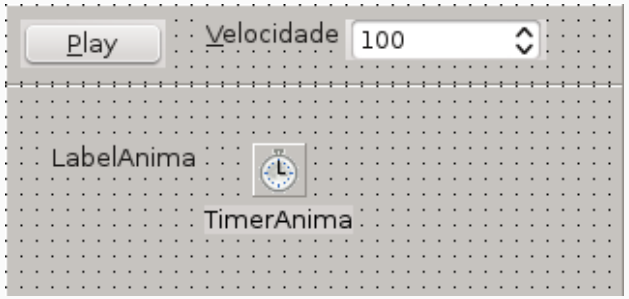
- O componente *Timer* é um componente que permite executar um procedimento em intervalos específicos;
- Sua principal propriedade é o *Interval* que específica a cada quantos milissegundos o procedimento será executado;
- Seus principais eventos são:

  StarTimer: Acontece quando o *Timer* é ativado;

  StopTimer: Acontece quando o *Timer* é desativado;

  Timer: Acontece automaticamente a cada *Interval* milissegundos.

Exemplo com Timer I




```
...
implementation

var
  VelocidadeX: Integer = 1;
  VelocidadeY: Integer = 1;
...
procedure TFormPrincipal.SpinVelocidadeChange(Sender: TObject);
begin
  TimerAnima.Interval := SpinVelocidade.Value;
end;
```

Exemplo com Timer II

```
procedure TFormPrincipal.TimerAnimaTimer(Sender: TObject);
begin
  LabelAnima.Caption := IntToStr(LabelAnima.Left) + ',' +
 ↪   IntToStr(LabelAnima.Top);
  LabelAnima.Left := LabelAnima.Left + VelocidadeX;
  LabelAnima.Top := LabelAnima.Top + VelocidadeY;
  if (LabelAnima.Top + LabelAnima.Height >= PanelAnima.Height) or
  (LabelAnima.Top <= 0) then begin
    VelocidadeY := VelocidadeY * -1;
  end;
  if (LabelAnima.Left + LabelAnima.Width >= PanelAnima.Width) or
  (LabelAnima.Left <= 0) then begin
    VelocidadeX := VelocidadeX * -1;
  end;
end;
```

Exemplo com Timer III

```
procedure TFormPrincipal.ButtonPlayPauseClick(Sender: TObject);
begin
  if (TimerAnima.Enabled) then begin
    LabelAnima.Visible := False;
    ButtonPlayPause.Caption := '&Play';
    TimerAnima.Enabled := False;
  end else begin
    ButtonPlayPause.Caption := '&Pause';
    TimerAnima.Enabled := True;
    LabelAnima.Visible := True;
  end;
end;
```

Referências

- LCL Documentation. References for unit ExtCtrls. Disponível em http://lazarus-ccr.sourceforge.net/docs/lcl/extctrls/index-4.html;
- LCL Documentation. References for unit Grids. Disponível em http://lazarus-ccr.sourceforge.net/docs/lcl/grids/index-4.html;
- LCL Documentation. References for unit Dialogs. Disponível em http://lazarus-ccr.sourceforge.net/docs/lcl/dialogs/index-4.html;
- LCL Documentation. References for unit Spin. Disponível em http://lazarus-ccr.sourceforge.net/docs/lcl/spin/index-4.html.